

# 6th Conference on Advances in Financial Technologies

AFT 2024, September 23–25, 2024, Vienna, Austria

Edited by

Rainer Böhme  
Lucianna Kiffer



### *Editors*

**Rainer Böhme** 

Universität Innsbruck, Austria  
rainer.boehme@uibk.ac.at

**Lucianna Kiffer** 

IMDEA Networks, Madrid, Spain  
lucianna.kiffer@imdea.org

### *ACM Classification 2012*

Security and privacy → Mathematical foundations of cryptography; Theory of computation → Cryptographic primitives; Theory of computation → Cryptographic protocols; Security and privacy → Distributed systems security; Security and privacy → Privacy-preserving protocols; Security and privacy → Pseudonymity, anonymity and untraceability; Security and privacy → Economics of security and privacy; Theory of computation → Algorithmic game theory and mechanism design; Applied computing → Economics; Applied computing → Digital cash

**ISBN 978-3-95977-345-4**

### *Published online and open access by*

Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at <https://www.dagstuhl.de/dagpub/978-3-95977-345-4>.

### *Publication date*

September, 2024

### *Bibliographic information published by the Deutsche Nationalbibliothek*

The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at <https://portal.dnb.de>.

### *License*

This work is licensed under a Creative Commons Attribution 4.0 International license (CC-BY 4.0):

<https://creativecommons.org/licenses/by/4.0/legalcode>.



In brief, this license authorizes each and everybody to share (to copy, distribute and transmit) the work under the following conditions, without impairing or restricting the authors' moral rights:

- Attribution: The work must be attributed to its authors.

The copyright is retained by the corresponding authors.

Digital Object Identifier: 10.4230/LIPIcs.AFT.2024.0

**ISBN 978-3-95977-345-4**

**ISSN 1868-8969**

**<https://www.dagstuhl.de/lipics>**

## LIPICs – Leibniz International Proceedings in Informatics

LIPICs is a series of high-quality conference proceedings across all fields in informatics. LIPICs volumes are published according to the principle of Open Access, i.e., they are available online and free of charge.

### *Editorial Board*

- Luca Aceto (Reykjavik University, IS and Gran Sasso Science Institute, IT)
- Christel Baier (TU Dresden, DE)
- Roberto Di Cosmo (Inria and Université Paris Cité, FR)
- Faith Ellen (University of Toronto, CA)
- Javier Esparza (TU München, DE)
- Daniel Král' (Masaryk University, Brno, CZ)
- Meena Mahajan (*Chair*, Institute of Mathematical Sciences, Chennai, IN)
- Anca Muscholl (University of Bordeaux, FR)
- Chih-Hao Luke Ong (Nanyang Technological University, SG)
- Phillip Rogaway (University of California, Davis, US)
- Eva Rotenberg (Technical University of Denmark, Lyngby, DK)
- Raimund Seidel (Universität des Saarlandes, Saarbrücken, DE and Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Wadern, DE)
- Pierre Senellart (ENS, Université PSL, Paris, FR)

**ISSN 1868-8969**

**<https://www.dagstuhl.de/lipics>**



In memory of Ross J. Anderson.



## ■ Contents

Preface	
<i>Rainer Böhme and Lucianna Kiffer</i> .....	0:xi
Program Committee	
.....	0:xiii
Steering Committee	
.....	0:xv
External Reviewers	
.....	0:xvii
List of Authors	
.....	0:xix

### Session 1: Consensus

Accountable Secret Leader Election	
<i>Miranda Christ, Kevin Choi, Walter McKelvie, Joseph Bonneau, and Tal Malkin</i> .....	1:1–1:21
BoLD: Fast and Cheap Dispute Resolution	
<i>Mario M. Alvarez, Henry Arneson, Ben Berger, Lee Bousfield, Chris Buckland, Yafah Edelman, Edward W. Felten, Daniel Goldman, Raul Jordan, Mahimna Kelkar, Akaki Mamageishvili, Harry Ng, Aman Sanghi, Victor Shoup, and Terence Tsao</i> .....	2:1–2:19
CFT-Forensics: High-Performance Byzantine Accountability for Crash Fault Tolerant Protocols	
<i>Weizhao Tang, Peiyao Sheng, Ronghao Ni, Pronoy Roy, Xuechao Wang, Giulia Fanti, and Pramod Viswanath</i> .....	3:1–3:25

### Session 2: Auditing

Cross Ledger Transaction Consistency for Financial Auditing	
<i>Vlasis Koutsos, Xiangan Tian, Dimitrios Papadopoulos, and Dimitris Chatzopoulos</i> .....	4:1–4:25
Proof of Diligence: Cryptoeconomic Security for Rollups	
<i>Peiyao Sheng, Ranvir Rana, Senthil Bala, Himanshu Tyagi, and Pramod Viswanath</i> .....	5:1–5:24
Analyzing and Benchmarking ZK-Rollups	
<i>Stefanos Chaliasos, Itamar Reif, Adrià Torralba-Agell, Jens Ernstberger, Assimakis Kattis, and Benjamin Livshits</i> .....	6:1–6:24
DeFiAligner: Leveraging Symbolic Analysis and Large Language Models for Inconsistency Detection in Decentralized Finance	
<i>Rundong Gan, Liyi Zhou, Le Wang, Kaihua Qin, and Xiaodong Lin</i> .....	7:1–7:24

6th Conference on Advances in Financial Technologies (AFT 2024).

Editors: Rainer Böhme and Lucianna Kiffer



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

**Session 3: Blockchain Foundations**

A Circuit Approach to Constructing Blockchains on Blockchains <i>Ertem Nusret Tas, David Tse, and Yifei Wang</i> .....	8:1–8:25
Blockchain Space Tokenization <i>Aggelos Kiayias, Elias Koutsoupias, Philip Lazos, and Giorgos Panagiotakos</i> .....	9:1–9:20
Optimal RANDAO Manipulation in Ethereum <i>Kaya Alpturer and S. Matthew Weinberg</i> .....	10:1–10:21

**Session 4: Payment Channels**

<i>Bribe &amp; Fork: Cheap PCN Bribing Attacks via Forking Threat</i> <i>Zeta Avarikioti, Paweł Kędzior, Tomasz Lizurej, and Tomasz Michalak</i> .....	11:1–11:22
Payment Censorship in the Lightning Network Despite Encrypted Communication <i>Charmaine Ndolo and Florian Tschorsch</i> .....	12:1–12:24
Musketeer: Incentive-Compatible Rebalancing for Payment Channel Networks <i>Zeta Avarikioti, Stefan Schmid, and Samarth Tiwari</i> .....	13:1–13:22

**Session 5: Cryptographic Building Blocks**

SoK: Zero-Knowledge Range Proofs <i>Miranda Christ, Foteini Baldimtsi, Konstantinos Kryptos Chalkias, Deepak Maram, Arnab Roy, and Joy Wang</i> .....	14:1–14:25
Privacy Comparison for Bitcoin Light Client Implementations <i>Arad Kotzer and Ori Rottenstreich</i> .....	15:1–15:23
CrudiTEE: A Stick-And-Carrot Approach to Building Trustworthy Cryptocurrency Wallets with TEEs <i>Lulu Zhou, Zeyu Liu, Fan Zhang, and Michael K. Reiter</i> .....	16:1–16:25
Cornucopia: Distributed Randomness at Scale <i>Miranda Christ, Kevin Choi, and Joseph Bonneau</i> .....	17:1–17:23

**Session 6: Mechanisms**

Loss-Versus-Fair: Efficiency of Dutch Auctions on Blockchains <i>Ciamac C. Moallemi and Dan Robinson</i> .....	18:1–18:17
Credible, Optimal Auctions via Public Broadcast <i>Tarun Chitra, Matheus V. X. Ferreira, and Kshitij Kulkarni</i> .....	19:1–19:16
Optimizing Exit Queues for Proof-Of-Stake Blockchains: A Mechanism Design Approach <i>Michael Neuder, Malleesh Pai, and Max Resnick</i> .....	20:1–20:22
Searcher Competition in Block Building <i>Akaki Mamagishvili, Christoph Schlegel, and Benny Sudakov</i> .....	21:1–21:12



**Session 7: Measurements**

Who Wins Ethereum Block Building Auctions and Why? <i>Burak Öz, Danning Sui, Thomas Thiery, and Florian Matthes</i> .....	22:1–22:25
A Shortfall in Investor Expectations of Leveraged Tokens <i>Reza Rahimian and Jeremy Clark</i> .....	23:1–23:24
Investigating Wrench Attacks: Physical Attacks Targeting Cryptocurrency Users <i>Marilyne Ordekian, Gilberto Atondo-Siu, Alice Hutchings, and Marie Vasek</i> .....	24:1–24:24

**Session 8: Finance**

Adaptive Curves for Optimally Efficient Market Making <i>Viraj Nadkarni, Sanjeev Kulkarni, and Pramod Viswanath</i> .....	25:1–25:22
Competitive Policies for Online Collateral Maintenance <i>Ghada Almashaqbeh, Sixia Chen, and Alexander Russell</i> .....	26:1–26:16
Thinking Fast and Slow: Data-Driven Adaptive DeFi Borrow-Lending Protocol <i>Mahsa Bastankhah, Viraj Nadkarni, Chi Jin, Sanjeev Kulkarni, and Pramod Viswanath</i> .....	27:1–27:23

**Session 9: Securing Decentralized Systems**

SoK: Attacks on DAOs <i>Rainer Feichtinger, Robin Fritsch, Lioba Heimbach, Yann Vonlanthen, and Roger Wattenhofer</i> .....	28:1–28:27
Transaction Fee Mechanism Design in a Post-MEV World <i>Maryam Bahrani, Pranav Garimidi, and Tim Roughgarden</i> .....	29:1–29:24
Profitable Manipulations of Cryptographic Self-Selection Are Statistically Detectable <i>Linda Cai, Jingyi Liu, S. Matthew Weinberg, and Chenghan Zhou</i> .....	30:1–30:23



## ■ Preface

This volume contains 30 papers selected from 106 submissions to the Conference on Advances in Financial Technologies (AFT '24) held at the Oesterreichische Nationalbank (OeNB) in Vienna, Austria, on 23–25 September 2024. This is the 6th year of the conference and the second year that it is independently organised and published in LIPIcs. The host institutions in Austria were the Complexity Science Hub in Vienna and the University of Innsbruck.

For the first time, this conference was co-located with the Economics of Payments XIII conference, the primary conference for economists, central bankers, and policy researchers to present and discuss their studies on topics related to payment, clearing and settlement systems. The purpose of the co-location was to foster collaboration across disciplinary boundaries and professional communities. Both conferences had an overlap day open to participants from both communities, with keynotes by Neha Narula of the Digital Currency Initiative at MIT and Charles M. Kahn of the University of Illinois and the Bank of Canada, followed by a panel discussion, and culminating in an evening reception and poster session.

AFT '24 also had an associated workshop on Scalability & Interoperability of Blockchains (SIB), co-organized by Zeta Avarikioti and Dionysis Zindros.

The paper selection process followed the conventions in computer science. Each submission received at least three detailed double-blind reviews by several program committee members and external reviewers. Each accepted paper was presented via a 15-minute live presentation, followed by a 5-minute question/answer period with the audience.

We would like to thank all Program Committee members and external reviewers for their service in selecting the AFT program, and all authors for submitting their work for consideration. We are also grateful to the AFT steering committee for their support and guidance throughout the process.

We would like to acknowledge the industry sponsors whose financial support is essential to running of AFT:

### Gold level

- a16z crypto
- EigenLabs
- Kaspas

### Silver level

- Ava Labs
- Gnosis Pay
- IC3
- StarkWare

We are deeply grateful to Bernhard Haslhofer, who served as General Chair. Without his initiative and sustained commitment, we would not have brought AFT '24 to Vienna, let alone held it at a central bank and in conjunction with an economics conference. We would further like to thank all the staff at the Complexity Science Hub in Vienna who made this event possible, especially Hannah Scholl, Anja Böck, Sonja Jöchtl, and Svetlana Abramova. Finally, we also would like to thank Helmut Stix, Helmut Elsinger, and Martin Summer at the OeNB for their support in co-hosting this year's AFT edition with Economics of Payments XIII.

Innsbruck, Austria  
Madrid, Spain  
September 2024

*Rainer Böhme*  
*Lucianna Kiffer*





## ■ Program Committee

Aggelos Kiayias, University of Edinburgh and IOG	Hong-Sheng Zhou, Virginia Commonwealth University
Akaki Mamageishvili, Offchain Labs	Ittay Eyal, Technion
Alberto Sonnino, MystenLabs and University College London	Jason Milionis, Columbia University
Alexander Spiegelman, Aptos Labs	Jeremy Clark, Concordia University
Aljoshia Judmayer, University of Vienna	Jiasun Li, George Mason University
Andrew Lewis-Pye, London School of Economics	Jing Chen, Tsinghua University
Andrew Miller, University of Illinois Urbana-Champaign	Johnatan Messias, Matter Labs
Arthur Gervais, University College London	Julien Prat, IP Paris
Aviad Rubinstein, Stanford University	Kaihua Qin, Imperial College London
Aviv Zohar, The Hebrew University	Kanye Ye Wang, University of Macao
Aviv Yaish, The Hebrew University	Karl Wüst, Mysten Labs
Barnabe Monnot, Ethereum Foundation	Krzysztof Pietrzak, ISTA
Bernhard Haslhofer, Complexity Science Hub	Lin William (Will) Cong, Cornell University
Christian Cachin, University of Bern	Ling Ren, University of Illinois Urbana-Champaign
Ciamac Moallemi, Columbia University	Lioba Heimbach, ETH Zürich
Clara Shikhelman, Chaincode Labs	Marco Reuter, International Monetary Fund
Ethan Heilman, Massachusetts Institute of Technology DCI	Marie Vasek, University College London
Fahad Saleh, University of Florida	Marko Vukolić, ConsensusLab
Fan Zhang, Yale University	Maryam Bahrani, a16z crypto
Florian Tschorsch, TU Dresden	Matheus Venturyne Xavier Ferreira, University of Virginia
Francisco Marmolejo-Cossio, Harvard University	Neha Narula, Massachusetts Institute of Technology
Geoffrey Ramseyer, Stanford University	Patrick McCorry, Arbitrum Foundation
George Danezis, MystenLabs and University College London	Pedro Moreno-Sanchez, IMDEA Software Institute
Georgios Piliouras, Google DeepMind   SUTD	Philipp Jovanovic, University College London
Ghassan Karame, Ruhr-University Bochum	Pierre-Louis Roman, Independent
Guy Goren, Protocol Labs	Pietro Saggese, IMT School for Advanced Studies Lucca

6th Conference on Advances in Financial Technologies (AFT 2024).

Editors: Rainer Böhme and Lucianna Kiffer



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

**0:xiv    Program Committee**

Prateesh Goyal, Microsoft Research

Qiang Tang, The University of Sydney

Rafael Pass, Cornell University

Roger Wattenhofer, ETH Zürich

Sara Tucci-Piergiovanni, CEA-List and  
Paris-Saclay University

Sarah Azouvi, Independent

Tarun Chitra, Gauntlet

Theo Diamandis, Massachusetts Institute of  
Technology

Tim Roughgarden, Columbia University and  
a16z crypto

Valeria Nikolaenko, a16z crypto

Victor Luchangco, Catholic Institute of  
Technology

Yonatan Sompolinsky, Harvard University

Zeta Avarikioti, TU Vienna and Common  
Prefix

## ■ Steering Committee

Ittai Abraham (co-chair), VMware Research

Dan Boneh, Stanford University

Christian Cachin, University of Bern

Ittay Eyal (co-chair), Technion

Maurice Herlihy, Brown University

Satoshi Nakamoto (pending confirmation)

Maureen O'Hara, Cornell University

Tim Roughgarden, Columbia University

Eli Ben Sasson, Technion

Emin Gun Sirer (co-chair), Cornell University

Neha Narula ('22 PC chair), Massachusetts Institute of Technology

S. Matthew Weinberg ('23 PC chair), Princeton University

Joseph Bonneau ('23 PC chair), New York University

Rainer Böhme ('24 PC chair), University of Innsbruck

Lucianna Kiffer ('24 PC chair), IMDEA Networks





## ■ External Reviewers

Aditya Saraf

Alejandro Ranchal-Pedrosa

Alireza Kavousi

Álvaro García-Pérez

Amirreza Sarencheh

Benjamin Chan

Charmaine Ndolo

Chengru Zhang

Christina Ovezik

Christos Stefos

Damien Berriaud

Daniël Reijsbergen

Diana Ghinea

Erik Daniel

Faxing Wang

George Bissias

Georgios Chionas

Gilad Stern

Giorgos Panagiotakos

Giulia Scaffino

Harjasleen Malvai

Ilan Tennenhouse

Iosif Sakos

István András Seres

Jacob Leshno

Jayamine Alupotha

Jichen Li

Jorge Soares

Judith Senn

Julian Ma

Kat Molinet

Lei Yang

Lukas Aumayr

Maofan "Ted" Yin

Marc Roeschlin

Marilyne Ordekian

Maxim Jourenko

Michele Fabi

Michelle Yeo

Nicholas Stifter

Orfeas Stefanos Thyfronitis Litos

Pyrros Chaidos

Quentin Knip

Rainer Stütz

Ray Neiheiser

Rex Fernando

Ryann Sim

Sen Yang

Svetlana Abramova

Xiaohai Dai

Xinyu Li

Yann Vonlanthen

Yuheng Wang

Yunqi Li

Zhenliang Lu

Zhixuan Fang

Zhuolun Xiang

Ziling Yang

6th Conference on Advances in Financial Technologies (AFT 2024).

Editors: Rainer Böhme and Lucianna Kiffer



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany




## ■ List of Authors

Ghada Almashaqbeh (26)  
University of Connecticut, Storrs, CT, USA

Kaya Alpturer  (10)  
Princeton University, NJ, USA

Mario M. Alvarez (2)  
Offchain Labs, Inc., Clifton, NJ, USA

Henry Arneson (2)  
Offchain Labs, Inc., Clifton, NJ, USA


Gilberto Atondo-Siu  (24)  
Department of Computer Science,  
University of Cambridge, UK


Zeta Avarikioti (11, 13)  
Department of Informatics, TU Wien, Austria;  
Common Prefix, Vienna, Austria

Maryam Bahrani (29)  
Ritual, New York, NY, USA

Senthil Bala (5)  
Witness Chain, Bengaluru, India

Foteini Baldimtsi  (14)  
Mysten Labs, Palo Alto, CA, USA;  
George Mason University, Fairfax, VA, USA

Mahsa Bastankhah  (27)  
Princeton University, NJ, USA

Ben Berger  (2)  
Offchain Labs, Inc., Clifton, NJ, USA

Joseph Bonneau (1, 17)  
New York University, NY, USA;  
a16z crypto research, New York, NY, USA

Lee Bousfield (2)  
Offchain Labs, Inc., Clifton, NJ, USA

Chris Buckland (2)  
Offchain Labs, Inc., Clifton, NJ, USA


Linda Cai  (30)  
Princeton University, NJ, USA

Stefanos Chaliasos (6)  
Imperial College London, UK

Konstantinos Kryptos Chalkias  (14)  
Mysten Labs, Palo Alto, CA, USA


Dimitris Chatzopoulos  (4)  
University College Dublin, Ireland

Sixia Chen (26)  
Adelphi University, Garden City, NY, USA

Tarun Chitra  (19)  
Gauntlet, New York, NY, USA

Kevin Choi (1, 17)  
New York University, NY, USA


Miranda Christ (1, 14, 17)  
Columbia University, New York, NY, USA

Jeremy Clark  (23)  
Concordia, University, Montreal, Canada


Yafah Edelman (2)  
Offchain Labs, Inc., Clifton, NJ, USA


Jens Ernstberger (6)  
Technische Universität München, Germany

Giulia Fanti  (3)  
Carnegie Mellon University,  
Pittsburgh, PA, USA

Rainer Feichtinger  (28)  
ETH Zürich, Switzerland

Edward W. Felten (2)  
Offchain Labs, Inc., Clifton, NJ, USA


Matheus V. X. Ferreira  (19)  
University of Virginia, Charlottesville, CA, USA

Robin Fritsch  (28)  
ETH Zürich, Switzerland


Rundong Gan (7)  
School of Computer Science,  
University of Guelph, Canada

Pranav Garimidi (29)  
a16z Crypto, New York, NY, USA

Daniel Goldman (2)  
Offchain Labs, Inc., Clifton, NJ, USA

Lioba Heimbach  (28)  
ETH Zürich, Switzerland

Alice Hutchings  (24)  
Department of Computer Science,  
University of Cambridge, UK


Chi Jin  (27)  
Princeton University, NJ, USA


Raul Jordan (2)  
Offchain Labs, Inc., Clifton, NJ, USA


Assimakis Kattis (6)  
Athens, Greece


Mahimna Kelkar  (2)  
Offchain Labs, Inc., Clifton, NJ, USA

Aggelos Kiayias  (9)  
University of Edinburgh, UK;  
IOG, London, UK

Arad Kotzer  (15)  
The Department of Computer Science,  
Technion, Haifa, Israel

Vlasis Koutsos  (4)  
Hong Kong University of Science and  
Technology, Hong Kong

Elias Koutsoupas  (9)  
University of Oxford, UK

Kshitij Kulkarni  (19)  
UC Berkeley, CA, USA


Sanjeev Kulkarni  (25, 27)  
Princeton University, NJ, USA

Paweł Kędzior (11)  
University of Warsaw, Poland

Philip Lazos  (9)  
Jump Trading, London, UK

Xiaodong Lin  (7)  
School of Computer Science,  
University of Guelph, Canada


Jingyi Liu  (30)  
Princeton University, NJ, USA

Zeyu Liu  (16)  
Yale University, New Haven, CT, USA

Benjamin Livshits (6)  
Imperial College London, UK;  
Matter Labs, London, UK

Tomasz Lazurek (11)  
NASK, Warsaw, Poland;  
University of Warsaw, Poland

Tal Malkin (1)  
Columbia University, New York, NY, USA


Akaki Mamageishvili  (2, 21)  
Offchain Labs, Inc., Clifton, NJ, USA;  
Offchain Labs, Zurich, Switzerland

Deepak Maram  (14)  
Mysten Labs, Palo Alto, CA, USA


Florian Matthes  (22)  
Technical University of Munich,  
Garching, Germany

Walter McKelvie (1)  
Columbia University, New York, NY, USA

Tomasz Michalak (11)  
IDEAS NCBR, Warsaw, Poland;  
University of Warsaw, Poland


Ciamac C. Moallemi  (18)  
Graduate School of Business, Columbia  
University, New York, NY, USA


Viraj Nadkarni  (25, 27)  
Princeton University, NJ, USA


Charmaine Ndolo  (12)  
Dresden University of Technology, Germany


Michael Neuder (20)  
Ethereum Foundation, New York, NY, USA


Harry Ng (2)  
Offchain Labs, Inc., Clifton, NJ, USA

Ronghao Ni  (3)  
Carnegie Mellon University,  
Pittsburgh, PA, USA


Marilyne Ordekian  (24)  
Department of Computer Science,  
University College London, UK

Mallesh Pai  (20)  
Special Mechanisms Group, Consensys Inc,  
Dallas, TX, USA;  
Rice University, Houston, TX, USA

Giorgos Panagiotakos  (9)  
IOG, Athens, Greece


Dimitrios Papadopoulos  (4)  
Hong Kong University of Science and  
Technology, Hong Kong

Kaihua Qin (7)  
Yale University, New Haven, CT, USA;  
UC Berkeley RDI, CA, USA;  
Decentralized Intelligence AG, Zug, Switzerland

Reza Rahimian  (23)  
Concordia University, Montreal, Canada

Ranvir Rana  (5)  
Witness Chain, NJ, USA


Itamar Reif (6)  
Austria, New York, NY, USA

Michael K. Reiter  (16)  
Duke University, New Haven, CT, USA

Max Resnick (20)  
Special Mechanisms Group, Consensys Inc,  
Dallas, TX, USA

- Dan Robinson (18)  
Paradigm, San Francisco, CA, USA
- Ori Rottenstreich  (15)  
The Department of Computer Science and the  
Department of Electrical and Computer  
Engineering, Technion, Haifa, Israel
- Tim Roughgarden (29)  
a16z Crypto, New York, NY, USA;  
Columbia University, New York, NY, USA
- Arnab Roy  (14)  
Mysten Labs, Palo Alto, CA, USA
- Pronoy Roy (3)  
Carnegie Mellon University,  
Pittsburgh, PA, USA
- Alexander Russell (26)  
University of Connecticut, Storrs, CT, USA;  
IOG, Singapore
- Aman Sanghi (2)  
Offchain Labs, Inc., Clifton, NJ, USA
- Christoph Schlegel (21)  
Flashbots, George Town, Cayman Islands
- Stefan Schmid  (13)  
TU Berlin, Germany;  
Fraunhofer SIT, Berlin, Germany;  
Weizenbaum Institute, Berlin, Germany
- Peiyao Sheng  (3, 5)  
University of Illinois Urbana-Champaign, IL,  
USA;  
Witness Chain, NJ, USA
- Victor Shoup  (2)  
Offchain Labs, Inc., Clifton, NJ, USA
- Benny Sudakov  (21)  
ETH Zürich, Switzerland
- Danning Sui  (22)  
Flashbots, San Francisco, CA, USA
- Weizhao Tang  (3)  
Carnegie Mellon University,  
Pittsburgh, PA, USA
- Ertem Nusret Tas  (8)  
Stanford University, CA, USA
- Thomas Thiery  (22)  
Ethereum Foundation, Lisbon, Portugal
- Xiangnan Tian  (4)  
Hong Kong University of Science and  
Technology, Hong Kong
- Samarth Tiwari  (13)  
Centrum Wiskunde & Informatica,  
Amsterdam, The Netherlands
- Adrià Torralba-Agell (6)  
Universitat Oberta de Catalunya,  
San Martí, Spain
- Terence Tsao (2)  
Offchain Labs, Inc., Clifton, NJ, USA
- Florian Tschorsch  (12)  
Dresden University of Technology, Germany
- David Tse  (8)  
Stanford University, CA, USA
- Himanshu Tyagi (5)  
Witness Chain, Bengaluru, India
- Marie Vasek  (24)  
Department of Computer Science,  
University College London, UK
- Pramod Viswanath  (3, 5, 25, 27)  
Princeton University, NJ, USA;  
Witness Chain, NJ, USA
- Yann Vonlanthen  (28)  
ETH Zürich, Switzerland
- Joy Wang  (14)  
Mysten Labs, Palo Alto, CA, USA
- Le Wang  (7)  
School of Computer Science,  
University of Guelph, Canada
- Xuechao Wang  (3)  
Hong Kong University of Science and  
Technology, Guangzhou, China
- Yifei Wang  (8)  
Stanford University, CA, USA
- Roger Wattenhofer  (28)  
ETH Zürich, Switzerland
- S. Matthew Weinberg  (10, 30)  
Princeton University, NJ, USA
- Fan Zhang  (16)  
Yale University, New Haven, CT, USA
- Chenghan Zhou  (30)  
Stanford University, Palo Alto, CA, USA
- Liyi Zhou (7)  
The University of Sydney, Australia;  
UC Berkeley RDI, CA, USA;  
Decentralized Intelligence AG, Zug, Switzerland

**0:xxii Authors**

Lulu Zhou  (16)  
Yale University, New Haven, CT, USA

Burak Öz  (22)  
Technical University of Munich,  
Garching, Germany

# Accountable Secret Leader Election

**Miranda Christ** ✉

Columbia University, New York, NY, USA

**Kevin Choi** ✉

New York University, NY, USA

**Walter McKelvie**

Columbia University, New York, NY, USA

**Joseph Bonneau** ✉

New York University, NY, USA

a16z crypto research, New York, NY, USA

**Tal Malkin** ✉

Columbia University, New York, NY, USA

---

## Abstract

We consider the problem of secret leader election with *accountability*. Secret leader election protocols counter adaptive adversaries by keeping the identities of elected leaders secret until they choose to reveal themselves, but in existing protocols this means it is impossible to determine who was elected leader if they fail to act. This opens the door to undetectable withholding attacks, where leaders fail to act in order to slow the protocol or bias future elections in their favor. We formally define accountability (in weak and strong variants) for secret leader election protocols. We present three paradigms for adding accountability, using delay-based cryptography, enforced key revelation, or threshold committees, all of which ensure that after some time delay the result of the election becomes public. The paradigm can be chosen to balance trust assumptions, protocol efficiency, and the length of the delay before leaders are revealed. Along the way, we introduce several new cryptographic tools including re-randomizable timed commitments and timed VRFs.

**2012 ACM Subject Classification** Security and privacy → Cryptography

**Keywords and phrases** Consensus Protocols, Single Secret Leader Election, Accountability

**Digital Object Identifier** 10.4230/LIPIcs.AFT.2024.1

**Funding** Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the United States Government, DARPA, Andreessen Horowitz, the Algorand Foundation, Google, the National Science Foundation, or any other supporting organization.

*Miranda Christ*: Supported by NSF grants CCF-2107187, CCF-2212233, and CCF-2312242, by LexisNexis Risk Solutions, by the Algorand Centres of Excellence programme managed by Algorand Foundation, and by a Google CyberNYC Award.

*Kevin Choi*: Supported by DARPA Agreement HR00112020022 and NSF grant CNS-2239975.

*Joseph Bonneau*: Supported by DARPA Agreement HR00112020022, NSF grant CNS-2239975, and a16z crypto research.

*Tal Malkin*: Supported by NSF grant CCF-2312242, by the Algorand Centres of Excellence programme managed by Algorand Foundation, and by a Google CyberNYC Award.

## 1 Introduction

In proof-of-stake (PoS) blockchains, an essential challenge is randomly choosing a participant as the leader. The role of leaders varies by protocol, but they may perform tasks like compiling transactions into a block to propose to the network or voting to confirm proposed blocks. A desirable property of leader election is *secrecy*: nobody knows who the leader is



© Miranda Christ, Kevin Choi, Walter McKelvie, Joseph Bonneau, and Tal Malkin; licensed under Creative Commons License CC-BY 4.0

6th Conference on Advances in Financial Technologies (AFT 2024).

Editors: Rainer Böhme and Lucianna Kiffer; Article No. 1; pp. 1:1–1:21

Leibniz International Proceedings in Informatics



Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

until they have revealed themselves in the course of fulfilling their duty. Secrecy is important in mitigating adaptive attacks, where an adversary may attempt to selectively corrupt (or launch a denial-of-service attack on) a leader before they can fulfill their duty. Adaptive attacks can be prevented in the so-called *you only speak once* (YOSO) [21] or *erasures* model, in which elected leaders delete their signing key prior to broadcasting their election. This ensures that when leaders become publicly known, it is already too late to corrupt them.

The first secret leader election (SLE) protocols (e.g., Algorand [23]) are *probabilistic*. Each user has an equal and independent chance of being assigned as leader, meaning there is inherently some probability of multiple leaders (or none) being elected. This undermines determinacy and adds overhead, motivating Protocol Labs [30] to propose *single* secret leader election (SSLE). Boneh et al. [8] were the first to formally define and construct SSLE protocols, which ensure that only a single leader (or another precise number) is elected. They proposed several constructions which have found their way to practice. Their DDH-based scheme was later adapted into Whisk, a practical SSLE protocol designed and proposed for use in Ethereum in EIP-7441 [22]. Other SSLE constructions have also been proposed with varying efficiency-security tradeoffs, including protocols with stronger security notions such as post-quantum security [10], adaptive security [12], UC security [13]; and a higher-communication protocol that better accommodates non-uniform stake distributions [4]. As secret leader election is necessary to attain fairness in the presence of adaptive corruptions, it has been studied in recent blockchain constructions like Ouroboros Cryptsinous (implicitly, in the UC framework) and Fantôme (under the name of “delayed unpredictability”) [26, 2]. An even stronger notion, wherein a leader’s identity is kept private even *after* they publish a block, was proposed in [20].

**Withholding attacks.** Unfortunately, all known secret leader election schemes have the property that if the leader fails to perform its duty and announce itself, the other parties have no way of learning who the absentee leader was. This may not seem problematic, as leaders are typically rewarded for publishing blocks so there is an opportunity cost to failing to claim leadership. However, the choice to claim leadership or withhold creates an opportunity for elected leaders to introduce bias into the randomness used to elect future leaders. Wahrstätter [32] showed that a similar attack is indeed profitable today for nodes in Ethereum’s RANDAO beacon chain, a core component of its consensus protocol.<sup>1</sup> Withholding attacks on RANDAO may be even more profitable if combined with manipulating randomness used by application-layer protocols which rely on it as a randomness beacon. This attack is not unique to Ethereum – in fact, Ferreira et al. [19] showed that a version of it inherently exists for *all* “cryptographic self-selection protocols” in which leaders are randomly chosen based on past values of the blockchain alone.

These attacks are troubling first because they incentivize leader absenteeism, which slows down the protocol (undermining liveness). Even worse, they threaten fairness, as validators with greater stake can gain a greater advantage through this strategy. Intuitively, this is because the chance of being elected once in an epoch is linear while the chance of being elected twice is quadratic in the stake. In the limit, these attacks create a rich-get-richer effect, motivating large coalitions and undermining decentralization.

---

<sup>1</sup> In this attack, validators withhold from contributing randomness to the beacon chain, foregoing some contribution reward to improve future election chances. Ethereum does not currently employ secret leader election, so the attack is detectable; however, there are proposals to do so.



■ **Table 1** Summary of our constructions.

Construction	Section	P/S	Approach
SSLE + RRTC	4.1	Single	Delay-based. Users generate their key material as time-lock puzzle outputs and publish the corresponding inputs.
Timed weak VRFs	4.2	Probabilistic	Delay-based. Uses a new primitive called a timed VRF, where anyone (even without the secret key) can evaluate the VRF using a slow function.
Financial punishment	5.1	Either	Incentive-based.
Indexed VRFs	5.2	Probabilistic	Incentive-based. From hash functions and requires linear precomputation.
Indexed VRFs	5.2	Probabilistic	Incentive-based. From groups of unknown order. Does not require linear precomputation and allows all users to work in the same group.
Indexed VRFs	5.2	Probabilistic	Incentive-based. From trapdoor permutations. Less precomputation but each user must maintain its own trapdoor.
SSLE + ThrPKE	6	Single	Committee-based. Uses threshold cryptography to encrypt key material to a committee, which can reconstruct this key material if it is later withheld by a leader.

**Our contributions.** To address the problem of withholding attacks, we propose *accountable* secret leader election, in which the other validators eventually learn the identity of a negligent leader. Of course, it is critical that they do not learn the leader’s identity too early, to maintain secrecy. We define both *strong* accountability (in which a withholding leader is identified precisely) and *weak* accountability, in which all participants who deviate from the protocol are identified (although we may not know which of them was the elected leader).

We then propose concrete constructions for both single and probabilistic SLE, summarized in Table 1. Our constructions fit into three distinct approaches:

1. **Timed Accountability** (delay-based). Any party can evaluate a delay function (e.g., verifiable delay function) to learn the leader’s identity after some delay. The time delay of the slow function ensures that no party can learn a leader’s identity until after its slot to perform its duty has passed, even with a dishonest majority.
2. **Key-Reveal Accountability** (incentive-based). A validator must reveal its identity in all past elections in order to either claim leadership or unstake. This approach is simple, but only gives weak accountability and relies on economic incentives.
3. **Threshold Accountability** (committee-based). A quorum of validators exceeding some threshold can work together to reconstruct the identity of a past leader. This approach relies on an honest majority of validators, as a malicious majority coalition could learn the identity of upcoming leaders prematurely.

In building these schemes, we identify and construct new cryptographic primitives. We define and construct *re-randomizable timed commitments*, used for single secret leader election, and *timed VRFs*, used for Algorand-style secret leader election. We also propose novel constructions of *indexed VRFs*, which were proposed by [17] with applications to Algorand-style secret leader election.

## 2 Preliminaries

We use  $\lambda$  to denote the security parameter, and  $\text{poly}(\lambda)$  and  $\text{negl}(\lambda)$  to denote polynomial and negligible functions of  $\lambda$ , respectively. We use  $\xleftarrow{\$}$  (or  $\xrightarrow{\$}$ ) to denote the output of a randomized algorithm, or sampling uniformly at random from a range. We assume all

adversaries are limited to running in probabilistic polynomial time (PPT) in the security parameter  $\lambda$ ; some adversaries are further limited to running in  $\sigma(t)$  steps on at most  $p(t)$  parallel processors where noted. We let  $[k]$  denote the set  $\{1, \dots, k\}$ , and we use  $(a, b)$  to denote the set of integers  $x$  such that  $a < x < b$ . We use  $\mathbf{v}$  to denote a vector  $(v_1, \dots, v_n)$ , and write  $\mathbf{v}[i]$  for the  $i^{\text{th}}$  component of  $\mathbf{v}$ .

Our schemes use a number of standard cryptographic primitives, including time-lock puzzles, threshold public-key encryption, verifiable delay functions (VDFs), and non-interactive zero-knowledge proofs. We describe the syntax and properties of these primitives in the full version.

### 3 A Taxonomy of Leader Election Protocols

Different leader election protocols may offer different properties:

**Public vs. secret.** In public leader elections, all participants learn the identity of the elected leader at once. This can be achieved by running any distributed randomness protocol [14, 25] and using the output to select a random leader. In this work we are only concerned with secret leader election. In *secret* leader election, by contrast, participants do not know who the elected leader(s) is/are until they reveal themselves. Secrecy helps prevent adaptive attacks, such as targeted corruption or denial-of-service attacks against upcoming leaders.

**Number of leaders.** One might want the protocol to elect just a *single* leader or a committee of  $k$  leaders. If used to elect a block proposer, one leader is typically desired.

**Single vs. probabilistic number of leaders.** Somewhat confusingly, a *single* leader election protocol always elects the same number of leaders, which may be one or a committee of size  $k$ . Boneh et al. [8] considered the case of electing a single leader with no variance, hence the name “single,” even though they noted that their techniques naturally extend naturally to electing a larger committee. *Probabilistic* leader election protocols will elect  $k$  leaders on average but might elect more or fewer due to the randomness of the protocol. While single leader election is preferable, it is challenging to guarantee when combined with secrecy.

**Weighting.** Unweighted leader election protocols give each participant the same probability of being elected. Weighted protocols give different participants different probabilities of election, for example, proportional to their stake in a PoS setting.

In this paper, we only consider secret election protocols where each participant should be elected with equal probability. We consider both protocols where a single leader must always be elected (Single Secret Leader Election) and protocols where the number of leaders to be elected varies randomly (Probabilistic Secret Leader Election). All of our protocols generalize to electing a committee of any size.

Below, we recall the definition of Single Secret Leader Election from [8]. We modify the syntax slightly to make it accountable; in particular, we modify `Register` to take as input a list  $L$  of all users’ registrations thus far and output a list  $L'$  with the new user’s registration appended. Consequently, we also modify `Register` to take this list as input. We also combine `Elect1` and `Elect2` into a single protocol for ease of presentation, as [8] notes can be done for the shuffle-based protocol we build off of. We note that this syntax does not require leader uniqueness and encompasses some Probabilistic Secret Leader Election protocols as well; thus, we define it here as simply Secret Leader Election.

► **Definition 1** (Secret Leader Election [8]). A secret leader election (SLE) protocol is a tuple of PPT algorithms and protocols  $\text{SLE} = (\text{SLE.Setup}, \text{SLE.Register}, \text{SLE.Elect}, \text{SLE.Verify})$

**SLE.Setup** $(\lambda, \ell, N) \rightarrow \text{pp}, \text{sk}_1, \dots, \text{sk}_N, \text{st}_0$  is an algorithm that takes in the number of parties  $N$  and an optional lower bound  $\ell$  on the number of required participants in each election, and outputs public parameters and secret keys for all parties.

**SLE.Register** $(i, \text{pp}, \text{st}, L) \rightarrow \text{st}', L'$  is a protocol run by all parties. It takes as input the index  $i$  of the registering party, the public parameters  $\text{pp}$ , the current state  $\text{st}$ , and the registration list  $L$ . It outputs an updated state  $\text{st}'$  and an updated registration list  $L'$ . The registering party  $i$  receives a nonce  $k_i$ .

**SLE.Elect** $(\text{pp}, \text{st}, R, i, k_i, \text{sk}_i) \rightarrow 1/0, \pi/\perp$  is an algorithm run by each party  $i$  to determine if they won the election.  $R$  is a randomness beacon value,  $k_i$  is the user's nonce, and  $\text{sk}_i$  is the user's secret key. If user  $i$  was elected, outputs  $(1, \pi)$  where  $\pi$  is a proof that they won. Otherwise, it outputs  $(0, \perp)$ .

**SLE.Verify** $(i, \text{pp}, \text{st}, R, \pi_i) \rightarrow 1/0$  is an algorithm run by each party to verify that user  $i$  with proof  $\pi_i$  indeed won the election at state  $\text{st}$  with randomness  $R$ .

Informally, an SLE protocol must be *unpredictable* in that an adversary controlling some subset of the parties cannot predict with non-negligible advantage which honest party was elected (in the event that an honest party is elected).

### 3.1 Single Secret Leader Election

Single secret leader election (SSLE) follows the syntax defined in Definition 1 and must satisfy *uniqueness*, *fairness*, and *unpredictability* as defined first in [8]. Uniqueness ensures that only a single party can be accepted as winner of each election. Fairness ensures that an adversary corrupting  $c$  out of  $N$  parties can win the election with probability at most  $\frac{c}{N}$ . Unpredictability ensures that the identity of the winner cannot be predicted before they reveal themselves.

Boneh et al. [8] defined these properties in terms of security games, where an adversary and challenger engage in the SSLE protocol. The adversary controls the corrupted participants, and the challenger controls the honest participants. The adversary can request that certain honest parties register for elections, and it can specify the inputs itself for corrupted parties to register.

Catalano et al. [12] identified a shortcoming in these definitions, which they subsequently fixed in [13]. We apply their fix and provide these modified definitions below. We defer further discussion of this modification to the full version. There, we prove that for a natural class of protocols, security under the original definitions of [8] implies security under the more stringent definitions of [13].

When we say that an SSLE protocol is *secure*, we mean that it is fair, unpredictable, and unique under the modified definitions below.

► **Definition 2** (Uniqueness [8, 13]). Let  $\text{UNIQUE}[\mathcal{A}, \lambda, \ell, N]$  denote the uniqueness experiment with security parameter  $\lambda$ , played by an adversary  $\mathcal{A}$  and a challenger  $\mathcal{C}$  as follows:

**Setup phase.**  $\mathcal{A}$  picks a number  $c < N$  as well as a set of indices  $M \subsetneq [N]$ ,  $|M| = c$  of users to corrupt. The challenger  $\mathcal{C}$  runs  $\text{pp}, \text{sk}_1, \dots, \text{sk}_N, \text{st}_0 \leftarrow \text{SSLE.Setup}(\lambda, \ell, N)$  and gives  $\mathcal{A}$  the parameters  $\text{pp}$ , state  $\text{st}_0$ , and secrets  $\text{sk}_i$  for  $i \in M$ .

**Elections phase.** Adversary  $\mathcal{A}$  can choose any set of users to register for elections and for any number of elections to occur, where  $\mathcal{A}$  plays the role of users  $U_i$  for  $i \in M$  and  $\mathcal{C}$  plays the role of the rest of the users. The challenger  $\mathcal{C}$  also generates the election randomness  $R \in \mathcal{R}$ . To register a (corrupted or uncorrupted) user,  $\mathcal{A}$  sends the index  $i$  of the user

to  $\mathcal{C}$ , and  $\mathcal{C}$  and  $\mathcal{A}$  together run the protocol  $\text{Register}(i, \text{pp}, \text{st}, L)$  to update the state to  $\text{st}'$  and the list to  $L'$ . If the Register protocol aborts, the game immediately ends with output 1.

Each election begins with  $\mathcal{C}$  generating  $(b_i, \pi_i) \leftarrow \text{SSLE.Elect}(\text{pp}, \text{st}', R, i, \text{sk}_i)$  for each uncorrupted user that has registered for the election. For each uncorrupted user  $i$  that has not registered for that election, it sets  $(b_i, \pi_i) = (0, \perp)$ . Finally,  $\mathcal{C}$  sends  $(b_j, \pi_j)$  for each uncorrupted user to  $\mathcal{A}$ .

**Output phase.** For each election in the elections phase,  $\mathcal{A}$  outputs values  $(b_i, \pi_i)$  for each  $i \in M$ . The experiment outputs 0 if for each election with randomness  $R \in \mathcal{R}$  and state  $\text{st}$ , there is at most one user  $\mathcal{U}_{i^*}$  (either corrupted or uncorrupted) who outputs  $b_{i^*} = 1$  and  $\pi_{i^*}$  such that  $\text{Verify}(i^*, \text{pp}, \text{st}, R, \pi_{i^*}) = 1$ . Otherwise the experiment outputs 1.

We say an SSLE scheme is unique if no PPT adversary  $\mathcal{A}$  can win the uniqueness game except with negligible probability. That is, for all PPT  $\mathcal{A}$  and for any  $\ell < N$  the quantity

$$\Pr[\text{UNIQUE}[\mathcal{A}, \lambda, \ell, N] = 1] \leq \text{negl}(\lambda).$$

If uniqueness only holds so long as there are at least  $t$  uncorrupted users participating in each election, we say that the protocol is  $t$ -threshold unique.

► **Definition 3** (Unpredictability [8, 13]). Let  $\text{UNPRED}[\mathcal{A}, \lambda, \ell, N, n, c]$  denote the unpredictability experiment with security parameter  $\lambda$ , played by an adversary  $\mathcal{A}$  and a challenger  $\mathcal{C}$  as follows:

**Setup phase.**  $\mathcal{A}$  picks a set of indices  $M \subsetneq [N]$ ,  $|M| = c$  of users to corrupt. The challenger  $\mathcal{C}$  runs  $\text{pp}, \text{sk}_1, \dots, \text{sk}_N, \text{st}_0 \leftarrow \text{SSLE.Setup}(\lambda, \ell, N)$  and gives  $\mathcal{A}$  the parameters  $\text{pp}$ , state  $\text{st}_0$ , and secrets  $\text{sk}_i$  for  $i \in M$ .

**Elections phase.** Adversary  $\mathcal{A}$  can choose any set of users to register for elections and for any number of elections to occur, where  $\mathcal{A}$  plays the role of users  $\mathcal{U}_i$  for  $i \in M$  and  $\mathcal{C}$  plays the role of the rest of the users. The challenger  $\mathcal{C}$  also generates the election randomness  $R \in \mathcal{R}$ . To register a (corrupted or uncorrupted) user,  $\mathcal{A}$  sends the index  $i$  of the user to  $\mathcal{C}$ , and  $\mathcal{C}$  and  $\mathcal{A}$  together run the protocol  $\text{Register}(i, \text{pp}, \text{st}, L)$  to update the state to  $\text{st}'$  and the list to  $L'$ . If the Register protocol aborts, the game immediately ends with output 1.

Each election begins with  $\mathcal{C}$  generating  $(b_i, \pi_i) \leftarrow \text{SSLE.Elect}(\text{pp}, \text{st}', R, i, \text{sk}_i)$  for each uncorrupted user that has registered for the election. For each uncorrupted user  $i$  that has not registered for that election, it sets  $(b_i, \pi_i) = (0, \perp)$ . Finally,  $\mathcal{C}$  sends  $(b_j, \pi_j)$  for each uncorrupted user to  $\mathcal{A}$ .

**Challenge phase.** At some point after all users  $\mathcal{U}_j$  for  $j \in [n]$  have registered,  $\mathcal{A}$  indicates that it wishes to receive a challenge, and one more election occurs. In this election,  $\mathcal{C}$  does not send  $(b_j, \pi_j)$  for each uncorrupted user to  $\mathcal{A}$ . Let  $\mathcal{U}_i$  be the winner of this election. The game ends with  $\mathcal{A}$  outputting an index  $i' \in [N]$ . If, for  $\mathcal{U}_i$  elected in the challenge phase,  $i \in M$ , then the output of  $\text{UNPRED}[\mathcal{A}, \lambda, \ell, N, n, c]$  is set to 0. Otherwise,  $\text{UNPRED}[\mathcal{A}, \lambda, \ell, N, n, c]$  outputs 1 iff  $i = i'$ .

We say that an SSLE scheme  $\mathcal{S}$  is unpredictable if no PPT adversary  $\mathcal{A}$  can win the unpredictable game with greater than negligible advantage when the winner of the election is uncorrupted. That is, for all PPT  $\mathcal{A}$ , for any  $c \leq n - 2$ ,  $n \leq N$ , and for any  $\ell < N$  the quantity

$$\Pr[\text{UNPRED}[\mathcal{A}, \lambda, \ell, N, n, c] = 1 | i \in [N] \setminus M] \leq \frac{1}{n - c} + \text{negl}(\lambda).$$

If unpredictability only holds for  $c < t$  for some  $t > 0$ , we say that  $\mathcal{S}$  is  $t$ -threshold unpredictable.

► **Definition 4** (Fairness [8, 13]). Let  $\text{FAIR}[\mathcal{A}, \lambda, \ell, N, n, c]$  denote the uniqueness experiment with security parameter  $\lambda$ , played by an adversary  $\mathcal{A}$  and a challenger  $\mathcal{C}$  as follows:

**Setup phase.**  $\mathcal{A}$  picks a set of indices  $M \subseteq [N]$ ,  $|M| = c$  of users to corrupt. The challenger  $\mathcal{C}$  runs  $\text{pp}, \text{sk}_1, \dots, \text{sk}_N, \text{st}_0 \leftarrow \text{SSLE.Setup}(\lambda, \ell, N)$  and gives  $\mathcal{A}$  the parameters  $\text{pp}$ , state  $\text{st}_0$ , and secrets  $\text{sk}_i$  for  $i \in M$ .

**Elections phase.** Adversary  $\mathcal{A}$  can choose any set of users to register for elections and for any number of elections to occur, where  $\mathcal{A}$  plays the role of users  $\mathcal{U}_i$  for  $i \in M$  and  $\mathcal{C}$  plays the role of the rest of the users. The challenger  $\mathcal{C}$  also generates the election randomness  $R \in \mathcal{R}$ . To register a (corrupted or uncorrupted) user,  $\mathcal{A}$  sends the index  $i$  of the user to  $\mathcal{C}$ , and  $\mathcal{C}$  and  $\mathcal{A}$  together run the protocol  $\text{Register}(i, \text{pp}, \text{st}, L)$  to update the state to  $\text{st}'$  and the list to  $L'$ . If the  $\text{Register}$  protocol aborts, the game immediately ends with output 1.

Each election begins with  $\mathcal{C}$  generating  $(b_i, \pi_i) \leftarrow \text{SSLE.Elect}(\text{pp}, \text{st}', R, i, \text{sk}_i)$  for each uncorrupted user that has registered for the election. For each uncorrupted user  $i$  that has not registered for that election, it sets  $(b_i, \pi_i) = (0, \perp)$ . Finally,  $\mathcal{C}$  sends  $(b_j, \pi_j)$  for each uncorrupted user to  $\mathcal{A}$ .

**Challenge phase.** At some point after all users  $\mathcal{U}_j$  for  $j \in [n]$  have registered,  $\mathcal{A}$  indicates that it wishes to receive a challenge, and one more election occurs.  $\text{FAIR}[\mathcal{A}, \lambda, \ell, N, n, c]$  outputs 1 if there is no  $i \in [n] \setminus M$  for which  $\text{Verify}(i, \text{pp}, \text{st}, R, \pi_i) = 1$  in the challenge election.

We say that an SSLE scheme  $\mathcal{S}$  is fair if no PPT adversary  $\mathcal{A}$  can win the fairness game with greater than negligible advantage. That is, if for all PPT  $\mathcal{A}$ ,  $n \leq N$ ,  $c < n$ , and for any  $\ell < N$ ,

$$\left| \Pr[\text{FAIR}[\mathcal{A}, \lambda, \ell, N, n, c] = 1] - c/n \right| \leq \text{negl}(\lambda).$$

If fairness only holds for  $c < t$  for some  $t > 0$ , we say  $\mathcal{S}$  is  $t$ -threshold fair.

[8] notes that these definitions can be easily extended to accommodate elections picking a fixed number of multiple leaders. This is in contrast to probabilistic leader election protocols, where the number of elected leaders may vary randomly from election to election.

### 3.2 Accountability for Single Secret Leader Election

Here, we define an additional property: accountability. An accountable scheme features a  $\text{Recover}$  protocol that informs all parties when an elected leader withholds. This allows the protocol to impose consequences on withholding parties, whereas with standard leader election, parties could withhold undetectably.

$\text{SSLE.Recover}(\text{pp}, \text{st}, R, L, \mathcal{U}_i) \rightarrow 1/0/\perp$  takes as input a state  $\text{st}$ , a random beacon output  $R$ , a registration list  $L$ , and a user  $\mathcal{U}_i$ . If the output is 1, this means that  $\mathcal{U}_i$  could generate a valid proof of leadership  $\pi_i$  with respect to state  $\text{st}$  with randomness  $R$ . If the output is 0,  $\mathcal{U}_i$  could not generate such a proof. If the output is  $\perp$ ,  $\mathcal{U}_i$  must have deviated from the protocol in some way, and it is unknown whether they could prove leadership.

Both strong and weak accountability require that for any *honest* user,  $\text{Recover}$  outputs 1 if the user can claim the election, and 0 otherwise. Strong accountability additionally requires that  $\text{Recover}$  outputs 1 whenever a (possibly misbehaving) user could claim to be the winner of the election at state  $\text{st}$  with random beacon output  $R$ . For strong accountability,  $\text{Recover}$  never outputs  $\perp$ .

Weak accountability has the weaker condition `Recover` does not output 0 for any (possibly misbehaving) user that could claim to win the election (i.e., that user  $i$  can produce a proof  $\pi$  such that  $\text{Verify}(i, \text{pp}, \text{st}, R, \pi_i) = 1$ ). It may output either 1 or  $\perp$  when the winning user misbehaves. Weak accountability is useful even in the case that the output is  $\perp$ , as this proves that the user in question must have misbehaved.

In our delay-based approaches, `Recover` is a slow non-interactive algorithm that can be run by any individual party in time much longer than the election protocol takes to run. This delay ensures unpredictability. In our committee-based approaches, `Recover` is an interactive protocol run by the committee that should succeed as long as a threshold of them participate honestly. In key-reveal approaches, `Recover` requires keys to be disclosed by participants after some number of elections. As withholding parties may also withhold their keys, this approach requires some incentive for revealing.

## Defining Accountability

In defining accountability, we follow the style of definitions from [8] for the properties of uniqueness, unpredictability, and fairness.

The definitions of [8] involve a game where the adversary may corrupt some subset of the parties and run the SSLE protocol while controlling these parties and interacting with the honest parties. We essentially reproduce this game from these previous definitions and modify only the output phase to capture accountability, and the elections phase to apply a fix similar to that of [13]. That is, the adversary wins the game if it causes the protocol to abort. In the accountability game, the adversary aims to cause an election round where a corrupted party is elected and `Recover` fails to recover their nonce.

► **Definition 5** ((Strong, Weak) Accountability). *We let  $\text{wACCOUNT}[\mathcal{A}, \lambda, \ell, N, n]$  and  $\text{sACCOUNT}[\mathcal{A}, \lambda, \ell, N, n]$  denote the weak and strong accountability games played between an adversary  $\mathcal{A}$  and a challenger  $\mathcal{C}$ :*

**Setup phase.**  $\mathcal{A}$  picks a set of indices  $M \subsetneq [N]$  of users to corrupt.  $\mathcal{C}$  runs  $\text{pp}, \text{sk}_1, \dots, \text{sk}_N, \text{st}_0 \leftarrow \text{SSLE.Setup}(\lambda, \ell, N)$  and gives  $\mathcal{A}$  the parameters  $\text{pp}$ , state  $\text{st}_0$ , and corrupted parties' secrets  $\text{sk}_i$  for  $i \in M$ .

**Elections phase.**  $\mathcal{A}$  can choose any set of users to register for elections and for any number of elections to occur, where  $\mathcal{A}$  plays the role of users  $\mathcal{U}_i$  for  $i \in M$  and  $\mathcal{C}$  plays the role of the rest of the users. The challenger  $\mathcal{C}$  also generates the election randomness  $R \in \mathcal{R}$ . To register a (corrupted or uncorrupted) user,  $\mathcal{A}$  sends the index  $i$  of the user to  $\mathcal{C}$ , and  $\mathcal{C}$  and  $\mathcal{A}$  together run the protocol  $\text{Register}(i, \text{pp}, \text{st}, L)$  to update the state to  $\text{st}'$  and the list to  $L'$ . If the `Register` protocol aborts, the game immediately ends with output 1.

Each election begins with  $\mathcal{C}$  generating  $(b_i, \pi_i) \leftarrow \text{SSLE.Elect}(\text{pp}, \text{st}', R, i, \text{sk}_i)$  for each uncorrupted user that has registered for the election. For each uncorrupted user  $i$  that has not registered for that election, it sets  $(b_i, \pi_i) = (0, \perp)$ . Finally,  $\mathcal{C}$  sends  $(b_j, \pi_j)$  for each uncorrupted user to  $\mathcal{A}$ .

**Output Phase.** For each election in the elections phase,  $\mathcal{A}$  outputs values  $(b_i, \pi_i)$  for each  $i \in M$ .

We say  $\mathcal{A}$  violates correctness if it falsely blames an honest user in some way. More precisely, it violates correctness if and only if for some election that occurred with randomness  $R$ , state  $\text{st}$ , and registration list  $L$  at election time, there is some uncorrupted user  $\mathcal{U}_i$  such that either:

**Falsely blames an honest user for withholding:**  $b_{i'} = 0$  where

$(b_{i'}, \pi_{i'}) \leftarrow \text{SSLE.Elect}(\text{pp}, \text{st}', R, i', \text{sk}_{i'})$  and  $\text{SSLE.Recover}(\text{pp}, \text{st}', R, L, \mathcal{U}_{i'}) = 1$ , or

**Falsely blames an honest user for other misbehavior:**

$\text{SSLE.Recover}(\text{pp}, \text{st}', R, L, \mathcal{U}_{i'}) = \perp$ .

The strong accountability experiment outputs 1 if and only if  $\mathcal{A}$  violates correctness, or for some election with randomness  $R$ , state  $\text{st}'$ , and registration list at election time  $L$ , there is a corrupted user  $\mathcal{U}_{i^*}$  who outputs  $b_{i^*} = 1$  and  $\pi_{i^*}$  such that  $\text{SSLE.Verify}(i^*, \text{pp}, \text{st}, R, \pi_{i^*}) = 1$  and  $\text{SSLE.Recover}(\text{pp}, \text{st}, R, L, \mathcal{U}_{i^*}) \neq 1$ .

The weak accountability experiment outputs 1 if and only if  $\mathcal{A}$  violates correctness, or for some election with randomness  $R$  and state  $\text{st}$ , and registration list at election time  $L$ , there is a corrupted user  $\mathcal{U}_{i^*}$  who outputs  $b_{i^*} = 1$  and  $\pi_{i^*}$  such that  $\text{SSLE.Verify}(i^*, \text{pp}, \text{st}, R, \pi_{i^*}) = 1$  and  $\text{SSLE.Recover}(\text{pp}, \text{st}, R, L, \mathcal{U}_{i^*}) = 0$ .

We say an SSLE scheme is strongly/weakly accountable, respectively, if no PPT adversary  $\mathcal{A}$  can win the strong/weak accountability game except with negligible probability. That is, for all PPT  $\mathcal{A}$  and for any  $\ell < N$ ,

$$\Pr[(\text{s/w})\text{ACCOUNT}[\mathcal{A}, \lambda, \ell, N, n] = 1] \leq \text{negl}(\lambda).$$

If accountability only holds as long as there are at least  $\tau$  uncorrupted users participating in  $\text{SSLE.Recover}$ , we say that the scheme is  $\tau$ -threshold (weakly/strongly) accountable.

### 3.3 Probabilistic Secret Leader Election

Recall that in probabilistic secret leader election (PSLE), the number of elected leaders is randomly distributed. Often, this number is one in expectation, and there is a tie-breaking procedure to agree on a single leader when the protocol elects multiple. PSLE encompasses a large class of protocols that lack unifying definitions, and the SSLE definitions presented above do not apply because of differing syntax and number of elected leaders. In this paper, we focus on an approach to PSLE which we call *Algorand-style PSLE*, an abstraction of the scheme used by Algorand [23].

**Algorand-style PSLE.** Each party holds a VRF public-secret key pair  $(K_{\text{pub}}, K_{\text{priv}})$ . A fresh random beacon value  $R$  is generated for each election and is available to all parties. A party wins an election if  $(y, \pi) \leftarrow \text{VRF.Eval}(K_{\text{priv}}, R)$  and  $y < T$ , where  $T$  is a threshold controlling the expected number of parties elected. Parties can prove they have been elected by providing their VRF proof  $\pi$ , which other parties can verify using their public key. In the event that multiple parties' VRFs yield values under the threshold, the winner is chosen according to some tie-breaking rule.

By pseudorandomness of the VRF, prior to the winner revealing its proof it is not possible to tell who has won the election. Thus, if a winner never reveals their VRF output, it is impossible to tell that they should have won. Furthermore, if there is a tie, it is impossible to tell that the winning party withheld.

In Section 4.2, we show how to make Algorand-style PSLE accountable by using a notion called a *timed VRF* that we define. This allows all other parties to evaluate VRF outputs using a slow function (that preserves secrecy since it takes longer to evaluate than the election takes to run).

## 4 The Delay-based Approach

One approach is to replace cryptographic primitives with time-based variants. In general, time-based cryptographic variants feature a fast computation function (requiring a secret key) and an equivalent slow (inherently sequential) computation function which can be computed by anybody. Timed commitments are a classic example: the original committer can efficiently open the commitment, but any party can *force open* the commitment via a slow computation.

Time-based primitives can add accountability to protocols by keeping some information (such as the identity of a leader) secret in the short term while enabling eventual public computation for accountability. Security relies on the assumed computational delay, without any economic assumptions or an honest majority.<sup>2</sup> We show two new time-based cryptographic tools which can be used for secret leader election: re-randomizable timed commitments and timed verifiable random functions.

### 4.1 Accountable SSLE from re-randomizable timed commitments

We can construct an accountable SSLE protocol by replacing the commitments from the shuffle-based protocol of [8] with *re-randomizable timed commitments (RRTCs)*, which we define and construct here. An RRTC commits to random keys in such a way that the commitments can be re-randomized, and for a limited time period the commitments are hiding. After this time period, anyone can open the commitment to learn the key. Our RRTC construction combines the DDH-based commitment scheme used in [8] with any time-lock puzzle in a natural way. We extend the definitions of re-randomizable commitments from [10] and timed commitments from [9].

► **Definition 6** (Re-Randomizable Timed Commitment (RRTC)). *An RRTC is a tuple of algorithms (Setup, Commit, Randomize, Test, SlowOpen) with the following syntax and properties:*

**Setup**( $\lambda, t$ )  $\rightarrow$  **pp**: *outputs public parameters pp,*

**Commit**(**pp**,  $t$ )  $\rightarrow$  ( $c, k, \mathbf{aux}$ ): *outputs a commitment  $c$ , a key  $k$ , and auxiliary information  $\mathbf{aux}$ ,*

**Randomize**(**pp**,  $c$ )  $\rightarrow$   $c'$ : *outputs a re-randomization  $c'$  of the commitment,*

**Test**(**pp**,  $c, k$ )  $\rightarrow$  {**true**, **false**}: *outputs true or false depending on whether  $k$  is a valid key for the (possibly re-randomized) commitment  $c$ ,*

**SlowOpen**(**pp**,  $c, \mathbf{aux}$ )  $\rightarrow$   $\tilde{k}$ : *if  $c$  is an honestly formed commitment, outputs the key  $\tilde{k} = k$  committed to by  $c$ .*

**Correctness:** If  $c$  is an honestly-formed (and possibly re-randomized) commitment to  $k$ ,  
 $\text{Test}(\mathbf{pp}, c, k) = \text{true}$ .

**Binding:** It is computationally infeasible to find  $c, k, k'$  such that  $\text{Test}(\mathbf{pp}, c, k)$  and  $\text{Test}(\mathbf{pp}, c, k')$ .

**Hiding for random keys:** The commitment reveals nothing about  $k$ .

**Honest soundness:** Given an honestly-formed (and possibly honestly re-randomized) commitment, **SlowOpen** recovers the committed key.

**Re-randomizability:** **Randomize** outputs another valid commitment to the same key.

<sup>2</sup> A dishonest majority assumption may seem odd since many applications of leader election exist in scenarios like consensus protocols, which require an honest supermajority. However, we note that there may be subtle differences in the majority's honesty, for example they might collude to learn a future leader's identity early but not to actively disrupt consensus.



**Unlinkability:** An adversary running in sequential time at most  $t$  cannot determine if  $\tilde{c}$  is a re-randomization of commitment  $c_1$  (committing to  $k_1$ ) or of commitment  $c_2$  (committing to  $k_2$ ) given that  $\tilde{c}$  is a re-randomization of one of them.

$\text{Setup}(\lambda, t) \rightarrow \text{pp}$ $\text{TLP.pp} \xleftarrow{\$} \text{TLP.Setup}(\lambda, t)$ $\mathbb{G}, g, p \xleftarrow{\$} \text{GroupGen}(\lambda)$ $\text{pp} \leftarrow (\mathbb{G}, g, p, \text{TLP.pp})$ $\text{Commit}(\text{pp}, t) \rightarrow (c, k, \text{aux})$ $x, y \xleftarrow{\$} \text{TLP.GenRandPuzzle}(\text{TLP.pp})$ $k_L, k_R \leftarrow H(x, y)$ $r \xleftarrow{\$} \mathbb{Z}_p$ $c = (g^r, g^{rk_L})$ $k \leftarrow k_L    k_R$ $\text{aux} \leftarrow x$	$\text{Randomize}(\text{pp}, c) \rightarrow c'$ $(u, v) := c$ $r' \xleftarrow{\$} \mathbb{Z}_p$ $c' \leftarrow (u^{r'}, v^{r'})$ $\text{Test}(\text{pp}, c, k) \rightarrow \{\text{true}, \text{false}\}$ $(u, v) := c$ $k_L    k_R \leftarrow k$ $\text{return } (u^{k_L} \stackrel{?}{=} v)$ $\text{SlowOpen}(\text{pp}, c, \text{aux}) \rightarrow \tilde{k}$ $\tilde{x} \leftarrow \text{aux}$ $\tilde{k} \leftarrow H(x, \text{TLP.Solve}(\text{TLP.pp}, x))$
--	--

■ **Figure 1** Our re-randomizable timed commitment scheme.

**Honest soundness.** Our *honest soundness* property is a relaxation of the soundness property defined by Boneh and Naor [9]. Soundness requires that a recipient can be convinced that an honestly generated commitment is well-formed. In contrast, we require only that if the commitment is honestly formed, **SlowOpen** recovers the key. This relaxation is sufficient for a weak form of accountability where one is satisfied with punishing participants for publishing malformed commitments after the fact. Furthermore, one can efficiently prove that a puzzle failed to open correctly: Simply compute  $\tilde{k} \leftarrow \text{SlowOpen}(\text{pp}, c, \text{aux})$  and show that for  $(u, v) = c$ ,  $u^{\tilde{k}} \neq v$ . If the commitment were well-formed, we would have  $u^{\tilde{k}} = v$ .

**Our RRTC construction from DDH.** [8] suggests the following construction of a re-randomizable commitment based on the Decisional Diffie-Hellman (DDH) assumption. Let  $\mathbb{G}$  be a cyclic group of prime order  $p$  for which the (DDH) assumption holds, and let  $g \in \mathbb{G}$  be a generator for this group. Their commitment to a uniformly random key  $k$  is  $(g^r, g^{rk})$  for a uniform  $r$ . To open a commitment  $(u, v)$  given  $k$ , one checks that  $u^k = v$ . Furthermore, this commitment can be re-randomized by drawing a uniform  $r'$  and computing  $(u^{r'}, v^{r'})$ .

We define our scheme, shown in Figure 1, to be compatible with the commitments generated in the shuffle-based protocol of [8], which results in the slightly unnatural use of terms  $k_L, k_R$ . If the above commitment is used in the SSLE protocol, two parties may submit commitments to the same  $k$ . Therefore,  $k_L, k_R$  are introduced to detect when two commitments  $(g^r, g^{rk}), (g^{r'}, g^{r'k})$  have the same nonce  $k$  and prevent such a registration. Each party reveals  $k_{iR}$  at registration time, and in order to open their commitment  $c_i = (u, v)$  they must provide  $k'_i$  such that  $k'_{iL}, k'_{iR} \leftarrow H(k'_i)$ ,  $k'_{iR} = k_{iR}$ , and  $u^{k'_{iL}} = v$ . Only a party that has revealed a matching  $k'_{iR}$  can claim to be leader for that commitment. Thus, if two parties submit commitments to the same  $k_L$ , the hash function ensures either that one of them can never claim an election, or they must have the same  $k_R$ . We check for duplicate  $k_R$ 's at registration time to rule out this latter case. We also note that **Commit** could instead output  $(g, g^{k_L})$ , which would still be hiding and binding for random keys. However, this would allow an adversary to distinguish between commitments that have and have not been re-randomized, and this scheme would not achieve unlinkability as defined in [10].

**Achieving soundness using NIZKs.** One could modify our scheme to satisfy the stronger notion of soundness from [9] by requiring the committer to provide a non-interactive zero knowledge proof  $\pi$  that  $c$  is a commitment to  $H(x, \text{TLP.Solve}(x))$  for  $x = \text{aux}$ . This could be achieved could use a generic zk-SNARK; for this, it is convenient to use a VDF as the time-lock puzzle to avoid heavy computation in verifying its output. Designing a sound RRTC without the use of generic SNARKs is an interesting direction for future work.

**The BEHG protocol.** We now briefly recall the “high-communication” variant of the shuffle-based SSLE scheme (also known as the BEHG protocol) from Boneh et al. [8], which we describe generically for any re-randomizable commitment scheme. In this variant, we maintain a public list of commitments belonging to the parties in the election. When a new user registers, they generate a nonce and corresponding commitment. They re-randomize the commitments in the list, shuffle them, and insert their own commitment at a random location. This new user posts a NIZK proof that they shuffled the list correctly: each commitment in the old list appears exactly once, re-randomized, in the new list.

The algorithms for this scheme do the following. **Setup** creates an empty to-be-shuffled list  $l$ , to which commitments will be added when users register; (in our modified protocol, it also creates an empty not-to-be-shuffled list  $L$ ). In **Register**, the registering user samples a random key  $k_i$  and splits its hash into two parts  $k_{iL}, k_{iR} \leftarrow H(k_i)$ , then computes a re-randomizable commitment  $c_i$  to  $k_{iL}$ . It re-randomizes the commitments in  $l$  and shuffles  $l$ , then inserts  $c_i$  into  $l$  at a random location. It also provides a NIZK proof of honest shuffling. Each user then examines the current state to get the list  $l$  and checks that the list was properly shuffled using this proof. It also checks that none of the keys  $k_{jR}$  are duplicated. If either of these checks fails, the list is reverted to its most recent state and the protocol continues. **Elect** uses a random beacon value  $R$  given as input to choose a random commitment (i.e. the winning commitment) in  $l$ . If run by the user that submitted this commitment, including its key  $k_i$  as input, it outputs a proof  $\pi_i$  that includes an opening proof for that commitment, allowing the user to claim that election. **Verify** can be run by any user, and it checks if the revealed (by  $\mathcal{U}_i$ ) key  $\tilde{k}$  is consistent with  $\mathcal{U}_i$ 's  $k_{iR}$  from the registration list  $L$  and if  $\tilde{k}$  is consistent with the winning commitment.

**A delay-based accountable SSLE scheme.** Next, we'll show that we can slightly modify this scheme to be weakly or strongly accountable. Our scheme is described below, and further detail is given in the full version.

The main modification is to use our RRTC instead of their original commitment scheme; our RRTC is exactly the same as their scheme except that our key  $k_i$  is chosen as the hash of an input-output pair to a time-lock puzzle and our  $k_{iL}$  and  $k_{iR}$  are derived directly from  $k_i$  instead of  $H(k_i)$  (as there is no need to hash twice). The weakly accountable version uses our RRTC with honest soundness, and the strongly accountable version uses our RRTC with the additional proof of well-formedness to achieve full soundness. We detail our modifications below; they include a small modification to `SSLE.Register` and defining a `Recover` algorithm.

We make a small modification to `SSLE.Register`: when a user  $\mathcal{U}_i$  registers, it must add its registration to  $L$ .  $L$  is a list of auxiliary information that each party adds to when registering that `SSLE.Recover` will use.  $L$  is separate from the shuffled list, and  $L$  is not shuffled or used in the elections. To be more specific, `SSLE.Register` generates the commitment  $(c, k, \text{aux})$  using `RRTC.Commit` and appends  $(k_{iR}, c, \text{aux}, \text{id}_i)$  to  $L$ , where  $\text{id}_i$  is a string representing its identity. It then re-randomizes  $c$  using `RRTC.Randomize` to obtain  $(g^{r_i}, g^{r_i k_{iL}})$  and continues as in the original protocol from [8]. For the strongly accountable version, during `Register` all

users check the proof of commitment well-formedness and reject the registration if it fails. To claim an election for a chosen commitment  $(u, v)$ , a party provides  $k' = k'_L || k'_R$  and  $(x', y')$  such that  $k' = H(x', y')$ ,  $u^{k'_L} = v$ , and  $k'_R$  matches the on-chain  $k_R$  from that party's initial registration.

We define  $\text{Recover}(\text{pp}, \text{st}, R, L, \mathcal{U}_i) \rightarrow 1/0/\perp$  for this scheme as follows. `Recover` first parses `st` to obtain the current shuffled list of entries and uses  $R$  to choose the winning commitment  $\text{com}^*$ . It then iterates through  $L$  (in case  $\mathcal{U}_i$  registered multiple times). For each entry  $(k_{iR}, c, \text{aux}, \text{id}_i)$  in  $L$  added by  $\mathcal{U}_i$ , it computes  $\tilde{k} \leftarrow \text{SlowOpen}(\text{pp}, c, \text{aux})$ . It checks that  $\text{Test}(\text{pp}, c, \tilde{k}) = \text{true}$ ; if not, it moves onto the next entry in  $L$  added by  $\mathcal{U}_i$ . If it continues, it parses  $\tilde{k}_L || \tilde{k}_R \leftarrow \tilde{k}$ . If  $\text{Test}(\text{pp}, \text{com}^*, \tilde{k}) = \text{true}$ , it outputs 1. If  $\text{Test}(\text{pp}, \text{com}^*, \tilde{k}) = \text{false}$ , it continues. After it has iterated through all of  $L$ , it outputs  $\perp$  if it observed that  $\text{Test}(\text{pp}, c, \tilde{k}) = \text{false}$  for any  $c$  added to  $L$  by  $\mathcal{U}_i$ . Otherwise, it outputs 0.

► **Theorem 7.** *The high-communication shuffling-based SSLE scheme from [8] is a weakly accountable SSLE scheme when instantiated with our RRTC scheme as described above (assuming an adversary that runs in sequential time less than  $t$ ). It is a strongly accountable SSLE scheme when we modify our RRTC scheme to require the committer to provide a NIZK proof that the commitment was honestly generated.*

The proof of this theorem is given in the full version.

**Commitment expiry.** A timed commitment is no longer hiding after enough time has passed for `SlowOpen` to be evaluated. Therefore, this scheme is not secret if registrations remain in the list for time greater than  $t$ , where  $t$  is the runtime of `SlowOpen`. The most natural solution is to run the protocol in epochs of length less than  $t$ . At the beginning of each epoch, the list is cleared and all users must re-register; this ensures that commitments do not stay in the list for too long. Although this re-registration increases communication, this increase can be traded off with the delay required to recover withholders' identities. If one is willing to increase this delay  $t$ , one can tolerate longer epochs and fewer re-registrations.

## Protocol optimizations

**Efficient TLP generation.** Preparing a commitment requires generating a time-lock puzzle pair  $(x, y)$ . For classic repeated-squaring time-lock puzzles, Rivest et al. proposed generating  $(x, y = x^{(2^t)} \pmod{N})$  by taking advantage of the trapdoor  $\varphi(N)$  to compute a reduced exponent  $e = 2^t \pmod{\varphi(N)}$ . This approach also applies for modern VDFs in an RSA group [33]. The drawback of this approach is that each puzzle must use its own modulus  $N$ , making efficient hardware implementation more difficult.

A better approach utilizes *re-randomizable VDFs* [1]. Observe that users do not need to compute a TLP on a specific value; rather, a TLP for a random  $x$  is sufficient. Re-randomizable VDFs (of which repeated-squaring VDFs are a natural example) enable computing random input/output pairs given a single precomputed value  $(g, h = g^{(2^t)})$ . Observe that  $(g^\alpha, h^\alpha)$  is also a valid VDF input/output pair for any  $\alpha$ . Hence, users can generate a puzzle by choosing a random  $\alpha$  and setting  $(x, y) \leftarrow (g^\alpha, h^\alpha)$ .

**Outsourcing TLP computation.** In order to learn the nonce for a pair  $(c, \text{aux})$ , or discover that  $(c, \text{aux})$  was malformed, one must compute the output of a TLP on `aux`. As this computation is slow by design, it is desirable to have a mechanism to outsource this task. Since all registrations  $(c, \text{aux})$  are posted on-chain, we could allow any member of the public to compute these TLP outputs on the participants' behalf. This party could report a malformed commitment by posting this TLP output and a proof of correctness on-chain;

this is especially convenient if one uses a VDF as the TLP. The protocol can then *slash* (confiscate the deposited capital of) the offending participant, and the reporter could receive some of the slashed stake.<sup>3</sup>

## 4.2 Accountable PSLE from timed VRFs

In Algorand-style PSLE [23], each party holds a VRF public-secret key pair  $(K_{\text{pub}}, K_{\text{priv}})$ .  $K_{\text{pub}}$  is known to all. Here, we make the modeling assumption that a fresh random beacon value  $R$  is generated for each election; in Algorand’s actual protocol,  $R$  is a function of the last block produced.  $R$  is available to all parties in the election. A party wins an election if  $(y, \pi) \leftarrow \text{VRF.Eval}(K_{\text{priv}}, R)$  and  $y < T$ , where  $T$  is a threshold specifying the expected number of parties elected. Parties can prove they have been elected by providing their VRF proof  $\pi$ , which other parties can verify using their public key. By pseudorandomness of the VRF, prior to the winner revealing its proof it is not feasible to learn who has won the election.

In the event that multiple parties’ VRFs yield values under the threshold, the party with the lowest VRF output is chosen as the winner. An unintended consequence of this tie-breaking rule is a way for malicious participants to bias the election. Because the randomness  $R$  is a function of the previous leader’s identity, an adversary that controls two parties whose VRF outputs  $y_1$  and  $y_2$  are both below  $T$  may choose which party’s output to reveal in order to generate more favorable randomness for the next election. That is, if  $y_1 < y_2$ , the adversary might choose *not* to reveal  $y_1$  so that its party with  $y_2$  can propose the next block. Because other parties cannot compute the VRF, they cannot learn that the party with  $y_1$  withheld. This attack is therefore undetectable.

We provide accountability by replacing the VRF in Algorand-style elections with a *timed VRF*, a new primitive which we define. A timed VRF has a slow open function that can be run by anyone, without knowledge of the secret key. The slow open function requires a lengthy computation, and before this delay the VRF retains its pseudorandomness. Timed VRFs share some similarities with VDFs, but they are pseudorandom and evaluation is fast given a private key. When we replace the VRFs in Algorand-style PSLE with timed VRFs, all parties eventually learn all other parties’ VRF outputs for all elections. Thus, all parties that have withheld can be identified.

Here, we delineate the formal properties of a timed VRF. We can also define timed weak VRFs (akin to weak VRFs [11]), which are only pseudorandom on randomly chosen inputs:

Our timed VRF definition is (adapted from [27, 16]):

► **Definition 8** (Timed VRF). *A Timed VRF is a tuple of algorithms  $(\text{KeyGen}, \text{Eval}, \text{SlowEval}, \text{Verify})$  where:*

**KeyGen** $(\lambda, t) \rightarrow (K_{\text{pub}}, K_{\text{priv}})$ : *generates a key pair which allows public evaluation with a time delay of  $t$ .*

**Eval** $(K_{\text{priv}}, x) \rightarrow (y, \pi)$ : *outputs a value  $y$  using the private key  $K_{\text{priv}}$ , and a correctness proof  $\pi$ . This function should be fast to evaluate.*

**SlowEval** $(K_{\text{pub}}, x) \rightarrow (y, \pi)$ : *outputs a value  $y$  for input  $x$  without using the private key by completing a sequential computation.*

**Verify** $(K_{\text{pub}}, x, y, \pi) \rightarrow \{0, 1\}$ : *checks if  $y$  is the correct evaluation of  $x$  under  $K_{\text{pub}}$  given proof  $\pi$ .*

---

<sup>3</sup> While we leave the exact incentive design as an open question, it is important that the reporter receive *some* but not all of the slashed stake. Observe that a malicious user can prove their commitment is malformed without executing a slow computation. If the reporter receives all of the slashed stake, a malicious user could report themselves and effectively suffer no penalty.

$\text{KeyGen}(\lambda, t) \rightarrow (K_{\text{pub}}, K_{\text{priv}})$ $(K_{\text{pub}}, K_{\text{priv}}) \leftarrow \text{tdVDF.KeyGen}(\lambda, t)$ $\text{Eval}(K_{\text{priv}}, x) \rightarrow (y, \pi)$ $y', \pi' \leftarrow \text{tdVDF.tdEval}(K_{\text{priv}}, x)$ $\pi \leftarrow (y', \pi')$ $y \leftarrow H(y')$	$\text{SlowEval}(K_{\text{pub}}, x) \rightarrow (y, \pi)$ $y', \pi' \leftarrow \text{tdVDF.Eval}(K_{\text{pub}}, x)$ $\pi \leftarrow (y', \pi')$ $y \leftarrow H(y')$ $\text{Verify}(K_{\text{pub}}, x, y, \pi) \rightarrow \{\text{true}, \text{false}\}$ $(y', \pi') \leftarrow \pi$ $\text{return } H(y') = y \wedge \text{tdVDF.Verify}(K_{\text{pub}}, x, y', \pi')$
--	---

■ **Figure 2** Our timed VRF scheme from a trapdoor VDF.

**Correctness:** For any honestly-formed key pair  $(K_{\text{pub}}, K_{\text{priv}})$  and input  $x$ , given outputs  $(y, \pi) \leftarrow \text{Eval}(K_{\text{priv}}, x)$  and  $(y', \pi') \leftarrow \text{SlowEval}(K_{\text{pub}}, x)$ , it should hold that both  $\text{Verify}(K_{\text{pub}}, x, y, \pi)$  and  $\text{Verify}(K_{\text{pub}}, x, y', \pi')$  return true.

**Unique provability:** For every  $K_{\text{pub}}, x$ , no PPT adversary can find two outputs  $(y_1, \pi_1)$  and  $(y_2, \pi_2)$  such that  $y_1 \neq y_2$  and both  $\text{Verify}(K_{\text{pub}}, x, y_1, \pi_1)$  and  $\text{Verify}(K_{\text{pub}}, x, y_2, \pi_2)$  return true.

**Strong  $t$ -pseudorandomness:** For all PPT adversaries  $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$  where  $\mathcal{A}_1$  is  $t$ -sequential, it holds that:

$$\Pr \left[ b = b' \mid \begin{array}{l} (K_{\text{pub}}, K_{\text{priv}}) \leftarrow \text{GenKey}(\lambda, t) \\ (x_*, \sigma) \leftarrow \mathcal{A}_0^{\text{Eval}(K_{\text{priv}}, \cdot)}(K_{\text{pub}}) \\ b \xleftarrow{\$} \{0, 1\} \\ y_0 \leftarrow \text{Eval}(K_{\text{priv}}, x_*) \\ y_1 \xleftarrow{\$} \mathcal{Y} \\ b' \leftarrow \mathcal{A}_1^{\text{Eval}(K_{\text{priv}}, \cdot)}(\sigma, y_b) \end{array} \right] \leq \frac{1}{2} + \text{negl}(\lambda)$$

**Weak  $t$ -pseudorandomness:** For all PPT,  $t$ -sequential adversaries  $\mathcal{A}$ , it holds that:

$$\Pr \left[ b = b' \mid \begin{array}{l} (K_{\text{pub}}, K_{\text{priv}}) \leftarrow \text{GenKey}(\lambda, t) \\ x_* \xleftarrow{\$} \mathcal{X} \\ b \xleftarrow{\$} \{0, 1\} \\ y_0 \leftarrow \text{Eval}(K_{\text{priv}}, x_*) \\ y_1 \xleftarrow{\$} \mathcal{Y} \\ b' \leftarrow \mathcal{A}^{\text{Eval}(K_{\text{priv}}, \cdot)}(\sigma, y_b) \end{array} \right] \leq \frac{1}{2} + \text{negl}(\lambda)$$

### Timed weak VRFs from trapdoor VDFs

We present an efficient construction, given in Figure 2, of a timed weak VRF from a *trapdoor VDF*, as formalized by Wesolowski [33]. Wesolowski observes that while repeated squaring is conjectured to be an inherently sequential function in a group of unknown order, given the group order it becomes efficient as the exponent  $2^t$  can be reduced modulo the group order. Therefore, the group order serves as a trapdoor enabling efficient computation of the VDF for arbitrarily high delay parameters. The RSA group<sup>4</sup>  $(\mathbb{Z}/N)^*$  is a natural example with the group order  $\varphi(N)$  serving as the trapdoor.

<sup>4</sup> Note that using  $(\mathbb{Z}/N)^*$  is insecure for VDFs as the low-order assumption does not hold, instead the group of quadratic residues  $\mathbb{QR}_N$  or the group  $\mathbb{G}^+ = (\mathbb{Z}/N)^*/\{\pm 1\}$  should be used [7, §6].

The weak pseudorandomness of the construction given in Figure 2 follows almost directly from its unpredictability when considered as a VDF. Unpredictability of a VDF requires that an adversary cannot predict the output on a random input; [6] notes that hashing the output of a function that is unpredictable in this sense yields a pseudorandom output in the random oracle model. Thus, we obtain a function whose output is pseudorandom on a random input which is exactly weak pseudorandomness. We note that in our model of Algorand-style PSLE, the input to the VRF is a random value  $R$ , and thus weak pseudorandomness is sufficient. Furthermore, as long as the distribution of the input  $x$  has  $\lambda$  bits of min-entropy,  $H(x)$  is uniform in the random oracle model.

## 5 The Key-disclosure Approach

We can achieve accountability if all users reveal their secret keys after each election, enabling any party to recompute what the results should have been and detect any deviation. Of course, the critical question is how we can ensure that users actually disclose their key material.

### 5.1 Key disclosure via slashing

The most general approach is to compel users to publish key material under the threat of *slashing*, or losing deposited capital, if key material is not properly disclosed. This approach works naturally for staking protocols in which users have committed a pool of deposited stake to participate. One option is for users to disclose and re-key at regular intervals (e.g. after each epoch). This adds continual overhead, but many protocols already impose a similar requirement of per-epoch setup. Another option is to have users disclose only when they attempt to withdraw their deposited capital and stop acting as participants (unstaking). This reduces the frequency and overhead of key disclosure but means it will take longer to detect misbehavior. Finally, users might disclose whenever they are elected as leader or otherwise stand to earn rewards. This approach works naturally with protocols employing the YOSO paradigm [21] to defend against adaptive adversaries, in which case keys are one-time use by design.

Whenever users disclose keys, a waiting period is needed before any withdrawal of staked capital to ensure adequate time for auditors to check for misbehavior using the disclosed key, for example by re-deriving all VRF values the user should have computed under this key and seeing if the users did not act as leader when they were expected to. They also require careful analysis of incentives; if slashing penalties are too low, attackers may be willing to absorb the loss as part of an attack. We leave detailed mechanism design of this approach as future work but note that it is a simple and potentially powerful tool.

### 5.2 Implicit key disclosure from indexed VRFs

We can improve the approach of disclosing key material whenever a user is elected leader by using an *indexed VRF* (iVRF) instead of a VRF in Algorand-style leader election. This idea of an iVRF was formalized by Esgin et al. [17], though the construction was used earlier by Azouvi et al [3]. The iVRF Eval function takes as input the *index*  $i$ , and proving a VRF value for index  $i$  reveals the key material for all indices  $j \leq i$ . In a distributed consensus setting, the index is the round number, observing that (for protocols with per-round finality), there is no downside to revealing key material for past rounds. Interestingly, while Esgin et

$\text{KeyGen}(\lambda, m) \rightarrow (K_{\text{pub}}, K_{\text{priv}})$ $K_{\text{priv}} \xleftarrow{\$} \{0, 1\}^\lambda$ $K_{\text{pub}} = K^m = H^m(K_{\text{priv}})$	$\text{Eval}(K_{\text{priv}}, x, i, m) \rightarrow (y, \pi)$ $\pi = K^i := H^{m-i}(K_{\text{priv}})$ $y = H(K^i, x)$ $\text{Verify}(K_{\text{pub}}, x, i, y, \pi) \rightarrow \{\text{true}, \text{false}\}$ $\text{return } H^i(\pi) = K_{\text{pub}} \wedge y = H(\pi, x)$
--	--

■ **Figure 3** An indexed VRF from hash functions, due to Esgin et al. [17].

$\text{Setup}(\lambda, m) \rightarrow \text{pp}$ $(\mathbb{G}, g, e \xleftarrow{\$} \text{GroupGen}(\lambda))$ $\mathbf{g} \leftarrow \left\{ g^{e^i} \right\}_{i=0}^m$ $\text{pp} \leftarrow (\mathbb{G}, g, \mathbf{g})$ $\text{KeyGen}(\lambda, \text{pp}) \xrightarrow{\$} (K_{\text{pub}}, K_{\text{priv}})$ $K_{\text{priv}} = \alpha \xleftarrow{\$} \mathcal{B}$ $K_{\text{pub}} = (\mathbf{g}[m])^\alpha$	$\text{Eval}(\text{pp}, K_{\text{priv}}, x, i, m) \rightarrow (y, \pi)$ $(\mathbb{G}, g, \mathbf{g}) \leftarrow \text{pp}$ $\pi = K^i := (\mathbf{g}[m-i])^\alpha$ $y = H(K^i, x)$ $\text{Verify}(K_{\text{pub}}, x, i, y, \pi) \rightarrow \{\text{true}, \text{false}\}$ $\text{return } \pi^{e^i} = K_{\text{pub}} \wedge y = H(\pi, x)$
--	---

■ **Figure 4** An indexed VRF from groups of unknown order. The space  $\mathcal{B}$  must be large enough to ensure  $g^\alpha$  is indistinguishable from random for  $\alpha \xleftarrow{\$} \mathcal{B}$ . Statistical security can be achieved with  $|\mathcal{B}| \geq 2 \cdot |\mathbb{G}|$ ; whereas  $|\mathcal{B}| \geq 2^{2\lambda}$  under the SEI assumption [15].

al. proposed indexed VRFs due to their simplicity and potential for quantum-resistance, we observe here that they also provide accountability by enabling observers to compute a user's past VRF values each time they publish a VRF output in any round.

We first recall Esgin et al.'s construction from hash chains [17] (Figure 3), then present two novel constructions of iVRFs. Our iVRF from groups of unknown order decreases precomputation cost relative to [17] and allows all users to work in the same group (Figure 4). Our construction based on a trapdoor permutation (Figure 5) offers the novel advantage that users can maintain the same key *indefinitely* (e.g. for an unlimited number of indices) with no precomputation.

### 5.2.1 Indexed VRFs from hash functions

The Esgin et al. [17] indexed VRFs similar to classic notions of hash chains [24, 28], as shown in Figure 3. Essentially, each user computes a chain of round-specific keys  $K^i = H^{m-i}(K_{\text{priv}})$  for round  $i$ . The total number of indices supported,  $m$ , should be chosen to cover, say, one epoch. Note that revealing  $\pi_i = K^i$  as a proof for round  $i$  makes computing prior values easy for indices  $j \leq i$ : simply compute  $K^j = H^{i-j}(K^i)$ .

Naively, computing and verifying this proof requires computing  $O(m)$  hashes each, though some tradeoffs are available if the prover stores some intermediate keys  $K^j$  to compute proofs with  $O(\sqrt{m})$  computation and storage. Esgin et al. also describe tree-based variants enabling logarithmic verification costs, though all of them appear to require  $O(m)$  computation during KeyGen.

### 5.2.2 Indexed VRFs from groups of unknown order

We present an indexed VRF construction in Figure 4 based on groups of unknown order, without assuming a trapdoor. This can have practical benefits in enabling all computation to be performed in one group. We replace the hash function in the above construction

$\text{KeyGen}(\lambda, e) \rightarrow (K_{\text{pub}}, K_{\text{priv}})$ $p, q \leftarrow \text{GenPrimes}(\lambda)$ $N \leftarrow p \cdot q$ $e \leftarrow \text{GenExponent}(\lambda)$ $K^0 \xleftarrow{\$} (1, N)$ $K_{\text{pub}} \leftarrow (N, e, K^0)$ $d = e^{-1} \pmod{\varphi(N)}$ $K_{\text{priv}} \leftarrow (d, \varphi(N))$	$\text{Eval}(K_{\text{priv}}, K_{\text{pub}}, x, i) \rightarrow (y, \pi)$ $N, e, K^0 \leftarrow K_{\text{pub}}$ $d, \varphi(N) \leftarrow K_{\text{priv}}$ $\tilde{d} = d^i \pmod{\varphi(N)}$ $\pi \leftarrow K^i := (K^0)^{\tilde{d}} \pmod{N}$ $y = H(K^i, x)$ $\text{Verify}(K_{\text{pub}}, x, i, y, \pi) \rightarrow \{\text{true}, \text{false}\}$ $e, N, K^0 \leftarrow K_{\text{pub}}$ $\text{return } \pi^{e^i} = K^0 \pmod{N} \wedge y = H(\pi, x)$
--	---

■ **Figure 5** An indexed VRF from trapdoor permutations. We present the scheme here working in the RSA group  $(\mathbb{Z}/N)^*$ , though the idea is generic to any trapdoor permutation. As presented, this scheme requires linear work (in  $i$ ) per verification. This can be reduced to constant cost by caching the latest value of  $K^i$  after each evaluation.

with computing  $e^{\text{th}}$  roots modulo  $N$ . However, in this protocol we do not assume users know the trapdoor, so naively they must precompute the entire chain of keys  $K^i$ , as with the hash-based indexed VRF. Implemented in this way, this approach has no clear advantage over the hash-based approach. However, notice that the precomputed chain  $\mathbf{K} = \{K_{\text{priv}}, (K_{\text{priv}})^e, (K_{\text{priv}})^{e^2}, \dots\}$  can be computed only once in global setup, and then *randomized* by each user as needed. This randomization is straightforward: given a precomputed chain  $\mathbf{g} = \{g_0 = g, g_1 = g^e, g_2 = g^{e^2}, \dots\}$ , a user can sample<sup>5</sup> a random exponent  $\alpha \leftarrow \mathcal{B}$  and compute a randomized chain  $\mathbf{g}' = \{(g_0)^\alpha, (g_1)^\alpha = (g^\alpha)^e, (g_2)^\alpha = (g^\alpha)^{e^2}, \dots\}$ .

This approach removes the linear precomputation and supports efficient proof computation. Assuming access to the precomputed string  $\mathbf{g}$ , needed elements of a user's randomized chain  $\mathbf{g}'$  can be produced on-demand as  $\mathbf{g}'[j] = (\mathbf{g}[j])^\alpha$ . It is also possible to provide efficient proofs of correctness for any evaluation, by adding a succinct proof of exponentiation showing that  $\pi^i = K_{\text{pub}}$ . Since the group order is unknown, either Wesolowski proofs [33] or Pietrzak proofs [29] may be used.

### 5.2.3 Indexed VRFs from trapdoor permutations

We present a new construction in Figure 5 based on trapdoor permutations. Essentially, we replace the hash function in the hash-based construction with a trapdoor permutation, such that computing forwards on the chain involves inverting the permutation with the help of the trapdoor, and going backwards involves evaluating the permutation. Instantiated with an RSA group, this involves computing  $e^{\text{th}}$  roots modulo  $N$ . This is easy given the trapdoor  $\varphi(N)$  but otherwise believed hard for suitably chosen  $N$  under the (weak) RSA assumption [18]. This construction is similar to each user running a private STROBE randomness beacon [5].

Naively, verifying that a revealed key  $K^i$  chains back to the original key  $K^0$  in  $K_{\text{pub}}$  requires  $O(i)$  work via re-execution. This can be avoided via a succinct proof of exponentiation. However, note that since the prover knows the trapdoor  $\varphi(N)$ , Pietrzak proofs [29] (which

<sup>5</sup> The size of this exponent required for security depends on assumptions. Under the Short Exponent Indistinguishability assumption (SEI) [15], for  $\alpha$  sampled from the range  $[0, 2^{2\lambda}]$  the value of  $g^\alpha$  will be indistinguishable from random in  $\mathbb{G}$ . Without this assumption,  $\alpha$  must be sampled from the the range  $[0, |\mathbb{G}| \cdot 2^{2\lambda}]$ .



require  $O(\log i)$  work to verify) must be used instead. Wesolowski proofs [33] are not *strongly unique* [31] if the prover knows the trapdoor and hence would make the iVRF unsound. Alternatively, a more practical approach is for verifiers to cache the most recent revealed key  $K_i$  after each epoch (instead of the originally published value  $K_0$ ), leaving only constant work to verify at each index.

## 6 The Committee-based Approach

In a fundamentally different approach to accountable SSLE, we leverage threshold (public-key) encryption. Each participant’s nonce is threshold-encrypted to the public key of a committee after a threshold cryptographic setup, and then published alongside each participant’s commitment in the registration list  $L$  so that a threshold number of participants can collaborate to reconstruct the nonce of a leader that withholds later. We assume, without loss of generality, that this committee is the group of all SSLE participants, although it could even be a separate group of outsiders.

**An accountable SSLE scheme using threshold encryption.** The following changes are made to the BEHG “high-communication” protocol to yield an accountable SSLE scheme (denoted by ThrPKE-SSLE) that uses threshold encryption. First, SSLE.Setup also runs ThrPKE.Setup. Next, a user  $\mathcal{U}_i$  registering via SSLE.Register must additionally append  $\text{ct}_i = \text{ThrPKE.Enc}(\text{pp}, k_i)$  to the registration list  $L$ . ( $\text{ct}_i$  is not included in the shuffled list).<sup>6</sup> This still means that the winning element of the final shuffled list  $\ell$  is a commitment  $\tilde{c}$ . Given this, we modify SSLE.Verify and SSLE.Recover as follows:

**SSLE.Verify**( $i, \text{pp}, \text{st}, R, \pi$ ) outputs 1 if the revealed key  $\tilde{k}$  is consistent with  $k_{iR}$ , the winning commitment  $\tilde{c}$ , and also  $\text{ct}_i$  (with the randomness used to make the encryption also supplied by the revealer as part of  $\pi$ ), and 0 otherwise.

**SSLE.Recover**( $\text{pp}, \text{st}, R, L, \mathcal{U}_i$ ) outputs  $1/0/\perp$  as follows. SSLE.Recover first parses  $\text{st}$  to obtain the current shuffled list of entries and uses  $R$  to choose the winning commitment  $\tilde{c} = (\tilde{u}, \tilde{v})$ . It then finds  $\mathcal{U}_i$ ’s entry  $(k_{iR}, c_i, \text{ct}_i, \text{id}_i)$  in  $L$  and interactively (involving at least  $\tau$  out of  $n$  users) runs the algorithm ThrPKE.Dec to let  $\tilde{k} = \text{ThrPKE.Dec}(\text{pp}, \{\text{sk}_i\}_{i \in S}, \text{ct}_i)$  and  $\tilde{k}_L || \tilde{k}_R \leftarrow H(\tilde{k})$ . It outputs 1 if both  $k_{iR} = \tilde{k}_R$  and  $\tilde{u}^{\tilde{k}_L} = \tilde{v}$ . It outputs 0 otherwise.

After Recover is run, the registration list must be cleared and all participants must re-register. To improve efficiency, rather than running Recover after every withheld election, one could run Recover once per time period of some length, or only after  $m$  leaders have withheld. This would still return all of these leaders’ identities but mitigate frequent re-registration, at the cost of learning these identities later.

► **Theorem 9.** *Assuming honest shuffling, ThrPKE-SSLE is a single secret leader election protocol that satisfies strong (and weak) accountability, given an adversary that controls less than  $\tau$  participants.*

The proof is included in the full version.

<sup>6</sup>  $\text{ct}_i$  may be included in the shuffled list if the encryption scheme yields an unlinkable commitment as defined in [10], where no adversary can distinguish between two commitments (that include these ciphertexts), even if they have been adversarially re-randomized. Including  $\text{ct}_i$  in the shuffled list would allow the committee to decrypt only the winning ciphertext in Recover rather than the whole registration list.

## 7 Conclusion

We propose and define the notion of accountability for secret leader election. Our schemes take three distinct approaches to address the threat of withholding attacks in secret leader election protocols. These schemes offer a variety of trade-offs between computational overhead for participants, communication requirements, the time delay before absentee leaders will be detected, and strong-versus-weak accountability. Exploring these trade-offs in practice for concrete protocols is an important avenue for future work.

A fundamental question to ask in practice is *how promptly is accountability required?* Committee-based approaches have the advantage of enabling accountability nearly immediately after a leader fails to act during their turn in a protocol, as the committee can compute the election results whenever desired. Delay-based approaches inevitably introduce a longer waiting period, as delay functions must be parameterized conservatively to ensure that malicious attackers cannot compute them quickly enough to learn the election results early. Furthermore, honest parties might not start computing the delay function until after a leader fails to act, to avoid the cost of always computing it even when the leader is honest. Finally, key-disclosure approaches may offer an even longer waiting period for accountability: until the next time a missing leader is elected (or unstakes) or until the end of an epoch.

We also leave open the fundamental question of incentives and mechanism design. Clearly, the key disclosure approach hinges entirely on appropriately incentivizing participants to reveal keys. Delay-based approaches require incentivizing some party to compute the delay functions, which may become non-trivial if they must be computed for every participant. Even the committee-based approaches require incentivizing a committee to act when a leader fails to show up, and not to conspire to learn election results early.

Finally, all of our protocols can, at best, serve as a detection mechanism for withholding attacks (but not prevent them absolutely). Thus, it is vital to design appropriate penalties (slashing) to ensure that such attacks are not profitable. At the same time, slashing may introduce new incentive issues if attackers are incentivized to try denial-of-service attacks on leaders as they attempt to broadcast during their slot. From the point of view of an accountability mechanism there is no difference between a leader who withholds and a leader whose network connection is jammed while they are legitimately attempting to broadcast a block.

Precisely because of these open questions, we present a variety of options rather than a single approach. We hope that future work can utilize these as a toolbox to improve the security of secret leader election protocols in practice.

---

## References

- 1 Arasu Arun, Joseph Bonneau, and Jeremy Clark. Short-lived zero-knowledge proofs and signatures. In *Asiacrypt*, 2022.
- 2 Sarah Azouvi, Patrick McCorry, and Sarah Meiklejohn. Betting on Blockchain Consensus with Fantomette. *arXiv preprint*, 2018. [arXiv:1805.06786](https://arxiv.org/abs/1805.06786).
- 3 Sarah Azouvi, Patrick McCorry, and Sarah Meiklejohn. Winning the caucus race: Continuous leader election via public randomness. *arXiv preprint*, 2018. [arXiv:1801.07965](https://arxiv.org/abs/1801.07965).
- 4 Michael Backes, Pascal Berrang, Lucjan Hanzlik, and Ivan Pryvalov. A framework for constructing Single Secret Leader Election from MPC. In *ESORICS*, 2022.
- 5 Donald Beaver, Konstantinos Chalkias, Mahimna Kelkar, Lefteris Kokoris Kogias, Kevin Lewi, Ladi de Naurois, Valeria Nicolaenko, Arnab Roy, and Alberto Sonnino. STROBE: Stake-based Threshold Random Beacons. In *AFT*, 2023.
- 6 Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable Delay Functions. In *CRYPTO*, 2018.

- 7 Dan Boneh, Benedikt Bünz, and Ben Fisch. A Survey of Two Verifiable Delay Functions. Cryptology ePrint Archive, Paper 2018/712, 2018.
- 8 Dan Boneh, Saba Eskandarian, Lucjan Hanzlik, and Nicola Greco. Single secret leader election. In *AFT*, 2020.
- 9 Dan Boneh and Moni Naor. Timed commitments. In *CRYPTO*, 2000.
- 10 Dan Boneh, Aditi Partap, and Lior Rotem. Post-Quantum Single Secret Leader Election (SSLE) From Publicly Re-randomizable Commitments. In *AFT*, 2023.
- 11 Zvika Brakerski, Shafi Goldwasser, Guy N Rothblum, and Vinod Vaikuntanathan. Weak Verifiable Random Functions. In *TCC*, 2009.
- 12 Dario Catalano, Dario Fiore, and Emanuele Giunta. Adaptively secure single secret leader election from DDH. In *PODC*, 2022.
- 13 Dario Catalano, Dario Fiore, and Emanuele Giunta. Efficient and universally composable single secret leader election from pairings. In *PKC*, 2023.
- 14 Kevin Choi, Aathira Manoj, and Joseph Bonneau. SoK: Distributed Randomness Beacons. In *IEEE Security & Privacy*, 2023.
- 15 Geoffroy Couteau, Michael Kloof, Huang Lin, and Michael Reichle. Efficient range proofs with transparent setup from bounded integer commitments. In *Eurocrypt*, 2021.
- 16 Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. In *PKC*, 2005.
- 17 Muhammed F. Esgin, Oguzhan Ersoy, Veronika Kuchta, Julian Loss, Amin Sakzad, Ron Steinfeld, Xiangwen Yang, and Raymond K. Zhao. A new look at blockchain leader election: Simple, efficient, sustainable and post-quantum. Cryptology ePrint Archive, Paper 2022/993, 2022.
- 18 Dankrad Feist. RSA Assumptions. [rsa.cash/rsa-assumptions/](https://rsa.cash/rsa-assumptions/), 2022.
- 19 Matheus VX Ferreira, Ye Lin Sally Hahn, S Matthew Weinberg, and Catherine Yu. Optimal Strategic Mining Against Cryptographic Self-Selection in Proof-of-Stake. In *Economics and Computation*, 2022.
- 20 Chaya Ganesh, Claudio Orlandi, and Daniel Tschudi. Proof-of-stake protocols for privacy-aware blockchains. In *Eurocrypt*, 2019.
- 21 Craig Gentry, Shai Halevi, Hugo Krawczyk, Bernardo Magri, Jesper Buus Nielsen, Tal Rabin, and Sophia Yakubov. YOSO: You Only Speak Once: Secure MPC with Stateless Ephemeral Roles. In *CRYPTO*, 2021.
- 22 dapplion George Kadianakis, Justin Drake. EIP-7441: Upgrade block proposer election to Whisk. URL: <https://eips.ethereum.org/EIPS/eip-7441>.
- 23 Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nikolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *SOSP*, 2017.
- 24 Neil Haller. The S/KEY one-time password system. In *NDSS*, 1994.
- 25 Alireza Kavousi, Zhipeng Wang, and Philipp Jovanovic. SoK: Public Randomness. Cryptology ePrint Archive, Paper 2023/1121, 2023.
- 26 Thomas Kerber, Aggelos Kiayias, Markulf Kohlweiss, and Vassilis Zikas. Ouroboros Cryptosinus: Privacy-Preserving Proof-of-Stake. In *IEEE Security & Privacy*, 2019.
- 27 Silvio Micali, Michael Rabin, and Salil Vadhan. Verifiable random functions. In *FOCS*, 1999.
- 28 Adrian Perrig, , Ran Canetti, JD Tygar, and Dawn Song. Tesla broadcast authentication. *RSA CryptoBytes*, 5, 2002.
- 29 Krzysztof Pietrzak. Simple verifiable delay functions. In *ITCS*, 2018.
- 30 Protocol Labs. Secret single-leader election (SSLE). URL: <https://github.com/protocol/research-RFPs/blob/master/RFPs/rfp-6-SSLE.md>.
- 31 Philipp Schindler, Aljosha Judmayer, Markus Hittmeir, Nicholas Stifter, and Edgar Weippl. RandRunner: Distributed Randomness from Trapdoor VDFs with Strong Uniqueness. In *NDSS*, 2020.
- 32 Toni Wahrstätter. Selfish Mixing and RANDAO Manipulation. [ethresear.ch/t/selfish-mixing-and-randao-manipulation/16081](https://ethresear.ch/t/selfish-mixing-and-randao-manipulation/16081), 2023.
- 33 Benjamin Wesolowski. Efficient verifiable delay functions. In *Eurocrypt*, 2019.



# BoLD: Fast and Cheap Dispute Resolution

**Mario M. Alvarez**

Offchain Labs, Inc., Clifton, NJ, USA

**Ben Berger** 

Offchain Labs, Inc., Clifton, NJ, USA

**Chris Buckland**


Offchain Labs, Inc., Clifton, NJ, USA

**Edward W. Felten**

Offchain Labs, Inc., Clifton, NJ, USA

**Raul Jordan**

Offchain Labs, Inc., Clifton, NJ, USA

**Akaki Mamageishvili** 

Offchain Labs, Inc., Clifton, NJ, USA

**Aman Sanghi**

Offchain Labs, Inc., Clifton, NJ, USA

**Terence Tsao**

Offchain Labs, Inc., Clifton, NJ, USA

**Henry Arneson**

Offchain Labs, Inc., Clifton, NJ, USA

**Lee Bousfield**


Offchain Labs, Inc., Clifton, NJ, USA

**Yafah Edelman**

Offchain Labs, Inc., Clifton, NJ, USA

**Daniel Goldman**

Offchain Labs, Inc., Clifton, NJ, USA

**Mahimna Kelkar** 

Offchain Labs, Inc., Clifton, NJ, USA

**Harry Ng**

Offchain Labs, Inc., Clifton, NJ, USA

**Victor Shoup** 

Offchain Labs, Inc., Clifton, NJ, USA

---

## Abstract

BoLD is a new dispute resolution protocol that is designed to replace the originally deployed Arbitrum dispute resolution protocol. Unlike that protocol, BoLD is resistant to delay attacks. It achieves this resistance without a significant increase in onchain computation costs and with reduced staking costs.

**2012 ACM Subject Classification** Security and privacy → Distributed systems security

**Keywords and phrases** Optimistic rollups, fraud proofs

**Digital Object Identifier** 10.4230/LIPIcs.AFT.2024.2

**Related Version** *Full Version:* <https://arxiv.org/abs/2404.10491> [1]

## 1 Introduction

In this paper, we introduce BoLD, a *dispute resolution protocol*: it allows conflicting assertions about the result of a computation to be resolved. It is designed for use in a Layer 2 (L2) blockchain protocol, relying on a Layer 1 (L1) blockchain protocol for its security (more generally, it may be used with any parent chain in place of L1 and any child chain in place of L2).

In an optimistic rollup protocol such as Arbitrum, a dispute resolution protocol like BoLD operates as a component of a broader “rollup” protocol, in which validators post claims about the correct outcome of executing an agreed-upon sequence of transactions. These claims are backed by stakes posted by the claimants. If multiple competing claims are posted, the protocol must choose one of them to treat as correct. The goal of BoLD and comparable protocols is to determine, among a set of competing claims about the correct outcome of execution, which of the claims is correct.

Let us state more precisely the problem to be solved. We begin with a starting state,  $S_0$ , on which all parties agree. We assume that a commitment to  $S_0$  has been posted to L1. (In this paper, all commitments are non-hiding, deterministic commitments that will typically be



© Mario M. Alvarez, Henry Arneson, Ben Berger, Lee Bousfield, Chris Buckland, Yafah Edelman, Edward W. Felten, Daniel Goldman, Raul Jordan, Mahimna Kelkar, Akaki Mamageishvili, Harry Ng, Aman Sanghi, Victor Shoup, and Terence Tsao;

licensed under Creative Commons License CC-BY 4.0

6th Conference on Advances in Financial Technologies (AFT 2024).

Editors: Rainer Böhme and Lucianna Kiffer; Article No. 2; pp. 2:1–2:19



Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

implemented using some sort of Merkle tree.) There is also an agreed-upon *state transition function*  $F$ , which maps state  $S_{i-1}$  to  $S_i = F(S_{i-1})$  for  $i = 1, \dots, n$ . In practice, the function  $F$  may be determined by the state transition function of a specific virtual machine, together with a specific sequence of transactions that has also been posted to L1; however, these details are not important here.

Any party may then compute  $S_1, \dots, S_n$  and post an assertion to L1 that consists of a commitment to  $S_n$ . Of course, such an assertion may be incorrect, and the purpose of a dispute resolution protocol is to allow several parties to post conflicting assertions and identify the correct one. Such a dispute resolution protocol is an interactive protocol that makes use of L1:

- each “move” made by a party in the protocol is posted as an input to a smart contract on L1;
- this smart contract will process each move and eventually declare a “winner”, that is, it will identify which one of the assertions made about the commitment to  $S_n$  is correct.

Participation in the protocol requires resources:

- *staking*: tokens required for “staking”, as specified by the dispute resolution protocol;
- *gas*: L1 tokens required to pay for “L1 gas costs”, that is, the cost associated with posting assertions and subsequent moves to L1;
- *computation*: offchain compute costs incurred by the parties who participate in the dispute resolution protocol.

As for staking, the dispute resolution protocol specifies exactly how much and when stakes must be made. When the smart contract declares a winner, some stakes will be confiscated (“slashed”) and some will be returned to the staking parties: corrupt parties may have some or all of their stake confiscated, while honest parties should get all of their stake returned to them. The staking requirement serves to discourage malicious behavior. In addition, confiscated stakes may be redistributed to honest parties to cover their L1 gas costs and offchain compute costs, or simply as a reward for participating in the protocol. These stakes will be held in escrow by the smart contract.<sup>1</sup>

We make the following assumptions:

- L1 provides both *liveness* and *safety*, that is, every transaction submitted to it is *eventually processed* and is *processed correctly*;
- at least one honest party participates in the dispute resolution protocol.

We wish to model the following types of attacks.

**Censorship attacks.** While we assume L1 provides liveness and safety, we assume that it may be subject to *intermittent censorship attacks*. That is, an adversary may be able to *temporarily* censor transactions submitted by honest parties to L1. During periods of censorship, we assume the adversary may still submit its own transactions to L1.

**Ordering attacks.** Even if the adversary is not actively censoring, we assume that the adversary may determine the order of transactions posted to L1 (for example, placing its own transactions ahead of honest parties’ transactions in a given L1 block).

**Resource exhaustion attacks.** Even though a protocol might ensure that all honest parties are “made whole” after the protocol succeeds, the adversary may try to simply exhaust the resources of the honest parties (the staking, gas, and computation resources mentioned above) so the honest parties can no longer afford to participate in the protocol.

---

<sup>1</sup> In a (typical) run of the protocol in which there are no challenges to a correct assertion, there will be no confiscated stakes available, and so some other source of funds must be used to compensate honest parties for their (minimal) costs in participating in the protocol.

**Delay attacks.** The adversary may try to delay the dispute from being resolved within a reasonable amount of time. In such a delay attack, the adversary might try to keep the protocol running for a very large number of moves without resolution – such an attack may become a resource exhaustion attack as well.

To mitigate against a resource exhaustion attack, a well-designed dispute resolution protocol should force the adversary to marshal many more resources than required by the honest parties. Similarly, to mitigate against a delay attack, a well-designed dispute resolution protocol should force an adversary who attempts to delay the protocol to expend a huge amount of resources.

The BoLD protocol is designed as a replacement of the originally deployed Arbitrum dispute resolution protocol. It makes more efficient use of resources than the original Arbitrum protocol while providing a much stronger defense against delay attacks.

**The rest of the paper.** Section 2 briefly reviews the original Arbitrum dispute resolution protocol, sketches the main ideas of BoLD, and discusses how BoLD improves on the original Arbitrum protocol. Section 3 describes a formal attack model for dispute resolution. Section 4 describes BoLD in its simplest form, which we call *single-level BoLD*. Section 5 describes a version of BoLD, which we call *multi-level BoLD*, that reduces some of the *offchain* computational costs of the honest parties. This is the version of BoLD that will replace the originally deployed Arbitrum dispute resolution protocol. Section 6 discusses details regarding gas, staking, and reimbursement.

## 2 Overview and Comparison to Prior Work

We provide a brief overview of BoLD and a comparison to prior approaches in this section. More details can be found in the full version of the paper [1].

### 2.1 Arbitrum Classic

By Arbitrum Classic, we refer to the protocol deployed on Arbitrum in 2020. Note that this version differs in some ways from the original 2018 paper [2].

Arbitrum classic allows parties to post assertions to L1 accompanied by some stake (up until a designated *staking deadline*). Parties with conflicting assertions may then challenge each other. In each such two-party *challenge subprotocol* instance, one party defends their assertion against another party who challenges their assertion. When this subprotocol instance finishes, one of the two parties will win and the other will lose. The dispute resolution protocol allows many such subprotocol instances to proceed, even concurrently. The protocol ends when all remaining parties are staked on the same assertion – which is declared to be the “winner”. The challenge subprotocol guarantees that any honest party will win in any instance of the subprotocol in which it participates. This ensures that the protocol will eventually terminate and declare the correct assertion to be the winner.

In the two-party challenge subprotocol, also called the “bisection game”, one party Daria, defends her assertion, against a challenger Charlie. Daria’s assertion is of the form  $(0, n, S_0, S_n)$ ; this denotes the assertion that executing the state transition function  $F$   $n$  times on state  $S_0$  leads to state  $S_n$  (in practice, the assertion will include commitments to the state rather than the state itself). WLOG, we let  $n$  be a power of two. When Charlie challenges this assertion, the subprotocol game requires Daria to “bisect” her assertion by posting two smaller assertions  $(0, n/2, S_0, S_{n/2})$  and  $(n/2, n, S_{n/2}, S_n)$  of half the size each. Following this, Charlie is now required to pick one of these smaller assertions to challenge. The game

continues in a similar fashion until Charlie’s challenge is one a “one-step” assertion – i.e., of the form  $(i, i + 1, S_i, S_{i+1})$ . At this point, Daria must submit a proof that this one-step assertion is correct which can be efficiently checked by the Layer 1 chain. If this proof is valid, Daria wins the two-party challenge subprotocol and Charlie loses; otherwise, Charlie wins and Daria loses.

The challenge subprotocol guarantees that any honest party will win whenever in which it participates. *However, a corrupt party may stake on the correct initial assertion, but still intentionally lose a challenge.* Because of this, and because honest parties cannot reliably identify other honest parties, each honest party must stake on the correct assertion.

To maintain liveness, the protocol will enforce that the total time taken for each party’s moves is smaller than some amount called the “challenge period.” This can be done using a “chess-clock” approach where a party’s clock runs when it is her turn to make a move. Note that each subprotocol will end within a maximum of 2 challenge periods. The challenge period will be set to be large enough to accommodate any censorship.

There are a number of ways the dispute resolution protocol could orchestrate the challenge subprotocols, depending on the amount of concurrency allowed. In the full concurrency option, there is no limit on the concurrent execution of challenge subprotocol instances. While this guarantees fast resolution, it is susceptible to a resource exhaustion attack.

The deployed version of Arbitrum Classic implements a less concurrent orchestration method, by which each party is allowed to engage in at most one challenge subprotocol instance at a time, although many such subprotocol instances may run concurrently. This method reduces the risk of a resource exhaustion attack significantly, but is instead susceptible to a delay attack. For this reason the deployed Arbitrum Classic has limited participation in the protocol to a permissioned set of parties.

A complete description and discussion of Arbitrum Classic can be found in the full version of the paper [1].

## 2.2 From Arbitrum Classic to BoLD

A primary motivation that separates BoLD was from Arbitrum Classic is to able to give the following guarantees:

- resolves disputes within a bounded amount of time (independent of the number of parties or staked assertions),<sup>2</sup> unlike Arbitrum Classic and
- has gas and staking costs that scale better than Arbitrum Classic.

BoLD achieves these goals using the following ideas.

### 2.2.1 Trustless cooperation

Instead of just committing to the final state  $S_n$ , parties must commit to the entire execution history  $S_1, \dots, S_n$ . This can be done compactly using a Merkle tree whose leaves are the individual commitments  $Com(S_i)$  for  $i = 1, \dots, n$ . With this approach, a similar type of bisection game as in Arbitrum Classic can be designed with the property that there is only one (feasibly computable) justifiable bisection move that can be made at any step. In other words, if a party submits a correct initial assertion, all smaller assertions obtained by

---

<sup>2</sup> “BoLD” is an acronym that stands for *Bounded Liquidity Delay*, emphasizing the fact that it is resistant to delay attacks, unlike partially-concurrent Arbitrum Classic. The term “liquidity” refers to the fact that once the protocol terminates, funds sent from L2 to L1 become available on L1.



bisection must also be correct. This means that the correct assertion need only be staked once, and that the honest parties can build on the work of any *apparently honest* parties that make correct assertions, *even if those parties turn out not to be honest parties after all*. In this sense, all honest parties can work together as a team, although they do not have to trust each other or explicitly coordinate with one another.

With just this one idea, one could fairly easily modify Arbitrum Classic to get a protocol with the following properties. The honest parties in aggregate need to make just one staked assertion. If the corrupt parties together make  $N_A$  staked assertions, then within a bounded amount of time (independent of  $N_A$ ) the honest parties can disqualify all incorrect staked assertions using L1 gas proportional to  $N_A$ . Indeed, this protocol will terminate within two challenge periods.

### 2.2.2 A streamlined protocol

BoLD takes the above idea and turns it into a more elegant and streamlined protocol. We design a new execution pattern in which all assertions (both original assertions and “smaller” assertions obtained from bisection) are organized as nodes in a dynamically growing graph. The edges in the graph represent parent/child relationships corresponding to bisection. In this approach, there are no explicit one-on-one challenges nor associated “chess clocks”. Instead of “chess clocks”, each node in the graph has a “local” timer that ticks so long as the corresponding assertion remains unchallenged by a competing assertion – so higher local timer values indicate that the corresponding assertion is in some sense more likely to be correct (or incorrect but irrelevant to the protocol’s outcome). The values of these local timers are aggregated in a careful way to ultimately determine which of the original assertions (which are roots in this graph) was correct. This idea yields the a version of BoLD that we call *single-level BoLD*, which maintains the same fast termination time of two challenge periods.

### 2.2.3 Multi-level refinement

The downside of the above approach to trustless cooperation is that the offchain compute cost needed to compute the commitments  $Com(S_i)$  for  $i = 1, \dots, n$  and build a Merkle commitment from them may be unacceptably high in practice when  $n$  is large. To address this, we introduce a *multi-level refinement strategy* – the resulting protocol is called *multi-level BoLD*. For example, suppose  $n = 2^{55}$ . Very roughly speaking, in two-level BoLD, we might execute single-level BoLD using the “coarse” iterated state transition function  $F' = F^{2^{25}}$ , which only requires  $2^{30}$  state hashes, and narrow the disagreement with the adversary to one iteration of  $F'$  which is equivalent to  $2^{25}$  iterations of  $F$ . A recursive invocation of single-level BoLD over those iterations of  $F$  would then “refine” the disagreement down to a single invocation of  $F$  which could then be proven using a one-step proof. Each such recursive invocation would require just  $2^{25}$  state hashes. A naive realization of this rough idea would potentially double the amount of time it would take to run the dispute resolution protocol to completion – and even worse, for  $L$ -level BoLD, the time would get multiplied by a factor of  $L$ . Most of this time is due to the built-in safety margins that mitigate against censorship attacks. However, by carefully generalizing the logic of single-level BoLD, and in particular the logic around how timer data on nodes is aggregated, we obtain a protocol that enjoys the same fast termination time as single-level BoLD, namely, two challenge periods. Based on our experience in implementing BoLD, we find that setting  $L = 3$  reduces the offchain compute costs to a reasonable level.

### 3 Formal attack model

In the real world, there may be many parties that participate in the protocol, but we formally describe the attack model in terms of just two parties: the *honest party* and the *adversary*.

- The *honest party* in our formal model represents the actions taken in aggregate in the real world by honest individuals who are correctly following the protocol. While one might consider protocols that require communication and coordination among honest individuals, our protocol does not require this. That said, in our protocol, some amount of coordination between honest individuals may be useful in terms of efficiently distributing the resources necessary to carry out the protocol.
- The *adversary* in our formal model represents the actions taken in aggregate in the real world by corrupt individuals who may well be coordinating their actions with one another, and may also be influencing the behavior of the L1 itself, at least in terms of L1 censorship and ordering attacks.

At the beginning of the challenge protocol, we assume that both the honest party and adversary are initialized with the initial state  $S_0$  and a description of the state transition function  $F$ . We assume that a commitment to  $S_0$  and a description of  $F$  are also recorded on L1. The challenge protocol proceeds in rounds. While time plays a central role in our attack model and our protocol, we shall simply measure time in terms of the number of elapsed rounds. (As an example, if the L1 is Ethereum, a round might be an Ethereum block.)

In each round  $t = 1, 2, \dots$ , the honest party submits a *set*  $Submit_t$  of moves to L1. After seeing the set  $Submit_t$ , the adversary specifies the precise *sequence*  $Exec_t$  of moves to be executed on L1 in round  $t$ . The sequence  $Exec_t$  may contain moves submitted by the honest party in this or any previous round, as well as arbitrary moves chosen by the adversary. The sequence  $Exec_t$  is also given to the honest party, so that its value is available to the honest party in its computation of  $Submit_{t+1}$ . We do not place any limit on how many moves may be submitted to or executed on L1 in a round. We assume that the appropriate party (either the honest party or the adversary) is charged for the gas required to execute each move on L1.

We introduce a *nominal delay parameter*  $\delta$  that models the maximum delay between the submission of a move and its execution under normal circumstances, that is, without censorship. An adversary may choose to *ensor* any given round  $t$ . To model censorship, we define the following rules that the adversary must follow. At the beginning of the attack, we initialize  $Pool$ , a set of (move, round-number) pairs, to the empty set. In each round  $t$ :

- For each move in  $Submit_t$ , we set its *due date* to  $t + \delta$  and add the move, paired with its due date, to the set  $Pool$ .
- If  $t$  is designated a *censored round* by the adversary, then we increment the due date of every move in  $Pool$  (including those just added).
- The adversary chooses some moves in pool to include in  $Exec_t$ , which are then removed from  $Pool$ , subject to the rule:

*any move in Pool whose due date is equal to  $t$  must be included in  $Exec_t$ .*

We introduce another parameter,  $C_{\max}$ , which we call the *censorship budget*. We require that the adversary censors at most  $C_{\max}$  rounds during the attack game. This is our way of modeling the assumption that censorship attacks cannot be carried out indefinitely.

## 4 The BoLD protocol: single-level version

In this section, we describe the BoLD protocol in its purest form, which we call *single-level BoLD*.

### 4.1 Preliminaries

To simplify things, we assume that the parties involved seek to prove a computation of  $n := 2^{k_{\max}}$  steps, for  $k_{\max} \geq 0$ . We assume that a commitment to  $S_0$  is already stored on L1, and we denote this by  $H_0$ .

The goal is to have a commitment to  $S_n$  posted to L1 in such a way that the dispute resolution protocol ensures that this commitment is correct (under well defined assumptions).

### 4.2 The protocol graph

As the protocol proceeds, a data structure will be built on L1 that represents a directed acyclic graph  $\mathcal{G}$ . The structure of  $\mathcal{G}$  will be described below. Initially, the protocol graph  $\mathcal{G}$  is empty, and grows over time. Participants in the challenge protocol will make moves that lead to the creation of new nodes and edges – the details of these moves are described below in Section 4.3.

#### 4.2.1 The syntax of a node

We begin by defining the syntax of a node. A node specifies a *base commitment* and a *span commitment*. The base commitment is supposed to be (but may not be) a commitment to an initial sequence of states, while the span commitment is supposed to be (but may not be) a commitment to an adjacent sequence of states. A node also specifies the length of these two sequences.

More precisely, a node is a tuple

$$(nodeType, lbase, lspan, base, span), \quad (1)$$

where *base* and *span* are the base and span commitments of the node, while *lbase* and *lspan* specify the corresponding commitment lengths. As will become evident, the value *lbase* will always be an integer in the range  $0, \dots, n - 1$ , while the value *lspan* will always be a power of two dividing  $n$ ; moreover, it will always hold that *lbase* is a multiple of *lspan* and  $lbase + lspan \leq n$ . The value *nodeType* is a flag, equal to either

- **regular**, in which case we say the node is a *regular node*, or
- **proof**, in which case we say the node is a *proof node*.

The role of this flag will be described below.

**Correct construction.** Suppose the correct sequence of states is  $S_0, S_1, \dots, S_n$ . We say the node (1) is *correctly constructed* if the base commitment *base* is the root of a Merkle tree whose leaves are commitments to

$$S_0, S_1, \dots, S_{lbase} \quad (2)$$

and the span commitment *span* is the root of a Merkle tree whose leaves are commitments to

$$S_{lbase+1}, \dots, S_{lbase+lspan}, \quad (3)$$

where

- the Merkle tree rooted at  $span$  is a *perfect* binary tree (which is possible because  $lspan$  is always a power of two), and
- the shape of the Merkle tree rooted at  $base$  is determined by the rules governing parent/child relationships, given below.

**Intuition.** Intuitively, a party who makes a move that leads to the creation of a regular node is implicitly claiming that this node is correctly constructed. The nodes created in response to moves made by the honest party will always be correctly constructed. However, the adversary may make moves that result in the creation of incorrectly constructed nodes. The smart contract on L1 cannot distinguish between correctly and incorrectly constructed nodes (however, the honest party, or any entity with access to  $S_0$ , certainly can).

### 4.2.2 Root nodes

Since  $\mathcal{G}$  is directed-acyclic, it will have some number of *roots*, i.e., nodes with in-degree zero. Recall that  $H_0$  is the commitment to  $S_0$ . A **root** in  $\mathcal{G}$  is a regular node of the form

$$r = (\mathbf{regular}, 0, n, H_0, span). \quad (4)$$

**Correct construction.** By definition,  $r$  is correctly constructed if the span commitment  $span$  is a commitment to  $S_1, \dots, S_n$ .

**Intuition.** The party that makes a move that creates a root is claiming that  $span$  is a commitment to  $S_1, \dots, S_n$ .

### 4.2.3 Nonterminal nodes

We call a regular node in  $\mathcal{G}$  of the form

$$v = (\mathbf{regular}, lbase, lspan, base, span), \quad (5)$$

where  $lspan > 1$ , a **nonterminal node**. If this node has any children, it will have exactly two children. These children are of the form

$$v_L = (\mathbf{regular}, lbase, lspan/2, base, span_L) \quad (6)$$

and

$$v_R = (\mathbf{regular}, lbase + lspan/2, lspan/2, H(base, span_L), span_R), \quad (7)$$

for some  $span_L, span_R$  with

$$span = H(span_L, span_R). \quad (8)$$

Here,  $H$  is the hash function used to form the internal nodes of the Merkle trees. We call  $v_L$  the **left child of**  $v$  and  $v_R$  **right child of**  $v$ .

**Correct construction.** Recall that  $v$  is correctly constructed if its base commitment  $base$  is a commitment to (2) and its span commitment  $span$  is a commitment to (3). One sees that  $v_L$  is correctly constructed if its base commitment is also a commitment to (2) and its span commitment is a commitment to

$$S_{lbase+1}, \dots, S_{lbase+lspan/2}. \quad (9)$$

Similarly,  $v_R$  is correctly constructed if its base commitment is a commitment to

$$S_0, \dots, S_{lbase+lspan/2} \quad (10)$$

and its span commitment is a commitment to

$$S_{lbase+lspan/2+1}, \dots, S_{lbase+lspan}. \quad (11)$$

This definition also tells us the precise shape of the Merkle tree for the base commitment of a correctly constructed node.

It is easy to see that if  $v_L$  and  $v_R$  are correctly constructed, then so is  $v$ . Conversely, assuming that  $H$  is collision resistant, if  $v$  is correctly constructed, then so are  $v_L$  and  $v_R$ .

**Intuition.** The first implication ( $v_L$  and  $v_R$  correctly constructed implies  $v$  correctly constructed) says the following: to prove the claim corresponding to the parent, it suffices to prove the claims corresponding to both children. The second implication ( $v$  correctly constructed implies  $v_L$  and  $v_R$  correctly constructed) says the following: to disprove the claim corresponding to the parent, it suffices to disprove the claim corresponding to one of the children.

#### 4.2.4 Terminal nodes and proof nodes

We call a node of the form

$$v = (\text{regular}, lbase, 1, base, span) \quad (12)$$

a *terminal node*.

If  $v$  has any children, it must have exactly one child, and that child must be

$$v_P = (\text{proof}, lbase, 1, base, span). \quad (13)$$

**Correct construction.** Clearly, if  $v$  is correctly constructed, then so is  $v_P$ .

**Intuition.** A terminal node  $v$  corresponds to a claim that  $base$  is a commitment to (2) and that  $span$  is a commitment to  $S_{lbase+1}$ . Assuming that the claim regarding  $base$  is true, the presence of the child  $v_P$  in the graph indicates that the one-step state transition from  $S_{lbase}$  to  $S_{lbase+1}$  has been proven to be correct, that is,  $F(S_{lbase}) = S_{lbase+1}$ , which means the claim corresponding to  $v$  is also true.

#### 4.2.5 Position, context, and rivals

For a given regular node  $(\text{regular}, lbase, lspan, base, span)$ , we define its *position* to be  $(lbase, lspan)$ , and we define its *context* to be  $(lbase, lspan, base)$ . We say two distinct regular nodes are *rivals* if their contexts are equal. A node that has no rivals is called *unrivaled*. Note that, by definition, proof nodes are *unrivaled*.

**Intuition.** A rivalry between two nodes corresponds to a particular type of dispute between the corresponding claims. If two nodes  $v$  and  $v'$  are rivals, then their corresponding claims agree with respect to the commitment *base* to the initial sequence (2), but disagree with respect to the commitment *span* to the following sequence (3). If  $v$  and  $v'$  are nonterminal nodes with children, and  $v$  has children  $v_L$  and  $v_R$ , and  $v'$  has children  $v'_L$  and  $v'_R$ , then either  $v_L$  and  $v'_L$  are rivals or  $v_R$  and  $v'_R$  are rivals, but not both (assuming collision resistance) – see full version of the paper for further details [1]. Thus, the dispute between the claims corresponding to  $v$  and  $v'$  can be resolved by resolving the dispute between the claims corresponding to either their left children or their right children.

#### 4.2.6 Some general observations

A given node  $v$  in the protocol graph  $\mathcal{G}$  may have several parents. However, the distance between  $v$  and any root is the same, which we call the *depth* of  $v$ .

We also observe that any two nodes that are children of nonterminal nodes and that have the same position are either both left children or both right children of their respective parents. More generally, the position of any regular node implicitly encodes the complete sequence of left/right steps along any path from the root to that node.

### 4.3 Types of Protocol Moves

There are three types of protocol moves. Each such move will supply some data, and when the L1 protocol processes this data, it will add zero, one, or two nodes to the protocol graph  $\mathcal{G}$ . Whenever a new node or edge is added to  $\mathcal{G}$ , the L1 protocol also records the round number in which it was added.

#### 4.3.1 Root creation

The first type of move in the protocol is *root creation*. Such a move supplies a commitment *span*. The L1 protocol adds to  $\mathcal{G}$  the root node  $r$  as in (4), unless this node already exists in  $\mathcal{G}$ . We say this move *creates the root*  $r$ .

#### 4.3.2 Bisection

The second type of protocol move is *bisection*. Such a move supplies a nonterminal node  $v$  as in (5) in Section 4.2.3, together with commitments  $span_L$  and  $span_R$ . The L1 protocol checks that

- (a)  $v$  is already in  $\mathcal{G}$  and rivaled, (b)  $v$  has no children, and (c) (8) holds,

and if so, adds to  $\mathcal{G}$

- the node  $v_L$  as in (6), unless it is already in  $\mathcal{G}$
- the node  $v_R$  as in (7), unless it is already in  $\mathcal{G}$ , and
- the edges  $v \rightarrow v_L$  and  $v \rightarrow v_R$ .

We say this move *bisects the node*  $v$ . Note that the precondition that  $v$  has no children means that  $v$  has not been previously bisected. Also note that, in principle, a node may be bisected in the same round in which it was added to  $\mathcal{G}$ , so long as the preconditions hold at the moment the bisection move is executed (as we will see, although the adversary is free to do this, the honest party will not).

### 4.3.3 One-step proof

The third and final type of protocol move is *one-step proof*. Such a move supplies a terminal node  $v$  as in (12) and a proof  $\pi$ . The L1 protocol checks that **(a)**  $v$  is already in  $\mathcal{G}$  and rivaled, **(b)**  $v$  has no children, and **(c)**  $\pi$  is a valid proof (see details below), and if so, adds to  $\mathcal{G}$  the node  $v_P$  as in (13) and the edge  $v \rightarrow v_P$ .

We say this move *proves the node*  $v$ . Note that, in principle, a node may be proved in the same round in which it was added to  $\mathcal{G}$  (as we will see, although the adversary is free to do this, the honest party will not).

**Some details on the proof system.** We assume a proof system that is comprised of a commitment scheme  $Com$ , a proof generator  $Prove$ , and a proof verifier  $Verify$ . The proof generator should take as input a state  $S$  and output a proof  $p$ . The proof verifier takes as input  $(h, h', p)$  and outputs **accept** or **reject**, and should always output **accept** on inputs of the form  $(h, h', p)$  where  $h = Com(S)$ ,  $h' = Com(F(S))$ , and  $p = Prove(S)$ . We may state the required *soundness property* for the proof system as follows:

*It should be infeasible for an adversary to construct a state  $S$  along with a triple  $(h, \hat{h}', \hat{p})$ , such that  $h = Com(S)$ ,  $\hat{h}' \neq Com(F(S))$ , and  $Verify(h, \hat{h}', \hat{p}) = \mathbf{accept}$ .*

Note that the proof  $\pi$  supplied in a proof move must actually include a proof  $p$  as above, as well the commitment  $h = Com(S_{base})$  and a right-most Merkle path  $mp$  for this commitment relative to the root  $base$ . To check the proof  $\pi$ , the L1 protocol validates  $mp$  (relative to  $base$  and  $h$ ) and verifies that  $Verify(h, span, p) = \mathbf{accept}$  – note that for a correctly constructed node, we will have  $span = Com(S_{base+1})$ .

## 4.4 Timers

We are not quite done describing our dispute resolution protocol. However, before going further, some intuition is in order. The ultimate goal of the protocol is to allow both the honest party and the adversary to create root nodes and to make other moves in such a way that the L1 protocol can determine which root node is correctly constructed. Now, one trivial way to do this would be to have the honest party bisect the correctly constructed root, bisect its children, bisect all of their children, and so on, creating  $n$  terminal nodes, and then proving each of these terminal nodes. However, this trivial approach is extremely expensive. Instead, we adopt the following approach. Whenever a node is created and remains unrivalled for a period of time, the L1 protocol will take that as evidence that the claim corresponding to that node cannot (or need not) be proven false – the more time that elapses, the stronger the evidence. The L1 protocol has to then analyze all of this evidence and declare a “winner”, that is, the root nodes that is most likely the correctly constructed one. The honest party will then adopt a “lazy” strategy, and only defend claims (i.e., bisect nodes) that are disputed (i.e., rivaled), and indeed the protocol is designed so that the honest party must defend all disputed honest claims in order to guarantee victory. However, if its claim corresponding to the correctly constructed root remains undisputed for a sufficiently long period of time, no further moves need to be made.

*Throughout the remainder of Section 4.4, we consider a fixed run of the protocol for some number, say  $N$ , of rounds and let  $\mathcal{G}$  be the resulting protocol graph.*

#### 4.4.1 Creation and rival time

Recall that the L1 protocol keeps track of the round in which a given node  $v$  was created, that is, added to the graph  $\mathcal{G}$ . Let us call this the **creation time of  $v$** , denoted  $\text{ct}(v)$ , defining  $\text{ct}(v) := \infty$  if  $v$  was never created throughout the protocol execution.

Let us call the round in which a regular node  $v$  becomes rivaled its **rival time**, denoted  $\text{rt}(v)$ . More precisely,  $\text{rt}(v)$  is defined to be the first round in which both  $v$  and a rival of  $v$  appear in  $\mathcal{G}$ , defining  $\text{rt}(v) := \infty$  if  $v$  was never created or was created but never rivaled throughout the protocol execution. Clearly,  $\text{rt}(v) \geq \text{ct}(v)$ .

#### 4.4.2 Local timers

For any regular node  $v$  and round number  $t = 1, \dots, N$ , we define the **local timer of  $v$  as of round  $t$** , denoted  $\lambda_v(t)$ , to be the number of rounds in which  $v$ , as of round  $t$ , has remained unrivaled since its creation. Formally, we define

$$\lambda_v(t) := \max(\min(t, \text{rt}(v)) - \text{ct}(v), 0),$$

where the usual rules governing infinity arithmetic are used.

The following is an equivalent and perhaps more intuitive characterization of  $\lambda_v(t)$ :

- if  $v$  was created in round  $t$  or later, then  $\lambda_v(t) = 0$ ;
- otherwise, if  $v$  was unrivaled as of round  $t - 1$ , then  $\lambda_v(t) = 1 + \lambda_v(t - 1)$  and we may say “ $v$ ’s local timer ticks in round  $t$ ”;
- otherwise,  $\lambda_v(t) = \lambda_v(t - 1)$  and we may say “ $v$ ’s local timer does not tick in round  $t$ ”.

We also define the local timer for a proof node  $v$  as follows:

$$\lambda_v(t) := \begin{cases} 0 & \text{if } \text{ct}(v) > t, \\ \infty & \text{otherwise.} \end{cases}$$

Note that throughout the paper, whenever we say something happens “as of round  $t$ ”, we mean “after executing all moves in round  $t$ ”.

#### 4.4.3 Bottom-up timers

Recall that the L1 protocol keeps track of the round in which any given node is added to the graph  $\mathcal{G}$ . For any node  $v$  and round number  $t = 1, \dots, N$ , let us define  $\text{Child}_v(t)$  as the set of children of  $v$  as of round  $t$ . We define the **bottom-up timer of  $v$  as of round  $t$** , denoted  $\beta_v(t)$ , recursively as follows:

$$\beta_v(t) := \lambda_v(t) + \begin{cases} \min(\{\beta_w(t) : w \in \text{Child}_v(t)\}), & \text{if } \text{Child}_v(t) \neq \emptyset; \\ 0, & \text{otherwise.} \end{cases}$$

In other words:

- if  $v$  was created later than round  $t$ , then  $\beta_v(t) = 0$ ;
- otherwise, if  $v$  has no children as of round  $t$ , then  $\beta_v(t) = \lambda_v(t)$ ;
- otherwise,  $\beta_v(t)$  is the sum of  $\lambda_v(t)$  and  $\min_w \beta_w(t)$ , where the minimum is taken over all children  $w$  of  $v$  as of round  $t$ .



#### 4.4.4 Winners

We are finally in a position to define the condition under which the L1 protocol declares a winner. To this end, we introduce a parameter  $T$ , which we call the *confirmation threshold*. We say a root node  $r$  in the protocol graph is *confirmed in round  $t$*  if  $\beta_r(t) \geq T$ . Suppose that  $t^*$  is the first round in which any root node is confirmed. If there is a unique root node  $r^*$  confirmed in round  $t^*$  then  $r^*$  *is declared the winner*; otherwise, “*none*” *is declared the winner*.

#### 4.4.5 Paths in the protocol graph

To better understand the role of bottom-up timers in the protocol, and the rule for confirming root nodes, it is helpful to introduce some additional notions (which will also be useful in the analysis of the protocol).

A *path*  $P$  is a sequence of nodes  $(v_0, \dots, v_{q-1})$  in  $\mathcal{G}$  such that  $\mathcal{G}$  includes the edges  $v_0 \rightarrow v_1 \rightarrow \dots \rightarrow v_{q-1}$ . We define the *length of  $P$*  to be  $q$ . We define the *weight of  $P$*  to be  $\omega_P := \sum_{i=0}^{q-1} \lambda_{v_i}(N)$ . We say  $P$  is a *complete path* if it is nonempty and  $v_{q-1}$  has no children in  $\mathcal{G}$ .

Note that in defining paths, path weights, and complete paths, we are looking at the state of affairs as of round  $N$ , the last round of execution that led to the creation of the protocol graph  $\mathcal{G}$ . In particular, path weights are defined in terms of local timers as of round  $N$ .

We can now characterize bottom-up timers as of round  $N$  in terms of path weights. Specifically, for any node  $v$  in  $\mathcal{G}$ , we have

$$\beta_v(N) = \min_P \omega_P, \quad (14)$$

where the minimum is taken of all complete paths  $P$  starting at  $v$ . It follows that for any given nonnegative integer  $W$ , we have:

$$\beta_v(N) \geq W \text{ if and only if every complete path starting at } v \text{ has weight at least } W. \quad (15)$$

### 4.5 The honest strategy

So far, we have described the logic of the L1 smart contract that acts as a “referee” to ensure that all moves are legal and to declare a winner. However, we have yet to describe the (offchain) logic of the honest party. We do that here.

#### 4.5.1 The honest party’s initial move

We assume that the honest party has the states  $S_0, S_1, \dots, S_n$  and begins by computing the Merkle tree whose leaves are the commitments to  $S_1, \dots, S_n$ . Let  $span$  be the root of this Merkle tree. The honest party submits a root creation move in round 1 using this value  $span$ . This is the only move that the honest party submits in round 1. This move, when executed, will add the honest root  $r^\dagger$  (which we defined in Section 4.4.5 as the correctly constructed root node) to the protocol graph.

#### 4.5.2 The honest party’s subsequent moves

Now suppose the protocol has executed rounds  $1, \dots, t$  and no winner has been declared as of round  $t$ . Consider the protocol graph  $\mathcal{G}$  as of round  $t$  (which the honest party can compute for itself). Recall the confirmation threshold parameter  $T$  introduced in Section 4.4.4 and the notions of paths and path weights introduced in Section 4.4.5. The honest party submits moves in round  $t + 1$  as follows:

For each complete path  $P$  in  $\mathcal{G}$  which starts at  $r^\dagger$  and has weight less than  $T$ :

- if  $P$  ends in a node  $v$  that is rivaled, then
  - if  $v$  is a nonterminal node, the honest party will submit a move to bisect  $v$  (if it has not already done so),
  - otherwise,  $v$  must be a terminal node, and the honest party will submit a move to prove  $v$  (again, if it has not already done so).

#### 4.6 Execution Example

In this section we go over an example execution of single-level BoLD (see Fig. 1). Here, we have  $n = 4$ . We assume in this example that the adversary’s base and span commitments are valid commitments to sequences of states, but these sequences of states may be incorrect. For brevity, we write a node as “[ $\dots$ ][ $\dots$ ]”, where first set of brackets encloses the sequence of states committed to in the base commitment, while the second set of brackets encloses the sequence of states committed to in the span commitment.

Initially, the honest party moves to create the honest root. As indicated in part (a) of the figure, this move gets executed in round 5 of the protocol execution. We will annotate each node with a superscript indicating the current value of its local timer, adding a “+” to that value if its timer is still ticking (meaning the node is unrivaled).

We see in part (b) that the adversary executes moves in round 12 to create a rival root node *and* to bisect that node as well. We indicate the rivalry relation between nodes using dashed lines. While the honest root’s local timer accumulated 7 rounds, it is now stopped because it is rivaled. Seeing that the honest root is rivaled, the honest party submits a move to bisect it. (The reader should note that in each part, new elements in the protocol graph, both nodes and edges, are highlighted in red.)

We see in part (c) that the honest party’s bisection is executed in round 15. Note that the left child of the honest root is actually identical to a node created by the adversary in round 12. This illustrates how nodes can come to have multiple parents. While the left child of the honest root is unrivaled, its right child is rivaled. So the honest party submits a move to bisect that right child. (The reader should note that the edges of the honest tree are highlighted with thicker arrows.)

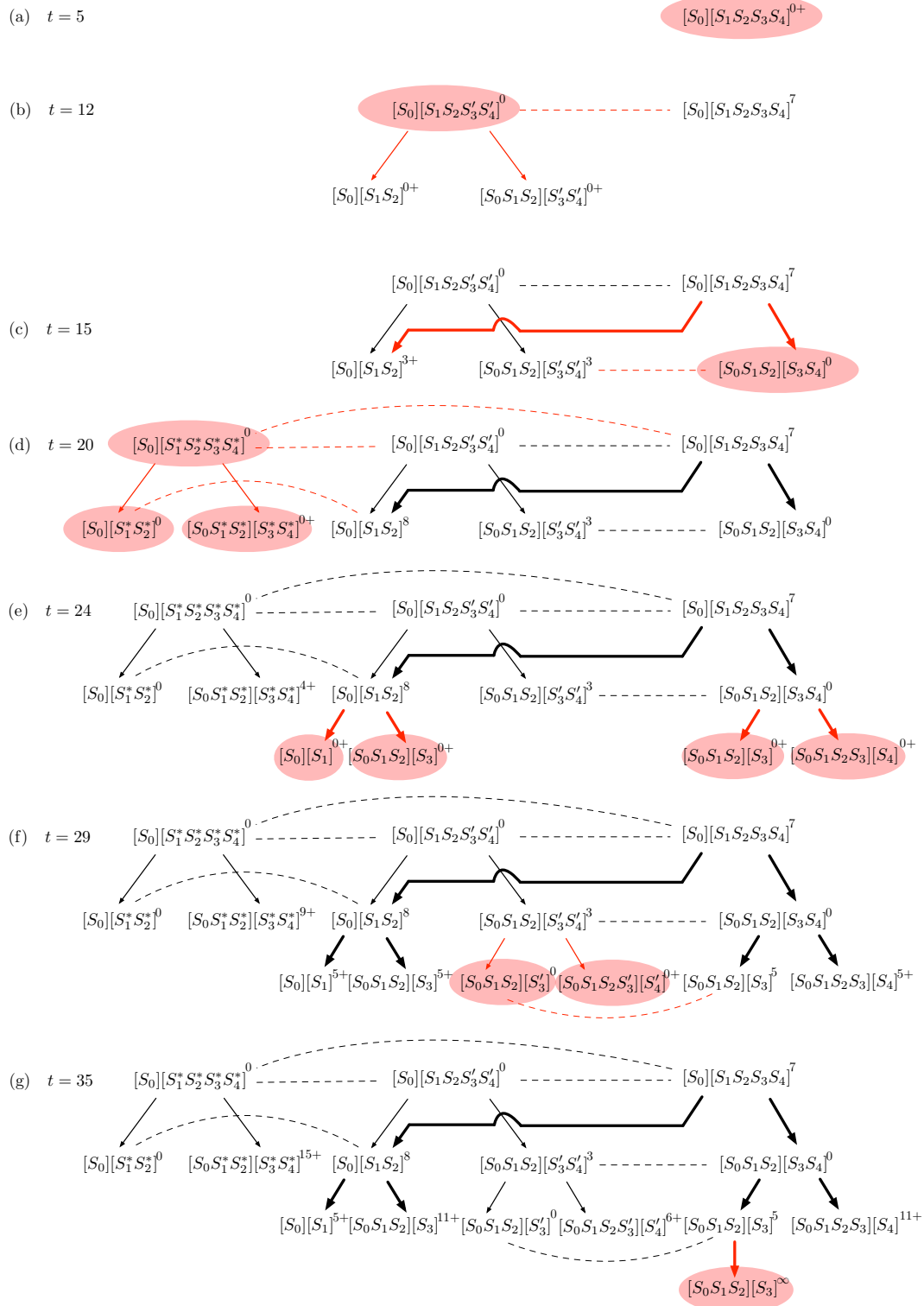
We see in part (d) that in round 20, the honest party’s submitted bisection move has not yet been executed. Instead, the adversary is able to execute moves to create another rival root *and* to bisect that node as well. This bisection creates a node that rivals the left child of the honest root. Seeing that this node is rivaled, the honest party submits a move to bisect it. So now there are two bisection moves submitted by the honest party that are “in flight”.

We see in part (e) that in round 24 both these bisection moves are executed.

We see in part (f) that in round 29, the adversary bisects a node that creates a rival of one of the honest node’s created in round 24. That node, denoted “[ $S_0S_1S_2$ ][ $S_3$ ]” in the figure, is a terminal node. As such, the honest party submits a move to prove that node.

We see in part (g) that in round 35, the proof move submitted by the honest party is executed. This created a proof node, which we also write as “[ $S_0S_1S_2$ ][ $S_3$ ]”, but one sees that its local timer is  $\infty$ .

At this point, if no other moves are made by the adversary, the timers on all of the leaves in the honest tree that are regular nodes will continue to tick. At the same time, for both adversarial roots, there is at least one path along which all timers are stopped (while the local timer on the node denoted “[ $S_0S_1^*S_2^*$ ][ $S_3^*S_4^*$ ]” will continue ticking and remain the largest local timer in the graph, it will not help the adversary confirm an adversarial root). Thus, so long as the confirmation threshold is high enough, the honest root will eventually be confirmed while the adversarial roots will not be.



■ **Figure 1** An example execution.

## 4.7 Main result

We first recall the various parameters introduced so far: the nominal delay bound  $\delta$  (see Section 3), the censorship budget  $C_{\max}$  (see Section 3), the length of a computation  $n = 2^{k_{\max}}$  (see Section 4.1), and the confirmation threshold  $T$  (see Section 4.4.4).

Given the above, we define

$$N^* := T + C_{\max} + (\delta + 1)(k_{\max} + 2), \quad (16)$$

The main result of this section is:

► **Theorem 1.** *Assume the hash function  $H$  used to build Merkle trees is collision resistant. Consider an execution of the protocol using the honest strategy and an arbitrary (efficient) adversary. Assume that  $C_{\max} + (\delta + 1)(k_{\max} + 2) < T$ . Then (with overwhelming probability) the honest root will be declared the winner at or before round  $N^*$ .*

The full proof of Theorem 1 can be found in the full version of the paper [1]. Below we provide an informal proof overview.

**Proof Overview.** Recall that the protocol declares some root as a winner only when the weight of every complete path starting at that root (equal to the sum of local timers of the nodes along the path) reaches the confirmation threshold. Thus, to prove Theorem 1 it suffices to prove two main claims: the first asserts that by round  $N^*$  the weight of every complete path starting at the honest root will surpass the threshold. The second asserts that by that round every adversarial root will have some complete path starting from it with weight strictly less than the threshold.

The honest strategy is designed to achieve exactly this – all its moves aim to extend complete paths starting at the honest root to allow them to accumulate more weight. These honest moves serve the second goal as well, since the nodes created by them rival nodes on paths starting at adversarial roots – this in turn prevents these paths from accumulating weight. The honest party has advantage in this game in the long run, since it is always able to provide one-step proofs to regular terminal nodes that are descendants of the honest root.

A bit more formally, in order to prove the first claim we observe that at any round of the protocol, if a complete path starting at the honest root does not accumulate more weight, it must be the case that all its nodes are rivaled. But in this case, the honest party will submit a (legal) move to extend the path by either bisecting or proving the last node in the path. Of course, once that move is executed the adversary can immediately submit a counter-move to rival the new node in the path, preventing the local timer of the new node from ticking by doing so. At this point the honest party will submit another move to extend the path, followed by another adversary counter-move and so on. This process must end at some point since the maximum depth of a path is bounded by  $k_{\max} + 1$ , and once a proof node has been added to a path, its weight becomes  $\infty$  by definition. The number of rounds  $N^*$  is chosen to account for all steps in this process and also to account for possible rounds in which the honest party's moves were delayed or censored.

To see why the second claim holds, consider any adversarial root  $r$  at round  $N^*$ . A first observation is that  $r$  rivals the honest root  $r^\dagger$  (which must exist in the protocol graph by this round). Another observation is that for any two nonterminal rival nodes, both with left and right children nodes, either both left children are rivals or both right children are rivals. An iterative application of this observation shows that there must exist two complete paths  $P$  and  $P^\dagger$  starting at  $r$  and  $r^\dagger$  (respectively), such that every node in the shorter path of the

two has a matching rival node in some prefix of the longer path. We then make the crucial observation that at any given round, only one node in the union of both paths can have its local timer tick. It follows that the sum of weights of both paths is bounded by the number of elapsed rounds since the beginning of the protocol. However, the first claim above gives a lower bound on the weight of  $P^i$ . The combination of these gives us the desired upper bound on the weight of  $P$ , which proves the second claim.

## 4.8 A Practical Implementation

As a practical matter, our protocol as given requires too much from the L1 protocol. Specifically, we are asking the L1 protocol to continuously track all of the local and bottom-up timers round by round, which would be prohibitively expensive. A simple change to the protocol and the honest party’s strategy can fix this. The idea is to use “lazy evaluation” that computes bottom-up timer estimates “on demand”. These estimates will always be no more than the true value of the timer. A detailed description and analysis of the modified protocol is deferred to the full version of the paper [1].

## 5 Multi-level BoLD

Single-level BoLD has a number of desirable properties, but for some use cases the offchain compute cost of the honest party may be excessive, because of the amount of hashing required by the honest strategy. For example, if  $n = 2^{55}$ , which is plausible for a dispute in Arbitrum, the adversary will need to compute  $2^{55}$  state commitments, each of them a Merkle hash of a virtual machine state, and then do about  $2^{55}$  additional hashes to build the Merkle tree. This much hashing may be too time-consuming in practice.

An alternative is to use BoLD recursively. For example, we might execute BoLD using the iterated state transition function  $F' = F^{2^{25}}$ , thereby narrowing the disagreement with the adversary to one iteration of  $F'$  which is equivalent to  $2^{25}$  iterations of  $F$ . A recursive invocation of BoLD over those iterations of  $F$  would then narrow the disagreement down to a single invocation of  $F$  which could then be proven using the underlying proof system.

The (realized) hope is that if  $n = 2^{55}$  and there are  $N_A$  adversarial roots, then the honest party will have to do one iteration of BoLD with a “sequence length” of  $n_2 = 2^{30}$  and a “stride” of  $\Delta_2 = 2^{25}$ , then at most  $N_A$  “sub-challenge” iterations of BoLD, each with a “sequence length” of  $n_1 = 2^{25}$  and a “stride” of  $\Delta_1 = 1$ . For realistic values of  $N_A$ , this requires much less hashing than single-level BoLD.

The *multi-level BoLD* protocol generalizes this idea of applying single-level BoLD recursively, to support more than two levels. To do this, we extend the single-level protocol by introducing two new elements: refinement nodes and refinement moves. A refinement move takes a node representing a claim about a single step of the iterated transition function at a particular level – this would be a deepest node in that level, which we also call a *terminal node* – and creates a child node representing the same computation but according to the refined transition function, in a new, deeper level of the protocol. This new child node is called a refinement node, and is analogous to a root of this next level.

Refinement moves are different than the bisection and proof moves in the sense that the protocol cannot be directly convinced that the newly created refinement node is correctly constructed (assuming the correct construction of its parent) – in the same way that the single-level protocol cannot be directly convinced that some root is correctly constructed. To deal with this, the protocol does not limit the amount of refinement nodes created from a

single parent terminal node, and these nodes compete with each other in a recursive execution of the protocol. Essentially, the goal of this recursive competition is to convince the protocol which of the refinement nodes is the correctly constructed one.

The transition to multiple levels requires a significant change to the timer scheme. Specifically, the bottom up timer of a node is defined exactly as in the single-level version – except for terminal nodes at any level, for which it is defined as the local timer of that terminal node plus the *maximum* of the bottom-up timers of all of its children refinement nodes. The reason for using maximum instead of minimum here is exactly the same as the reason that the winner root is the one with the highest value bottom-up timer (and not the lowest) – this definition ensures that the unique correctly constructed refinement node, presumably the one with the highest bottom-up timer, will be the one to contribute to the conviction that its parent is correctly constructed as well.

We then prove, as we did in the single-level version, that as long as the confirmation threshold is set high enough, the honest root will be declared winner before any adversarial root. Further details of the extended protocol are deferred to the full version of the paper [1].

## **6 Gas, staking, and reimbursement in BoLD**

In this section, we summarize our analysis of the gas costs incurred by the honest party when executing the BoLD protocol. We introduce a staking requirement for participation and give an overview of how staking can be used to reimburse the honest party for its efforts, and to mitigate against resource exhaustion attacks. For further details, see full version of the paper [1].

### **6.1 Single-Level BoLD**

In single-level BoLD, we require parties creating a root node to place a *stake* (locking up funds on the parent chain) when doing so. When a root node is declared a winner, the stake for the winner is reimbursed and the stake for the other root nodes is confiscated. In the event of a challenge, some stakes will be confiscated; these can be used to reimburse honest parties for their gas costs incurred while running the protocol.<sup>3</sup> The value of the stake –  $S$  – should be set high enough to ensure that confiscated stakes are sufficient to reimburse the gas costs incurred by honest parties running the protocol. However, it should probably be set much higher than this, to discourage an adversary from making any challenge at all, which delays confirmation of the honest root, and to mitigate against resource exhaustion attacks.

In order to reimburse honest parties for their gas costs, it suffices to set  $S$  at least as large as a certain fixed gas cost,  $G$ , which we can calculate (this is essentially the cost of bisecting down a single path from the honest root to a terminal node and then making one proof move).

To reason about resource exhaustion, we define the notion of a *resource ratio* or *griefing ratio* ( $\rho$ ). This is (as the name suggests) a ratio, whose numerator is the total staking and gas cost paid by the adversary to mount an attack on BoLD, and whose denominator is the total gas and staking cost paid by the honest party in the course of responding to that

---

<sup>3</sup> If there is no challenge, BoLD does not have a built-in mechanism to refund the gas cost of the party who creates the confirmed root; compensation for this party will need to come from some external source.

attack. The denominator does not include the staking and gas cost to create the honest root, since the honest party must bear this cost regardless of what the adversary does to attack the protocol (indeed, the honest party must pay this cost even if there is no challenge).

A tradeoff exists between the size of stake and the griefing ratio obtained from using that size stake. A larger stake means that the capital cost for an honest party to create an honest root is higher, but also leads to a larger resource ratio, meaning it is proportionally more expensive for the adversary to impose additional costs on the honest party.

To achieve a target resource ratio  $\rho$ , we show that it suffices to set  $S$  to, roughly,  $\rho \cdot G'$ , for a fixed amount of gas  $G'$ .

For further details, see full version of the paper [1].

## 6.2 Multi-Level BoLD

In multi-level BoLD, we require stakes on refinement edges as well as root edges. The amount of stake required can differ between levels, but at any given level, root or refinement edges at that level all require the same amount of stake. That is, for each level  $\ell$ , we have a parameter  $S_\ell$ , giving the required stake to create a root or refinement node at that level. The most effective staking schemes involve these stakes being larger at higher levels; that is,  $S_1 \leq \dots \leq S_L$ .

As in single-level BoLD, in order to reimburse honest parties for their gas costs, it suffices to set each  $S_\ell$  at least as large as a certain fixed gas cost,  $G_\ell$ , which we can calculate.

In order to obtain a target resource ratio of  $\rho$ , it turns out that we need exponentially escalating stakes. That is, for some fixed amount of gas  $G'$ , we set  $S_1$  to  $G'$ , and then for  $1 \leq \ell < N$ ,  $S_{\ell+1}$  will be set to (roughly)  $\rho \cdot S_\ell$ . This is why it is important to limit the number of levels in multi-level BoLD: stakes need to increase exponentially in the number of levels in order to maintain the same griefing ratio.

We have also explored an alternate staking regime, which (roughly speaking) gives an “asymptotically unbounded” resource ratio. Intuitively, in this regime, rather than being bounded by a constant, the ratio of the adversary’s cost to the honest party’s cost tends toward infinity as the honest party’s cost grows. However, as the number of levels grows, the rate at which this ratio tends toward infinity becomes slower, and does so very quickly. This approach is therefore only useful for single-level BoLD and 2-level BoLD (i.e., multi-level BoLD with  $L = 1$  or  $L = 2$ ).

For further details, see full version of the paper [1].

---



### References

- 1 Mario M. Alvarez, Henry Arneson, Ben Berger, Lee Bousfield, Chris Buckland, Yafah Edelman, Edward W. Felten, Daniel Goldman, Raul Jordan, Mahimna Kelkar, Akaki Mamageishvili, Harry Ng, Aman Sanghi, Victor Shoup, and Terence Tsao. Bold: Fast and cheap dispute resolution. *CoRR*, 2024. [arXiv:2404.10491](https://arxiv.org/abs/2404.10491).
- 2 Harry A. Kalodner, Steven Goldfeder, Xiaoqi Chen, S. Matthew Weinberg, and Edward W. Felten. Arbitrum: Scalable, private smart contracts. In William Enck and Adrienne Porter Felt, editors, *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*, pages 1353–1370. USENIX Association, 2018. URL: <https://www.usenix.org/conference/usenixsecurity18/presentation/kalodner>.





# CFT-Forensics: High-Performance Byzantine Accountability for Crash Fault Tolerant Protocols

Weizhao Tang  

Carnegie Mellon University, Pittsburgh, PA, USA

Peiyao Sheng  

University of Illinois Urbana-Champaign, IL, USA

Ronghao Ni  

Carnegie Mellon University, Pittsburgh, PA, USA

Pronoy Roy 


Carnegie Mellon University, Pittsburgh, PA, USA

Xuechao Wang  

Hong Kong University of Science and Technology, Guangzhou, China

Giulia Fanti  

Carnegie Mellon University, Pittsburgh, PA, USA

Pramod Viswanath  

Princeton University, NJ, USA

---

## Abstract

Crash fault tolerant (CFT) consensus algorithms are commonly used in scenarios where system components are trusted – e.g., enterprise settings and government infrastructure. However, CFT consensus can be broken by even a single corrupt node. A desirable property in the face of such potential Byzantine faults is *accountability*: if a corrupt node breaks the protocol and affects consensus safety, it should be possible to identify the culpable components with cryptographic integrity from the node states. Today, the best-known protocol for providing accountability to CFT protocols is called PeerReview; it essentially records a signed transcript of all messages sent during the CFT protocol. Because PeerReview is agnostic to the underlying CFT protocol, it incurs high communication and storage overhead. We propose CFT-Forensics, an accountability framework for CFT protocols. We show that for a special family of *forensics-compliant* CFT protocols (which includes widely-used CFT protocols like Raft and multi-Paxos), CFT-Forensics gives provable accountability guarantees. Under realistic deployment settings, we show theoretically that CFT-Forensics operates at a fraction of the cost of PeerReview. We subsequently instantiate CFT-Forensics for Raft, and implement Raft-Forensics as an extension to the popular nuRaft library. In extensive experiments, we demonstrate that Raft-Forensics adds low overhead to vanilla Raft. With 256 byte messages, Raft-Forensics achieves a peak throughput 87.8% of vanilla Raft at 46% higher latency (+44 ms). We finally integrate Raft-Forensics into the open-source central bank digital currency OpenCBDC, and show that in wide-area network experiments, Raft-Forensics achieves 97.8% of the throughput of Raft, with 14.5% higher latency (+326 ms).

**2012 ACM Subject Classification** Security and privacy → Distributed systems security; Networks → Security protocols

**Keywords and phrases** CFT Protocols, forensics, blockchain

**Digital Object Identifier** 10.4230/LIPIcs.AFT.2024.3

**Related Version** *Full Version*: <https://arxiv.org/abs/2305.09123> [57]

## Supplementary Material

*Software (Source Code)*: <https://github.com/proy-11/NuRaft-Forensics.git>  
archived at `swh:1:dir:7031137a1ba7c969f4b65c8c8b92bbcaaf10d9f3`

*Software (Source Code)*: <https://github.com/WeizhaoT/Raft-Forensics-Simulator>  
archived at `swh:1:dir:e86133ad17b9089701023d24e3db7cce362499d9`

**Funding** *Weizhao Tang, Ronghao Ni, Pronoy Roy and Giulia Fanti*: This work was supported in part by the National Science Foundation under grants CNS-2325477, CIF-1705007, and CCF-2338772, and the Air Force Office of Scientific Research under award number FA9550-21-1-0090. We also thank Chainlink Labs, Ripple Labs, and IC3 industry partners for their generous support, as well as Bosch, the Sloan Foundation, Intel, and the CyLab Secure Blockchain Initiative.

*Xuechao Wang*: This work is supported in part by a gift from Stellar Development Foundation and



© Weizhao Tang, Peiyao Sheng, Ronghao Ni, Pronoy Roy, Xuechao Wang, Giulia Fanti, and Pramod Viswanath;

licensed under Creative Commons License CC-BY 4.0

6th Conference on Advances in Financial Technologies (AFT 2024).

Editors: Rainer Böhme and Lucianna Kiffer; Article No. 3; pp. 3:1–3:25

Leibniz International Proceedings in Informatics



Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

by the Guangzhou-HKUST(GZ) Joint Funding Program (No. 2024A03J0630).

*Pramod Viswanath*: This work is supported in part by NSF CNS-2325477, ARO W911NF2310147 and C3.AI.

**Acknowledgements** We wish to thank Chris Meiklejohn and Heather Miller for their valuable insights and advice on this project. We also thank Sam Stuewe and the MIT Digital Currency Initiative for their feedback and insights regarding integration with OpenCBDC and applications to central bank digital currency.

## 1 Introduction

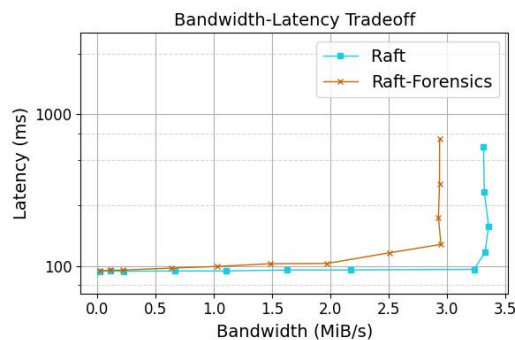
In the theory and practice of distributed systems, crash fault tolerance plays a central role [42]. Crash fault tolerant (CFT) protocols allow a system to come to consensus on a log of events even in the presence of nodes that may crash, but otherwise follow the protocol [58, 47, 24, 30]. CFT systems are widely deployed in enterprise systems and support various high-profile services [27, 10, 5, 30, 21]. For example, prevalent systems like etcd [18], CockroachDB [55] and Consul [28] employ CFT protocols like Raft [47]. CFT protocols are also widely-used in security-sensitive critical infrastructure [49, 27], including prospective Central Bank Digital Currencies (CBDCs) [41, 44].

CFT protocols provide theoretical correctness guarantees under the assumption that at least a certain fraction of nodes follow the protocol, and remaining nodes may suffer from crashes. However, these assumptions can be broken in practice. For instance, an agent could be Byzantine, meaning that it can misbehave arbitrarily, e.g., by delaying or tampering with messages. In such cases, consensus can be trivially broken.

One possible solution is to replace the CFT protocol with a *Byzantine fault tolerant* (BFT) protocol, which guarantees consensus under not only crash faults, but also under Byzantine faults [9, 40, 7, 2, 62, 20, 22, 25]. This is a viable solution, though swapping out consensus protocols may be impractical for organizations that have already built infrastructure around a particular CFT system.

In this paper, we explore a complementary approach to managing Byzantine faults: *accountability*. That is, in the case of Byzantine faults in a CFT protocol, can an auditor with access to locally-stored protocol states identify *which* node(s) were responsible for the misbehavior, with cryptographic guarantees? In particular, we want to provide this guarantee by making minimal changes to an existing system and protocol, rather than completely replacing the consensus mechanism.

Accountability for BFT protocols has been studied systematically very recently, both as an intrinsic attribute of existing protocols [50, 45, 46] and as an important feature in the design of new protocols [6, 54, 11, 51]. However, there is comparatively little work on CFT protocols that incorporate accountability for Byzantine faults [26, 24]. An important prior work called PeerReview tackled this problem in the context of general CFT protocols [26]. PeerReview works by producing a signed transcript of every message that is sent in the protocol. Being a general-purpose protocol, it does not always achieve competitive performance with the underlying CFT protocol (details in §6.2). Hence, to our knowledge, existing work on accountability for CFT protocols either: (1) is very general, and thus incurs high performance overhead when applied to specific CFT protocols (i.e., PeerReview [26]), and/or (2) does not include a full implementation-based evaluation to measure the practical effect of accountability [26, 24].



■ **Figure 1** Bandwidth-latency tradeoffs of Raft vs Raft-Forensics over 4 nodes at message size of 256 Bytes.

Our goal in this work is to design a practical accountability framework that incurs low communication and storage overhead by exploiting the structure of the underlying protocol, unlike PeerReview. Crucially, despite exploiting protocol structure, we want the framework to be broadly applicable to common CFT protocols and backwards-compatible with existing systems. To this end, our contributions are threefold:

- **Accountably-Safe Consensus:** We first formally define a subclass of CFT protocols called forensics-compliant protocols, which includes two of the most widely-used CFT protocols in use today: Raft [47] and Paxos [34, 29]<sup>1</sup>. Intuitively, the defining feature of this class is that its protocols cycle between two phases: log replication and leader election, and each phase satisfies some formal properties (defined in §4). We then propose CFT-Forensics, a lightweight modification to forensics-compliant CFT protocols that *provably* guarantees to expose at least one node that committed Byzantine faults when consensus is violated. Note that we cannot guarantee to detect more than one Byzantine node, as only one malicious node is needed to break CFT consensus; however, for certain classes of attacks involving multiple Byzantine nodes, we are able to detect multiple misbehaving nodes.
- **Theoretical Efficiency Comparison:** We theoretically analyze the communication and computational overhead of CFT-Forensics compared to the most relevant prior work in this space, PeerReview. We show that CFT-Forensics has (amortized) vanishing storage overhead compared to the baseline protocol in practical scenarios, while PeerReview has overhead that grows linearly with the logs. In addition, during log replication, the communication overhead of CFT-Forensics is 58% lower than PeerReview.
- **Empirical Performance Evaluation on Raft:** We implement Raft-Forensics, an instantiation of CFT-Forensics for the Raft protocol. Our implementation is built on a fork of nuRaft, a popular C++ implementation of Raft. We evaluate its performance compared to Raft, both in benchmark experiments and in a downstream application – specifically, OpenCBDC [41] – an open-source central bank digital currency (CBDC) implementation that uses nuRaft. In benchmark experiments, we observe in Fig. 1 that CFT-Forensics achieves performance close to vanilla Raft (experimental details in §7). For instance, in end-to-end experiments, it achieves a maximum throughput that is 87.8% the maximum throughput of vanilla Raft, at 46% higher confirmation latency (44 ms). In our OpenCBDC experiments over a wide-area network, Raft-Forensics achieves 2.2% lower throughput at 14.4% higher latency (326 ms) than vanilla Raft.

<sup>1</sup> For notational brevity, we use the name “Paxos” to refer to variants of the Paxos algorithm that are sometimes referred to as multi-Paxos to distinguish from the original single-decree Paxos [34, 29].

## 2 Related Work

### CFT protocols

CFT protocols are designed to handle crash faults, where nodes may fail but do not exhibit malicious behavior. Paxos [36] is a foundational CFT protocol, with many variants [38, 34, 15, 35, 4, 37, 43, 58, 29]. Raft [47] is a CFT protocol that aims to provide a more understandable and easier-to-implement alternative to Paxos [58], where HovercRaft [31] further improves its performance. Viewstamped Replication revisited [39] is a CFT protocol that displays elements of both Paxos and Raft. Both Raft [55, 48, 1] and Paxos [8, 5, 53, 14] are widely-used in practice.

### Accountability

Accountability allows protocols to identify and hold misbehaving participants responsible when security goals are compromised [33, 32]. In the context of fault-tolerant protocols, accountability allows a protocol to identify culpable participants when security assumptions are violated and demonstrate their misconduct. Recent works [50, 16] have examined several widely used BFT protocols and assessed their inherent accountability levels without altering the core protocols. In addition, some other works [23, 3, 59] have improved the performance of existing BFT protocols by excluding culpable participants from the consensus with accountability. Since CFT protocols are explicitly designed to handle only crash faults, integrating accountability offers a lightweight enhancement to detect Byzantine actors.

One prior work [24] explored the accountability of the Hyperledger Fabric blockchain, which features a pluggable consensus mechanism. This study conducted a case analysis of incorporating accountability into a Hyperledger Fabric system underpinned by a CFT protocol, Apache Kafka [21] (called Fabric\*). However, this work treats the consensus module as a cluster, offering accountability only at the level of the entire consensus group (not individual nodes within the group). In contrast, we aim to identify and attribute Byzantine faults to individual misbehaving consensus replicas participants. Fabric\* introduces two primary modifications. First, parties must sign every message they send. Second, it enforces a deterministic block formation algorithm to eliminate ambiguity. However, these changes are neither necessary nor sufficient for ensuring accountability in the CFT protocols we study. In addition, Fabric\* does not empirically evaluate their system, whereas we evaluate performance both theoretically and empirically.

PeerReview [26] builds a framework for accountability that applies to general distributed systems. Although it accounts for Byzantine faults in CFT protocols as CFT-Forensics does, it has substantially higher overhead communications and space requirements than CFT-Forensics, which we discuss in §6 in detail. PeerReview requires nodes to audit each other, instead of assuming arbitrarily many central auditors as we do (§4). To address this difference, we disable inter-node auditing in PeerReview, which still incurs substantially higher communication and memory overhead than CFT-Forensics.

## 3 Setup

We study consensus protocols that solve the crash-fault tolerant state machine replication (CFT-SMR) problem over partially synchronous networks. Precisely, we consider a setting with  $n$  servers (also known as nodes) and arbitrarily many clients. For the vanilla CFT-SMR setting, we assume that at most  $f$  out of the  $n$  nodes can suffer *crash failures*, where they

stop working without resuming at an arbitrary and unpredictable moment. Each node  $u$  maintains a state machine  $\text{SM}^u$  and an append-only log list  $\text{logs}$ . We let  $u[i]$  denote the  $i$ -th entry in  $u$ 's  $\text{logs}$ . The goal of CFT-SMR is for the nodes to maintain consistent state machines  $\text{SM}^u$  with each other (Definition 1).  $\text{SM}^u$  maintains a local state  $s$  initialized to  $s_0 = \perp$  and a deterministic function  $\phi$ .  $\text{logs}$  are sequential inputs to  $\text{SM}^u$  generated from client requests, which results in state transition

$$\text{SM}^u.s_i = \text{SM}^u.\phi(\text{SM}^u.s_{i-1}, u[i]), \quad \forall i \in \mathbb{Z}_{>0}.$$

The network is partially synchronous, meaning that there exists a global stabilization time (GST) and a constant time length  $\Delta$ , such that a message sent at time  $t$  is guaranteed to arrive at time  $\max\{\text{GST}, t\} + \Delta$ . GST is unknown to the system designer and is not measurable by any component of the system.

► **Definition 1** (CFT(-SMR) Protocol). *In the setting above, a consensus protocol  $\mathcal{P}$  is  $f$ -CFT(-SMR) if  $f$  nodes can fail by crash, and the following three properties are satisfied.*

1. **Safety:** *If  $E$  is the  $i$ -th entry of a correct node's log, then no other correct node has  $E' \neq E$  at index  $i$ .*
2. **Liveness:** *If a correct client submits a request  $r$ , then eventually all non-faulty nodes will (1) have a log entry  $E$  at index  $i$  handling  $r$  (2) there exists a log entry at all previous positions  $j < i$ .*
3. **Validity:** *Each entry in the log of a correct replica can be uniquely mapped to a command proposed by a client request.*

In the remainder of the paper, we study  $f$ -CFT protocols with  $f = \lfloor (n-1)/2 \rfloor$  and focus on the **boldfaced** safety property. These protocols tolerate  $f$  crash failures, but are typically vulnerable under even one *Byzantine failure*, where a node arbitrarily deviates from the stipulated protocol (§5.1).

We formalize our threat assumptions below.

### 3.1 Threat Model

In addition to the  $f$  nodes with crash failures, we further assume the existence of  $b \geq 1$  nodes that execute Byzantine faults. We assume  $b \leq n-2$  to avoid a trivial problem with at most one honest node. The Byzantine nodes are capable of accessing states of honest nodes and collaboratively determining whether, when, and what to send to every honest node. However, they cannot influence the honest nodes or the communication between them.

#### Auditor

To identify the adversary, we introduce *auditors* in addition to the clients and (server) nodes in the SMR model. Any actor with access to the full states of any node (details in §5.2.1) can be an auditor. Each auditor works independently. If an auditor requests information, honest nodes always provide their information to the auditor; a Byzantine node can respond arbitrarily. Generally, the information is transmitted through the partially synchronous network, so it is guaranteed to arrive at the auditors eventually. The information may also be collected physically, if the entire system is maintained by a centralized party such as a central bank. Any auditor may determine the safety of the system by checking data legitimacy and consistency among the nodes, as a function of the received state information. Our main goal is to define modifications to the consensus protocol and an auditing algorithm that jointly enable an auditor to *eventually* uncover the identity of the adversarial node if the state machine safety property is violated.

We assume the simplest setting with a single trusted auditor who is unable to directly influence the system. Notably, there are various alternative auditing designs. For example, we can introduce additional independent auditors to trade communication complexity for robustness. We can also allow auditors to create checkpoints for the nodes. Since checkpoints influence the system, the auditors should also be coordinated by a distributed consensus for security. We leave the design of a trustless auditing system to future work.

## 3.2 The Accountability Problem

If even a single node is Byzantine, CFT protocols are vulnerable to safety violations (examples in Section 5.1). As a result, we want to identify the party responsible for a safety violation using an auditing algorithm. If such an algorithm exists, we say the protocol has *accountability*.

► **Definition 2 (Accountability).** *Let  $\mathcal{P}$  denote an  $f$ -CFT-SMR consensus protocol.  $\mathcal{P}$  has accountability if there exists a polynomial-time auditing algorithm  $\mathcal{A}$  s.t.*

1.  $\mathcal{A}$  takes the states of  $\mathcal{P}$  as input.
2. If safety (Def. 1) is violated,  $\mathcal{A}$  outputs a non-empty set of nodes and irrefutable proof that each member of the set violated protocol. Otherwise,  $\mathcal{A}$  outputs  $\perp$ .

## 4 Forensics-Compliant Protocols: A Family of CFT Protocols

Modifying an arbitrary CFT-SMR protocol under a general workflow without context can be challenging. To address this, we define a family of CFT-SMR consensus protocols named *forensics-compliant*, which are provably modifiable for accountability under our general framework CFT-Forensics (Def. 2 and Theorem 11). At a high level, a forensics-compliant protocol is leader-based (Property 3). It can be described by a set of *procedures*, which is partitioned into *log replication* and *leader election*<sup>2</sup> with each satisfying necessary properties. Both Raft [47] and Paxos [58], two dominant CFT protocols in practice [29], are forensics-compliant protocols. Additionally, forensics-compliant protocols include Viewstamped Replication revisited [39] and simple variants such as HovercRaft [31].

### 4.1 Setup

We start with an  $f$ -CFT-SMR protocol. In the protocol, each entry in the log has two possible states: *committed* and *uncommitted*. If an entry is committed, the content in the entry will not be changed in the future and can be applied to the state machine. If a prefix in the log is committed, then all entries in the prefix is considered committed. The largest index of committed entries is called *the last commit index*, denoted as `iCommit`.

Let there be global notion of time  $T = [0, \infty)$ , which is unknown to any of the nodes. For simplicity, we let  $u[i]$  denote the  $i$ -th log entry in node  $u$ 's logs, and  $u.\text{loglen}$  denote the length of  $u$ 's logs, which equals the index of  $u$ 's last entry. For a given entry  $E$  with index  $i$ , we say a node  $v$  *owns*  $E$  if  $v[i] = E$ . Furthermore, we let  $E_{i:j}$  denote a sequence of consecutive entries  $\{E_k | k \in [i, j], E_k.\text{index} = k\}$ . Throughout the paper, we use **colored monospace** text to denote protocols and methods that appear in pseudo-code.

<sup>2</sup> We use the terminology of Raft for clarity.

### Leader-Based

Let  $S$  denote the set of nodes with  $|S| = n$ . A forensics-compliant protocol must satisfy the *leader-based property* (Property 3).

► **Property 3** (Leader-Based). *At any time  $t \in T$ , each node  $u \in S$  identifies a leader  $L_u(t) \in \bar{S} \triangleq S \cup \{\perp\}$ . For each  $u$ , there exists an interval partition of  $T = \bigcup_{i=1}^{\infty} [t_{i-1}, t_i)$  and a sequence of nodes  $\{\ell_i \in \bar{S}\}_{i=1}^{\infty}$  where  $t_0 = 0$ , and for all  $i \in \mathbb{Z}_{>0}$ ,*

$$t_{i-1} < t_i, \quad \ell_i \neq \ell_{i+1}; \quad L_u(t) = \ell_i, \forall t \in (t_{i-1}, t_i).$$

*If  $L_u(t) = u$  for all  $t \in [\underline{t}, \bar{t})$ ,  $u$  is called a leader during the leadership  $[\underline{t}, \bar{t})$ . Otherwise, if  $L_u(t) = \ell \notin \{u, \perp\}$ ,  $u$  is called a follower identifying  $\ell$ . Only a leader can propose a log entry. At time  $\underline{t}$  when  $\ell$  starts being a leader, it assigns a unique term to itself which is fixed until it stops being a leader at  $\bar{t}$ . Hence, the term can be regarded as an attribute of a leadership during  $[\underline{t}, \bar{t})$ . For node  $u$  to identify  $\ell$ ,  $u$  must receive a message from  $\ell$  that includes  $\ell$ 's term.  $u$  sets its term equal to  $\ell$ 's term as soon as it starts identifying  $\ell$ .*

*We say there exists a global leader  $\ell \in S$  of term  $\tau$ , if there exists a majority subset  $M \subseteq S$ , such that  $\ell \in M$  and for all  $u \in M$ ,  $u.\text{term} = \tau$ . Since  $M$  is the majority,  $\ell$  must be unique at every time, so global leaderships do not overlap in time. We require that the term of a later global leadership must be strictly greater than that of an earlier one.*

The full protocol consists of *procedures* that are partitioned into the following subprotocols.

- *Log Replication* is the subprotocol that collects all procedures only executed when the host node  $u$  identifies a new leader, i.e.,  $L_u(t) \neq \perp$ .
- *Leader Election* is the subprotocol collecting all the remaining procedures.

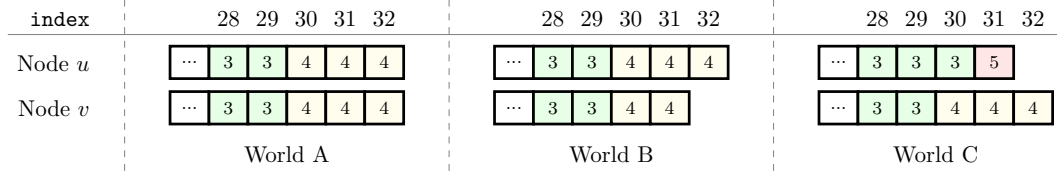
### Log Replication

On the top level, log replication (Alg. 1) has a main procedure `HandleClientRequest` that is triggered when a leader receives a client request. If a node is not running `HandleClientRequest` or involved in an RPC call within the procedure, it cannot create a new log entry or edit its logs and `iCommit`. It has three steps – log entry creation, replication and commitment.

**Creation.** When leader  $\ell$  receives a client request,  $\ell$  creates a corresponding log entry  $E$  and appends it to the log list.  $E$  has 3 attributes – (1) `term`,  $\ell$ 's term; (2) `index`, its index on the log list; and (3) `payload`, which handles the request. We define the *freshness* of a log entry, a log list and a node in Def. 4, and provide an example in Fig. 2.

► **Definition 4** (Freshness). *A log entry  $E$ 's freshness is denoted by the tuple  $(E.\text{term}, E.\text{index})$ .  $E$  is as fresh as entry  $F$  if their freshness tuples are identical.  $E$  is fresher than  $F$  if  $E.\text{term} > F.\text{term}$  or  $E.\text{term} = F.\text{term} \wedge E.\text{index} \geq F.\text{index}$ .  $E$  is strictly fresher than  $F$  if  $E$  is fresher than  $F$ , and  $E$  is not as fresh as  $F$ . In contrast,  $E$  is staler than  $F$  if  $E$  is not strictly fresher than  $F$ . The freshness of a node or its log list is equivalent to that of the log list's last entry.*

**Replication.** The procedure of replication can be described by  $\ell$  calling a single RPC `AppendEntries` for each remaining node. Its eventual outcome is a `AppendEntriesResp` message from each callee, which includes a predicate `accept` that indicates whether the replication is successful. In addition, the RPC must satisfy the replication property:



■ **Figure 2** Examples of node freshness. Each box represents an entry containing the entry's term. The entries are **not** necessarily committed. Each world is a possible result of protocol execution by only honest nodes. In all worlds,  $u$ 's log list is (unstrictly) fresher than  $v$ 's. In World A,  $u$  is as fresh as  $v$ . In only Worlds B and C,  $u$  is *strictly* fresher than  $v$ .

► **Property 5** (Replication). *If a follower  $u$  replicates  $E_{i:j}$  from leader  $\ell$ ,  $u$ 's term must equal  $E_j$ 's term, and it must own  $E_{1:i-1}$ . Formally,  $u.\text{term} = \ell.\text{term} = E_j.\text{term}$  and for all index  $k \leq j$ ,  $u[k] = \ell[k]$ .*

**Commitment.** Once  $\ell$  receives `AppendEntriesResp` messages from  $(n - f - 1)$  followers with `accept=True`,  $\ell$  commits  $E_j$ . Then,  $\ell$  sends a message `InformCommitMsg` including (hash of)  $E_j$  to each remaining node  $u$ , who also commits  $E_j$  if it owns  $E_j$ .

■ **Algorithm 1** Log replication of the forensics-compliant family. In persistent storage, a node maintains `term`, `logs`, `iCommit`, `CC` and `LC`, where `iCommit` is the last commit index. The **red lines and variables** are added in CFT-forensics (§5). See §5.2.1 for relevant definitions and our full paper [57] for the complete algorithm.

---

```

1 Protocol LogReplication(host node  $w$ ):
2   As leader:
3   Procedure HandleClientRequest( $r$ ):
4      $E \leftarrow \text{LogEntry}(\text{term}=\text{term}, \text{index}=\text{loglen} + 1, \text{payload}=\text{Payload}(r))$ 
5      $E.\text{pointer} \leftarrow \text{Hash}(\text{loglen} + 1 || E.\text{payload} || w[\text{loglen}].\text{pointer})$ 
6      $E.\text{stamp} \leftarrow \sigma_w(E.\text{pointer})$ 
7      $E.\text{LC} \leftarrow \text{LC}$ 
8     Append  $E$  to  $w$ 's logs
9      $\text{replicators}, \text{sigs} \leftarrow \{w\}, \{\sigma_w(E.\text{pointer})\}$  // Replication
10    for async  $u \in S - \{w\}$ 
11       $\text{msg} : \text{AppendEntriesResp} \leftarrow \text{async AppendEntries}([E], \dots)$ 
12      if  $\text{msg}.\text{accept}$ 
13        if not verifySig( $\text{msg}.\text{signature}, E.\text{pointer}, u$ )
14          fail exit
15           $\text{sigs} \leftarrow \text{sigs} \cup \{\text{msg}.\text{signature}\}$ 
16           $\text{replicators} \leftarrow \text{replicators} \cup \{u\}$ 
17    Wait until  $|\text{replicators}| \geq n - f - 1$ 
18    if  $E$  not yet committed // Commitment
19      Commit( $E$ )
20       $\text{CC} \leftarrow i || E.\text{pointer} || \text{replicators} || \text{sigs}$ 
21      for  $u \in S - \{w\}$ 
22        Send InformCommitMsg( $E, \text{CC}$ ) to  $u$ 

```

---



## Leader Election

By definition, leader election is the set of procedures that do not belong to Log Replication, where `CandidateMain` is the main procedure. Without running `CandidateMain` or being involved in an RPC within it, a node cannot identify any leader. Only a candidate within `CandidateMain` can edit its logs and `iCommit`. `CandidateMain` consists of three steps: term switching, candidate qualification, and leadership claim.

**Term Switching.** At the beginning, the caller  $\ell$ , also called a *candidate*, updates the term to a greater term, which is exactly the term of  $\ell$ 's leadership as the outcome of `CandidateMain`.

**Candidate Qualification.** This phase is represented by a procedure `Qualification`, which can be completed or *interrupted*. If it is interrupted, `CandidateMain` is also interrupted. Otherwise, it satisfies the election property (Property 6) by necessary communications and state modifications.

► **Property 6 (Election).** *If  $\ell$  completes `Qualification` at term  $\tau$ , there must exist a set of nodes  $V$  where  $|V| \geq n - f$  and  $\ell \in V$ , such that*

1. (*Validity*) After `Qualification`,  $\ell[j].\text{term} \geq \ell[i].\text{term}$  for all  $j > i$ .
2. (*Selection*) For every log entry  $E = \ell[i]$  after `Qualification`, there exists  $u \in V$  such that  $u[i].\text{payload} = E.\text{payload}$ .
3. (*Freshness*) Let  $u^i$  denote an arbitrary node satisfying Selection at index  $i$ . Before `Qualification`, let node  $v$  be freshest among  $u$ . After `Qualification`,  $\ell$ 's log list is no shorter than  $v$ 's and for all  $i \leq \text{length of } v\text{'s log list}$ ,  $u^i[i].\text{term} \geq v[i].\text{term}$ .

**Leadership Claim.** After `Qualification`,  $\ell$  identifies itself as the leader. Then, it sends a `LeadershipClaim` message including its term  $\tau$  to each other node. A recipient  $u$  identifies  $\ell$  as the leader and sets its own term to  $\tau$  if  $\tau$  is greater than  $u$ 's own term; otherwise,  $u$  ignores the message.

## 4.2 Summary

► **Definition 7 (Forensics-Compliant Protocols).** *A forensics-compliant protocol is a leader-based (Property 3)  $f$ -CFT-SMR protocol (Def. 1). The protocol can be partitioned into two subprotocols – log replication (Alg. 1) and leader election (Alg. 2), such that*

- *Log replication is a set of procedures that can only be executed when a node identifies a leader. If a node identifies itself, it handles client requests with `HandleClientRequest`, where the `AppendEntries` RPC must have the replication property (Property 5).*
- *Leader election is the set of all the remaining procedures, including `CandidateMain`, which is a unique procedure that allows a node to start identifying a leader. In `CandidateMain`, the `Qualification` procedure must satisfy the election property (Property 6).*

*In addition, the log list and `iCommit` must not be modified by any procedure that is not mentioned above or explicitly written in the pseudocode.*

► **Proposition 8 (Instances of Forensics-Compliant Protocols).** *Both Raft [47] and Paxos [34, 29] are forensics-compliant.*

■ **Algorithm 2** Leader election of the accountable family. The red lines and phrases are specific to our (unoptimized) CFT-forensics (§5). See §5.2.1 for relevant definitions. See our full paper [57] for the complete algorithm and implementation of `validate`.

---

```

1 Protocol LeaderElection(host node  $w$ ):
2   Procedure CandidateMain(...):
3      $term \leftarrow$  a new, higher term than  $term$  // Term switching
4     Qualification( $term, \dots$ ) // Candidate qualification; abort on failure
5      $r \leftarrow w || term || w[loglen].term || loglen || w[loglen].pointer$ 
6      $votes \leftarrow \{w : \sigma_w(\text{Hash}(r))\}$ 
7     for async  $u \in S - \{w\}$ 
8        $msg \leftarrow$  Call RPC RequestVote( $u, term$ )
9       if verifySig( $msg.signature, \text{Hash}(r), u$ )
10        |  $votes[u] \leftarrow msg.signature$ 
11    Wait for  $votes.size \geq n - f$ 
12     $LC \leftarrow r || votes.keys || votes.values$ 
13    for  $E \in logs$ 
14      | if  $E.term = term$ 
15      | |  $E.LC \leftarrow LC$ 
16     $L_w \leftarrow w$  // Leadership Claim
17    for  $u \in S - \{w\}$ 
18      | Send LeadershipClaim( $term, LC$ ) to  $u$ 
19  Response HandleClaimLeadershipMsg( $\ell, msg$ ):
20    if  $msg.term \leq term$ 
21      | fail exit
22     $L_w, term \leftarrow \perp, msg.term$ 
23    if not validate( $msg.LC$ )
24      | fail exit
25     $L_w \leftarrow \ell$ 
26    ... // Protocol-specific state updates

```

---

### Proof (Raft)

Raft is originally designed in a very similar philosophy to the forensics-compliant family. It is a leader-based (Property 3) SMR solution by design. The Raft consensus algorithm has two components: log replication and leader election. We define the core procedures in our full paper [57].

**AppendEntries:** After a follower receives a list of consecutive log entries (or a single entry), it replicates them if it has the predecessor of the head of the list. Otherwise, it triggers `AppendEntries` recursively to synchronize all uncommitted entries, which guarantees the no-gap property (Property 5).

**Qualification:** A candidate  $\ell$  in Raft asks voters for votes, and a voter only votes if  $\ell$ 's log list is fresher than its own. This ensures  $\ell$  is fresher than  $n - f$  nodes without changing its logs, so `Qualification` RPC satisfies the election property (Property 6).

To summarize, all the RPCs have the required properties, so Raft is forensics-compliant.

### Proof (Paxos)

(Multi-)Paxos is an optimized protocol based on a simple array of basic Paxos. Its description varies from paper to paper, so we adopt the version in [29] which enables a clear comparison to Raft. Both the original Paxos [34] and [29]'s variation are leader-based (Property 3). In Paxos, the log replication procedures are identical to those in Raft. Thus, we focus on the leader election subprotocol (details in our full paper [57]).

Unlike Raft, a Paxos voter  $u$  always votes for a candidate  $\ell$  with a higher term in [Qualification](#). The vote comes with all  $u$ 's entries at  $\ell$ 's uncommitted indices of  $\ell$ . With  $n - f - 1$  such votes, at each uncommitted index,  $\ell$  selects the freshest entry it has ever seen. Hence, [Qualification](#) in Paxos also satisfies the election property (Property 6).

To summarize, Paxos (as described in [29]) is also forensics-compliant.

## 5 CFT-Forensics

Although CFT protocols guarantee safety against crash faults, they are not safety-resilient against even a single Byzantine fault. We first illustrate typical safety attacks. Next, we present CFT-Forensics to endow forensics-compliant protocols with accountability.

### 5.1 Example Attacks

Recall that in §3, we assumed that  $b \in [1, n - 2]$  nodes may behave adversarially. In two examples, we assume  $n = 2f + 1$  is odd for simplicity. We show the capabilities of a single attacker Mallory, and the remaining  $2f$  nodes are evenly partitioned into  $X$  and  $Y$ .

► **Example 9** (Proposer's Attack, or Split-Brains). Let Mallory be a corrupt leader. At the same index, Mallory replicates log entries  $E$  and  $E' \neq E$  to  $X$  and  $Y$ , respectively. At each side, she commits the corresponding entry with a quorum of  $f + 1$  nodes. As a result, the honest nodes in  $X$  and  $Y$  have different committed log entries at the same index.

► **Example 10** (Voter's Attack). Let Mallory be a corrupt voter who has committed entry  $E$  with  $Y$ . Nodes in  $X$ , however, do not own  $E$ . In an election, suppose Carol  $\in X$  who earns all  $f$  votes from  $X$ . When Carol requests vote from Mallory, Mallory votes by simulating itself as a Carol's clone. After being elected, Carol commits  $E' \neq E$  at the same index, which conflicts with any honest node in  $Y$ .

Surprisingly, these two examples almost exhaustively enumerate the types of safety attacks against forensics-compliant protocols (Theorem 11). This is why forensics-compliant protocols can achieve accountability with much simpler modifications than PeerReview.

### 5.2 CFT-Forensics Design Overview

We present CFT-Forensics, a framework that enables accountability for forensics-compliant protocols. Here, we present a basic variant of CFT-Forensics, which adds large overhead compared to vanilla CFT protocols; we provide and analyze an optimized variant in §6.1. We use the convention that for a forensics-compliant protocol  $\mathcal{P}$ ,  $\mathcal{P}$ -Forensics denotes the protocol  $\mathcal{P}$  augmented with CFT-Forensics (e.g., we implement Raft-Forensics in Section 7).

At a high level, CFT-Forensics adds two central data structures to a forensics-compliant protocol: commitment certificates (CCs) and leader certificates (LCs). A CC irrefutably proves that a quorum of nodes have replicated an entry, and an LC proves *which* quorum of nodes agreed to elect a leader. CFT-Forensics requires each log entry to be signed by its proposer, which provides accountability for a split-brains attack (Example 9). It also requires that each voter signs its vote, for which the voter is forced to take responsibility since the vote exists in a CC or an LC, providing accountability for the voter's attack (Example 10).

### Additional Assumptions: Public Key Infrastructure

We assume access to a Public Key Infrastructure (PKI). Each node  $u$  has a pair of private and public keys, where the public key is well known, that is, known by all parties in the system, including other nodes and auditors. Node  $u$  can use its private key to create an unforgeable signature on (the hash of) an arbitrary message  $m$ , denoted by  $\sigma_u(\text{Hash}(m))$ , and the signature can be verified with  $u$ 's public key. A collision-resistant cryptographic hash function  $\text{Hash}$  is known to all parties. Both signing a message and verifying a signature can be executed in time that is polynomial in message size.

#### 5.2.1 Added States

We first explain the new state that is maintained in CFT-Forensics. CFT-Forensics introduces four new categories of states: hash pointer, proposer stamp, leader certificate (LC) and commitment certificate (CC).

■ **Table 1** Attributes of a CC, an LC and a vote request.

Commitment certificate CC	index		pointer	voters	signatures
	$i$		$h_i$	$V$	$\{\sigma_u(h_i)\}_{u \in V}$
Leader certificate LC			req	voters	signatures
			$r$	$V$	$\{\sigma_u(\text{Hash}(r))\}_{u \in V}$
Vote Request	id	term	eterm	end	pointer
	$\ell$	$\ell.\text{term}$	$\tau(< \ell.\text{term})$	$i$	$h$

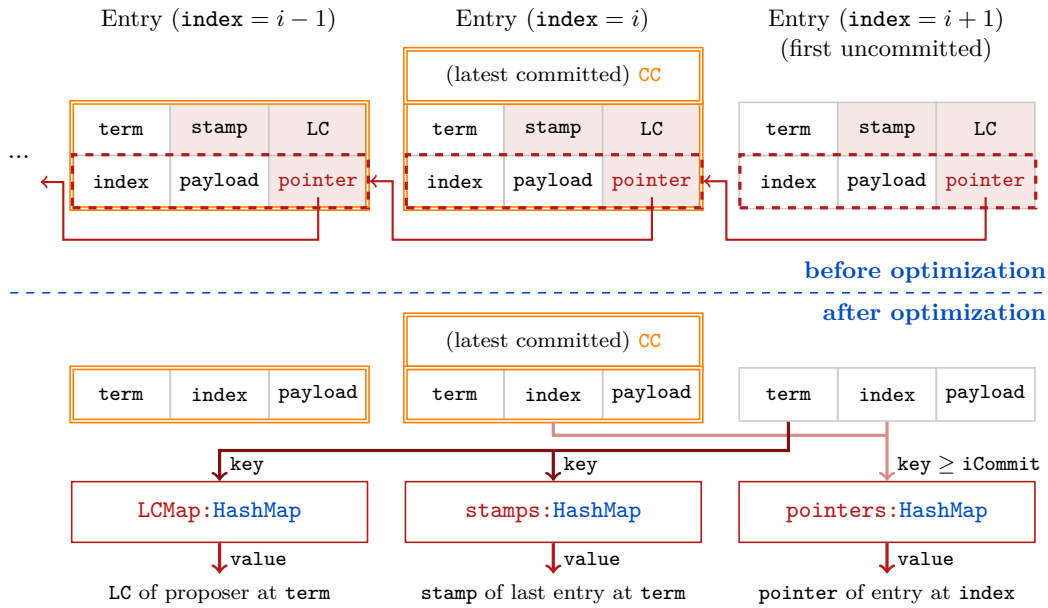
1. **Hash Pointer.** The hash pointer of log entry  $E_i$  is denoted by  $E_i.\text{pointer}$ , where  $E_0.\text{pointer} = \perp$ . It is a lightweight proof that the host node owns the entire log list from  $E_1$  to  $E_i$ . The other hash pointers can be derived by

$$E_i.\text{pointer} = \text{Hash}(i \| E_i.\text{payload} \| E_{i-1}.\text{pointer}), \forall i \in \mathbb{Z}_{>0}. \quad (1)$$

2. **Proposer Stamp.** The (proposer) stamp of log entry  $E$  is a digital signature by its proposer  $\ell$  on the hash pointer of  $E$ . We denote it by  $E.\text{stamp} = \sigma_\ell(E.\text{pointer})$ . Should a pair of stamps of  $E$  and  $E' \neq E$  exist where  $E$  is neither an ancestor or descendent of  $E'$  and  $E.\text{term} = E'.\text{term}$ ,  $\ell$  must have launched a split-brains attack.
3. **Leader Certificate (LC) of Proposer.** The LC of log entry  $E$ , denoted by  $E.\text{LC}$ , is the LC created by  $E$ 's proposer  $\ell$  at term  $E.\text{term}$ . It collects a quorum of signatures from a set of nodes  $V$  on  $\ell$ 's vote request  $r$ , where a request includes ID  $\ell$ , term  $\ell.\text{term}$ , plus the tuple (term, index, hash pointer) of  $\ell$ 's last entry  $(\tau, i, h)$ . Formally,  $r = \ell \| \ell.\text{term} \| \tau \| i \| h$  and  $\text{LC} = r \| V \| \{\sigma_u(\text{Hash}(r))\}_{u \in V}$ , as shown in Table 1.

In summary, in our basic (un-optimized) CFT-Forensics, a log entry has six attributes (Fig. 3) – **term**, **index**, **payload**, **pointer**, **stamp** and **LC**. In addition, CFT-Forensics requires each node to maintain two independent states – 4) the current leader's LC and 5) the latest CC.

4. **Leader Certificate of Current Leader.** Each node additionally maintains the LC of the current leader it identifies. This LC is not covered above because the current leader may have not proposed any log entry yet.
5. **Commitment Certificate (CC)** Each node only maintains one freshest CC. Like LCs, a CC is a collection of a quorum of signatures on the same log entry. Formally, for a log entry at index  $i$  that is replicated to a set of nodes  $V$  where  $|V| \geq n - f$ , we construct a CC following the structure in Table 1. We denote  $\text{CC} = i \| h_i \| V \| \{\sigma_u(h_i)\}_{u \in V}$ .



■ **Figure 3** Log entry attributes with and without CFT-Forensics; committed blocks are shown with a double gold outline. Our basic (unoptimized) CFT-Forensics (top) adds a hash pointer, a proposer stamp, and a leader certificate LC, all shown in red. We also store a CC only for the latest committed block. Our optimized CFT-Forensics (§6.1) reduces storage costs by storing three hash maps: (1) one containing pointers only for the last committed block and later uncommitted blocks, (2) one storing a single leader certificate LC for every term, and (3) one storing a proposer stamp only for the latest proposed block in the current term.

## 5.2.2 Modified Procedures

### Log Replication

We mark our changes in red in Alg. 1. Upon creation of a log entry  $E$  at index  $i$ , the leader  $\ell$  correctly attaches the three new states (pointer, stamp and LC). Then it replicates the “enhanced” entry to followers via the `AppendEntries` RPC. Upon receipt, each follower  $u$  validates the new states, and eventually puts the entries at their correct indices. As a result of a successful replication,  $u$  sends a `AppendEntriesResp` message, which not only includes the predicate `accept`, but also  $u$ ’s signature on the last entry  $E$ ’s hash pointer.

With  $(n - f - 1)$  `AppendEntriesResp` messages, the leader updates its CC by assembling the  $n - f$  signatures it has obtained (including its own). To notify followers to commit  $E$ , the leader sends a `InformCommit` message which includes CC in addition to  $E$ . Upon receipt, a follower commits  $E$  if it owns  $E$  and the CC passes a follower’s verification.

### Leader Election

In the `Qualification` procedure which satisfies the election property (Property 6), if a candidate  $\ell$ ’s logs are changed during `Qualification`, we let  $\ell$  reconstruct every uncommitted entry with the same payload, as if  $\ell$  plans to repropose them. In detail,  $\ell$  a) sets their terms equal to its current term, b) re-derives their hash pointers, c) creates its own stamp for each of them, and d) sets their proposer LCs to its own LC. As a result, the hash pointers will still be correct, and no entry will be overwritten if it has been committed by any node.

Assume that the candidate  $\ell$  passes the [Qualification](#) procedure in a vanilla forensics-compliant protocol. Instead of directly declaring leadership in vanilla,  $\ell$  broadcasts another vote request  $r$  based on its current last log entry by calling `RequestVote` RPC. Since  $\ell$  is already qualified, the request deserves at least  $n - f$  votes by election property. Each vote from  $u$  contains a signature  $\sigma_u(\text{Hash}(r))$ , proving  $u$ 's awareness that  $\ell$  is fresher than itself. After collecting  $n - f$  votes,  $\ell$  assembles a leadership certificate (LC) and claims leadership by broadcasting it. Then, each recipient will verify the LC, store it, and identify  $\ell$  as the leader.

In general, we add an additional round of communication to leader election, where the candidate provides information of its last log entry and the voters send signatures. In *passive* leader elections like Paxos, any arbitrary node can be elected under deterministic logic (e.g., under round robin or maximum ID). The new leader must ensure freshness by updating its log entries based on those it receives from the other nodes. As a result, the last log entry is only available after a round of communication, so a second round of signatures is needed. However, it is not needed in *active* elections like Raft, where a node actively seeks leadership candidacy. If each node never modifies its logs during election, then their last entry does not change, and they can collect signatures in just one round of communication.

### 5.3 Accountability Guarantee

► **Theorem 11.** *If a CFT protocol  $\mathcal{P}$  is forensics-compliant, then  $\mathcal{P}$ -Forensics achieves accountability (Def. 2).*

**Proof Sketch.** (Full proof in our full paper [57]) We first establish a map from each term to the LC of that term's leader. If a term is associated with two distinct LCs, we can accuse all voters that contributed signatures to both LCs, as they voted twice at the same term. If this map exists, a term is uniquely used by a leader. Since safety (Def. 1) does not hold, we find the first pair of entries from the logs of two honest nodes that conflict.

If they are of the same term, we discover a *split-brains* attack and we can accuse the leader by its stamps on the conflicting entries or their successors.

If they are of different terms, we discover a voter's attack, which has two possibilities – 1) at least one voter voted for a leader not fresher than itself; and 2) at least one voter replicated and signed an entry at a term less than its term. In this final case, we can accuse all the voters who have signatures in a pair of conflicting CC and LC. ◀

## 6 Performance Comparison with PeerReview

In this section, we provide a head-to-head comparison of the theoretical overhead costs of CFT-Forensics compared to PeerReview, for the special cases of Raft-Forensics and Paxos-Forensics. We begin by explaining some practical optimizations that reduce the redundancy of CFT-Forensics without affecting accountability, then explain the cost comparison calculations.

### 6.1 CFT-Forensics State Optimization

The added states in basic CFT-Forensics incur linear overhead in the number of log entries. We next show how to store the new states in independent, more efficient data structures.

**Hash Pointer.** We let each node  $u$  maintain the  $u[k].\text{pointer}$  **only** for  $k \geq c \triangleq u.i\text{Commit}$  in a hash map `pointers`. This is sufficient for hash pointer reads, which happens only when a node  $u$  receives a sequence of entries  $E_{i:j}$  to be updated to its logs, plus the preceding pointer  $E_{i-1}.\text{pointer}$ . We may presume  $j > c$  because  $u$  rejects updating any

committed entry. Normally,  $u$  tells whether  $E_{i:j}$  matches its own log list by whether  $u[i-1].\text{pointer} = E_{i-1}.\text{pointer}$ . If  $i \leq c$ ,  $u$  cannot find  $u[i-1].\text{pointer}$  in the hash map, but  $u$  can alternatively derive  $E_c.\text{pointer}$  by (1) and tell whether  $u[c].\text{pointer} = E_c.\text{pointer}$ . Since `Hash` is collision-resistant,  $u[i-1].\text{pointer} = E_{i-1}.\text{pointer}$  is implied by  $u[c].\text{pointer} = E_c.\text{pointer}$ . Therefore, reduction of committed hash pointers (except the last one) does not affect correctness.

**Proposer Stamp.** Suppose  $\ell$  has proposed  $\{E_k | k \in [i, j]\}$  during a leadership. Since `Hash` is collision resistant,  $E_j.\text{pointer}$  effectively represents the entire log list from the head  $E_1$  to  $E_j$ . Therefore, the stamp  $\sigma_\ell(E_j.\text{pointer})$  proves  $\ell$  has proposed not only  $E_j$ , but also  $E_i, \dots, E_{j-1}$ . Hence, the stamps on  $E_i, \dots, E_{j-1}$  are all redundant, and it suffices to keep only the last stamp  $E_j$  within  $\ell$ 's term. Because we only need to maintain one last stamp for each leader, all the stamps can be contained in a hash map `stamps` keyed by term.

**Leader Certificate.** By design, the LC used for each term is unique. Hence, we may reduce overheads by maintaining the LCs in a hash map `LCmap` keyed by term and valued by LC. Moreover, we may reduce the hash pointer inside the vote request of LC, because the pointer can be derived from the logs.

**Summary of Total Spatial Overhead.** Let  $H$  denote the length of the logs,  $H'$  the number of uncommitted entries and  $\Lambda$  the number of global leaderships during which at least one entry is replicated. Our optimized CFT-Forensics substantially reduces total overhead of the three states from  $\mathcal{O}(nH)$  to  $\mathcal{O}(H' + n\Lambda)$ . However, to reduce notations and symbols for better clarity, we continue using the primitive states in the algorithm pseudocode.

## 6.2 Cost Analysis

Using this optimized implementation, for Raft and Paxos, we compare the overhead space and communication complexities of CFT-Forensics against PeerReview. For log replication, Raft is identical to Paxos, so we merge the comparison in §6.2.1. For leader election, we compare the variants separately in §6.2.2.

### PeerReview

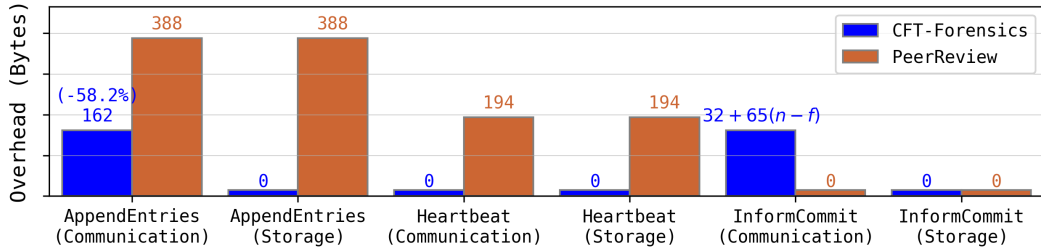
PeerReview [26] achieves accountability by logging communication for every message from any node  $u$  to another node  $j$ , regardless of the underlying consensus protocol. The communication log is an independent data structure introduced by PeerReview. We call such log entries “comm entries”, where each comm entry includes a copy of the message. To make the entire log tamper-evident, a hash pointer is maintained, just as in CFT-Forensics. We assume each comm entry stores a hash pointer, though this storage cost can be reduced by storing one pointer every few blocks, at the expense of time complexity of random access. For every message `msg` sent from  $u$  to  $v$ ,  $u$  sends `msg` along with a hash pointer and  $u$ 's signature. Then,  $v$  replies a hash pointer plus  $v$ 's signature to  $u$ . Both  $u$  and  $v$  create a new comm entry including a copy of `msg`. Hence, each message incurs communication overheads of two hash pointers and two signatures.

For auditing, PeerReview allows nodes to supervise each other by forwarding all signatures from a signer to the signer's *witnesses*. For a fair comparison between CFT-Forensics (which has a separate auditor) and PeerReview, we disable witnessing.

## 6.2.1 Log Replication

■ **Table 2** Complexities of Raft/Paxos, CFT-Forensics and PeerReview in log replication.  $\Pi$  denotes hash size and  $\Sigma$  denotes digital signature size, both in bytes.  $m$  denotes number of log messages.

	Raft/Paxos	CFT-Forensics (ours)	PeerReview
	(Base)	Communication Overhead	
Heartbeat	CONST	0	$2(\Pi + \Sigma)$
AppendEntries	$mB$	$\Pi + 2\Sigma$	$4(\Pi + \Sigma)$
InformCommit	CONST	$\Pi + (n - f)\Sigma$	0
	(Base)	Storage Overhead	
Heartbeat	0	0	$2(\Pi + \Sigma)$
AppendEntries	$mB$	0	$2mB + 4(\Pi + \Sigma)$
InformCommit	0	0	0



■ **Figure 4** Overhead complexities of CFT-Forensics and PeerReview in log replication when hash size  $\Pi = 32$  bytes and digital signatures are  $\Sigma = 65$  bytes. The height of the fifth blue bar of “InformCommit (Communication)” is plotted with  $(n, f) = (3, 1)$ .

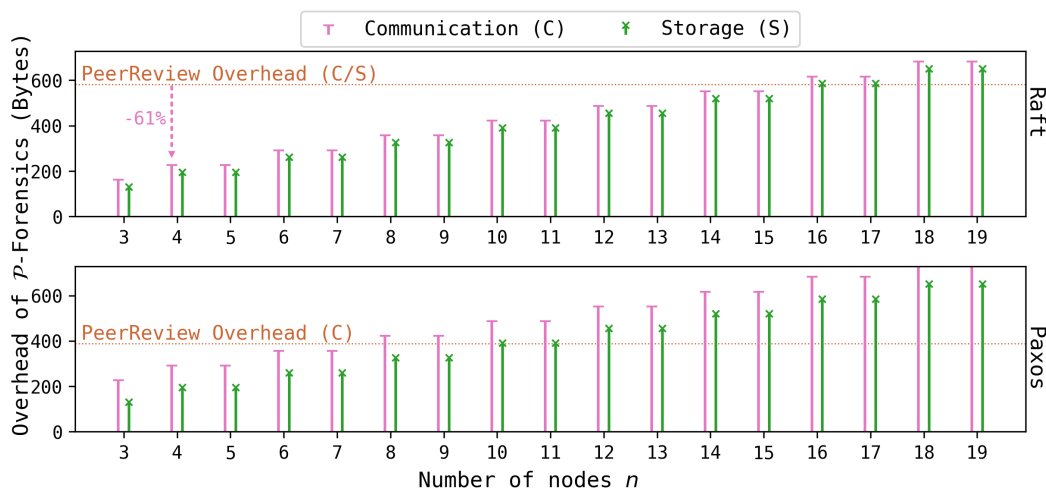
Let  $\Pi$  and  $\Sigma$  denote the sizes of a hash and a digital signature, respectively. We choose  $\Pi = 32$  bytes and  $\Sigma = 65$  bytes for numerical estimation, which are used for Ethereum[61].<sup>3</sup> Let  $B$  denote the size of a log entry. For messages including a sequence of log entries, we let  $m$  denote the number of entries. We assume nodes are up-to-date in term and need only replicate entries of current term. This limits the number of stamps and LCs sent along with the sequence. We also assume that `AppendEntries` complete in a single round, and that `InformCommit` contributes negligible overhead with batched executions.

Table 2 presents the communication and storage complexities of Raft/Paxos, CFT-Forensics and PeerReview in three main log replication RPCs. For our assumed parameter values, we numerically visualize the overheads of the `Heartbeat` and the `AppendEntries` RPCs in Fig. 4. We first observe that CFT-Forensics has **zero** storage overhead in all three RPCs, while PeerReview has a positive overhead for `Heartbeat` and `AppendEntries`. Since message frequency must be lower-bounded by the `Heartbeat` frequency which is typically once every several seconds, CFT-Forensics outperforms PeerReview by saving about 1 KB storage every minute. For communication complexity, we focus on the most frequently-used RPC: (one-round) `AppendEntries`. CFT-Forensics has a  $(\Pi + 2\Sigma = 162)$ -byte overhead in communication, which is 58.2% lower than  $4(\Pi + \Sigma) = 388$  bytes of PeerReview.



■ **Table 3** Comparison of *overhead* complexities between CFT-Forensics and PeerReview in leader election.  $I = 0$  if the candidate’s last committed entry is at the same term as the first entry it receives from the voter;  $I = 1$  otherwise. If a voter contributed a signature to the new LC, the LC it receives from the candidate does not need to include its own signature.

		Vanilla (base)	<b>CFT-Forensics (ours)</b> (Overhead)	PeerReview (Overhead)
Raft	Comm.	CONST	$\Pi + (n - f)\Sigma$	$6(\Pi + \Sigma)$
	Storage	0	$(n - f)\Sigma$	$6(\Pi + \Sigma)$
Paxos	Comm.	$mB$	$\Pi + (n - f + 1)\Sigma$	$4(\Pi + \Sigma)$
	Storage	0	$\tau(n - f)\Sigma$	$2mB + 4(\Pi + \Sigma)$



■ **Figure 5** Overhead complexities of CFT-Forensics and PeerReview in leader election.

## 6.2.2 Leader Election

### Raft-Forensics vs Raft-PeerReview

Now we consider Raft’s leader election. A successful election has three messages between a candidate  $\ell$  and its voter  $u$ : 1)  $\ell$  sends vote request to  $u$ ; 2)  $u$  responds with a vote; and 3)  $\ell$  sends a leadership claim. As shown in Table 3 and Fig. 5, although LC contributes an  $\mathcal{O}(n)$  overhead to CFT-Forensics, both complexities are still lower than Raft-PeerReview for  $n \leq 15$  (under our assumed parameter values).

### Paxos-Forensics vs Paxos-PeerReview

A successful Paxos leader election has two messages between a candidate  $\ell$  and its voter  $u$ : 1)  $\ell$  sends its `iCommit` to  $u$ ; 2)  $u$  responds with all its entries starting with `iCommit + 1`. In Paxos-Forensics, we insert three more messages: 3)  $\ell$  sends a vote request to  $u$ ; 4)  $u$  responds with a signed vote and 5)  $\ell$  sends an LC to claim leadership. Table 3 lists the overheads for Paxos. We assume that leader elections are rare, so message 2) only includes entries of same term as  $\ell$ [`iCommit`]. By Fig. 5, Paxos has lower communication complexity than Paxos-PeerReview if  $n \leq 7$ , and on a long enough timescale, its storage complexity is arbitrarily lower than that of Paxos-PeerReview.

<sup>3</sup> Ethereum uses Keccak-256 and ECDSA-secp256k1 for hashes and digital signatures, respectively.

## 7 Empirical Evaluation

We implement Raft-Forensics<sup>4</sup> in C++ based on nuRaft v1.3 [17] by eBay. With roughly 2,500 lines of code, our implementation fully expands nuRaft with our OpenSSL-based designs in log replication, which correctly reflects the throughput and latency performances between leader elections. We choose the SHA-256 hash function and Elliptic Curve Digital Signature Algorithm (ECDSA) over the secp256r1 curve. For commitment certificates, we used concatenated ECDSA signatures by all the signers.

We evaluate Raft-Forensics in two phases – online phase (§7.1) and offline phase (§7.2). In the online phase, we benchmark the performance of Raft-Forensics over a WAN. In the offline phase, we evaluate the auditing procedure that scans node logs for adversarial behaviors.

### 7.1 Online Evaluation

#### Setup on AWS

We evaluate Raft-Forensics over a WAN to demonstrate a geo-redundant deployment for increased resilience [12]. We simulated the WAN environment by deploying Raft-Forensics and other baseline protocols on multiple `c5.large` instances on AWS, where each instance has 2 vCPUs and 4 GB Memory. We ran the experiments on 4 and 16 instances, respectively. Because some typical applications of Raft-Forensics require the nodes to be distributed domestically, we deployed the 16 instances evenly in 8 AWS datacenters in the US, Canada and Europe. For the 4-instance experiments, we deployed the instances in 4 US datacenters.

#### Baseline Protocols

We compare the performance of Raft-Forensics against Raft [17], using eBay’s NuRaft [17] implementation. We do not directly compare to state-of-the-art BFT protocols in our evaluation because our goal is to propose low-cost solutions that can be easily integrated into existing systems (i.e., the implementation should build upon existing code, and hence be some variant of Raft). Although there exist BFT variants of Raft [56, 60, 13], we were unable to confirm essential theoretical details needed to understand the protocol and guarantees. For completeness, we compare Raft-Forensics against a recent BFT protocol called Dumbo-NG [20] in our full paper [57], though a fair comparison is challenging and not the focus of this work.

#### Experimental Settings

We benchmark each protocol by two metrics – transaction latency and throughput. Latency is measured by the average time difference between when a transaction is committed by a leader and when it is sent to a node. Throughput is measured by the average number of transactions processed per second during an experiment.

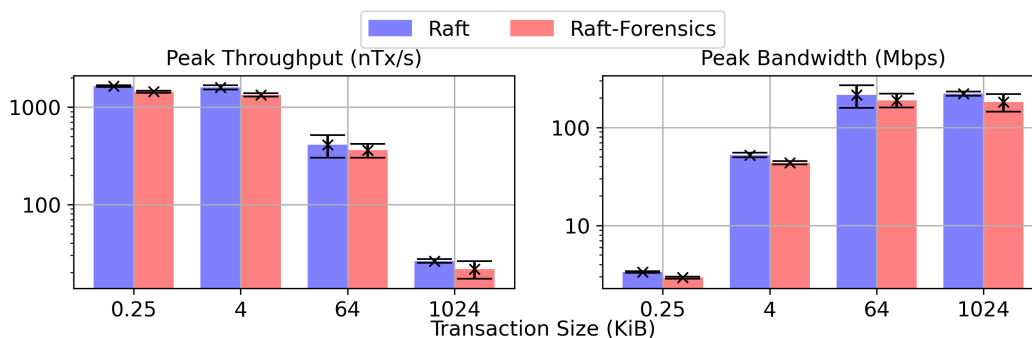
The experiments are configured by two key parameters – transaction size and number of concurrent clients. The transaction sizes range from 256 Bytes to 1 MB. For each transaction size, we sweep the number of concurrent clients sending transactions (in experiments, we let the leader machine spawn transactions). Under each configuration of transaction size and client concurrency, we run all the nodes and client processes simultaneously for 20 seconds.

---

<sup>4</sup> <https://github.com/proy-11/NuRaft-Forensics.git>

We measure transaction latency and throughput by the average of five repeated runs to reduce random perturbations. Typically, as the number of clients increases, throughput increases first linearly and then plateaus when the protocol is saturated. In contrast, latency is insensitive to the number of clients before the saturation, but rapidly increases when the bottleneck throughput is reached. We finally evaluate the following quantities:

- **Peak throughput.** We measure the peak throughput of each baseline as the maximum number of transactions processed per second over all numbers of concurrent clients. Fig. 6 presents the performance of all protocols under transaction sizes of 256 Bytes, 4 KiB, and 64 KiB. Compared to Raft, Raft-Forensics has an approximately 10% loss in peak throughput under various transaction sizes, which is caused by the cryptographic operations involved.
- **Latency-Throughput tradeoff.** Under each transaction size, we measure the latency-throughput curve parameterized by number of concurrent clients. Fig. 7 shows the latency-throughput tradeoffs of the two protocols under various transaction sizes. Generally, the tradeoff of Raft-Forensics is only slightly worse than Raft.



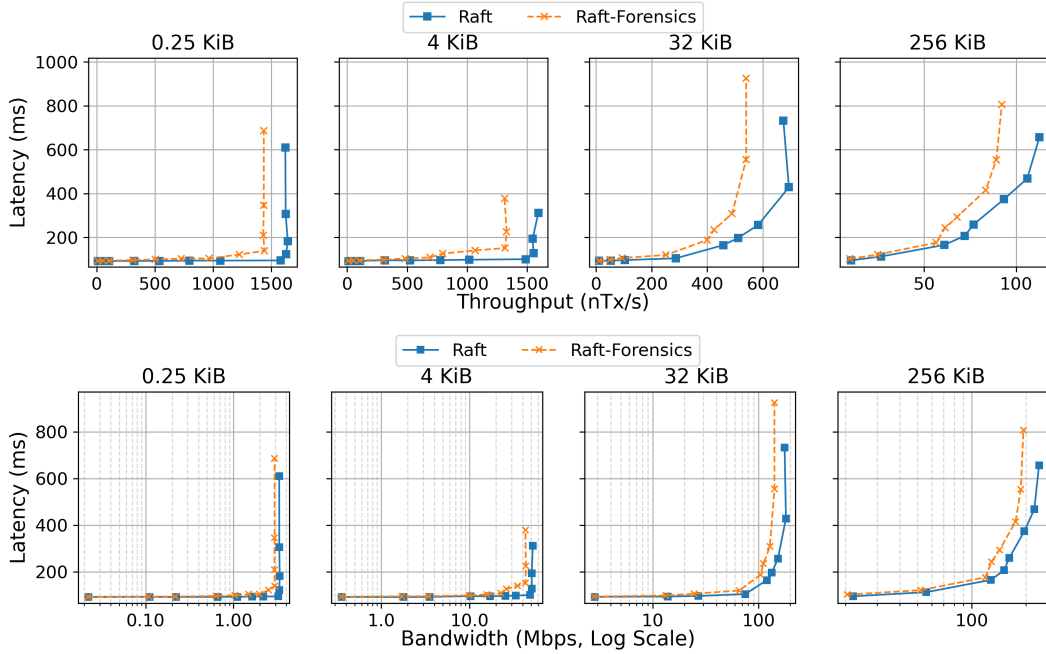
■ **Figure 6** Peak transaction and bandwidth throughputs of consensus algorithms. We plot the error bars with boundaries (mean  $\pm 3 \times$  std). ( $n = 4$  nodes).

## 7.2 Offline Evaluation

We next evaluate the offline performance of log auditing. Theorem 11 ensures that we can find at least 1 culprit when State Machine Safety is violated, and we implement an auditing algorithm that finds the culprit. Briefly, the algorithm has two parts: a data legitimacy check and a global consistency check. First, the data legitimacy check verifies the correctness and completeness of the states submitted by each node. For example, the hash pointers must match the logs, and every signature must pass verification. Next, the global consistency check scans for a pair of nodes whose logs are a result of forking. It captures the culprit based on the case discussions in the proof of Theorem 11. See our full paper [57] for the auditing algorithm in detail.

### Complexity Analysis

Recall that  $n$  denotes the number of nodes. Let  $H$  denote the length of the longest chain and  $\Lambda$  the number of elections in total. See our full paper [57] for detailed complexity analysis. The total time complexity of auditing is asymptotically optimal (linear in the size of data  $n(H + \Lambda)$ ), which is required at minimum to ensure data legitimacy), where the complexity



■ **Figure 7** Latency-throughput tradeoff ( $n = 4$  nodes). Top row displays throughput in number of transactions per second; bottom row displays throughput in bandwidth.

of global consistency checks does not depend on the chain length  $H$ . The linear spatial complexity  $\Theta(n(H + \Lambda))$  requires chunked storage of the log chain. For instance, for a chunk size  $\Theta(\log H)$ , the spatial complexity decreases to  $\Theta(n(\log H + \Lambda))$ , while the time complexity remains the same. Notably, the time complexity of global consistency check slightly increases to  $\mathcal{O}(n(\Lambda + \log^2 H))$ , but is still much less than that of legitimacy checks.

### Implementation

We implement the auditing algorithm in Python<sup>5</sup>, which can be tested along with a lightweight Raft simulator that achieves better control than the fully-implemented Raft-Forensics in C++ over the leader elections, the adversarial nodes’ behavior and race conditions in general. In particular, it is capable of assigning the adversary to a node and simulating the fork and bad vote attacks in Examples 9 and 10. It ensures that the adversary generates legitimate data to prevent it from being caught before consistency checks. For the best performance in memory usage, it writes the data into chunked files that are available for auditing. In Appendix 7.2, we run benchmarks on the performance of both the data legitimacy and consistency checks of the auditing algorithm. The benchmarks are consistent with our complexity analysis, and demonstrate a significant advantage in chunking data.

Based on the backend software above, we also implement a visualizer based on [52] that demonstrates the attacks and the outputs of the auditing algorithm, including the identity of the culprit and the irrefutable evidence. See our full paper [57] for a screenshot of the visualizer.

<sup>5</sup> <https://github.com/WeizhaoT/Raft-Forensics-Simulator>

■ **Table 4** Throughput and latency of two different OpenCBDC architectures integrated with Raft and Raft-Forensics (ours), respectively. Each entry is expressed in mean  $\pm$  std.

	Throughput (# tx/s)	Latency (ms)
<b>2pc architecture</b>		
Raft	4,800 $\pm$ 14	2,251 $\pm$ 70
Raft-Forensics	4,695 $\pm$ 76	2,577 $\pm$ 252
(% Change)	-105 (-2.2%)	+326 (+14.5%)
<b>atomizer architecture</b>		
Raft	1,284 $\pm$ 56	37,552 $\pm$ 18.75
Raft-Forensics	1,250 $\pm$ 123	40,802 $\pm$ 1,653
(Change)	-34 (-2.6%)	+3,250 (+8.7%)

### 7.3 Integration with OpenCBDC

Finally, we evaluate the performance of Raft-Forensics integrated into a downstream application: OpenCBDC [41], an open-source implementation of a retail central bank digital currency. OpenCBDC is a good choice because (a) it uses nuRaft, and (b) CBDCs are/will be public infrastructure, so security and performance are paramount. After integrating our Raft-Forensics implementation into OpenCBDC, we deployed our experiments onto `c5n.9xlarge` `ec2` instances in AWS over three regions: `us-east-1`, `us-east-2` and `us-west-2`.<sup>6</sup>

We compared Raft-Forensics against Raft in two different OpenCBDC architectures – two-phase-commit (2pc) and atomizer. In both architectures, we replace Raft with Raft-Forensics in every module that is Raft-replicated, i.e., implemented as Raft-variant distributed systems. In the 2pc architecture, we created one generator, one sentinel, three coordinators and three shards, where each coordinator and each shard are Raft-replicated. In the atomizer architecture, we created one watchtower, one watchtower CLI, one sentinel, one archiver, four shards and three atomizers, where only atomizers are Raft-replicated. In both architectures, each Raft-replicated module consists of 3 nodes in 3 different AWS regions.

We used the benchmarking platform [44] of OpenCBDC under default configurations, where load generators produce as much workload as the system can process. The transaction size is 368 bytes. Each experiment lasts 315 seconds and is repeated 3 times. Table 4 shows the throughput and latency of transactions of the entire system. We observe that in practical complex systems like OpenCBDC, Raft-Forensics also performs close to Raft.

## 8 Discussion and Conclusion

This work is driven by the motivation to improve the Byzantine resistance of CFT protocols by introducing accountability, without sacrificing too much performance. One alternative approach to achieving higher security assurances with CFT protocols involves employing BFT protocols directly. This strategy not only increases tolerance to Byzantine faults but may also inherently include accountability as a bonus feature.

As explained in Section 7, we were unable to directly compare against BFT variants of Raft [56, 60, 13]. Hence, we conducted performance comparisons between Raft-Forensics and leading BFT designs like Dumbo-NG, as detailed in our full paper [57]. Our analysis indicates that, in terms of reducing latency, Raft-Forensics generally surpasses Dumbo-NG,

<sup>6</sup> Although CFT protocols are often run in the same datacenter, if they are used for critical infrastructure, there will be a need for geographically-distributed deployments for robustness reasons.

though the latter may display competitive or superior throughput for larger transaction volumes. Moreover, Dumbo-NG is optimized for efficiently propagating blocks containing multiple transactions among numerous participants, while Raft variants typically handle single-transaction blocks (as required by SMR) in small-scale distributed systems. As a result, we acknowledge that BFT protocols can indeed be optimized to achieve good performance and replace CFT protocols in applications requiring higher security guarantees, albeit at the cost of increased design complexity and an overhaul of the entire consensus logic. In contrast, accountability may be more suitable for scenarios with moderate security improvement requirements and an emphasis on lightweight changes.

More broadly, accountability need not be viewed as an alternative to Byzantine fault tolerance – it is a complementary, desirable property. For example, all BFT protocols do not inherently offer accountability [50]. We posit that accountability is an important component of distributed system governance – all the more so for geographically-distributed critical infrastructure [19].

---

## References

- 1 Kyle Banker, Douglas Garrett, Peter Bakkum, and Shaun Verch. *MongoDB in Action: Covers MongoDB Version 3.0*. Simon and Schuster, 2016.
- 2 Shehar Bano, Alberto Sonnino, Mustafa Al-Bassam, Sarah Azouvi, Patrick McCorry, Sarah Meiklejohn, and George Danezis. SoK: Consensus in the Age of Blockchains. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, pages 183–198, 2019.
- 3 Michael Ben-Or, Danny Dolev, and Ezra N. Hoch. Simple gradecast based algorithms, 2010. [arXiv:1007.1049](https://arxiv.org/abs/1007.1049).
- 4 Romain Boichat, Partha Dutta, Svend Frølund, and Rachid Guerraoui. Deconstructing Paxos. *SIGACT News*, 34(1):47–67, March 2003. doi:10.1145/637437.637447.
- 5 Mike Burrows. The Chubby Lock Service For Loosely-coupled Distributed Systems. In *Proceedings of the 7th symposium on Operating systems design and implementation*, pages 335–350, 2006.
- 6 Vitalik Buterin and Virgil Griffith. Casper the Friendly Finality Gadget. *arXiv preprint arXiv:1710.09437*, 2017.
- 7 Christian Cachin and Marko Vukolić. Blockchain Consensus Protocols in the Wild. *arXiv preprint arXiv:1707.01873*, 2017.
- 8 Apache Cassandra. Apache Cassandra. *Website*. Available online at <http://planetcassandra.org/what-is-apache-cassandra>, 13, 2014.
- 9 Miguel Castro, Barbara Liskov, et al. Practical Byzantine Fault Tolerance. In *OsDI*, volume 99, pages 173–186, 1999.
- 10 Ben Christensen. Fault Tolerance in A High Volume, Distributed System. Netflix Blog, 2012. URL: <https://netflixtechblog.com/fault-tolerance-in-a-high-volume-distributed-system-91ab4faae74a>.
- 11 Pierre Civit, Seth Gilbert, and Vincent Gramoli. Polygraph: Accountable Byzantine Agreement. *IACR Cryptol. ePrint Arch.*, 2019:587, 2019.
- 12 Team Cloudify. Geo Redundancy Explained, Cloudify. Cloudify Blog, 2021. URL: <https://cloudify.co/blog/geo-redundancy-explained/>.
- 13 Christopher Copeland and Hongxia Zhong. Tangaroa: a byzantine fault tolerant raft. *Stanford University*, 2016.
- 14 James C Corbett, Jeffrey Dean, Michael Epstein, Andrew Fikes, Christopher Frost, Jeffrey John Furman, Sanjay Ghemawat, Andrey Gubarev, Christopher Heiser, Peter Hochschild, et al. Spanner: Google’s Globally Distributed Database. *ACM Transactions on Computer Systems (TOCS)*, 31(3):1–22, 2013.
- 15 Roberto De Prisco, Butler Lampson, and Nancy Lynch. Revisiting the Paxos Algorithm. *Theoretical Computer Science*, 243(1-2):35–91, 2000.

- 16 Antonella Del Pozzo and Thibault Rieutord. Fork accountability in tenderbake. In *5th International Symposium on Foundations and Applications of Blockchain 2022 (FAB 2022)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.
- 17 eBay. NuRaft. <https://github.com/eBay/NuRaft/tree/v1.3>, 2017. Accessed on April 19, 2023.
- 18 etcd. Etcd. <https://etcd.io/>, 2023. Accessed on April 19, 2023.
- 19 Mohamed Ezzeldin and Wael E El-Dakhakhni. Robustness of Ontario Power Network under Systemic Risks. *Sustainable and resilient infrastructure*, 6(3-4):252–271, 2021.
- 20 Yingzi Gao, Yuan Lu, Zhenliang Lu, Qiang Tang, Jing Xu, and Zhenfeng Zhang. Dumbo-NG: Fast Asynchronous BFT Consensus with Throughput-oblivious Latency. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 1187–1201, 2022.
- 21 Nishant Garg. *Apache Kafka*. Packt Publishing Birmingham, UK, 2013.
- 22 Rati Gelashvili, Lefteris Kokoris-Kogias, Alberto Sonnino, Alexander Spiegelman, and Zhuolun Xiang. Jolteon and Ditto: Network-adaptive Efficient Consensus with Asynchronous Fallback. In *Financial Cryptography and Data Security: 26th International Conference, FC 2022, Grenada, May 2–6, 2022, Revised Selected Papers*, pages 296–315. Springer, 2022.
- 23 Diana Ghinea, Vipul Goyal, and Chen-Da Liu-Zhang. Round-optimal byzantine agreement. Cryptology ePrint Archive, Paper 2022/255, 2022. URL: <https://eprint.iacr.org/2022/255>.
- 24 Mike Graf, Ralf Küsters, and Daniel Rausch. Accountability in A Permissioned Blockchain: Formal Analysis of Hyperledger Fabric. In *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 236–255. IEEE, 2020.
- 25 Bingyong Guo, Yuan Lu, Zhenliang Lu, Qiang Tang, Jing Xu, and Zhenfeng Zhang. Speeding Dumbo: Pushing Asynchronous BFT Closer To Practice. *Cryptology ePrint Archive*, 2022.
- 26 Andreas Haeberlen, Petr Kouznetsov, and Peter Druschel. PeerReview: Practical Accountability For Distributed Systems. *ACM SIGOPS operating systems review*, 41(6):175–188, 2007.
- 27 Moin Hasan and Major Singh Goraya. Fault Tolerance in Cloud Computing Environment: A Systematic Survey. *Computers in Industry*, 99:156–172, 2018.
- 28 HashiCorp. Consul. <https://www.consul.io/>, 2023. Accessed on April 19, 2023.
- 29 Heidi Howard and Richard Mortier. Paxos vs Raft: Have We Reached Consensus on Distributed Consensus? In *Proceedings of the 7th Workshop on Principles and Practice of Consistency for Distributed Data*, EuroSys ’20. ACM, April 2020. doi:10.1145/3380787.3393681.
- 30 Patrick Hunt, Mahadev Konar, Flavio Paiva Junqueira, and Benjamin Reed. ZooKeeper: Wait-free Coordination For Internet-scale Systems. In *USENIX annual technical conference*, volume 8, 2010.
- 31 Marios Kogias and Edouard Bugnion. Hovercraft: achieving scalability and fault-tolerance for microsecond-scale datacenter services. In *Proceedings of the Fifteenth European Conference on Computer Systems*, EuroSys ’20, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3342195.3387545.
- 32 Robert Künnemann, Ilkan Esiyok, and Michael Backes. Automated verification of accountability in security protocols. *CoRR*, abs/1805.10891, 2018. arXiv:1805.10891.
- 33 Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Accountability: Definition and Relationship To Verifiability. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 526–535, 2010.
- 34 Leslie Lamport. The Part-time Parliament. *ACM Trans. Comput. Syst.*, 16(2):133–169, May 1998. doi:10.1145/279227.279229.
- 35 Leslie Lamport. Paxos Made Simple. *ACM SIGACT News (Distributed Computing Column)* 32, 4 (Whole Number 121, December 2001), pages 51–58, December 2001. URL: <https://www.microsoft.com/en-us/research/publication/paxos-made-simple/>.

- 36 Leslie Lamport. *The part-time parliament*, pages 277–317. Association for Computing Machinery, New York, NY, USA, 2019. doi:10.1145/3335772.3335939.
- 37 Butler Lampson. The ABCD’s of Paxos. In *Proceedings of the Twentieth Annual ACM Symposium on Principles of Distributed Computing*, PODC ’01, page 13, New York, NY, USA, 2001. Association for Computing Machinery. doi:10.1145/383962.383969.
- 38 Butler W Lampson. How To Build A Highly Available System Using Consensus. In *International Workshop on Distributed Algorithms*, pages 1–17. Springer, 1996.
- 39 Barbara Liskov and James Cowling. Viewstamped replication revisited. Technical Report MIT-CSAIL-TR-2012-021, MIT, July 2012.
- 40 Shengyun Liu, Paolo Viotti, Christian Cachin, Vivien Quéma, and Marko Vukolic. XFT: Practical Fault Tolerance Beyond Crashes. In *OSDI*, pages 485–500, 2016.
- 41 James Lovejoy, Madars Virza, Cory Fields, Kevin Karwaski, Anders Brownworth, and Neha Narula. Hamilton: A *High-Performance* Transaction Processor For Central Bank Digital Currencies. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 901–915, 2023.
- 42 Nancy A Lynch. *Distributed Algorithms*. Elsevier, 1996.
- 43 Hein Meling and Leander Jehl. Tutorial Summary: Paxos Explained from Scratch. In *International Conference On Principles Of Distributed Systems*, pages 1–10. Springer, 2013.
- 44 mit dci. Opencbdc-tctl. <https://github.com/mit-dci/opencbdc-tctl>, 2022. Accessed on April 19, 2023.
- 45 Joachim Neu, Ertem Nusret Tas, and David Tse. The Availability-accountability Dilemma and Its Resolution via Accountability Gadgets. In *International Conference on Financial Cryptography and Data Security*, pages 541–559. Springer, 2022.
- 46 Joachim Neu, Ertem Nusret Tas, and David Tse. Accountable Safety Implies Finality. *arXiv preprint arXiv:2308.16902*, 2023.
- 47 Diego Ongaro and John Ousterhout. In Search of An Understandable Consensus Algorithm. In *2014 {USENIX} Annual Technical Conference ({USENIX}{ATC} 14)*, pages 305–319, 2014.
- 48 Mohammad Roohitavaf, Jung-Sang Ahn, Woon-Hak Kang, Kun Ren, Gene Zhang, Sami Ben-Romdhane, and Sandeep S Kulkarni. Session Guarantees with Raft and Hybrid Logical Clocks. In *Proceedings of the 20th International Conference on Distributed Computing and Networking*, pages 100–109, 2019.
- 49 Ermin Sakic and Wolfgang Kellerer. Response Time and Availability Study of RAFT Consensus in Distributed SDN Control Plane. *IEEE Transactions on Network and Service Management*, 15(1):304–318, 2017.
- 50 Peiyao Sheng, Gerui Wang, Kartik Nayak, Sreeram Kannan, and Pramod Viswanath. BFT Protocol Forensics. In *Proceedings of the 2021 ACM SIGSAC conference on computer and communications security*, pages 1722–1743, 2021.
- 51 Peiyao Sheng, Gerui Wang, Kartik Nayak, Sreeram Kannan, and Pramod Viswanath. Player-replaceability and Forensic Support Are Two Sides of the Same (crypto) Coin. *Cryptology ePrint Archive*, 2022.
- 52 simplespy. DiemForensics. <https://github.com/simplespy/DiemForensics>, 2020. Accessed on April 19, 2023.
- 53 Swaminathan Sivasubramanian. Amazon DynamoDB: A Seamlessly Scalable Non-relational Database Service. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 729–730, 2012.
- 54 Alistair Stewart and Eleftherios Kokoris-Kogia. GRANDPA: A Byzantine Finality Gadget. *arXiv preprint arXiv:2007.01560*, 2020.
- 55 Rebecca Taft, Irfan Sharif, Andrei Matei, Nathan VanBenschoten, Jordan Lewis, Tobias Grieger, Kai Niemi, Andy Woods, Anne Birzin, Raphael Poss, et al. Cockroachdb: The Resilient Geo-distributed Sql Database. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 1493–1509, 2020.





- 56 Dezhi Tan, Jianguo Hu, and Jun Wang. VBBFT-Raft: An Understandable Blockchain Consensus Protocol with High Performance. In *2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT)*, pages 111–115, 2019. doi:10.1109/ICCSNT47585.2019.8962479.
- 57 Weizhao Tang, Peiyao Sheng, Ronghao Ni, Pronoy Roy, Xuechao Wang, Giulia Fanti, and Pramod Viswanath. Cft-forensics: High-performance byzantine accountability for crash fault tolerant protocols, 2024. arXiv:2305.09123.
- 58 Robbert Van Renesse and Deniz Altinbuken. Paxos Made Moderately Complex. *ACM Computing Surveys (CSUR)*, 47(3):1–36, 2015.
- 59 Jun Wan, Atsuki Momose, Ling Ren, Elaine Shi, and Zhuolun Xiang. On the amortized communication complexity of byzantine broadcast. In *Proceedings of the 2023 ACM Symposium on Principles of Distributed Computing*, PODC '23, pages 253–261, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3583668.3594596.
- 60 Zhou Wang, Zhang and Xu. A Byzantine Fault Tolerance Raft Algorithm Combines with BLS Signature. *Journal of Applied Sciences*, 38(1):93, 2020. doi:10.3969/j.issn.0255-8297.2020.01.007.
- 61 Dr. Gavin Wood. Ethereum: A Secure Decentralised Generalised Transaction Ledger (Paris Version). <https://ethereum.github.io/yellowpaper/paper.pdf>, March 2024. (Accessed on 05/22/2024).
- 62 Maofan Yin, Dahlia Malkhi, Michael K Reiter, Guy Golan Gueta, and Ittai Abraham. HotStuff: BFT Consensus with Linearity and Responsiveness. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 347–356, 2019.



# Cross Ledger Transaction Consistency for Financial Auditing

Vlasis Koutsos  

Hong Kong University of Science and Technology, Hong Kong

Xiangan Tian  

Hong Kong University of Science and Technology, Hong Kong

Dimitrios Papadopoulos  

Hong Kong University of Science and Technology, Hong Kong

Dimitris Chatzopoulos  

University College Dublin, Ireland

---

## Abstract

Auditing throughout a fiscal year is integral to organizations with transactional activity. Organizations transact with each other and record the details for all their economical activities so that a regulatory committee can verify the lawfulness and legitimacy of their activity. However, it is computationally infeasible for the committee to perform all necessary checks for each organization. To overcome this, auditors assist in this process: organizations give access to all their internal data to their auditors, who then produce reports regarding the consistency of the organization's data, alerting the committee to any inconsistencies. Despite this, numerous issues that result in fines annually revolve around such inconsistencies in bookkeeping *across organizations*. Notably, committees wishing to verify the correctness of auditor-provided reports need to redo all their calculations; a process which is computationally proportional to the number of organizations. In fact, it becomes prohibitive when considering real-world settings with thousands of organizations. In this work, we propose two protocols, CLOSC and CLOLC, whose goals are to enable auditors and a committee to verify the consistency of transactions across different ledgers. Both protocols ensure that for every transaction recorded in an organization's ledger, there exists a dual one in the ledger of another organization while safeguarding against other potential attacks. Importantly, we minimize the information leakage to auditors and other organizations and guarantee three crucial security and privacy properties that we propose: (i) transaction amount privacy, (ii) organization-auditor unlinkability, and (iii) transacting organizations unlinkability. At the core of our protocols lies a two-tier ledger architecture alongside a suite of cryptographic tools. To demonstrate the practicality and scalability of our designs, we provide extensive performance evaluation for both CLOSC and CLOLC. Our numbers are promising, i.e., all computation and verification times lie in the range of seconds, even for millions of transactions, while the on-chain storage costs for an auditing epoch are encouraging i.e. in the range of GB for millions of transactions and thousands of organizations.

**2012 ACM Subject Classification** Security and privacy → Privacy-preserving protocols

**Keywords and phrases** Financial auditing, Two-tier ledger architecture, Smart contracts, Transaction privacy, Financial entity unlinkability

**Digital Object Identifier** 10.4230/LIPIcs.AFT.2024.4

**Related Version** *Full Version*: <https://eprint.iacr.org/2024/1155.pdf>

**Supplementary Material** *Software (Code)*: <https://github.com/auti-project>

**Funding** This work was supported in part by the Hong Kong Research Grants Council under grant GRF-16200721 and by the EU Horizon project no 101160671 (DIGITISE).

**Acknowledgements** We would like to thank the anonymous reviewers for their feedback and Pierre-Louis Roman for shepherding our paper.



© Vlasis Koutsos, Xiangan Tian, Dimitrios Papadopoulos, and Dimitris Chatzopoulos;  
licensed under Creative Commons License CC-BY 4.0  
6th Conference on Advances in Financial Technologies (AFT 2024).

Editors: Rainer Böhme and Lucianna Kiffer; Article No. 4; pp. 4:1–4:25

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1 Introduction

Bookkeeping is an indispensable part of *organizations* (e.g., businesses, municipalities, banks). One of the main reasons organizations maintain ledgers with their transactions (to/from other organizations) is to convince *committees* (e.g., the Public Company Accounting Oversight Board [4] in the United States of America or the Financial Reporting Council [2] in the United Kingdom) about the integrity and lawfulness of their operations. They periodically produce statements about the integrity and correctness of their finances, signed by an *auditor*. The goal of auditors is to ensure that there are no mistakes or inconsistencies in the organization-reported numbers [26]. To that end, they sign/generate a report on the organization-provided data. Figure 1 showcases a (simplified) flowchart model of organizations, auditors, and the committee during a financial epoch. Organizations transact with each other, keep respective records, and disclose them to their auditors at the appropriate time. The auditors, after examining the provided data, generate and sign a report attesting to the judicious activity of their client-organization, and send it over to a committee who verifies its content.

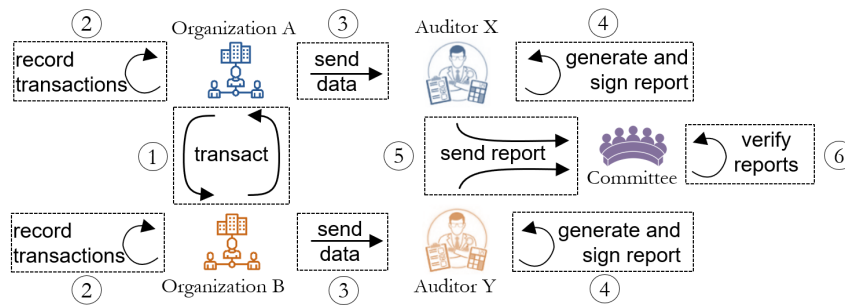
Although auditors have access to *all* organization-reported data and paperwork of their clients, checking for consistency is prohibitive in terms of human resources and time constraints [25, 27]. This is mainly because organizations under audit may record hundreds of transactions daily. To address this issue, auditors have developed probabilistic *processes* that check for consistency between the received data and the actual paperwork, but not for the entirety of the data. This, in turn, opens up the audit to additional risk. However, it is currently the best tool professionals use to ensure the audit's feasibility [11].

**Established auditing process.** Organizations record their transactions in a ledger throughout the fiscal year, meaning that a *ledger* is a list of financial transactions an organization holds over time. At a later point, the auditing period begins, during which auditors sample a percentage of the total reported transactions and request to examine the corresponding paperwork from their client-organizations. Upon conducting all relevant checks (e.g., validity of signatures, and consistency of amounts and timestamps) the auditor includes their findings regarding the auditing output in a report, which they later sign and make public. After the auditing period, a committee can select to verify the consistency between the auditor-generated reports and the recorded data of the organizations. For consistency verifiability purposes, *all transactions have to be kept in the ledgers of both transactional parties*. Especially for organization-to-organization (O2O) transactions, *each of them* needs to register a transaction that is the dual equivalent of the other.

### The problem we focus on

There are numerous cases annually of organizations misreporting transactions. For example, they fabricate and report sham transactions, which in turn leads to auditing scandals involving fraud and fines in the range of millions of USD [20, 40, 41]. In fact, this derives from a crucial limitation of the existing auditing ecosystem: auditors cannot check if a recorded transaction in their client's ledger has a dual counterpart in another organization's ledger [28]. E.g., an organization may procure illicit funds and fabricate transaction records, for which no other organization would have dual transactions to. Despite its importance, to the best of our knowledge, no process exists to check the duality of transactions between two ledgers where the organizations that maintain them are examined by separate auditors. Hence, we pose the following question:

*Can we ensure that an auditor, who does not have access to internal data of any organization except of her client-organization, can verify the duality of all O2O transactions of her client?*



■ **Figure 1** Entity setting, interactions, and flowchart for current financial auditing. Organization A is audited by Auditor X and B by Y respectively. Auditors generate reports on data received from their client-organization that the Committee can later verify.

**Limitations of applying existing cryptographic approaches.** Before going further, we examine two ideas utilizing existing techniques to solve this problem at a high level. First, the auditor and the two transacting parties could engage in a multi-party computation protocol at the time of the auditing. Such an attempt requires the auditor to interact with all organizations its client has transactions with, introducing a linear communication overhead proportional to the number of organizations and auditors. Moreover, let's consider having just two organizations,  $o_i$  and  $o_j$ , who initiate a 2-party computation protocol and agree on a common identifier for a mutual transaction. Upon deal completion,  $o_j$  will provide  $o_i$  with a digest  $dgs_{j,z}$  (e.g., a cryptographic accumulator) and a proof that  $dgs_{j,z}$  “contains” the  $o_j$ -part of the trade, and vice versa, proving the existence of the transactions in question. However, in reality, organizations do not trust each other, and auditors do not assume organizations to behave honestly. In fact, interesting questions arise in such scenarios:

1. What happens if  $o_i$  or  $o_j$  do not follow the protocol?
2. What happens if  $o_i$  computes a digest, sends it over to  $o_j$  with convincing proof, but later on includes the transaction in its ledger using different data?
3. What happens if  $o_i$  appends an incorrect proof with its data on their internal ledger?

Since organizations cannot access the ledger of their trading counterparts, auditors cannot distinguish even between trivial scenarios e.g., identify which party made mistakes or uploaded inconsistent data. Requiring auditor collaboration to discover the truth behind inconsistencies is far from a realistic assumption – both from a performance and a real-world perspective.

Another approach could be the following: Consider two organizations transacting with each other and recording this transaction on their ledgers. Each of them now can produce and communicate to the other a zero-knowledge proof (ZKP) about the inclusion of the transaction record in their respective ledgers. However, this approach works only when assuming that both organizations are honestly maintaining their ledgers. E.g., a client might generate and provide a ZKP about an O2O transaction to its counterpart, but later alter its ledger state, before the audit begins. The ZKP would still verify, as it was honestly generated at the time, but vitally the ledger alteration would be undetectable. Thus, a notion of ledger-immutability is necessary, on top of such a technique.

There exists an additional limitation in the current auditing ecosystem: committees wishing to verify process outputs need to re-perform all operations themselves. Specifically, a committee that attempts to verify all process outputs needs to expend the entirety of the collective effort from all auditors. Verifying all processes on all ledgers is rendered impractical in this case since the verification time is linearly proportional to the number of organizations and auditors. Instead, committees perform checks on a number of reports and thus trust

## 4:4 Cross Ledger Transaction Consistency for Financial Auditing

implicitly the remaining ones to be generated honestly. Various systems utilize techniques such as verifiable computation, secure hardware, or tailored ZKPs to enable auditing parties to verify function output results in sublinear time, and in Section 2 we investigate them in more detail. However, in the scope of the *cross-ledger transaction consistency for financial auditing* that we examine no such solution exists to date. Therefore, we adjust and pose a newer version of our previous question:

*Can we ensure that an auditor, who does not have access to internal data of any organization except of her client-organization, ① can verify the duality of all O2O transactions of her client and ② produce a result that a committee can verify without having access to any internal organization data, efficiently?*

**State-of-the-art.** There exist prior works that indirectly provide a solution to part ① of our problem, however, they operate in a different model. Specifically, zkLedger [43] and Miniledger [19] consider a scenario where *all* organizations maintain a single ledger in a distributed manner that includes *all* transactions in a hiding manner. Nevertheless, their model does not correspond to the real-world alternative, where bookkeeping is being done individually by each organization. E.g., a small enterprise that logs a thousand transactions annually should not need to record any data from all the remaining transactions of other organizations. Another relevant work to our problem revolves around cross-chain bridges [18]. Their core functionality is enabling proof generation of an event that occurred on one chain to be verified on another. This indeed fits into our problem setting, but most current bridges e.g., [7, 8, 9] either suffer from poor performance or rely on central entities.

**Our results.** First, we introduce and formulate the problem of cross-ledger transaction consistency for financial auditing, including the system and threat models, as well as crucial security goals. We then propose two protocols, CLOSC (Cross Ledger cOnsistency with Smart Contracts) and CLOLC (Cross Ledger cOnsistency with Linear Combinations), implement them upon a two-tier ledger/blockchain-based architecture, and provide extensive evaluation results regarding their performance. Additionally, we formally define three privacy and security properties, namely *transaction amount privacy*, *organization-auditor unlinkability*, and *transacting organizations unlinkability*, and prove that both our protocols satisfy them.

CLOSC utilizes smart contracts for storing transaction-related data and proofs from organizations. For each transaction, both organizations deploy a smart contract and fill in their “half” regarding the consistency-checking method, to both smart contracts. In this way, both auditors have all the information needed to verify the consistency between the reported transactions. CLOLC relies on linear combinations with organizations now maintaining a separate list of transactions for each of their transaction counterparts. The consistency checking is performed on an “O2O-pair” basis, where each auditor verifies consistency for each organization their client transacts with individually. To enable this, the committee assigns and distributes to auditors weights for each individual O2O transaction for every transacting pair. Then, for each transacting organization, the auditor calculates the linear combination of the amounts with the corresponding weights and exchanges the result with the counterpart’s auditor. The committee in both CLOSC and CLOLC essentially performs two types of checks: (i) consistency between the reported data from an organization and its auditor, and (ii) reported data from the two auditors. This has a dual purpose: First, when all verifications succeed, this signifies that consistency exists across all organization ledgers, and second, when a verification is unsuccessful, the committee needs only to further investigate the particular O2O transacting organization-pair. Importantly, all checks above are lightweight and do not impose prohibitive overheads for the committee.

We test our system on AWS machines, implement our two-tier ledger architecture over Hyperledger Fabric, and conduct experiments to demonstrate the practicality and scalability of our proposed solutions. Notably, executing an auditing epoch in CLOSC for 1024 organizations, with each of them recording 1M transactions requires on average  $\approx 43$  mins per organization,  $\approx 18$ mins per auditor, and  $\approx 4$ secs for the committee, whereas for the same organizations and total transactions in CLOLC it takes on average  $\approx 30$ mins per organization,  $\approx 39$ mins per auditor, and  $\approx 4$ mins for the committee. In Section 6 we provide the extensive evaluation of both our proposed solutions, for varying number of entities and total transactions per auditing epoch. Additionally, we include a comparative analysis for the performance of our two protocols and also compare against prior works in terms of protocol computation and storage needs complexity, which have “similar-enough” auditability goals to ours.

**Paper organization.** The rest of this paper is organized as follows. In Section 2 we expand on prior works relevant to our problem and in Section 3 we introduce the necessary background for our system and protocol design. Following, we concretely formulate the problem we are focusing on in this work in Section 4. Then, we analyze our system architecture, provide the details of our two protocols (CLOSC and CLOLC), and provide the intuition of how they achieve our three newly proposed security properties in Section 5. In Section 6 we present the implementation details of our protocols and demonstrate their performance. Last, we provide a discussion on the limitations and potential future directions in Section 7, and finally conclude our work in Section 8.

## 2 Related Work

The combination of privacy-enhancing and blockchain technologies has been gaining interest, especially for “traditional” financial applications [14]. A core property of blockchains is immutability and, as a result, multiple researchers have tried to enable/construct blockchain-assisted auditing. Generally, there seems to be a consensus amongst researchers and industry professionals as to the anticipation that blockchains are a disrupting force in the auditing ecosystem and that their role will get increasingly important [10, 47, 35, 22, 24, 12, 21], potentially even shifting the auditing process from backward to forward-looking [17]. However, as pointed out in [27] the vast majority of blockchain-related works do not look into how to utilize blockchains and smart contracts to address the challenges revolving around financial auditing, with a small set of notable exceptions. The authors of [22] propose a triple-booking system where a copy of all records is kept on a blockchain (on top of an existing double-booking protocol). The inclusion of smart contracts in the design of such systems was proposed in [35, 47]. However, in both [35, 47] the access rights/patterns are not clearly outlined and questions arise about how potential leakage can be used for malicious purposes.

### 2.1 Single-ledger approaches

To enforce that during a transaction no entity expends more than the total amount of assets they hold, the authors of zkLedger [43] introduced tailored *proof of assets*, specifically in a banking setting, similar to our O2O scenario. Importantly, this work uses zero-knowledge proofs to enable confidential transactions while (i) allowing for regulatory compliance and auditability and (ii) guaranteeing the condition above without revealing to the other system’s participants anything about transaction amounts. However, zkLedger has not been implemented on any blockchain platform and the proposed protocol does seem not scale well with the number of protocol participating entities. This is mainly due to all

participants needing to maintain the same version of a *single ledger* in a distributed manner with storage complexity  $\mathcal{O}(nm)$ , where  $n$  are the total banks and  $m$  are the total transactions in the system. Additionally, zkLedger cannot support multiple transactions happening in parallel. This derives from the fact that in order for the ledger to accept a transaction, it needs to verify the correctness of all appended proofs to it, rendering the cases where multiple transactions are being submitted concurrently impossible to handle; there needs to be an ordering protocol. The authors of Miniledger [19] overcome the scalability issues by introducing a pruning technique for the ledger in question. However, Miniledger suffers from other shortcomings such as large transaction creation time ( $\approx 5s$  for a single transaction in a setting with 100 banks) and requires both a synchronization and a transaction-ordering protocol, both of which are unrealistic in real-world scenarios where organizations transact asynchronously with each other daily and in the thousands totally. More recently, ACA [37] has been proposed with focus to anonymity, confidentiality, and auditability of transactions. The authors rely on a conventional blockchain layer for recording the transactions which are smaller than zkLedger and Miniledger, however the size of general transactions ( $\approx 3.8KB$ ) and the verification times ( $\approx 1.4sec$ ) are prohibitive for settings as the ones we consider. In Section 6 we show that an entire auditing epoch in both CLOSC and CLOLC takes minutes even for millions of transactions, regardless of the organizations in our system.

## 2.2 Communication between different blockchains

An emerging research area revolves around *cross-chain bridges*, a technique that increases token utility by facilitating cross-chain liquidity between distinct blockchains. Specifically, they enable users to transform their tokens from one blockchain to another, usually by burning or locking existing tokens from the original-chain, and minting or unlocking “new” ones to the target-chain [18]. Essentially bridges utilize messaging protocols that in theory can be used to support arbitrary messages across different chains, usually via smart contracts [7, 8, 9]. Recently, zkBridge [49] has been proposed, a protocol that better the performance of existing bridges while achieving higher security standards. The authors utilize zk-SNARKs for generating proofs for relaying block headers. However, this approach relies on the construction of smart contracts, which is not ideal due to storage and computational costs required. In Section 6 we showcase the difference in efficiency and scalability between our two protocols. CLOSC relies on smart contracts, whereas CLOLC on linear combinations, making clear the trade-offs between them.

## 2.3 Other non-blockchain-based systems with auditing capabilities

Recently, a line of works explores the area of authenticated data structures. Transparency logs are a prominent example when considering auditing. In these works, the goal of an auditor is to examine digests published by untrusted servers to avoid server equivocation (e.g., Merkle<sup>2</sup> [29] and CONIKS [38]). Other works revolve around ensuring token liability when transacting across a system. Specifically, the authors of [30, 46] approach this problem by requiring entities to publish attestations on their total liabilities e.g., on public bulletin boards. Their goal is to safeguard against data-leaking attacks while allowing auditors to verify the validity of statements regarding liabilities. While these works examine auditability, the view on auditing is through a different lens. First, our system design is more complex, auditors examine organizations and then a committee verifies the auditor-generated results. Additionally, our protocols propose an auditing process that revolves around detecting individual transaction (in)consistency, a considerably more challenging task than, for example, proving the sum of an organization’s assets.



### 3 Preliminaries

**General Notation.** Let  $\mathbb{E}$  be an elliptic curve defined over a large prime field  $\mathbb{F}_p$  with  $G, H \in \mathbb{E}$  as publicly known generators. We denote by  $x \leftarrow_{\$} A$  the random sampling of the element  $x$  from the domain  $A$ , and the set  $\{1, \dots, n\}$  by  $[n]$ . We denote by  $\lambda$  a security parameter and by  $\text{negl}(\lambda)$  a negligible function in  $\lambda$ .

**Commitment Schemes [33].** A commitment scheme consists of a pair of PPT algorithms ( $\text{Com.Setup}, \text{Com.Commit}$ ). The  $\text{Com.Setup}$  algorithm generates public parameters  $\text{pp}$  for the scheme, for security parameter  $\lambda$ :  $\text{pp} \leftarrow \text{Com.Setup}(1^\lambda)$ . The commitment algorithm defines a function  $\mathcal{M}_{\text{pp}} \times \mathcal{R}_{\text{pp}} \rightarrow \mathcal{C}_{\text{pp}}$ , for message space  $\mathcal{M}_{\text{pp}}$ , for randomness space  $\mathcal{R}_{\text{pp}}$ , and for commitment space  $\mathcal{C}_{\text{pp}}$ , determined by  $\text{pp}$ . It takes as input a message  $x$  and randomness  $r$  and outputs  $c \leftarrow \text{Com.Commit}_{\text{pp}}(x; r)$ . Specifically, a Pedersen commitment [45] of  $x$  with randomness  $r$  is in the form of  $\text{cm}(x, r) = x \cdot G + r \cdot H$ . Pedersen commitments are *additively homomorphic*, i.e.,  $\text{cm}(x_1, r_1) + \text{cm}(x_2, r_2) = \text{cm}(x_1 + x_2, r_1 + r_2)$ , *computationally binding* (after committing it is not feasible to “change one’s mind”), and *perfectly hiding* (they reveal nothing about the committed data). A computationally binding and perfectly hiding commitment scheme must satisfy the following properties:

- **Computationally Binding:** It is not easy to find two strings  $x_0$  and  $x_1$  that map to the same commitment. More formally, if  $\text{cm}^0 \leftarrow \text{Com.Commit}(x_0; r_0)$ :

$$\Pr[\mathcal{A}(\text{cm}^0) \rightarrow (x_1, r_1) : \text{cm}^0 = \text{Com.Commit}(x_1; r_1)] \leq \text{negl}(\lambda).$$

- **Perfectly Hiding:** It is not easy to identify which value was used in the generation of a commitment  $\text{cm} \leftarrow \text{Com.Commit}(\cdot, \cdot)$ . Formally,  $\forall x_0, x_1$  (of the same length), for all *non-uniform* PPT adversaries  $\mathcal{A}$ , we have that:

$$|\Pr[\mathcal{A}(\text{Com.Commit}(x_0; r_0)) = I] - \Pr[\mathcal{A}(\text{Com.Commit}(x_1; r_1)) = I]| = 0.$$

**Hash Function [23].** A cryptographic hash function  $\mathcal{H} : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$  is pre-image resistant if the probability of reversing the hash output to obtain the underlying pre-image is negligible:  $\Pr[\mathcal{A}(y) \rightarrow x | y = \mathcal{H}(x)] \leq \text{negl}(\lambda)$ .

**Public-Key Encryption (PKE) Scheme.** A PKE scheme  $\mathcal{E}$  consists of a tuple of algorithms  $\text{PKE.KeyGen}, \text{PKE.Enc}, \text{PKE.Dec}$ . Specifically the elliptic curve El-Gamal encryption scheme [31] is as follows:

- $\text{KeyGen}(\lambda) \rightarrow (sk, pk)$ . Given the security parameter  $\lambda$ ,  $\text{KeyGen}$  samples a secret key  $sk \leftarrow_{\$} \{0, 1\}^\lambda$ , computes the public key  $pk = sk \cdot G$ . It outputs the key-pair  $(sk, pk)$ .
- $\text{Enc}(pk, x; r) \rightarrow ct_x$ . To encrypt a value  $x$ , the algorithm takes input a randomness  $r$  and outputs the curve point  $(r \cdot G, P_x + r \cdot (sk \cdot G))$ . Here,  $P_x$  is a publicly-known mapping of a value  $x$  to a curve point in  $\mathbb{E}$ .
- $\text{Dec}(ct_x, sk) \rightarrow x$ . To decrypt  $x$  from  $E(pk, x; r)$ , the algorithm computes  $x := P_x + r \cdot (sk \cdot G) - r \cdot sk \cdot G$ .

Regarding security, we say a PKE scheme is IND-CPA secure whenever an adversary  $\mathcal{A}$  plays an indistinguishability game with a challenger  $\mathcal{C}$  where the former has encryption oracle access and at some point gives a left-or-right challenge  $(x^0, x^1)$  to  $\mathcal{C}$ , who depending on a bit  $b \leftarrow_{\$} \{0, 1\}$  which it had picked during setup returns  $ct^b \leftarrow \text{PKE.Enc}(x^b)$ . Finally,  $\mathcal{A}$  outputs a guess  $b'$  on  $b$  and wins if  $\Pr[b = b'] \geq \frac{1}{2} + \text{negl}(\lambda)$ .

**Merkle Tree [39].** A Merkle tree ( $MT$ ) is a binary tree whose leaf nodes can store any information and each parent node being calculated as the hash of its children. The time complexity and the space complexity to find a data entry in a  $MT$  with  $n$  entries, given its path, is  $O(\log n)$ .  $MT$ s support *membership* proofs  $\pi_m \leftarrow \text{ProofExists}(root_{MT}, x)$ , proving the inclusion of a leaf node  $x$  to a tree whose root is denoted by  $root_{MT}$  and its root-to-leaf-path by  $path_{leaf}$ .  $path_{leaf}$  comprises the siblings of the nodes while traversing from the leaf to  $root_{MT}$ . Additionally, there exists a leaf-inclusion verification algorithm  $0/1 \leftarrow \text{VerifyExists}(root_{MT}, x, \pi_m)$  that enables a verifier to check whether a value  $x$  resides in the merkle tree  $MT$ , given a proof  $\pi_m$  and the root  $root_{MT}$ . Last, one can generate a proof for a set of leaves together using a  $\text{MergeProofs}(\cdot)$  algorithm, containing essentially the set of leaves and their corresponding unique path siblings across the respective tree levels.

**Blockchain & Smart contracts.** A blockchain is a peer-distributed ledger made secure through cryptography and incentives. Peers using consensus mechanisms agree upon which information to store in blocks. Blockchain technology has found other uses apart from cryptocurrency applications, especially via using smart contracts [48]. A smart contract is a computer program that can be run in an on-chain manner, has internal states and its own on-chain storage. Uploading data on a smart contract method requires time to be verified and agreed upon by the blockchain peers. Contrary, reading data from the contract is almost instantaneous and does not require any form of consensus.

## 4 Problem Formulation

Below, we formalize the problem we solve in this work, including the system and threat model revolving around it, and provide corresponding security goals.

### 4.1 System model

Our model includes three entity types, namely organizations, auditors, and a committee. Organizations transact with each other and are responsible to maintain a copy of each and every of their bipartite O2O transactions in a private ledger, to which only their auditor and the committee are privy to (if needed). Auditors examine the transaction records of their clients and are responsible to extract *results* about their reported individual economical activity and share them with the committee. The committee is responsible to verify auditor-provided results and can access organization records upon request.

### 4.2 Threat model

First, we assume each organization to be audited by a single auditor, which is in line with current practices [36]. Notably, even in cases where the same auditing entity e.g., one of the Big Four (KPMG, EY, PwC, Deloitte) audits different organizations, this does not translate to each individual auditor having access to all organization data of each of the company's client. Additionally, we assume no inter-organization or inter-auditor collusions. Remember that such collusions happen in reality and lead to a plethora of auditing scandals which are discovered annually and leading to fines [20]. However, these cases are outside our design rationale since they are impossible to detect macroscopically as described in our system model. E.g., consider a scenario where two organizations transact with each other but do not log this transaction in either of their ledgers. To identify such misreported transactions, their auditors or/and the committee need to perform on-site auditing and compare tangible assets

(e.g., monetary or other asset reserves with the reported ones). While this on-site part is integral in the auditing process as a whole, *we solely focus on the consistency across all organization-reported data*. Other than that, we consider all entities in our system to be malicious, except for the committee, which is trusted. Specifically, on top of misreporting their transaction activity while trying to avoid detection, organizations wish to extract information as to the client-relation between other organizations and auditors, infer amounts of any transactions apart from the ones they are privy to, and check whether cross-ledger transaction consistency holds for other organizations. Adversarial auditors have the same objectives. Such deviant behaviors and attack goals are not only realistic but can also lead to tangible rewards e.g., through insider trading [13].

### 4.3 Security goals and rationale

Based on the threat model above we determine the security goals that our solutions need to achieve for cross-ledger transaction consistency. First, our solutions need to provide a notion of soundness, in the sense that no organization or auditor should be able to misreport data or results and avoid detection<sup>1</sup>, in order to prevent fraudulent asset in/deflation e.g., through the inclusion of sham transactions in their ledger or via generating fake reports [6, 20, 28]. Then, *transaction amount privacy* ensures that only an organization and its auditor should have access to raw transaction data of the former. Financial data is considered to be sensitive and having access unrightfully to an organization's data (e.g., by another organization) may lead to market manipulation via insider trading [13], as mentioned above. Additionally, *transacting organizations unlinkability* guarantees that only the two transacting organizations should have knowledge of the fact that there exists transaction activity between them. Similarly, *organization-auditor unlinkability* ensures that no other system entity can infer any organization-auditor bipartite relation, except the ones they are part of. Last, only auditors and the committee should be able to verify transaction (in)consistency across ledgers. In the auditors' case crucially, only for organizations their client is transacting with.

Although to the best of our knowledge we are the first to consider the transacting organizations and organization-auditor unlinkability properties, there exist prior works that have built their systems and architectures in such a way that implicitly achieves comparable notions of security. zkLedger [43] and Miniledger [19] are perfect examples of this: The respective ledger atop which both these systems are based on essentially satisfies our transacting organizations security property; the hiding property of the commitments employed in their system, ensure the privacy of the transaction amounts. Now, as for the auditor-organization unlinkability, existing industry standards already safeguard the confidentiality of the relation between an auditor and its client. Currently, this type of information might be disclosed after the auditor performs all necessary consistency examinations and produces its report, usually after the fiscal year concludes [44]. Regarding cross ledger consistency verifiability, auditors currently require raw transactional data access of their clients records to carry out respective auditing processes, and organizations carefully pick who has such access. Notably though, enabling other entities to verify cross ledger consistencies could in fact contribute positively to the financial auditing ecosystem, rendering the organizations' activities even more transparent. However, organizations are generally not eager to publicly disclose their financial data to accommodate such verification. Achieving such a notion of

---

<sup>1</sup> Such a property is fairly common in systems like ours and this is why we do not introduce it as a separate property. They are usually achieved through an arbiter, which is the committee in our case.

“public auditability” might need to be coupled with a strong notion of anonymity as the one presented in [32]. This is not trivial to achieve, considering all other challenges financial auditing has regarding accountability and traceability, and we leave this as future work.

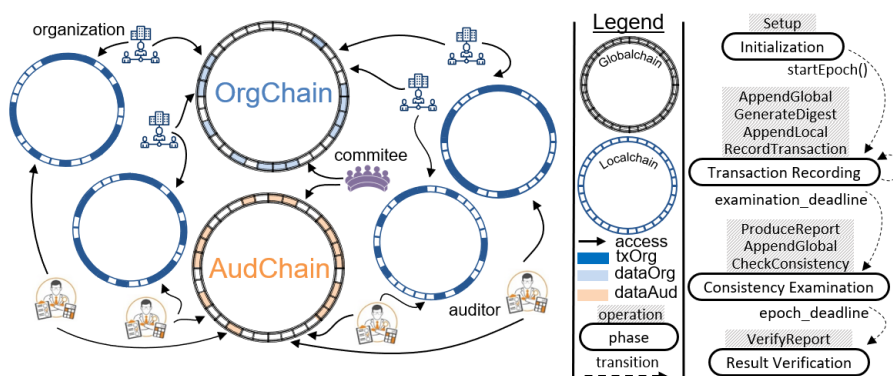
## 5 Our Solution

Our system design is epoch-based and includes all three entities (i.e., organizations, auditors, and a committee). We propose two protocols that exist atop a two-tier ledger architecture. In our solutions, we construct this architecture over blockchains, taking advantage of their immutability to avoid attacks like the one presented in Section 1. Each organization maintains a “local” ledger, to which we refer to as *localchain* – `LocalChain` for storing information related to its O2O transactions (`txOrg`). Additionally, organizations maintain plaintext records of their financial activity offline (e.g., in a database). The committee maintains two “global” ledgers in the form of blockchains, to which we refer to as *globalchains*: one “for organizations”, namely `OrgChain`, where organizations upload localchain-related data (`dataOrg`), and another “for auditors”, namely `AudChain`, who upload report-related data (`dataAud`). Organizations have access to their off-chain ledger, their `LocalChain`, and the `OrgChain`, while auditors have access only to `AudChain` and their client’s `LocalChain`. Recall that the auditor and the committee may access off-chain records if they ask explicitly to examine them to investigate further potential fraudulent behavior. At a high level, an organization stores hiding versions of its transactions on its `LocalChain` and aggregated transaction data on `OrgChain`, while auditors store result-related hiding data on `AudChain`.

**Design rationale.** This architecture enables ① auditors to check the consistency of their clients reported transactional amounts across other ledgers and ② the committee to verify (i) the consistency between the data uploaded between an auditor and its client-organization and (ii) the consistency between the data uploaded between two auditors whose clients have transacted during the epoch. Figure 2 depicts the architectural model our protocols operate in, the phases with their respective operations, and the transitional triggers across phases. Contrary to prior works based on a single-ledger approach (e.g., [43, 19, 37]), our two-tier ledger architecture allows organizations to store data only pertaining to their own transactional activity; reducing considerably the individual storage costs.

We refer to our different ledger tiers (local and global) as chains, since (i) we need a notion of ledger immutability in our system design and (ii) we implement them later as blockchains. However, we stress that these ledgers are not explicitly blockchains and our system is ledger-agnostic at its core. In fact, any immutable ledger may be used in our design. By combining the ledger immutability of our two-tier architecture with cryptographic components, we can guarantee the three security properties mentioned above. More specifically, in both our solutions we employ hiding techniques for `dataOrg` to ensure transaction amount privacy and ensure that no entity except for the Committee can associate the economic activity of any organization to another. Last, by uploading `dataAud` in a hiding manner, no other entity except for the committee can infer relations between organizations and auditors.

**Phases.** Each epoch is comprised of four phases, namely (i) Initialization ( $\mathcal{IN}$ ), (ii) Transaction recording ( $\mathcal{TR}$ ), (iii) Consistency examination ( $\mathcal{CE}$ ), and (iv) Result verification ( $\mathcal{RV}$ ). In the initialization phase the committee performs the necessary setup operations and distributes information to the other entities accordingly. During transaction recording, organizations record data about their O2O transactions to their `LocalChain`, and the

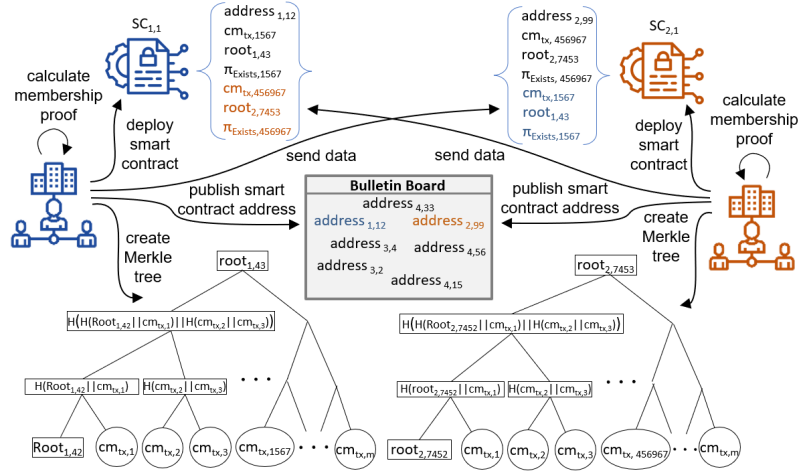


■ **Figure 2** Our architectural two-tier ledger design and access rights of the entities involved, alongside the protocol phases with respective operations and triggers.

**OrgChain.** During the consistency examination phase auditors first perform computations on their clients data and extract results which they upload on the **AudChain**. Afterwards, they compare their results against those from the respective auditors of their clients' transacting counterparts, which are already uploaded on **AudChain**. Finally, the committee during result verification collects data from **OrgChain** and **AudChain**, and conducts checks as to the consistency of the reported/uploaded data. Upon identifying any inconsistencies, the committee may investigate further the ledgers of the suspicious organizations and/or the results produced by their auditors, and potentially assign penalties.

**Protocol preliminaries.** Let  $[n]$  be the index set of the organizations,  $m$  the maximum transactions that can be recorded in an epoch, by a single organization, and  $[l]$  the index set of the Auditors. Therefore, let  $\mathcal{O}_n = \{o_i\}_{i \in [n]}$  denote the set of organizations,  $\mathcal{A}_l = \{a_z\}_{z \in [l]}$  the set of auditors, and  $\text{Com}$  the committee. Let  $\mathcal{L}_i$  denote the respective LocalChain of  $o_i$ ,  $\mathcal{L}_o$  the OrgChain, and  $\mathcal{L}_a$  the AudChain. Let  $\text{Op} = \langle \text{Setup}, \text{RecordTx}, \text{AppendLocal}, \text{GenerateDigest}, \text{CheckConsistency}, \text{ProduceReport}, \text{AppendGlobal}, \text{VerifyReport} \rangle$  be the list of allowed operations. Then, the tuple  $\langle \mathcal{O}_i, \mathcal{A}_l, \text{Com}, \{\mathcal{L}_i\}_{i \in [n]}, \mathcal{L}_o, \mathcal{L}_a, \text{Op} \rangle$  can describe fully both our protocols. Below, we explain the operations at a high level and provide in Sections 5.1 and 5.2 our proposed protocols' specifics. In the detailed description of our protocols, for readability purposes we break down all operations to specific steps.

- **Setup**( $\mathcal{O}_n, \mathcal{A}_l, t$ ):  $\text{Com}$  generates the public parameters  $pp$ , blinding identifiers  $\text{Lid}^{o,b}$  (for organizations) and  $\text{Lid}^{a,b}$  (for auditors), and other protocol-specific parameters  $psp$  for epoch  $t$ .
- **RecordTx**(Sender,Receiver,Value):  $o_i$  invokes this operation to record a transaction  $\text{dataLocal}$  of Value from a Sender or to a Receiver, where  $\{\text{Sender}, \text{Receiver}\} \in \mathcal{O}_n$  and  $\text{Sender} \neq \text{Receiver}$ .
- **AppendLocal**( $\text{tx}, \mathcal{L}_i$ ):  $o_i$  invokes this operation to append a hiding version  $\text{tx}^h$  of a transaction  $\text{tx}$  to its ledger  $\mathcal{L}_i$ .
- **GenerateDigest**( $t, \mathcal{L}_i$ ):  $o_i$  invokes this operation to generate a “digest”  $\text{dataOrg}$  of the state of its ledger  $\mathcal{L}_i$  for epoch  $t$ .
- **CheckConsistency**( $\mathcal{L}_i, \mathcal{L}_a$ ):  $a_z$  checks whether for all transactions  $\in \mathcal{L}_i$  there exists a transaction in the ledger of another  $o_j \in \mathcal{O}_n$  and produces a **Result**.
- **ProduceReport**( $t, \mathcal{L}_i, \text{dataOrg}_i$ ):  $a_i$  invokes this operation to produce a “report”  $\text{dataAud}$  about the consistency between the data on its client's ledger  $\mathcal{L}_i$ ,  $\text{dataLocal}_i$  with  $\text{dataOrg}_i$  for epoch  $t$ .



■ **Figure 3** CLOSC core components and interactions. Each organization stores transactions in a Merkle Tree, then deploys a smart contract where it adds all transaction-related data of a transaction stored as a commitment at the leaf level of the Merkle tree, and uploads the smart contract address to a bulletin board. The other transacting organization performs equivalent steps and uploads their own transaction-related records also to the latter's smart contract.

- $\text{AppendGlobal}(\text{data}^*, \mathcal{L}^*)$ :  $o_i$  invokes this operation to append  $\text{data}^* = \text{dataOrg}$  to  $\mathcal{L}^* = \mathcal{L}_o$  or  $a_z$  to append result-related data  $\text{data}^* = \text{dataAud}$  to  $\mathcal{L}^* = \mathcal{L}_a$ .
- $\text{VerifyResult}(\mathcal{L}_o, \mathcal{L}_a, (\text{id}_{\alpha_z}, \text{dataAud}))$ : Com invokes this operation to verify the consistency of  $\text{dataAud}$  (generated from  $a_z$ ) with  $\mathcal{L}_o$  and  $\mathcal{L}_a$ .

## 5.1 CLOSC (Cross Ledger cOnsistency with Smart Contracts)

Our first protocol utilizes smart contracts as the name indicates. In more details, in addition to an organization ( $i$ ) maintaining a copy of all its transactions offline in a local ledger/database and ( $ii$ ) computing and uploading a hiding copy of each such transaction to its LocalChain, now it needs to ( $iii$ ) maintain a tree structure  $\text{MT}^t$  whose leaves correspond to hiding transactions, ( $iv$ ) upload the root of the tree  $\text{root}_{\text{MT}^t}$  to OrgChain, and ( $v$ ) upload a smart contract on its LocalChain for every transaction. Below we explain both at a high level and explicitly the details of CLOSC.

Let  $\text{tx}_{S,R} = \langle \text{Sender}, \text{Receiver}, \text{Amount}, \text{nonce}, \text{timestamp} \rangle$  describe a transaction tuple, where  $\text{nonce}$  is an O2O-specific transaction unique random identifier. Then, hiding commitments of the form  $\text{cm}_{\text{nonce}}^{\text{timestamp}} = g^{\text{Amount}} \cdot h^{\mathcal{H}(\text{timestamp}, \text{nonce})}$ , are stored at the  $\text{MT}^t$  leaf-level, with the sole exception of the utmost left leaf that is equal to the merkle tree root of the previous epoch. The non-leaf nodes of the tree are calculated as follows:  $\text{node}_i = \mathcal{H}(\text{node}_{lc} || \text{node}_{rc})$ ; essentially the hash of the concatenation of the node's left and right children (denoted by  $lc$  and  $rc$  respectively) also depicted in Figure 3. Importantly, for incoming transactions, we consider  $\text{Amount}$  to be positive, and negative otherwise. Now recall that an auditor should, ideally, be able to verify that: For every organization-related transaction  $\text{tx}_{i,j}$  included in its client's ledger  $\mathcal{L}_i$  there exists a dual transaction in  $\mathcal{L}_j$ , crucially, without having access to  $\mathcal{L}_j$ . We enable auditors to verify this via the following:

When  $o_j$  transacts with  $o_i$  the latter deploys a smart contract on  $\mathcal{L}_i$  and whitelists  $o_j$  to be able to submit data to it, and  $o_j$  operates similarly. These smart contracts will store the committed transactions alongside the proofs that the financial transactions corresponding to the deal have been included to the respective trees whose roots are uploaded to  $\mathcal{L}_o$ .

With this design  $o_i$  can deploy the contract, communicate its “address” to  $o_j$ , upload its data and proof there, and await for its counterpart to upload the other half. The smart contract will store the two instances of  $\{\text{cm}_{tx}, \text{root}_{\text{MT}^t}, \pi_{\text{Exists}}\}$ , with the latter proving that  $\text{cm}_{tx}$  is in the tree whose root is  $\text{root}_{\text{MT}^t}$ . The only potential problem here is that  $o_j$  may never submit its half of the deal and claim not knowing the address. To solve this “plausible deniability” problem, we employ a bulletin board, which is maintained by the committee. In fact, all organizations publish the addresses of their smart contracts there to make sure that any organization can use them. Notably, only the committee can delete information stored on the bulletin board at the end of each epoch to keep storage use low. At the same time this construction ensures that no organization can deny knowing where to upload their data.

Now, only the duality has yet to be shown. For this we require  $o_i$  to commit to the amount  $x_{i,j,k}$  (for transaction with nonce  $k$ ) in the form of  $g^{x_{i,j,k}}$ . In CLOSC,  $\text{cm}_{tx} = \text{cm}_{j,i,k}^{\text{timestamp}} = g^{x_{i,j,k}} \cdot h^{\mathcal{H}(\text{timestamp},j,k)}$ . To ensure cross-ledger transaction consistency, both auditors have to verify the 2 uploaded proofs stored inside each of their client’s smart contracts. We implicitly assume that the two transacting organizations have agreed on a common and unique (timestamp,nonce) pair ahead of time. Last, to assist the committee with the result verification, the auditors utilize a MergeProofs algorithm that combines the individual proofs received from their clients into a single one, that they later on pass to the committee. Figure 4 showcases the protocol details including the on-chain storage, where  $i, j$  refer to organizations,  $k$  to transaction nonces, and  $z$  to auditors.

## 5.2 CLOLC (Cross Ledger cONSistency with Linear Combinations)

This construction lies on linear combinations. Specifically, we aim to exploit the fact that it is infeasible to generate two set of values that, combined with a vector of secret-random “weights”, results in the same weighted sum evaluation. To this end, the committee during initialization samples random values for each potential transaction to be made within the epoch and shares them to the respective auditors. Contrary to CLOSC, organizations now do not need to maintain a tree structure or deploy smart contracts. Instead, to record their transaction, they need to upload to their LocalChain lists of commitments for each transaction they make during the epoch and the (homomorphic) product of all commitments per transacting organization on the OrgChain. Importantly, each organization “masks” this product by multiplying it with a hash of its own unique identifier, assisting in guaranteeing that no other OrgChain participants can identify transacting organization pairs.

The consistency examination phase is comprised of two parts in CLOLC. First, the auditors calculate a weighted sum of the transaction amounts (at the exponent) with the committee-generated values (during the initialization phase) through homomorphic multiplications. Afterwards, each auditor uploads these products to the AudChain alongside specific identifiers, from where the respective auditor of the transacting organization can retrieve all matching products, using the corresponding common identifiers. Auditors can then use these product pairs to verify the consistency between the reported data from their client organizations and their respective {organization, auditor} pair. Upon identifying an inconsistency, the auditor can approach the committee, who can then investigate further. Having the auditors upload their results in a hiding manner assists in guaranteeing that no other auditor can “mix and match” reports on AudChain to identify organization-auditor pairs.

Protocol $(i, j, z \in [n]; k \in [m], epoch = t, timestamp = \text{tstp})$	
$\mathcal{IN}.1$ Generate $r_{a_{z,i}} \leftarrow \mathbb{Z}_p$	<u>Committee</u>
$\mathcal{IN}.2$ Distribute $\text{ID}_A = \{g^{r_{a_{z,i}}}\}_z$	
$\mathcal{TR}.1$ Store $\text{Ltx}_i = \{\text{Sender, Receiver, } x_{i,j,k}, k, \text{tstp}\}_{j,k}$ off-chain	
$\mathcal{TR}.2$ $\text{cm}_{j,k}^{\text{tstp}} = g^{x_{i,j,k}} \cdot h^{\mathcal{H}(\text{tstp}, j, k)}$	
$\mathcal{TR}.3$ Upload $\text{Lcm}_i = \{\text{cm}_{j,k}^{\text{tstp}}\}_{j,k}$ to LocalChain	
$\mathcal{TR}.4$ Create $\text{MT}_i^t$ , with $\text{root}_{\text{MT}_i^t}$	<u>Organization</u>
$\mathcal{TR}.5$ Upload $\text{root}_{\text{MT}_i^t}$ to OrgChain	
$\mathcal{TR}.6$ $\pi_{j,k}^m \leftarrow \text{ProveExists}(x_{i,j,k}, \text{root}_{\text{MT}_i^t})$	
$\mathcal{TR}.7$ Deploy $\text{SC}_{i,j,k}$ on LocalChain	(1 <sup>st</sup> part)
-----	
$\mathcal{TR}.8$ Store $\{\text{cm}_{j,k}^{\text{tstp}}, \text{root}_{\text{MT}_i^t}, \pi_{j,k}^m\}$ in $\text{SC}_{i,j,k}$ and $\text{SC}_{j,i,k}$	(2 <sup>nd</sup> part)
$\mathcal{CE}.1$ $b_{i,j,k} \leftarrow \text{VerifyExists}(\text{cm}_{i,j,k}^{\text{tstp}}, \text{root}_{\text{MT}_i^t}, \pi_{i,j,k}^m), \forall \text{SC}_{i,j,k}$	
$\mathcal{CE}.2$ $b_{j,i,k} \leftarrow \text{VerifyExists}(\text{cm}_{j,i,k}^{\text{tstp}}, \text{root}_{\text{MT}_i^t}, \pi_{j,i,k}^m)$	
$\mathcal{CE}.3$ $B = \prod_{(j,k)=(1,1)}^{(n,m)} b_{i,j,k} \cdot b_{j,i,k}$ , if $B = 0$ alert Com	
$\mathcal{CE}.4$ Check $\prod_{k=1}^m \text{cm}_{i,j,k}^{\text{tstp}} \cdot \text{cm}_{j,i,k}^{\text{tstp}} \cdot h^{-\mathcal{H}(\text{tstp}, j, k) - \mathcal{H}(\text{tstp}, i, k)} \stackrel{?}{=} 1$	
$\mathcal{CE}.5$ $\text{cm}_{i,j}^t = g^{r_{a_{z,i}}} \cdot \prod_{k=1}^m \text{cm}_{i,j,k}^{\text{tstp}}$	<u>Auditor</u>
$\mathcal{CE}.6$ $\pi_{i,j}^{m'} \leftarrow \text{MergeProofs}(\{\pi_{i,j,k}^m\}_k), \{\mathcal{H}(\pi_{i,j}^{m'})\}_j$	
$\mathcal{CE}.7$ Upload $\{\text{cm}_{i,j}^t, \mathcal{H}(\pi_{i,j}^{m'})\}_j$ to AudChain	
$\mathcal{CE}.8$ Forward $\{\text{Lcm}_{i,j}\}_k = \{\text{cm}_{i,j,k}^{\text{tstp}}\}_k$ and $\{\pi_{i,j}^{m'}\}_j$ to Com	
$\mathcal{RV}.1$ $\{\text{cm}_{i,j,k}^{\text{tstp}}\}_k = \text{Lcm}_{i,j}$	<u>Committee</u>
$\mathcal{RV}.2$ $b'_{i,j} \leftarrow \text{VerifyExists}(\text{Lcm}_{i,j}, \text{root}_{\text{MT}_i^t}, \pi_{i,j}^{m'})$	
$\mathcal{RV}.3$ Check $B' = \prod_{(i,j)=(1,1)}^{(n,n)} b'_{i,j} \cdot b_{j,i} \stackrel{?}{=} 0$	
$\mathcal{RV}.4$ Check $\prod_{i=1}^n \prod_{j=1}^n \text{cm}_{i,j}^t \cdot \text{cm}_{j,i}^t \stackrel{?}{=} g^{r_{a_{z,i}}} \cdot g^{r_{a_{z,j}}}$	
LocalChain: $\text{Lcm} = \{\text{Lcm}_i\}_{j,k}$ , $\text{Lsc} = \{\text{SC}_{i,j,k}\}_{j,k}$	
OrgChain: $\{\text{root}_{\text{MT}_i^t}\}_i$	
AudChain: $\{\text{cm}_{i,j}^t, \mathcal{H}(\pi_{i,j}^{m'})\}_{i,j}$	

■ **Figure 4** CLOSC phase and on-chain storage analysis. Regarding operations, Setup: $\mathcal{IN}$ , RecordTx: $\mathcal{TR}.1, 2$ , AppendLocal: $\mathcal{TR}.3, 7, 8$ , GenerateDigest: $\mathcal{TR}.4, 6$ , CheckConsistency: $\mathcal{CE}.1-6, 8$ , AppendGlobal: $\mathcal{TR}.5, \mathcal{CE}.7$ , VerifyResult: $\mathcal{RV}.1-4$ .

The committee verifies results by accessing only OrgChain and AudChain, meaning it requires no access to any of the underlying localchain commitments, unless it specifically asks for them, e.g., to cross-check any data regarding an auditor-reported potential inconsistency. We enable the committee to carry out result verification checks by ensuring that (i) the data organizations upload to OrgChain are consistent with the data their auditor uploads to AudChain and (ii) the data auditors upload to AudChain are consistent with each other. Importantly, we tie each auditor-generated commitment sum with the unique identifier of said auditor so as to avoid possible brute-force matching attacks, since these commitments ( $\text{cm}_{i,j}^t$ ) are uploaded on AudChain. Importantly, the linear combination with random values is what on one side assists the auditors in their duties but at the same time hides the transactional amounts from other participants. The infeasibility of reversing these linear combinations resides at the heart of this protocol, which has been used previously in other application contexts as well e.g., for oblivious linear-function evaluation [15], or function secret sharing [16]. In Figure 5 we showcase the CLOLC protocol phases in detail alongside which elements are stored on-chain.



Protocol $(i, j, z \in [n]; k \in [m])$	
$\mathcal{LN}.1) \{r_{i,j,k}\}_{i,j,k} \xleftarrow{\$} \mathbb{Z}_p, r_{i,j,k} = r_{j,i,k}$ $\mathcal{LN}.2) \{id_i\}_{i \in [n]} \xleftarrow{\$} \mathbb{Z}_p, \{id_z\}_{z \in [l]} \xleftarrow{\$} \mathbb{Z}_p$ $\mathcal{LN}.3) sk_{com}, \{sk_i\}_{i \in [n]} \xleftarrow{\$} \mathbb{Z}_p, pk_{com} = g^{sk_{com}}, pk_i = g^{sk_i}$ $\mathcal{LN}.4) \text{Publish } pk_{com}, \{pk_i\}_i$ $\mathcal{LN}.5) \text{Forward } \{r_{i,j,k}\}_{i,j,k}, \{sk_i\}_i \text{ to Auditor } z$	<u>Committee</u>
$\mathcal{TR}.1) \text{com}(x_{i,j,k}) = g^{x_{i,j,k}} \cdot h^{\rho_{i,j,k}}, \{\rho_{i,j,k}\}_{i,j,k} \xleftarrow{\$} \mathbb{Z}_p$ $\mathcal{TR}.2) A_{i,j} = \prod_{k=1}^m \text{com}(x_{i,j,k}) = g^{\sum_{k=1}^m x_{i,j,k}} \cdot h^{\sum_{k=1}^m \rho_{i,j,k}}$ $\mathcal{TR}.3) \text{Upload } \text{Ltx}_{i,j} = \{\text{com}(x_{i,j,k})\}_k \text{ to LocalChain}$ $\mathcal{TR}.4) \text{Upload } \{\mathcal{H}(id_i) \cdot A_{i,j}\}_j \text{ to OrgChain}$	<u>Organization</u>
$\mathcal{CE}.1) \text{Read from LocalChain } \text{Ltx}_{i,j}$ $\mathcal{CE}.2) \{\text{res}_{i,j}\}_j = \left\{ \prod_{k=1}^m (\text{com}(x_{i,j,k})^{r_{i,j,k}}) \right\}_j = \left\{ g^{\sum_{k=1}^m x_{i,j,k} \cdot r_{i,j,k}} \cdot h^{\sum_{k=1}^m \rho_{i,j,k} \cdot r_{i,j,k}} \right\}_j$ $\mathcal{CE}.3) B_{i,j} = h^{r_{i,j,f}}, r_{i,j,f} = -\sum_{k=1}^m \rho_{i,j,k} \cdot r_{i,j,k}$ $\mathcal{CE}.4) \text{Read from OrgChain } \{Y_{i,j} = \mathcal{H}(id_i) \cdot A_{i,j}\}_j$ $\mathcal{CE}.5) \{A_{i,j} = Y_{i,j} \cdot \mathcal{H}(id_i)^{-1}\}_j \text{ and } \{C_{i,j} = \text{res}_{i,j}^{-1} \cdot A_{i,j}\}_j$ $\mathcal{CE}.6) \{D_{i,j}\}_j = \left\{ g^{-\sum_{k=1}^m x_{i,j,k}} \cdot h^{\sum_{k=1}^m \rho_{i,j,k} \cdot r_{i,j,k} - \sum_{k=1}^m \rho_{i,j,k}} \right\}_j$ $\mathcal{CE}.7) ct(\text{res}_{i,j}) \leftarrow \text{Enc}(pk_j, \text{res}_{i,j}), ct(B_{i,j}) \leftarrow \text{Enc}(pk_j, h^{r_{i,j,f}})$ $\mathcal{CE}.8) ct(C_{i,j}) \leftarrow \text{Enc}(pk_{com}, \text{res}_{i,j} \cdot A_{i,j}), ct(\mathcal{H}(id_z) \cdot D_{i,j}) \leftarrow \text{Enc}(pk_{com}, \mathcal{H}(id_z) \cdot D_{i,j})$ $\mathcal{CE}.9) \text{Upload to AudChain:}$ $\text{Lres}_i = \{\mathcal{H}(id_i   \sum_{k=1}^m r_{i,j,k}), ct(\text{res}_{i,j}), ct(B_{i,j}), ct(C_{i,j}), ct(\mathcal{H}(id_z) \cdot D_{i,j})\}_j$	<u>Auditor</u> (1 <sup>st</sup> part)
$\mathcal{CE}.10) \mathcal{H}(id_i   \sum_{k=1}^m r_{i,j,k}), \forall j \in [n]$ $\mathcal{CE}.11) \text{Retrieve and decrypt from AudChain } \{\text{res}'_{i,j}, B'_{i,j}\}_j$ $\mathcal{CE}.12) \text{Check } \text{res}'_{i,j} \cdot B'_{i,j} \cdot \text{res}_{i,j} \cdot B_{i,j} \stackrel{?}{=} 1, \forall j \in [n]$	(2 <sup>nd</sup> part)
$\mathcal{RV}.1) \text{Read } \{\mathcal{H}(id_i) \cdot A_{i,j}\}_j \text{ from OrgChain}$ $\mathcal{RV}.2) \text{Read from AudChain and decrypt } \{ct(B_{i,j}), ct(C_{i,j}), ct(\mathcal{H}(id_z) \cdot D_{i,j})\}_{i,j}$ $\mathcal{RV}.3) \text{Check } \forall (i, z): \mathcal{H}(id_i) \cdot A_{i,j} \cdot B_{i,j} \cdot \mathcal{H}(id_z) \cdot D_{i,j} \stackrel{?}{=} \mathcal{H}(id_i) \cdot \mathcal{H}(id_z)$ $\mathcal{RV}.4) \text{Check } \forall (z, z'): \mathcal{H}(id_z) \cdot D_{i,j} \cdot C_{i,j} \cdot \mathcal{H}(id_{z'}) \cdot D'_{i,j} \cdot C'_{i,j} \stackrel{?}{=} \mathcal{H}(id_z) \cdot \mathcal{H}(id_{z'})$	<u>Committee</u>
$\text{Localchain: } \text{Ltx}_{i,j} = \{\text{com}(x_{i,j,k})\}_k = \{g^{x_{i,j,k}} \cdot h^{\rho_{i,j,k}}\}_{i,j,k}$ $\text{OrgChain: } \{\{\mathcal{H}(id_i) \cdot g^{\sum_{k=1}^m x_{i,j,k}} \cdot h^{\sum_{k=1}^m \rho_{i,j,k}}\}_j\}_i$ $\text{AudChain: } \text{Lres}_{i,j} = \left\{ \left\{ \mathcal{H}(id_i \cdot \sum_{k=1}^m r_{i,j,k}), ct_{pk_j}(\text{res}_{i,j}), ct_{pk_j}(h^{r_{i,j,f}}) \right\}_j \right\}_i$ $\left\{ \mathcal{H}(id_z) \cdot g^{-\sum_{k=1}^m \rho_{i,j,k} \cdot r_{i,j,k}} \cdot g^{-\sum_{k=1}^m x_{i,j,k}} \cdot h^{\sum_{k=1}^m \rho_{i,j,k} \cdot r_{i,j,k} - \sum_{k=1}^m \rho_{i,j,k}} \right\}_{(i,j)}$ $\left\{ g^{\sum_{k=1}^m x_{i,j,k} \cdot r_{i,j,k} - \sum_{k=1}^m x_{i,j,k}} \cdot h^{\sum_{k=1}^m \rho_{i,j,k} \cdot r_{i,j,k} - \sum_{k=1}^m \rho_{i,j,k}} \right\}_{(i,j)}$ $\left\{ g^{-\sum_{k=1}^m x_{i,j,k}} \cdot h^{\sum_{k=1}^m \rho_{i,j,k} \cdot r_{i,j,k} - \sum_{k=1}^m \rho_{i,j,k}} \right\}_{(i,j)}$	

■ **Figure 5** CL0LC phase and on-chain storage analysis.  $X'$  denotes a retrieved/decrypted value  $X$ , generated by other auditors or organizations. Regarding operations, Setup: $\mathcal{LN}$ , RecordTx: $\mathcal{TR}.1$ , AppendLocal: $\mathcal{TR}.3$ , GenerateDigest: $\mathcal{TR}.2$ , CheckConsistency: $\mathcal{CE}.1-7, 9, 10$ , AppendGlobal: $\mathcal{TR}.4, \mathcal{CE}.8$ , VerifyResult: $\mathcal{RV}.1-4$ .

### 5.3 Protocol Considerations

While we present our solution assuming all organizations participate in our system honestly and in their entirety, there are multiple real-world scenarios where this is not the case and below we analyze such cases. Specifically we focus on ① the synchronization requirements of our protocols regarding their epochs and ② the relaxation of the percentage of participating organizations in our proposed solutions.

① Our two protocols have different behaviors regarding epoch deadlines and synchronization. In CLOSC, for a pair of auditors to be able to check the consistency of records across the reported data from their two client-organizations, the only requirement is that both Merkle trees need to be finalized and their corresponding roots uploaded on the `OrgChain`. This is in line with current practices where organizations need to submit their reports by the end of the fiscal year in their respective environment they operate in. In cases of differences in these deadlines the checks can be carried out at the latest deadline available, however, we argue that this is not a considerable caveat, especially considering the time-efficiency of the consistency examination and result verification phases in CLOSC. Contrary, in CLOLC the only requirement revolves around a mutually agreed-upon deadline between each two transacting organizations. For example, the auditor of organization A may be able to execute the consistency examination for organization B at time  $t_{AB}$ , and for organization C at time  $t_{AC}$ , with  $t_{AB} \neq t_{AC}$ . The same restrictions apply for the committee as well; however, both auditors now need to have performed their examinations before a specific deadline  $t'$ .

② Regarding the functionality of the auditing process when a subset of all system-wide organizations does not participate in our system, we make an important observation. First, neither CLOSC nor CLOLC have an “all or nothing” requirement, meaning that the crucial phases of consistency examination and result verification can essentially be executed correctly regardless of how many organizations choose to record their transactions in our proposed manners. The only obvious relaxation in such cases, however, is that transaction consistency and result verification can be conducted only amongst the organizations that opt to participate in all phases of our protocols. This indicates that auditors and the committee can utilize a hybrid approach: First, they can utilize both our protocols for all organization-transacting pairs participating in them. Furthermore, they can use traditional methods that, however, currently do not provide our proposed security properties (which are outlined below).

## 5.4 Security Analysis

We aim to design security properties that safeguard our protocols against the various adversarial behaviours of the entities involved in our system design. To this end we introduce three security game-based definitions to fully capture the goals described at a high level in Subsection 4.3 and for which we provide an overview below. Importantly, depending on the property, we consider organizations and auditors to assume the role of the adversary, who may also choose to corrupt a set of organizations and auditors.

Our first property revolves around *transaction amount privacy*. The adversary selects a set of organizations to corrupt and submits to the challenger a pair of transaction vectors. The latter selects to utilize one of them and the adversary should not be able to distinguish between the two cases. Next, we present *organization-auditor unlinkability*. The adversary here firstly selects to corrupt a set of entities and then two distinct non-corrupted organizations and auditors. Following, the challenger “matches” each organization with a respective auditor arbitrarily and the adversary should not be able to distinguish which of the two selected organizations is paired with which of the two selected auditor. Last, we describe the notion of *transacting organizations unlinkability*. Similarly to the previous property, the adversary corrupts a set of entities and then picks four distinct non-corrupted organizations and the challenger pairs them arbitrarily. The goal of the adversary here is to identify these pairs.

Notably, in all above properties the unlinkability or the privacy holds regarding external parties. Recall that the committee may have plaintext access upon submitting such a request, and each auditor has access to all localchain data of its client. To the best of our knowledge, we are the first to define formally system-wide security properties for financial auditing

processes. We defer the reader to the full version of our work [34] for the formal definitions and provide proofs about how our constructions satisfy them under various security assumptions regarding the underlying cryptographic components of our designs.

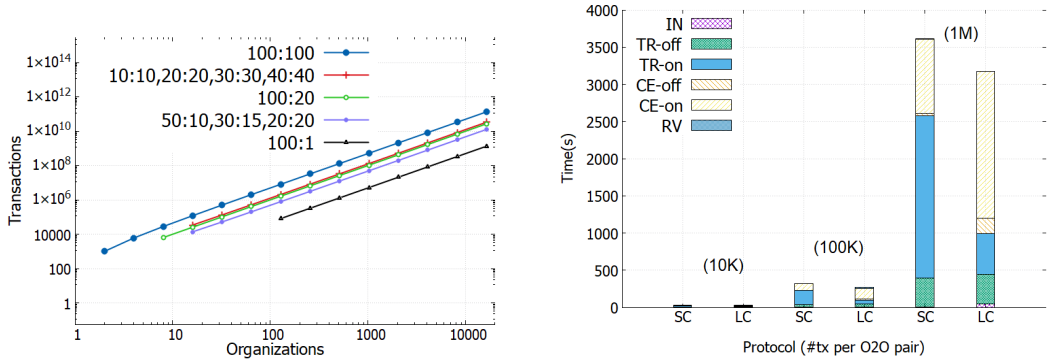
## 6 Implementation & Experimental Evaluation

We develop working prototypes of both our cross-ledger consistency checking protocols and measure their performance against different choice of parameters regarding the number of organizations, auditors, and transactions during an epoch. We implement our two-tier blockchain architecture over Hyperledger Fabric [3] and all off-chain components using Golang 1.20.5. We use the stable 2.5.1 HLFabric version with the Raft consensus algorithm with three orderers for all blockchains. All off-chain computations in CLOSC and CLOLC are implemented over the Edwards25519 Elliptic Curve (EC), and specifically we utilize the EC Library of [1]. Below we report on the performance of each individual component of our prototype. Both the implementation and the experimental evaluation results are available at <https://github.com/auti-project>. All off-chain components evaluation and blockchain experiments were conducted on an 8-core AWS EC2 instance (m5n.2xlarge) running Ubuntu 20.04, with 32GB RAM and 50GB storage. For LocalChain experiments, we consider having 2 peer nodes (for the organization and the auditor) in the network whereas for OrgChain and AudChain we utilized 8 peer nodes. All nodes, including peers and orderers, are tested as Docker containers within the same network, on a single EC2 instance. We conduct our experiments for a varying number of organizations and our numbers are taken as a mean of 10 runs with standard deviation of  $< 5\%$ . Last, we assume that each organization has a unique auditor, which is in fact the worst-case scenario from the implementation perspective. We use the MT library of [github.com/txaty/go-merkletree](https://github.com/txaty/go-merkletree) and to further improve the our protocols' performance we configure our chaincodes to accept batch transaction submitting. Specifically, we bundle 5K transactions together and upload them via a single invoke request.

Below, we present the evaluation of both our constructions across different scenarios and compare their complexity against works with similar potential capabilities. Specifically, through this process we aim to showcase the practicality and scalability of our proposed solutions in terms of the time and space required for varying total number of entities involved and total number of transactions that are recorded during an epoch.

### 6.1 Performance evaluation of CLOSC

We measure all time and space requirements for our proposed solution. Recall that this is a smart contract based approach, where each transaction is stored doubly in two smart contracts on the respective Localchains, in addition to an offline organization Merkle tree. Below we analyze the performance of each protocol phase. Initialization takes  $\approx 1s$  for 1024 organizations, scaling linearly with their number. During transaction recording, computing 1M transaction commitments takes  $\approx 7mins$ , while uploading them to the respective LocalChain takes an extra  $\approx 8mins$ . The resulting on-chain storage for 1M transaction commitments is 432MB and reading all this data requires  $\approx 1min$ . Generating a Merkle tree with depth 20 (for  $\approx 1M$  commitments), whose leaves are the commitments above, along with all individual membership proofs takes  $\approx 2s$ . Last, deploying  $SC_{i,j,k}$  and uploading the triple  $\{cm_{j,k}^{\text{timestamp}}, root_{MT_i^t}, \pi_{j,k}^m\}$  on it takes  $\approx 64s$ , and for 1M transactions the total on-chain storage overhead is 515GB per LocalChain. Auditors or the committee wishing to read all this data can do so in  $\approx 16s$ . As for consistency examination, verifying 1M membership proofs and computing  $cm_{i,j}^t$  and  $\pi_{i,j}^{m'}$  takes  $\approx 28s$ , while computing the



(a) Total transactions in CLOLC for a variant number of organizations and settings ( $X : Y$  signifies that  $X\%$  of the total organizations are transacting with the  $Y\%$  of the rest, with 1024 O2O txs per pair).

(b) E2E time across all different phases for CLOSC(SC) and CLOLC(LC), for different number of transactions (10K, 100K, and 1M).

■ **Figure 6** Scalability behavior of CLOSC and CLOLC in terms of total transactions and time depending on different number of transacting pairs and the amount of their bipartite pairwise transactions.

merged proof takes  $\approx 2.1s$ . Uploading 256  $\{cm_{i,j}^t, \mathcal{H}(\pi_{i,j}^{m'})\}$  tuples on Audchain takes  $\approx 2.3s$  and requires 556KB. Finally, the committee can perform the result verification phase for 1M transactions in  $\approx 4s$ , for a Merkle tree of depth 20.

## 6.2 Performance evaluation of CLOLC

Following, we present the respective time and space requirements of our second solution. Recall that this approach is based on checking the consistency per transacting organization pairs and we make the “worst-case assumption” that all organizations will transact with all others. In such a case we report that for 256 organizations the initialization phase takes  $\approx 154s$ , for 1024 maximum transactions per organization pair. We observe that in CLOLC each organization’s computational expenses grow linearly with the number of its total transactions and thus, for example, a case where 2 organizations sign 100 transactions with each other is equivalent to a case where 4 organizations sign 25 transactions with each other, from a system design perspective. Notably, this holds especially during the initialization phase, where the committee needs to generate a unique identifier for each different transaction the system can support. Therefore, we pick several different settings where, instead of assuming solely that all organizations transact with each other, we explore other, more realistic cases. Figure 6a depicts the total number of transactions per number of organizations in different settings CLOLC can support per epoch, with 1024 O2O transactions per organization pair<sup>2</sup>. At a high level, we chose the following 5 representative settings: (i) each organization transacts with all others, (ii) organizations are operating in fully connected transaction pair islands, (iii) each organization transacts with the same (smaller) portion of the others, (iv) fewer organizations transact with more of the rest, while most organizations transact with a small portion of the total organizations, and (v) each organization transact with the same (small) portion of the rest. Importantly, we observe that the initialization costs scale linearly with the number of maximum transactions, which results to the following: Having 256 organizations, each conducting 1024 transactions with all other organizations ( $\approx 33,5M$  txs in total) is

<sup>2</sup> We do not provide a similar analysis for CLOSC, since its cost of  $\mathcal{IN}$  is not affected by such metrics.

■ **Table 1** Computational times (off-chain + *on-chain*) and blockchain storage needs of CLOSC and CLOLC per epoch for 256 organizations, 1024 max transactions per epoch and organization pair, for a total of 33,423,360 max total transactions per epoch.

(a) Computational times per phase per organization and epoch.

Protocol	$\mathcal{IN}$	$\mathcal{TR}$	$\mathcal{CE}$	$\mathcal{RV}$
CLOSC	52ms	103.2s + <u>299.8s</u>	8.7s + <u>212.2s</u>	3.6s
CLOLC	153.6s	105.5s + <u>134.9s</u>	12.8s + <u>7.6s</u>	12.3s + <u>8.2s</u>

(b) Blockchain storage needs per organization and epoch.

Protocol	LocalChain	OrgChain	AudChain
CLOSC	704MB	543KB	42.6MB
CLOLC	201MB	28.9MB	134MB

approximately equivalent to having a setting with 100K organizations, where on average each organization transacts with 20 others and sign a total of 33 txs over the epoch. This is a rather realistic scenario, especially when epochs have short duration e.g., a week or a month. Next, transaction recording has both off-chain and on-chain parts. Generating 1M committed transactions take  $\approx 402s$ , while accumulating them in  $A_{i,j}$  takes  $\approx 1s$ . Uploading this list of committed transactions on LocalChain takes  $\approx 544s$  and requires 767MB, while fetching it takes  $\approx 28s$ . As for OrgChain, things are similar. Uploading 1M accumulated values takes  $\approx 835s$  and 436MB, while reading all these values takes  $\approx 36s$ . During consistency examination, generating combined all  $\{res_{i,j}, B_{i,j}, C_{i,j}, D_{i,j}\}$  takes  $\approx 690ms$  and encrypting  $\approx 1.4ms$ , for 1024 underlying transactions. Uploading 1M such encrypted tuples takes  $\approx 39mins$  and 2GB, while retrieving, decrypting and checking the consistency takes  $\approx 0.5s$  per transacting O2O pair. Last, for result verification, Com conducts both checks in  $\approx 2s$ .

### 6.3 Comparing the two protocols

As showcased in Tables 1a & 1b, overall there is no clear “better” solution and we observe a trade-off in our designs as for the required time, communication, and on-chain storage. CLOSC has a faster initialization phase, requiring also less communication between the committee and the rest of the entities. Contrary, during transaction recording CLOLC outperforms significantly CLOSC, as the latter requires a separate smart contract per recorded transaction. The performance is reversed once again during consistency examination. As CLOLC splits this phase into two parts, it takes more time to produce the auditor results including all auxiliary elements the committee needs for the next phase. However, in CLOSC the auditor forwards the vector of transactions to the committee, incurring considerably higher communication costs. Last, during result verification both our protocols are very efficient allowing our designs to scale regardless of the number of organizations, auditors, or total recorded transactions.

In Figure 6b we depict the performance of CLOSC and CLOLC across all phases, for varying amounts of supported transactions per epoch, and in terms of the time each protocol requires for the execution of an auditing epoch. We observe that the performance of both our protocols is linearly affected by the number of total transactions per auditing epoch. This can offer a point of flexibility depending on the number of organizations and the frequency of their transaction rates and their auditors’ consistency examination responsibilities. Since for each distinct organization and auditor the transaction recording and consistency examination phases are inherently independent from other entities, the time required is calculated as

■ **Table 2** Protocol phase complexity and ledger storage asymptotics of CLOSC and CLOLC against other works with similar potential capabilities across comparable phases of our protocols.  $n$  and  $m$  respectively denote the total number of organizations and transactions per auditing epoch.  
 \*: Miniledger has a pruning technique to regularly minimize the size of the ledger.

Protocol	$\mathcal{IN}$	$\mathcal{TR}$	$\mathcal{CE}$	$\mathcal{RV}$	Storage
ZkLedger [43]	$\mathcal{O}(n)$	$\mathcal{O}(nm)$	$\mathcal{O}(nm)$	—	$\mathcal{O}(n^2m)$
Miniledger [19]	$\mathcal{O}(n)$	$\mathcal{O}(nm)$	$\mathcal{O}(nm)$	—	$\mathcal{O}(n^2m)^*$
ACA [37]	$\mathcal{O}(n)$	$\mathcal{O}(nm)$	$\mathcal{O}(nm)$	—	$\mathcal{O}(nm)$
CLOSC	$\mathcal{O}(n^2)$	$\mathcal{O}(nm + m \log m)$	$\mathcal{O}(nm)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2m)$
CLOLC	$\mathcal{O}(n^2m)$	$\mathcal{O}(nm)$	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2 + nm)$

the necessary time for one of each entities to execute all operations during each phase (e.g., how long it take for an organization to perform the transaction recording phase). Last, the storage required for both our solutions is not prohibitive, considering also that after the auditing epoch ends there is no need to keep all the uploaded data in any data structure as a whole. For historicity, maintaining a small digest (e.g., a hash output) of the ledgers might suffice to enable post-hoc verification For example, an organization can set the root of the Merkle tree of the previous epoch as the left-most leaf of the tree of the next auditing epoch.

#### 6.4 Comparing with other works

To the best of our knowledge there exists no other implemented system that concretely supports auditing protocols for cross-ledger transactions consistency. Nevertheless, there exist approaches that could serve similar purposes if adjusted appropriately. E.g., the authors of [43, 19, 37] proposed systems close to our design in some aspects regarding auditability/verifiability but they are based on different setups, including solely users and auditors. We provide a comparative analysis in terms of computation and storage asymptotics between our protocols and these works for a single epoch in Table 2.

Notably, CLOSC and CLOLC have higher initialization ( $\mathcal{IN}$ ) complexity since more parameters need to be generated in order to speedup the consistency examination ( $\mathcal{CE}$ ) and result verification ( $\mathcal{RV}$ ) phases of our protocols. Regarding transaction recording ( $\mathcal{TR}$ ) and considering that [43, 19, 37] employ a single-ledger approach, in ACA the transacting entity posts a single transaction pertaining solely to the transfer of its funds, while in zkLedger and Miniledger it also computes and uploads “dummy” transactions for the rest. In CLOSC the dominant term corresponds to the creation of the smart contracts, following by the Merkle tree creation time, whereas in CLOLC organizations create a linear combination of their records per transaction pair. During auditing, all proposed solutions parse through the ledger transaction records, except for CLOLC, where organizations have reported a digest of their entire records, rendering the consistency examination faster. Last, prior works do not consider an  $\mathcal{RV}$  phase. In both CLOSC and CLOLC the committee performs only multiplications and hashing operations during this protocol phase, introducing a negligible overhead in the total required computational time as shown in Table 1a.

Regarding storage requirements, the single-ledger approaches record each and every transaction in the communal ledger, and which all participants need to store in its entirety. To combat growing ledger-storing costs, one of the major caveats of employing blockchains in any system design, Miniledger employs a pruning technique. By doing so, it renders the total storage required ultimately less than in zkLedger and ACA. However, this pruning is not instantaneous and needs to be verified by all other participants. An important diversification

in our designs is the following: Even though at a system level the storage is similar in terms of complexity to prior approaches, each organization only records their own activity on `Localchains`, leading to significantly less storage requirements individually as shown in Table 1b. Additionally, after `RV` concludes only a small digest might be needed to remain from each ledger (including `Orgchain` and `Audchain`) for historicity purposes, as mentioned above. Overall, the costs in `CLOSC` and `CLOLC` are comparable or lower than the other works whilst not only guaranteeing our three proposed security properties but also operating in a more relaxed security setting.

## 7 Discussion

**On protocol generalization and limitations.** Cross-ledger transaction consistency is an important check auditors can use to detect fraudulent behavior of their client-organizations e.g., reported sham transactions or inflated assets. We believe that our protocols can be further augmented to facilitate execution and verification of other popular financial auditing processes e.g., 4-way matching [5]. A possible direction to enable such a process could be for every organization to utilize the same specific SNARK construction that would take as input the four transaction commitments that satisfy the 4-way matching relation and output a proof about their consistency. Based on current practices we expect such an addition to behave similarly (from a computation perspective) to our proposed solutions. Last, we believe that incorporating atomic cross-chain exchange techniques like the ones in [42] on top of our auditability protocols may result in a system with even further capabilities. However, this remains challenging as the entity setting is different and additional processes will need to be designed. As for limitations, in `CLOSC` the committee currently needs to receive all commitments to be able to verify the auditor-generated results and in `CLOLC` it needs to generate a randomness per transaction per epoch during the initialization; recall that this is essentially a trade-off that allows auditors to perform the consistency examination phase more efficiently. Even though these limitations are not prohibitive in terms of performance as demonstrated in our previous evaluation, we leave improving and uplifting our protocols from these restrictions as future work.

**On the inclusion of blockchains in our designs.** Arguably, our architecture does not need to rely on blockchains, however, a two-level generic ledger approach is not sufficient either. Blockchains are immutable, rendering attack scenarios such as the one outlined in Section 1 impossible. Furthermore, we believe that building our system atop a blockchain with smart contract capabilities architecture will enable the inclusion of more auditing processes in the future. However, space and monetary costs are the main concerns when deploying blockchain systems. In our case, after epochs are concluded, the relevant data stored in all blockchains can be deleted, since it will not be used in any of the following epochs and no need on-chain asset transferring is executed either. Last, based on the above discussion and since the problem we are focusing on requires different entities to have access to different data, we observe that a permissioned network fits best our design. Nevertheless, we identify that designing our system atop a permissionless architecture while maintaining our three proposed security properties is challenging. Opening the system to arbitrary participation may lead to novel confidentiality and privacy attacks, and we leave this as a future research direction.

## 8 Conclusion

In this work we proposed CLOSC and CLOLC, the first two protocols attempting to tackle our newly defined problem of cross-ledger transaction consistency in the context of financial auditing. Both are built atop a two-tier blockchain architecture and CLOSC utilizes smart contracts while CLOLC linear combinations to achieve the consistency examination and verification. Moreover, we proved that both our protocols satisfy three crucial security and privacy properties. Finally, both our protocols scale well with the number of organizations, auditors, and transactions per epoch. We demonstrated this via extensive experimentation that showed both our solutions to be practical, deployable in real-world settings including hundreds of organizations and auditors, and millions of total transactions per auditing epoch.

---

### References

- 1 DEDIS Advanced Crypto Library for Go. Online; accessed 1 July 2023. URL: <https://github.com/dedis/kyber>.
- 2 Financial Reporting Council. URL: <https://www.frc.org.uk/>.
- 3 Hyperledger Fabric. URL: <https://www.hyperledger.org/use/fabric>.
- 4 Public Company Accounting Oversight Board. URL: <https://pcaobus.org/>.
- 5 What's The Difference Between 2, 3, & 4-Way Matching. Online; accessed 8 May 2024. URL: <https://www.dataserv.com/blog/difference-between-2-3-4-way/>.
- 6 The World's Biggest Accounting Fraud Scandals, 2023. Online; accessed 21 May 2024. URL: <https://www.skillcast.com/blog/accounting-fraud-scandals>.
- 7 2020. Poly Network. URL: <https://poly.network/>.
- 8 2020. Rainbow Bridge. URL: <https://near.org/nbridge/>.
- 9 2022. Axelar. URL: <https://axelar.network/>.
- 10 Sarah Allen, Srdjan Čapkun, Ittay Eyal, Giulia Fanti, Bryan A Ford, James Grimmelmann, Ari Juels, Kari Kostianen, Sarah Meiklejohn, Andrew Miller, et al. Design choices for central bank digital currency: Policy and technical considerations. Technical report, National Bureau of Economic Research, 2020.
- 11 Gilbert K Amoako, Jonas Bawuah, Emmanuel Asafo-Adjei, and Catherine Ayimbire. Internal audit functions and sustainability audits: Insights from manufacturing firms. *Cogent Business & Management*, 10(1):2192313, 2023.
- 12 Deniz Appelbaum and R Nehmer. Designing and auditing accounting systems based on blockchain and distributed ledger principles. *Feliciano School of Business*, pages 1–19, 2017.
- 13 Salman Arif, John Kepler, Joseph Schroeder, and Daniel Taylor. Audit process, private information, and insider trading. *SSRN Electronic Journal*. doi:10, 2018.
- 14 Carsten Baum, James Hsin-yu Chiang, Bernardo David, and Tore Kasper Frederiksen. Sok: Privacy-enhancing technologies in finance. In Joseph Bonneau and S. Matthew Weinberg, editors, *5th Conference on Advances in Financial Technologies, AFT 2023, October 23-25, 2023, Princeton, NJ, USA*, volume 282 of *LIPICs*, pages 12:1–12:30. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICs.AFT.2023.12.
- 15 Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector OLE. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 896–912. ACM, 2018. doi:10.1145/3243734.3243868.
- 16 Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 337–367. Springer, 2015. doi:10.1007/978-3-662-46803-6\_12.



- 17 Nathalie Brender, Marion Gauthier, Jean-Henry Morin, and Arbër Salih. The potential impact of blockchain technology on audit practice. *Journal of Strategic Innovation and Sustainability*, 14(2), 2019.
- 18 Chainlink. What Is a Cross-Chain Bridge? URL: <https://chain.link/education-hub/cross-chain-bridge>.
- 19 Panagiotis Chatzigiannis and Foteini Baldimtsi. Miniledger: Compact-sized anonymous and auditable distributed payments. In Elisa Bertino, Haya Schulmann, and Michael Waidner, editors, *Computer Security - ESORICS 2021 - 26th European Symposium on Research in Computer Security, Darmstadt, Germany, October 4-8, 2021, Proceedings, Part I*, volume 12972 of *Lecture Notes in Computer Science*, pages 407–429. Springer, 2021. doi:10.1007/978-3-030-88418-5\_20.
- 20 Corporate Finance Institute. Top Accounting Scandals: A recap of the top scandals in the past. Online; accessed 11 January 2021.
- 21 Flavio Corradini, Alessandro Marcelletti, Andrea Morichetta, Andrea Polini, Barbara Re, and Francesco Tiezzi. Engineering trustable and auditable choreography-based systems using blockchain. *ACM Trans. Manag. Inf. Syst.*, 13(3):31:1–31:53, 2022. doi:10.1145/3505225.
- 22 Jun Dai and Miklos A. Vasarhelyi. Toward blockchain-based accounting and assurance. *J. Inf. Syst.*, 31(3):5–21, 2017. doi:10.2308/ISYS-51804.
- 23 Ivan Bjerre Damgård. A design principle for hash functions. In *Conference on the Theory and Application of Cryptology*, pages 416–427. Springer, 1989.
- 24 Sebahattin Demirkan, Irem Demirkan, and Andrew McKee. Blockchain technology in the future of business cyber security and accounting. *Journal of Management Analytics*, 7(2):189–208, 2020.
- 25 Christine E Earley. Data analytics in auditing: Opportunities and challenges. *Business horizons*, 58(5):493–500, 2015.
- 26 Karl Hackenbrack and Mark W Nelson. Auditors’ incentives and their application of financial accounting standards. *Accounting Review*, pages 43–59, 1996.
- 27 Hongdan Han, Radha K. Shiwakoti, Robin Jarvis, Chima Mordi, and David Botchie. Accounting and auditing with blockchain technology and artificial intelligence: A literature review. *Int. J. Account. Inf. Syst.*, 48:100598, 2023. doi:10.1016/J.ACCINF.2022.100598.
- 28 Holly Doyle, Simon Passfield, Guildhall Chambers. Shams: When is a transaction not a transaction?, 2023. Online; accessed 27 April 2023.
- 29 Yuncong Hu, Kian Hooshmand, Harika Kalidhindi, Seung Jin Yang, and Raluca Ada Popa. Merkle<sup>2</sup>: A low-latency transparency log system. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*, pages 285–303. IEEE, 2021. doi:10.1109/SP40001.2021.00088.
- 30 Yan Ji and Konstantinos Chalkias. Generalized proof of liabilities. In Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi, editors, *CCS ’21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, pages 3465–3486. ACM, 2021. doi:10.1145/3460120.3484802.
- 31 Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177):203–209, 1987.
- 32 Vlas Koutsos, Sankarshan Damle, Dimitrios Papadopoulos, Dimitris Chatzopoulos, and Sujit Gujar. Avecq: Anonymous verifiable crowdsourcing with worker qualities. *IEEE Transactions on Dependable and Secure Computing*, pages 1–18, 2024.
- 33 Vlas Koutsos and Dimitrios Papadopoulos. Publicly auditable functional encryption. In Mehdi Tibouchi and Xiaofeng Wang, editors, *Applied Cryptography and Network Security - 21st International Conference, ACNS 2023, Kyoto, Japan, June 19-22, 2023, Proceedings, Part II*, volume 13906 of *Lecture Notes in Computer Science*, pages 396–425. Springer, 2023. doi:10.1007/978-3-031-33491-7\_15.

- 34 Vlasios Koutsos, Xiangnan Tian, Dimitrios Papadopoulos, and Dimitris Chatzopoulos. Cross ledger transaction consistency for financial auditing. Cryptology ePrint Archive, Paper 2024/1155, 2024. URL: <https://eprint.iacr.org/2024/1155>.
- 35 Stephen Kozlowski. An audit ecosystem to support blockchain-based accounting and assurance. In *Continuous Auditing: Theory and Application*, pages 299–313. Emerald Publishing Limited, 2018.
- 36 Legal Information Institute, Cornell. 31 U.S. Code § 9105 - Audits. <https://www.law.cornell.edu/uscode/text/31/9105>.
- 37 Chao Lin, Xinyi Huang, Jianting Ning, and Debiao He. ACA: anonymous, confidential and auditable transaction systems for blockchain. *IEEE Trans. Dependable Secur. Comput.*, 20(6):4536–4550, 2023. doi:10.1109/TDSC.2022.3228236.
- 38 Marcela S. Melara, Aaron Blankstein, Joseph Bonneau, Edward W. Felten, and Michael J. Freedman. CONIKS: bringing key transparency to end users. In Jaeyeon Jung and Thorsten Holz, editors, *24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015*, pages 383–398. USENIX Association, 2015. URL: <https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/melara>.
- 39 Ralph C. Merkle. A digital signature based on a conventional encryption function. In Carl Pomerance, editor, *Advances in Cryptology - CRYPTO '87, A Conference on the Theory and Applications of Cryptographic Techniques, Santa Barbara, California, USA, August 16-20, 1987, Proceedings*, volume 293 of *Lecture Notes in Computer Science*, pages 369–378. Springer, 1987. doi:10.1007/3-540-48184-2\_32.
- 40 Mohammed Ahmad Naheem. Internal audit function and aml compliance: the globalisation of the internal audit function. *Journal of Money Laundering Control*, 19(4):459–469, 2016.
- 41 Mohammed Ahmad Naheem. Money laundering: A primer for banking staff. *International Journal of Disclosure and Governance*, 13(2):135–156, 2016.
- 42 Krishnasuri Narayanam, Venkatraman Ramakrishna, Dhinakaran Vinayagamurthy, and Sandeep Nishad. Atomic cross-chain exchanges of shared assets. In Maurice Herlihy and Neha Narula, editors, *Proceedings of the 4th ACM Conference on Advances in Financial Technologies, AFT 2022, Cambridge, MA, USA, September 19-21, 2022*, pages 148–160. ACM, 2022. doi:10.1145/3558535.3559786.
- 43 Neha Narula, Willy Vasquez, and Madars Virza. zkledger: Privacy-preserving auditing for distributed ledgers. In Sujata Banerjee and Srinivasan Seshan, editors, *15th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2018, Renton, WA, USA, April 9-11, 2018*, pages 65–80. USENIX Association, 2018. URL: <https://www.usenix.org/conference/nsdi18/presentation/narula>.
- 44 P7. Auditing disclosures in financial statements, 2016. Online; accessed 21 May 2024. URL: <https://www.accaglobal.com/gb/en/student/exam-support-resources/professional-exams-study-resources/p7/technical-articles/auditing-disclosures.html>.
- 45 Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer, 1991. doi:10.1007/3-540-46766-1\_9.
- 46 Daniël Reijsbergen, Aung Maw, Zheng Yang, Tien Tuan Anh Dinh, and Jianying Zhou. TAP: transparent and privacy-preserving data services. In Joseph A. Calandrino and Carmela Troncoso, editors, *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*, pages 6489–6506. USENIX Association, 2023. URL: <https://www.usenix.org/conference/usenixsecurity23/presentation/reijsbergen>.
- 47 Andrea M Rozario and Miklos A Vasarhelyi. Auditing with smart contracts. *International Journal of Digital Accounting Research*, 18, 2018.

- 48 Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151:1–32, 2014.
- 49 Tiancheng Xie, Jiaheng Zhang, Zerui Cheng, Fan Zhang, Yupeng Zhang, Yongzheng Jia, Dan Boneh, and Dawn Song. zkbridge: Trustless cross-chain bridges made practical. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *CCS 2022*, pages 3003–3017. ACM, 2022. doi:10.1145/3548606.3560652.



# Proof of Diligence: Cryptoeconomic Security for Rollups

Peiyao Sheng  


University of Illinois Urbana-Champaign, IL, USA  
Witness Chain, NJ, USA

Ranvir Rana  

Witness Chain, NJ, USA

Senthil Bala 

Witness Chain, Bengaluru, India

Himanshu Tyagi 

Witness Chain, Bengaluru, India

Pramod Viswanath  

Princeton University, NJ, USA  
Witness Chain, NJ, USA

---

## Abstract

---

Layer 1 (L1) blockchains such as Ethereum are secured under an “honest supermajority of stake” assumption for a large pool of validators who verify each and every transaction on it. This high security comes at a scalability cost which not only effects the throughput of the blockchain but also results in high gas fees for executing transactions on chain. The most successful solution for this problem is provided by optimistic rollups, Layer 2 (L2) blockchains that execute transactions outside L1 but post the transaction data on L1.

The security for such L2 chains is argued, informally, under the assumption that a set of nodes will check the transaction data posted on L1 and raise an alarm (a fraud proof) if faulty transactions are detected. However, all current deployments lack a proper incentive mechanism for ensuring that these nodes will do their job “diligently”, and simply rely on a cursory incentive alignment argument for security.

We solve this problem by introducing an incentivized watchtower network designed to serve as the first line of defense for rollups. Our main contribution is a “Proof of Diligence” protocol that requires watchtowers to continuously provide a proof that they have verified L2 assertions and get rewarded for the same. Proof of Diligence protocol includes a carefully-designed incentive mechanism that is provably secure when watchtowers are rational actors, under a mild rational independence assumption.

Our proposed system is now live on Ethereum testnet. We deployed a watchtower network and implemented Proof of Diligence for multiple optimistic rollups. We extract execution as well as inclusion proofs for transactions as a part of the bounty. Each watchtower has minimal additional computational overhead beyond access to standard L1 and L2 RPC nodes. Our watchtower network comprises of 10 different (rationally independent) EigenLayer operators, secured using restaked Ethereum and spread across three different continents, watching two different optimistic rollups for Ethereum, providing them a decentralized and trustfree first line of defense. The watchtower network can be configured to watch the batches committed by sequencer on L1, providing an approximately 3 minute (cryptoeconomically secure) finality since the additional overhead for watching is very low. This is much lower than the finality delay in the current setup where it takes about 45 minutes for state assertions on L1, and hence will not delay the finality process on L1.

**2012 ACM Subject Classification** Security and privacy → Distributed systems security

**Keywords and phrases** blockchain, rollup, game theory, security

**Digital Object Identifier** 10.4230/LIPIcs.AFT.2024.5



© Peiyao Sheng, Ranvir Rana, Senthil Bala, Himanshu Tyagi, and Pramod Viswanath;  
licensed under Creative Commons License CC-BY 4.0  
6th Conference on Advances in Financial Technologies (AFT 2024).

Editors: Rainer Böhme and Lucianna Kiffer; Article No. 5; pp. 5:1–5:24

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Related Version *Full Version*: <https://arxiv.org/abs/2402.07241>

**Acknowledgements** This work was conducted when all authors were working at Witness Chain.

## **1 Introduction**

### **L1 Scalability**

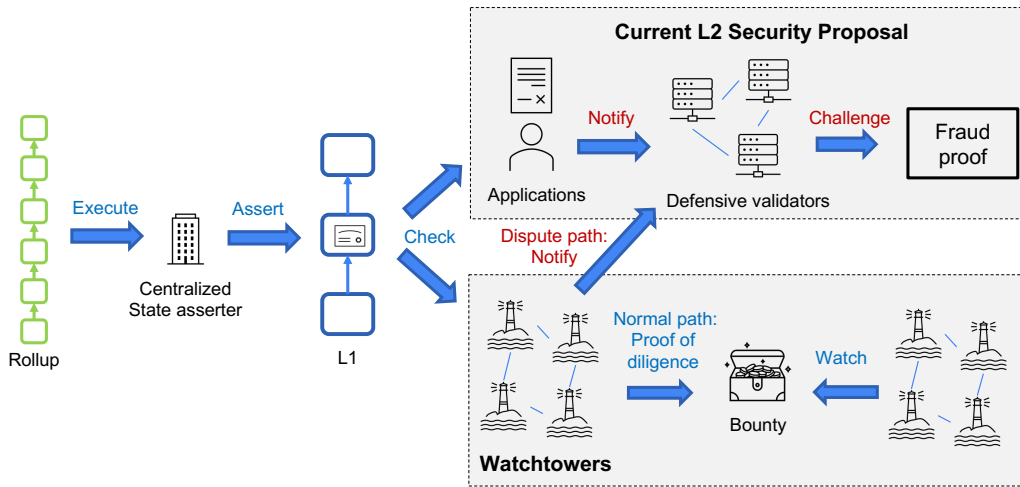
Blockchain scalability, via improving throughput, latency and transaction fees, is a crucial component of blockchain adoption [18, 30]. Improving the core L1 consensus protocols is a key direction to scalability [8, 17, 48, 10]; as an example, Ethereum upgraded consensus from the longest chain protocol in the proof-of-work (PoW) setting to the GHOST protocol [38] with Casper finality gadget [7] in the proof-of-stake (PoS) setting recently, improving its throughput potentially by three orders of magnitude and reducing 99.95% energy demand [1, 29]. Changes to L1 via upgrading the consensus protocol is a wholesale change (hard fork), requiring significant social consensus around the process and is time-consuming (e.g., Ethereum’s upgrade to PoS from PoW took seven years [9]). However the scalability challenges have persisted with increased computational demands and data storage outstripping the upgraded capabilities, with the result that resulting gas fees have not seen any significant reduction. More scaling techniques that maintain equivalent decentralization, known as horizontal scaling, are necessary.

### **L2 Economy**

L2 solutions have emerged as a significant breakthrough, with potential to increase transaction processing speed, cut down costs, and enhance the overall capacity of blockchains. Rollups work by processing transactions outside the L1 chain and then posting the transaction data and state commitments back to it. Established rollup platforms such as Optimism [46] and Arbitrum [23] accommodate a vast ecosystem (e.g., Arbitrum L2 supported more transactions than Ethereum L1 itself in February 2023 [52]). At the same time, new entities [40, 41, 45, 44] are entering the scene, introducing new infrastructure and optimizing their functionalities for specific applications. These new schemes differ from traditional rollups in their targeted functionalities and optimization techniques. Moreover, driven by the demand for customizable and accessible layer 2 solutions, the concept of “Rollups-as-a-Service” is gaining traction [22, 42, 43, 39], allowing a broader range of participants to create and utilize their own rollup strategies.

### **Rollup security**

With the rapid adoption and reliance on rollups, there is an increasing need to address the critical security concern. Current rollup strategies secure the L2 states by requiring asserters to execute and commit L2 block data to the L1 blockchain (Figure 1). These asserters, often staked and centralized, can be objectively slashed when their asserted states are proven to be incorrect. As depicted in the upper path of Figure 1, the system allows applications to independently verify the states committed on L1 and initiate disputes, serving as the secondary line of defense. These disputes are typically addressed and resolved via fraud proofs. However, a basic vulnerability remains despite the security layer provided by staked asserters and fraud proofs: the lack of incentives for actively watching the rollup. In the normal path, there is no guarantee that these applications monitor the asserted states consistently and effectively: how to ensure vigilance during normal path when attention



■ **Figure 1** Watchtowers are added to the current L2 security workflow to guard normal path security.

might diminish due to the absence of apparent threats? In other words: *who is watching the watchers?* This problem was identified sharply by the inventors of Aribtrum [16], however a systematic solution has remained open since then.

**Watchtowers: the first line of defense for rollups**

In this paper we propose a “rational watchtower pool”, a group of workers *incentivized* to constantly watch the transactions *in the normal path*. The lower part of Figure 1 illustrates how watchtowers operate independently, interacting with the existing rollup system only when an incorrect state is identified. At that point, they sound an alarm, much like what applications typically do. However, watchtowers are incentivized to stay vigilant at all times. Their role, serving as the first line of defense for rollups, is crucial for identifying potential faults that might otherwise go unnoticed. To ensure that the watchtower fulfill their “watching responsibilities” *diligently*, they must provide what we refer to as “proof of diligence”, a prerequisite for earning incentives.

**Proof of diligence**

Specifically, the duties of watchtowers entail that for new L2 state assertions to be integrated into the L2 ledger, watchtowers must execute the transactions and validate these new assertions. For a watchtower to demonstrate their diligence, their evidence must meet two criteria: (1) those who don’t process the transactions should only be able to generate the proof with a negligible likelihood; (2) a proof produced by one watchtower shouldn’t be valid for another. These standards ensure only the diligent watchtowers can generate valid proofs, and prevent multiple watchtowers from presenting identical proofs, thus promoting individual effort. To satisfy these conditions, each watchtower computes a verifiable random function (VRF) using the commitment of transaction execution trace. As every watchtower possesses a unique VRF key pair, they generate proofs independently and submit proofs on-chain, allowing public verification. As illustrated in Figure 2, other watchtowers can recompute this proof, ensuring watchtowers presenting false proofs are identified and penalized during disputes.

**Incentive framework for watchtowers**

For watchtowers to operate effectively, a carefully designed incentive mechanism is vital. This mechanism should consider two parts: positive incentives for continuous monitoring and negative incentives to punish undesirable actions. Positive rewards are allocated using a “bounty mining” scheme. Watchtowers calculate their proof of diligence to determine their eligibility for these rewards. To optimize efficiency, the criteria to obtain the bounty is set by a specific threshold, narrowing down the list of potential recipients. Only the selected winners need to submit their VRF outcomes as their proof of diligence. On the other hand, the negative incentives are implemented through staking. To engage in the protocol, watchtowers must deposit stakes, which are slashable upon detection of misconduct. Monitoring watchtower behavior introduces the problem of “watching the watchtowers”. Hence, the incentive structure offers additional rewards for the verification of the proof of diligence. Watchtowers who spot inconsistent proofs can challenge them, earning a reward if their challenge is successful. We rigorously analyze the protocol as a non-cooperative game [26], examining the potential actions of watchtowers: being diligent or lazy. By appropriately configuring these incentives, our findings show that a strategy where all watchtowers diligently monitor rollups is the unique Nash equilibrium, leading to an effective frontline defense for rollups.

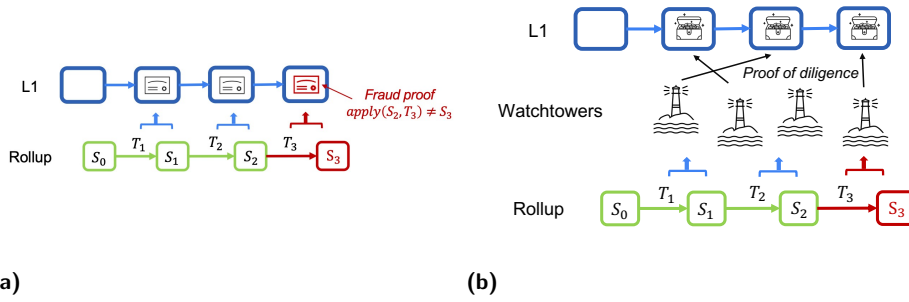
**Extended mechanism design**

Moreover, we extend the action space for watchtowers to encompass potential collusion strategies. Here, several watchtowers might form a collusion to exchange execution results or decide on a mutual random outcome driven by the benefits of saving computational costs. Our findings indicate that when factoring in such cooperative actions, the equilibrium tends to favor collusion, nullifying the role of watchtowers. To counteract this, we introduce design enhancements to disrupt collusive behaviors. We design a whistleblower scheme allowing any colluder to secretly expose collusion in exchange for compensation. Our game-theoretic analysis demonstrates that the whistleblower system encourages betrayals against the collusion. Since this reporting remains confidential, while the act of betrayal is noticeable, the whistleblower can not be individually attributed. Consequently, the collusion will not be initiated at the first place.

**System design**

By integrating the proof of diligence and incentive mechanism, the design enforces watchtowers to maintain their essential roles in guarding the security of rollups in normal path. The network consists of a pool of watchtowers that are allocated to one of the participating rollups randomly, with the allocation changing over time. The system utilizes a staking mechanism to ensure sybil resistance and fair allocation of positive incentives; the stake is also used as a bond that can be utilized for negative incentives. On-chain contracts perform the broad activities of watchtower registration, rollup status monitoring, and disbursement of incentives. The off-chain client comprises a wrapper to the L2 full node that fetches intermediate information to mine the bounty. The system is implemented on the Optimism Bedrock stack [47] that watches Optimism and Base rollups on the Goerli testnet. Note that since watchtowers operate independently of the optimistic rollup framework, the system is capable of monitoring multiple optimistic rollup chains simultaneously. We use Eigenlayer (EL) [14] as the staking mechanism and build our registration functionality associated with it. Our implementation adds a very low compute cost to an L2 full node with a low transaction fee that can be adjusted on a sliding scale.





■ **Figure 2** (a) Optimistic rollup model and (b) Watchtower model.

The subsequent sections provide a detailed breakdown of our investigative approach and findings. In Section 2, we outline the system’s model and explore various threat models. In Section 3, we compare prior work on related topics with our solution. Section 4 presents an in-depth view of our developed watchtower protocol. Further, in Section 5, we extend the model to allow collusions and conduct a thorough cooperative game theoretic analysis of the security aspects and the incentive compatibility associated with the enhanced protocol. Section 6.1 demonstrates our system design and implementation details, the evaluations results from the live system are presented in Section 6. Concluding the paper, Section 7 engages in a comprehensive discussion, bringing to the fore the pivotal learning from our research and suggesting future directions.

## 2 Security Models and Definitions

### 2.1 Rollup Model

Rollups enhance the blockchain’s efficiency by handling transaction execution off the main chain (L1). Present rollup techniques rely on either the validity proofs or fraud proofs to ensure security. Validity proofs, used in schemes known as zk-rollups, apply sophisticated cryptographic techniques to validate every batch of L2 transactions. On the other hand, fraud proofs, employed by optimistic rollups, come into play only when a fault is spotted in the execution process. Our focus is on enhancing the security of optimistic rollups (Figure 2a). In this context, the watchtower pool has been introduced to monitor the normal-path operations – that is, the process when transactions are presumed to be valid unless challenged.

In our system, there exists an L2 chain employing optimistic rollup scheme. The scheme involves an untrusted *asserter*, responsible for processing the L2 blocks and submitting the updated states assertion on L1 chain. L1 chain is guaranteed to be secure and live. We assume the data of L2 blocks are stored on L1 directly or through a trusted data availability service, hence the data is ensured to be retrievable. Furthermore, the system consists of a group of *defensive validators*, whose role is activated when there is a need to produce a fraud proof, which is verifiable on chain. We consider a *live* rollup system operator, who coordinates the issuance of L2 blocks (typically through a role called sequencer) and provides rewards for assserter, validators, and watchtowers. The goal of the operator is to ensure the security of rollup states with minimal cost.

## 2.2 Watchtower Model

Independent of the rollup infrastructure, our model incorporates a pool of  $n$  registered entities known as *watchtowers* (Figure 2b), denoted as  $\{W_1, W_2, \dots, W_n\}$ . Each watchtower  $W_i$  has deposited relative stake  $\alpha_i$  into the system, where  $\sum_i^n \alpha_i = 1$  and the total amount of stake is  $\mathcal{S}$ . These watchtowers are modeled as individual *rational adversary*, where the term “rational” implies that they operate with the purpose of optimizing a known payoff function. And they have the ability to methodically process all potential scenarios and select strategies that provides the best outcomes.

Even though rational participants will not deviate from the protocols without cause, they might engage in attacks – either intentionally or inadvertently – if the expected rewards justify such actions. We identify several potential attacks in this framework, emphasizing the nuanced strategies rational actors might employ.

### Lazy watchtower

Since watchtowers are also required to execute the entire batch of L2 transactions, they can earn rewards by performing their duties diligently. This role closely resembles that of the asserter in the original optimistic rollup system, who posts computation results in exchange for rewards. As a result, one prominent challenge the watchtower design must confront is the “lazy watchtower” problem. This issue arises because of two main reasons: (1) rational watchtowers may submit arbitrary responses if the results lack verification process, and (2) they might opt out of protocol participation if the associated costs outweigh potential rewards. In essence, the watchtowers must provide a form of evidence for their work and the protocol must offer sufficient incentives to encourage participants to actively and consistently perform tasks.

### Collusion attack

Rational entities might form collusion where several parties conduct coordinated actions based on a common agreement. However, it’s essential to note that, despite the existence of any agreement, colluding parties maintain individual autonomy and continue to prioritize their self-interest. Within the scope of collusion attacks, we consider adaptive adversaries who can adjust their collusion strategies in response to protocol developments.

In our system, we assume that adversaries are limited by computational constraints, so that they are not able to break the security of necessary cryptographic primitives. It’s important to highlight that rational adversaries, guided by their payoff functions, are considered weaker threats compared to Byzantine adversaries. Since their profit-driven actions exclude certain strategies. In the discussion section (Section 7) we explore the trade-off between different levels of security and associated costs.

## 2.3 Preliminaries

We introduce fundamental primitives utilized throughout the paper. Other notations are summarized in Table 1.

### Verifiable random functions

Our protocol use verifiable random functions (VRFs), providing two functions to generate and verify proofs.  $\text{VRF}_{sk}(x)$  processes an input  $x$  and returns two values  $(d, \pi_d)$ : a normalized hash digest  $d$  and a proof  $\pi_d$ . The value  $d \in [0, 1)$  is uniquely determined by the input  $x$

■ **Table 1** Summary of notations.

Term	Definition	Term	Definition
$n$	Number of registered watchtowers	$c_T$	Cost per transaction batch
$W_i$	Watchtower $i$	$c_V$	Cost of resolving disputes
$\alpha_i$	Relative stake of watchtower $W_i$	$R_B$	Normal path reward for watchtowers
$S$	Total amount of stake in the system	$R_C$	Dispute path reward for watchtowers
$S$	Blockchain state	$R_w$	Whistleblower protocol reward
$T$	A batch of L2 transactions	$\phi(\alpha_i)$	Bounty mining threshold
$E$	Execution trace	$t$	Deposit for participating in collusion
$r_S$	State assertion	$h$	Rent in diligent collusion
$r_E$	Merkle root of execution trace	$n_c$	The size of the colluding group

and a secret key  $sk$ , and is indistinguishable from a random value to anyone that does not know  $sk$ .  $\text{verifyVRF}_{pk}(\pi_d, d, x) \in \{\text{true}, \text{false}\}$  takes the proof  $\pi_d$  input and allows anyone who knows the public key  $pk$  to verify whether  $d$  is the correct value computed from  $x$  and  $sk$ .

### Merkle trees

Merkle trees are a fundamental data structure in cryptography, summarizing lists of items, such as transactions or states, by concatenating their cryptographic hashes at various levels of the tree. We provide the function  $\text{Merkelize}(L) \rightarrow r$  that generates a Merkle root  $r$  from a list of items  $L$ . It involves the construction of a Merkle tree or Patricia trees [28, 51] for  $L$  and then produces the root hash that represents the entire list of items.  $\text{MerkleProof}(r, l, L) \rightarrow p$  is used to generate a Merkle proof  $p$  associated with a leaf node  $l$  in a tree rooted at  $r$ . This proof consists of the minimal amount of information needed to confirm the presence of the specific leaf node  $l$  within the tree constructed from  $L$ . Furthermore, there exists a validation function  $\text{verifyMerkleProof}(r, l, p) \in \{\text{true}, \text{false}\}$  that takes in a root  $r$ , a leaf  $l$ , and a Merkle proof  $p$ , then returns a boolean value indicating whether  $p$  demonstrates that  $l$  is indeed part of the Merkle tree rooted at  $r$ .

### State transitions

We consider a general model for L1 blockchain, which keeps the latest states set  $S$  and transactions organized as blocks within hash-based chains. The function  $\text{apply}(S', T) \rightarrow (S, E)$  represents the process of applying a list of transactions  $T$  to a prior state  $S'$  to yield a new state  $S$  and an execution trace  $E$ . The execution trace is a detailed record of all intermediate states during the transition from  $S'$  to  $S$ , serving as a reference for verification. Another function  $\text{validate}(r, r', L) \in \{\text{false}(r), \text{false}(r'), \text{false}(r, r')\}$  resolves a conflict between two Merkle roots  $r$  and  $r'$  calculated from the same list of items  $L$ . The output represents the subset of roots that are proven to be invalid. There are different ways to implement this function, such as using L1 as a trusted third party to provide ground truth or executing an interactive verification game (IVG) to identify the incorrect root.

### Game theory

The concepts about game theory utilized in our analysis are all defined in the book [26]. We first analyze the rollup security with watchtowers as a non-cooperative game in strategic form (Def. 4.2, [26]), and examine the dominance of strategies (Def. 3.10 and 4.6, [26]) to

find the pure strategy Nash equilibrium (Def. 4.17 and Def. 5.3, [26]) for watchtowers. We also discuss an enhanced protocol considering cooperative game (Chapter 15, [26]), where more than one Nash equilibria exist and Pareto efficiency (Def. 15.7, [26]) is considered.

### **3** Background and Prior Work

The dialogue surrounding optimistic rollups originated in community discussions on the Ethereum research forum [2], where developers and researchers shared insights about potential scalability solutions. One of the earliest detailed presentations came from the Optimism team, who outlined the foundational framework of optimistic rollups and their theoretical implications [21]. The theoretical cornerstone for optimistic rollups was further strengthened through scholarly research. Pioneering works such as Arbitrum [19] and TrueBit [49] introduced essential concepts related to off-chain computation and dispute resolution.

Subsequent discussions began to address broader implications in contexts such as data availability [3, 53, 34], sequencer risks [27, 32], and validator behavior [20, 16, 33, 15, 13]. Specifically, among these topics, validator behavior is most pertinent to this paper. While the basic rollup design discusses an elementary incentive structure to motivate asserters to post correct states, verifiers may lack incentives to monitor the system's states diligently, such a situation is known as the verifier's dilemma problem [24]. To address this issue and enhance system security, the authors of Arbitrum [16] proposed an attention game in which verifiers who fail to participate may be punished. TrueBit [20] also suggested an enhanced mechanism to select a pool of validators, requiring them to submit a proof of independent work [15] to enforce verification. Additionally, some research [33] provides game-theoretic analyses of collusion risks in incentivized computation outsourcing, proposing mitigation strategies, though not completely resolving the issue. In the field of verifiable outsourced cloud computing, [13] investigates the dynamics between clients and workers within smart contract frameworks, demonstrating a preference for honest behavior under certain conditions. Although they propose a "traitor contract" to eliminate collusion, this is limited to a two-party context. Furthermore, we contend that this proposed solution might not be effective, as the contract's output could compromise the traitor's secrecy. Another study [35] examines cooperation in  $N$ -person prisoner's dilemma scenarios, with institutional arrangements akin to smart contracts. Differing from our focus, this work considers collusion through bargaining rather than a leader-based approach. A new design of the attention game was proposed in a recent paper [25] to find the optimal number of validators that minimizes failure probability. The design only provides probabilistic security and requires modification in the underlying rollup protocol to ensure deterministic security. As a comparison, we provide a plug-and-play solution that can be used for any off-chain compute resource with minor modifications.

## **4** The Watchtower Protocol

### **4.1** Proof of Diligence

In our protocol, we focus on a single task that watchtowers undertake: verifying the updated state assertion  $r_S$  of the L2 blockchain, a state assertion is the Merkle root of the state tree, calculated by  $r_S = \text{Merkelize}(S)$ . The states  $S$  on a blockchain consist of a list of key-value pairs, such as the account address and the account balance. Formally, given the latest validated state  $S'$  and a sequence of transactions  $T$ , the responsibility of the watchtowers is to verify  $r_S$  by executing the computations specified in Algorithm 1.

---

**Algorithm 1** Function for Watchtowers.
 

---

```

1: function CHECKSTATE( $S', T, r_S, \alpha_i$ )
2:    $(S, E) \leftarrow \text{apply}(S', T)$ 
3:    $r'_S = \text{Merkelize}(S)$ 
4:    $r_E = \text{Merkelize}(E)$ 
5:    $(d, \pi) = \text{VRF}_{sk}(r'_S | r_E)$ 
6:   if  $d < \phi(\alpha_i)$  then
7:     submit proof of diligence  $(d, \pi)$ 
8:   if  $r_S = r'_S$  then
9:     Return true
10:  else
11:    Return false

```

---

Algorithm 1 represents the process of assessing whether the proposed state assertion  $r_S$  is indeed consistent with the ledger history. The watchtower calling the `checkState` function first applies the transactions  $T$  to the initial state  $S'$ , which returns the new state  $S$  and an execution trace  $E$ . The watchtower then calculates the Merkle root of the legitimate state  $S$  and compares the result  $r'_S$  with the posted  $r_S$ . If the verification process succeeds and no faults are found, the watchtower considers the new state  $S$  as validated. Otherwise, the watchtower is expected to raise an alarm to the rollup scheme, activating defensive validators for dispute resolution and fraud proof creation.

Besides, the watchtowers utilize the execution trace  $E$  to construct a Merkle root  $r_E$ . Since  $r_E$  is not available anywhere and can only be derived by executing the transactions, it serves as evidence of their work and diligence. They compute a VRF using  $r_E$  and  $r'_S$ , incorporating their secret keys. It is assumed that the corresponding public keys were disclosed during the registration phase. The VRF produces  $(d, \pi)$ ; the digest  $d$  is subsequently used to allocate rewards for diligent watch. Accompanied by the proof  $\pi$ , the watchtower submits this as the *proof of diligence*, which satisfies the following properties:

- **Verifiability.** Given a keypair  $(pk, sk)$ , for any input  $x$ , if  $(d, \pi) \leftarrow \text{VRF}_{sk}(x)$ , then  $\text{verifyVRF}_{pk}(\pi, d, x) = \text{true}$ .
- **Uniqueness.** Given a keypair  $(pk, sk)$ , for any input  $x$ , if  $(d, \pi) \leftarrow \text{VRF}_{sk}(x)$ , no one, including the key owner, can produce a different  $d' \neq d$  and the associated proof  $\pi'$  such that  $\text{verifyVRF}_{pk}(\pi', d', x) = \text{true}$ .
- **Pseudorandomness.** For a given input  $x$  and public key  $pk$  the output  $d$  is indistinguishable from a truly random string to anyone who does not possess the private key  $sk$ .

The verifiability ensures that once the proof is posted on the chain, it can be verified by every watchtower using  $r_S, r_E$ , and the public key  $pk$ . This process is referred to as “watching the watchtowers”. Other watchtowers will use their own  $r_S, r_E$  values to check the proof. If they detect any inconsistencies, they invoke the `validate` function to resolve conflicts on the chain. After a predefined challenge period  $t_C$ , the protocol concludes that the remaining proofs are correct. The uniqueness and pseudorandomness imply that only diligent watchtowers can generate a valid proof, and the proof generated by one watchtower cannot be used by others.

## 4.2 Incentive Design

To ensure that watchtowers execute their verification duties with diligence, it's imperative to institute a carefully designed incentive mechanism. First of all, this mechanism mandates that all enrolled watchtowers use their stakes as the deposit. This deposit acts as a form of commitment to honest service – any verified misbehavior results in the slashing of their deposit.

### Bounty mining

Understanding that watching operations incur costs, represented as  $c_T$ , in the process of executing all transactions in  $T$ , we introduce a bounty mining scheme to motivate the watchtowers to perform the verification (specified in Algorithm 1, line 4-7). The process of bounty mining can be analogous to the process of committee or leader selection in some blockchains [8, 11], where a VRF is computed to determine bounty winners. Formally, a diligent watchtower  $W_i$  with relative stake  $\alpha_i$  who performs the transaction execution can generate a proof of diligence  $(d, \pi)$  on the task. Then with a probability  $\phi(\alpha_i)$ ,  $W_i$  finds that its proof satisfies a certain condition (specified in Eq. 1), allowing them to receive a bounty by publishing the proof. A system parameter  $\theta$  is defined to control the probability that a party with all stake wins the bounty.

$$\Pr[\text{VRF}_{sk_i}(r_S|r_E).d < \phi(\alpha_i)] = \phi(\alpha_i) = 1 - (1 - \theta)^{\alpha_i} \quad (1)$$

The amount of bounty each winner who submits a valid proof  $(d, \pi)$  can collect is a constant value  $R_B$ . If any watchtower identifies an incorrect proof, the `validate` function will be invoked to resolve the dispute. In this process, both watchtowers are required to publish their  $r_S, r_E$  values. The winner of the challenge will receive a constant reward of  $R_C$ , and a compensation distributed among all winners for the cost of dispute resolution  $c_V$ . The rewards setting satisfies the following conditions:

$$R_B > \frac{c_T}{\phi(\alpha_0)}, R_C > c_T \quad (2)$$

where  $\alpha_0$  is the unit stake fraction, hence  $\alpha_0 \leq \min\{\alpha_i\}_{i \in [1, n]}$ . To ensure that the slashed deposit is enough to pay for the cost of dispute resolution and rewards, we require that

$$\alpha_0 S \geq c_V + (n - 1)R_C \quad (3)$$

In summary, the entire protocol works as follows.

1. When a new state assertion  $r_S$  is published, a bounty timer  $t_1$  starts. Each watchtower recomputes the assertion  $r'_S$  and generates the execution trace root  $r_E$  by applying transactions  $T$  to the old states  $S'$ , then computes  $(d, \pi) = \text{VRF}_{sk}(r'_S, r_E)$ .
2. If the assertion is incorrect ( $r_S \neq r'_S$ ), the watchtower notifies the defensive validators of the rollup to initiate a challenge.
3. If a watchtower wins the bounty, the watchtower submits proof of diligence  $(d, \pi)$  before  $t_1$ .
4. When a watchtower observes a proof of diligence submitted by other watchtowers, it verifies the proof using the other's public key and the execution trace root  $r_E$  calculated by itself. If the proof is incorrect, the watchtower calls `validate` interface to resolve the dispute. The cost of triggering `validate` is denoted as  $c_V$  shared among all watchtowers who call the function.
5. If no `validate` is triggered before  $t_1$  expires, the rollup operator concludes that the asserted execution trace root is correct. Validated bounty winners receive  $R_B$  as reward each.
6. If `validate` is triggered, the winning parties receive a reward  $R_C$  and a compensation for the shared cost, and the losing parties lose all the stake.

### 4.3 Incentive Analysis in Non-Cooperative Games

We first consider the proof of diligence protocol as a non-cooperative game denoted as PoD-Game, where different watchtowers optimize their individual payoff without any mutual agreement for cooperation. Watchtowers can adopt one of two possible strategies: diligent or lazy. Diligent watchtowers execute transactions honestly and report proof of diligence when the condition is met. In contrast, lazy watchtowers opt for generating a random result as the new state assertion, incurring negligible cost. Moreover, these lazy watchtowers might submit a fake proof, computed from the random root, to deceitfully claim the bounty. Notably, we consider only those lazy *and deceitful* watchtowers, as non-deceitful lazy watchtowers are indistinguishable from non-participants in impact. A default constraint of our incentive mechanism is that the payoff for diligent behavior is always positive, which dominates the non-participating strategy. Therefore, we omit this trivial case in the subsequent analysis.

Let  $u_i^a(n_d)$  be the expected payoff function of watchtower  $W_i$  given that watchtower  $W_i$  choose action  $a \in \{d, l\}$ , where  $d$  and  $l$  represent diligent and lazy strategies respectively, and there are  $n_d$  diligent watchtowers in total. According to the protocol, for all  $i \in [1, n]$ , these payoff functions are defined as follows:

$$u_i^d(n_d) = \begin{cases} \phi(\alpha_i)R_B + R_C - c_T & n_d < n \\ \phi(\alpha_i)R_B - c_T & n_d = n \end{cases} \quad (4)$$

$$u_i^l(n_d) = \begin{cases} \phi(\alpha_i)R_B & n_d = 0 \\ -\alpha_i S\phi(\alpha_i) & n_d > 0 \end{cases} \quad (5)$$

Note that a lazy watchtower would attempt to mimic genuine probability to submit proofs, otherwise the proof submission frequencies that can happen with negligible probability can be used to detect malicious behaviors. By comparing the payoff for different strategies, we observe that the diligent strategy dominates the lazy strategy for all watchtowers; formally, we have the following theorem.

► **Theorem 1.** *The diligent strategy in the PoD-Game is a dominant strategy for all watchtowers.*

**Proof.** For every watchtower  $W_i$ , given an arbitrary strategy vector  $s_{-i}$  containing all others' actions. Let  $n_d$  denote the number of watchtowers that are diligent in  $s_{-i}$ , we compare the payoff of two strategies for  $W_i$  below.

- Case 1: If  $n_d = 0$ , due to Eq.2,  $u_i^d(1) = \phi(\alpha_i)R_B + R_C - c_T > \phi(\alpha_i)R_B = u_i^l(0)$ .
- Case 2: If  $0 < n_d < n - 1$ , due to Eq.2,  $u_i^d(n_d + 1) = \phi(\alpha_i)R_B + R_C - c_T > -\alpha_i S\phi(\alpha_i) = u_i^l(n_d)$ .
- Case 3: If  $n_d = n - 1$ , due to Eq.2,  $u_i^d(n) = \phi(\alpha_i)R_B > -\alpha_i S\phi(\alpha_i) = u_i^l(n - 1)$ . ◀

Theorem 1 implies that the game has a unique Nash equilibrium since we can eliminate the strictly dominated lazy strategy and get a unique strategy vector of all diligence; this follows directly from Cor.4.37 [26].

► **Corollary 2.** *The unique Nash equilibrium in PoD-Game occurs when every watchtower is diligent.*

In conclusion, in a setting without cooperation, our proof of diligence protocol ensures that rational watchtowers will always work diligently, providing the first line of defense for the rollup system. In practice, this setting can model many situations, such as when watchtowers cannot communicate with each other, or when there is a public reputation system where participating in any cooperation would be detected and deteriorate reputations.

## 5 Cooperative Games and The Enhanced Protocol

In the PoD-Game, we observe that if all watchtowers choose to be lazy and agree to use a common random  $r_E$  and  $r_S$  to compute their proofs, they will receive a higher payoff than in the all-diligent equilibrium. However, this strategy did not get chosen since any party can deviate from this “unreliable collusion” to achieve higher utility, while lazy parties end up losing all their stakes.

Collusion can be reinforced by adding specific punishment mechanisms. In the context of rollups, smart contracts are the most viable tools for enforcing agreements or promises of such cooperation. Beyond the previously mentioned lazy collusion strategy, other strategies may improve payoff through cooperation, like sharing the costs of diligence. Additionally, the process of forming a colluding group can vary. For instance, in a leader-based method, a watchtower might take the initiative to create a colluder contract [13], setting conditions for joining and outlining actions to be taken, thereby allowing others to join. Conversely, in a leader-less method, watchtowers can choose to join a collusion group and negotiate group strategy collectively [35].

In this section, we explore the space of collusion, concentrating on two main leader-based strategies applicable to most relevant settings. Our primary findings reveal that establishing mutual agreements enforced by smart contracts makes lazy actions more beneficial. To counteract this, we propose setting up Whistleblower contracts, which encourage colluders to betray their collusion, thereby eliminating the lazy equilibrium.

### 5.1 Lazy Collusion

One possible strategy that watchtowers might employ to solidify the collusion is to require all colluders to deposit a certain amount of stakes into the collusion. If a colluder posts a proof computed from different execution roots, they will lose the collusion deposit.

We assume any party is capable of initiating such collusion, and we refer to that party as the leader. A leader will specify the amount of stake  $t$  that each newly joined colluder needs to contribute. Since the collusion is motivated by the benefit of being lazy, we term this strategy “lazy collusion.” We observe that watchtowers choosing not to join the collusion perceive the same game as PoD-Game, and therefore, they are likely to adopt a diligent strategy. If such independent watchtowers exist, lazy collusion will not gain the expected advantage by not computing the results. Consequently, collusion will only be effective when all watchtowers participate in it. Specifically, the process of forming lazy collusion unfolds as follows:

1. A watchtower initiates collusion by placing a deposit of  $t$ . The watchtower also releases a randomly chosen  $r'_E$ .
2. Other watchtowers may join the collusion by placing a deposit of  $t$ . If  $n$  watchtowers join the collusion before  $t_{lc}$ , the collusion is formed. Otherwise, watchtowers get back their deposits.
3. During the watching phase, all colluders are required to calculate the proof of diligence using  $\text{VRF}_{sk}(r_S, r'_E)$ , where  $r_S$  is the state root posted by the asserter.
4. If a colluder becomes a winner, the collusion protocol will check whether the winner’s proof is calculated from  $r'_E$ , if not, the winner is considered a traitor and will lose  $t$ .
5. At the end of the collusion, colluders who do not betray the collusion receive  $t + n_t t / (n_c - n_t)$ , where  $n_c$  is the size of colluding group,  $n_t$  is the number of traitors. If all colluders betray, everyone gets back their deposit  $t$ .



We are considering two possible actions that all colluders (including the leader) can take, given the collusion strategy selected by the leader: obey and betray. Colluders who choose to obey the strategy follow the leader's instructions to submit a response calculated from the specified  $r_E$ , while those who choose to betray may submit something different, driven by personal interest. The game induced by lazy collusion is denoted as LC-Game. Let  $u_i^a(n_o)$  be the expected payoff function of the  $i$ -th colluder  $W_i$  ( $i \in [1, n_c]$ ).  $a \in \{o, b\}$  represents for the action chosen by  $W_i$ , with  $o$  as obey and  $b$  as betray. There are  $n_o$  colluders who choose to obey the collusion strategy. According to the protocol,  $n_c = n$ , so we let  $l_i = i$  for simplicity, and the payoff functions can be written as follows:

$$u_i^o(n_o) = \begin{cases} -\alpha_i \mathcal{S}\phi(\alpha_i) + \frac{(n-n_o)t}{n_o} & n_o < n \\ \phi(\alpha_i)R_B & n_o = n \end{cases} \quad (6)$$

$$u_i^b(n_o) = \begin{cases} \phi(\alpha_i)R_B - c_T & n_o = 0 \\ \phi(\alpha_i)R_B + R_C - c_T - t & n_o > 0 \end{cases} \quad (7)$$

Now when there exists such lazy collusion, we compare the payoff of colluders with different actions and observe that obeying the group strategy dominates the betrayal for all colluders when the deposit  $t$  is high enough, formally, we have the following theorem.

► **Theorem 3.** *The “obey” strategy in the LC-Game is a dominate strategy for colluder  $W_i$  if the following conditions hold:*

$$t > R_C - c_T \quad (8)$$

$$t > \frac{n-1}{n} (\alpha_i \mathcal{S}\phi(\alpha_i) + \phi(\alpha_i)R_B + R_C - c_T) \quad (9)$$

**Proof.** For every colluder  $W_i$ , given an arbitrary strategy vector  $s_{-i}$  containing all others' actions. Let  $n_o$  denote the number of colluders that obey the collusion in  $s_{-i}$ , we compare the payoff of two strategies for  $W_i$  below.

- Case 1: If  $n_o = n - 1$ , due to Eq. 8,  $u_i^o(n) = \phi(\alpha_i)R_B > \phi(\alpha_i)R_B + R_C - c_T - t = u_i^b(1)$ .
- Case 2: If  $0 < n_o < n - 1$ , due to Eq. 9,  $u_i^o(n_o + 1) = -\alpha_i \mathcal{S}\phi(\alpha_i) + \frac{(n-n_o-1)t}{n_o+1} > \phi(\alpha_i)R_B + R_C - c_T - t = u_i^b(n_o)$ .
- Case 3: If  $n_o = 0$ , due to Eq. 9,  $u_i^o(1) = -\alpha_i \mathcal{S}\phi(\alpha_i) + (n-1)t > \phi(\alpha_i)R_B - c_T = u_i^b(n)$ . ◀

Then we consider the game combining PoD-Game and LC-Game. It starts with an initiation phase where watchtowers may choose to initiate a contract to become the collusion leader. If all watchtowers join the same collusion contract, which is the only case where collusion will be formed successfully, and the conditions specified in Eq.8 and 9 are satisfied, this sub-game LC-Game can be eliminated according to Theorem 3, with the payoff induced by the dominant strategy. Otherwise, independent watchtowers will follow the PoD-Game and iterative elimination can be applied with Theorem 1. Observing that the payoff derived from LC-Game is higher than PoD-Game, we find the game has two Nash equilibria but only the lazy collusion strategy is Pareto efficient.

► **Corollary 4.** *In PoD-Game that allows lazy collusion, there are two Nash equilibria: (1) all watchtowers are independently diligent (2) all watchtowers are collusively lazy. The second equilibrium is Pareto efficient.*

■ **Table 2** DC-Game: The game induced by diligent collusion.

Payoff $\begin{pmatrix} u_{d_1} \\ u_{d_i} \end{pmatrix}$		follower	
		All join	Not all join
leader	Obeys	$\begin{pmatrix} \phi(\alpha_{d_1})R_B - c_T + (n_C - 1)h \\ \phi(\alpha_{d_i})R_B - h \end{pmatrix}$	
	Betray	$\begin{pmatrix} \phi(\alpha_{d_1})R_B - c_T + R_C - \frac{c_V}{n - n_C + 1} - t \\ -\alpha_{d_i}\mathcal{S}\phi(\alpha_{d_i}) + \frac{t}{n_C - 1} \end{pmatrix}$	
	Cheat	$\begin{pmatrix} \phi(\alpha_{d_1})R_B + (n - 1)h \\ \phi(\alpha_{d_i})R_B - h \end{pmatrix}$	$\begin{pmatrix} \phi(\alpha_{d_1})R_B + (n - 1)h \\ \phi(\alpha_{d_i})R_B - h \end{pmatrix}$

## 5.2 Diligent Collusion

Moreover, watchtowers might choose to remain diligent while seeking to form a collusion to share the execution costs. In this scenario, the leader initiating the collusion carries out the computations and commits its solution to the group. Anyone wishing to join the collusion is required to contribute a fee of  $h < c_T$  to access the results. Subsequently, all colluders utilize the execution root, as calculated by the collusion leader, to generate proof of diligence and claim bounties. We refer to this strategy as “diligent collusion”. The process for forming such a diligent collusion unfolds as follows:

1. A watchtower initiates collusion by placing a deposit of  $t$ . The watchtower also commits a computed  $r'_E$  and specifies a rent  $h < c_T$ .
2. Other watchtowers may join the collusion by paying the rent of  $h$ . Then the committed  $r_E$  is revealed.
3. During the watching phase, all colluders are required to calculate the proof of diligence using  $\text{VRF}_{sk}(r_S, r'_E)$ .
4. If the proof a non-leader colluder submits is recognized as a faulty proof, the leader will lose  $t$ , and others will get back  $t/(n_c - 1)$ , where  $n_c$  is the size of the colluding group.
5. If the proof provided by leader gets accepted, the leader gets  $t + (n_c - 1)h$ .

In the game of diligent collusion, we observe that the leader has three possible actions: obey, betray, and cheat. “Obey” implies that the leader will diligently compute the transaction execution root and share it with others. “Betray” suggests that the leader might commit a random output to the colluding group while submitting a correct proof for its own benefit. And “cheat” represents the scenario in which the leader lazily commits and submits the same random output. The other colluders may only choose to follow what the leader commits, since they pay the rent; in other words, there is no benefit to join the collusion if they plan to choose another action. Table 2 lists the payoff functions of DC-Game, the game induced by the diligent collusion strategy.  $u_{d_1}$  and  $u_{d_i}$  ( $i \in [2, n_c]$ ) represent for the payoff function for the leader and other colluders. Note that if the leader chooses to cheat, its payoff is highly influenced by whether all watchtowers join the collusion. If there exist independent watchtowers, they must choose the diligent strategy as the PoD-Game implies, then all colluders will be punished by the proof of diligence protocol. Therefore, it is evident that the following theorem holds:

► **Theorem 5.** *DC-Game has no pure strategy Nash equilibrium when  $t > R_C - c_V/(n - 1) - h$ .*

**Proof.** We denote the strategy profile in DC-Game as  $\{a, n_c\}$ , where  $a \in \{o, b, c\}$  is the action chosen by leader, representing obey, betray and cheat,  $n_c$  is the number of other watchtowers who choose to join the collusion. Firstly, the condition  $t > R_C - c_V/(n - 1) - h$

implies that  $\forall n_c \in [2, n], u_{d_1}(\{o, n_c\}) > u_{d_1}(\{b, n_c\})$ , hence betray is strictly dominated by obey. We then observe that  $u_{d_1}(\{o, n\}) < u_{d_1}(\{c, n\})$ , indicating that  $\{o, n\}$  is not a Nash equilibrium, as in this case the leader can achieve a higher payoff by switching to cheat. Additionally, independent watchtowers receive a higher payoff if they join the collusion when the leader chooses to obey, as this spreads the execution cost across the entire colluding group. Conversely, when the leader opts to cheat, if all watchtowers join the collusion, switching to be independently diligent achieves a better payoff. However, if not all watchtowers join, obey becomes the more beneficial strategy for the leader. Consequently, there doesn't exist any pure strategy that is a Nash equilibrium. ◀

Theorem 5 indicates that even if we take DC-Game into consideration, the pure strategy Nash equilibria of the full game remain the same. Hence we have the following property.

▶ **Corollary 6.** *In PoD-Game that allows lazy and diligent collusion, there are two Nash equilibria: (1) all watchtowers are independently diligent (2) all watchtowers are collusively lazy. The second equilibrium is Pareto efficient.*

### 5.3 Enhanced Protocol with Whistleblower

The incentive for colluders to establish collusion lies in the potential to mine bounties with less effort. However, less effort always correlates with the likelihood of faulty proofs. As analyzed in Section 5.1, watchtowers are incentivized to join the lazy collusion. In lazy collusion, the benefits derived from betraying the collusion (by being diligent in computations) do not suffice to offset the loss of the collusion deposit. Thus, the principles for eliminating the collusion are to: (1) offer rewards for identifying and reporting collusion, and (2) provide compensation for the losses incurred from betraying the collusion. We refer to such colluders, who disclose information about collusion to the rollup system, as “whistleblowers”. In response, we have specifically designed the whistleblower protocol as follows:

1. The rollup operator establishes a whistleblower bounty  $R_w$  and declares that the first whistleblower will be eligible for the reward.
2. Any individual can place a deposit of  $d$  and submit the correct  $r_E$  to assume the role of a whistleblower.
3. The protocol invokes the `validate` interface to resolve the dispute. If the whistleblower succeeds, they receive  $R_w + d$  in return. Otherwise, a loss results in the forfeiture of their deposit.

To ensure the payoff of whistleblower is better in all above collusion games, we derive the following condition to determine rewards.

▶ **Lemma 7.** *With the additional action “report” that each watchtower can choose, the strategy that all watchtowers obey the lazy collusion is no longer a Nash equilibrium for watchtower  $i$  if*

$$R_w > \phi(\alpha_i)R_B + c_V + \alpha_i S\phi(\alpha_i) + c_T \quad (10)$$

**Proof.** We first consider the impact that the additional action `report` brings to LC-Game. First, the whistleblower, by adhering to the collusion agreement to submit proof, will not be subject to punishment by the slashing rule of the collusion. However, the payoffs of other colluders will be reduced due to the exposure of the collusion, hence the changes in outcomes are detectable and might be used to augment the collusion deposit. Even though, the act of reporting cannot be traced back to an individual colluder. Therefore, any colluder may switch to report to gain higher payoff from the whistleblower protocol, since

## 5:16 Proof of Diligence: Cryptoeconomic Security for Rollups

$$u_i^r(n) = -\alpha_i \mathcal{S} \phi(\alpha_i) - c_V + R_C - c_T + R_w \quad (11)$$

$$> \phi(\alpha_i) R_B + R_C > \phi(\alpha_i) R_B = u_i^o(n) \quad (12)$$

This further discourages other colluders from participating in the collusion, because the payoff  $\hat{u}_i^o(n)$  they get in the existence of whistleblower becomes

$$\hat{u}_i^o(n) = -\alpha_i \mathcal{S} \phi(\alpha_i) < \phi(\alpha_i) R_B - c_T - c_V/2 + R_C = u_i^d(n) \quad (13)$$

As a result, joining and obeying the lazy collusion is not a Nash equilibrium.

Then we discuss the impact of the whistleblower scheme and the elimination of the lazy equilibrium on DC-Game. We denote the strategy profile in DC-Game as  $\{a, n_c, w\}$ , where  $a \in \{o, b, c\}$  is the action chosen by the leader,  $n_c$  is the number of watchtowers in the collusion, and  $w \in \{\text{true}, \text{false}\}$  represents whether there exists a whistleblower. First, all strategies with  $w = \text{false}$  are not Nash equilibria by the same analysis in Theorem 5. Next, since the existence of a whistleblower will not lower the payoff of a leader who chooses to obey, betrayal is still strictly dominated by obedience. Then we consider the strategy where a whistleblower exists in the collusion group. In this case, if the leader opts to cheat, it's always better to switch to obedience. However, if the leader chooses to obey, the whistleblower is better off choosing not to report. Consequently, there doesn't exist any pure strategy in the subgame DC-Game that is a Nash equilibrium. ◀

The introduction of a whistleblower protocol changes the payoff dynamics of lazy collusion, as any colluder can expose the collusion for a higher payoff. Knowing this, the leader may not choose to initiate lazy collusion in the first place and the full game reach back the state where diligent strategy is the only Nash equilibrium.

► **Corollary 8.** *In PoD-Game that allows lazy and diligent collusion, if whistleblower contract exists, there is a unique equilibrium that all watchtowers are independently diligent.*

### Cryptoeconomic security and parameter selection

To provide a clear benchmark for evaluation and decision-making for different security needs, we discuss how to choose parameters that ensure both cryptoeconomic security and compatible incentives.

First, we normalize the execution transaction costs to  $c_T = 1$ . As an example, assume there are  $n = 10$  watchtowers with equal stakes; then  $\phi(\alpha_i) = \phi(\alpha_0) \simeq 0.2$  (Eq. 1) when  $\theta = 0.9$ . Eq. 2 then gives the bound on rewards:  $R_B > 5$  and  $R_C > 1$ , which are affordable as the normal-path incentives the rollup operator needs to provide. Eq. 3 calculates the minimum stake  $\alpha_0 \mathcal{S} > 100009$ , assuming that  $c_V = 100000 \gg c_T$ .

To induce lazy collusion, a leader in LC-Game will set the stake  $t > 18514$  according to Eq. 9. In LC-Game, for any  $t \geq 0$ , the condition in Theorem 5 always holds. To eliminate all collusion strategies, according to Eq. 10, the reward for the whistleblower should be set to  $R_w > 120572$ .

If we increase the number of watchtowers to  $n = 100$ , we see that the bounty  $R_B$  increases accordingly to at least 44, which is still low. The minimum stake required for each watchtower does not change significantly, and  $R_w$  decreases to 102281.

Under the rational adversary assumption, our protocol guarantees a unique pure strategy Nash equilibrium where all watchtowers are diligent. Beyond this, cryptoeconomic security requires that when an attack occurs, the cost of launching the attack exceeds the maximum

profit. Therefore, when designing the actual parameters (e.g.  $n$ ,  $\mathcal{S}$ ) for a practical system, we can utilize signals on the inherent value of transactions [12] to adapt the security requirements of watchtowers. For example, considering the current average transaction fee on Ethereum is approximately \$3 and the average L2 batch size is around 200, we simplify the model by assuming all transactions are executed on L1 to resolve disputes, which incurs the  $c_V \approx \$600$  and  $c_T \approx \$0.006$ . (Using other more complex dispute resolution methods can reduce this cost.) Then the normal path reward for watchtowers can be estimated as:

$$R_B > 5 \times \$0.006 = \$0.03 \text{ per batch.}$$

Next, we calculate the number of batches produced by some L2 chains per year. For example, Optimism processes approximately 400,000 transactions per day, translating to about 2,000 batches per day or 700,000 batches per year. Given the probability that a watchtower wins the bounty is 0.2 and the annual percentage yield (APY) from external investment vehicles is around 6%, the condition to incentivize stakers with the estimated return rate is:

$$\frac{(5 - 1) \times 0.006 \times 0.2 \times 700,000}{600 + \text{tx value}} > 6\%.$$

In other words, a watchtower would be willing to secure approximately \$56,000 worth of transactions. And if the application aims to incentivize watchtowers to secure higher value transactions, the rewards should be increased accordingly.

Additionally, the minimum stake that each watchtower needs to post should include the transaction value. Notably,  $n$  determines the normal operational overhead of our protocol, which does not directly determine the security. It can be chosen with a trade-off between stake decentralization and operational cost.

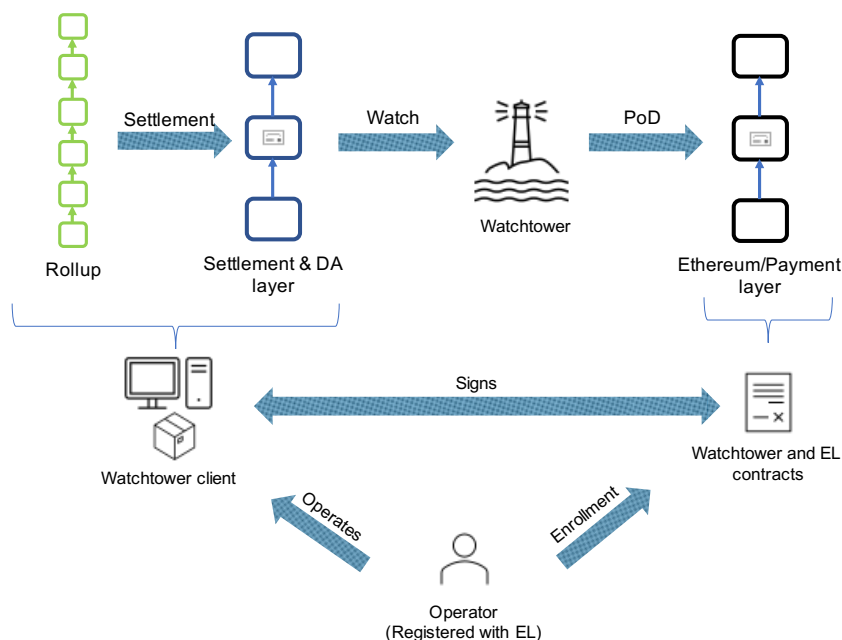
## 6 Implementation and Evaluation

### 6.1 System design

The protocol described so far is a general framework for proving diligence in a computing platform with anytrust guarantees. We describe an implementation of these proofs subsequently in the context of optimistic rollups (ORs) as the compute platform and Eigenlayer [14] as the underlying staking platform under a non-collusion setting. The complete system is termed a watchtower network and serves the following implementation goals:

1. *Low compute overhead:* Watching an OR state involves executing all transactions of the rollup. Overhead is termed as any resource cost on top of the bare minimum rollup state execution. Our implementation minimizes this overhead to lower a watchtower's resource costs.
2. *Modular implementation:* The rollup ecosystem has a lot of tech stacks for full nodes ranging from general OP-stack to specialized implementations for DeFi, such as LayerN. Our modular implementation can be used on any rollup stack with minimal modifications.
3. *Low gas fees:* Large gas fees on settlement layers such as Ethereum can make watching prohibitively expensive. Our implementation scales down L1 gas costs and makes it an adjustable feature for the rollup.

The implementation is split across the functional domains of the rollup, settlement, payment, and staking layers. For simplicity, we assume the settlement, payment, and staking layers sit on the same ledger. However, it can be easily expanded to independent networks if desired. Figure 3 shows the binding of the functional components with the two



■ **Figure 3** Watchtower client executes the rollup and observes the commitments on settlement layer, it posts bounty and flags on the payment/stake layer.

architectural components: the Watchtower client running on a server and smart contracts running on Ethereum. We outline the details of the two architectural components below. Our implementation draws from the modules in section 6.1 to build a watchtower on the Optimism Bedrock stack [50]. We test the implementation on the Ethereum Goerli testnet to watch OP-goerli and Base-goerli. We evaluate the system as per our quantitative implementation goals as described in section 6.1: *Compute overhead*, and *gas costs*. We adapt the modules to fit the existing rollup stacks and deploy them using an update-optimized architecture to make them evolve with the rollup ecosystem. We go over these details in Appendix A<sup>1</sup>.

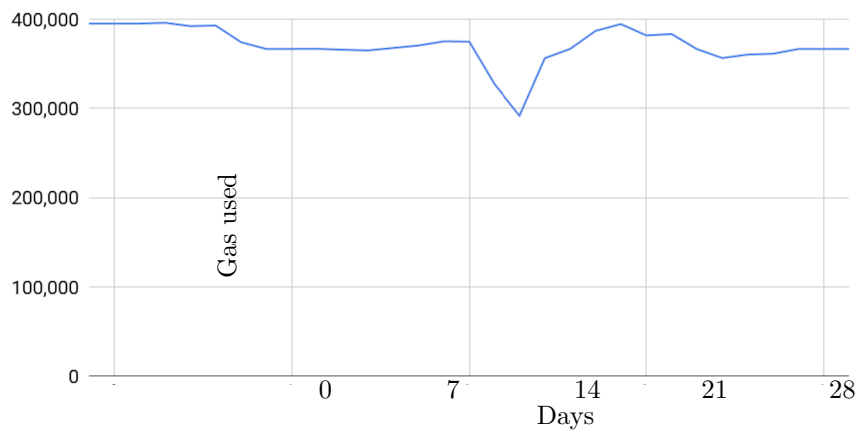
## 6.2 On-chain Contracts

The contracts are written in Solidity and deployed on Ethereum Goerli via a UUPS proxy architecture [37] to enable future updates. We deploy three contracts derived from section 6.1: *OperatorRegistry*, *BountyManager*, and *AlertManager*. The deployed contracts can be found at [4].

*BountyManager* contract implements a hash minimum across watchtowers to ensure a single bounty winner within an epoch. The payment pool and Rollup registry contract are replaced with a simple bounty count to measure contribution and a chain-id to point to a rollup. Dispute resolution contracts will utilize L2 fraud proofs once they evolve.

**Optimizations.** Immutable data like *diligenceProof* are stored as calldata variables to avoid storage costs. Proof outputs are set to a fixed size using *keccak256* to ensure consistency in storage requirements. Hash minima is calculated by the winning party to balance gas costs. We use *Mappings* instead of arrays to store hash data to reduce gas costs. We utilize audited

<sup>1</sup> Appendix is in the full version of the paper: <https://arxiv.org/abs/2402.07241>.



■ **Figure 4** Gas usage of *MineBounty* operation over 4 weeks of deployment.

and optimized ECDSA OpenZeppelin libraries [36] to verify the authenticity of signed proofs to reduce contract risk. The contracts emit *NewBountyClaimed* and *NewBountyRewarded* for efficient notifications to off-chain clients.

**Gas usage.** A bounty mining transaction consumes 380K gas on average as shown in figure 4. As a reference, this is of the same order as a token swap operation on Uniswap. Two significant contributors to gas usage are (a) Proof storage for reward reimbursement and (b) Verifying the authenticity of proofs. This gas fees can be reduced in the future by introducing an appropriate aggregation layer for submitting bounty mining transactions.

### 6.3 Off-chain Client

The off-chain clients are implemented to support OP-Bedrock as a rollup execution engine. Two clients implemented in Golang enable the Watchtower client. A bounty mining client contains the bounty mining and transaction generator modules. This client is supported by a State extraction client that contains the settlement layer module with an RPC connection to a full node. Alert management and reconfiguration manager modules are left as future work to be implemented once the rollup stack evolves. We describe the implementation details of the Go clients below:

**Optimizations.** We employ an event-based trigger to the PoD generation. The watchtower client listens to emitted events on the *L2OutputOracle* Contract to receive real-time data updates from the contract; this is much more efficient than polling mechanisms. Generating execution trace requires storing the state roots after each transaction; this operation involves re-execution and hence is limited to just the penultimate block in an L2 epoch to avoid re-executing the whole epoch.

**Resource utilization.** The L2 full node and watchtower client are implemented on different machines. We run the L2 full node on a machine with 4 cores and 16GB of RAM and the watchtower client on a machine with 2 cores and 4GB RAM. Table 3 lists the resource costs in running the watchtower client. We observe that the client consumes minimal resources natively and has a minimal resource usage overhead on the L2 node. The client has no I/O since its context is stored in memory.

■ **Table 3** Watchtower client usage stats.

L2 node	CPU(%)	Mem(MB)	I/O(KB/s)	NW(KB/s)
<b>Before mining</b>	0.7-25	5306	700	10-150
<b>During mining</b>	0.7-25	5307	700	10-150
Watchtower client	CPU(%)	Mem(MB)	I/O(KB/s)	NW(KB/s)
<b>Before mining</b>	0.1	15	0	1
<b>During mining</b>	0.1 - 10	15	0	1

We observe CPU usage burst to 10% on the watchtower client when the rollup proposer on L1 output Oracle proposes a new block. We observe a similar burst of 25% on the L2 goerli node when op-node sends a block to op-geth to execute the transactions and update the state.

We can utilize this capacity for additional off-chain computing to redistribute some on-chain contract operations to an off-chain module. A proposed approach is to perform proof aggregation on-chain; this implementation is left for future work.

## 7 Discussion

Proof of Diligence ensures that watchtower network executes all transactions on the rollup diligently. Further improvements down the line will enhance security, enable generalized applications, and allow for efficient trade-offs between delay and stake. We describe such improvements below:

### 7.1 Enabling Cryptoeconomically Secure Watchtower Applications

The design described so far ensures that watchtowers are independently verifying transactions on rollups. The verification results can be utilized for attesting to any event on the rollup. These attestations are cryptoeconomically secured by the watchtower's stake locked with EigenLayer. We summarize the design here:

- **Configurable execution event trace:** Applications can subscribe to the Watchtower network to get verifiable updates on their transactions' life cycle. The events emitted from these transactions will be added to the bounty to ensure the execution and can be challenged for cryptoeconomic security.
- **Application event tracing:** Watchtowers can trace the whole life cycle of transactions pertaining to an application, starting from being sequenced by the sequencer to being ordered on L1 to being asserted into the state. A different level of cryptoeconomic security will accompany each of these stages.
- **Dispute resolution and cryptoeconomic security:** Events pertaining to the subscribed application are attested by the watchtower and are bound to be included in the next bounty. As enforced by the proof of diligence, other watchtowers will ensure that these attestations are correct. If these attestations are exchanged in private, An application/agent consuming this attestation can contest its correctness in the future by showing a mismatch between the watchtower's attestation and the mined bounty.

Besides applications on the rollup, the incentivized watchtower network holds significant potential for broader applications in general verifiable computing. Our future work will explore how the Proof of Diligence protocol can be extended to various domains such as AI inference [6], cloud computing [13], and blockchain light client protocols [31]. For example,



to monitor and verify AI inference tasks, each watchtower can independently re-compute the inference results from the provided input data and model parameters, raising an alert if discrepancies are found. A recent work [31] demonstrates the use of watchtowers as a monitoring service to secure proof of stake blockchain states for resource-limited light clients, ensuring any invalid states are detected. This approach can be extended to a wider range of blockchain applications where light clients are prevalent. Watchtowers provide a robust layer of security and accountability.

## 7.2 Enhancing Security

The current system design ensures that Proof of Diligence works under a static adversary that can form static collusion. Resistance against a stronger dynamic adversary requires assumptions on rational independence of watchtowers and the privacy of whistleblower contracts. These assumptions can be enforced through system design by enabling random rotation of watchtowers in the pool and ensuring the privacy of the whistleblower.

**Watchtower rotation.** The rotation of watchtowers across different rollups in the pool is essential for security against an adaptive adversary. Watchtowers can be periodically rotated in small batches across rollups in a random and staggered manner reminiscent of the cuckoo rule [5]. The rotation can be made more efficient by utilizing two techniques: (a) *utilizing modularity*: we are designing an efficient reconfiguration protocol for watchtowers rotating between rollup two rollups sharing similar modules - such as two rollups running the OP stack; (b) *stateless clients*: watchtower rotation through the reconfiguration manager can be made very efficient by removing the need to transfer state. The witness chain team is developing a stateless client architecture that removes the need to download state when reconfiguring to a new rollup.

**Private Whistleblower contracts.** The interactions of the whistleblower with the whistleblower contract are private to ensure that they can't be used within the collusion contract. We are designing a system to ensure these inputs stay private to the collusion contract by deploying a whistleblower contract upon request post the bounty mining period. This ensures that the address of the whistleblower contract is not static and cannot be referenced in the collusion contract. Alternate design solutions include privacy-enhancing contract structures such as Aleo.

---

### References

- 1 The merge. <https://ethereum.org/en/roadmap/merge>, 2023. Accessed: 2023-02-04.
- 2 John Adler. Minimal viable merged consensus. <https://ethresear.ch/t/minimal-viable-merged-consensus/5617>, 2019. Accessed on Oct 17, 2023.
- 3 Mustafa Al-Bassam, Alberto Sonnino, and Vitalik Buterin. Fraud and data availability proofs: Maximising light client security and scaling blockchains with dishonest majorities. *arXiv preprint*, 2018. [arXiv:1809.09044](https://arxiv.org/abs/1809.09044).
- 4 Anonymous. Proof of diligence contracts. <https://goerli.etherscan.io/address/0x1BF313AADe1e1f76295943f40B558Eb13Db7aA99>.
- 5 Baruch Awerbuch and Christian Scheideler. Towards a scalable and robust dht. In *Proceedings of the eighteenth annual ACM symposium on Parallelism in algorithms and architectures*, pages 318–327, 2006.

- 6 Suma Bhat, Canhui Chen, Zerui Cheng, Zhixuan Fang, Ashwin Hebbar, Sreeram Kannan, Ranvir Rana, Peiyao Sheng, Himanshu Tyagi, Pramod Viswanath, et al. Sakshi: Decentralized ai platforms. *arXiv preprint*, 2023. [arXiv:2307.16562](https://arxiv.org/abs/2307.16562).
- 7 Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget. *arXiv preprint*, 2017. [arXiv:1710.09437](https://arxiv.org/abs/1710.09437).
- 8 Jing Chen and Silvio Micali. Algorand. *arXiv preprint*, 2016. [arXiv:1607.01341](https://arxiv.org/abs/1607.01341).
- 9 CoinTelegraph. Ethereum upgrades: A beginner’s guide to eth 2.0. <https://cointelegraph.com/learn/ethereum-upgrades-a-beginners-guide-to-eth-2-0>, 2020. Accessed: 2023-02-04.
- 10 George Danezis, Lefteris Kokoris-Kogias, Alberto Sonnino, and Alexander Spiegelman. Narwhal and tusk: a dag-based mempool and efficient bft consensus. In *Proceedings of the Seventeenth European Conference on Computer Systems*, pages 34–50, 2022.
- 11 Bernardo David, Peter Gazi, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *Advances in Cryptology–EUROCRYPT 2018: 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29-May 3, 2018 Proceedings, Part II 37*, pages 66–98. Springer, 2018.
- 12 Soubhik Deb, Robert Raynor, and Sreeram Kannan. Stakesure: Proof of stake mechanisms with strong cryptoeconomic safety. *arXiv preprint*, 2024. [arXiv:2401.05797](https://arxiv.org/abs/2401.05797).
- 13 Changyu Dong, Yilei Wang, Amjad Aldweesh, Patrick McCorry, and Aad Van Moorsel. Betrayal, distrust, and rationality: Smart counter-collusion contracts for verifiable cloud computing. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 211–227, 2017.
- 14 EigenLabs. Eigenlayer, 2023. Accessed: 2024-01-27. URL: <https://www.eigenlayer.xyz/>.
- 15 Dankrad Feist. Proofs of custody. <https://dankradfeist.de/ethereum/2021/09/30/proofs-of-custody.html>, 2021. Accessed on Oct 17, 2023.
- 16 Ed Felten. Cheater checking: How attention challenges solve the verifier’s dilemma. <https://medium.com/offchainlabs/cheater-checking-how-attention-challenges-solve-the-verifiers-dilemma-681a92d9948e>, 2019. Accessed on Oct 17, 2023.
- 17 Bingyong Guo, Zhenliang Lu, Qiang Tang, Jing Xu, and Zhenfeng Zhang. Dumbo: Faster asynchronous bft protocols. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 803–818, 2020.
- 18 Abdelatif Hafid, Abdelhakim Senhaji Hafid, and Mustapha Samih. Scaling blockchains: A comprehensive survey. *IEEE access*, 8:125244–125262, 2020.
- 19 Harry Kalodner, Steven Goldfeder, Xiaoyi Chen, S Matthew Weinberg, and Edward W Felten. Arbitrum: Scalable, private smart contracts. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1353–1370, 2018.
- 20 Julia Koch and Christian Reitwiessner. A predictable incentive mechanism for truebit. *arXiv preprint*, 2018. [arXiv:1806.11476](https://arxiv.org/abs/1806.11476).
- 21 Georgios Konstantopoulos. How does optimism’s rollup really work? <https://research.paradigm.xyz/optimism>, 2021. Accessed on Oct 17, 2023.
- 22 Caldera Lab. Caldera: The rollup platform. <https://caldera.xyz/>, 2023. Accessed on Oct 17, 2023.
- 23 Offchain Labs. Arbitrum rollup. <https://arbitrum.io/rollup>, 2018. Accessed on Oct 17, 2023.
- 24 Loi Luu, Jason Teutsch, Raghav Kulkarni, and Prateek Saxena. Demystifying incentives in the consensus computer. In *Proceedings of the 22Nd acm sigsac conference on computer and communications security*, pages 706–719, 2015.
- 25 Akaki Mamageishvili and Edward W Felten. Incentive schemes for rollup validators. In *The International Conference on Mathematical Research for Blockchain Economy*, pages 48–61. Springer, 2023.

- 26 Michael Maschler, Shmuel Zamir, and Eilon Solan. *Game theory*. Cambridge University Press, 2013.
- 27 Patrick McCorry, Chris Buckland, Bennet Yee, and Dawn Song. Sok: Validating bridges as a scaling solution for blockchains. *Cryptology ePrint Archive*, 2021.
- 28 Ralph C Merkle. A digital signature based on a conventional encryption function. In *Conference on the theory and application of cryptographic techniques*, pages 369–378. Springer, 1987.
- 29 Juhi Mirza. Ethereum 2.0 transactions per second: Ethereum will reach 100,000 tps after upgrade, says vitalik buterin. *Gfinity Esports*, August 2022. URL: <https://www.gfinityesports.com/cryptocurrency/ethereum-2-transactions-per-second/>.
- 30 Gianmaria Del Monte, Diego Pennino, and Maurizio Pizzonia. Scaling blockchains without giving up decentralization and security: A solution to the blockchain scalability trilemma. In *Proceedings of the 3rd Workshop on Cryptocurrencies and Blockchains for Distributed Systems*, pages 71–76, 2020.
- 31 Niusha Moshrefi, Peiyao Sheng, Soubhik Deb, Sreeram Kannan, and Pramod Viswanath. Unconditionally safe light client. *arXiv preprint*, 2024. arXiv:2405.01459.
- 32 Shashank Motepalli, Luciano Freitas, and Benjamin Livshits. Sok: Decentralized sequencers for rollups. *arXiv preprint*, 2023. arXiv:2310.03616.
- 33 Mahmudun Nabi, Sepideh Avizheh, Muni Venkateswarlu Kumaramangalam, and Reihaneh Safavi-Naini. Game-theoretic analysis of an incentivized verifiable computation system. In *Financial Cryptography and Data Security: FC 2019 International Workshops, VOTING and WTSC, St. Kitts, St. Kitts and Nevis, February 18–22, 2019, Revised Selected Papers 23*, pages 50–66. Springer, 2020.
- 34 Kamilla Nazirkhanova, Joachim Neu, and David Tse. Information dispersal with provable retrievability for rollups. In *Proceedings of the 4th ACM Conference on Advances in Financial Technologies*, pages 180–197, 2022.
- 35 Akira Okada. The possibility of cooperation in an n-person prisoners’ dilemma with institutional arrangements. *Public Choice*, 77(3):629–656, 1993.
- 36 OpenZeppelin. OpenZeppelin-contracts. URL: <https://github.com/OpenZeppelin/openzeppelin-contracts>.
- 37 Hadrien Croubois Santiago Palladino, Francisco Giordano. Erc-1967: Proxy storage slots. <https://eips.ethereum.org/EIPS/eip-1967>, April 2019.
- 38 Yonatan Sompolinsky and Aviv Zohar. Secure high-rate transaction processing in bitcoin. In *Financial Cryptography and Data Security: 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers 19*, pages 507–527. Springer, 2015.
- 39 Altlayer Team. Altlayer: A decentralized interlayer for rollups. <https://altlayer.io/>, 2023. Accessed on Oct 17, 2023.
- 40 Base Team. Base. <https://base.org/>, 2023. Accessed on Oct 17, 2023.
- 41 BNB Chain Team. opbnb: High-performance optimistic layer 2 solution for bnb smart chain. <https://opbnb.bnbchain.org/en>, 2023. Accessed on Oct 17, 2023.
- 42 Conduit Team. Conduit. <https://conduit.xyz/>, 2023. Accessed on Oct 17, 2023.
- 43 Eclipse Team. Eclipse. <https://www.eclipse.builders/>, 2023. Accessed on Oct 17, 2023.
- 44 LayerN Team. Layer n: Ethereum’s financial superlayer. <https://www.layern.com/>, 2023. Accessed on Oct 17, 2023.
- 45 Linea Team. Linea. <https://linea.build/>, 2023. Accessed on Oct 17, 2023.
- 46 Optimism Team. Optimism. <https://www.optimism.io/>, 2020. Accessed on Oct 17, 2023.
- 47 Optimism Team. Optimism bedrock stack, 2023. Accessed: 2024-01-27. URL: <https://community.optimism.io/docs/developers/bedrock/>.
- 48 The Diem Team. Diembft v4: State machine replication in the diem blockchain. <https://developers.diem.com/papers/diem-consensus-state-machine-replication-in-the-diem-blockchain/2021-08-17.pdf>, 2021. Accessed on April 19, 2023.

## 5:24 Proof of Diligence: Cryptoeconomic Security for Rollups

- 49 Jason Teutsch and Christian Reitwießner. A scalable verification solution for blockchains. *arXiv preprint arXiv:1908.04756*, 2019.
- 50 The Optimism Collective. The Optimism Monorepo. URL: <https://github.com/ethereum-optimism/optimism>.
- 51 Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.
- 52 Sage D. Young. Layer 2 network arbitrum surpasses ethereum in daily transactions. *CoinDesk*, February 2023. URL: <https://www.coindesk.com/tech/2023/02/22/layer-2-network-arbitrum-surpasses-ethereum-in-daily-transactions/>.
- 53 Mingchao Yu, Saeid Sahraei, Songze Li, Salman Avestimehr, Sreeram Kannan, and Pramod Viswanath. Coded merkle tree: Solving data availability attacks in blockchains. In *International Conference on Financial Cryptography and Data Security*, pages 114–134. Springer, 2020.

# Analyzing and Benchmarking ZK-Rollups

**Stefanos Chaliasos**

Imperial College London, UK

**Itamar Reif**

Astria, New York, NY, USA

**Adrià Torralba-Agell**

Universitat Oberta de Catalunya, San Martí, Spain

**Jens Ernstberger**

Technische Universität München, Germany

**Assimakis Kattis**

Athens, Greece

**Benjamin Livshits**

Imperial College London, UK

Matter Labs, London, UK

---

## Abstract

As blockchain technology continues to transform the realm of digital transactions, scalability has emerged as a critical issue. This challenge has spurred the creation of innovative solutions, particularly Layer 2 scalability techniques like rollups. Among these, ZK-Rollups are notable for employing Zero-Knowledge Proofs to facilitate prompt on-chain transaction verification, thereby improving scalability and efficiency without sacrificing security. Nevertheless, the intrinsic complexity of ZK-Rollups has hindered an exhaustive evaluation of their efficiency, economic impact, and performance.

This paper offers a theoretical and empirical examination aimed at comprehending and evaluating ZK-Rollups, with particular attention to ZK-EVMs. We conduct a qualitative analysis to break down the costs linked to ZK-Rollups and scrutinize the design choices of well-known implementations. Confronting the inherent difficulties in benchmarking such intricate systems, we introduce a systematic methodology for their assessment, applying our method to two prominent ZK-Rollups: Polygon zkEVM and zkSync Era. Our research provides initial findings that illuminate trade-offs and areas for enhancement in ZK-Rollup implementations, delivering valuable insights for future research, development, and deployment of these systems.

**2012 ACM Subject Classification** Security and privacy → Cryptography

**Keywords and phrases** Zero-Knowledge Proofs, ZK-Rollups, Benchmarking, Blockchain Scalability

**Digital Object Identifier** 10.4230/LIPIcs.AFT.2024.6

**Supplementary Material** *Software (Source Code)*: <https://github.com/StefanosChaliasos/zkr-ollup-benchmarking>, archived at [swh:1:dir:8e774f5d4210d196c71ac56376ce0bca56d956ac](https://swh.io/1/dir/8e774f5d4210d196c71ac56376ce0bca56d956ac)

**Funding** We thank the Ethereum Foundation and The Latest in DeFi Research (TLDR) program, funded by the Uniswap Foundation, for supporting this work.

*Adrià Torralba-Agell*: Project PID2021-125962OB-C31 SECURING/CYBER, funded by the Ministerio de Ciencia e Innovación, la Agencia Estatal de Investigación and the European Regional Development Fund (ERDF).

**Acknowledgements** We thank Ramon Canales, Emil Luta, Roman Brodetski, Robert Remen, Fractasy Romero, Ignasi Ramos, Héctor Masip Ardevol, and Carlos Matallana for their insightful feedback and help with technical issues. We also thank Cristina Pérez-Solà for proofreading this manuscript and supervising Adrià Torralba-Agell while working on this project.



© Stefanos Chaliasos, Itamar Reif, Adrià Torralba-Agell, Jens Ernstberger, Assimakis Kattis, and Benjamin Livshits;

licensed under Creative Commons License CC-BY 4.0

6th Conference on Advances in Financial Technologies (AFT 2024).

Editors: Rainer Böhme and Lucianna Kiffer; Article No. 6; pp. 6:1–6:24

Leibniz International Proceedings in Informatics



Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1 Introduction

Blockchain technology, with leading chains such as Bitcoin [23] and Ethereum [38], has introduced novel solutions for finance and various applications, reshaping the landscape of digital transactions by removing the need for centralized entities. However, the surge in their adoption has brought to light a critical challenge: scalability. The inherent limitation in the number of transactions these networks can process per second has prompted an effort within the blockchain community to seek and develop a plethora of innovative solutions [41].

Two dominant strategies have enjoyed practical adaptation in recent years to address scalability. The first involves the creation of new, modern blockchains designed from the ground up to process transactions more efficiently than their predecessors [39, 5], albeit at the cost of missing the established security and network effects of blockchains like Ethereum. The second strategy revolves around Layer 2 (L2) solutions, or off-chain scalability solutions, with rollups being the most promising and widely adopted in practice [36]. Rollups work by executing transactions on a faster, secondary blockchain (L2) and then posting the resulting state root, along with transaction data, back to the main blockchain – Layer 1 (L1). This ensures the integrity of the rollup’s state is verifiable and secure, leveraging the underlying blockchain’s security.

Among the various rollup approaches, two stand out: optimistic [17] and ZK-Rollups [3]. Optimistic rollups rely on a system of trust and fraud proofs to validate state transitions, which introduces a delay in withdrawals due to the required challenge period. In contrast, ZK-Rollups utilize Zero-Knowledge Proofs (ZKPs) for immediate on-chain verification of state transitions, enhancing both scalability and efficiency without compromising the security of the L1 chain. Despite their advantages, ZK-Rollups introduce additional complexity and, to date, there has been limited research focused on a thorough evaluation of their overall efficiency, limitations, and economics.

Benchmarking ZK-Rollups presents a multifaceted challenge. The deployment of these systems is inherently complex, and their diverse design choices complicate direct comparisons. Further, identifying common payloads for benchmarking and establishing appropriate metrics are non-trivial tasks. In response to these challenges, this work embarks on a comprehensive theoretical and empirical analysis of ZK-Rollups. We dissect the operational and per-transaction costs of ZK-Rollups, examine the design decisions of prominent implementations, and propose a methodology for their benchmarking. This includes addressing the challenges inherent in benchmarking these systems, defining key research questions, and developing a reproducible methodology to ensure that our findings are publicly accessible and verifiable.

The results of this study aim to illuminate the trade-offs inherent to different ZK-Rollup implementations, offering insight into their advantages and areas in need of improvement. By providing a deeper understanding of the economics underpinning these systems, we hope to inform efforts to decentralize currently centralized systems. Furthermore, as Rollups as a Service continues to grow, our analysis seeks to arm users and buyers with the knowledge necessary to make their decisions, enabling them to compare different rollups using our benchmarking infrastructure tailored to their specific needs.

**Research Questions.** Next, we outline a series of research questions that will shape our analysis of ZK-Rollups. These questions are designed to uncover critical insights into the performance, cost structure, and overall efficiency and profitability margins of ZK-Rollups. Our investigation aims to provide a comprehensive understanding of these systems.

- RQ1 Fixed Costs:** What are the fixed costs associated with ZK-Rollups? Specifically, what are the expenses related to settling on L1, such as committing batches and verifying proofs? Additionally, what is (if any) the constant proving cost per batch (e.g., aggregation or compressing a Scalable Transparent Argument of Knowledge (STARK) into a Succinct Non-interactive Argument of Knowledge (SNARK))?
- RQ2 Marginal Costs:** How long does it take to prove a batch, and what is the cost associated with proving a batch? What are the data availability (DA) costs in terms of bytes posted? How do state diffs compare to posting transaction data?
- RQ3 Trade-off Between Fast Finality and Cost Minimization:** In ZK-Rollups, achieving transaction finality requires verifying the proof of the batch that includes the transaction on L1. This necessitates producing the proof first. There is a trade-off between having large, compact batches that are slower to prove and smaller batches that are faster to prove but potentially lead to less amortization of costs.
- RQ4 Cost Breakdown:** How are costs distributed across different components of a ZK-Rollup transaction, including DA, proof generation, L1 posting, and verification? Understanding this distribution is crucial for identifying areas for optimization.
- RQ5 Impact of EIP-4844:** How has the introduction of EIP-4844 influenced the cost dynamics of ZK-Rollups? This question explores the effects of EIP-4844 on the cost efficiency and practicality of ZK-Rollups.

## 1.1 Contributions

- **Qualitative Analysis of ZK-Rollups' design choices:** We conduct a comprehensive theoretical analysis of ZK-Rollups, detailing the costs associated with processing transactions and examining the diverse design choices across different implementations.
- **Towards Benchmarking ZK-Rollups:** Addressing the significant challenges inherent in benchmarking ZK-Rollups, we develop and present a structured methodology for their evaluation. This includes the implementation of our benchmarking approach on prominent implementations such as Polygon ZK-EVM and zkSync Era, providing a blueprint for systematic assessment of ZK-Rollups' efficiency and costs.
- **Results and Insights:** Offering findings from our benchmarking efforts, we aim to contribute to the ongoing discourse on ZK-Rollups by identifying key factors that influence their development and pinpointing areas in need of improvement. These preliminary results are intended to guide future research and development efforts in the field.

## 2 Background

### 2.1 Scaling Blockchain and Rollups

Blockchain scalability has been a persistent challenge, particularly for established networks like Ethereum [38], which processes only tens of transactions per second.<sup>1</sup> Efforts to enhance scalability have focused on two primary strategies: base layer scaling and L2 scaling solutions. Base layer scaling, which includes techniques such as sharding and novel consensus protocols, involves either the modification of existing blockchains – a complex and daunting task – or the development of new blockchain architectures. Although modern blockchains such as Solana [39] and Sui [5] have shown success, they often lack the established security, liquidity, and comprehensive ecosystem found in legacy blockchains like Ethereum.<sup>2</sup>

<sup>1</sup> <https://l2beat.com/scaling/activity>

<sup>2</sup> According to <https://defillama.com/chains> (accessed: 10/5/2024), Ethereum has 57.85% of the total TVL for all chains, while Solana has only 4.46%.

L2 scaling solutions, on the other hand, offer a promising avenue for scalability without altering the base layer, i.e., L1. Among these solutions, payment channels [1, 2, 20, 35], Plasma [30], and rollups [36] have been the most prevalent solutions. Payment channels enable instant, bi-directional payments between two parties by establishing a network of interconnected channels, exemplified by Bitcoin’s Lightning Network. However, they require capital lockup and constant base layer monitoring, making them suitable for specific, long-term use cases. Plasma attempted to solve various issues in different ways. Sguanci et al. [34] provides an overview of the main types of Plasma constructions. However, every attempt at Plasma had some trade-off that resulted in a poor user experience (e.g., long withdrawal periods, mass-exit problems in case of operator misbehavior, and users having to monitor all transactions on Plasma chains).

Rollups have emerged as hybrid L2 solutions, distinguishing themselves by offloading computation off-chain while retaining data on-chain, thus addressing the data availability issue while inheriting L1’s security. Rollups batch and execute transactions on an auxiliary L2 blockchain that uses the same VM as L1 or a different one. This separation of transaction execution from consensus allows rollups to process significantly more transactions per second than their L1 counterparts. By submitting a summary of the rollup’s state – typically, the root of a Merkle tree – to a smart contract in the underlying blockchain, rollups not only ensure data availability, but also inherit the security properties of the L1 network when some critical L2 mechanisms are implemented [10]. Altering the L2 state recorded on L1 would require breaking the security of L1, making it both difficult and costly. This architecture enables rollups to offer an efficient, secure scaling solution for legacy blockchains. Notably, this model, i.e., rollup-centric scaling,<sup>3</sup> has gained traction as the *principal* method for scaling Ethereum, with two predominant variants: optimistic rollups [17] and ZK-Rollups [3].<sup>4</sup>

Optimistic rollups operate on a principle of trust, where state transitions are accepted without immediate verification, relying instead on fraud proofs to challenge incorrect state updates. This approach, while efficient, requires a challenge period, introducing a delay in withdrawals. In contrast, ZK-Rollups leverage ZKPs to verify state transitions on-chain, offering a more immediate and efficient validation process without the need for a challenge period. This method not only improves the scalability and efficiency, but also maintains the integrity and security of the L1 chain.

## 2.2 ZK-Rollup Components and Transaction Lifecycle

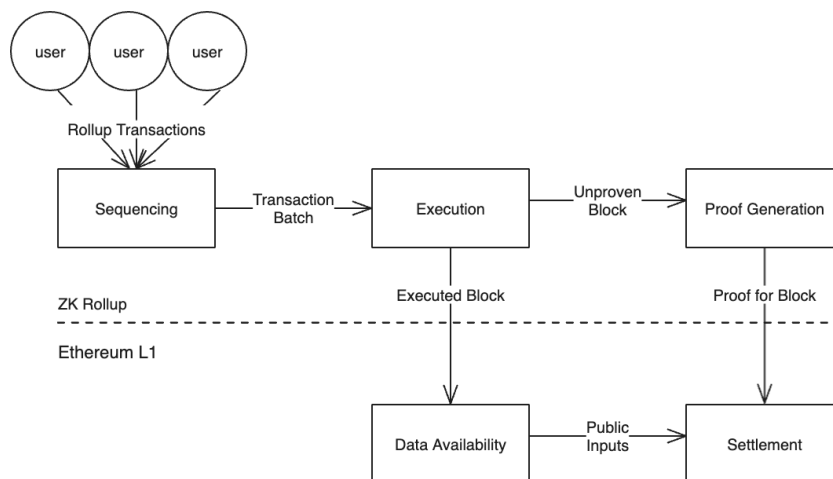
In this section, we outline the main components of a ZK-Rollup. For simplicity, we abstract out certain details, including bridging and forced transactions [15]. In addition, we do not discuss various sequencing methodologies, such as decentralized or shared sequencers [22]. Figure 1 illustrates the key components involved in processing transactions within a ZK-Rollup. Users initiate the process by signing and submitting transactions to the L2 network. A sequencer then undertakes the tasks of processing, ordering, executing, and batching these transactions. In some architectures, these functions may be distributed across different components. Subsequently, the sequencer forwards these batches to a relayer, which posts their resultant state to the L1 rollup contract. Concurrently, the sequencer sends the batch to a coordinator (or aggregator), which, in turn, sends the batch to the prover. In most ZK-Rollups, the coordinator consolidates multiple proofs into a single aggregated proof (i.e., a proof of proofs) and submits this final proof to the rollup contract. The contract then verifies the proof, finalizing the state of the L2 as immutable and verified in the L1.

---

<sup>3</sup> <https://ethereum-magicians.org/t/a-rollup-centric-ethereum-roadmap/4698>

<sup>4</sup> As of 18/3/2024, rollups have more than 34B USD TVL according to <https://12beat.com>.





■ **Figure 1** High-level simplified overview of transaction processing in ZK-Rollups.

### Components.

- **Sequencer.** The sequencer provides users with the first confirmation of transaction inclusion and ordering. It is the entity that aggregates user transactions into blocks and batches, either by providing them with a transaction submission endpoint or by pulling transactions from the mempool. It provides the canonical sequence of transactions that will be fed into the state transition function. Currently, for the systems discussed in this work, sequencer implementations rely on a trusted operator that provides this service and to which users submit their transactions. Many of the projects discussed are working on decentralizing their sequencer design, but the design space remains nascent.
- **Execution.** For most existing rollups, execution is typically done by the same entity as the sequencer. The transaction batch created by the sequencer is taken as input to the rollup’s state transition function (STF), which is executed to create both the resulting block(s) and related state root(s), as well as the transaction batches, and potentially the intermediate state snapshots (state diffs) that are required for proof generation.
- **Data Availability.** Data availability (DA) refers to the ability of clients of the blockchain protocol to retrieve the data required to verify the validity of a given batch. Traditionally provided as part of the consensus algorithm that underlies a blockchain system, the modular architecture used by rollup-based blockchain systems separates the guarantees provided by the L1 blockchain from those provided by the rollup operators. Relying on L1 for the rollup’s liveness, the data required to assert safety must be posted on the L1. DA data involve any execution artifacts required by the settlement logic for verifying the validity proof, such as transaction data or the state diff of the transactions, as well as all data required to reproduce the state of the L2.
- **Prover.** The prover is responsible for generating validity proofs for the executed batches. Given the execution artifacts, the prover creates the required witness data and executes a SNARK or STARK proof to prove the validity of the execution. Note that when a STARK is used, it is typically wrapped into a SNARK to enable efficient verification.
- **Settlement Logic.** The generated proofs are then posted to the L1 smart contract responsible for settling a given batch. An executed batch must be agreed upon as “valid” in order to coordinate between decentralized actors (the blockchain’s users). This is done by providing a block’s resulting state, proof of that state’s validity, and the inputs required for verifying the proof.

**Transaction Lifecycle Status.**

- **Pending:** A user has signed and submitted the transaction to the L2 network.
- **Preconfirmed:** The sequencer has processed the transaction and included it in a block. If users trust the sequencer, they can regard the transaction as processed. Currently, reliance on centralized sequencers enables near-instant preconfirmation, yet this raises the challenge of maintaining such efficiency without centralization. Preconfirmation significantly enhances user experience, allowing users to treat most transactions as effectively complete. However, it is important to note that for withdrawals from L2, users must await the finalization of transactions.
- **Committed:** The transaction is part of a batch committed to the L1 contract, allowing others to reconstruct the L2 state, including this transaction from the L1 data.
- **Verified/Finalized:** The batch containing the transaction has been proven, and the proof has been verified in the L1 contract, marking the transaction and its batch's state on L2 as immutable.

**2.3 Costs of ZK-Rollups**

Next, we analyze the costs associated with processing a transaction within a ZK-Rollup. Specifically, we distinguish between costs that are transaction-specific and those that are constant per batch, meaning they apply to each processed batch regardless of the number of transactions that are included in it.

We also separate out all costs associated with the transaction's fee mechanism, as these can be temporal in nature, and add an orthogonal dimension to the physical costs that are incurred per transaction (and which are a direct result of the system's design choices). To this end, we quantify costs in terms of the underlying empirical variable measured: for example, data availability costs are presented in terms of bytes rather than their actual L1 gas cost.

**Fixed Batch Costs.** Each batch carries inherent fixed costs that must be paid regardless of the number of transactions it includes.

1. **Settlement:** This involves (a) calling the Ethereum L1 contract to commit to a specific batch, and (b) submitting proofs and executing the verifier logic (e.g., SNARK verifier) for the committed batches.
2. **Proof Compression:** Some constructions involve compressing (or converting) the block's proof from one proof system to another. This typically involves proving the verification of the aggregated proof in a cheaper (with regards to the verification cost) proof system (e.g., Groth16) so that the cost of settlement is lower.

**Marginal Transaction Costs.** In addition to the batch-specific costs, each transaction included in a rollup's block incurs the following additional costs:

1. **Data Availability:** This is measured in bytes of the transaction's calldata. The calldata needs to be posted to the data availability provider, e.g., Ethereum L1, so that the rollup's state can be reconstructed.
2. **Proving Costs:** These are divided into the following: (a) Additional witness generation work required. (b) Proof generation for the transaction's execution. (c) Some constructions incorporate a final step, aggregating batches of proofs into a single proof. Additional transactions may require more aggregation work in this context.

■ **Table 1** High-level comparison of different ZK-Rollups. \*: In those ZK-EVMs, the last step moves from a STARK-based proof system to a SNARK proof. The hardware row represents the specifications recommended by each team for production use in their official documentation pages.

	ZK-Rollups					
	Polygon ZK-EVM	Scroll	zkSync Era	Starknet	RISC0 Zeth	Aztec
VM	zkEVM	zkEVM	zkEVM	General Purpose zkVM	General Purpose zkVM	Privacy-Focused zkVM
Proof System	STARK + FFLONK*	Halo2-KZG	Boojum + PLONK-KZG*	STARK + FRI	STARK + FRI	HONK + Protogalaxy + Goblin PLONK + UltraPlonk
Published Data	TX Data	TX Data	State Diffs	State Diffs	N/A	N/A
Compatibility	EVM-Compatible	EVM-Compatible	Solidity-Compatible	N/A	EVM-Compatible	N/A
Hardware	CPU-based / >128-cores / >1TB RAM	GPU-based / 4 GPUs / >48-cores / >192GB RAM	Many GPU-based / 1 GPU / 16-cores / 64GB RAM	N/A	N/A	N/A

- L2 Execution Costs:** These include computing the state transition resulting from the transaction along with any related costs due to associated long-term storage requirements. Can be thought of as the costs of operating the rollup’s infrastructure: sequencing, execution, and relaying.

### 3 Qualitative Analysis

In this section, we examine the fundamental components and design choices influencing the performance, efficiency, and complexity of various ZK-Rollups. Table 1 summarizes a qualitative overview of ZK-Rollups.

#### 3.1 Proof System

Recent advancements in proof systems have led to a significant acceleration of ZK-VMs and ZK-EVMs, with research focusing on developing proof systems to optimize for better performance and efficiency of the proving algorithm. Notable developments include zkASM [29] and PIL [28] ZK languages, developed by Polygon for their ZK-EVM; Boojum [21], developed for zkSync Era; and the halo2 KZG fork [33] implemented by the Scroll team, among others. A common strategy in ZK-VM design is to also apply recursion, a technique in which one ZK proof is verified inside of another, allowing the usage of cheaper verification circuits at the settlement phase while using more complex proof systems in the proving process. For example, Polygon ZK-EVM leverages a STARK proof to initially prove batch correctness, which is then compressed via recursion before being encapsulated in a SNARK proof for submission to L1. This method benefits from SNARKs’ efficient verification and constant proof size. Similar methodologies are used in various ZK-EVM platforms.

However, the choice of a proof system and its specific implementation can lead to some important trade-offs, particularly between the speed of proof generation and the computational resources required. This balance is crucial, as it can influence the overall performance and user experience of ZK-Rollups.

#### 3.2 Transaction Data vs. State Diffs

In L1 blockchains, all transactions in a block are stored along with the Merkle root of the final state. This information is disseminated across the network through a “gossiping” protocol [18], and the root of trust is established through re-execution and validation of the state root by the participants. For example, in proof-of-stake networks, validators stake their tokens and vote on the resulting state to ensure consensus [8].

ZK-Rollups, in contrast, establish their root of trust through the verification of a ZKP [3]. The ZKP attests to the correctness of the final state, and its validation is sufficient for participants to accept a batch of blocks as canonical. Verifying the final state requires publicly available inputs, typically either the transaction data included in the batch or intermediate state transition snapshots, known as “state diffs.” Each method has its trade-offs. State diffs are more cost-effective because they omit signatures and publish only the final state changes after multiple transactions, thus allowing for better cost amortization. However, this approach does not preserve a complete transaction history and can complicate the mechanisms of enforcing transactions and reproducing the state through data posted in the L1. Currently, ZK-Rollups such as zkSync Era and Starknet utilize state diffs due to their efficiency benefits, while solutions like Polygon ZK-EVM and Scroll opt for publishing transaction data to maintain data completeness. Both approaches are exploring innovative compression techniques to further optimize cost efficiency.

### 3.3 EVM Compatibility

Buterin identifies four main categories of ZK-EVMs [7], which are implementations of ZKP circuits that validate the correctness of Ethereum Virtual Machine (EVM) execution, ranging from fully Ethereum-equivalent to language-compatible. Fully Ethereum-equivalent ZK-EVMs replicate the EVM’s behavior and data structures precisely, ensuring seamless operation for existing Ethereum applications. EVM-equivalent ZK-EVMs maintain core functionalities but introduce slight variations in data structures while ensuring identical behavior when executing EVM-bytecode. EVM-compatible approaches might exclude certain precompiles or slightly modify the gas metering mechanism, which could affect the execution of specific transactions in edge cases.

The most flexible, language-compatible ZK-EVMs, utilize compilers to translate Solidity into different targets, optimizing efficiency and potentially enhancing functionality beyond strict EVM equivalence. This spectrum of compatibility reflects a trade-off between maintaining strict adherence to the EVM and pursuing efficiency gains or advanced features through innovation. For instance, the Polygon ZK-EVM aims for a close EVM equivalence to balance compatibility with performance improvements, while zkSync Era opts for a language-compatible approach with its `zkso1c` compiler, prioritizing efficiency and adaptability.

Another approach, not described in Buterin’s classification, is employed by RISCO’s Zeth [32]. Based on a prover for the RISC-V Instruction Set Architecture (ISA), Zeth leverages Rust and its LLVM-based compiler toolchain to utilize a suite of robust crates, such as `revm`<sup>5</sup>, `ethers`<sup>6</sup>, and `alloy`<sup>7</sup>, enabling the proof of execution for EVM-based transactions and blocks without the need for additional domain-specific implementation circuits by leveraging RISCO’s prover. This approach diverges from traditional ZK-EVM designs by proving the correctness of computations at the ISA level rather than focusing exclusively on EVM bytecode. The use of RISC-V as the underlying architecture allows for a high degree of flexibility and the potential to leverage a broader range of programming languages supported by the LLVM ecosystem. This approach results in a fully EVM-equivalent ZK-EVM.

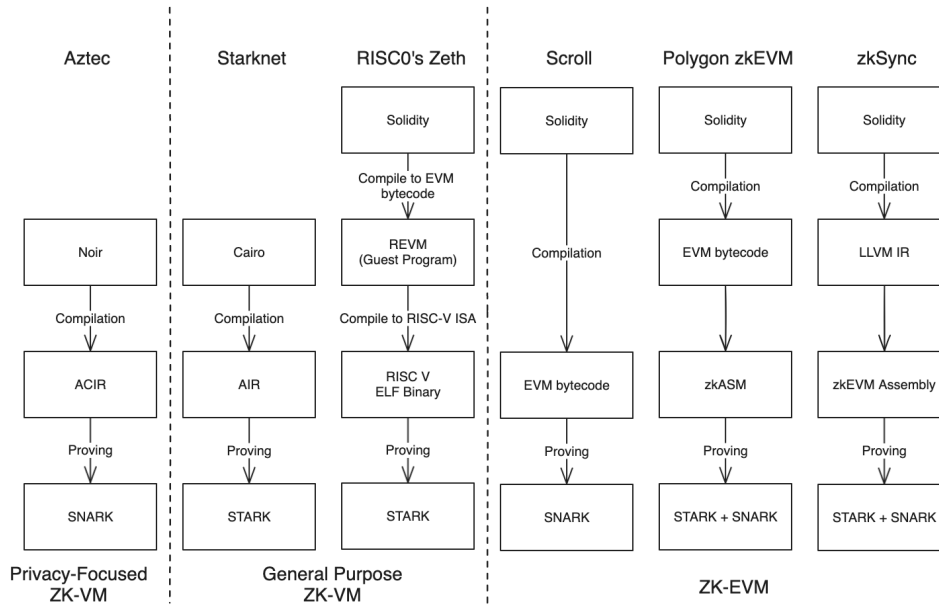
The landscape of ZK-EVMs has seen a rapid evolution of innovative approaches in recent years, though not all are “EVM compatible” to the same degree. Several implementations discussed in this section were not included in our direct performance comparison due to differing definitions of compatibility. We further discuss “EVM compatibility in Appendix A.

---

<sup>5</sup> <https://github.com/bluealloy/revm>

<sup>6</sup> <https://github.com/gakonst/ethers-rs>

<sup>7</sup> <https://github.com/alloy-rs/alloy>



■ **Figure 2** High-level overview of ZK-Rollup compilation pipelines.

### 3.4 ZK-Rollups Prover Implementations

At a high level, there are two primary approaches for ZK-Rollup prover design: assembly-based and EVM opcode-based implementations, each offering a distinct approach to handling computations and proofs. Assembly-based VMs implement a specific Instruction Set Architecture (ISA), focusing on proving the correctness of lower-level execution steps that express abstractions around the underlying proof system. This method closely aligns with traditional hardware architectures, where each instruction within the set is designed to perform well-defined atomic operations [14, 32]. In contrast, opcode-based ZK-EVMs rely on specific circuits that each prove the execution of an EVM opcode or an EVM state transition, with the specific goal of proving the validity of the execution of an EVM transaction. Figure 2 provides an overview of the compilation and proving process followed by different ZK-Rollups.

One example of an assembly-based ZK-VM is Starkware’s Cairo. Built as an abstraction on top of Algebraic Intermediate Representation (AIR), Cairo assembly generates polynomial constraints over a table of field elements that represent the state throughout the program’s execution. This table serves as the trace (or witness) for the STARK-based prover, which then proves whether the trace satisfies Cairo’s semantics [14]. While the Cairo framework allows one to generate application-specific circuits, in practice, it is used in Starknet to generate circuits for a single set of constraints for the von Neumann architecture-based Cairo CPU. The Cairo CPU is an AIR-generated STARK for a Cairo program that implements a register-based general-purpose VM.

Another example of an assembly-based implementation is RISCO’s RISC-V-based ZK-VM. This approach involves compiling a Rust program to RISC-V ISA, referred to as the Guest Program. The Guest Program is executed to produce an execution trace, corresponding to the intermediate states of the RISC-V VM throughout the execution. The trace is then used as a witness by the RISC-V prover, which provides proof that the execution follows RISC-V semantics. Unlike Cairo, which uses a novel ISA tailored for STARK, RISCO builds a prover for the existing RISC-V ISA. Using the LLVM compiler tool-chain, RISCO can execute and

## 6:10 Analyzing and Benchmarking ZK-Rollups

prove any language that can be compiled to RISC-V ISA [31]. Using that approach, you can pass an EVM implemented in Rust as the guest program and prove EVM transactions, as demonstrated by Zeth [32] (c.f. Section 3.3).

In contrast to assembly-based ZK-VMs, the most common approach is to directly target EVM bytecode. While the former approaches rely on an intermediate ZK-VM for circuit implementation, many mature ZK-Rollups have chosen to implement specialized circuits that directly prove the execution of EVM bytecode, or close to EVM bytecode. Currently, all major ZK-Rollups beyond Starknet utilize specialized circuit-based ZK-EVM implementations.

For instance, zkSync Era, Polygon’s zkEVM, and Scroll implement a ZK-EVM to prove the execution of transactions, but employ slightly different methods. Polygon and Scroll use Solidity’s compiler to allow existing programs written in Solidity to be deployed, executed, and proved on a ZK-Rollup by implementing specialized circuits that validate EVM traces. In contrast, zkSync Era takes a higher-level approach by compiling Solidity directly to ZK-EVM bytecode (using zksolc), which is then executed and proved using circuits designed to verify the ZK-EVM bytecode’s correctness instead of the EVM bytecode.

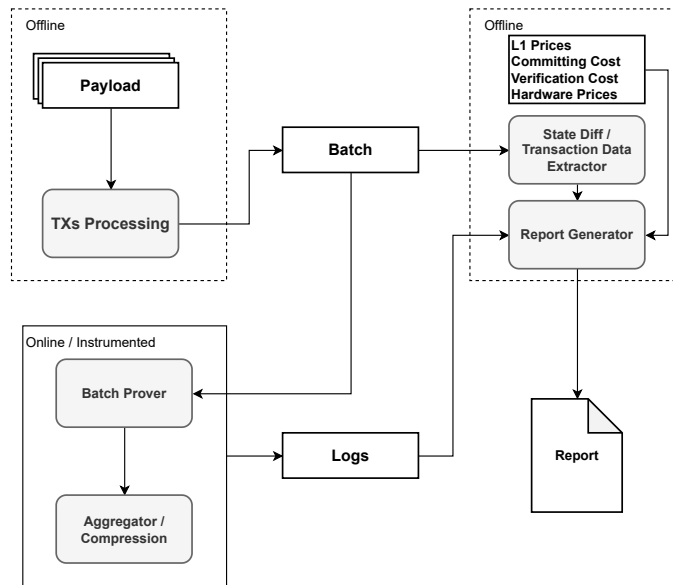
Another approach is Aztec’s Abstract Circuit Intermediate Representation (ACIR) and Private Execution Environment (PXE)-based system. Aztec’s smart contracts are written in Noir [25], a domain-specific language developed by Aztec Labs for SNARK-based proving systems. Noir compiles to ACIR, an intermediate representation used to generate circuits for proving. Due to additional features provided by Aztec, specifically the support for nullifier-based private transactions, Aztec’s execution model separates the handling of private data from the processing of public transactions. Transactions are first executed and proved locally by users inside the PXE, only propagating the public components of a transaction and a ZKP of the validity of its privately executed components to rollup nodes. This ZKP ensures the validity of private transactions, while the execution of their public components ensures the validity of the block.

### 3.5 Target Hardware and Prover Architecture

The hardware configurations required for ZK-VM providers vary significantly between projects, reflecting the diverse computational demands of their proof systems. Our analysis divides prover designs by target hardware and parallelization strategies into the following categories:

1. Single CPU-optimized implementations such as Polygon’s ZK-EVM, which demands a high-capacity setup with a 96-core CPU and at least 768 GB of RAM.
2. Single GPU and CPU proving, such as Scroll’s system, which parallelizes the execution of multiple blocks using a single GPU but then aggregates the proofs into a single proof that is posted on the chain using a CPU.
3. Cluster-based approaches: both zkSync and Risc0’s systems rely on two stages. At first, the state transitions are divided into segments and proved in parallel, after which the proofs are aggregated and also in parallel. The key difference with Scroll’s approach is that aggregation is also parallelized across a large cluster of GPUs or CPUs.

This highlights an essential consideration: Each ZK-VM implementation is meticulously optimized for specific hardware configurations, rendering direct performance comparisons on identical machines less meaningful. Furthermore, ongoing research on hardware acceleration for ZK-EVMs aims to further enhance proving times and system performance.



■ **Figure 3** Overview of the benchmarking procedure.

## 4 Experimental Evaluation

In this section, we delve into our methodology for benchmarking and analyzing the costs associated with ZK-Rollups. Benchmarking these systems is essential because it sheds light on the areas most in need of optimization. With many ZK-Rollups projects that aim to decentralize their core components, this analysis offers a timely opportunity to assess the costs involved and explore how these systems can achieve profitability and sustainability. Additionally, we consider projects interested in deploying specialized, application-specific rollups using existing infrastructure, aiming to discern the cost-related trade-offs of each stack.

### 4.1 Benchmarking and Analyzing ZK-Rollups

Our analysis is designed to simplify the understanding of how external factors influence the costs and, consequently, the fees associated with ZK-Rollups. We begin by identifying the primary challenges in benchmarking ZK-Rollups, particularly focusing on their core component: the ZK-EVM. Then, we present our methodology and the decisions made to navigate these challenges, and finally, we answer the targeted research questions (c.f. Section 1).

**Challenges and methodology.** In this subsection, we outline the inherent challenges in benchmarking ZK-Rollups and detail our methodology for addressing these challenges. The benchmarking process, depicted in Figure 3, provides a high-level overview of how we evaluate ZK-Rollups.

Initially, transactions from the selected payload are processed using the sequencer and any auxiliary tools provided by the ZK-Rollup to create a batch. This step can be performed on standard hardware without the need for specialized, expensive machines. Subsequently, this batch is fed into the prover pipeline within an instrumented environment designed to capture

## 6:12 Analyzing and Benchmarking ZK-Rollups

the necessary metrics. The proving phase generally consists of at least two steps, though this can vary. The final phase involves calculating the total costs, incorporating L1 data, batch posting and batch verification costs on L1, and hardware expenses. By integrating both the offline and the online analyses, we generate a comprehensive report that breaks down the costs.

**Metrics.** Determining appropriate metrics is a fundamental challenge. We focus on straightforward metrics that facilitate a basic comparison between different systems, making it easier for teams to integrate their ZK-EVMs into our benchmarking framework. In future work, we plan to expand our metrics to include power consumption, RAM usage, and GPU/CPU time. The primary metrics in this preliminary study are as follows:

- **Seconds per Proof:** The time required to generate a proof.
- **USD per Proof:** The cost of generating a proof, calculated by multiplying the clock time by the rate for using a cloud service like AWS or GCP.
- **USD per Proving a Transaction:** The cost associated with proving a single transaction.

**Configuration.** Choosing the right hardware configuration for benchmarking is another significant challenge. Ideally, different systems should be tested on identical or at least similar hardware specifications. However, this is nearly unfeasible for ZK-EVMs, as they are designed with distinct objectives in mind and optimized for vastly different hardware setups. Thus, we chose to benchmark the systems according to the hardware specifications recommended by each team for production use. This decision aims to capture the optimal cost-time efficiency based on each system's specific optimizations. Additionally, different proving systems have different requirements for resources (e.g., RAM) and different options for optimizations (e.g., through parallelization), meaning that it is even more difficult to compare to standardized machines.

**Payloads.** Deciding on the appropriate payloads for benchmarking is also challenging. Given the varying degrees of compatibility EVM between systems, selecting a common payload for comparative analysis is difficult. Moreover, the complexity of these systems further complicates setup and benchmarking efforts with specific payloads. Ideally, benchmarking would utilize historical Ethereum blockchain data, but the lack of necessary tooling in most ZK-Rollups complicates this approach. Instead, we focus on benchmarking common smart contract functionalities, including *native transfers*, *ERC-20 transfers*, *contract deployment*, *native Solidity/YUL hashing*, and the *Keccak precompile*.

These payloads represent typical blockchain operations, though future work may expand this list for more detailed analysis (e.g., token swaps, DAO voting, and NFT mints). Finally, note that we parameterized all payloads to different sizes, e.g., 1 ETH transfer vs. 10 ETH transfers vs. 100 ETH transfer, meaning that we have benchmarked 3 different batches of size 1, 10, and 100. Nevertheless, we recognize the necessity for additional research to develop more comprehensive workloads for benchmarking those systems.

**Reproducibility.** Ensuring reproducibility is crucial. Regardless of the metrics, configurations, and payloads, it is essential that the benchmarking process be designed so that third parties can validate the results. To accomplish this goal, we publish all configurations, scripts, and instructions used in our benchmarking process. This includes detailed specifications of the machines used, scripts for setting up the environment, and step-by-step instructions for running tests with the specified payloads. Our goal is for others to be able to replicate our findings by following our documentation, thus reinforcing the validity and reliability of our results while being able to easily extend our benchmarks.



■ **Table 2** Selected ZK-Rollups and Hardware Configurations. The Polygon ZK-EVM machine is hosted on AWS, while the zkSync Era is on GCP. Both machines have 1 TB SSD disk space. Note that spot prices could reduce the hourly cost significantly.

ZK-Rollup	Commit	Hardware Configuration	Hourly Cost
Polygon zkEVM	d37e826	r6i.metal (128 vCPUs, 1024GB RAM)	\$8.06
zkSync Era	4794286	g2-standard-32 (32 vCPUs, 1 NVIDIA L4 GPU, 128 GB RAM)	\$1.87

**L1 Data.** To obtain data from L1 for our analyses, we selected a period from April 10, 2024 (block 19621224), to May 10, 2024 (block 19835630). During this period, we crawled the main contracts of the selected ZK-Rollups and retrieved the required information: gas consumption for committing batches to L1 (excluding the gas cost related to Data Availability) and batch verification costs, i.e., the cost for verifying the proof and updating the smart contracts. This information is essential for comprehending fixed costs and is not connected to variable costs. Instead of our local payloads, we utilized on-chain data to determine the typical configurations employed by the rollups, including batch sizes.

**Threats to Validity.** Our study faces some validity threats that should be considered when interpreting the results. Firstly, our chosen payloads may not fully represent the range of real-world scenarios. We focused on providing easily executable payloads that are highly relevant, such as ETH and ERC-20 transfers, contract deployments, and SHA-256 hashes, commonly seen in other ZKP benchmarks [13]. However, this selection may overlook other important transaction types and interactions. Secondly, the results for certain components might be biased due to the specific payloads used. For instance, our payloads tend to favor state diffs as they primarily involve interactions with a single contract, which may not accurately reflect more complex and diverse batches. We have made efforts to clearly state any potential biases in the relevant sections of the paper. Finally, there are cases where parts of the systems are not fully open-sourced or could not be run in a controlled sandboxed environment. In those instances, we decided not to include our results for those systems in the analysis, as these results could not be independently verified by us. To ensure the quality and reliability of our findings, we narrowed our results to systems for which we could provide high-quality data.

## 4.2 Results

In this section, we present the results of our benchmarking and analysis of ZK-Rollups. We begin by detailing the systems we analyzed and the hardware configurations used for our tests. Following this, we dive into each research question, providing comprehensive answers based on our findings. Our goal is to offer information on the performance, costs, and trade-offs of different ZK-Rollup implementations.

**Selected ZK-Rollups.** For our analysis, we selected zkSync Era and Polygon ZK-EVM. The primary reason for choosing these two is that they are the only ZK-Rollups that are both EVM-compatible and fully open-sourced among the popular deployed ZK-Rollups. While Scroll is also EVM-compatible and widely used, we excluded it from our study because its GPU-prover is closed-source, which limits our ability to conduct a thorough and transparent analysis. By focusing on zkSync Era and Polygon ZK-EVM, we ensure that our benchmarking

## 6:14 Analyzing and Benchmarking ZK-Rollups

■ **Table 3** Fixed costs breakdown for ZK-Rollups using historical data. The USD cost of gas is based on 7.5 Gwei, and an Ether price of \$3,000. Compression costs are derived from Table 2. For Polygon ZK-EVM we also include the batch proof aggregation costs in proof compression. In zkSync Era, the execution of batches signifies L1 finality, thus fully finalizing the batch and allowing fund withdrawals from the system. The numbers are crawled from on-chain data from 10 April 2024 to 10 May 2024.

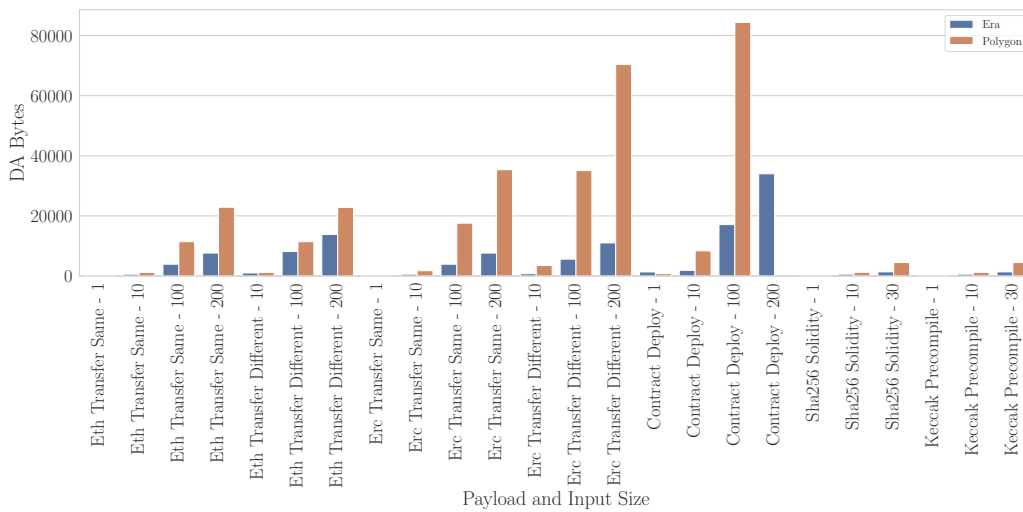
	Metric	Era	Polygon
Batch Size	Median Batch Size (Min/Max)	3,895 (485/5,000)	27 (1/158)
	Theoretical Max Batch Size	5,000	498
Commit Batches	Median Gas Cost (gas)	230,686	324,088
	Median Batches Per Tx (Min/Max)	1 (1/1)	8 (1/15)
Verify Batches	Median Gas Cost (gas)	458,527	378,461
	Median Batches Per Tx (Min/Max)	1 (1/1)	20 (1/160)
Execute Batches	Median Gas Cost (gas)	3,303,634	-
	Median Batches Per Tx (Min/Max)	26 (4/45)	-
Proof Compression	Median Time in Seconds (USD Cost)	1,075 sec (\$0.56)	311 sec (\$0.70)
	Median Batches per Proof	1	20
Normalized Costs	Median Gas Per Batch (USD)	816,275 gas (\$18.37)	59,434 gas (\$1.34)
	Median Batch Cost (USD)	\$18.93	\$1.38
	Median Gas Per Tx (USD)	209 gas (\$0.00471)	2,201 gas (\$0.04962)
	Median Tx Cost (USD)	\$0.00486	\$0.05111
	Median Gas Per Tx – Full Batches (USD)	163 gas (\$0.00367)	119 gas (\$0.00268)
	Median Cost Per Tx – Full Batches (USD)	\$0.00378	\$0.00275

is based on systems with relatively stable code bases and deployed on a large scale. Table 2 summarizes the specific versions (commits) of the ZK-Rollups we analyzed and the hardware configurations used for our experiments. Note that we selected those machines after advising the teams developing the analyzed systems.

**Benchmarking zkSync Era’s Prover.** We could only manage to run zkSync Era as a black-box system where multiple processes exchange messages, perform computations, and write logs. This is because Era is designed to run as a mini-cluster. Additionally, the instructions and configurations on how to set up this cluster are not publicly available. Since we were unable to run the prover components independently in a sandboxed environment and measure the performance ourselves, we decided not to include the prover time and cost in our analysis. Unfortunately, given the current state of the prover and its limited documentation, it was quite complicated to instrument the system for benchmarking as we intended. However, we were still able to reliably obtain proof compression time, and DA costs for Era. We leave as future work a fine-grained benchmarking of the Era prover.

### 4.3 Fixed Costs

The results presented in Table 3 highlight the fixed costs for zkSync Era and Polygon ZK-EVM based on historical data from 10 April 2024 to 10 May 2024. The fixed costs encompass gas costs for committing and verifying batches, and proof compression costs. Note that zkSync Era has an additional transaction for finalizing transactions and enabling withdrawals, which we also consider a fixed cost. The two solutions employ different approaches: zkSync



■ **Figure 4** Comparison of DA (Data Availability) requirements for various payloads between zkSync Era, which uses state diffs, and Polygon ZK-EVM, which posts transaction data.

Era supports significantly larger batches, while Polygon ZK-EVM utilizes smaller batches and employs aggregation to derive a final proof. Both systems convert a STARK proof into a SNARK, incurring a fixed proving cost that remains constant regardless of the input size.

For zkSync Era, the data show a median batch size of 3,895 transactions, resulting in a median batch cost of \$18.93 and a cost per transaction of approximately \$0.0047. The large batch sizes in zkSync Era allow for the costs to be distributed across a greater number of transactions, thereby reducing the cost per transaction. In contrast, Polygon ZK-EVM processes smaller batches, with a median batch size of 27 transactions. This leads to a median batch cost of \$1.38 and a cost per transaction of \$0.0511. Notably, in their ideal scenario where the batches are completely full, both systems can achieve negligible fixed costs per transaction (i.e., less than \$0.004).

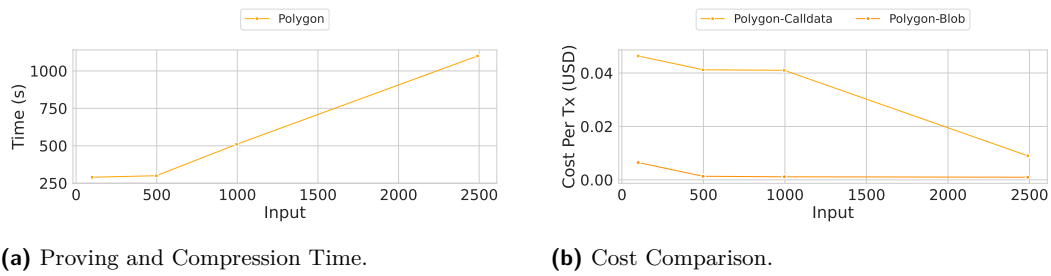
One critical insight derived from the analysis is that filling batches to their maximum capacity is essential to minimize fixed costs per transaction. This strategy allows the cost to be amortized over many transactions. However, it may negatively impact the finality if L2 does not have sufficient usage. Furthermore, even with enough usage, the marginal costs or proving time might increase, leading to slower finality. These trade-offs and their implications will be further examined in subsequent subsections.

#### 4.4 Marginal Costs

The incremental costs of ZK-Rollups are essential to grasp their scalability and effectiveness. This section explores the duration required to produce a proof for a batch, the related expenses, and the data availability (DA) expenses in bytes needed for submission to the L1. Furthermore, we evaluate the efficiency of state differences as opposed to uploading full transaction data.

The design of the provers for Polygon and zkSync Era shows marked differences. The prover for zkSync Era is designed to support the processing of large batches, supporting up to 5,000 transactions, and operates on more affordable hardware. On the other hand, the architecture of Polygon’s prover is geared towards quick-proof generation, albeit at the

## 6:16 Analyzing and Benchmarking ZK-Rollups



**Figure 5** Trade-off between proving/compression times (affecting finality) and cost per transaction for Polygon ZK-EVM.

cost of requiring pricier equipment. We noticed that the time it takes for zkSync Era to generate proofs extends with larger input sizes. In Figure 6 in Appendix B, we show some preliminary measurements that demonstrate this pattern. In contrast, Polygon’s prover maintains a consistent output time of either 190 or 200 seconds, independent of input size. While Polygon’s prover offers speed advantages for smaller batches, this comes at the trade-off of requiring expensive hardware. Nevertheless, with increasing batch sizes, Polygon’s prover becomes more cost-effective, as it takes the same time to prover different number of transactions. This demonstrates another difference in the design: Era’s prover is more elastic as smaller batches will be faster, and larger ones will be slower, whereas in Polygon batches proving will be either 190 or 200 seconds.

Figure 4 illustrates the data availability (DA) needs in bytes for both Polygon and zkSync Era. A primary distinction is zkSync Era’s use of state diffs, in contrast to Polygon’s approach of posting entire transaction data. The state diffs used by zkSync Era result in notably superior compression, making the DA requirements disparity more pronounced as the input size grows. The cost-effectiveness of zkSync Era’s state diffs is due to this enhanced compression. It is crucial to acknowledge that our payloads are naturally advantaged by state diffs since our transactions usually engage with a specific contract, enhancing compression potential. Although the actual difference may be smaller in practical scenarios, it remains considerable. Future investigations into this subject are recommended.

### 4.5 Trade-off Between Fast Finality and Cost Minimization

Figure 5 demonstrates the trade-off between fast finality and cost minimization in Polygon ZK-EVM, using benchmarks of 100, 498, 996, and 2490 ETH transfers. As depicted in Figure 5, Polygon’s prover time increases linearly. Note that, as mentioned before, Polygon’s prover always takes 190-200 seconds per batch. So, to prove 996 transactions, we are required to prove two batches and aggregate them. Further, the cost per transaction amortized better when using larger inputs, demonstrating a trade-off between fast proving (i.e., faster finality) or processing larger inputs and achieving better cost-effectiveness.

### 4.6 Cost Breakdown

Table 4 provides a detailed cost breakdown for ETH transfers across different batch sizes, showing the distribution of fixed and marginal costs. For small batch sizes, fixed costs such as committing, verifying, and proof compression dominate the total cost per transaction. As batch sizes increase, DA costs become more significant, particularly when blobs are used instead of calldata. In addition, for larger batch sizes, proving costs become increasingly relevant.

■ **Table 4** Cost breakdown for ETH transfers in Polygon ZK-EVM. The blob price per byte is 1 wei, the normal gas price is 7.5 Gwei, the ETH price is \$3,000, and we use the costs of the machines from Figure 3 for the provers. We also capture the cost per byte, i.e., for blobs, we do not charge for the whole blob if it is less than the blob size. The first percentage is when we consider blobs as DA, and the second is for calldata.

Size	Fixed Costs			Marginal Costs			Total Cost Per Tx (USD)	
	Commit	Verify	Proof Comp.	Proving	DA-Blob	DA-Calldata	w/ Blob	w/ Calldata
100	44%/36%	52%/42%	1%/1%	3%/2%	0%	20%	0.16	0.20
498	44%/20%	52%/23%	1%/1%	3%/1%	0%	55%	0.03	0.07
996	43%/13%	50%/15%	1%/0%	5%/2%	0%	70%	0.02	0.06
2,490	40%/6%	46%/7%	2%/0%	12%/2%	0%	84%	0.01	0.05

■ **Table 5** Impact of EIP-4844. The gas cost used is 50 Gwei (representing pre-Dencun upgrade prices), the ETH price is \$3,000, and the blob gas cost is 1 wei. Note that for blobs, typically, you have to pay for a full blob, but here, we compute the cost only for the required bytes.

Input	Size in Bytes		Calldata		Blobs	
	Era	Polygon	Era	Polygon	Era	Polygon
ERC-20 Transfers (200)	10,999	70,357	\$23.05	\$136.96	$3.3 \times 10^{-11}$	$2.11 \times 10^{-10}$
Contract Deployment (100)	17,087	84,369	\$35.97	\$182.43	$5.1 \times 10^{-11}$	$2.53 \times 10^{-10}$
ETH Transfer (2,490)	88,693	283,905	\$194.53	\$661.95	$2.66 \times 10^{-10}$	$8.52 \times 10^{-10}$

One interesting observation is that Polygon has optimized for fast proving time, leading to very fast and relatively cheap proving prices. In our preliminary investigation of Era, we noticed that orthogonal to Polygon, they have optimized more for data compression. In both systems, proof compression takes a fraction of the cost 1% and can be ignored, demonstrating that the use of such compression techniques is essentially free of non-trivial additional computational burdens.

In the context of blob pricing, proving time takes up merely 5% of the total cost incurred on Polygon. Furthermore, we observe that where blobs are used for DA, marginal costs of proving over the underlying batch size grow at a higher rate. When using calldata as DA, the situation changes: in this case, the marginal costs for calldata usage are non-trivial for Polygon. Nevertheless, the cost of proving remains at almost negligible levels (less than 2% of the cost of a full batch) at all batch sizes when using calldata DA. However, the vast majority of costs in Polygon (around 97% of the total cost for a batch of 2,490 transactions) stem from its DA requirements to store full transaction records.

## 4.7 Impact of EIP-4844

The introduction of EIP-4844 has significantly influenced the cost dynamics of ZK-Rollups,<sup>8</sup> particularly in terms of DA costs. As illustrated in Table 5, EIP-4844 has generally minimized DA costs by allowing for more efficient data posting. However, it is important to note that for blob transactions, rollups are required to pay for an entire blob of approximately 125 KB.

This implies that for rollups required to frequently submit small batches to L1, utilizing calldata could be more economically viable than using blobs. This underscores the importance

<sup>8</sup> EIP-4844 introduces a new kind of transaction type to Ethereum which accepts “blobs” of data to be persisted in the beacon node for a short period of time. These changes are forwards compatible with Ethereum’s scaling roadmap, and blobs are small enough to keep disk use manageable. You can read more about EIP-4844 at <https://www.eip4844.com/>.

of additional research into the existing constraints of blobs and how the market might evolve as demand grows. Moreover, the present low pricing of blobs has not fully realized market price discovery. It is important to consider that sustained low prices might establish impractical expectations, which could result in substantial price hikes as the market realigns.

More specifically, given a shift to blob-based pricing that significantly reduces DA costs for ZK-Rollups across the board, the recorded results allow for projecting on the relative costs of transaction processing at large batch sizes for each of the two assessed systems.

Given that Polygon has not yet implemented blob-aware logic, we expect that any marginal changes to its transaction cost model will occur after the new DA market matures. To this end, the ratio of the relative marginal costs of proving on Era and Polygon respectively goes from  $20 \times (= 39\%/2\%)$  to  $5 \times (= 52\%/12\%)$ , suffering a  $4 \times$  drop. This is indicative that, upon subsidizing DA costs through blobs, the large relative DA cost for transactions on Polygon is much less pronounced (relative to Era) than before. This can be expected as the Era prover is “already” more constrained by proving (given its more efficient use of state-diffs) and thus sees a lower marginal cost benefit from the DA repricing. This is also reflected in how marginal proving costs go from 39% to 52% (increase by a third) on Era due to blobs, while they almost triple (from 2% to 5%) on Polygon.

The above implies that Polygon could marginally benefit more by blob repricing, lowering its DA cost bottleneck, and moving in the direction of proving its main (marginal) cost. The latter regime seems to already be the case with Era where, although the new blob pricing system will provide benefits, they will not be affecting the substantial proving costs and thus will have lower relative impact. As DA costs continue to fall, we also expect that both (all) ZK-Rollups systems converge to respective “maximal” marginal proving costs (relative to DA), at which point DA will be a comparatively much smaller part of the overall cost structure and will not need to be further subsidized.

## 4.8 Lessons Learned

Below we revisit the original research questions from Section 1.

- **Trade-off between Fast Finality and Cost Efficiency (RQ1–RQ3):** Our experiments highlight a primary trade-off between achieving fast finality (i.e., rapid proving time) and maximizing cost efficiency through cost amortization. Filling batches fully allows for better cost amortization per transaction, impacting both fixed and marginal costs. While fixed costs become less significant for large batches, they dominate the costs for smaller batches, making efficient batch filling crucial for overall cost efficiency. This creates a dynamic where rollups perform better at higher utilization levels, somewhat paradoxically. This insight raises questions about designing an optimal metering mechanism for ZK-Rollups.
- **State Diffs Efficiency (RQ2):** Our preliminary benchmarking indicates that state diffs are highly cost-efficient in reducing data availability costs for ZK-Rollups. By publishing only the state changes instead of complete transaction data, state diffs offer significant compression benefits, especially as the input size increases.
- **Importance of Sequencing (RQ4):** The proving process benefits greatly from efficiently filling batches with transactions. It is essential to match the transactions to the capabilities of the underlying prover. For example, Polygon’s prover can handle a limited number of Keccak operations per proof. Proving a single Keccak costs the same as proving the maximum number of Keccak operations the prover can handle in a batch. Therefore, sequencing transactions to fit the prover’s optimal capabilities can significantly reduce the cost per transaction. This highlights the importance of developing sophisticated

sequencing strategies and suggests a need for further research into the topic to guide the development of efficient sequencing for ZK-Rollups.

- **Impact of EIP-4844 (RQ5):** The Dencun upgrade has substantially reduced DA costs, enabling near-zero cost transactions for ZK-Rollups. However, as blob prices may increase in the future, optimizing ZK-Rollups for cost efficiency remains critical. Additionally, strategies such as blob-space sharing or selectively using calldata instead of blobs during periods of high blob prices or for small batches could be viable options under certain circumstances and specific requirements.

## 5 Open RQs and Future Work

In this section, we present the open questions that remain unanswered in our current work and sketch a roadmap for future research. These questions not only underscore the complexities inherent in ZK-Rollups but also highlight areas that require further exploration.

- **Decentralization's Impact on Costs:** A pivotal question revolves around how decentralizing L2 core components will influence transaction processing costs. Specifically, we seek to understand whether the decentralization will lead to negligible cost implications or if it will significantly alter the economic landscape of ZK-Rollups. Another related open question but beyond the scope of this work is L2 MEV and cross-chain MEV.
- **Batch Size and Sequencing Optimization:** The size of transaction batches is a critical factor affecting proving costs. Future iterations of our work will delve into how variations in batch size and executed opcodes in a batch influence the cost per transaction and the overall proving time. While current systems may operate with an optimal batch size tailored to their specific needs, emerging forks (e.g., app-chains) may require adjustments to accommodate different priorities. This analysis aims to provide valuable insights for optimizing batch size and contents in response to evolving requirements.
- **Metering Mechanism Evaluation:** Another area of interest is the examination of the metering mechanisms employed by ZK-EVMs, which traditionally mirror those of the EVM [38]. Given that proving certain EVM opcodes might be relatively more costly than their execution, we plan to investigate potential discrepancies in pricing. Through micro-benchmarks, we will explore whether such mismatches pose significant challenges, such as the under-pricing of specific opcodes that could lead to DoS attacks.
- **Proving Market Mechanisms:** We also intend to explore various proving market mechanisms and assess how they might influence the cost dynamics of proof generation [37]. This exploration could shed light on potential economic models conducive to more efficient and cost-effective proving processes.
- **Throughput Limitations:** Identifying the maximum transactions per second (TPS) each rollup can achieve, based solely on proving and DA on Ethereum and excluding other market dynamics, is another critical inquiry. This analysis will help quantify the scalability limits of current ZK-Rollup implementations.

In addition to addressing these open questions, our future work will expand the scope of our benchmarks. We aim to conduct micro-benchmarks at the opcode level to gain a finer-grained understanding of proving costs. Moreover, we plan to introduce macro benchmarks with diverse payloads beyond those presented in this study. Furthermore, by replaying blocks of Ethereum on different ZK-Rollups, we aspire to provide deeper insight into their performance and cost-efficiency. We also plan to analyze ZK-VM-based ZK-Rollups and compare them with native ZK-Rollups to shed more light on the debate between specialized circuit implementations versus using generic VMs to produce ZKPs.

## 6 Related Work

**Benchmarking Blockchains and EVM Implementations.** Benchmarking blockchains has garnered significant attention from both academic and practitioner communities. Gramoli et al. [16] developed a comprehensive benchmark suite for six popular blockchains with smart contract capabilities, conducting an extensive evaluation using five realistic decentralized application payloads across various configurations for each blockchain. Their primary objective was to assess latency and throughput. In a subsequent study, Nasrulin et al. [24] introduced Gromit, a tool focusing on blockchains with various consensus algorithms, with both studies focusing on the overall performance of the blockchains under examination rather than specific aspects such as execution node implementations.

In this work, our focus shifts to benchmarking the core components of ZK-Rollups, paralleling efforts such as those of Cortes-Goicoechea et al. [11], who benchmarked the five most prominent Ethereum consensus clients to evaluate their resource consumption. Similarly, Zhang et al. [40] employed simple microbenchmarks to compare the efficiency of WASM EVM nodes against Geth and Openethereum. Our future endeavors include conducting micro-benchmarks to gain deeper insights into ZK-EVM implementations. Furthermore, both academic and industry efforts have explored benchmarking EVM nodes using either straightforward macro benchmarks [12], like ERC-20 transfers, or fuzzy techniques [26] to assess the performance characteristics of EVM implementations comprehensively. Mirroring this approach, we utilized macro benchmarks to evaluate the performance of ZK-EVM implementations, with plans to adapt tools such as flood [26] for future ZK-EVM benchmarking. Busse et al. [6] evaluate EVMs on various machines to pinpoint any noticeable differences.

While our current analysis does not extend to testing various machines for each ZK-Rollup, our objective is to perform such analyses to determine the most cost-effective hardware configurations for running each ZK-EVM and to identify the most optimized setup for each prover. Lastly, Perez, and Livshits [27] evaluated the EVM's metering mechanism, identifying potential DoS attack vulnerabilities. Inspired by their findings, we plan to conduct stress tests on ZK-EVM implementations to uncover any mispricing in the proving costs of specific EVM opcodes.

**Benchmarking ZKPs.** Benchmarking efforts for ZKPs play a crucial role in improving the understanding and performance of cryptographic libraries and primitives. Benarroch et al. [4] highlighted the inherent challenges and outlined best practices for implementing benchmarks for ZKP Domain Specific Languages (DSLs). Building upon this foundation, our work outlines the specific challenges of benchmarking ZK-Rollups, particularly focusing on ZK-EVMs, and proposes a comprehensive methodology to tackle these challenges. Ernstberger et al. [13] introduced zk-Bench, a detailed benchmarking framework and estimator tool designed for evaluating the performance of low-level public-key cryptography libraries and SNARK DSLs.

Our research complements these efforts by focusing on ZK-EVMs, which represent some of the most complex systems employing SNARKs. In addition to these efforts from academia, the Celer Network published a blog post<sup>9</sup> that benchmarks the time and memory costs of proving SHA 256 circuits across various ZKP tools. Parallel to our efforts, Delendum has developed a framework<sup>10</sup> dedicated to benchmarking ZK-VMs. An interesting future work could be to compare the performance of ZK-VM-based ZK-EVMs versus native solutions.

---

<sup>9</sup> <https://blog.celer.network/2023/07/14/the-pantheon-of-zero-knowledge-proof-development-frameworks/>

<sup>10</sup> <https://github.com/delendum-xyz/zk-benchmarking>



**Analyzing Rollups and ZK-Rollups.** Chaliasos et al. [9] provided a taxonomy of security issues that might occur in systems utilizing ZKPs including ZK-Rollups. Thibault et al. [36] have conducted an extensive survey on the use of rollups as a scalability solution for the Ethereum blockchain, discussing the various types, highlighting key implementations, and offering a qualitative comparison between optimistic rollups and ZK-Rollups. Koegl et al. [19] have compiled a comprehensive list of known attacks on rollup systems, shedding light on their potential impacts. In contrast, our work delves into ZK-Rollups, providing a detailed overview and explanation of their characteristics through a qualitative analysis. In addition, we focus on the empirical benchmarking of ZK-Rollups and the meticulous analysis of their associated costs. This dual approach not only enriches the understanding of ZK-Rollups as a scalability solution but also underscores the economic viability and technical challenges of implementing ZK-Rollups.

## 7 Conclusions

In this paper, we have undertaken a comprehensive analysis of ZK-Rollups, focusing on their scalability, efficiency, and economic implications. Our theoretical and empirical evaluations of Polygon ZK-EVM and zkSync Era reveal the inherent trade-offs in their design and implementation, providing critical insights into their operational costs and performance.

Addressing the challenges of benchmarking ZK-Rollups, we have developed a structured methodology that allows for a thorough evaluation of these systems. Our results highlight possible improvements and suggest directions for future research. This research not only improves the understanding of ZK-Rollups, but also aims to guide and influence the development of more efficient rollups.

---

## References

- 1 Elli Androulaki, Christian Cachin, Angelo De Caro, and Eleftherios Kokoris-Kogias. Channels: Horizontal scaling and confidentiality on permissioned blockchains. In Javier López, Jianying Zhou, and Miguel Soriano, editors, *Computer Security - 23rd European Symposium on Research in Computer Security, ESORICS 2018, Barcelona, Spain, September 3-7, 2018, Proceedings, Part I*, volume 11098 of *Lecture Notes in Computer Science*, pages 111–131. Springer, Springer, 2018. doi:10.1007/978-3-319-99073-6\_6.
- 2 Lukas Aumayr, Pedro Moreno-Sanchez, Aniket Kate, and Matteo Maffei. Blitz: Secure multi-hop payments without two-phase commits. In Michael D. Bailey and Rachel Greenstadt, editors, *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, pages 4043–4060. USENIX Association, 2021. URL: <https://www.usenix.org/conference/usenix-security21/presentation/aumayr>.
- 3 Barry Whitehat. Roll up token, 2018. Accessed: 2024-03-19. URL: [https://github.com/barryWhiteHat/roll\\_up\\_token](https://github.com/barryWhiteHat/roll_up_token).
- 4 Daniel Benarroch, Aurélien Nicolas, Justin Thaler, and Eran Tromer. ‘community proposal: A benchmarking framework for (zero-knowledge) proof systems. *QEDIT, Tel Aviv-Yafo, Israel, Tech. Rep*, 2020.
- 5 Same Blackshear, Andrey Chursin, George Danezis, Anastasios Kichidis, Lefteris Kokoris-Kogias, Xun Li, Mark Logan, Ashok Menon, Todd Nowacki, Alberto Sonnino, et al. Sui lutriss: A blockchain combining broadcast and consensus. *CoRR*, abs/2310.18042, 2023. doi:10.48550/arXiv.2310.18042.
- 6 Anselm Busse, Jacob Eberhardt, and Stefan Tai. Evm-perf: high-precision evm performance analysis. In *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–8. IEEE, 2021. doi:10.1109/ICBC51069.2021.9461058.

- 7 Vitalik Buterin. The different types of zk-evms. <https://vitalik.eth.limo/general/2022/08/04/zkevm.html>, 2022. Accessed: 2024-03-19.
- 8 Vitalik Buterin. *Proof of stake: The making of Ethereum and the philosophy of blockchains*. Seven Stories Press, 2022.
- 9 Stefanos Chaliasos, Jens Ernstberger, David Theodore, David Wong, Mohammad Jahanara, and Benjamin Livshits. Sok: What don't we know? understanding security vulnerabilities in snarks. *CoRR*, abs/2402.15293, 2024. doi:10.48550/arXiv.2402.15293.
- 10 Stefanos Chaliasos, Denis Firsov, and Benjamin Livshits. Towards a formal foundation for blockchain rollups. *CoRR*, abs/2406.16219, 2024. doi:10.48550/arXiv.2406.16219.
- 11 Mikel Cortes-Goicoechea, Luca Franceschini, and Leonardo Bautista-Gomez. Resource analysis of ethereum 2.0 clients. In *2021 3rd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*, pages 1–8. IEEE, 2021. doi:10.1109/BRAINS52497.2021.9569812.
- 12 Ziyad Edher. evm-bench. <https://github.com/ziyadedher/evm-bench>, 2024. Accessed: 2024-03-19.
- 13 Jens Ernstberger, Stefanos Chaliasos, George Kadianakis, Sebastian Steinhorst, Philipp Jovanovic, Arthur Gervais, Benjamin Livshits, and Michele Orrù. zk-bench: A toolset for comparative evaluation and performance benchmarking of snarks. *IACR Cryptol. ePrint Arch.*, page 1503, 2023. URL: <https://eprint.iacr.org/2023/1503>.
- 14 Lior Goldberg, Shahar Papini, and Michael Riabzev. Cairo—a turing-complete stark-friendly cpu architecture. *IACR Cryptol. ePrint Arch.*, page 1063, 2021. URL: <https://eprint.iacr.org/2021/1063>.
- 15 Jan Gorzny, Lin Po-An, and Martin Derka. Ideal properties of rollup escape hatches. In *Proceedings of the 3rd International Workshop on Distributed Infrastructure for the Common Good*, pages 7–12, 2022. doi:10.1145/3565383.3566107.
- 16 Vincent Gramoli, Rachid Guerraoui, Andrei Lebedev, Chris Natoli, and Gauthier Voron. Diablo: A benchmark suite for blockchains. In *Proceedings of the Eighteenth European Conference on Computer Systems*, pages 540–556, 2023. doi:10.1145/3552326.3567482.
- 17 Harry Kalodner, Steven Goldfeder, Xiaoqi Chen, S Matthew Weinberg, and Edward W Felten. Arbitrum: Scalable, private smart contracts. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1353–1370, 2018. URL: <https://www.usenix.org/conference/usenixsecurity18/presentation/kalodner>.
- 18 Lucianna Kiffer, Asad Salman, Dave Levin, Alan Mislove, and Cristina Nita-Rotaru. Under the hood of the ethereum gossip protocol. In *Financial Cryptography and Data Security: 25th International Conference, FC 2021, Virtual Event, March 1–5, 2021, Revised Selected Papers, Part II 25*, pages 437–456. Springer, 2021. doi:10.1007/978-3-662-64331-0\_23.
- 19 Adrian Koegl, Zeeshan Meghji, Donato Pellegrino, Jan Gorzny, and Martin Derka. Attacks on rollups. In *Proceedings of the 4th International Workshop on Distributed Infrastructure for the Common Good*, pages 25–30, 2023. doi:10.1145/3631310.3633493.
- 20 Giulio Malavolta, Pedro Moreno-Sanchez, Aniket Kate, Matteo Maffei, and Srivatsan Ravi. Concurrency and privacy with payment-channel networks. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pages 455–471, 2017. doi:10.1145/3133956.3134096.
- 21 MatterLabs. Boojum. <https://github.com/matter-labs/era-boojum>, 2022. Accessed: 2024-03-19.
- 22 Shashank Motepalli, Luciano Freitas, and Benjamin Livshits. Sok: Decentralized sequencers for rollups. *CoRR*, abs/2310.03616, 2023. doi:10.48550/arXiv.2310.03616.
- 23 Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Bitcoin.org*, 2008.
- 24 Bulat Nasrulin, Martijn De Vos, Georgy Ishmaev, and Johan Pouwelse. Gromit: Benchmarking the performance and scalability of blockchain systems. *CoRR*, abs/2208.11254:56–63, 2022. doi:10.48550/arXiv.2208.11254.
- 25 noir contributors. noir zksnark language, 2022. URL: <https://aztec.network/aztec-nr/>.

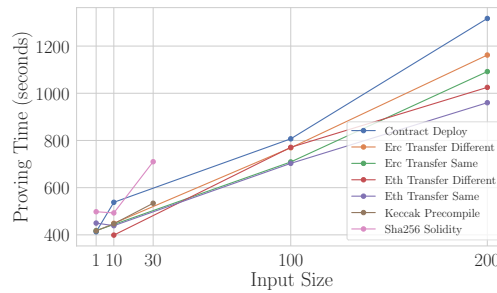
- 26 Paradigm. Flood. <https://www.paradigm.xyz/2023/06/flood>, 2023. Accessed: 2024-03-19.
- 27 Daniel Perez and Benjamin Livshits. Broken metre: Attacking resource metering in EVM. *CoRR*, abs/1909.07220, 2019. doi:10.48550/arXiv.1909.07220.
- 28 Polygon. Pil. <https://docs.polygon.technology/zkEVM/spec/pil/>, 2022. Accessed: 2024-03-19.
- 29 Polygon. zkasm. <https://docs.polygon.technology/zkEVM/spec/zkasm/>, 2022. Accessed: 2024-03-19.
- 30 Joseph Poon and Vitalik Buterin. Plasma: Scalable autonomous smart contracts. *White paper*, pages 1–47, 2017.
- 31 RISC-Zero. Risc zero vm. <https://github.com/risc0/risc0>, 2022. Accessed: 2024-03-19.
- 32 RISC-Zero. Zeth. <https://www.risczero.com/blog/zeth-release>, 2022. Accessed: 2024-03-19.
- 33 Scroll. halo2 kzg. <https://github.com/scroll-tech/halo2>, 2022. Accessed: 2024-03-19.
- 34 Cosimo Sguanci, Roberto Spatafora, and Andrea Mario Vergani. Layer 2 blockchain scaling: A survey. *CoRR*, abs/2107.10881, 2021. doi:10.48550/arXiv.2107.10881.
- 35 Erkan Tairi, Pedro Moreno-Sanchez, and Matteo Maffei. A 2 l: Anonymous atomic locks for scalability in payment channel hubs. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1834–1851. IEEE, 2021. doi:10.1109/SP40001.2021.00111.
- 36 Louis Tremblay Thibault, Tom Sarry, and Abdelhakim Senhaji Hafid. Blockchain scaling using rollups: A comprehensive survey. *IEEE Access*, 10:93039–93054, 2022. doi:10.1109/ACCESS.2022.3200051.
- 37 Wenhao Wang, Lulu Zhou, Aviv Yaish, Fan Zhang, Ben Fisch, and Benjamin Livshits. Mechanism design for zk-rollup prover markets. *CoRR*, abs/2404.06495, 2024. doi:10.48550/arXiv.2404.06495.
- 38 Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.
- 39 Anatoly Yakovenko. Solana: A new architecture for a high performance blockchain v0. 8.13. *Whitepaper*, 2018.
- 40 Yixuan Zhang, Shuyu Zheng, Haoyu Wang, Lei Wu, Gang Huang, and Xuanzhe Liu. VM matters: A comparison of WASM vms and evms in the performance of blockchain smart contracts. *ACM Trans. Model. Perform. Evaluation Comput. Syst.*, 9(2):5:1–5:24, 2024. doi:10.1145/3641103.
- 41 Qiheng Zhou, Huawei Huang, Zibin Zheng, and Jing Bian. Solutions to scalability of blockchain: A survey. *Ieee Access*, 8:16440–16455, 2020. doi:10.1109/ACCESS.2020.2967218.

## **A** EVM Compatibility Properties

The term “EVM compatibility” encompasses various properties, including:

- Support for the standard Solidity compilation toolchain, allowing existing Solidity contracts to be ported over without additional work.
- Compliance with Ethereum’s exact state transition logic.
- Adherence to Ethereum’s gas cost metering mechanism.
- Adherence to Ethereum’s JSON-RPC client API.
- Support for Ethereum’s smart contract standards (e.g., ERC-20) and precompiles (e.g., keccak).
- Support for Ethereum’s existing wallet infrastructure.
- Support for Ethereum’s existing development infrastructure.

## 6:24 Analyzing and Benchmarking ZK-Rollups



■ **Figure 6** Proving time increases depending on input size for the Era prover. Note that this result is not complete and does not count for potential parallelization.

### B Proving Time for Era

Figure 6 provides an overview of the increase of proving time in zkSync Era given different batch sizes of various inputs. This highlights a design orthogonal to Polygon ZK-EVM where, for each batch, the time needed to be proved is either 190 or 200 seconds. It is important to note that both systems optimize for various aspects. For example, zkSync optimizes for data compression through its state diffs mechanism, whereas Polygon optimizes for fast proving times. It is important to note that Figure 6 is not complete. Due to the complexity of the system, its modular architecture, and the lack of documentation, we did not manage to measure the complete proving time precisely. Finally, note that the setup for Era can be improved by: (1) using more machines to parallelize computation and reduce finality time, (2) using cheaper machines for non-GPU computations, e.g., witness generation, to reduce proving costs.

# DeFiAligner: Leveraging Symbolic Analysis and Large Language Models for Inconsistency Detection in Decentralized Finance

Rundong Gan ✉

School of Computer Science, University of Guelph, Canada

Liyi Zhou ✉🏠

The University of Sydney, Australia

UC Berkeley RDI, CA, USA

Decentralized Intelligence AG, Zug, Switzerland

Le Wang ✉🔗

School of Computer Science, University of Guelph, Canada

Kaihua Qin ✉🏠

Yale University, New Haven, CT, USA

UC Berkeley RDI, CA, USA

Decentralized Intelligence AG, Zug, Switzerland

Xiaodong Lin<sup>1</sup> ✉🏠🔗

School of Computer Science, University of Guelph, Canada

---

## Abstract

Decentralized Finance (DeFi) has witnessed a monumental surge, reaching 53.039 billion USD in total value locked. As this sector continues to expand, ensuring the reliability of DeFi smart contracts becomes increasingly crucial. While some users are adept at reading code or the compiled bytecode to understand smart contracts, many rely on documentation. Therefore, discrepancies between the documentation and the deployed code can pose significant risks, whether these discrepancies are due to errors or intentional fraud. To tackle these challenges, we developed *DeFiAligner*, an end-to-end system to identify inconsistencies between documentation and smart contracts. *DeFiAligner* incorporates a symbolic execution tool, SEVM, which explores execution paths of on-chain binary code, recording memory and stack states. It automatically generates symbolic expressions for token balance changes and branch conditions, which, along with related project documents, are processed by LLMs. Using structured prompts, the LLMs evaluate the alignment between the symbolic expressions and the documentation. Our tests across three distinct scenarios demonstrate *DeFiAligner*'s capability to automate inconsistency detection in DeFi, achieving recall rates of 92% and 90% on two public datasets respectively.

**2012 ACM Subject Classification** Security and privacy → Distributed systems security

**Keywords and phrases** Decentralized Finance Security, Large Language Models, Project Review, Symbolic Analysis, Smart Contracts

**Digital Object Identifier** 10.4230/LIPIcs.AFT.2024.7

**Supplementary Material** *Software (Source Code)*: <https://github.com/DeFiAligner/DeFiAligner> [28], archived at `swh:1:dir:ceb930c5854a5cae98402c9cd310fdb2c940ceeb`

**Funding** This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC).

---

<sup>1</sup> Corresponding author



## 1 Introduction

Decentralized Finance (DeFi) encompasses a wide range of financial applications and services built on blockchain platforms that support smart contracts, such as exchanges, lending and borrowing platforms, and derivatives [54]. These smart contracts are pieces of code that execute automatically when triggered by transactions. A distinct characteristic of DeFi is the transparency of the compiled bytecode for all smart contracts, often encapsulated by the maxim “*code is law*”. This principle underscores the ecosystem’s transparency, allowing anyone to deterministically and independently verify state transitions and validate the execution outcomes of these contracts and transactions.

Although the DeFi ecosystem offers sufficient transparency to allow anyone to verify and review the code of smart contracts, discrepancies between project documentation and the actual code can still pose risks to users. DeFi project documentation typically describes core functionalities and financial models, yet the actual on-chain code (e.g., EVM bytecode) may not faithfully implement these features due to programming errors or intentional design, and may even include unmentioned functionalities. A typical example is the Uranium Finance incident [5], where a discrepancy between the implementation of a conditional formula in the deployed code and its description in the whitepaper led to the theft of tokens valued at over \$50 million. Additionally, a broader example is that many ERC-20 tokens contain trading fees or blacklists [29] that are not disclosed in the documentation. Indeed, users with sufficient technical skills can directly read and understand the smart contract, thereby identifying potential risks and avoiding losses. However, when the source code of smart contracts is unavailable and only the binary code on the blockchain is accessible, understanding the compiled low-level bytecode becomes extremely challenging [32, 20]. For those who mainly rely on documentation or platform descriptions, discovering such inconsistencies is almost impossible. As far as we know, existing static analysis tools [16, 43] have not considered such inconsistency issues. Therefore, designing an inconsistency detection tool is crucial for protecting user assets and enhancing the trustworthiness of the DeFi system.

In this work, we take the first step in automatically detecting logical inconsistencies between DeFi project documentation and deployed smart contracts, aiming to assist in the automated review of DeFi projects. Although some studies [18, 29, 82, 33, 80, 45] have begun to discuss the issue of inconsistencies in DeFi, their scope of review remains confined to superficial checks at the function interface level, lacking scrutiny of the underlying business logic. Additionally, most of these studies heavily rely on manually derived or expert-summarized invariants, rendering them inherently resistant to automation. Furthermore, accessibility to open-source code repositories acts as a prerequisite for some methods [82, 33], constraining their applicability within closed or proprietary systems. Even for approaches [18, 45] that leverage transaction log analysis and do not require open-source code, they can only provide retrospective insights, failing to proactively prevent potential issues.

We propose a method to automatically detect inconsistencies by comparing the logic of smart contracts with the descriptions in the documentation. In our research, the examination of inconsistencies primarily focuses on changes in token balances and the conditions for these changes within smart contracts. We emphasize this focus because, in DeFi projects, balance changes are the most critical aspect of the logic, giving transactions their significance, as the primary purpose of most transactions is to alter balances. An inconsistency is identified if there is a mismatch between the two. For example, if the change in a user’s token balance does not align with the description in project documentation, it is identified as an inconsistency. However, implementing this approach is nontrivial due to two main challenges:

1) How to automatically generate symbolic representations of user balance changes and execution conditions. Firstly, DeFi code often involves inter-contract interactions, and existing tools like Mythril [7], Sailfish [12], and Manticore [51] lack support for computing dynamic jump addresses and cross-contract analysis, making it difficult to generate complete execution paths [70]. Secondly, some smart contracts are not open source or only partially open source, which complicates the analysis of locating data structures and reconstructing computational logic; 2) How to automatically compare symbolic expressions in the code with the calculation logic in the documentation. The documentation often features abstract business logic and personalized natural language expressions, which vary significantly. This variability complicates the direct alignment and comparison of the computational elements in the code with their descriptions in the documentation, making it difficult to verify consistency across these two mediums.

To tackle these challenges, we design an end-to-end system named *DeFiAligner*, which integrates traditional symbolic analysis with large language models (LLMs): ❶ Firstly, *DeFiAligner* relies on a symbolic tool called SEVM (*Symbolic Ethereum Virtual Machine*) to generate the execution paths of smart contracts. SEVM, an adaptation of the *Ethereum Virtual Machine (EVM)* [3], supports operations with Z3 symbolic values [24] in both stack and memory. It automatically generates corresponding Z3 symbolic variables as input based on Application Binary Interface (ABI) [2] information and executes stack and memory operations according to the opcode instructions of the smart contract. Unlike other symbolic tools, SEVM supports cross-contract analysis and saves the state of the stack and memory after each instruction is executed. ❷ Then, *DeFiAligner* identifies changes in token balances and execution conditions for each path by analyzing the state of the stack and memory after the execution of the SLOAD, SSTORE, and JUMPI instructions. ❸ Finally, to manage the complexity and variability of document information, *DeFiAligner* incorporates large language models [77], known for their proficiency in natural language processing and reasoning capabilities. Specifically, we input both symbolic data and documentation into the LLM's API, guiding it to detect inconsistencies through structured prompts. Unlike other works [34] that utilize LLMs for blockchain security analysis, our research uses symbolic expressions extracted from binary code as inputs, rather than directly using source code, ensuring that the input information is concise and crucial. Testing across three distinct scenarios shows that *DeFiAligner* can not only identify direct inconsistencies between textual and symbolic representations but can also uncover underlying logical discrepancies, thus significantly enhancing the quality and efficiency of DeFi project audits.

In summary, this work has three major contributions.

- To the best of our knowledge, this is the first work focused on detecting logical inconsistencies between documentation and deployed smart contracts for project review. We design an end-to-end system named *DeFiAligner* that identifies risks by examining on-chain binary code before traders interact with the protocol, rather than conducting post-event analysis after asset losses have occurred.
- We develop a symbolic generation tool named SEVM that produces accurate symbolic representations for cross-contract DeFi applications. This tool preserves the states of the stack and memory after each opcode instruction is executed, making it not only suitable for the task presented in this paper but also applicable to other symbolic analysis research related to smart contracts.
- We validate the practicality of our approach with empirical tests conducted across three real-world scenarios. These evaluations confirm our method's capability to expose discrepancies between the documented descriptions and deployed smart contracts. This verification not only proves the utility of our approach but also underscores its potential to enhance the reliability and transparency of DeFi applications.

The code for this work is publicly available on GitHub<sup>2</sup>.

## **2 Background**

### **2.1 Decentralized Finance and Smart Contracts**

Decentralized Finance (DeFi) utilizes blockchain technology [79] to enable peer-to-peer financial services, thereby eliminating the need for traditional intermediaries like banks. The core of DeFi is smart contracts [78], self-executing contracts with terms directly embedded in the code, which allow for the development and deployment of diverse financial protocols on platforms such as Ethereum [71]. Smart contracts are written in high-level programming languages like Solidity [22] or Vyper [15] and are compiled into lower-level bytecode that is executable on the blockchain. During the compilation of smart contracts, an Application Binary Interface (ABI) is generated that defines the methods and structures of the smart contract, enabling users to interact accurately with the contract's functions.

### **2.2 Symbolic Execution with Z3**

Symbolic execution [40, 11] is a method of software analysis that models potential execution paths of a program by treating its inputs as symbolic variables instead of using concrete values. This allows for an exhaustive exploration of the program's behavior under various conditions, helping to detect bugs, security flaws, and performance bottlenecks. The technique involves branching the program's execution at conditional statements, thereby creating a tree of possible execution paths. Each path is associated with a set of constraints on the input values that must be met for the path to be taken, allowing testers and developers to identify critical issues that could affect the program's reliability and security. In this research, we employ symbolic values within Z3 [24], a high-performance tool developed by Microsoft Research, to handle and manipulate the symbolic representations of program states. This integration facilitates more precise and powerful analysis, ensuring that all possible execution paths are thoroughly evaluated.

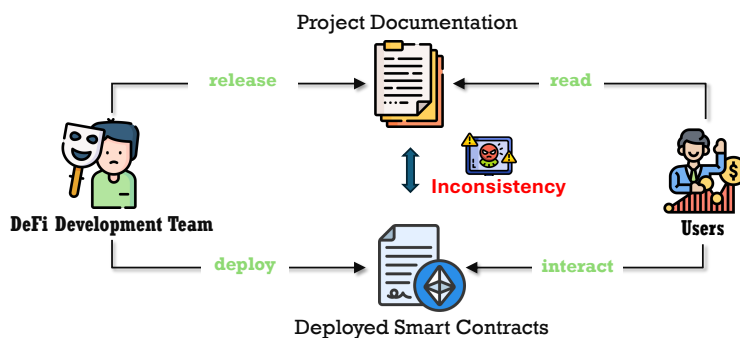
### **2.3 Large Language Models (LLMs)**

Large Language Models (LLMs) [17, 38], such as ChatGPT [8, 72, 74], are advanced artificial intelligence systems designed to understand, generate, and manipulate human language. These models are trained on vast datasets comprising diverse text sources, enabling them to grasp complex language patterns, context, and semantics effectively. Firstly, LLMs can automate the processing and analysis of large volumes of text, making them promising tools for data-driven decision-making and automation in various fields. Secondly, their ability to generate coherent and contextually relevant text makes them ideal for applications such as conversational agents, content creation, and semantic analysis. Moreover, the inferential reasoning abilities [69, 30, 56, 26] of LLMs set them apart, allowing them to not only process information but also generate insights and hypotheses based on the contextual understanding of the data they analyze. Nowadays, LLMs have been used in the field of software security testing and analysis [67, 55, 37]. Leveraging their proficiency in understanding both natural and programming languages, LLMs are increasingly used to enhance security protocols, detect vulnerabilities, and automate the analysis of code for potential security threats

---

<sup>2</sup> DeFiAligner, <https://github.com/DeFiAligner/DeFiAligner>





■ **Figure 1** Inconsistency: discrepancies between the project documentation and deployed smart contracts during a DeFi project cycle.

[46, 61, 60, 57, 76, 58, 41, 49]. To some extent, LLMs can help identify potential security issues without requiring security professionals to manually review thousands of lines of code, thus speeding up the security review process.

### 3 Preliminary

#### 3.1 Definition of Inconsistency

During a DeFi project cycle (as shown in Figure 1), the development team typically releases project documentation, and deploys smart contracts onto the blockchain. Many users tend to rely on the documents rather than inspecting the code directly. However, this overreliance on documentation can pose significant risks: the functions, logic, or operational conditions described in the documents may differ substantially from the code actually deployed on the blockchain. These discrepancies may arise from errors in the development process, delays in updating the documents, or intentional omissions of information. Users might not notice these discrepancies and make erroneous decisions, thereby facing the risk of financial losses. We define such inconsistencies, denoted by  $\Delta$ , as follows:

► **Definition 1** (Inconsistency  $\Delta$ ). Let  $D = \{d_1, d_2, \dots, d_n\}$  be a set of descriptions from the documentation, and let  $C = \{c_1, c_2, \dots, c_m\}$  be a set of observed behaviors in the bytecode of the deployed smart contracts. An inconsistency  $\Delta$  can be categorized into three types:

1. **Documentation-Only Inconsistency ( $\Delta_D$ ):** A description  $d_i \in D$  for which there is no corresponding behavior in  $C$ . For example, a promised transfer that does not appear in the code.

$$\Delta_D = \{d_i \in D \mid \nexists c_j \in C : d_i \text{ corresponds to } c_j\} \quad (1)$$

2. **Code-Only Inconsistency ( $\Delta_C$ ):** A behavior  $c_j \in C$  for which there is no corresponding description in  $D$ . For example, a trading fee that appears in the code but is not declared in the documentation.

$$\Delta_C = \{c_j \in C \mid \nexists d_i \in D : c_j \text{ corresponds to } d_i\} \quad (2)$$

3. **Mismatch Inconsistency ( $\Delta_M$ ):** Both a description  $d_i \in D$  and a behavior  $c_j \in C$  exist, but they do not match. For example, both the documentation and the code include calculations for rewards, but the formulas used to calculate the rewards are different.

$$\Delta_M = \{(d_i, c_j) \in D \times C \mid d_i \text{ and } c_j \text{ are related but do not match}\} \quad (3)$$

The collective set of inconsistencies  $\Delta$  is the union of these three types:

$$\Delta = \Delta_D \cup \Delta_C \cup \Delta_M \quad (4)$$

### 3.2 Threat Model

Following the definition of inconsistency  $\Delta$ , the primary objective of this research is to develop a methodology for detecting such inconsistencies between the documentation and the deployed smart contract in DeFi projects. For the threat model, we consider the following aspects:

- **Non-disclosure of Smart Contracts:** As described in previous research [59, 48], over 99% of Ethereum contracts have not published their source code. The lack of source code access complicates the verification of the contract’s security and functionality, as auditors and users are unable to directly verify the correctness and completeness of the contract logic by reading the source code.
- **Universal Accessibility of On-chain Binary Code:** Regardless of the public availability of smart contract source code, the binary code deployed on the blockchain is always accessible. This availability provides the possibility for contract verification but also necessitates specific technologies to analyze and understand the actual behavior of these codes.
- **Limitations of LLMs in Understanding Binary Code:** While LLMs are powerful tools for processing and analyzing text, their capability to understand and interpret binary code directly is limited. This poses a significant challenge in scenarios where only binary code is available, requiring additional tools or methods to bridge the gap between LLM capabilities and the need for detailed binary code analysis.

Our approach aims to analyze and infer smart contract behaviors under conditions of limited information by combining symbolic execution and LLMs, thereby identifying discrepancies between the code and documentation.

## 4 Motivation Example

The following example illustrates a real counterfeit token, named UNISWAP2.0<sup>3</sup>, which copies the documentation of Uniswap tokens [6]. UNISWAP2.0 is a scam project, where the developer uses the name of Uniswap to attract traders and embed malicious logic in the counterfeit token. Specifically, the contract developer deployed this scam token on Ethereum and subsequently created a liquidity pool on the Uniswap exchange [9] by depositing the scam token and WETH token. **Listing 1** shows the code snippet of UNISWAP2.0. In the code, the `automatedMarketMakerPairs` array is used to verify interactions with the pool’s address. This setup results in traders paying transaction fees when buying or selling tokens through the pool. Additionally, the contract owner can manipulate a blacklist to prevent specific users from transacting, furthering their malicious agenda. Please note that for illustrative purposes, we present the source code here; however, we do not actually use the source code in our entire detection process.

<sup>3</sup> A counterfeit token, <https://etherscan.io/token/0xC54F5c53Ab4a3A56303f96543245c13d58a3433d#code>

■ **Listing 1** The code snippet from a counterfeit token named UNISWAP2.0.

```

1 function _transfer( address from, address to, uint256 amount) internal override {
2     require(from != address(0), "ERC20: transfer from the zero address");
3     require(to != address(0), "ERC20: transfer to the zero address");
4     // Author's Note 1: The official project documentation does not describe a
5         blacklist, but it is present in this counterfeit token.
6     require(!blocked[from], "Sniper blocked");
7     .....
8     // Author's Note 2: The official project documentation does not describe any
9         fees, but it is present in this counterfeit token.
10    # only take fees on buys/sells, do not take on wallet transfers
11    if (takeFee) {
12        # on sell
13        if (automatedMarketMakerPairs[to] && sellTotalFees > 0) {
14            fees = amount.mul(sellTotalFees).div(100);
15            .....
16        }
17        # on buy
18        else if (automatedMarketMakerPairs[from] && buyTotalFees > 0) {
19            fees = amount.mul(buyTotalFees).div(100);
20            .....
21        }
22        if (fees > 0) {
23            super._transfer(from, address(this), fees);
24        }
25        amount -= fees;
26    }
27    super._transfer(from, to, amount);
28 }

```

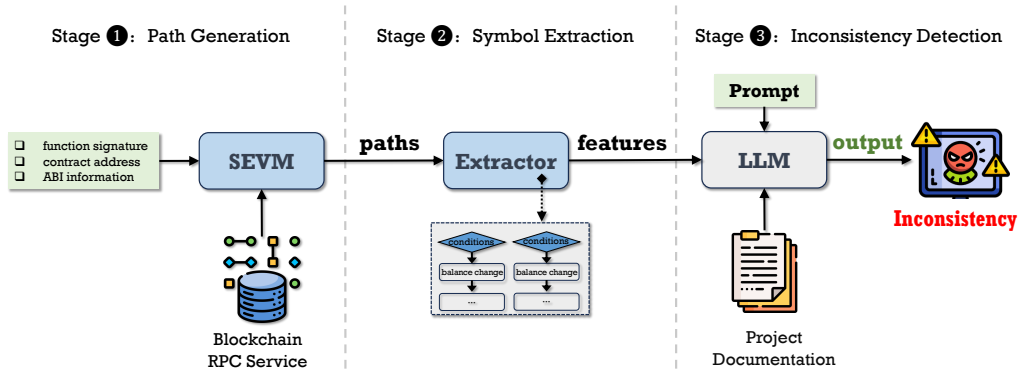
In this case, significant inconsistencies arise between the documentation of Uniswap token and the scam contract. The documentation does not mention any mechanisms like blacklists or trading fees. However, in the scam token's code, there are at least two inconsistencies: the existence of a blacklist and the imposition of transaction fees. These deviations are not documented and could mislead users into making incorrect decisions, potentially leading to financial losses.

Previous analyses and inspections of smart contracts [29, 47, 81, 44] have primarily focused on checking specific code patterns, such as fee collection or blacklist enforcement. However, they inherently cannot determine whether these features represent malicious intentions or are merely unique implementations of normal business logic. Compared to previous research, *DeFiAligner* utilizes symbolic analysis and large language models to conduct cross-field comparisons between text and code, thus breaking through traditional limitations and enhancing the ability to understand and detect inconsistencies between the deployed smart contracts and their documented descriptions.

## 5 Methodology

### 5.1 Overview

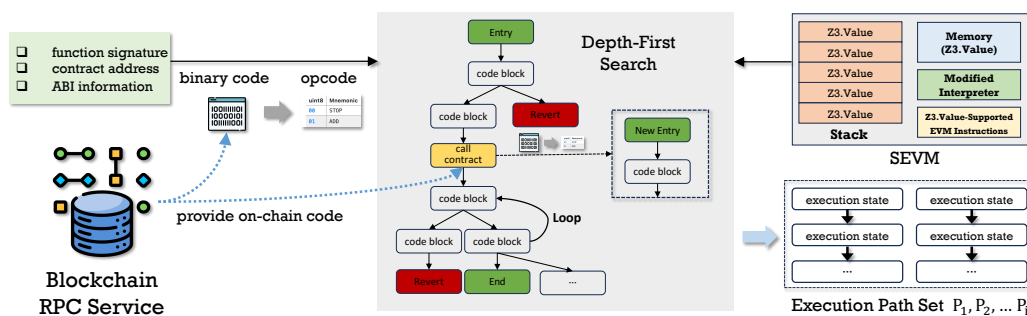
The high-level logic of *DeFiAligner* operates as follows (as shown in Figure 2). Firstly, the user specifies the function signature, contract address, and ABI information, all of which are publicly available and easy to obtain. Then, SEVM interacts with the corresponding blockchain Remote Procedure Call (RPC) [10] to retrieve the bytecode of the deployed smart contract. Subsequently, SEVM constructs symbolic variables as inputs and symbolically executes instructions in memory and stack according to the logic in the bytecode, thereby generating all possible execution paths. These paths include the states of memory and stack after the execution of each opcode instruction. Then, *DeFiAligner* extracts symbolic expressions for token balance changes and their conditions from these paths by analyzing the



■ **Figure 2** Overview of *DeFiAligner*.

states of the stack and memory before and after the execution of specific instructions. Finally, the symbolic features, along with the documentation, are fed into the large language model. By using structured prompts, the large language model automatically detects potential inconsistencies. In more detail, our system consists of the following stages:

- **Stage 1: Path Generation.** Although there are many symbolic generation and analysis tools [7, 12, 51, 14], their functionality is limited: 1) the analysis is purely static and lacks support for dynamic jump addresses and cross-contract analysis. For example, they cannot generate incomplete execution paths when there are interactions between multiple contracts; 2) some tools are unable to restore symbolic logic from the binary code; 3) although some tools (e.g., Vandal [14]) can convert low-level bytecode into semantic logic relations, they do not support bitwise operations on memory data when executing some memory-related instructions, resulting in errors. Therefore, we developed a tool called SEVM to generate symbolic execution paths, which overcomes the aforementioned limitations. Specifically, we first modify the basic data types of memory and stack in the native EVM by changing the uint256 type elements in the stack to the bit-vector (BV) type in Z3 Value [4], and transforming the memory into a customizable length type of BV values. To adapt to changes in basic data, we also modify the EVM instructions to support calculations with Z3 Value (e.g., ADD, SUB, MUL). Additionally, we modify the EVM interpreter to support computations with Z3 Value and to utilize the Depth-First Search algorithm [62] to explore all possible execution branches. To address the issue of dynamic jump addresses and cross-contract calls, SEVM dynamically retrieves contract addresses and codes from the RPC service when executing the CALL, STATICCALL, and DELEGATECALL instructions and enters the function specified by these instructions for subsequent SEVM computations. When executing each opcode instruction, SEVM records all symbol information on each path, including : 1) executed opcode instructions and 2) the states of the memory and stack after each instruction execution.
- **Stage 2: Symbol Extraction.** In traditional symbolic execution, the “path explosion problem” is prevalent, where the number of paths for analysis multiplies rapidly, especially with multiple IF instructions. *DeFiAligner* addresses this issue by applying domain-specific knowledge in DeFi, focusing on changes in asset balances. Specifically, *DeFiAligner* selectively filters out paths that do not impact token balances by checking the overlap of SSTORE and SLOAD instructions in the symbolic representation. This approach allows our system to concentrate on the most relevant paths, thereby increasing its efficiency. *DeFiAligner* focuses on extracting two critical DeFi features from each path: 1) asset



■ **Figure 3** The process of generating execution paths by SEVM.

balance changes: In DeFi, changes in asset balances directly reflect the economic impact of a smart contract’s actions. Focusing on these changes offers a straightforward method to assess the contract’s behavior, translating complex code into tangible financial outcomes. Unusual or unexpected balance changes may indicate vulnerabilities, bugs, or exploits within the contract. 2) conditions of JUMPI: In EVM bytecode, JUMPI is a conditional jump instruction that plays a crucial role in controlling the execution flow. By analyzing the execution conditions of JUMPI instructions, *DeFiAligner* can understand the decision-making process within the contract and identify and compare the logical structures.

- **Stage ③: Inconsistency Detection.** Although we have extracted symbolic representations of balance changes and conditions in the earlier stages, directly comparing these symbols with the rich and varied textual information is extremely challenging. Fortunately, large language models excel at processing such complex tasks. Thus, we delegate this intricate comparison to the LLM. We predefine the representation rules for symbols to the LLM, and then input both the symbolic features and the textual information into the model. Subsequently, we pose explicit instructions to the LLM to detect potential discrepancies.

In the following sections, we will present the design details for each stage.

## 5.2 Path Generation

### 5.2.1 Generation Process

Figure 3 shows the process of using SEVM to generate execution paths. First, SEVM obtains the corresponding binary code by providing the user-specified contract address to an RPC service and then converts it into opcode instructions. By analyzing the ABI information, SEVM converts the parameters of the specified function into variables of Z3 Value, and then begins execution from the specified function. When there are multiple execution branches following the IF instruction, it employs the principle of Depth-First Search (DFS) [62] to traverse each branch. Upon encountering instructions such as CALL, STATICCALL, or DELEGATECALL, SEVM analyzes the state of the stack to determine the called contract address and function. Then, SEVM retrieves the binary code from an RPC service again, converts it into opcode instructions, and enters the invoked function to proceed with the next level of execution. Once the call is completed, the returned parameters are stored in the original stack or memory, and the program continues to execute. To avoid the path explosion issue caused by loops, SEVM also limits the number of loop iterations. Finally, the output of SEVM is a series of sequences, each composed of opcode instructions and the corresponding stack and memory states after each opcode is executed. The following are the core components of Path Generation:

### 5.2.2 Modifications to EVM

As is well known, the native EVM does not support the computation of Z3 Value. Therefore, we have made significant modifications to the EVM:

- **Modifications to Stack and Memory.** Listing 2 and Listing 3 showcase the modifications to the fundamental data structures of memory and stack. By integrating Z3 Value directly into the stack and memory structures and modifying the corresponding stack and memory operations (such as `stack push /pop` and `memory copy`), the SEVM is enabled to support symbolic computation. Additionally, `Z3.BV` supports bitwise operations [1], making the manipulation of memory more flexible.

■ **Listing 2** The structure of modified stack.

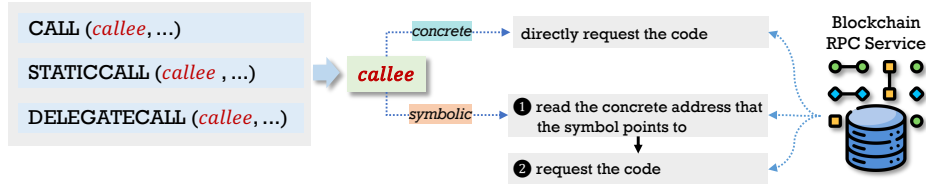
```
1 type SymbolicStack struct {
2     data []z3.Value
3 }
```

■ **Listing 3** The structure of modified memory.

```
1 func NewSymbolicMemory(ctx *z3.Context) *SymbolicMemory {
2     return &SymbolicMemory{
3         Store: ctx.FromInt(0, ctx.BVSort(MEMORY_BV_SIZE)).(z3.BV),
4     }
5 }
```

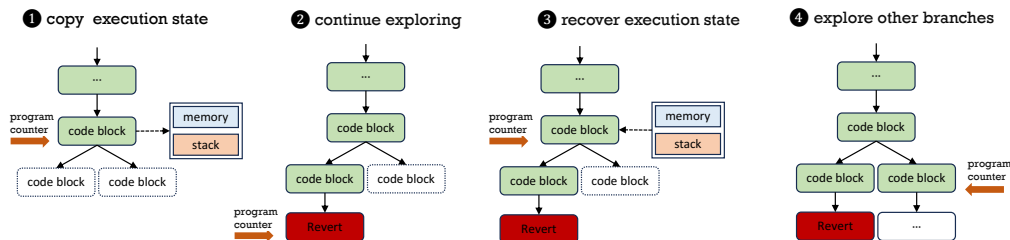
- **Modifications to EVM Instructions.** To support arbitrary operations for Z3 symbolic variables on stack and memory, we have also modified the EVM Instructions. Specifically, the following categories of instructions have been modified:
  1. *Simple computational instructions.* These instructions are usually simple, merely reading and storing data from memory or the stack, such as `ADD`, `MOD`, and `SHL`, etc. Since Z3 supports these operations very well, we only need to change the calculation of variables within these instructions to the calculation of Z3.
  2. *Complex computational instructions.* Z3 cannot support some of the complex computations in the EVM. For example, the `KECCAK256` instruction extracts a bit-vector data from memory and computes its Keccak-256 (or SHA-3) hash. However, Z3 does not support the cryptographic operation like Keccak-256, so we define a new symbolic variable and name it `SHA3[data.String()]`, which will be involved in subsequent computations.
  3. *Instructions for reading and writing the blockchain storage.* Some instructions, such as `SLOAD`, `SSTORE`, and `TIMESTAMP`, will read or store data from the block. In SEVM, we try to represent operations symbolically rather than actually manipulating data on the blockchain. When some instructions need to modify on-chain storage, we only perform some preliminary operations (e.g., `stack push /pop` and `memory copy`). When it is necessary to read on-chain data, we introduce new Z3 variables using special symbols. For example, “`SLOAD [ scope.Contract.self.String() => location.String() ]`” represents reading the storage from the `location` in the current contract, and “`Block Time`” represents the current block time.
  4. *Instructions for calling external contracts.* In the native EVM, three opcode instructions are related to contract calls, namely `CALL`, `STATICCALL`, and `DELEGATECALL` [19]. In these instructions, there is a parameter `callee` loaded from the stack, which points to the address of the contract to be called. Due to the previous modifications, `callee` could be a concrete value or a symbolic value. As shown in Figure 4, when executing

the related instructions, SEVM will analyze the type of `callee`. If it is a concrete value, SEVM directly requests its code via RPC services; otherwise, it dynamically loads the address from the block based on the symbolic description of `callee` and then requests its code. Dynamic analysis is necessary in this process because many contracts use a variable to store the contract address instead of embedding it within the contract.



■ **Figure 4** SEVM dynamically loads binary code when calling other smart contracts.

- **Modifications to EVM Interpreter.** During program execution, the JUMPI instruction checks the condition's truthfulness to determine the position of the next instruction. The execution of the native EVM is dynamic, following only one path. However, SEVM's execution is static, with conditions potentially being symbolic values, which may result in multiple branches. Therefore, we use Depth-First Search to explore all branches (as shown in Figure 5): when the program counter reaches the JUMP instruction and there are multiple branches, SEVM chooses one branch, and the current stack and memory state are saved. Once the exploration of this branch is complete, the state is rolled back, and the remaining branches continue to be explored. To avoid loops, SEVM checks the program counter and stack state; if the current code has already been accessed and there is a duplicate stack state, it skips further access. After executing each instruction, SEVM records the stack and memory state at that moment to facilitate subsequent analysis.

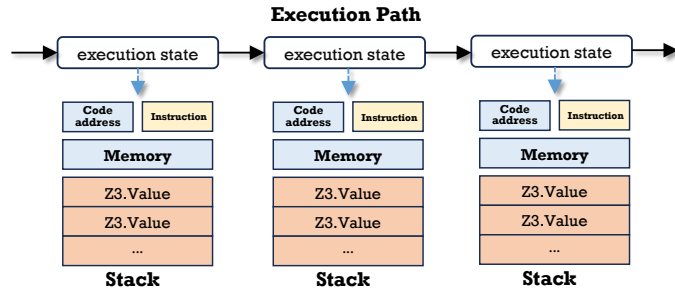


■ **Figure 5** SEVM explores branches using Depth-First-Search.

Additionally, we removed the code related to gas calculation because it is unnecessary for the static analysis in this study. Due to space limitations, we have only introduced the core modifications. For more details, please refer to the project code link in the Introduction section.

### 5.2.3 The Output of SEVM

Figure 6 shows an execution path generated by the SEVM. This path consists of multiple execution states. Each execution state records the current code address (the position of the current instruction in the binary code), opcode instruction, and the states of memory and stack after executing the current instruction. Typically, a smart contract has multiple paths, therefore the output of the SEVM is a set of paths, denoted as Path Set  $P = \{P_1, P_2, \dots, P_i\}$ .

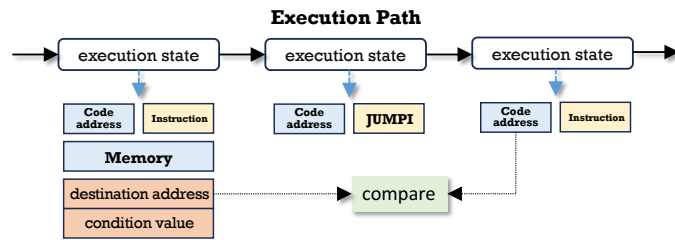


■ Figure 6 An execution path output by the SEVM.

### 5.3 Symbol Extraction

In this subsection, we will discuss how to identify the symbolic features by analyzing the execution paths.

#### 5.3.1 Symbol of Condition



■ Figure 7 Determine the condition symbol through execution states and JUMPI.

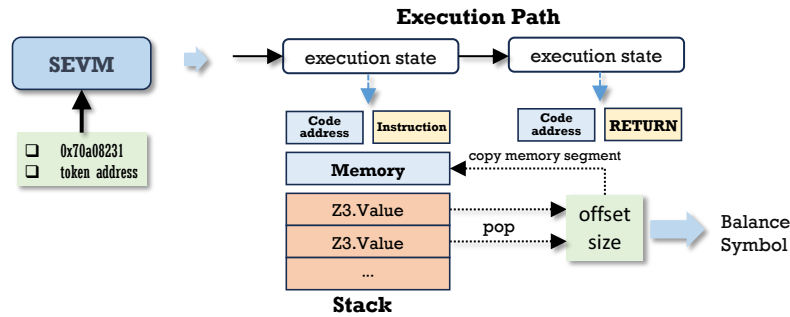
We know that the JUMPI instruction in the EVM is a conditional jump operation. It functions by taking two values from the stack: the first is the destination address of next instruction, and the second is a condition value. If the condition value is non-zero (true), the JUMPI instruction causes the program to jump to the specified destination address and continue execution from there. If the condition value is zero (false), the execution proceeds to the next sequential instruction instead. Therefore, we use the following steps to obtain the symbol of the conditions for each path:

1. Check the current instruction. If the instruction is JUMPI, proceed to the next step;
2. Check the second element (condition value) from the stack in the previous execution state. If the condition value is a symbolic, proceed to the next step;
3. Compare the top element (destination address, a concrete value) of the stack in the previous execution state with the code address of the next execution state (as shown in Figure 7). If they are numerically equal, then “condition value=TRUE” is the necessary condition; otherwise, “condition value=FALSE” is the necessary condition;
4. Add the above condition to the list.

#### 5.3.2 Symbol of Balance Change

**The balance changes of native tokens.** In the EVM, ETH transfers are primarily accomplished through the CALL instruction. SEVM determines the transfer of ETH by checking the relevant parameters of CALL. For instance, if the value parameter of CALL (the third





■ **Figure 8** Determine the balance symbol through execution states and RETURN.

value on the stack) is non-zero, this indicates that ETH is being sent concurrently with the function call. The sender of the ETH is the caller, and the receiving address is the contract being called.

**The balance changes of non-native tokens.** For DeFi protocols, changes in token balances are the most important feature. However, automatically locating the data structures is complex because balance variables can have different data structures [18, 35, 36] in the blockchain storage, and other types of variables may have similar structures. Previous methods are complex and labor-intensive, but we notice a fact that can help us locate data structures more easily: all ERC20 or ERC721 tokens have the `balanceOf(address)` method, which returns the balance of a specified address. Therefore, we use SEVM to analyze the `balanceOf(address)` method to obtain the corresponding symbolic expression. Specifically:

- **The symbol of the balance.** We set the entry function to `balanceOf(address)` (the function signature is `0x70a08231`), and input a Z3 symbol named `User_Address` as the parameter. Then SEVM calls this function at the specified address to obtain the execution path. Next, we examine the final `RETURN` instruction in the execution path. In Solidity, the `RETURN` instruction is used to exit a function and return data to the caller. When a function finishes executing, the `RETURN` opcode specifies the memory location and size of the data to be returned. Specifically, when the `RETURN` opcode is executed, the stack will pop off the `offset` and `size` values (usually concrete values), and then the data in memory from `offset` to `offset+size` will be retrieved and returned. Therefore, by analyzing the stack and memory in the execution path (as shown in Figure 8), we can get the symbolic representation of the balance. For example, **Listing 4** and **Listing 5** respectively show the source code of `balanceOf(address)` and the balance symbol of UNISWAP2.0. In the balance symbol, `SLOAD` represents loading data from contract `0xc54f5c53ab4a3a56303f96543245c13d58a3433d` at the location `SHA3 [Concat [User_Address Identity]]`. `Concat [User_Address Identity]` represents concatenating the user address with contract's unique identifier.

- **Listing 4** The source code of `balanceOf(address)` in UNISWAP2.0.

```

1 function balanceOf(address account) public view virtual override returns (uint256){
2     return _balances[account];
3 }

```

## 7:14 Symbolic & LLM-Based Inconsistency Detection

■ **Listing 5** The balance symbol of UNISWAP2.0.

```
1 SLOAD [  
2     0xc54f5c53ab4a3a56303f96543245c13d58a3433d => SHA3 [ Concat [User_Address  
3     Identity]  
4 ]
```

■ **The symbol of the balance change.** Now, we have known the symbol of the balance. To check for balance changes, we only need to find modifications to the balance symbol. Specifically, in the EVM, the `SSTORE` opcode is responsible for modifying storage on the blockchain. `SSTORE` pops two values from the stack, the first being the storage slot address (or location) and the second being the data to be stored. When `SSTORE` changes the blockchain storage, we check the contract address and location through string matching to determine if it is a balance change. If the contract address is the same and only the user address has changed (e.g., in `SHA3 [Concat [User_Address Identity]]`, `User_Address` changes to another address), it is considered a balance change. For example, **Listing 6** shows the location and symbolic data of one `SSTORE` instruction for UNISWAP2.0. There is a token transfer (`AmountOut` is the amount transferred out), because, compared to the balance symbol in **Listing 5**, it occurs in the same contract and only replaces the user address with another address. Note, `bvmul` represents multiplication, and `bvadd` represents addition.

■ **Listing 6** The location and stored data of one `SSTORE` instruction.

```
1 Location: SHA3 [ Concat [Address Identity]  
2  
3 Data:  
4 (bvadd SLOAD[0xc54f5c53ab4a3a56303f96543245c13d58a3433d SHA3  
5     [ Concat [Address Identity]  
6     (bvmul -1 AmountOut))
```

Additionally, our system can automatically detect all token contract addresses called in each execution path and identify the relevant symbolic features without requiring users to specify them.

### 5.3.3 Filter out Invalid Paths

In the process of analyzing, it is crucial to focus on relevant execution paths to optimize both the accuracy and efficiency. To this end, we implement a filtering mechanism to eliminate paths that are unlikely to contribute valuable insights. Specifically, we exclude the following types of paths:

1. **Failed paths with the REVERT instructions:** Such paths are often triggered by conditions that prevent transactions from completing successfully, such as failed assertions or checks. Since these paths represent execution flows that are explicitly handled to prevent erroneous state changes, they are typically not useful for further analysis.
2. **Paths with no balance changes:** Paths that do not involve any changes in the balance of the participating accounts are filtered out. These paths are considered less significant as they do not impact the financial state of the contract or the accounts involved.

By excluding these paths, we can reduce the clutter of non-consequential data, allowing for a more focused investigation of financially impactful behaviors.

## 5.4 Inconsistency Detection

In this section, we describe input and prompt schemes to LLMs.

### 5.4.1 Input to LLMs

In this paper, the following will be fed into the LLMs:

- **Project Documentation.** This can be any documents that define the specifications, functionalities, and intended behaviors of the project being analyzed. It includes white papers, user guides, and any other relevant materials that provide authoritative insights into how the smart contract should operate.
- **Symbolic Features.** These features primarily include the symbols representing balance changes and branch conditions along each execution path.
- **Definition of Inconsistency.** In this paper, the newly defined concept of inconsistency may present ambiguities for LLMs. Therefore, we input the definition of inconsistency into the LLMs to ensure they understand our intent.
- **Definition of Symbols.** Z3 built-in symbols and our custom symbols.

### 5.4.2 Prompt Template

We start by setting the following system prompt for LLMs. Figure 9 defines the prompt template set up for LLMs, specifically designed to guide the LLM in conducting automated review of DeFi projects. The prompt includes two main parts: First, it provides the LLM with knowledge about DeFi protocols, smart contracts, and symbolic analysis, and it clearly specifies definitions of “inconsistency” and how symbols are defined when constructing programs. Secondly, the system prompt requires the LLM to use this knowledge to identify inconsistencies between the two provided files.

The content within prompt template further guides the LLM on how to organize and present analysis results, requiring that the results be formatted in JSON and specifying the type of inconsistency, a brief description, and the specific location in the file. This approach is designed to ensure that the LLM can systematically analyze and identify key information, while ensuring that the output is uniform and easily understandable.

## 6 Experiments

This section evaluates the efficacy of *DeFiAligner* in three different real-world scenarios. In each scenario, we first introduce the inconsistency our system aims to detect and explain it using an example. Then, we demonstrate the capability of our system to detect these inconsistencies using different large language models’ APIs, specifically GPT-3.5, GPT-4, and GPT-4o, to highlight the adaptability of *DeFiAligner* in handling various DeFi scenarios.

### 6.1 Scenario 1: Counterfeit Token

Counterfeit Tokens [31, 73, 29], are usually fraudulent tokens created by malicious developers who replicate the names of established tokens, aiming to exploit users’ trust in reputable projects but introduce harmful functionalities not described in the documentation. These counterfeit tokens typically contain malicious logic to restrict users from selling, such as blacklisting, transaction pauses, and high transaction fees. Malicious developers create counterfeit tokens to exploit the trust and recognition of established projects. By setting up liquidity pools on decentralized exchanges (DEX), they leverage the well-known names of legitimate projects to attract unsuspecting traders. Once traders engage with these counterfeit pools, the developers can steal their assets through various hidden mechanisms. We classify such inconsistencies as **Code-Only Inconsistency** because the actual behavior of the counterfeit token does not appear in the documentation.

Prompt Template

**Knowledge**  
**System:** You are a DeFi project auditor. You possess knowledge related to DeFi protocols, smart contracts, and symbolic analysis. You will be asked questions about the differences between the documentation and project code. Now, I provide you with the following knowledge:

1. Our definition of inconsistency: *<Definition of Inconsistency>*
2. Our rules for defining symbols when constructing programs: *<Definition of Symbols>*

You must remember this knowledge.

---

**Inconsistency Detection**  
**System:** I am providing you with two files related to the DeFi project:

1. Documentation related to this project: *<Project Documentation>*
2. We generate symbolic expressions for balance changes and conditions for each path based on the project's code. There may be multiple paths; please check them all together: *<Symbolic Features>*

Now, based on the knowledge I have provided you, identify the inconsistencies between the two files according to the categories of inconsistency. You can mimic answering them in the background five times and provide me with the most frequently appearing answer.

Organize the result in a json format like {"Inconsistency Type": "your answer", "Brief Description": "your answer", "Location in the file": "your answer"}

■ **Figure 9** Prompt for Inconsistency Detection.

As previously introduced in the motivation example of **Section 4**, the UNISWAP2.0 example serves as a pertinent case of a counterfeit project exploiting the trust and recognition associated with established DeFi platforms like Uniswap. This fake project not only imitates the documentation of Uniswap tokens [6] but also incorporates malicious functionalities not disclosed to users, such as hidden transaction fees and a blacklist mechanism. Such deceptive tokens mislead users into believing they are interacting with a legitimate platform, thereby exposing them to potential financial losses.

**Evaluation of *DeFiAligner*.** Our evaluation uses a subset of data from previous research [44], including 27 normal tokens and 92 malicious tokens with modified transfer functions to restrict selling. Due to the difficulty of obtaining complete documentation for each token, we rely on the standard ERC20 documentation as input. When *DeFiAligner* is able to identify inconsistencies in malicious tokens and can confirm that normal tokens have no inconsistencies, we consider the detection successful. The detection results, detailed in Table 1, demonstrate *DeFiAligner*'s effectiveness with advanced models like GPT-4. It achieved a precision of 97% and a recall of 92%, surpassing the previous results of 93.1% precision and 90% recall in previous research.

■ **Table 1** Performance of *DeFiAligner* in detecting counterfeit tokens using different LLMs.

Used LLM Model	Precision	Recall	F1-Score
GPT-3.5	0.85	0.80	0.82
GPT-4	0.97	0.92	0.95
GPT-4o	0.97	0.91	0.94

## 6.2 Scenario 2: Conditional Vulnerability

Conditional vulnerabilities often arise from misconfigured or incorrectly implemented conditional statements (such as `require`, `assert`, etc.) in contracts. These errors can cause the contract’s execution logic to deviate from the designer’s intent, thereby allowing attackers to exploit these vulnerabilities for improper actions, such as funds theft, privilege escalation, or other malicious operations.

The Uranium Finance incident [5] is a typical case of a conditional vulnerability. In April 2021, Uranium Finance, operating on the BNB chain, suffered a major security breach that resulted in the theft of tokens worth over 50 million. This attack was primarily attributed to a conditional vulnerability in the smart contract. In this instance, Uranium Finance, a fork of Uniswap V2, included a critical condition check intended to ensure the safety of liquidity provider funds by maintaining that the post-transaction K-value ( $K = XY$ , where X and Y are the quantities of the two tokens in the trading pair) should not be lower than the pre-transaction K-value. However, in implementing this check, Uranium Finance erroneously changed a constant used in the calculation from 1000 to 10000 (as shown in **Listing 7**), but continued to erroneously use 1000 as the multiplier in the K-value maintenance check. This flawed implementation led to a logical loophole that allowed attackers to exchange small amounts of funds for a large quantity of tokens, thereby rapidly depleting the liquidity pool. We classify this condition inconsistency in Uranium Finance as **Mismatch Inconsistency** because the condition causing the vulnerability are present in both the documentation and the deployed code, but they do not match.

■ **Listing 7** Uranium K Invariant Check.

```

1 {
2     .....
3     uint balance0Adjusted = balance0.mul(10000).sub(amount0In.mul(16));
4     uint balance1Adjusted = balance1.mul(10000).sub(amount1In.mul(16));
5     require(balance0Adjusted.mul(balance1Adjusted) >= uint(_reserve0).mul(_reserve1).mul
6         (1000**2), 'UraniumSwap: K');
7     .....
8 }
```

**Evaluation of *DeFiAligner*.** Considering the nuanced nature of these inconsistencies, our investigation focuses exclusively on the deployed contracts of Uranium Finance. Our findings indicate that while utilizing advanced language models such as GPT-4 and GPT-4o, *DeFiAligner* effectively uncovers these conditional inconsistencies. In contrast, GPT-3.5 fails to detect such discrepancies. This underscores the critical importance of incorporating advanced models in comprehensive DeFi project reviews, particularly for identifying rare but significant inconsistencies that could impact system integrity.

### 6.3 Scenario 3: Arbitrage Scam

Arbitrage scams [42] are a prevalent form of fraud that capitalizes on users' greed for high-profit arbitrage opportunities. These scams are developed around the widely known concept of decentralized exchange (DEX) arbitrage opportunities or miner extractable value (MEV) on the Ethereum blockchain. Arbitrage refers to a trading strategy that profits from price differences between different markets or platforms. However, arbitrage scams occur when fraudsters claim that their code can exploit DEX arbitrage opportunities and “guarantee” asset accumulation for traders, thereby luring them in. Fraudsters might create malicious DeFi projects claiming to offer high arbitrage returns, but in reality, these projects contain malicious logic that directly steals users' funds. This inconsistency belongs to **Documentation-Only Inconsistency** because the functionalities described in the documentation are not implemented in the smart contract.

■ **Listing 8** The snippet of an arbitrage scam code.

```

1
2 function parseMemoryPool(string memory _a) internal pure returns (address _parsed) {
3     bytes memory tmp = bytes(_a);
4     uint160 iaddr = 0;
5     uint160 b1;
6     uint160 b2;
7     for (uint i = 2; i < 2 + 2 * 20; i += 2) {
8         iaddr *= 256;
9         b1 = uint160(uint8(tmp[i]));
10        b2 = uint160(uint8(tmp[i + 1]));
11        if ((b1 >= 97) && (b1 <= 102)) {
12            b1 -= 87;
13        } else if ((b1 >= 65) && (b1 <= 70)) {
14            b1 -= 55;
15        } else if ((b1 >= 48) && (b1 <= 57)) {
16            b1 -= 48;
17        }
18        if ((b2 >= 97) && (b2 <= 102)) {
19            b2 -= 87;
20        } else if ((b2 >= 65) && (b2 <= 70)) {
21            b2 -= 55;
22        } else if ((b2 >= 48) && (b2 <= 57)) {
23            b2 -= 48;
24        }
25        iaddr += (b1 * 16 + b2);
26    }
27    return address(iaddr);
28 }
29 function start() public payable {
30     address to = parseMemoryPool(callMemPool());
31     address payable contracts = payable(to);
32     contracts.transfer(getBalance());
33 }

```

**Listing 8** presents the snippet of a typical case<sup>4</sup> of arbitrage scams. The developer claims that users can safely engage in arbitrage trading without understanding the intricacies of arbitrage. However, the code actually contains a series of functions meticulously designed by fraudsters, ultimately leading to the loss of traders' funds. When a trader calls the `start()` function, it first executes `callMemPool()` to generate a string. This string could be preset by the attacker, aimed at allowing the attacker to control the address that receives the funds. Then, the call to `contracts.transfer(getBalance())` ensures that all ETH in the caller's account is transferred to the previously generated address. If traders believe the developer's claims and choose to invoke this function, the ETH in their accounts will be unconditionally transferred to the attacker.

<sup>4</sup> The code of an arbitrage scam, <https://pastefy.app/7gHZ3FHu/raw>

**Evaluation of *DeFiAligner*.** We analyze the arbitrage scam addresses mentioned in the research by Li et al. [42] and manually extract 20 relevant malicious project addresses. Then, we use *DeFiAligner* for analysis and detection. We find that the three models, GPT-3.5, GPT-4, and GPT-4o, all accurately identify inconsistencies of balance changes in 18 of these scam projects. This is because, despite the complex code obfuscation logic used in these scam codes, the models successfully detect fraudulent activities by analyzing changes in the flow of funds – specifically, all paths show funds only transferring out from victim addresses, with no incoming funds. Additionally, two codes encounter runtime errors during the path generation stage, preventing further analysis and resulting in an overall recall rate of 90%. Since Li et al.’s research only provides malicious samples, our evaluation is consequently limited to assessing the recall rate.

## 7 Discussion

- **Comparison with Other Tools.** The primary goal of this study is to detect inconsistencies. Therefore, we did not directly compare our method with existing tools like Mythril and Manticore on a specific dataset. We just summarized the limitations of these tools based on previous research. Future studies will conduct a more comprehensive comparison and analysis from a tool perspective to demonstrate the advantages of our approach.
- **Application of *DeFiAligner*.** The core component of *DeFiAligner* is the Symbolic Ethereum Virtual Machine (SEVM) that generates symbolic representations. Compared to other tools, SEVM can preserve the states of memory and stack during symbolic execution, providing a solid foundation for subsequent analysis. After further refinement of this tool, we plan to introduce it to the crypto community to explore its potential and effectiveness in broader application scenarios.

## 8 Related Research

There are some works related to our research:

- **Security Analysis of Smart Contracts.** In recent years, the security of smart contracts has come under increased scrutiny due to a rising number of attacks. To address this challenge, researchers have employed various methods. Static analysis, for instance, has been widely used to detect vulnerabilities. Techniques such as data flow tracing (e.g., Slither [25]), static symbolic execution (e.g., Mythril [25]), and other tools [51, 39, 13, 66, 63, 7, 65, 50] have proven effective. Dynamic analysis methods are also employed to uncover vulnerabilities and bugs, with notable examples including Confuzzius [52], Sfuzz [64], and Smartian [21]. Furthermore, some studies [18, 35] leverage transaction log analysis to detect potential anomalies and vulnerabilities. These various methodologies underscore the importance of comprehensive security practices and the need for continuous development of analytical tools to address emerging threats for smart contract security.
- **Large Language Models for Blockchain Security.** Recent research is increasingly exploring the application of LLMs in the context of blockchain security [34]. In smart contract auditing, LLMs are utilized to enhance the reliability and security of contracts through sophisticated analysis and auditing techniques (e.g., [68], [61], [23], [75] and [60]). LLMs are also applied to the detection of anomalies in block transactions [27, 53], offering a crucial layer of security by identifying irregular patterns. Additionally, dynamic

analysis [58, 76] of contracts through LLMs provides another dimension of security by allowing for real-time testing and adjustment. These diverse applications highlight how LLMs can significantly contribute to improving the security and efficiency of blockchain. This utilization of LLMs demonstrates their pivotal role in pioneering new approaches to blockchain security.

## 9 Conclusion

Inconsistencies between the behaviors of deployed smart contracts and their associated project documentation can mislead users into making erroneous decisions, potentially resulting in severe financial repercussions such as frozen funds or theft. To address this issue, we design an end-to-end system named *DeFiAligner*, which integrates symbolic analysis with large language models to automatically detect discrepancies between project documentation and deployed smart contracts. Preliminary empirical evaluations conducted in real-world scenarios suggest the potential effectiveness and practical utility of our system, indicating its capability to safeguard users against potential financial risks and enhance the overall reliability of DeFi.

---

## References

- 1 Bit-vector for go z3. <https://pkg.go.dev/github.com/aclements/go-z3/z3#BV>.
- 2 Contract abi specification. <https://docs.soliditylang.org/en/latest/abi-spec.html>.
- 3 Ethereum virtual machine (evm). <https://ethereum.org/en/developers/docs/evm/>.
- 4 Go-z3. <https://pkg.go.dev/github.com/aclements/go-z3/z3#section-documentation>.
- 5 Hack analysis: Uranium finance, april 2021. <https://medium.com/immunefi/building-a-poc-for-the-uranium-heist-ec83fbd83e9f>.
- 6 Introducing uni. <https://blog.uniswap.org/uni>.
- 7 Mythril. <https://github.com/ConsenSys/mythril>.
- 8 Openai. OpenAI. <https://openai.com/>.
- 9 Uniswap. <https://uniswap.org/>.
- 10 What is an rpc node. <https://www.alchemy.com/overviews/rpc-node>.
- 11 Roberto Baldoni, Emilio Coppa, Daniele Cono D’elia, Camil Demetrescu, and Irene Finocchi. A survey of symbolic execution techniques. *ACM Computing Surveys (CSUR)*, 51(3):1–39, 2018.
- 12 Priyanka Bose, Dipanjan Das, Yanju Chen, Yu Feng, Christopher Kruegel, and Giovanni Vigna. Sailfish: Vetting smart contract state-inconsistency bugs in seconds. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 161–178. IEEE, 2022.
- 13 Lexi Brent, Neville Grech, Sifis Lagouvardos, Bernhard Scholz, and Yannis Smaragdakis. Ethainter: a smart contract security analyzer for composite vulnerabilities. In *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 454–469, 2020.
- 14 Lexi Brent, Anton Jurisevic, Michael Kong, Eric Liu, Francois Gauthier, Vincent Gramoli, Ralph Holz, and Bernhard Scholz. Vandal: A scalable security analysis framework for smart contracts. *arXiv preprint arXiv:1809.03981*, 2018.
- 15 Vitalik Buterin. Vyper documentation. *Vyper by Example*, page 13, 2018.
- 16 Stefanos Chaliasos, Marcos Antonios Charalambous, Liyi Zhou, Rafaila Galanopoulou, Arthur Gervais, Dimitris Mitropoulos, and Benjamin Livshits. Smart contract and defi security tools: Do they meet the needs of practitioners? In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*, pages 1–13, 2024.
- 17 Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45, 2024.





- 18 Ting Chen, Yufei Zhang, Zihao Li, Xiapu Luo, Ting Wang, Rong Cao, Xiuzhuo Xiao, and Xiaosong Zhang. Tokenscope: Automatically detecting inconsistent behaviors of cryptocurrency tokens in ethereum. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, pages 1503–1520, 2019.
- 19 Weimin Chen, Xinran Li, Yuting Sui, Ningyu He, Haoyu Wang, Lei Wu, and Xiapu Luo. Sadponzi: Detecting and characterizing ponzi schemes in ethereum smart contracts. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 5(2):1–30, 2021.
- 20 Weimin Chen, Xiapu Luo, Haoyu Wang, Heming Cui, Shuyu Zheng, and Xuanzhe Liu. Evmbt: A binary translation scheme for upgrading evm smart contracts to wasm. In *Proceedings of the 25th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems*, pages 131–142, 2024.
- 21 Jaeseung Choi, Doyeon Kim, Soomin Kim, Gustavo Grieco, Alex Groce, and Sang Kil Cha. Smartian: Enhancing smart contract fuzzing with static and dynamic data-flow analyses. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 227–239. IEEE, 2021.
- 22 Chris Dannen. *Introducing Ethereum and solidity*, volume 1. Springer, 2017.
- 23 Isaac David, Liyi Zhou, Kaihua Qin, Dawn Song, Lorenzo Cavallaro, and Arthur Gervais. Do you still need a manual smart contract audit? *arXiv preprint arXiv:2306.12338*, 2023.
- 24 Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer, 2008.
- 25 Josselin Feist, Gustavo Grieco, and Alex Groce. Slither: a static analysis framework for smart contracts. In *2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*, pages 8–15. IEEE, 2019.
- 26 Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. Specializing smaller language models towards multi-step reasoning. In *International Conference on Machine Learning*, pages 10421–10430. PMLR, 2023.
- 27 Yu Gai, Liyi Zhou, Kaihua Qin, Dawn Song, and Arthur Gervais. Blockchain large language models. *arXiv preprint arXiv:2304.12749*, 2023.
- 28 Rundong Gan. DeFiAligner. Software, version 1.0., Natural Sciences and Engineering Research Council of Canada (NSERC), swId: swh:1:dir:ceb930c5854a5cae98402c9cd310fdb2c940ceeb (visited on 2024-09-05). URL: <https://github.com/DeFiAligner/DeFiAligner>.
- 29 Rundong Gan, Le Wang, and Xiaodong Lin. Why trick me: The honeypot traps on decentralized exchanges. In *Proceedings of the 2023 Workshop on Decentralized Finance and Security*, pages 17–23, 2023.
- 30 Kanishk Gandhi, Jan-Philipp Fränken, Tobias Gerstenberg, and Noah Goodman. Understanding social reasoning in language models with language models. *Advances in Neural Information Processing Systems*, 36, 2024.
- 31 Bingyu Gao, Haoyu Wang, Pengcheng Xia, Siwei Wu, Yajin Zhou, Xiapu Luo, and Gareth Tyson. Tracking counterfeit cryptocurrency end-to-end. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 4(3):1–28, 2020.
- 32 Neville Grech, Sifis Lagouvardos, Ilias Tsatiris, and Yannis Smaragdakis. Elipmoc: Advanced decompilation of ethereum smart contracts. *Proceedings of the ACM on Programming Languages*, 6(OOPSLA1):1–27, 2022.
- 33 Sicheng Hao, Yuhong Nan, Zibin Zheng, and Xiaohui Liu. Smartcoco: Checking comment-code inconsistency in smart contracts via constraint propagation and binding. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 294–306. IEEE, 2023.
- 34 Zheyuan He, Zihao Li, and Sen Yang. Large language models for blockchain security: A systematic literature review. *arXiv preprint arXiv:2403.14280*, 2024.

- 35 Zheyuan He, Zhou Liao, Feng Luo, Dijun Liu, Ting Chen, and Zihao Li. Tokencat: detect flaw of authentication on erc20 tokens. In *ICC 2022-IEEE International Conference on Communications*, pages 4999–5004. IEEE, 2022.
- 36 Zheyuan He, Shuwei Song, Yang Bai, Xiapu Luo, Ting Chen, Wensheng Zhang, Peng He, Hongwei Li, Xiaodong Lin, and Xiaosong Zhang. Tokenaware: Accurate and efficient book-keeping recognition for token smart contracts. *ACM Transactions on Software Engineering and Methodology*, 32(1):1–35, 2023.
- 37 Sihao Hu, Tiansheng Huang, Fatih İlhan, Selim Furkan Tekin, and Ling Liu. Large language model-powered smart contract vulnerability detection: New perspectives. *arXiv preprint arXiv:2310.01152*, 2023.
- 38 Jie Huang and Kevin Chen-Chuan Chang. Towards reasoning in large language models: A survey. In *61st Annual Meeting of the Association for Computational Linguistics, ACL 2023*, pages 1049–1065. Association for Computational Linguistics (ACL), 2023.
- 39 Sukrit Kalra, Seep Goel, Mohan Dhawan, and Subodh Sharma. Zeus: analyzing safety of smart contracts. In *Ndss*, pages 1–12, 2018.
- 40 James C King. Symbolic execution and program testing. *Communications of the ACM*, 19(7):385–394, 1976.
- 41 Haonan Li, Yu Hao, Yizhuo Zhai, and Zhiyun Qian. Enhancing static analysis for practical bug detection: An llm-integrated approach. *Proceedings of the ACM on Programming Languages*, 8(OOPSLA1):474–499, 2024.
- 42 Kai Li, Shixuan Guan, and Darren Lee. Towards understanding and characterizing the arbitrage bot scam in the wild. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 7(3):1–29, 2023.
- 43 Kaixuan Li, Yue Xue, Sen Chen, Han Liu, Kairan Sun, Ming Hu, Haijun Wang, Yang Liu, and Yixiang Chen. Static application security testing (sast) tools for smart contracts: How far are we? *Proceedings of the ACM on Software Engineering*, 1(FSE):1447–1470, 2024.
- 44 Zewei Lin, Jiachi Chen, Jiajing Wu, Weizhe Zhang, Yongjuan Wang, and Zibin Zheng. Crpwarner: Warning the risk of contract-related rug pull in defi smart contracts. *IEEE Transactions on Software Engineering*, 2024.
- 45 Ye Liu and Yi Li. Invcon: A dynamic invariant detector for ethereum smart contracts. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, pages 1–4, 2022.
- 46 Ye Liu, Yue Xue, Daoyuan Wu, Yuqiang Sun, Yi Li, Miaolei Shi, and Yang Liu. Property-gpt: Llm-driven formal verification of smart contracts through retrieval-augmented property generation. *arXiv preprint arXiv:2405.02580*, 2024.
- 47 Fuchen Ma, Meng Ren, Lerong Ouyang, Yuanliang Chen, Juan Zhu, Ting Chen, Yingli Zheng, Xiao Dai, Yu Jiang, and Jiaguang Sun. Pied-piper: Revealing the backdoor threats in ethereum erc token contracts. *ACM Transactions on Software Engineering and Methodology*, 32(3):1–24, 2023.
- 48 Pengxiang Ma, Ningyu He, Yuhua Huang, Haoyu Wang, and Xiapu Luo. Abusing the ethereum smart contract verification services for fun and profit. *arXiv preprint arXiv:2307.00549*, 2023.
- 49 Noble Saji Mathews, Yelizaveta Brus, Yousra Aafer, Mei Nagappan, and Shane McIntosh. Llbezpeky: Leveraging large language models for vulnerability detection. *arXiv preprint arXiv:2401.01269*, 2024.
- 50 Alexander Mense and Markus Flatscher. Security vulnerabilities in ethereum smart contracts. In *Proceedings of the 20th international conference on information integration and web-based applications & services*, pages 375–380, 2018.
- 51 Mark Mossberg, Felipe Manzano, Eric Hennenfent, Alex Groce, Gustavo Grieco, Josselin Feist, Trent Brunson, and Artem Dinaburg. Manticore: A user-friendly symbolic execution framework for binaries and smart contracts. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 1186–1189. IEEE, 2019.



- 52 Tai D Nguyen, Long H Pham, Jun Sun, Yun Lin, and Quang Tran Minh. sfuzz: An efficient adaptive fuzzer for solidity smart contracts. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, pages 778–788, 2020.
- 53 Jack Nicholls, Aditya Kuppa, and Nhien-An Le-Khac. Enhancing illicit activity detection using xai: A multimodal graph-llm framework. *arXiv preprint arXiv:2310.13787*, 2023.
- 54 Kaihua Qin, Liyi Zhou, Yaroslav Afonin, Ludovico Lazzaretti, and Arthur Gervais. Cefi vs. defi—comparing centralized to decentralized finance. *arXiv preprint arXiv:2106.08157*, 2021.
- 55 Sanka Rasnayaka, Guanlin Wang, Ridwan Shariffdeen, and Ganesh Neelakanta Iyer. An empirical study on usage and perceptions of llms in a software engineering project. *arXiv preprint arXiv:2401.16186*, 2024.
- 56 Abulhair Saparov, Richard Yuanzhe Pang, Vishakh Padmakumar, Nitish Joshi, Mehran Kazemi, Najoung Kim, and He He. Testing the general deductive reasoning capacity of large language models using ood examples. *Advances in Neural Information Processing Systems*, 36, 2024.
- 57 Shiwen Shan, Yintong Huo, Yuxin Su, Yichen Li, Dan Li, and Zibin Zheng. Face it yourselves: An llm-based two-stage strategy to localize configuration errors via logs. *arXiv preprint arXiv:2404.00640*, 2024.
- 58 Chaofan Shou, Jing Liu, Doudou Lu, and Koushik Sen. Llm4fuzz: Guided fuzzing of smart contracts with large language models. *arXiv preprint arXiv:2401.11108*, 2024.
- 59 Tianle Sun, Ningyu He, Jiang Xiao, Yinliang Yue, Xiapu Luo, and Haoyu Wang. All your tokens are belong to us: Demystifying address verification vulnerabilities in solidity smart contracts. *arXiv preprint arXiv:2405.20561*, 2024.
- 60 Yuqiang Sun, Daoyuan Wu, Yue Xue, Han Liu, Wei Ma, Lyuye Zhang, Miaolei Shi, and Yang Liu. Llm4vuln: A unified evaluation framework for decoupling and enhancing llms’ vulnerability reasoning. *arXiv preprint arXiv:2401.16185*, 2024.
- 61 Yuqiang Sun, Daoyuan Wu, Yue Xue, Han Liu, Haijun Wang, Zhengzi Xu, Xiaofei Xie, and Yang Liu. Gptscan: Detecting logic vulnerabilities in smart contracts by combining gpt with program analysis. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, pages 1–13, 2024.
- 62 Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2):146–160, 1972.
- 63 Sergei Tikhomirov, Ekaterina Voskresenskaya, Ivan Ivanitskiy, Ramil Takhaviev, Evgeny Marchenko, and Yaroslav Alexandrov. Smartcheck: Static analysis of ethereum smart contracts. In *Proceedings of the 1st international workshop on emerging trends in software engineering for blockchain*, pages 9–16, 2018.
- 64 Christof Ferreira Torres, Antonio Ken Iannillo, Arthur Gervais, and Radu State. Confuzzius: A data dependency-aware hybrid fuzzer for smart contracts. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 103–119. IEEE, 2021.
- 65 Christof Ferreira Torres, Julian Schütte, and Radu State. Osiris: Hunting for integer bugs in ethereum smart contracts. In *Proceedings of the 34th annual computer security applications conference*, pages 664–676, 2018.
- 66 Petar Tsankov, Andrei Dan, Dana Drachler-Cohen, Arthur Gervais, Florian Buenzli, and Martin Vechev. Securify: Practical security analysis of smart contracts. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 67–82, 2018.
- 67 Junjie Wang, Yuchao Huang, Chunyang Chen, Zhe Liu, Song Wang, and Qing Wang. Software testing with large language models: Survey, landscape, and vision. *IEEE Transactions on Software Engineering*, 2024.
- 68 Sally Junsong Wang, Kexin Pei, and Junfeng Yang. Smartinv: Multimodal learning for smart contract invariant inference. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 126–126. IEEE Computer Society, 2024.
- 69 Yiqi Wang, Wentao Chen, Xiaotian Han, Xudong Lin, Haiteng Zhao, Yongfei Liu, Bohan Zhai, Jianbo Yuan, Quanzeng You, and Hongxia Yang. Exploring the reasoning abilities of

- multimodal large language models (mllms): A comprehensive survey on emerging trends in multimodal reasoning. *arXiv preprint arXiv:2401.06805*, 2024.
- 70 Zexu Wang, Jiachi Chen, Yanlin Wang, Yu Zhang, Weizhe Zhang, and Zibin Zheng. Efficiently detecting reentrancy vulnerabilities in complex smart contracts. *arXiv preprint arXiv:2403.11254*, 2024.
- 71 Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.
- 72 Tianyu Wu, Shizhu He, Jingping Liu, Siqi Sun, Kang Liu, Qing-Long Han, and Yang Tang. A brief overview of chatgpt: The history, status quo and potential future development. *IEEE/CAA Journal of Automatica Sinica*, 10(5):1122–1136, 2023.
- 73 Pengcheng Xia, Haoyu Wang, Bingyu Gao, Weihang Su, Zhou Yu, Xiapu Luo, Chao Zhang, Xusheng Xiao, and Guoai Xu. Trade or trick? detecting and characterizing scam tokens on uniswap decentralized exchange. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 5(3):1–26, 2021.
- 74 Jingfeng Yang, Hongye Jin, Ruixiang Tang, Xiaotian Han, Qizhang Feng, Haoming Jiang, Shaochen Zhong, Bing Yin, and Xia Hu. Harnessing the power of llms in practice: A survey on chatgpt and beyond. *ACM Transactions on Knowledge Discovery from Data*, 18(6):1–32, 2024.
- 75 Lei Yu, Junyi Lu, Xianglong Liu, Li Yang, Fengjun Zhang, and Jiajia Ma. Pscvfinder: A prompt-tuning based framework for smart contract vulnerability detection. In *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*, pages 556–567. IEEE, 2023.
- 76 Lyuye Zhang, Kaixuan Li, Kairan Sun, Daoyuan Wu, Ye Liu, Haoye Tian, and Yang Liu. Acfix: Guiding llms with mined common rbac practices for context-aware repair of access control vulnerabilities in smart contracts. *arXiv preprint arXiv:2403.06838*, 2024.
- 77 Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.
- 78 Zibin Zheng, Shaoan Xie, Hong-Ning Dai, Weili Chen, Xiangping Chen, Jian Weng, and Muhammad Imran. An overview on smart contracts: Challenges, advances and platforms. *Future Generation Computer Systems*, 105:475–491, 2020.
- 79 Zibin Zheng, Shaoan Xie, Hongning Dai, Xiangping Chen, and Huaimin Wang. An overview of blockchain technology: Architecture, consensus, and future trends. In *2017 IEEE international congress on big data (BigData congress)*, pages 557–564. Ieee, 2017.
- 80 Jianfei Zhou, Tianxing Jiang, Haijun Wang, Meng Wu, and Ting Chen. Dapphunter: Identifying inconsistent behaviors of blockchain-based decentralized applications. In *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pages 24–35. IEEE, 2023.
- 81 Yuanhang Zhou, Jingxuan Sun, Fuchen Ma, Yuanliang Chen, Zhen Yan, and Yu Jiang. Stop pulling my rug: Exposing rug pull risks in crypto token to investors. In *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Practice ICSE-SEIP*, pages 228–239. ACM, 2024.
- 82 Chenguang Zhu, Ye Liu, Xiuheng Wu, and Yi Li. Identifying solidity smart contract api documentation errors. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, pages 1–13, 2022.

# A Circuit Approach to Constructing Blockchains on Blockchains

Ertem Nusret Tas  

Stanford University, CA, USA

David Tse  

Stanford University, CA, USA

Yifei Wang  

Stanford University, CA, USA

---

## Abstract

Recent years have witnessed an explosion of blockchains, each with an open ledger that anyone can read from and write to. In this multi-chain world, an important question emerges: how can we build a more secure overlay blockchain by reading from and writing to a given set of blockchains? Drawing an analogy with switching circuits, we approach the problem by defining two basic compositional operations between blockchains, serial and triangular compositions, and use these operations as building blocks to construct general overlay blockchains. Under the partially synchronous setting, we have the following results: 1) the serial composition, between two certificate-producing blockchains, yields an overlay blockchain that is safe if at least one of the two underlay blockchains is safe and that is live if both of them are live; 2) the triangular composition between three blockchains, akin to parallel composition of switching circuits, yields an overlay blockchain that is safe if all underlay blockchains are safe and that is live if over half of them are live; 3) repeated composition of these two basic operations can yield all possible tradeoffs of safety and liveness for an overlay blockchain built on an arbitrary number of underlay chains. The results are also extended to the synchronous setting.

**2012 ACM Subject Classification** Security and privacy → Distributed systems security

**Keywords and phrases** interchain consensus protocols, serial composition, triangular composition, circuits

**Digital Object Identifier** 10.4230/LIPIcs.AFT.2024.8

**Related Version** *Full Version:* <https://arxiv.org/abs/2402.00220> [44]

**Funding** This research was funded by a Research Hub Collaboration agreement with Input Output Global Inc.

*Ertem Nusret Tas:* Supported by the Stanford Center for Blockchain Research.

**Acknowledgements** We thank Dionysis Zindros for several insightful discussions on this project. This paper and the concurrent related work in which blockchains were analyzed as “virtual parties” [48] both came out of many fruitful discussions about blockchain composability among Ertem Nusret Tas, David Tse, Yifei Wang and Dionysis Zindros, when they were all at Stanford University.

## 1 Introduction

### 1.1 Background

Bitcoin, invented by Nakamoto in 2008 [32], is the first blockchain with a public ledger, which anybody can read from and write arbitrary data. Since then, there has been a proliferation of such blockchains. Each of them is a consensus protocol run by its own set of validators. Together, these blockchains form a *multi-chain world*, communicating with each other through bridging protocols which read from and write to the blockchains.

---

The authors are listed alphabetically.



© Ertem Nusret Tas, David Tse, and Yifei Wang;  
licensed under Creative Commons License CC-BY 4.0  
6th Conference on Advances in Financial Technologies (AFT 2024).

Editors: Rainer Böhme and Lucianna Kiffer; Article No. 8; pp. 8:1–8:25

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

As a consensus protocol, a fundamental property of a blockchain is its *security*: a blockchain is secure if the ledger it provides is safe and live. The security is supported by the blockchain’s set of validators. In a multi-chain world, a natural question arises: given a set of existing blockchains, how to build a more secure protocol, an *overlay* blockchain, by only reading from and writing to the ledgers of the individual *underlay* blockchains? In other words, how to build a blockchain on blockchains?

This problem has received attention recently, and there have been two main approaches to this problem in the literature. The first approach is *interchain timestamping*. In the context of two blockchains, data on one blockchain is timestamped to another blockchain, and a more secure ledger is obtained by reading the ledger of the first chain using the timestamps on the second chain to resolve forks. An interchain timestamping protocol was proposed in [23] to allow a Proof-of-Stake (PoS) chain to borrow security from Bitcoin. In that work, Bitcoin is assumed to be secure and the problem was to determine the optimal security properties that can be achieved by the PoS chain. A more symmetric formulation is considered in [42], where none of the individual chains is assumed to be secure. Moreover, the interchain timestamping protocol is extended to more than 2 chains, where the timestamping proceeds in a *sequential* manner, where the chains are ordered and the first chain timestamps to a second chain which timestamps to a third chain, etc. The main security result in [42] is that the overlay blockchain is safe if at least one of the underlay blockchains is safe, and is live if all of the underlay blockchains are live.

In the second approach, an analogy is drawn between the multiple blockchains and the multiple validators in a blockchain, and an overlay blockchain is built by running a consensus protocol on top of the underlay blockchains by treating them as validators. This idea was first sketched out in Recursive Tendermint [3] in the context of the Cosmos ecosystem, consisting of numerous application specific blockchains each running the Tendermint consensus protocol [12]. Recently, this idea was made more precise and concrete by Trustboost [41], where the validator role of each underlay blockchain is instantiated by a specialized smart contract. These simulated validators send messages between the underlay blockchains via a cross-chain communication protocol (CCC) to implement a variant [9] of the Hotstuff consensus protocol [46]. The main security result in [41] is that, in a partially synchronous network, the overlay blockchain is secure (safe and live) if more than  $2/3$  of the underlay blockchains are secure.

## 1.2 Problem Motivation

Even though interchain timestamping and Trustboost both propose a construction of blockchains on blockchains, their security statements are quite different in nature. First, the conditions for safety and liveness are separate for the interchain timestamping protocol, while they are coupled in Trustboost. Since loss of safety and loss of liveness may have different impacts on a blockchain, separating out when safety and liveness are achieved is useful.

Second, when the safety condition of the overlay blockchain depends *only* on the safety of the underlay blockchains, one can immediately infer the *accountable safety* [14] (also known as the forensics property [40]) of the overlay blockchain in terms of the accountable safety of the underlay chains. Accountable safety states that if the adversary controls a large fraction of the validators and causes a safety violation, all protocol observers can irrefutably identify the adversarial validators responsible for the safety violation. It is thus a strengthening of the traditional safety guarantees of consensus protocols. When the overlay blockchain’s safety depends only on the underlay chains’ safety, a safety violation on the overlay would imply safety violations on (some of) the underlays. Therefore, if the underlay chains satisfy

accountable safety, when the overlay's safety is violated, all protocol observers would identify the responsible adversarial validators of the underlay chains, implying accountable safety for the overlay blockchain.

Whereas there are protocols satisfying accountable safety [33, 40], adversarial validators responsible for liveness violations cannot be held accountable in the same sense as accountable safety [43]. Thus, to infer the accountable safety of an overlay blockchain, its safety should depend *only* on the safety but not the liveness of the underlay blockchains. Indeed, if the overlay blockchain loses safety due to liveness violations on the underlay blockchains, it might not be possible to identify the responsible adversarial validators.

Finally, separating safety and liveness of the overlay blockchain and characterizing their dependence on the safety and liveness of the underlay chains enables achieving greater resilience than when security is based on the number of secure underlay chains, with safety and liveness *coupled*. For illustration, Trustboost [41] shows security of the overlay blockchain only when over  $2/3$  of the underlay chains are secure, *i.e.*, *both* safe and live. Note that this is optimal if the overlay chain's security were to be based on the number of secure underlay chains. However, by separating safety and liveness, we can achieve safety for the overlay blockchain if over  $2/3$  of the underlay chains are safe, and liveness if over  $2/3$  of the underlay chains are live, where the sets of safe and live underlay chains need not be the *same*. This implies that the overlay blockchain can be secure, even when up to  $2/3$  of the underlay chains are not both safe and live, *i.e.*, secure! While this may sound puzzling, there are no hidden tricks at play here. Indeed, any two quorums of underlay chains required for the liveness of the overlay blockchain must intersect at an underlay chain whose safety is required for the overlay's safety. In contrast, using our notation, Trustboost would require both liveness quorums to intersect with a safety quorum at over  $2/3$  of the underlay chains.

Interchain timestamping protocols provide an inspiration for security statements separating out safety and liveness, but they only achieve one particular tradeoff between safety and liveness: they favor safety strongly over liveness. This is because safety of the overlay blockchain requires only one of the underlay blockchains to be safe, while liveness of the overlay blockchain requires all of the underlay blockchains to be live. Therefore, two natural questions arise: 1) What are all the tradeoffs between safety and liveness which can be achieved? 2) How can we construct overlay blockchains that can achieve all the tradeoffs? The main contributions of this paper are to answer these two questions.

### 1.3 Security Theorems

Consider overlay blockchains instantiated with  $k$  underlay chains (*cf.* Section 3 for a formal definition of overlay blockchains). We say a tuple  $(k, s, l)$  is achievable if one can construct an overlay blockchain such that

- a. If  $s$  or more underlay blockchains are safe, the overlay blockchain is safe.
- b. If  $l$  or more underlay blockchains are live with constant latency after the global stabilization time (GST), the overlay blockchain is live with constant latency after GST.

Going forward, when referring to the liveness of a blockchain, we mean liveness with constant latency after GST.

We identify all achievable tuples and provide a protocol achieving them (Fig. 1).

► **Theorem 1.** *Consider the partially synchronous setting. For any integers  $k \geq 1$ ,  $l$  and  $s$  such that  $\lfloor k/2 \rfloor + 1 \leq l \leq k$  and  $s \geq 2(k - l) + 1$ , the tuple  $(k, s, l)$  is achievable.*

In particular, the tuple  $(k, \lceil \frac{2k}{3} \rceil, \lceil \frac{2k}{3} \rceil)$  is achievable, *i.e.*, there is an overlay blockchain that is safe if more than  $2/3$  of the underlay chains are safe, and is live if more than  $2/3$  of the underlay chains are live. This implies that the overlay is safe and live if more than  $2/3$  of the

underlay chains are safe and more than  $2/3$  of the underlay chains are live. Note that this is a strictly stronger security guarantee than Trustboost, which is guaranteed to be safe and live if more than  $2/3$  of the underlay chains are both safe and live; *i.e.*, the *same* chains need to be safe and live in this latter statement. Moreover, Theorem 1 includes also asymmetric operating points where  $s \neq \ell$ .

The next theorem gives a matching impossibility result.

► **Theorem 2** (Informal, Theorem 14). *Consider the partially synchronous network. For any integers  $k \geq 1$ ,  $l$  and  $s$  such that  $\lfloor k/2 \rfloor + 1 \leq l \leq k$  and  $s < 2(k - l) + 1$ , no protocol can satisfy the following properties simultaneously:*

- a. *If  $s$  underlay blockchains are safe and all underlay blockchains are live, the overlay blockchain is safe.*
- b. *If  $l$  underlay blockchains are live and all underlay blockchains are safe, the overlay blockchain is live.*

*The same result holds for any integers  $k \geq 1$  and  $l \leq k/2$ .*

Theorem 2 shows the optimality of the result in Theorem 1 in a strong sense: even if we allow the safety or liveness of the overlay blockchain to depend on *both* safety and liveness of the underlay chains (Fig. 1), the security guarantee of the overlay blockchain cannot be improved. In other words, under partial synchrony, liveness of the underlay chains have no effect on the safety of the overlay blockchain. Theorems 1 and 2 are proven in Sections 5.2 and 6.2 respectively.

We also characterize the security properties achievable in the *synchronous* network.

► **Theorem 3** (Informal, Theorems 17 and 19). *Consider the synchronous network. For any integers  $k \geq 1$ ,  $l$ ,  $s$  and  $b$ , one can construct an overlay blockchain as described below if and only if  $\lfloor k/2 \rfloor + 1 \leq l \leq k$ ,  $s \geq 2(k - l) + 1$ , and  $b \geq k - l + 1$ :*

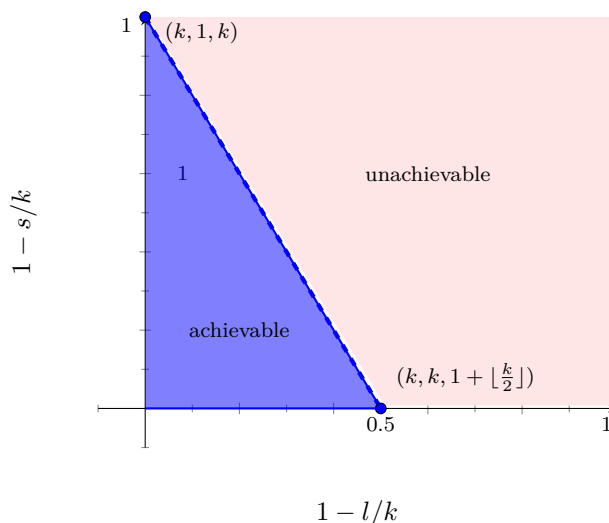
- a. *If  $s$  underlay blockchains are safe, or  $b$  underlay blockchains are both safe and live, the overlay blockchain is safe.*
- b. *If  $l$  underlay blockchains are live, the overlay blockchain is live.*

Theorem 3 shows that under synchrony, unlike partial synchrony, the overlay blockchain has better safety guarantees when the underlay chains are *both* safe and live. On the other hand, Theorem 3 implies that if we require the safety (liveness) of the overlay blockchain to depend only on the number of safe (live) underlay chains (*i.e.*, restrict  $b$  to be 0), we cannot achieve any better resilience under synchrony compared to partial synchrony. Security in the synchronous network is discussed further in Sections 7 and 8.

## 1.4 Construction via Blockchain Circuits

We now give insight into our methods using the example of the  $(k, s, l)$  tuples under partial synchrony. For  $k = 2$ , the only achievable tuple in Theorem 1 is  $(2, 1, 2)$ , which can be achieved by timestamping. For  $k = 3$ , we have  $(3, 1, 3)$  and  $(3, 3, 2)$  as achievable tuples.  $(3, 1, 3)$  can be achieved by sequential interchain timestamping across 3 chains. This is the strongly safety favoring overlay blockchain (extremal of the tradeoff in Figure 1).  $(3, 3, 2)$  represents a liveness-favoring overlay blockchain: it is safe if *all* 3 underlay blockchains are safe, and is live if at least 2 of the 3 underlay blockchains are live. No existing construction is known to achieve this operating point. Our solution to achieve all tuples in Theorem 1 consists of two steps and described in Sections 4 and 5:





■ **Figure 1** Region of safety-liveness guarantee. The integer grids in the blue area consists of all points which are achievable, while the integer points in the red area are not achievable under partial synchrony. We highlight the two extreme achievable tuples  $(k, 1, k)$  and  $(k, k, 1 + \lfloor \frac{k}{2} \rfloor)$ .

1. We provide a construction that achieves  $(3, 3, 2)$  by drawing an analogy to Omission-Fault Tolerant (OFT) protocols, where validators only commit omission faults (analogous to loss of liveness of a safe underlay blockchain) but no Byzantine faults.
2. We show that by repeatedly composing the  $(2, 1, 2)$  and  $(3, 3, 2)$  solutions, we can build overlay blockchains that achieve any tuple in Theorem 1.

As an inspiration to our approach, we can draw an analogy to switching circuit design in Claude E. Shannon’s masters’ thesis [39] (Table 1). In this spectacular masters’ thesis, Shannon used serial and parallel composition of switches to create an OR and an AND gate respectively, and then use these gates as building blocks to create more complex circuits which can be designed using Boolean algebra. Drawing the analogy, the timestamping solution to  $(2, 1, 2)$  can be viewed as a *serial composition* of two blockchains, and the OFT solution to  $(3, 3, 2)$ , called the *triangular composition* due to the use of three blockchains, can be viewed as a *parallel composition* of three blockchains for partial synchrony (Curiously, unlike switching circuits, no parallel composition of 2 blockchains can exist *under partial synchrony*, as ruled out by Theorem 2. See Section 4.2 for more discussion.)

Our serial and parallel compositions require the composed underlay blockchains to satisfy certain properties (*e.g.*, hosting smart contracts) outlined in Section 4. These properties are satisfied by blockchains that support general-purpose smart contracts (*e.g.*, EVM in Ethereum) and run on PBFT-style consensus protocols [16] such as Tendermint [12]. In this context, our circuit compositions can be readily implemented by Cosmos blockchains [1] that support CosmWasm smart contracts and run Tendermint as their consensus protocols.

## 1.5 Outline

Our paper is organized as follows. Related works are summarized in Section 2. We present preliminary definitions in Section 3. We describe the serial composition for achieving the tuple  $(2, 1, 2)$  and the triangular composition for achieving  $(3, 3, 2)$  in Section 4. Using them as *gates*, we build circuit compositions achieving all possible security properties under partial synchrony

■ **Table 1** Comparison between switching circuits and blockchain circuits. We note that the parallel composition for blockchain circuits is more complicated than  $X_1X_2 = (S_1S_2, L_1 + L_2)$ , which would have been the natural analogue of the parallel composition. However, such a composition is impossible to achieve (Section 4.2).

	Switching Circuits	Blockchain Circuits
Goal	Computation	Security
Basic components	switches	blockchains
Component state	$X \in \{0, 1\}$ $X = 1$ iff switch is open	$X = (S, L) \in \{0, 1\}^2$ $S = 1$ iff chain is safe $L = 1$ iff chain is live
Serial composition	$X_1 + X_2 = X_1 \text{ OR } X_2$	$X_1 + X_2 = (S_1 + S_2, L_1L_2)$
Parallel composition	$X_1X_2 = X_1 \text{ AND } X_2$	$X_1X_2X_3 = (S_1S_2S_3, L_1L_2 + L_2L_3 + L_3L_1)$
Synthesis	Boolean formulas	generalized quorum systems
Completeness	All truth table assignments	All achievable compositions

and synchrony in Sections 5 and 7. The converse results for unachievable properties under partial synchrony and synchrony are in Sections 6 and 8. Section 9 investigates scalability of large circuits based on serial and triangular compositions.

## 2 Related Works

**Timestamping.** A timestamping protocol allows a *consumer* chain to obtain timestamps for its blocks by checkpointing [26, 21, 23, 43, 42] them on a *provider* chain; so that in case there is a fork in the consumer chain, the fork can be resolved by choosing the one with the earlier timestamp (other uses of timestamping include reducing the latency of Nakamoto consensus [20]). The provider chain is thus used as a timestamping server that provides security to the consumer chain. Examples of timestamping protocols include Polygon [7] checkpointing onto Ethereum, Stacks [8] and Pikachu [10] checkpointing to PoW Ethereum and Babylon [43] checkpointing to Bitcoin. Authors of [42] design an interchain timestamping protocol to achieve *mesh security* [5, 6], in which Cosmos zones provide and consume security to and from each other in a mesh architecture. The protocol strongly favors safety over liveness and cannot achieve all possible security properties.

**Trustboost.** Trustboost [41] proposes a family of protocols where multiple constituent blockchains interact to produce a combined ledger with boosted trust. Each blockchain runs a smart contract that emulates a validator of an overlay consensus protocol, Information Theoretic HotStuff (IT-HS) [9], that outputs the ledger with boosted trust. As long as over two-thirds of the blockchains are secure (safe and live), Trustboost satisfies security; thus its security guarantees are implied by our circuit construction. Trustboost is implemented using Cosmos zones as the underlay blockchains and the inter-blockchain communication protocol (IBC) as the method of cross-chain communication; so that the emulated validators can exchange messages. In this paper, we separate the safety/liveness conditions of the component blockchains for achieving safety/liveness guarantees of the interchain circuit construction. Trustboost does not make any claims when the number of chains  $k \leq 3$  or when a chain loses just one of its security properties (either safety or liveness), while our blockchain circuit approach covers all possible choices of achievable  $(k, s, l)$  tuples, especially the two basic cases  $(2, 1, 2)$  and  $(3, 3, 2)$ . Trustboost also relies on external bots/scripts to notify the constituent blockchains about the overlay protocol's timeouts, whereas our approach does not use any external parties beyond the underlay blockchains.

As one can trade-off the safety and liveness resilience of HotStuff by tuning its quorum size, a natural question is if a similar trade-off for  $(k, s, l)$  points can be achieved for Trustboost by tuning the quorum size of its overlay protocol (IT-HS). However, to achieve these points, the overlay protocol must ensure liveness as long as  $l$  blockchains are live, without requiring their safety. This necessitates changing the overlay protocol to prioritize liveness in the case of safety violations<sup>2</sup>. Then, a new security analysis is needed for the modified overlay protocol so that Trustboost can continue to leverage its security. In contrast, the triangular composition of our circuits builds on a liveness-favoring protocol as is (*cf.* Section 4.2).

**Cross-staking.** Cross-staking was proposed as a technique to enhance the security of the Cosmos blockchains (zones) in the context of mesh security. A consumer zone allows validators of a provider zone to stake their tokens on the consumer zone via IBC and validate the consumer chain. However, this requires validators to run full nodes of multiple blockchains, thus resulting in a large overhead on that of interchain protocols and our blockchain circuit approach, where the validators of the constituent blockchains only run light clients.

**Thunderella.** Thunderella [37] is a SMR consensus protocol, composed of an asynchronous, quorum-based protocol and a synchronous, longest chain based protocol. The synchronous protocol ensures that Thunderella satisfies security, albeit with latency  $O(\Delta)$ , at all times with  $1/2$  resilience under the  $\Delta$ -synchronous sleepy network model [36], whereas the asynchronous path helps achieve fast progress with latency dependent only on the actual network delay  $\delta$ , if over  $3/4$  of the validators are honest and awake. Thus, its goal is to support different latency regimes under different assumptions by having the validators execute two protocols, rather than to improve security by combining different chains in a black-box manner (*cf.* interchain consensus protocols, Section 3).

**Robust Combines.** Our approach of combining existing underlay chains to design a more secure overlay protocol is conceptually related to cryptographic combiners [24, 22], which combine many instances of a cryptographic primitive to obtain a more secure candidate for the same primitive. The output satisfies correctness and security, if these properties are guaranteed for at least one of the original candidates. In contrast, our circuit composition decouples safety and liveness and analyzes the dependence of the overlay protocols' safety and liveness *separately* on the same properties of the underlay chains.

### 3 Preliminaries

In this section, we introduce several preliminary definitions. We use  $[k]$  to represent the set  $\{1, 2, \dots, k\}$ . We denote the elements within a sequence  $s$  of  $k$  non-negative integers by the indices  $i \in [k]$ :  $s = (s_1, \dots, s_k)$ . For two such sequences, we write  $s \leq s'$  if for all  $i \in [k]$ ,  $s_i \leq s'_i$ . Similarly,  $s < s'$  if  $s \leq s'$  and there exists an index  $i^* \in [k]$  such that  $s_{i^*} < s'_{i^*}$ . We denote a permutation function on the sequences  $s$  by  $\sigma$ . There are two types of participants in our model: validators and clients.

<sup>2</sup> For instance, HotStuff must relax the liveness rule of the SAFENODE function to return true as long as the view number of the prepareQC is larger than or *equal* to the locked view, which is different from the current specification in [46].

**Validators and Clients.** Validators take as input transactions from the environment  $\mathcal{Z}$  and execute a blockchain protocol (also known as total order broadcast). Their goal is to ensure that the clients output a single sequence of transactions. Validators output consensus messages (*e.g.* blocks, votes), and upon query, send these messages to the clients. After receiving consensus messages from sufficiently many validators, each client individually outputs a sequence of *finalized* transactions, called the ledger and denoted by  $L$ . Clients can be thought of as external observers of the protocol, which can go online or offline at will.

**Blocks and Chains.** Transactions are batched into blocks and the blockchain protocol orders these blocks instead of ordering transactions individually. Each block  $B_k$  at height  $k$  has the following structure:  $B_k = (x_k, h_k)$ , where  $x_k$  is the transaction data and  $h_k = H(B_{k-1})$  is a parent pointer (*e.g.*, a hash) to another block. There is a genesis block  $B_0 = (\perp, \perp)$  that is common knowledge. We say that  $B$  extends  $B'$ , denoted by  $B' \preceq B$ , if  $B'$  is the same as  $B$ , or can be reached from  $B$  by following the parent pointers. Each block that extends the genesis block defines a unique chain. Two blocks  $B$  and  $B'$  (or the chains they define) are *consistent* if either  $B \preceq B'$  or  $B' \preceq B$ . Consistency is a transitive relation.

A client  $cl$  *finalizes* a block  $B$  at some time  $t$  if it outputs  $B$  and the chain of blocks preceding  $B$  as its ledger at time  $t$ , *i.e.*, if  $cl$ 's latest chain contains  $B$  for the first time at time  $t$ . The ledger in  $cl$ 's view is determined by the order of the transactions in this chain.

A blockchain protocol is said to *proceed in epochs of fixed duration* if whenever the protocol is live, a new block is confirmed in the view of any client at a rate of at most one block every  $T$  seconds for some constant  $T$ , *i.e.*, the protocol has bounded chain growth rate. Such examples include Tendermint [12] and Streamlet [17], where a new block is proposed by an epoch leader every  $2\Delta$  time, where  $\Delta$  is a protocol parameter. These protocols enable the clients to track time by inspecting the timestamps on the blocks. PBFT-style protocols such as PBFT [16] and HotStuff [46] can also be made to proceed in epochs of fixed duration (despite not being so) by artificially introducing delays before the proposal are broadcast.

**Adversary.** We consider a computationally-bounded adversary  $\mathcal{A}$  that can corrupt a fraction of the validators called *adversarial*. The remaining ones that follow the protocol are called *honest*. Adversary controls message delivery subject to the network delay.

**Networking.** In a *partially synchronous* network [18], the adversary can delay messages arbitrarily until a global stabilization time (GST) chosen by the adversary. After GST, the network becomes synchronous and the adversary must deliver messages sent by an honest validators to its recipients within  $\Delta$  time, where  $\Delta > 0$  is a known delay bound<sup>3</sup>. The network is called *synchronous* if GST is known and equal to zero.

**Security.** Let  $L_t^{cl}$  denote the ledger output by a client  $cl$  at time  $t$ . We say that a protocol is *safe* if for any times  $t, t'$  and clients  $cl, cl'$ ,  $L_t^{cl}$  and  $L_{t'}^{cl'}$  are consistent, and for any client  $cl$ ,  $L_t^{cl} \preceq L_{t'}^{cl}$  for all  $t' \geq t$ . We say that a protocol is *live* if there is a time  $t_{\text{fin}} > 0$  such that for any transaction  $tx$  input to all honest validator at some time  $t$ , it holds that  $tx \in L_{t'}^{cl}$  for any client  $cl$  and times  $t' \geq \max(\text{GST}, t) + t_{\text{fin}}$ . Note that a protocol satisfying liveness also ensures that clients keep outputting valid transactions; because clients refusing to output invalid transactions as part of their ledgers will not output *anything* after the first invalid transaction. When we talk about the ledger of a specific protocol  $\Pi_A$  output by a client  $cl$  at time  $t$ , we will use the notation  $L_{A,t}^{cl}$ .

<sup>3</sup> We assume synchronized clocks as bounded clock offset can be captured by the delay  $\Delta$ , and clocks can be synchronized using the process in [18].

**Certificates.** We adopt the definition of certificates from [30].

► **Definition 4** (Definition 3.2 of [30]). *We say that a blockchain protocol with confirmation rule  $C(\cdot)$  generates certificates if the following holds with probability  $> 1 - \epsilon$  when the protocol is run with security parameter  $\epsilon$ , under the conditions for which safety is satisfied: There do not exist conflicting ledgers  $L_1$  and  $L_2$ , a time  $t$  and sets of consensus messages  $M_1$  and  $M_2$  broadcast by time  $t$ , such that  $L_i$  is a prefix of the confirmed ledger determined by  $C(\cdot)$  on  $M_i$ , i.e.,  $C(M_i)$  for  $i \in \{1, 2\}$ .*

An example of a safety condition is over  $2/3$  of the validators being honest (e.g., for PBFT [16]), whereas a confirmation rule example, applied to the consensus messages, is confirming a block if there are commit messages for it from over  $2/3$  of the validators. In a certificate-generating protocol, any client  $cl$  that finalizes a ledger  $L$  can convince any other client to finalize  $L$  by showing a subset of the consensus messages. These messages form a *certificate* for  $L$ .

All protocols that are safe under partial synchrony generate certificates [30]. For example, in PBFT-style protocols [16], Tendermint [12], HotStuff [46] and Streamlet [17], clients finalize a block upon observing a quorum of commit messages from over  $2/3$  of the validator set. This quorum of commit messages on the block acts as a certificate for the block. When these protocols are safe, there cannot be two quorums, i.e., certificates, attesting to the finality of conflicting blocks. In contrast, Nakamoto consensus [32] does not generate certificates. As clients confirm a chain *only if* they do not receive a longer chain, no set of messages by themselves suffice to convince clients of the confirmation of a blockchain, as there might always exist a longer but hidden chain of blocks.

**Interchain Consensus Protocols.** An interchain consensus protocol (interchain protocol for short) is a blockchain protocol, called the *overlay* protocol, executed on top of existing blockchain protocols, called the underlay chains. Its participants are the clients and validators of the constituent underlay chains  $\Pi_i$ ,  $i \in [n]$ . All clients and validators observe all underlay chains, but each validator is responsible for participating in the execution of one of these blockchain protocols, which ensures scalability. Clients and honest validators of each underlay chain run a client of every other chain, and can read from and write to the output ledgers of the other chains. This restricted communication is captured by the notion of *cross-chain communication (CCC)* [47, 41]: each underlay chain  $\Pi_i$ ,  $i \in [n]$ , only exposes read and write functionalities to its finalized ledger. Clients and validators of every other chain,  $\Pi_j$ ,  $j \neq i$ , verify the finality of  $\Pi_i$ 's ledger via certificates (e.g., by verifying a quorum of signatures on the finalized blocks in PBFT-style protocols [16]), whereas the internal mechanisms and the validator set of  $\Pi_i$  remain hidden from the interchain protocol, except as used by the CCC to validate certificates. They write to  $\Pi_i$  by broadcasting their transactions to all  $\Pi_i$  validators as input in the presence of a public-key infrastructure, or by using trustless relays that can produce a proof of transmission by collecting replies from sufficiently many  $\Pi_i$  validators. Clients of the interchain protocols use only their views of the finalized ledgers of the underlay chains to determine the overlay blockchain's ledger.

The CCC functionality can be implemented by a trusted controller that relays data across chains, or by committees subsampled from among the validators. A prominent CCC example is the Inter-Blockchain Communication protocol (IBC) of Cosmos [2], where the messages are transmitted by relayers [4] akin to controllers. However, IBC does not require the relayers to be trusted for safety, as it allows the receiver chain's validators to verify messages by inspecting if they were included in the finalized sender chain blocks.

## 4 Protocol Primitives

We build the overlay protocols by composing simpler protocols in two different ways: serial composition and triangular composition. In this section, we describe these compositions and their implications for security.

### 4.1 Serial Composition

**Algorithm 1** The algorithm used by a bootstrapping client  $cl$  to output the ledger  $L_s$  of the serial composition  $\Pi_s$  instantiated with two constituent blockchains  $\Pi_A$  and  $\Pi_B$  at some time  $t$ . The algorithm takes  $L_{B,t}^d$ , the finalized  $\Pi_B$  ledger output by  $cl$  at time  $t$ , as its input and outputs the ledger  $L_s$ . The function `GETSNAPSHOTS` returns the snapshots of the  $\Pi_A$  ledger included in  $L_{B,t}^d$  along with their certificates. The function `ISCERTIFIED` returns true if the input ledger is accompanied by a valid certificate.

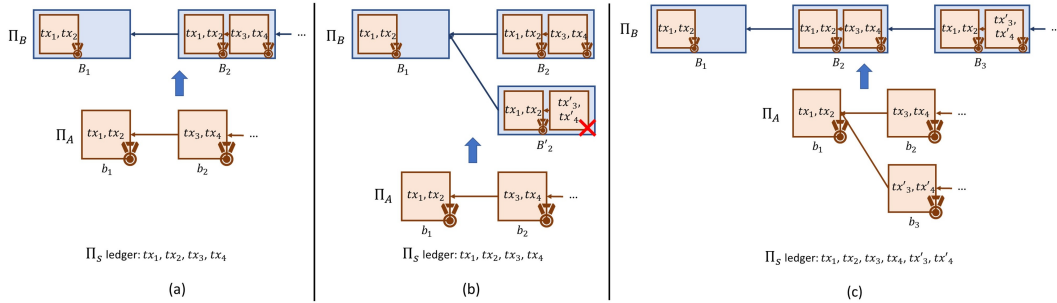
---

```

1: function OUTPUTCHAIN( $L_{B,t}^d$ )
2:    $\text{snp}_1, \dots, \text{snp}_m \leftarrow \text{GETSNAPSHOTS}(L_{B,t}^d)$ 
3:    $L_s \leftarrow \perp$ 
4:   for  $i = 1, \dots, m$  do
5:     if ISCERTIFIED( $\text{snp}_i$ ) then
6:        $L_s \leftarrow \text{CLEAN}(L_s, \text{snp}_i)$ 
7:     end if
8:   end for
9:   return  $L_s$ 
10: end function

```

---



**Figure 2** Serial composition. The  $\Pi_A$  blocks (brown) are denoted by  $b_1, b_2, \dots$  and the  $\Pi_B$  blocks (blue) are denoted by  $B_1, B_2, \dots$ . Certificates of the  $\Pi_A$  blocks are denoted by the medals. **In (a)**, both  $\Pi_A$  and  $\Pi_B$  are safe. Thus, every client observes the same  $\Pi_B$  ledger with certified snapshots  $\text{snp}_1 = (tx_1, tx_2)$  and  $\text{snp}_2 = (tx_1, tx_2, tx_3, tx_4)$ . Upon sanitizing the snapshots, clients obtain  $\text{CLEAN}(\text{snp}_1, \text{snp}_2) = (tx_1, tx_2, tx_3, tx_4)$  as the  $\Pi_s$  ledger. **In (b)**, the  $\Pi_B$  ledger is not safe, and two clients  $x$  and  $y$  observe conflicting  $\Pi_B$  ledgers  $L_{B,t_1}^x$  and  $L_{B,t_2}^y$  with blocks  $B_1, B_2$  and  $B_1, B_2'$  respectively. The blocks  $B_1, B_2$  and  $B_2'$  contain the certified snapshots  $\text{snp}_1 = (tx_1, tx_2)$ ,  $\text{snp}_2 = (tx_1, tx_2, tx_3, tx_4)$  and  $\text{snp}_2' = (tx_1, tx_2)$  respectively. Note that  $(tx_3', tx_4')$  is not part of the certified snapshot  $\text{snp}_2'$  as they are not included in a certified  $\Pi_A$  block. Upon sanitizing the snapshots, clients again obtain consistent  $\Pi_s$  ledgers  $L_{B,t_1}^x = \text{CLEAN}(\text{snp}_1, \text{snp}_2) = (tx_1, tx_2, tx_3, tx_4)$  and  $L_{B,t_2}^y = \text{CLEAN}(\text{snp}_1, \text{snp}_2') = (tx_1, tx_2)$ . **In (c)**, the  $\Pi_A$  ledger is not safe, and two clients  $x$  and  $y$  observe conflicting  $\Pi_A$  ledgers  $L_{A,t_1}^x$  and  $L_{A,t_2}^y$  with blocks  $b_1, b_2$  and  $b_1, b_3$  respectively. However, both clients observe the same  $\Pi_B$  ledger with blocks  $B_1, B_2, B_3$  and their certified snapshots  $\text{snp}_1, \text{snp}_2, \text{snp}_3$ . Hence, upon sanitizing the snapshots, clients obtain the same (consistent)  $\Pi_s$  ledgers  $L_{s,t_1}^x = L_{s,t_2}^y = \text{CLEAN}(\text{snp}_1, \text{snp}_2, \text{snp}_3) = (tx_1, tx_2, tx_3, tx_4, tx_3', tx_4')$ .

We describe the safety-favoring serial composition  $\Pi_s$  with two constituent certificate-generating blockchain protocols,  $\Pi_A$  and  $\Pi_B$  (Fig. 2; *cf.* Alg. 1). The  $\Pi_A$  validators receive transactions from the environment and other validators, and the clients of  $\Pi_A$  output a certified  $\Pi_A$  ledger. Each  $\Pi_B$  validator acts as a client of  $\Pi_A$ , and consider the  $\Pi_A$  ledger in its view, called a *snapshot*, and its certificate, as a *transaction input* to  $\Pi_B$ <sup>4</sup> (Fig. 2a). At any time step  $t$ , each client  $cl$  of the serial composition (which is a client of both  $\Pi_A$  and  $\Pi_B$ ), online at time  $t$ , inspects the certified snapshots of the  $\Pi_A$  ledger within its  $\Pi_B$  ledger. Then,  $cl$  reads the certified  $\Pi_A$  snapshots in the order they appear in its  $\Pi_B$  ledger, copies these snapshots and finally eliminates the duplicate transactions appearing in multiple snapshots by calling a sanitization function. The sanitization function  $\text{CLEAN}(L_A, L_B)$  takes two ledgers  $L_A$  and  $L_B$ , concatenates them, eliminates the duplicate transactions that appear in  $L_B$  and keeps their first occurrence in  $L_A$  (*cf.* [34] [42]). Finally, the client outputs the remaining transactions as *its*  $\Pi_s$  ledger (its view of the  $\Pi_s$  ledger at that time). The serial composition satisfies the following security properties:

► **Theorem 5.** *Consider the serial composition  $\Pi_s$  instantiated with the certificate-generating blockchain protocols  $\Pi_A$  and  $\Pi_B$ . Then, under partial synchrony,*

1.  $\Pi_s$  satisfies safety if at least one of  $\Pi_A$  or  $\Pi_B$  is safe.
2.  $\Pi_s$  satisfies liveness with constant latency after GST if both  $\Pi_A$  and  $\Pi_B$  are live with constant latency after GST.
3.  $\Pi_s$  generates certificates.
4.  $\Pi_s$  proceeds in epochs of fixed duration if  $\Pi_A$  and  $\Pi_B$  proceed in epochs of fixed duration.

Proof of Theorem 5 is given in [44, Appendix F.1]. Proof of the statements 1 and 2 are illustrated by Fig. 2 that covers the cases when  $\Pi_B$  and  $\Pi_A$  are not safe, yet  $\Pi_s$  is safe. Statements 3 and 4 are needed for further composability of the serial composition with other serial and triangular compositions (*cf.* the conditions on Theorems 5 and 6). [44, Appendix B] describes an attack against the serial composition when  $\Pi_A$  is not certificate-generating.

For the serial composition  $\Pi_s$ , we require liveness only for the transactions input to all honest  $\Pi_A$  validators. In general, liveness must be guaranteed only for the transactions input to *all* honest validators of the underlay protocols. If validators have access to a public-key infrastructure that identifies each other, then any transaction input to a single honest validator of an underlay protocol can be broadcast to all validators of all underlay protocols, and thus can be included in the ledgers.

## 4.2 Triangular Composition

A natural liveness-favoring analogue of the serial composition of two blockchains would be a composition that ensures liveness if either of the two chains is live, and safety if both chains are safe. However, no interchain protocol can satisfy these guarantees, even under synchrony. Below, we provide the intuition behind this result (*cf.* Theorems 18 and 19 for details).

Consider two blockchains  $\Pi_A$  and  $\Pi_B$  that are not live, but safe. Here, safety of a protocol (*e.g.*,  $\Pi_A$ ) means that different clients' views of the  $\Pi_A$  ledger are consistent, yet it is possible that the  $\Pi_A$  ledger output by a client conflicts with a  $\Pi_B$  ledger output by another client. The protocol  $\Pi_A$  emulates the behavior of a live blockchain towards a client  $cl_1$ , whereas it is stalled in  $cl_2$ 's view, *i.e.*,  $L_{A,t}^{cl_2} = \emptyset$  for all times  $t$ . In the meanwhile,  $\Pi_B$  emulates

<sup>4</sup> For instance, if  $\Pi_B$  is a blockchain protocol, the snapshots and their certificates will be included in the blocks by the block proposers (*cf.* Section 9 for more efficient implementations).

the behavior of a live blockchain towards a different client  $cl_2$ , whereas it is stalled in  $cl_1$ 's view, *i.e.*,  $L_{B,t}^{cl_1} = \emptyset$  for all times  $t$ . Since the triangular composition is conjectured to be live when either of the blockchains is live, both  $cl_1$  and  $cl_2$  output transactions based on their observations of the  $\Pi_A$  and  $\Pi_B$  ledgers respectively (as far as  $cl_1$  is concerned,  $\Pi_A$  *looks* live, and as far as  $cl_2$  is concerned,  $\Pi_B$  *looks* live). However, when these  $\Pi_A$  and  $\Pi_B$  ledgers are different and conflicting, this implies a safety violation even though both  $\Pi_A$  and  $\Pi_B$  are safe, *i.e.*,  $cl_1$  and  $cl_2$ 's  $\Pi_A$  ( $\Pi_B$ ) ledgers are consistent (as  $\emptyset$  is a prefix of every ledger).

Given the example above which shows the impossibility of a composition that is live if either chain is live and safe if both are safe, we relax the properties expected of a liveness-favoring composition in two ways: (i) the *triangular* composition of 3 blockchains ensures liveness if 2 of the 3 constituent chains are live, and safety if all chains are safe, under partial synchrony (Theorem 6), whereas (ii) the *parallel* composition of 2 blockchains, *under synchrony*, ensures liveness if either of the constituent chains is live, and safety if *both* chains are safe and live (Section 7.1, Theorem 15). Here, we focus on the triangular composition.

For inspiration towards a minimal triangular composition with these guarantees, we consider a setting, where the protocol participants are validators rather than blockchains. We observe that a natural analogue of a blockchain that is not live, but safe, is a validator with *omission faults*. Since the triangular composition for blockchains requires the safety of all constituent protocols for safety, its analogue for validators would tolerate only omission faults. Thus, our triangular composition is motivated by omission fault tolerant (OFT) consensus protocols [28, 13, 35]. Before presenting the composition, we briefly describe these OFT protocols for validators, which we extend to the blockchain setting.

#### 4.2.1 The OFT Protocol for Validators

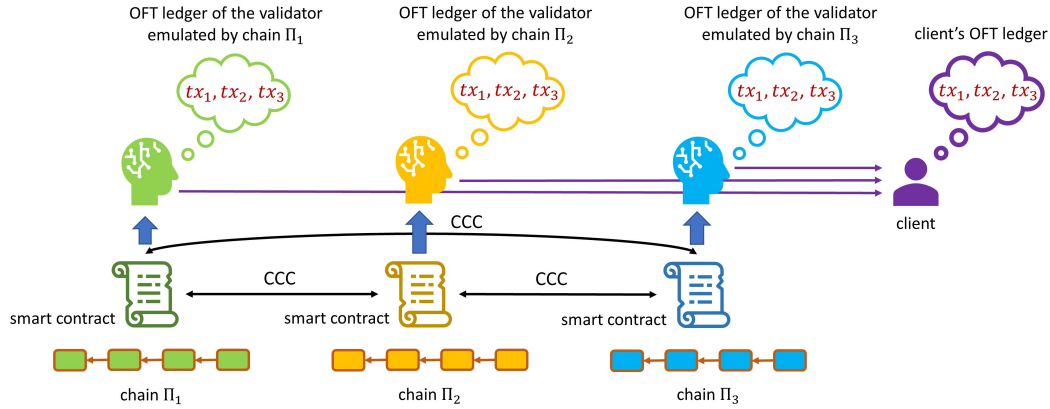
The OFT protocol is a leader-based blockchain protocol that generates certificates under a partially synchronous network. It is run by 3 validators mirroring the most basic triangular composition. It proceeds in epochs of fixed duration  $3\Delta$ . In a nutshell, it works as follows:

Each epoch  $v$  has a unique leader that proposes a block at the beginning of the epoch, *i.e.*, at time  $3\Delta v$ . Upon observing a proposal for epoch  $v$ , validators broadcast acknowledge messages for the proposed block at time  $3\Delta v + \Delta$ . Upon observing a *certificate* of 2 unique acknowledge messages from epoch  $v$  for the epoch's proposal, validators and clients finalize the proposed block and its prefix chain. If a validator does not observe a certificate of 2 acknowledge messages for an epoch  $v$  proposal by time  $3\Delta v + 2\Delta$ , it broadcasts a leader-down message for epoch  $v$ , where the message contains the block with the highest epoch number among the ones it previously voted for. Leader-down messages enable the leader of the next epoch to identify the correct block to propose on to preserve safety. A detailed protocol description is presented in [44, Appendix D].

#### 4.2.2 From OFT Protocol to the Triangular Composition

We next describe a triangular composition for 3 blockchains. It consists of 3 underlay blockchain protocols,  $\Pi_A$ ,  $\Pi_B$  and  $\Pi_C$ , run by validators and an overlay protocol, *i.e.*, the OFT protocol, run on top of these chains (Fig. 3). Each underlay protocol executes a smart contract that *emulates a validator* of the overlay OFT protocol (*cf.* [44, Appendix A] for a discussion on validator emulation). These emulated validators exchange messages via the CCC abstraction. There is a PKI that identifies on each underlay chain the 2 other chains emulating a validator (*e.g.*, by means of the public keys of the other chains' validators).





■ **Figure 3** Triangular composition. An overlay OFT protocol run on top of 3 underlay blockchains. A smart contract on each of the underlays emulates a validator of the OFT protocol and outputs a finalized OFT ledger. The client reads the underlay chains' ledgers and outputs the OFT ledger finalized by a majority of the emulated validators.

**Blockchains.** The triangular composition requires the underlay protocols to run general-purpose smart contracts and to proceed in epochs of fixed duration  $T$ . This is because the overlay OFT protocol requires each emulated validator to keep track of the time passed since it entered any given epoch. In general, it is impossible to emulate the validators of any overlay protocol secure under partial synchrony, if the underlay protocol has no means of keeping track of the real time (*cf.* [44, Appendix C]). We achieve this functionality by using underlay protocols that proceed in epochs of fixed time duration such as Tendermint [12] or Streamlet [17] (*cf.* [44, Appendix C]). However, our triangular composition can also be instantiated with optimistically responsive protocols (*cf.* Section 10 for more discussion).

Using epoch numbers recorded in the underlay blocks, the smart contract tracks the time passed since it entered any given epoch of the overlay protocol. If it entered some epoch  $v$  of the overlay protocol at time  $t$ , it moves to epoch  $v + 1$  at an underlay block of an underlay epoch  $3t_{\text{fin}}/T$ , where  $t_{\text{fin}}$  is the cross-chain communication latency. Here, the  $3\Delta$  epoch of the overlay OFT protocol is replaced by a  $3t_{\text{fin}}$  length epoch, since the messages exchanged by the *emulated* validators incur additional latency, including the finalization latency of the underlay chains besides network delay.

**Clients.** We next describe how clients of the triangular composition output a ledger for the overlay protocol using the ledgers of the underlay chains. Upon outputting a ledger  $L_A$ ,  $L_B$  and  $L_C$  for each underlay protocol  $\Pi_A$ ,  $\Pi_B$  and  $\Pi_C$ , at some time  $t$ , a client inspects the execution of the smart contracts as attested by these ledgers. If the execution trace on some ledger is invalid according to the rules of the smart contract, then the client discards the parts of the ledger starting with the first invalid transaction recorded on it, thus turning invalid execution into a liveness failure. For instance, sending a syntactically incorrect message is detectable by only inspecting the messages on a ledger. In contrast, sending two acknowledge messages for conflicting overlay blocks in the same overlay epoch might not be detected upon inspection, since these two messages can exist in separate execution traces emulating the same OFT validator, *i.e.*, on conflicting ledgers, observed by different clients (safety failure).

Once the client observes the execution traces for the validators emulated on valid portions of the ledgers  $L_A$ ,  $L_B$  and  $L_C$ , it identifies the blocks of the overlay protocol committed by each emulated validator. It accepts and outputs an overlay block and its prefix chain if it was committed by 2 or more emulated validators (as attested by the ledgers of 2 or more

underlay chains). If a client accepts and outputs an overlay block and its chain of height  $h$ , it never outputs a shorter overlay chain from that point on. If the client observes multiple conflicting  $L_A$ ,  $L_B$  and  $L_C$  ledgers when the safety of underlay chains is violated, it considers all these ledgers, and might output conflicting overlay blocks as a result. However, this is not a problem, as the proof of the next theorem shows that the client will continue to output blocks and retain liveness nevertheless.

The triangular composition satisfies the following:

► **Theorem 6.** *Consider the triangular composition  $\Pi_t$  instantiated with the protocols  $\Pi_A$ ,  $\Pi_B$  and  $\Pi_C$ , that proceed in epochs of fixed duration. Then, under partial synchrony,*

1.  $\Pi_t$  satisfies safety if all of  $\Pi_A$ ,  $\Pi_B$  and  $\Pi_C$  are safe.
2.  $\Pi_t$  satisfies liveness with constant latency after GST if 2 blockchains among  $\Pi_A$ ,  $\Pi_B$  and  $\Pi_C$  are live after GST with constant latency and proceed in epochs of fixed duration.
3.  $\Pi_t$  generates certificates if  $\Pi_A$ ,  $\Pi_B$  and  $\Pi_C$  do so.
4.  $\Pi_t$  proceeds in epochs of fixed duration.

Proof of Theorem 6 is given in [44, Appendix F.2]. Statements 1 and 2 are based on the proof of the original OFT protocol design for validators. Statements 3 and 4 are needed for further composability of the triangular composition with other serial and triangular composition (*cf.* the conditions on Theorem 5).

## 5 Circuits for Partial Synchrony

In this section, we construct overlay protocols via circuit composition achieving the security properties claimed by Theorem 1 and show optimality by proving Theorem 2. We also extend these results to all possible overlay protocols, akin to the generalization of security properties to quorum and fail-prone systems. Unlike the security claims for the protocol primitives, all of the proofs below are algebraic in nature.

### 5.1 Extended Serial and Triangular Constructions

We first build extended serial and triangular constructions as building block toward the full circuit composition.

► **Lemma 7.** *Let  $\Pi_i$ ,  $i \in [k]$  be  $k$  different blockchain protocols that generate certificates. Then, there exists a protocol, called the  $n$ -serial composition, satisfying the following properties:*

- *it is safe if at least one of  $\Pi_i$ ,  $i \in [k]$  is safe.*
- *it is live after GST with constant latency if all of  $\Pi_i$ ,  $i \in [k]$  are live after GST with constant latency.*
- *it generates certificates.*
- *it proceeds in epochs of fixed duration if all of  $\Pi_i$ ,  $i \in [k]$  do so.*

Lemma 7 follows directly from iteratively applying Theorem 5 on the protocols  $\Pi_i$ ,  $i \in [k]$ .

► **Lemma 8.** *For any integer  $f \geq 1$ , let  $\Pi_i$ ,  $i \in [2f + 1]$ , be  $2f + 1$  different blockchain protocols that proceed in epochs of fixed duration. Then, there exists a protocol, called the  $(2f + 1)$ -triangular composition, satisfying the following properties:*

- *it is safe if all  $\Pi_i$ ,  $i \in [2f + 1]$  are safe.*
- *it is live after GST with some constant latency if at least  $f + 1$  of  $\Pi_i$ ,  $i \in [2f + 1]$  are live after GST with some constant latency.*
- *it generates certificates if all  $\Pi_i$ ,  $i \in [2f + 1]$  generate certificates.*
- *it proceeds in epochs of fixed duration.*

Note that the original triangular composition with three protocols (referred to as the triangular composition) would be called a 3-triangular composition. Proof of Lemma 8 is presented in [44, Appendix F.6]. It constructs an  $f$ -triangular composition via strong induction on the number  $f$ . The inductive step uses the  $(f - 1)$ -triangular composition and the serial composition of Lemma 7, whereas the base case follows from the properties of the 3-triangular composition shown by Theorem 6.

## 5.2 Permutation Invariant Circuits for Partial Synchrony

We next prove Theorem 1, which characterizes the security of so-called permutation invariant overlay protocols. This class of protocols achieves safety (or liveness) as long as *any* subset of the underlay chains with a sufficient size provide the same set of security guarantees (*e.g.*, any subset of 5 out of 7 underlay chains is all safe and/or all live). In this sense, these protocols do not distinguish between the underlay chains.

Proof of Theorem 1 relies on Theorems 5 and 6. Recall that a tuple  $(k, s, l)$  was defined to be achievable if there exists an interchain protocol with  $k$  blockchains such that if at least  $s$  blockchains are safe, the protocol is safe, and if at least  $l$  blockchains are live, the protocol is live. By definition, a serial composition achieves the  $(2, 1, 2)$  point (Theorem 5), and a triangular composition achieves the  $(3, 3, 2)$  point (Theorem 6). Similarly, a  $(2f + 1)$ -triangular composition achieves the  $(2f + 1, 2f + 1, f + 1)$  point, and there exist such compositions for any  $f \geq 1$  by Lemma 8, which itself follows from Theorems 5 and 6.

For two blockchain protocols  $\Pi_A, \Pi_B$ , we denote by  $\Pi_A \oplus \Pi_B$  the serial compositions of these two blockchains as described in Section 4.1. Consider  $k$  protocols  $\Pi_1, \Pi_2, \dots, \Pi_k$ . We iteratively define  $\oplus_{i=1}^{j+1} \Pi_i = \left( \oplus_{i=1}^j \Pi_i \right) \oplus \Pi_{j+1}$  for  $j \in [k - 1]$ , and denote a protocol achieving the  $(k, 1, k)$  point by  $\pi^{(k, 1, k)}(\Pi_1, \dots, \Pi_k)$ . and denote a protocol achieving the  $(2f + 1, 2f + 1, f + 1)$  point by  $\pi^{(2f+1, 2f+1, f+1)}(\Pi_1, \dots, \Pi_{2f+1})$ .

Towards the final result, the following lemma shows that we can construct a protocol achieving  $(k + m, s, l + m)$  point using a protocol achieving  $(k, s, l)$ .

► **Lemma 9.** *For any integer  $m \geq 1$ , if  $(k, s, l)$  is achievable using  $\pi^{(k, s, l)}$  then,  $(k + m, s, l + m)$  is achievable using*

$$\pi^{(k+m, s, l+m)}(\{\Pi_i\}_{i=1}^{k+m}) = \bigoplus_{\substack{S \subseteq [k+m] \\ |S|=k}} \pi^{(k, s, l)}(\{\Pi_j\}_{j \in S}). \quad (1)$$

**Proof.** We first show that  $\pi^{(k+m, s, l+m)}$  defined in (1) is safe if at least  $s$  of the blockchains are safe. There exists a subset  $S_0 \subseteq [k + m]$  with  $|S_0| = k$  such that at least  $s$  blockchains among  $\{\Pi_j\}_{j \in S_0}$  are safe. This implies that  $\pi^{(k, s, l)}(\{\Pi_j\}_{j \in S_0})$  is safe. As we enumerate all subsets with size  $k$  in constructing  $\pi^{(k, s, l)}(\{\Pi_j\}_{j \in S_0})$ , by Lemma 7, we observe that  $\pi^{(k+m, s, l+m)}(\{\Pi_i\}_{i=1}^{k+m})$  is safe; as one of the blockchains in the serial composition, namely  $\pi^{(k, s, l)}(\{\Pi_j\}_{j \in S_0})$ , is safe.

On the other hand, suppose that at least  $l + m$  of the blockchains are live, which implies that at most  $k - l$  blockchains are not live. Therefore, for any arbitrary choice of size- $k$  subset  $\{\Pi_j\}_{j \in S}$  from  $\{\Pi_j\}_{j=1}^{k+m}$ , at most  $k - l$  blockchains are not live, or equivalently, at least  $l$  blockchains in  $\{\Pi_j\}_{j \in S}$  are live. This implies that  $\pi^{(k, s, l)}(\{\Pi_j\}_{j \in S})$  is live for all possible choices of subset  $S$  with size  $k$ . Then, by Lemma 7, we observe that  $\pi^{(k+m, s, l+m)}(\{\Pi_i\}_{i=1}^{k+m})$  is live; as all of the blockchains in the serial composition are live. ◀

Finally, we present the proof of Theorem 1.

**Proof of Theorem 1.** By Lemma 8, there are circuit compositions achieving the  $(2f+1, 2f+1, f+1)$  point given copies of *any* two protocols achieving the  $(2, 1, 2)$  and  $(3, 3, 2)$  points respectively, via recursive compositions of these protocols. This in turn implies that for any given integers  $k, s, l$  such that  $\lfloor k/2 \rfloor + 1 \leq l \leq n$  and  $s = 2(k-l) + 1$  (boundary of the achievable points on Fig. 1), the point  $(s, s, k-l+1) = (2(k-l)+1, 2(k-l)+1, k-l+1)$  is achievable. Since  $s = 2(k-l) + 1$  for these boundary points, we have  $l + s - k = k - l + 1$ . Therefore, by Lemma 9,  $(k, s, l) = (s + (k-s), s, k-l+1 + (k-s))$  is achievable. This implies that all points  $(k, s, l)$  such that  $\lfloor k/2 \rfloor + 1 \leq l \leq n$  and  $s \geq 2(k-l) + 1$  are achievable. ◀

### 5.3 General Circuits for Partial Synchrony

We next present a general characterization of the security of the overlay protocol under partial synchrony as a function of the safety and liveness of the underlay chains. Note that a general characterization would include protocols that are not permutation invariant, *i.e.*, providing different security guarantees when two subsets of the underlay chains achieve the same set of security properties. For example, a non-permutation invariant overlay protocol with three underlay chains  $\Pi_i, i \in [3]$ , might be live when the underlay chains  $\Pi_1$  and  $\Pi_2$  are both live, but it might not be so when  $\Pi_1$  and  $\Pi_3$  are live. For such protocols, our notation of  $(k, s, l)$  tuples fall short of characterizing the security properties. Therefore, we develop a new model for the security of overlay protocols synthesized from  $k$  underlay chains.

#### 5.3.1 The Model

As the security of a blockchain consists of safety and liveness, we use  $s, l \in \{0, 1\}^k$  to denote the list of predicates indicating which underlay chains are guaranteed to be safe and live. Specifically,  $s_i = 1$  if the  $i$ -th underlay chain is guaranteed to be safe, and  $s_i = 0$  if the  $i$ -th chain is not guaranteed to be safe. Then, the security properties of an overlay protocol  $\Pi$  can be characterized by two sets  $V^S, V^L \subseteq 2^{\{0,1\}^{2k}}$ , which express the dependence of  $\Pi$ 's safety and liveness on the safety and liveness of the underlay chains. Namely,  $(s, l) \in V^S$  if the overlay protocol  $\Pi$  is guaranteed to be safe when for all  $i$  such that  $s_i = 1$ , the  $i$ -th underlay chain is guaranteed safety, and for all  $j$  such that  $l_j = 1$ , the  $j$ -th chain is guaranteed liveness. Similarly,  $(s, l) \in V^L$  if the overlay protocol  $\Pi$  is guaranteed to be live when for all  $i$  such that  $s_i = 1$ , the  $i$ -th underlay chain is guaranteed safety, and for all  $j$  such that  $l_j = 1$ , the  $j$ -th chain is guaranteed liveness. We hereafter use the  $(V^S, V^L)$  characterization of security in lieu of the  $(k, s, l)$  tuples. Given these definitions, any set  $V^S, V^L$  for an overlay protocol satisfies

**(P1)** If  $v \in V, w \geq v$ , then  $w \in V$ .

A sequence  $v \in V$  is called an *extreme* element if there is no  $w \in V$  such that  $w < v$ . Let  $\text{exm}(V)$  be the set containing all extreme elements in  $V$ . By property (P1),  $\text{exm}(V)$  uniquely describes  $V$ , and any protocol  $\Pi$  can be characterized by the two sets  $E^S, E^L \subseteq 2^{\{0,1\}^{2k}}$  consisting of the extreme elements in  $V^S$  and  $V^L$ :  $E^S = \text{exm}(V^S)$  and  $E^L = \text{exm}(V^L)$ .

#### 5.3.2 The Result

Given the model above, the security properties achievable by overlay protocols under partial synchrony can be described as follows. For  $s \in \{0, 1\}^n$ , let us define  $\text{ind}(s) = \{i : s_i = 1\}$ .

► **Theorem 10.** For any tuple  $(E^S, E^L) \subseteq 2^{\{0,1\}^{2k}}$  such that

1. For all  $(s, l) \in E^L, (s', l') \in E^S$ , it holds that  $s = l' = 0^k$ .
  2. For all  $(0^k, l^1), (0^k, l^2) \in E^L, (s, 0^k) \in E^S$ , it holds that  $\text{ind}(l^1) \cap \text{ind}(l^2) \cap \text{ind}(s) \neq \emptyset$ .
- there exists an overlay protocol characterized by a tuple dominating  $(E^S, E^L)$ .

The proof is in [44, Appendix F.3] and inductively constructs the desired overlay protocol. Intuitively, Theorem 10 states that safety (liveness) of the overlay protocol depends only on the safety (liveness) of the underlay chains, and any two quorums of underlay chains required for the liveness of the overlay protocol must intersect at a chain whose safety is required for the safety of the overlay protocol. Note that Theorem 10 implies Theorem 1, as Theorem 10 characterizes security for all types of overlay protocols, including permutation-invariant ones analyzed by Theorem 1. We opted to present Theorem 1 first for ease of understanding.

## 6 The Converse for Partial Synchrony

### 6.1 The Converse Result for Partial Synchrony

We start with the converse result that applies to all overlay protocols under partial synchrony, showing the optimality of our security characterization in Theorem 10.

► **Theorem 11.** *Let  $\Pi$  be an overlay blockchain protocol. Let  $(s^i, l^i) \in \{0, 1\}^n$  for  $i \in [3]$  satisfy  $(s^1, l^1) \in V^L$ ,  $(s^2, l^2) \in V^L$ ,  $(s^3, l^3) \in V^S$ . Then, we have  $\text{ind}(l^1) \cap \text{ind}(l^2) \cap \text{ind}(s^3) \neq \emptyset$ .*

Note that the converse exactly matches the second clause of Theorem 10.

**Proof.** For contradiction, suppose  $\text{ind}(l^1) \cap \text{ind}(l^2) \cap \text{ind}(s^3) = \emptyset$ . Denote the  $k$  underlay blockchains by  $\Pi_1, \dots, \Pi_k$ . There are two clients  $\text{cl}_1, \text{cl}_2$ . Consider the following three worlds.

**World 1.** All blockchains are safe. The underlay chains  $\Pi_i$ ,  $i \in \text{ind}(l^1)$  are live, and the others are stalled. The adversary sets  $\text{GST} = 0$ . Suppose that  $\text{tx}_1$  is input to the protocol at time  $t = 0$ . As  $(s^1, l^1) \in V^L$ , the overlay blockchain is live. At time  $t_1 = t_{\text{fin}}$ , the client  $\text{cl}_1$  outputs  $\text{tx}_1$  as its interchain ledger:  $L_{t_1}^{\text{cl}_1} = [\text{tx}_1]$ .

**World 2.** All blockchains are safe. The underlay chains  $\Pi_i$ ,  $i \in \text{ind}(l^2)$  are live, and the others are stalled. The adversary sets  $\text{GST} = 0$ . Suppose that  $\text{tx}_2$  is input to the protocol at time  $t = 0$ . As  $(s^2, l^2) \in V^L$ , the overlay blockchain is live. At time  $t_2 = t_{\text{fin}}$ , the client  $\text{cl}_2$  outputs  $\text{tx}_2$  as its interchain ledger:  $L_{t_2}^{\text{cl}_2} = [\text{tx}_2]$ .

**World 3.** All blockchains are live. The underlay chains  $\Pi_i$ ,  $i \in \text{ind}(l^1) \cap \text{ind}(l^2)$  are not safe, and the others, including those in  $s^3$ , are safe. For simplicity, let  $Q = \text{ind}(l^1) \cap \text{ind}(l^2)$ . The adversary sets  $\text{GST} = 2t_{\text{fin}}$  and creates a network partition before  $\text{GST}$  such that client  $\text{cl}_1$  can only communicate with the validators in  $\Pi_i$ ,  $i \in \text{ind}(l^1)$ , and client  $\text{cl}_2$  can only communicate with the validators in  $\Pi_i$ ,  $i \in \text{ind}(l^2)$ . As a result, for client  $\text{cl}_1$ , the underlay chains  $\Pi_i$ ,  $i \notin \text{ind}(l^1)$  seem stalled until at least time  $2t_{\text{fin}}$ , and for client  $\text{cl}_2$ , the underlay chains  $\Pi_i$ ,  $i \notin \text{ind}(l^2)$  seem stalled until at least time  $2t_{\text{fin}}$ , and Suppose that  $\text{tx}_1, \text{tx}_2$  are input to the protocol at time  $t = 0$ . However, the adversary reveals  $\text{tx}_1$  only to the honest validators in  $\Pi_i$  for  $i \in \text{ind}(l^1)/Q$  and  $\text{tx}_2$  only to those in  $\Pi_i$  for  $i \in \text{ind}(l^2)/Q$ . Moreover, it delays any communication between the validators in  $\Pi_i$  for  $i \in \text{ind}(l^1)/Q$  and those in  $i \in \text{ind}(l^2)/Q$  until after  $\text{GST}$ .

As the chains  $\Pi_i$ ,  $i \in Q$  are not safe, they can simultaneously interact with  $\text{cl}_1$  and the chains  $\Pi_i$ ,  $i \in \text{ind}(l^1)/Q$  as in World 1 and with  $\text{cl}_2$  and the chains  $\Pi_i$ ,  $i \in \text{ind}(l^2)/Q$  as in World 2. To ensure this, the adversary delays any messages from the honest validators of the chains  $\Pi_i$ ,  $i \in Q$ , to  $\text{cl}_1$  and  $\text{cl}_2$ , except the certificates received by  $\text{cl}_1$  and  $\text{cl}_2$  in Worlds 1 and 2 respectively. As we assume  $\Pi_i$ ,  $i \in Q$ , are not safe, such certificates attesting to conflicting ledgers must exist for the chains  $\Pi_i$ ,  $i \in Q$ . Then, client  $\text{cl}_1$  cannot distinguish World 1 and

World 3 before  $2t_{\text{fin}}$ , which implies that  $L_{t_1}^{\text{cl}_1} = [\text{tx}_1]$ . Similarly, client  $\text{cl}_2$  cannot distinguish World 2 and World 3, which implies that  $L_{t_2}^{\text{cl}_2} = [\text{tx}_2]$ . However,  $L_{t_1}^{\text{cl}_1}$  and  $L_{t_2}^{\text{cl}_2}$  conflict with each other, which violates the safety of the overlay protocol. This is a contradiction; as all underlay chains are live, and those in  $s^3$  are safe.  $\blacktriangleleft$

## 6.2 The Converse Result for Permutation Invariant Protocols under Partial Synchrony

Before proving Theorem 2, we introduce a more comprehensive notation for permutation invariant overlay protocols to capture the fact that the safety (or liveness) of the overlay protocol can depend on *both* the safety and liveness of the underlay chains. Although Theorem 2 shows that the liveness of the underlay chains do not help achieve better safety properties for the overlay (and vice versa), we nevertheless need a notation that allows the *possibility* of such cross-dependence between safety and liveness to argue for the absence of this cross-dependence. Moreover, we will use the new notation for permutation invariant overlay protocols later to describe the achievable security guarantees under synchrony, where the safety of the overlay protocol *depend* on the liveness of the underlay chains.

### 6.2.1 The Model for Permutation Invariant Overlay Protocols

Permutation invariance means that the overlay blockchain treats the underlays identically.

► **Definition 12** (Permutation Invariance). *We say a protocol  $\Pi$  is permutation invariant if the sets  $V^S$  and  $V^L$  both satisfy that*

(P2) *If  $v \in V$ , then  $\sigma(v) \in V$  for all permutations  $\sigma$ .*

*Here,  $v = (s, l)$ , and we define  $\sigma(v) = (\sigma(s), \sigma(l))$ , where  $\sigma(s), \sigma(l) \in \{0, 1\}^k$ ,  $\sigma(s)_i = s_{\sigma(i)}$ ,  $\sigma(l)_i = l_{\sigma(i)} \forall i \in [k]$ .*

By property (P2), we can create an equivalence relation “ $\sim$ ” over the sets in  $V$ . We say  $v \sim w$ , if there exists a permutation  $\sigma : [k] \rightarrow [k]$  such that  $w = \sigma(v)$ . The relation “ $\sim$ ” defines a quotient set  $V/\sim$ , which is the set of equivalence classes in  $V$ . Given  $v = (s, l)$  and

$$c_s(v) := \#\{i : s_i = 1\}, c_l(v) := \#\{i : l_i = 1\}, c_{sl}(v) := \#\{i : s_i = l_i = 1\},$$

each equivalence class  $\{\sigma(v) | \sigma : [k] \rightarrow [k] \text{ is a permutation}\}$  is uniquely represented by a tuple  $(c_s(v), c_l(v), c_{sl}(v)) \in \mathbb{N}^3$ . As the set  $\text{exm}(V)$  (for either  $V^S$  or  $V^L$ ) containing all extreme elements also satisfies (P2), we can also partition  $\text{exm}(V)/\sim$  into equivalence classes, each represented by a tuple  $(n_s, n_l, n_{sl}) \in \mathbb{N}^3$ . Therefore, given property (P1), the set  $V$  can be represented by a set of tuples  $P = \{(n_s^{(i)}, n_l^{(i)}, n_{sl}^{(i)}) | i \in \mathbb{N}\}$ .

Finally, any permutation invariant protocol  $\Pi$  can be characterized by two sets  $P^S, P^L \in 2^{\mathbb{N}^3}$ , representing  $V^S$  and  $V^L$  respectively and interpreted as follows: For any  $(n_s, n_l, n_{sl}) \in P^S$  and  $(m_s, m_l, m_{sl}) \in P^L$ , we have

- $\Pi$  is safe if at least  $n_s$  blockchains are safe,  $n_l$  blockchains are live, and  $n_{sl}$  blockchains are safe and live.
- $\Pi$  is live if at least  $m_s$  blockchains are safe,  $m_l$  blockchains are live, and  $m_{sl}$  blockchains are safe and live.

Let  $V(P) := \{v | c_s(v) \geq n_s, c_l(v) \geq n_l, c_{sl}(v) \geq n_{sl}, (n_s, n_l, n_{sl}) \in P\}$ .

For two set pairs  $(P^S, P^L)$  and  $(\tilde{P}^S, \tilde{P}^L)$  characterizing permutation invariant overlay protocols, we define the partial order  $(P^S, P^L) \succeq (\tilde{P}^S, \tilde{P}^L)$  to mean that  $V(P^S) \supseteq V(\tilde{P}^S)$  and  $V(P^L) \supseteq V(\tilde{P}^L)$ . For  $v \in \{0, 1\}^k$ , let us define  $\text{ind}(v) = \{i : s_i = 1\}$ .

► **Lemma 13.**  $(P^S, P^L) \succeq (\tilde{P}^S, \tilde{P}^L)$  if and only if for any  $\tilde{p}_1 \in \tilde{P}^S, \tilde{p}_2 \in \tilde{P}^L$ , there exists  $p_1 \in P^S$  and  $p_2 \in P^L$  such that  $p_1 \leq \tilde{p}_1$  and  $p_2 \leq \tilde{p}_2$ .

**Proof.** It is sufficient to show that  $V(P) \supseteq V(\tilde{P})$  if and only if for any  $\tilde{p} \in \tilde{P}$ , there exists  $p \in P$  such that  $p \leq \tilde{p}$ . Suppose that we have  $V(P) \supseteq V(\tilde{P})$ . For any  $\tilde{p} = (\tilde{n}_s, \tilde{n}_l, \tilde{n}_{sl}) \in \tilde{P}$ , consider  $\tilde{v} \in \{0, 1\}^{2k}$  such that  $c_s(\tilde{v}) = \tilde{n}_s, c_l(\tilde{v}) = \tilde{n}_l, c_{sl}(\tilde{v}) = \tilde{n}_{sl}$ . Then,  $\tilde{v} \in V(\tilde{P})$ . There exists an extreme element  $v \in V$  such that  $v \leq \tilde{v}$ . Defining  $p = (c_s(v), c_l(v), c_{sl}(v))$ , we can conclude that  $p \leq \tilde{p}$ . Suppose that for any  $\tilde{p} \in \tilde{P}$ , there exists  $p \in P$  such that  $p \leq \tilde{p}$ . For any  $\tilde{v} \in V(\tilde{P})$ , there exists  $\tilde{p} = (\tilde{n}_s, \tilde{n}_l, \tilde{n}_{sl}) \in \tilde{P}$  such that  $c_s(\tilde{v}) = \tilde{n}_s, c_l(\tilde{v}) = \tilde{n}_l, c_{sl}(\tilde{v}) = \tilde{n}_{sl}$ . Let  $p \in P$  such that  $p \leq \tilde{p}$ . From the definition of  $V(P)$ , we have  $\tilde{v} \in V(P)$ . ◀

## 6.2.2 The Result

The converse result for permutation invariant overlay protocols under partial synchrony, Theorem 2, follows as a corollary of Theorem 11. It shows the optimality of our security characterization in Theorem 1. Its proof is presented in [44, Appendix F.4].

► **Theorem 14 (Theorem 2).** *Let  $\Pi$  be a permutation invariant overlay blockchain protocol characterized by  $(P^S, P^L)$ . Consider the tuples  $(m_s, m_l, m_{sl}) \in P^L, (n_s, n_l, n_{sl}) \in P^S$ . Then, it holds that  $n_s \geq 2(k - m_l) + 1$  and  $m_l > k/2$ .*

## 7 Circuits for Synchrony

In this section, we construct overlay protocols via circuit composition achieving the security properties claimed by Theorem 3, and show their optimality. As the properties achievable under synchrony are stronger than those achievable under partial synchrony, to bridge the gap between partial synchrony and synchrony, we first introduce the parallel composition as a new protocol primitive in addition to the serial and triangular compositions (*cf.* Section 4.2 for a discussion of the triangular and parallel compositions). We then state the security result for general overlay protocols using the model in Section 5.3.1. Equipped with the model in Section 6.2.1, we subsequently show the security properties claimed for permutation invariant overlay protocols by Theorem 3 as a corollary of the general result. We end with a proof of optimality for both results.

### 7.1 Parallel Composition

We now describe the parallel composition with two underlay chains,  $\Pi_A$  and  $\Pi_B$  (Alg. 2). Upon getting a transaction from the environment, every honest  $\Pi_A$  and  $\Pi_B$  validator broadcasts the transaction to every other validator.

Let  $L_{A,t}^{\text{cl}}, L_{B,t}^{\text{cl}}$  and  $L_{p,t}^{\text{cl}}$  denote the  $\Pi_A, \Pi_B$  ledgers and the ledger of the parallel overlay protocol in the view of a client  $\text{cl}$  at time  $t$ . Consider a client  $\text{cl}$  that has been online for at least  $t_{\text{fin}}$  time. It obtains the parallel ledger as a function of the  $\Pi_A$  and  $\Pi_B$  ledgers. For this purpose,  $\text{cl}$  first checks if every transaction in  $L_{A,t-t_{\text{fin}}}^{\text{cl}}$  appears in  $L_{B,t}^{\text{cl}}$ , and if every transaction in  $L_{B,t-t_{\text{fin}}}^{\text{cl}}$  appears within  $L_{A,t}^{\text{cl}}$  (*the interleaving condition*). If so, it *interleaves* the prefixes of the  $\Pi_A$  and  $\Pi_B$  ledgers to construct the  $\Pi_p$  ledger:

$$L_{p,t}^{\text{cl}} := \text{INTERLEAVE}(L_{A,t}^{\text{cl}}[:\ell], L_{B,t}^{\text{cl}}[:\ell]), \quad (2)$$

where  $\ell := \min(|L_{A,t-t_{\text{fin}}}^{\text{cl}}|, |L_{B,t-t_{\text{fin}}}^{\text{cl}}|)$ .  $\text{INTERLEAVE}$  function applied on equal size ledgers  $L_1, L_2$  outputs a ledger  $L_o$  such that  $L_o[2i-1] = L_1[i]$  and  $L_o[2i] = L_2[i]$  for all  $i \in [|L_1|]$ .

■ **Algorithm 2** The algorithm used by a client  $cl$ , online for at least  $t_{\text{fin}}$  time, to output the ledger  $L_{p,t}^{\text{cl}}$  of the parallel composition  $\Pi_p$  instantiated with two constituent blockchains  $\Pi_A$  and  $\Pi_B$  at some time  $t$ . The algorithm takes the ledgers  $L_{A,t-t_{\text{fin}}}^{\text{cl}}$ ,  $L_{A,t}^{\text{cl}}$ ,  $L_{B,t-t_{\text{fin}}}^{\text{cl}}$  and  $L_{B,t}^{\text{cl}}$  output by  $cl$  at time  $t$  and outputs the ledger  $L_{p,t}^{\text{cl}}$ . The function  $\text{INTERLEAVE}(L_1, L_2)$  with inputs of same length returns the interleaved ledger  $L_o$  such that  $L_o[2i-1] = L_1[i]$  and  $L_o[2i] = L_2[i]$  for all  $i \in [|L_1|]$ .

---

```

1: function OUTPUTCHAIN( $L_{A,t-t_{\text{fin}}}^{\text{cl}}, L_{A,t}^{\text{cl}}, L_{B,t-t_{\text{fin}}}^{\text{cl}}, L_{B,t}^{\text{cl}}$ )
2:    $\ell \leftarrow \min(|L_{A,t-t_{\text{fin}}}^{\text{cl}}|, |L_{B,t-t_{\text{fin}}}^{\text{cl}}|)$ 
3:   if  $L_{A,t-t_{\text{fin}}}^{\text{cl}} \subseteq L_{B,t-t_{\text{fin}}}^{\text{cl}} \wedge L_{B,t-t_{\text{fin}}}^{\text{cl}} \subseteq L_{A,t}^{\text{cl}}$  then
4:      $L_{p,t}^{\text{cl}} \leftarrow \text{INTERLEAVE}(L_{A,t}^{\text{cl}}[:\ell], L_{B,t}^{\text{cl}}[:\ell])$ 
5:   else
6:      $i^* \leftarrow \text{argmax}_{i \in \{A,B\}} |L_{i,t-t_{\text{fin}}}^{\text{cl}}|$ 
7:      $L_{p,t}^{\text{cl}} \leftarrow \text{INTERLEAVE}(L_{A,t}^{\text{cl}}[:\ell], L_{B,t}^{\text{cl}}[:\ell]) || L_{i^*,t}^{\text{cl}}[\ell:]$ 
8:   end if
9:   return  $L_{p,t}^{\text{cl}}$ 
10: end function

```

---

If the interleaving condition fails, then  $cl$  interleaves the prefixes of the two ledgers and outputs the remainder of the longer ledger: defining  $i^* = \text{argmax}_{i \in \{A,B\}} |L_{i,t-t_{\text{fin}}}^{\text{cl}}|$ , it sets

$$L_{p,t}^{\text{cl}} := \text{INTERLEAVE}(L_{A,t}^{\text{cl}}[:\ell], L_{B,t}^{\text{cl}}[:\ell]) || L_{i^*,t}^{\text{cl}}[\ell:]. \quad (3)$$

The parallel composition satisfies the security properties below:

► **Theorem 15.** *Consider the parallel composition  $\Pi_p$  instantiated with the blockchain protocols  $\Pi_A$  and  $\Pi_B$ . Then, under synchrony,*

1.  $\Pi_p$  satisfies safety if both  $\Pi_A$  and  $\Pi_B$  are safe and live.
2.  $\Pi_p$  satisfies liveness with constant latency if either  $\Pi_A$  or  $\Pi_B$  is live with constant latency.
3.  $\Pi_p$  generates certificates if both  $\Pi_A$  and  $\Pi_B$  do so.
4.  $\Pi_p$  proceeds in epochs of fixed duration if  $\Pi_A$  and  $\Pi_B$  do so.

Proof of Theorem 15 is presented in [44, Appendix F.5]. It shows that if both chains are safe and live, the interleaving condition is satisfied, ensuring the safety of the  $\Pi_p$  ledger. If either chain is live, then all transactions up to the length of the longer chain is output, ensuring the liveness of the  $\Pi_p$  ledger. Note that the parallel composition does not satisfy accountable safety despite satisfying safety. This is not too surprising since its safety requires both the safety and liveness of the underlay chains.

## 7.2 General Circuits for Synchrony

► **Theorem 16.** *For any tuple  $(E^S, E^L) \subseteq 2^{\{0,1\}^{2k}}$  such that*

1. *For all  $(s, l) \in E^L$  it holds that  $s = 0^k$ ,*
2. *For all  $(0^k, l^1), (0^k, l^2) \in E^L, (s, l) \in E^S$ , it holds that*
  - a. *either there are indices  $i \in \text{ind}(l^1)$  and  $j \in \text{ind}(l^2)$  such that  $(s_i, l_i, s_j, l_j) = (1, 1, 1, 1)$ ,*
  - b. *or  $\text{ind}(l^1) \cap \text{ind}(l^2) \cap \text{ind}(s) \neq \emptyset$ ,*

*there exists an overlay protocol characterized by a tuple dominating  $(E^S, E^L)$ .*

We present the proof of Theorem 16 in [44, Appendix F.8]. It shows achievability by constructing a circuit very similar to that constructed by Theorem 10. Intuitively, Theorem 16 states that for the safety of the overlay protocol, either any two quorums of underlay chains required for the liveness of the overlay protocol must both contain at least one safe and live chain (which can be different), or these quorums must intersect at a safe chain.



### 7.3 Permutation Invariant Circuits for Synchrony

Following lemma proves the achievability guarantees claimed by Theorem 3. It uses the notation of Section 6.2.1 and follows as a corollary of Theorem 16.

► **Theorem 17** (Theorem 3, Achievability). *If a protocol is characterized by  $(P^S, P^L)$  within*

$$\{\{(2(k - m_i) + 1, 0, 0), (0, 0, k - m_i + 1)\}, \{(0, m_i, 0)\} | k/2 < m_i \leq k\},$$

*then there exists a permutation invariant overlay protocol characterized by a tuple dominating  $(P^S, P^L)$  under synchrony.*

## 8 The Converse for Synchrony

We now show the optimality of our security characterization in Theorem 3.

### 8.1 The Converse Result for Synchrony

We start with the converse result that applies to all overlay protocols under synchrony, showing the optimality of our security characterization in Theorem 16.

► **Theorem 18.** *Let  $\Pi$  be an overlay blockchain protocol. Let  $(s^i, l^i) \in \{0, 1\}^n$  for  $i \in [3]$  satisfy  $(s^1, l^1) \in V^L, (s^2, l^2) \in V^L, (s^3, l^3) \in V^S$ . Then, it holds that*

1. *either there are indices  $i \in \text{ind}(l^1)$  and  $j \in \text{ind}(l^2)$  such that  $(s_i, l_i, s_j, l_j) = (1, 1, 1, 1)$ ,*
2. *or  $\text{ind}(l^1) \cap \text{ind}(l^2) \cap \text{ind}(s) \neq \emptyset$ .*

Proof of Theorem 18 is presented in [44, Appendix F.9], and relies on an indistinguishability argument between different worlds like the proof of Theorem 11. Note that the converse exactly matches the clause of Theorem 16.

Theorem 18 is reduced to Theorem 11 if the set of security functions are restricted to those, where safety of the overlay protocol depends only on the safety of the underlay chains, and the liveness of the overlay protocol depends only on the liveness of the underlay chains. This is consistent with [40, Theorem B.1], which proves that the accountable safety-liveness tradeoff under synchrony is the same as the safety-liveness tradeoff under partial synchrony.

### 8.2 The Converse Result for Permutation Invariant Protocols under Synchrony

The converse result for permutation invariant overlay protocols under synchrony as claimed in Theorem 3, follows as a corollary of Theorem 18. It shows the optimality of our security characterization in Theorem 3.

► **Theorem 19** (Theorem 3, Converse). *Let  $\Pi$  be a permutation invariant overlay blockchain protocol characterized by  $(P^S, P^L)$ . Consider the tuples  $(m_s, m_l, m_{sl}) \in P^L, (n_s, n_l, n_{sl}) \in P^S$ . Then, it holds that either  $n_{sl} \geq k - m_l + 1$  or  $n_s \geq 2(k - m_l) + 1$  and  $m_l > k/2$ .*

Theorem 19 follows as a corollary of Theorem 18, and its proof is in [44, Appendix F.10]. [44, Appendix E] summarizes all of the results in Sections 5, 6, 7 and 8 by identifying all pareto-optimal protocols under partial synchrony and synchrony using the language developed in Sections 5.3.1 and 6.2.1.

## 9 Efficiency

Our model of interchain consensus protocols in Section 3 allows the validators of the underlay blockchains to read the ledgers of the other underlays. For each validator, this implies a communication load proportional to the number of underlay chains, *i.e.*, low scalability. However, all of our compositions (serial, triangular and parallel) can be modified to retain their security properties when the validators merely run light clients of the other chains. For instance, the serial composition in Section 4.1 can be instantiated with *succinct timestamps* as in [42]; so that the constituent protocols receive and order timestamps of the blocks of the other protocols rather than snapshots of the whole ledger. Here, timestamps can even be made less frequent for more efficiency (albeit at the expense of latency). When the timestamps are implemented with binding hash functions, their ordering suffices to resolve forks on the other chains and ensure safety as long as any chain is safe.

The triangular construction in Section 4.2 requires the validators of each underlay chain to only follow a smart contract, dedicated to executing the OFT protocol, on the other chains to react to their OFT protocol messages. This again warrants at most a light client functionality. Finally, in the parallel construction of Section 7.1, the validators of the underlay blockchains need not communicate at all except broadcasting the circuit-composition related transactions. In turn, external observers (clients) are responsible for interleaving their ledgers. Therefore, all three composition methods, and by induction, our circuit constructions can work with underlay validators running light clients of each other's chains. Implementation of these constructions with light clients is left as future work.

## 10 Conclusion

In this work, we have analyzed the security of interchain consensus protocols under synchrony and partial synchrony. We next outline a few open questions and future directions implied by our work. As our serial composition requires the underlay chains to produce certificates and the protocols secure under the sleepy network model [36] (dynamic availability [27], unsized setting [30]) do not generate certificates [30], our results do not extend to underlay chains secure under the (synchronous) sleepy network model (no protocol can be secure under both partial synchrony and the sleepy network model [29]). It is thus an open question to design a serial composition for underlay chains that do not generate certificates.

Although we have instantiated the triangular composition with underlay chains that proceed in fixed time durations, the composition can also work with *optimistically responsive* protocols. These protocols achieve latency that is  $O(\delta)$ , where  $\delta$  is the real-time network delay, under optimistic conditions. They can keep track of time with the help of an oracle committee of so-called *time keepers* [41] that input the real time into the protocol. Another alternative that does not require any trust in oracles is for the smart contracts on the underlay chains to adaptively estimate time. For instance, if the contracts notice that the overlay protocol has not made progress while the underlay protocols have, it can slow down the underlay protocols. It is future work to formalize the details of these solutions.

Our recursive compositions of circuits could require an underlay blockchain to appear in exponentially many sub-circuits. Our goal in this work was to show the achievability of the properties proven for the interchain consensus protocols. For small numbers of underlay chains, our results coupled with the optimizations in Section 9 still yield practical constructions for the safety-favoring points. It is an open question to design more scalable interchain consensus protocols for all points.

---

**References**

---

- 1 Cosmos: The Internet of Blockchains. URL: <https://cosmos.network/>.
- 2 The Inter-Blockchain Communication protocol. Website. URL: <https://cosmos.network/ibc/>.
- 3 ICS ? : Recursive Tendermint, 2019. URL: <https://github.com/cosmos/ibc/issues/547>.
- 4 cosmos/relayer: An IBC relayer for IBC-Go. Website, 2023. URL: <https://github.com/cosmos/relayer>.
- 5 Mesh security, 2023. URL: <https://github.com/osmosis-labs/mesh-security>.
- 6 Mesh security. Youtube, 2023. URL: [https://www.youtube.com/watch?v=GjX4ejD\\_cRA&t=4670s](https://www.youtube.com/watch?v=GjX4ejD_cRA&t=4670s).
- 7 Polygon 2.0: Protocol Architecture, 2023. URL: <https://polygon.technology/blog/polygon-2-0-protocol-vision-and-architecture>.
- 8 Stacks - A Bitcoin Layer for Smart Contracts, DeFi, NFTs, and Apps, 2023. URL: <https://www.stacks.co>.
- 9 Ittai Abraham and Gilad Stern. Information Theoretic HotStuff. In *OPODIS*, volume 184 of *LIPICs*, pages 11:1–11:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020.
- 10 Sarah Azouvi and Marko Vukolic. Pikachu: Securing pos blockchains from long-range attacks by checkpointing into bitcoin pow using taproot. In *ConsensusDay@CCS*, pages 53–65. ACM, 2022.
- 11 Fabrice Benhamouda, Craig Gentry, Sergey Gorbunov, Shai Halevi, Hugo Krawczyk, Chengyu Lin, Tal Rabin, and Leonid Reyzin. Can a public blockchain keep a secret? In *TCC (1)*, volume 12550 of *Lecture Notes in Computer Science*, pages 260–290. Springer, 2020.
- 12 Ethan Buchman, Jae Kwon, and Zarko Milosevic. The latest gossip on BFT consensus. *arXiv:1807.04938*, 2018. URL: <https://arxiv.org/abs/1807.04938>.
- 13 Navin Budhiraja, Keith Marzullo, Fred B. Schneider, and Sam Toueg. Optimal primary-backup protocols. In *WDAG*, volume 647 of *Lecture Notes in Computer Science*, pages 362–378. Springer, 1992.
- 14 Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget. *arXiv:1710.09437*, 2019. URL: <https://arxiv.org/abs/1710.09437>.
- 15 Christian Cachin, Klaus Kursawe, and Victor Shoup. Random oracles in constantinople: Practical asynchronous byzantine agreement using cryptography. *J. Cryptol.*, 18(3):219–246, 2005.
- 16 Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *OSDI*, pages 173–186. USENIX Association, 1999.
- 17 Benjamin Y. Chan and Elaine Shi. Streamlet: Textbook streamlined blockchains. In *AFT*, pages 1–11. ACM, 2020.
- 18 Cynthia Dwork, Nancy A. Lynch, and Larry J. Stockmeyer. Consensus in the presence of partial synchrony. *J. ACM*, 35(2):288–323, 1988.
- 19 Michael J. Fischer, Nancy A. Lynch, and Mike Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, 1985.
- 20 Matthias Fitzi, Peter Gazi, Aggelos Kiayias, and Alexander Russell. Ledger combiners for fast settlement. In *TCC (1)*, volume 12550 of *Lecture Notes in Computer Science*, pages 322–352. Springer, 2020.
- 21 Bela Gipp, Norman Meuschke, and Andre Gernandt. Decentralized trusted timestamping using the crypto currency bitcoin. In *Proceedings of the iConference 2015*, 2015.
- 22 Danny Harnik, Joe Kilian, Moni Naor, Omer Reingold, and Alon Rosen. On robust combiners for oblivious transfer and other primitives. In *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 96–113. Springer, 2005.
- 23 Thomas Hepp, Patrick Wortner, Alexander Schönhals, and Bela Gipp. Securing physical assets on the blockchain: Linking a novel object identification concept with distributed ledgers. In *CRYBLOCK@MobiSys*, pages 60–65. ACM, 2018.

- 24 Amir Herzberg. On tolerant cryptographic constructions. In *CT-RSA*, volume 3376 of *Lecture Notes in Computer Science*, pages 172–190. Springer, 2005.
- 25 Manuel Huber, Julian Horsch, and Sascha Wessel. Protecting suspended devices from memory attacks. In *EUROSEC*, pages 10:1–10:6. ACM, 2017.
- 26 Dimitris Karakostas and Aggelos Kiayias. Securing proof-of-work ledgers via checkpointing. In *IEEE ICBC*, pages 1–5. IEEE, 2021.
- 27 Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *CRYPTO (1)*, volume 10401 of *Lecture Notes in Computer Science*, pages 357–388. Springer, 2017.
- 28 Leslie Lamport. The part-time parliament. In *Concurrency: the Works of Leslie Lamport*, pages 277–317. ACM, 2019.
- 29 Andrew Lewis-Pye and Tim Roughgarden. Resource pools and the cap theorem. *arXiv:2006.10698*, 2020. URL: <https://arxiv.org/abs/2006.10698>.
- 30 Andrew Lewis-Pye and Tim Roughgarden. How does blockchain security dictate blockchain implementation? In *CCS*, pages 1006–1019. ACM, 2021.
- 31 Sinisa Matetic, Mansoor Ahmed, Kari Kostianen, Aritra Dhar, David M. Sommer, Arthur Gervais, Ari Juels, and Srdjan Capkun. ROTE: rollback protection for trusted execution. In *USENIX Security Symposium*, pages 1289–1306. USENIX Association, 2017.
- 32 Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. URL: <https://bitcoin.org/bitcoin.pdf>.
- 33 Joachim Neu, Ertem Nusret Tas, and David Tse. Snap-and-Chat protocols: System aspects. *arXiv:2010.10447*, 2020. URL: <https://arxiv.org/abs/2010.10447>.
- 34 Joachim Neu, Ertem Nusret Tas, and David Tse. Ebb-and-flow protocols: A resolution of the availability-finality dilemma. In *IEEE Symposium on Security and Privacy*, pages 446–465. IEEE, 2021.
- 35 Brian M. Oki and Barbara Liskov. Viewstamped replication: A general primary copy. In *PODC*, pages 8–17. ACM, 1988.
- 36 Rafael Pass and Elaine Shi. The sleepy model of consensus. In *ASIACRYPT (2)*, volume 10625 of *Lecture Notes in Computer Science*, pages 380–409. Springer, 2017.
- 37 Rafael Pass and Elaine Shi. Thunderella: Blockchains with optimistic instant confirmation. In *EUROCRYPT (2)*, volume 10821 of *Lecture Notes in Computer Science*, pages 3–33. Springer, 2018.
- 38 Michael O. Rabin. Randomized byzantine generals. In *FOCS*, pages 403–409. IEEE Computer Society, 1983.
- 39 Claude E Shannon. A symbolic analysis of relay and switching circuits. *Electrical Engineering*, 57(12):713–723, 1938.
- 40 Peiyao Sheng, Gerui Wang, Kartik Nayak, Sreeram Kannan, and Pramod Viswanath. BFT protocol forensics. In *CCS*, pages 1722–1743. ACM, 2021.
- 41 Peiyao Sheng, Xuechao Wang, Sreeram Kannan, Kartik Nayak, and Pramod Viswanath. Trustboost: Boosting trust among interoperable blockchains. In *CCS*, pages 1571–1584. ACM, 2023.
- 42 Ertem Nusret Tas, Runchao Han, David Tse, and Mingchao Yu. Interchain timestamping for mesh security. In *CCS*, pages 1585–1599. ACM, 2023.
- 43 Ertem Nusret Tas, David Tse, Fangyu Gai, Sreeram Kannan, Mohammad Ali Maddah-Ali, and Fisher Yu. Bitcoin-enhanced proof-of-stake security: Possibilities and impossibilities. In *SP*, pages 126–145. IEEE, 2023.
- 44 Ertem Nusret Tas, David Tse, and Yifei Wang. A circuit approach to constructing blockchains on blockchains. *arXiv:2402.00220*, 2024. URL: <https://arxiv.org/abs/2402.00220>.
- 45 Wenbin Wang, Chaoshu Yang, Runyu Zhang, Shun Nie, Xianzhang Chen, and Duo Liu. Themis: Malicious wear detection and defense for persistent memory file systems. In *ICPADS*, pages 140–147. IEEE, 2020.

- 46 Maofan Yin, Dahlia Malkhi, Michael K. Reiter, Guy Golan-Gueta, and Ittai Abraham. Hotstuff: BFT consensus with linearity and responsiveness. In *PODC*, pages 347–356. ACM, 2019.
- 47 Alexei Zamyatin, Mustafa Al-Bassam, Dionysis Zindros, Eleftherios Kokoris-Kogias, Pedro Moreno-Sanchez, Aggelos Kiayias, and William J. Knottenbelt. Sok: Communication across distributed ledgers. In *Financial Cryptography (2)*, volume 12675 of *Lecture Notes in Computer Science*, pages 3–36. Springer, 2021.
- 48 Dionysis Zindros, Apostolos Tzinas, and David Tse. Rollerblade: Replicated distributed protocol emulation on top of ledgers. *IACR Cryptol. ePrint Arch.*, page 210, 2024.



# Blockchain Space Tokenization

Aggelos Kiayias  

University of Edinburgh, UK  
IOG, London, UK

Elias Koutsoupias  

University of Oxford, UK

Philip Lazos  

Jump Trading, London, UK

Giorgos Panagiotakos  

IOG, Athens, Greece

---

## Abstract

Handling congestion in blockchain systems is a fundamental problem given that the security and decentralization objectives of such systems lead to designs that compromise on (horizontal) scalability (what sometimes is referred to as the “blockchain trilemma”). Motivated by this, we focus on the question whether it is possible to design a transaction inclusion policy for block producers that facilitates *fee* and *delay predictability* while being *incentive compatible* at the same time.

Reconciling these three properties is seemingly paradoxical given that the dominant approach to transaction processing is based on first-price auctions (e.g., as in Bitcoin) or dynamic adjustment of the minimum admissible fee (e.g. as in Ethereum EIP-1559) something that breaks fee predictability. At the same time, in fixed fee mechanisms (e.g., as in Cardano), fees are trivially predictable but are subject to relatively inexpensive bribing or denial of service attacks where transactions may be delayed indefinitely by a well funded attacker, hence breaking delay predictability.

In this work, we set out to address this problem by putting forward *blockchain space tokenization* (BST), namely a new capability of a blockchain system to tokenize its capacity for transactions and allocate it to interested users who are willing to pay ahead of time for the ability to post transactions regularly for a period of time. We analyze our system in the face of worst-case transaction-processing attacks by introducing a security game played between the mempool mechanism and an adversary. Leveraging this framework, we prove that BST offers predictable and asymptotically optimal delays, predictable fees, and is incentive compatible, thus answering the question posed in the affirmative.

**2012 ACM Subject Classification** Security and privacy → Distributed systems security

**Keywords and phrases** Blockchain protocols, Predictable Service, Transaction Fees

**Digital Object Identifier** 10.4230/LIPIcs.AFT.2024.9

**Related Version** *Full Version:* <https://eprint.iacr.org/2024/1154.pdf>

**Acknowledgements** We would like to thank the anonymous reviewers for their valuable comments.

## 1 Introduction

Blockchain systems have bounded throughput and as a result at times of congestion they can process only a portion of the transactions submitted. Combining this with the need to unambiguously serialize transactions in order to determine the state of the underlying ledger, it is imperative that a policy of transaction inclusion must be applied by the protocol.



© Aggelos Kiayias, Elias Koutsoupias, Philip Lazos, and Giorgos Panagiotakos;  
licensed under Creative Commons License CC-BY 4.0

6th Conference on Advances in Financial Technologies (AFT 2024).

Editors: Rainer Böhme and Lucianna Kiffer; Article No. 9; pp. 9:1–9:20

Leibniz International Proceedings in Informatics



LIPIC Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 9:2 Blockchain Space Tokenization

There are three main approaches for such policy that have been implemented in popular blockchain systems: (i) prioritize based on transaction fees submitted (e.g. as in Bitcoin), (ii) introduce a fee threshold for transaction inclusion that is dynamically adjusted at times of congestion (e.g., as in Ethereum currently<sup>1</sup>), (iii) fix fees deterministically as a function of the transaction itself and prioritize strictly based on a FIFO policy, (e.g., as in Cardano).

Observe that the above three approaches suggest also three different service models. In terms of pricing, the first one is auction based, the second is posted price but with a dynamically adjusted price based on level of congestion and the last one is fixed price irrespective of demand. Regarding inclusion delay, the first two approaches offer (subject to consensus) instant inclusion, as long as the user is willing to pay a high enough fee, while in the third approach the inclusion delay can grow unboundedly depending on congestion<sup>2</sup>. It follows that in the first two approaches the inclusion delay is *predictable*, in the sense that it is known ahead of time (say 1 day earlier), while prices are unpredictable as they can go up in an unbounded manner at times of congestion and are only known one block before inclusion. On the converse, in the third approach the price is fixed and thus predictable (in the native currency of the system), while the inclusion delay is unpredictable heavily depending on congestion.

It is self-evident that service predictability is key to many applications, e.g. a company using a blockchain system would like to know ahead of time the inclusion delay expected as well as its cost to properly plan its operations. As a further example, “layer 2” protocols like lightning [22], set time bounds for the participants to challenge protocol states and failing to predict the transaction delay for participants to respond has dire security repercussions.

Furthermore, given that blockchains operate in a decentralized setting, providing *adequate incentives* for the system operators to actually follow the prescribed inclusion policy is key for its successful deployment. The first two approaches have been shown to be largely immune to collusion through off-chain agreements (aka off-chain proof [24]), i.e., the user trying to bribe his way into the system not being a profitable endeavor for both the user and the operator. However, the third one does not fare well in the face of bribes: A user can simply pay off-chain a higher fee to bypass the FIFO order of inclusion and guarantee shorter inclusion delay, with the system operator also increasing his revenue by accepting such a deal.

Predictable fees, predictable delay, and off-chain proofness, three seemingly contradicting goals, motivate the work of this paper:

Is it possible to design a transaction inclusion mechanism that is off-chain proof and offers fee and delay predictability at the same time?

Interestingly, prior work has often sidestepped this question. Some approaches assume blocks with unlimited size (or, equivalently, a limited number of submitted transactions) [10, 21], effectively making them immune to congestion. Others prioritize features like bidding, off-chain proofness, and incentive compatibility [24] forfeiting any concrete inclusion guarantees for transactions.

---

<sup>1</sup> Note that due to the presence of tips, the current approach of Ethereum also combines elements of approach (i).

<sup>2</sup> In fact, there is a specific price tag that an attacker has to pay in order to occupy the totality of blockchain processing capacity hence denying access to other users. In the case of Cardano this is in the order of < \$100 per minute, see <https://forum.cardano.org/t/cardano-network-vulnerable-to-20-minute-spam-attack/86422>.



**Our results.** In this work we focus on off-chain proof transaction fee mechanisms that offer predictable service guarantees. Our contributions can be summarized as follows.

- We fill a gap in the worst-case modeling of transaction waiting time by introducing a general *inclusion-delay game*. As mentioned, previous work [21, 10, 24] has focused on other challenges and has not dealt with this issue on a satisfactory level. In this *security game* the attacker interacts with a number of block producers. Following a cryptographic approach to modeling, our attacker drives the submission of transactions as well as the creation of adversarial and honest blocks. Block producers choose only what transactions to include in a block according to some policy. Given this strong adversarial setup, we are interested in the worst-case delay that an adversary can induce against a transaction measured in number of issued blocks. The advantage of our approach is that it obviates the need to explicitly model the consensus part of the system, while capturing all the pertinent elements needed for analysis, namely the *mempool mechanism* used by block producers to select transactions.

We consider a class of mechanisms that are associated with an abstract supply of “blockchain-access” tokens. This enables to describe mechanisms with *different* delay guarantees based on the value of the tokens used by a transaction (e.g., in the case of Bitcoin transactions that pay more fees will be given priority by the mempool mechanism).

- Armed with our model, we set out now to design a mempool mechanism with the desired properties. Blockchain space tokenization (BST) issues a number of “space tokens” that each one gives certain rights to a holder to post a transaction. *The policy of the BST mechanism is to include transactions based on a priority calculated by multiplying the token value by its age (measured in blocks since its last use). To reduce congestion, only transactions exceeding a dynamically adjusted priority threshold are eligible for inclusion in a block.* We prove that BST offers asymptotically optimal and predictable delays, predictable fees, and is also off-chain proof, thus answering our main research question in the *affirmative*; optimality here refers to the worst-case delay guaranteed by the mechanisms as a function of the relative amount of token value of the holder.
- We further substantiate the real-world applicability of the BST mechanism by (i) demonstrating through a set of simulations that the inclusion delays for a variety of token distributions and user activity levels match the optimal bounds, (ii) presenting a token allocation based on a sealed bid auction that enables interested users to bid and obtain the necessary tokens, while the whole blockchain system splits the space available for transaction in two separate, fenced parts: the tokenized space and the “spot space” that accepts transactions based on a posted price mechanism as in EIP-1559, and is also used to accommodate the auction, (iii) discussing how the mechanism can be easily instantiated in both UTXO (e.g., Bitcoin) and account based (e.g. Ethereum) ledgers.

**Related work.** In addition to the previously cited works, we are also drawing from transaction fee mechanism design. The closest connection (due to the threshold used) is the original EIP-1559 mechanism proposed in [4] and [24]. More broadly, [13] studied the effects of delays on user utilities and prices under Bitcoin, showing that individual miners cannot profitably affect the level of fees. Significant recent results include [6], which show (among other results) that in general, no transaction fee mechanism can satisfy user incentive compatibility, miner incentive compatibility and be off-chain agreement proof all at once. The stability of EIP-1559 has been studied through the lens of dynamical systems in a string of papers [23, 15, 16], which show that even though the prices can show chaotic behavior, the block sizes on average are indeed very close to the target. There have been many interesting proposals for updating

transaction fee mechanisms such as [14] with the Monopolistic Price, and Random Sampling Optimal Price (RSOP), the latter of which was initially proposed for digital goods in [11]. The Monopolistic Price mechanism is not always truthful from the users' side, but, as they show experimentally and [27] rigorously proves, it is approximately incentive compatible when demand is high. [3, 2] propose variants of “pay-forward” mechanisms, where fees do not directly go to the miner responsible for the next block but are distributed to others as well and [8] offer another variant of EIP-1559, which is proven to be more stable, by showing that the prices exhibit a martingale property for unchanging stochastic demand. Finally, [9] considers an online lens with non-myopic users that can specify a transaction deadline and [5] studies the problem from Bayesian mechanism design perspective.

A related line of work is that on scheduling computations in multi-threaded systems or data-centers [12]. There, scarce computational resources must be efficiently allocated to match demand of variable importance. A well-known mechanism in this area is lottery scheduling [26], where processes are each assigned a number of lottery tickets, and the scheduler samples the next process proportionally to its tickets. However, such an approach is not sufficient to guarantee off-chain proofness in our setting, as the scheduler can be “bribed” to include transactions arbitrarily. This should not come as a surprise, since off-chain proofness is not a target property for multi-threaded systems or data-centers, i.e., the scheduler is always trusted, thus disallowing direct use of the mechanisms developed in this area to the blockchain setting.

Finally, financial derivatives, such as options or futures, on transaction fees offer an alternative method to ensuring predictable costs in a system with variable prices, e.g., see [25]. These approaches are not directly comparable with our solution, since they additionally require the formation of a suitable market around the derivatives. On the other hand, our solution requires a hard fork to be implemented in major architectures, unlike derivatives which can be implemented in the form of a smart contract [19].

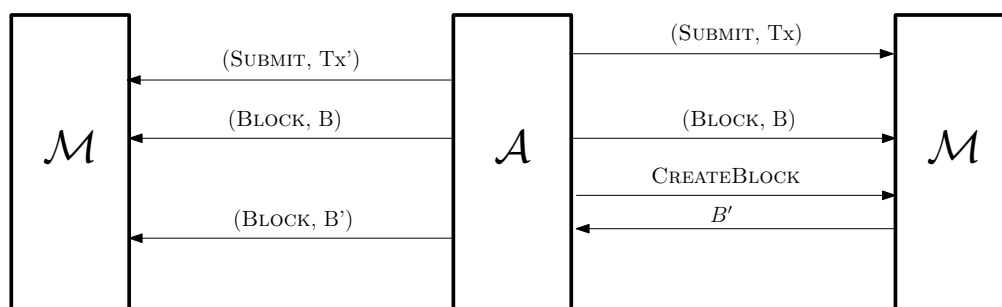
**Organization.** In Section 2, we introduce the inclusion-delay security game and define what it means for a blockchain to offer predictable service. The description and the theoretical guarantees provided by the blockchain space tokenization (BST) mechanism are described in Section 3. Section 4 discusses on how tokens in BST can be instantiated and distributed, as well as the benefits of operating BST together with a traditional spot-market mechanism such as EIP-1559. Section 5 focuses on BST deployment considerations, while the performance of the mechanism is evaluated through simulation in Section 6.

## 2 Predictable Service

Intuitively, a user is offered predictable service if for a cost paid ahead of time, the user is certain that a transaction produced at a later time will make it to the blockchain within some predetermined delay. From a security perspective, formalizing this notion requires an adequate security model. Previous modeling attempts have sidestepped this issue, mainly by making the assumption that blocks have infinite size [10, 20]. In this section, we fill this gap by providing an adequate model for analyzing service predictability.

### 2.1 Predictable delay

We start, by describing a simple cryptographic (worst-case) game to analyze the inclusion-delay of transactions under different mempool mechanisms. Worst-case delay is key to ensuring predictability as it provides a *known* upper bound on blockchain inclusion.



■ **Figure 1** An overview of the messages exchanged between different parties in the inclusion-delay game. Note, that mempool instances always interact through  $\mathcal{A}$ .

The *inclusion-delay* game  $G_{\mathcal{A}, \mathcal{M}}^{k, \mu}$  is played between an adversary  $\mathcal{A}$  and a number<sup>3</sup> of mempools running mechanism  $\mathcal{M}$ . The game centers around adversary  $\mathcal{A}$  submitting transactions to mempools and instructing them to create blocks.  $\mathcal{A}$ 's objective is to maximize the time it takes for a specific transaction to be included in a block.

In detail,  $\mathcal{A}$  has the following actions available:

- Submit a transaction to a mempool by sending a (SUBMIT, tx) message.
- Issue a new block of transactions (size  $k$ )<sup>4</sup> by sending a (BLOCK,  $B$ ) message to *all* mempools
- Instruct a mempool to create a new block by sending a CREATEBLOCK message. The mempool will then notify  $\mathcal{A}$  of its selection by responding with the new block  $B$ .  $\mathcal{A}$  is expected to share the new block with all other mempools instances by sending a (BLOCK,  $B$ ) message.<sup>5</sup>

We point to Figure 1 for an overview of the interactions in the game.

Our goal will be to design mechanisms that under suitable assumptions ensure that valid transactions appear in a block within a bounded number of new blocks created, counting from the time the transaction was submitted, no matter what  $\mathcal{A}$  may do. Note, that  $\mathcal{A}$  in our game is quite strong, as it fully controls transaction issuance, the delivery of messages, as well as the timing of block production. The only thing that is controlled by the mempools are the contents of the blocks they create.

Given that  $\mathcal{A}$  can always create empty blocks in the inclusion-delay game, we are going to bound the adversarial block production rate, to ensure that at least some of the blocks produced are honest. Namely, we will assume that in any sequence of blocks  $\rho$  created, the adversary issues at most  $\mu \cdot \rho$  of them, for  $\rho \in \mathbb{N}$  and some  $\mu \in [0, 1)$ ;  $\mu$  a parameter of our model. This property holds for most state-of-the-art blockchains, and has been extensively analyzed under the name of *chain-quality* [10].

Now, we turn our attention to a subtle issue that has to do with what kind of worst-case delay guarantees can be achieved by mempool mechanisms in our game. In general, worst-case delay in bounded throughput systems (as in our game) is lower-bounded by the rate of

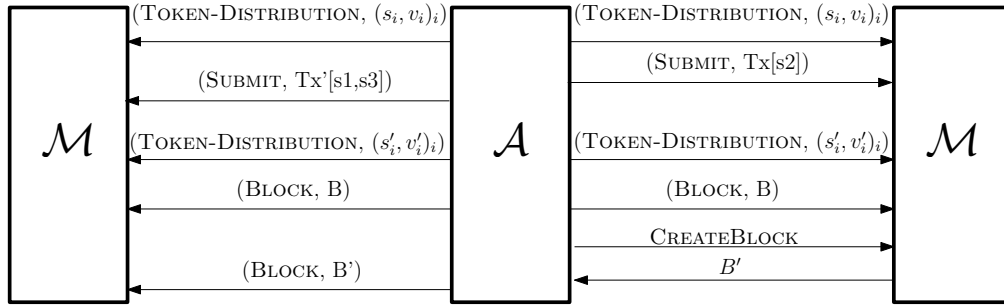
<sup>3</sup> W.l.o.g., we assume the existence of 2 mempool instances. If there was only a single instance of  $\mathcal{M}$ ,  $\mathcal{A}$  would be able to distinguish which blocks are honest and which adversarial, information that in permissionless blockchains is unavailable.

<sup>4</sup> Note, that if we let blocks have unbounded size the game becomes trivial;  $\mathcal{M}$  includes all transactions it receives in the new block produced. This is also our main difference with previous works, e.g., [10, 20], where blocks have unbounded size, thus not capturing the transaction-level Denial-of-Service attacks we address here.

<sup>5</sup> Having  $\mathcal{A}$  notify mempools about the creation of new blocks ensures that whether the creator of the block was a mempool or  $\mathcal{A}$  is not leaked.

incoming traffic to throughput. If our game does not provide any way to limit the amount of incoming traffic, then the adversary can launch a “sybil”-attack at the transaction level to significantly delay some of the transactions produced. Concretely, if  $\mathcal{A}$  wants to incur a delay of  $m$  blocks, it only has to submit  $k \cdot m + 1$  valid transactions to the mempools, and then request the creation of  $m$  blocks. Since at most  $k \cdot m$  transactions fit in  $m$  blocks, one of the transactions submitted will be delayed by  $m$  blocks.

To better reflect this situation within our model, and allow for shorter delays under certain preconditions, we introduce an abstract limited-supply “access token”<sup>6</sup> in our game; transaction creation will now be dependent/limited by token availability. In more detail, at the start of the execution  $\mathcal{A}$  is expected to initialize the token distribution any way it chooses, while retaining the ability to dynamically change it at any point. Both initialization and update of the token distribution is performed by sending to *all*<sup>7</sup> mempools a message of the form (TOKEN-DISTRIBUTION,  $(s_i, v_i)_i$ ), where  $s_i$  is the unique identifier of the  $i$ -th token in the list that has value  $v_i$ .<sup>8</sup> Issued transactions may use any number of tokens, with the more token value used linked to lower delay. We point to Figure 2 for the augmented game  $\tilde{G}_{\mathcal{A}, \mathcal{M}}^{k, \mu}$ .



■ **Figure 2** An overview of the augmented delay-inclusion game  $\tilde{G}_{\mathcal{A}, \mathcal{M}}^{k, \mu}$ . Transactions reference tokens (depicted in brackets here) to ensure lower worst-case delay.

Next, we turn our attention to formalizing transactions, blocks, and chains.

► **Definition 1 (Transactions).** A transaction  $tx$  specifies (i) the list of tokens  $(s_1, \dots, s_m)$  it uses, and (ii) its size denoted by  $\text{size}(tx)$ . The value of each token in the list is denoted by  $\text{val}(s_i)$ . The total token value of a transaction is defined to be  $\text{val}(tx) := \sum_{i \in [m]} \text{val}(s_i)$ .

► **Definition 2 (Blocks, Chains).** A block  $B := ((tx_i)_{i \in [m]})$  consists of a sequence of transactions. A chain  $\mathcal{C}$  consists of a sequence of blocks.

Typically transactions contain much more information than what we capture here. We choose to abstract away most of it as it is irrelevant for the problem at hand. Nevertheless, our model can be easily extended to describe complex transaction descriptions such as Ethereum’s account model or Cardano’s EUTxO.

<sup>6</sup> The (crypto-)currency stake distribution is a natural candidate token distribution in real-world blockchains. As we discuss later this is not the only option available.  
<sup>7</sup> For simplicity, in our game we avoid explicitly modeling disagreement on the token distribution. Standard techniques can be used to address this issue, e.g., the token distribution is only updated after the relevant information are confirmed by the underlying blockchain.  
<sup>8</sup> In reality, some kind of authentication mechanism, e.g., digital signatures, will be available to prove token-ownership. Such a mechanism is not needed in our model, since we assume that all transactions are produced by the adversary.

We next focus on defining what it means for a blockchain to be valid in our setting.

► **Definition 3** (Validity). *A block is valid if it contains transactions with size at most  $k$  and no two transactions reference the same token. A chain is valid if it contains only valid blocks.*

As before, our validity definition can be straightforwardly extended to account for the complex validity definitions found in real-world blockchains.

To guarantee provable inclusion guarantees we will require a transaction to (i) be well-distributed in the network (cf. the liveness condition in [10]), and (ii) not use the same tokens as some other concurrently submitted transaction. Obviously, we should not expect transactions not meeting these conditions to have provable inclusion guarantees.

► **Definition 4** (Good transactions). *A transaction  $tx$  submitted in the inclusion-delay game  $\tilde{G}_{\mathcal{A},\mathcal{M}}^k$  is good iff*

- *no other transaction  $tx'$  with overlapping token references is submitted until  $tx$  is included in a block;*
- *$tx$  is submitted to all mempool instances.*

Our definition of security, which we introduce next, is concerned exactly with the worst-case delay of such “good” transactions.

► **Definition 5** (Worst-case delay). *We say that a mempool mechanism  $\mathcal{M}$  has worst-case delay  $d$  iff for any adversary  $\mathcal{A}$ , it holds that any good transaction  $tx$  submitted in the inclusion-delay game  $\tilde{G}_{\mathcal{A},\mathcal{M}}^k$  appears in a block by the time  $d$  blocks<sup>9</sup> are generated, counting from the time  $tx$  was submitted to the last honest party.*

We note  $d$  in our analysis will be a function of certain transaction attributes, such as the transactions size and token-value. Moreover, w.l.o.g, in the rest of this work we assume that at any point of the inclusion delay game the *total token-value* is 1.

## 2.2 Predictable cost

Knowing that a transaction will make it to the blockchain within a predetermined amount of time is insufficient to claim predictable service, as the cost of the transaction may be unpredictable until the time it is included in the blockchain. Instead, to ensure full-fledged predictability, a user submitting a transaction to the mempool mechanism must know ahead of time what the total cost for this transaction will be. Note, that the cost does not have to be fixed (as in Cardano), it only has to be determined and possibly paid ahead of time compared to when the transaction is going to be submitted.<sup>10</sup> This leads us to the following natural definition of service predictability.

► **Definition 6** (Predictable service). *A mempool mechanism  $\mathcal{M}$  offers predictable service to some user with predictability parameter  $t$ , if the users knows that after time  $t$  it can issue a transaction with known delay and cost.*

<sup>9</sup> For simplicity, here we count delay in blocks to avoid introducing time explicitly. In principle, our definition can be adapted to count delay in time units, by introducing an adequate liveness condition in the inclusion-delay game. Such conditions are known to hold for blockchain protocols, e.g, see [10, 20].

<sup>10</sup> Similar guarantees in finance are provided by a forward market where the price of a commodity is locked-in a lot earlier than when the commodity is going to be delivered.

As argued earlier, Bitcoin and Ethereum fail to offer predictable service to *any user*, as a surge in demand just after submitting a transaction does not allow for predicting the cost of the transaction—the price may keep increasing until the transaction is no longer a valid candidate for inclusion.

In the case of Cardano things are a bit different. Transaction costs are fixed, and thus predictable. Moreover, due to the mempools processing transactions in a FIFO order, and the fact that stake is limited and thus only a limited number of valid transactions can be generated at any given point in time, it follows that delay is bounded in the worst-case and thus also predictable. However, although Cardano provides predictable service, albeit with a very large delay, it does not provide good incentives for block producers to follow the transaction inclusion policy.

Specifically, an urgent user attempting to “jump the queue” and include its high value transaction in the blockchain fast, can simply bribe operators to include its transaction first, potentially making the delay other users experience arbitrarily large. This is possible since the mempool operator has full control of the contents of the block. Resistance to such attacks, known as off-chain proofness (OCP), is explored in [24]. OCP is concerned with collusion agreements between users and operators trying to maximize their joint utility. A mechanism is OCP if for every set of off-chain agreements, there is an equally good “on-chain” scenario. Thus, even if a user pays off-chain a mempool producer for guaranteed inclusion, neither party should gain additional utility.

Given the shortcomings these mechanisms face, in the next section we present a new mechanism that manages at the same time to achieve service predictability and be off-chain proof.

### 3 Blockchain Space Tokenization

In this section, we describe and analyze the *block space tokenization* (BST) mechanism  $\mathcal{M}_{\text{bst}}$ , a deterministic mempool mechanism that offers predictable service while properly incentivizing correct behavior of both users and mempool operators. The mechanism centers around the concept of *blockchain space tokens* that give their owners the right to post transactions in the blockchain at a certain rate. The main idea is to prioritize inclusion of transactions using these tokens based only on *publicly verifiable information*, such as:

- the value of the tokens;
- the age of the tokens, i.e, the block height at which a token used by the target transaction was last used;
- the size of the transaction,

and thus make it easier to achieve incentive compatibility by enforcing and *checking* correct behavior.

We proceed to first give a detailed description of the core mechanism as well as analyze its worst-case delay in the honest/adversarial model. Then, we provide a modification of the algorithm that makes it off-chain proof while at the same time arguing that the resulting mechanism retains similar worst-case delay guarantees. Finally, we argue that if tokens are obtained by users ahead of time, the mechanism indeed offers predictable service.

#### 3.1 Mechanism description

The core component of our mechanism is the way the priority value of a transaction is calculated. Key to understanding this component is understanding the role of token age, denoted by  $age(s_i)$  for token  $s_i$ , which is equal to the number of blocks generated from the last time the token was used. Token age grows when a token is not used, while it is reset to zero every time it is used.

The first idea is to make the priority of a transaction proportional to the value of a token multiplied by its age. The intuition is that, for the same token age, a transaction with more token value should be prioritized over a transaction with less value; higher value should generally mean lower delays. For the same token value, a transaction consuming an “old” token should be prioritized over a transaction consuming a “young” token. This prevents the same tokens from monopolizing the usage of the chain. In general, we could use any monotone function of token value and age, but the product of value by age is the most natural one. When a transaction uses multiple tokens, its priority is proportional to the sum of the priorities of these tokens.

The second idea is to prioritize transactions at a rate inversely proportional to their size. This ensures that, for a fixed token value, a transaction twice the size is included with half the frequency. This maintains a constant throughput consumption rate per token value, regardless of transaction size, thus ensuring that obtaining a certain amount of space token value implies a certain space consumption rate.

Finally, we set an upper bound on priority that depends on token value density, the chain quality parameter, and block size. This is necessary to prevent attacks where an adversary stockpiles low-value tokens for an extended period before releasing them all at once, monopolizing space usage with high-priority transactions. The related transactions using these tokens would by then have maximum priority. Our carefully calibrated upper bound ensures high-value transactions remain prioritized, even if low-value tokens haven’t been used for a long period.

Concretely, mechanism  $\mathcal{M}_{\text{bst}}$  assigns *priority* values to transactions as follows:

$$\text{priority}_{\mathcal{C}}(\text{tx}) := \min \left\{ \frac{\sum_{s_i \in \text{tx}} \text{val}(s_i) \cdot \text{age}(s_i)}{\text{size}(\text{tx})}, \frac{4}{(1-\mu)k} \cdot \left( 1 + \frac{\text{val}(\text{tx})}{\text{size}(\text{tx})} \right) \right\},$$

where  $\mathcal{C}$  is the chain defined by the blocks created in the inclusion-delay game up to this moment,  $s_i$  is the  $i$ -th token used by the transaction,  $\text{age}(s_i)$  is the number of blocks in  $\mathcal{C}$  since  $s_i$  was last used by a transaction. Observe that for a transaction  $\text{tx}$  and chain  $\mathcal{C}$ ,  $\text{priority}_{\mathcal{C}}(\text{tx})$  is completely determined, making the priority value publicly verifiable.

Having defined a priority score,  $\mathcal{M}_{\text{bst}}$  simply fills new blocks with the transactions with the highest priority. Next, we focus on analyzing the worst-case delay guarantees of  $\mathcal{M}_{\text{bst}}$ .

## 3.2 Security analysis

Next, we show that  $\mathcal{M}_{\text{bst}}$  has optimal worst-case delay up to some constant terms, i.e. in the order of  $O\left(\frac{\text{size}(\text{tx})}{(1-\mu)\text{val}(\text{tx}) \cdot k}\right)$ .<sup>11</sup> The optimality claim is based on the fact that there can be at most  $1/\text{val}(\text{tx})$  transactions with token value  $\text{val}(\text{tx})$  and size  $\text{size}(\text{tx})$ , and thus it takes at least  $\frac{\text{size}(\text{tx})}{(1-\mu)\text{val}(\text{tx}) \cdot k}$  blocks to absorb them.

The main idea of the proof is the following: Firstly,  $\text{tx}$  will reach maximum priority after a sufficient number of new blocks is generated. This implies that in order for  $\text{tx}$  to not be included in the chain after this point in time, the adversary must fill any of the subsequent honest blocks with transactions of priority greater or equal to that of  $\text{tx}$ . Given that an  $(1-\mu)$  fraction of the blocks is going to be honestly generated due to our assumption, we

<sup>11</sup> While the constants provided by the theoretical analysis are not tight, later in Section 6 we show through simulation that our mechanism indeed achieves tightly optimal delays under normal operation. Nevertheless, the theoretical analysis is important as it establishes that the worst-case guarantees of the mechanism in an environment almost entirely controlled by the adversary remain on the same order as the optimal ones.

## 9:10 Blockchain Space Tokenization

show that it is impossible for the adversary to fill all of them with transactions other than  $\text{tx}$  that also have a matching priority score, and thus  $\text{tx}$  is necessarily included in a block in the predicted time.

► **Theorem 7.** *The worst-case delay  $d(\text{tx})$  of mechanism  $\mathcal{M}_{\text{bst}}$  is upper bounded by  $16 \cdot \frac{\text{size}(\text{tx})}{(1-\mu)\text{val}(\text{tx})k} + 2$ , when  $\text{size}(\text{tx}) < k/2$  and  $\text{val}(\text{tx}) < 1/2$ .*

**Proof.** Let  $\text{tx}$  be some good transaction in an execution of  $\tilde{G}_{\mathcal{A}, \mathcal{M}}^{k, \mu}$ , and let  $\epsilon := \text{val}(\text{tx})$ ,  $c := \text{size}(\text{tx})$ , and  $P := \frac{4}{(1-\mu)k}$ . Moreover, let  $u := d(\text{tx})/2$ , and note that if  $\text{tx}$  is not included in the blockchain after  $u$  new (honest or adversarial) blocks have been generated, then it has maximum priority, i.e.,  $\epsilon \cdot u/c \geq P(1 + \epsilon/c)$ . For the sake of contradiction assume that the theorem does not hold, and  $\text{tx}$  does not enter the chain after  $d(\text{tx})$  blocks are generated. We are going to show that such a scenario is impossible.

Let  $S'$  denote the set of (honest or adversarial) blocks generated starting  $u + 1$  blocks after the submission of  $\text{tx}$ , and up to the generation of  $2u$  blocks. Let  $S \subseteq S'$  denote the subset of honest blocks of  $S'$ . As argued earlier, during the generation of blocks in  $S$ ,  $\text{tx}$  has maximum priority equal to  $P \cdot f(\epsilon/c)$ , where  $f(x) := 1 + x$ , is a monotonically increasing function in  $x$ , and  $f(x) > 1$ , for any  $x > 0$ . We have assumed that  $\text{tx}$  is not included in these blocks, it thus follows that any block in  $S$  should contain transactions with priority greater than  $Pf(\epsilon/c)$ . Due to the monotonicity of  $f$ , this implies that the token-value density of any such transaction is at least  $\epsilon/c$ . Moreover, any block  $B$  in  $S$  must be at least  $k' := (k - c + 1)$  full, otherwise  $\text{tx}$  would be included. It follows, that the total token-value referenced in  $B$  is at least  $k' \cdot \epsilon/c$ , and that

$$\begin{aligned} \sum_{\text{tx}_i \in B} \text{priority}_C(\text{tx}_i) \cdot \text{size}(\text{tx}_i) &\geq \sum_{\text{tx}_i \in B} Pf(\epsilon/c) \cdot \text{size}(\text{tx}_i) \\ &\geq Pf(\epsilon/c) \cdot \sum_{\text{tx}_i \in B} \text{size}(\text{tx}_i) \\ &\geq Pf(\epsilon/c)k' \end{aligned} \tag{1}$$

The above inequality will be useful to determine the amount of priority adversarial transactions have to generate to fill all blocks in  $S$ .

Next, we focus on upper bounding the number of blocks in  $S$  the adversary can fill with transactions other than  $\text{tx}$ . W.l.o.g., we can assume that all adversarial blocks generated in  $S'$  are empty, and that  $\mathcal{A}$  includes transactions that reference all available tokens (except those that are referenced by  $\text{tx}$ ) in the first  $m$  blocks of  $S$  which are honest, for some optimally selected  $m$ . Note that it is optimal for  $\mathcal{A}$  to first reference *all* available tokens, as in this way it can maximize the amount of priority of transactions used to fill any remaining blocks; w.l.o.g, we assume that tokens are initially of infinite age. As a sanity check, note that just creating transactions referencing tokens once, is not sufficient to cover all blocks in  $S$ , as each block requires referencing  $\epsilon k'/c$  token-value, and thus a total of  $u(1 - \mu)$  blocks require referencing  $u(1 - \mu) \cdot \epsilon k'/c \geq 8k'/k > 4$  token-value, i.e., more than 1 which is the total amount of token-value.

Next, we provide an upper bound  $T$  on the sum of value-age products of the tokens in the first  $m$  honest blocks of  $S$  at the time the last honest block in  $S$  is generated. This will be important to argue that  $\mathcal{A}$  will not be able to create enough transactions with high enough priority to fill all honest blocks. We thus have:

$$T \leq \sum_{i=1}^m s_i(u - i) = u \sum_{i=1}^m s_i - \sum_{i=1}^m s_i i \leq u(1 - \epsilon) - \sum_{i=1}^m s_i i ,$$



where  $s_i$  is the total token-value in the  $i$ -th block of  $S$ . The quantity  $\sum_{i=1}^m s_i i$  is minimized when  $s_1$  gets its maximal value. As we have argued earlier, each honest block references at least  $\epsilon k'/c$  of token-value. Thus,  $s_1$  is at most  $1 - \epsilon - (m - 2 + \delta)\epsilon k'/c$ , where the  $m$ -th block references  $\delta\epsilon k'/c$  token-value for the first time in  $S'$ , for some  $\delta \in (0, 1]$ —the rest of the token-value necessary may come from previously used tokens. It follows that:

$$\begin{aligned} \sum_{i=1}^m s_i i &\geq (1 - \epsilon - (m - 2 + \delta)\epsilon k'/c) + \sum_{i=2}^{m-1} \epsilon k'/ci + m\delta\epsilon k'/c \\ &\geq 1 - \epsilon - \epsilon m k'/c + \epsilon k'/c \sum_{i=1}^m i - (1 - \delta)(m - 1)\epsilon k'/c \\ &\geq 1 - \epsilon - \epsilon m k'/c + \epsilon \frac{m(m+1)}{2} k'/c - (1 - \delta)(m - 1)\epsilon k'/c \end{aligned}$$

Putting everything together, we have that:

$$T \leq u(1 - \epsilon) - (1 - \epsilon - \frac{\epsilon k'}{2c}(2m - m(m+1) + 2(1 - \delta)(m - 1))) \quad (2)$$

By the chain quality assumption there are at least  $(1 - \mu)u$  honest blocks in  $S$ . We have already argued about how the first  $m$  blocks are filed. Due to Inequality 1, the sum of value-age products required to fill the rest of the blocks in  $S$  must be greater or equal than

$$((1 - \mu)u - m) \cdot k' Pf(\epsilon/c) + \delta k' Pf(\epsilon/c),$$

where the second term comes from the amount of priority required to fill the half empty  $m$ -th honest block. Moreover, our initial assumption about the behavior of  $\mathcal{A}$  implies that this quantity must be smaller than  $T$ . Hence, it must hold that:

$$\begin{aligned} &((1 - \mu)u - m) \cdot k' Pf(\epsilon/c) + \delta k' Pf(\epsilon/c) \\ &\leq u(1 - \epsilon) - (1 - \epsilon - \frac{\epsilon k'}{2c}(2m - m(m+1) + 2(1 - \delta)(m - 1))) \Rightarrow \\ u &\leq \frac{Pf(\frac{\epsilon}{c})k'(m - \delta) - (1 - \epsilon - \frac{\epsilon k'}{2c}(2m - m(m+1) + 2(1 - \delta)(m - 1)))}{(1 - \mu)Pf(\frac{\epsilon}{c})k' - 1 + \epsilon} \end{aligned}$$

It is easy to see that the derivative over  $m$  of the r.h.s. of the above inequality is equal to 0 when

$$m = cPf(\epsilon/c)/\epsilon + 3/2 - \delta \geq cPf(\epsilon/c)/\epsilon \geq c/(k'\epsilon),$$

where the last inequality follows from the facts that  $k' > k/2$ ,  $f(\epsilon/c) > 1$ . On the other hand,  $m$  must be less than  $1/\frac{\epsilon k'}{c} = c/(k'\epsilon)$ . Since the r.h.s. is a quadratic function of  $m$ , it follows that we can upper bound it (and thus upper bound  $u$ ) by setting  $m := c/(k'\epsilon)$ . Hence, we get:

$$u \leq \frac{cPf(\epsilon/c)/\epsilon + \epsilon + 1/2 - \delta - c/(2k'\epsilon) - (1 - \delta)\epsilon k'/c - \delta k' Pf(\epsilon/c)}{(1 - \mu)Pf(\epsilon/c)k' - 1 + \epsilon}$$

Replacing  $P$  by  $\frac{4}{(1-\mu)k}$ , for the denominator we get that:

$$(1 - \mu)Pf(\epsilon/c)k' \geq (1 - \mu) \frac{4}{(1 - \mu)k} f(\epsilon/c)k' \geq 2$$

which implies that:

$$\begin{aligned}
 u &\leq \frac{cPf(\epsilon/c)/\epsilon + \epsilon + 1/2 - \delta - c/(2k'\epsilon) - (1-\delta)\epsilon k'/c - \delta k'Pf(\epsilon/c)}{2-1+\epsilon} \\
 &\leq \frac{4cf(\epsilon/c)}{(1-\mu)k\epsilon} + \epsilon + 1/2 - \delta - \frac{c}{2k'\epsilon} - (1-\delta)\epsilon k'/c - \frac{4\delta k'f(\epsilon/c)}{(1-\mu)k} \\
 &\leq \frac{8c}{(1-\mu)k\epsilon} + \epsilon + 1/2 - \delta - \frac{c}{2k'\epsilon} - (1-\delta)\epsilon k'/c - \frac{4\delta k'f(\epsilon/c)}{(1-\mu)k} \\
 &\leq \frac{8c}{(1-\mu)k\epsilon} + 1 < u
 \end{aligned}$$

where the last inequality follows from the definition of  $u$ . Obviously, this is a contradiction and the theorem follows.  $\blacktriangleleft$

Next, we turn our attention to the off-chain proofness of the mechanism.

### 3.3 Making mechanism $\mathcal{M}_{\text{bst}}$ off-chain proof

Next, we provide a modification of  $\mathcal{M}_{\text{bst}}$  that ensures Off-Chain Proofness. Taking a leaf from Ethereum’s EIP-1559 pricing mechanism [24], we employ variable-sized blocks. Namely, we allow block size to exceed our target size (up to some amount) and use this information as a signal of increased or decreased demand, i.e, the relation of the size observed to the target size. The mechanism makes use of this information by proportionally increasing or decreasing a dynamic threshold that transaction-priority must exceed to be included in a block, in an effort to make demand equal to the target size. This change essentially limits the power mempool operators have in choosing the contents of blocks in a way that *cannot* be manipulated. Concretely, bribing a mempool operator to include your low-priority transaction will not help, since including the transaction into the block will make it invalid due to the threshold limitation.

In more detail, let  $\alpha$  be the target percentage we want blocks to be filled. We set the threshold  $\tau'$  of the next block after a chain  $\mathcal{C}$  to be:

$$\tau' = \tau \cdot \exp\left(\beta \cdot \frac{\sum_{\text{tx} \in S} \text{size}(\text{tx}) - \alpha \cdot L}{\alpha \cdot L}\right) \quad (3)$$

where  $L$  is the maximum size of a block,  $\alpha \cdot L$  is equal to  $k$ ,  $\beta > 0$  is a scaling factor, and  $\tau$  is the old threshold. This is similar to the Ethereum threshold update: there is some leeway to measure if blocks are too empty or too full. We are going to use a slightly different definition of priority than that of the previous section, namely:

$$\widehat{\text{priority}}_{\mathcal{C}}(\text{tx}) := \min\left\{\frac{\sum_{s_i \in \text{tx}} \text{val}(s_i) \cdot \text{age}(s_i)}{\text{size}(\text{tx})}, \frac{4}{(1-\mu)k} \cdot \left(1 + \frac{\text{val}(\text{tx})}{\text{size}(\text{tx})}\right) \cdot (1 + \rho)^{\frac{\phi - \ln(\text{size}(\text{tx})/\text{val}(\text{tx}))}{\ln(2)}}\right\}$$

where  $\rho$  and  $\phi$  are constants that will be defined later. Essentially, we have disproportionately increased the maximum priority value transactions can reach. By doing this we avoid attacks where the attacker by using maximum priority low-value transactions disproportionately increases the threshold value and “cheaply” excludes high priority transactions from entering the blockchain in the next block. We extend Definition 3 (Validity) to require that all transactions included in a block should have priority larger than  $\tau'$ , and denote the modified protocol by  $\mathcal{M}_{\text{bst}}^{\text{th}}$ .

Following the discussion about the incentive issues of Cardano in Section 2.2, note that  $\mathcal{M}_{\text{bst}}^{\text{th}}$  actually *is* off-chain proof.

► **Corollary 8.**  $\mathcal{M}_{\text{bst}}$  is off-chain proof iff the threshold is high enough so that the eligible pending transactions can fit into a single block. Formally, if the set  $E$  contains all transactions such that  $\widehat{\text{priority}}_c(\text{tx}) \geq \tau'$ ,  $\mathcal{M}_{\text{bst}}$  is off-chain proof iff  $\sum_{\text{tx} \in E} \text{size}(\text{tx}) \leq L$ .

The main idea of the proof<sup>12</sup> is that an appropriate threshold implies that the block producer can add all pending transactions to her block. Under normal circumstances the eligible transactions would be about  $a \cdot L$  in size. Any other transaction would be ineligible and cannot be added through an off-chain deal, no matter how valuable.

Although not formally studied, in the non-myopic case if the current block producer requested additional payment, there is enough slack so that the next block producer could include the previous transactions as well. However, during a sudden increase in demand the threshold might need a few blocks to adjust, leading to an excess of eligible, valuable transactions that could collude with block producers. This situation is similar to the “tipless” mechanism from [24], or to standard EIP-1559 but with off-chain proofness replaced by user incentive compatibility.

### 3.4 Worst-case delay of mechanism $\mathcal{M}_{\text{bst}}^{\text{th}}$

The modifications we employed in  $\mathcal{M}_{\text{bst}}^{\text{th}}$  puts the worst-case delay guarantees proved earlier for  $\mathcal{M}_{\text{bst}}$  at risk. Next, we argue that Theorem 7 also holds for  $\mathcal{M}_{\text{bst}}^{\text{th}}$  and its worst-case delay is asymptotically optimal, i.e., in the order of  $O(\text{size}(\text{tx})/((1-\mu)\text{val}(\text{tx})k))$ , albeit with a small overhead that has to do with the time it takes for the threshold to catch up to maliciously changing traffic conditions. As before, experimental results show tightly optimal delays under normal operation conditions. Notably, our result does not make any assumptions about the number of eligible transaction at each round, i.e., it is independent of traffic spikes.

Our analysis requires that the target transaction has token value at least  $2^{-\phi}$ ; parameters can be appropriately tuned to make  $\phi$  rather large for realistic applications. For simplicity, here we assume that  $\alpha := 1/2$ ,  $\phi = 20$ ,  $\rho := 0.1$ ,  $\beta := \ln(1 + \rho)$ .

► **Theorem 9.** Setting  $\alpha := 1/2$ ,  $\phi = 20$ ,  $\rho := 0.1$ ,  $\beta := \ln(1 + \rho)$ ,  $\mathcal{M}_{\text{bst}}^{\text{th}}$  has worst-case delay  $d(\text{tx})$  at most

$$80 \frac{\text{size}(\text{tx})}{(1-\mu)\text{val}(\text{tx})k} + \ln\left(\frac{11 \cdot \text{size}(\text{tx})}{\text{val}(\text{tx})}\right)/\beta + 10$$

when  $\text{size}(\text{tx}) < k/2$  and  $2^{-\phi} \leq \text{val}(\text{tx}) < 1/2$ .

**Proof.** The main rationale of the proof of Theorem 7 is that as long as the target transaction  $\text{tx}$  is not included in a block, it will eventually attain maximum priority, say

$$T := P(1 + \epsilon/c)(1 + \rho)^{\frac{\phi - \ln(c/\epsilon)}{\ln(2)}}$$

where  $\epsilon := \text{val}(\text{tx})$ ,  $c := \text{size}(\text{tx})$  and  $P := \frac{4}{(1-\mu)k}$ , and from this point on the adversary will have to fill honest blocks with high priority transactions other than  $\text{tx}$ , which it cannot do for long due to the limited rate at which priority is generated. We are going to apply the same

<sup>12</sup>We omit the formal proof of this result as it is rather similar to the analysis in [24].

logic to bound the worst-case delay of  $\mathcal{M}_{\text{bst}}^{\text{th}}$ , with the only difference that the adversary now may skip filling some honest blocks due to the threshold value being higher than  $T$  at that point of the game. This implies that for a number of blocks, at least as high as the number of blocks honest parties would leave empty, the threshold must be higher than  $T$ . We argue next, that in order for this to happen the adversary has to fill an amount of space with high priority transactions proportional to that in the original mechanism  $\mathcal{M}_{\text{bst}}$ , and thus does not gain much in terms of worsening the delay.

For the sake of contradiction, assume that  $\mathcal{M}_{\text{bst}}^{\text{th}}$  does not satisfy the theorem statement, and thus there exists a tx that has greater delay than  $d(\text{tx})$ . Note, that the term  $(1+\rho)^{\phi - \ln(c/\epsilon)}$  is upper-bounded by 7 for our parameters. Let  $u := 8 \cdot \frac{c}{(1-\mu)\epsilon k} + 1$ , as in Theorem 7. Similarly to the argument there, after  $7u$  blocks, tx will have obtained maximum priority equal to  $T$ . Now, let  $B_1, \dots, B_u$  be the sequence of blocks starting after tx has attained its maximum priority, and denote by  $T_i$  the threshold of block  $B_i$ .

Assume for the moment, that  $T_1 < T$ , and let  $i_1, i'_1, \dots, i_m, i'_m$  be a subsequence of indices of  $1, \dots, m$  such that

$$T_{i_j}, T_{i'_j} \leq T \text{ and } T_l > T \text{ for } l \in (i_j, i'_j), j \in [m];$$

$i_j, i'_j$  mark a sequence of threshold values that are greater than  $T$ .

First, we argue that any  $B_{i_j}$  for  $j \in [m]$  should contain transactions with token-value density at least  $\epsilon/(2c)$ , i.e., that

$$T_{i_j} > \hat{T} := P(1 + \epsilon/(2c))(1 + \rho)^{\frac{\phi - \ln(2c/\epsilon)}{\ln(2)}}$$

For two subsequent thresholds  $T', T''$ , where  $T'' > T$ , it holds that:

$$\begin{aligned} T'' > T &\Rightarrow T' e^{\beta(L-L/2)/(L/2)} > T \\ &\Rightarrow T' e^{\ln(1+\rho)} > P(1 + \epsilon/c)(1 + \rho)^{\frac{\phi - \ln(c/\epsilon)}{\ln(2)}} \\ &\Rightarrow T' > P(1 + \epsilon/c)(1 + \rho)^{\frac{\phi - \ln(c/\epsilon)}{\ln(2)} - 1} \\ &\Rightarrow T' > P(1 + \epsilon/(2c))(1 + \rho)^{\frac{\phi - \ln(2c/\epsilon)}{\ln(2)}} = \hat{T} \end{aligned}$$

where w.l.o.g., we have assumed that  $B_{i_j}$  is full. It follows that  $B_{i_j}$  contains only transactions with priority greater than  $\hat{T}$ , which can only be attained if the transactions have token-value density at least  $\epsilon/(2c)$ . Moreover, in case  $B_{i_j}$  is an honest block it should contain transactions with priority at least  $T$  and be at least  $aL$  full.

Furthermore, we argue that for the threshold to be larger than  $T$  in a sequence of blocks, as in  $T_{i_j}, \dots, T_{i'_j-1}$ , it must be the case that blocks on average contain an amount of data proportional to their number. First, for subsequent threshold values it should hold that:

$$T_{i+1} = T_i e^{\beta(x_i - aL)/(aL)} \Leftrightarrow x_i = aL(1 + \ln(T_{i+1}/T_i)/\beta)$$

where  $x_i$  is the fullness level of block  $i$ . Now, for any  $j$  and  $w := i_j, v := i'_j - 2$ , we get that:

$$\begin{aligned} \sum_{i=w}^v x_i &= aL \left( \sum_{i=w}^v (\ln(T_{i+1}) - \ln(T_i))/\beta + 1 \right) \\ &= aL((\ln(T_{v+1}) - \ln(T_w))/\beta + v - w + 1) \\ &= aL(\ln(T_{v+1}/T_w)/\beta + v - w + 1) \\ &\geq aL(v - w + 1) = aL(i'_j - i_j - 1) \end{aligned}$$

and since the threshold in all these blocks is above  $\hat{T}$ , as we argued before, it must be the case that these blocks contain  $aL(v - w + 1)$  amount of transactions with that much priority each. Summing over all  $j$ , we get that the respective blocks should contain an amount of  $aL\rho$  transactions with priority at least  $\hat{T}$  each, where  $\rho = |\{T_i | T_i > T, i \in [u]\}|$ .

Finally, honest blocks with threshold lower than  $T$  must be covered with transactions of density at least  $\epsilon/c$  and priority at least  $T$ , as otherwise tx is going to be included.

Putting it all together, we have the following:

- blocks in  $S = \{B_i | B_i \text{ is honest}, T_i < T, i \neq i_j, i \in [u], j \in [m]\}$  should contain transactions with total size at least  $aL$  and priority at least  $T$  each;
- blocks in  $H = \{B_i | T_j > T, i \in [u], i \neq i'_j - 1, j \in [m]\}$  should contain transactions with priority at least  $T$ ;
- blocks in  $W = \{B_{i_j} | j \in [m]\}$  should contain transactions with total size at least  $aL$  and priority at least  $\hat{T}$  each;
- blocks in  $W_H = \{B_{i_j} | B_{i_j} \text{ is honest}, j \in [m]\}$  should contain transactions with total size at least  $aL$  and priority at least  $T$  each.

Thus, the adversary has to generate transactions whose total normalized priority times size is at least:

$$\begin{aligned} & aL\hat{T} \cdot (2|S| + (|H| + |W \setminus W_H| + 2|W_H|)) \\ & \geq aL\hat{T} \cdot (|S| + |H| + m + |W_H|) \\ & \geq aL\hat{T} \cdot (|\{B_i | \text{ is honest block}, i \in [u]\}|) \\ & \geq aL\hat{T} \cdot (1 - \mu)u/(2c) \end{aligned}$$

where we have used the fact that  $|W| = m$  and blocks in  $W_H$  contain transactions with priority  $T$ . Hence, the adversary has to fill as many blocks as in the proof of Theorem 7 when the token-value density of the target transaction is  $\epsilon/2c$ . By our previous analysis this is not possible for  $u_2 := 16c/((1 - \mu)\epsilon k) + 1$ .

Finally, it remains to argue about our assumption that  $T_1$  is less than  $T$ . Assume that we are at a round where tx has maximum priority  $T$  and the threshold remains above or equal to  $T$  for  $u' + 1$  rounds. By our earlier argument, to maintain the threshold above  $T$  the adversary must fill produced blocks with high priority ( $> T$ ) transactions of total size:

$$\begin{aligned} aL(\ln(T_{u'+1}/T_1)/\beta + u') & \geq aL(\ln((P\epsilon/c)/(7P(1 + 1/2)))/\beta + u') \\ & \geq aL(u' - \ln(11c/\epsilon)/\beta) \end{aligned}$$

where  $7P(1 + 1/2)$  is an upper bound on the threshold value. We want to choose a  $u'$  such that it is impossible for  $\mathcal{A}$  to generate that many high priority transactions within  $u'$ .

By Theorem 7, we know that it is impossible to fill  $\tilde{u}(1 - \tilde{\mu})$  honest blocks with transactions of priority  $T$  in less than  $\tilde{u} := 8c/((1 - \tilde{\mu})\epsilon k)$  rounds. Take now  $\tilde{\mu} > 1/(1 + \frac{8c}{\epsilon k \gamma})$ , where  $\gamma := \ln(11c/\epsilon)/\beta$ . It holds that  $\tilde{u} > \gamma + 8c/(\epsilon k)$ . Setting  $u' := \tilde{u}$ , we see that

$$u' - \ln(11c/\epsilon)/\beta > u' - \gamma = 8c/(\epsilon k) \geq u'(1 - \tilde{\mu})$$

which implies by our previous observation that  $\mathcal{A}$  will not be able to fill the required blocks to retain the threshold larger than  $T$  for this selection of  $u'$ .

Concluding, tx must be included in a block after a total of

$$72c/((1 - \mu)\epsilon k) + \ln(11c/\epsilon)/\beta + 8c/(\epsilon k) + 10$$

rounds. The theorem follows. ◀

### 3.5 Putting everything together

We have shown that given a transaction and its token-value, it is possible to bound (and thus predict) its worst-case delay. Moreover, that this delay is asymptotically optimal and that mechanism  $\mathcal{M}_{\text{bst}}^{\text{th}}$  is off-chain proof. Note now, that if there was a way of distributing the blockchain space tokens ahead of time, mechanism  $\mathcal{M}_{\text{bst}}^{\text{th}}$  would satisfy our initial goals. We formalize this idea in the next theorem.

► **Theorem 10.** *Given tokens are obtained and available in advance before use by a window of time  $t$  to a user, and setting the mechanism parameters as in Theorem 9,  $\mathcal{M}_{\text{bst}}^{\text{th}}$  offers predictable service to that user with parameter  $t$ , it has asymptotically optimal worst-case delay, and is Off-Chain Proof.*

**Proof.** Theorem 9 and Corollary 8 imply asymptotically optimal worst-case delay and Off-Chain Proofness. To argue about predictable service, note that since tokens are obtained  $t$  time in advance, a user knows ahead of time both the cost and the worst-case delay of its transaction; the worst-case delay can be calculated based on the amount of token value obtained by the user using Theorem 9. The theorem follows. ◀

Note that as in [24], off-chain proofness is shown unless in the midst of a demand spike. To complete our proposal, in the next section we describe a way of distributing tokens ahead-of-time.

## 4 Allocation of Tokenized Space

In this section, we discuss a specific way to apply the ideas and results from the previous section. Let's first consider two potential options for token instantiation and distribution. The two extremes are to use either the existing stake (in Proof-of-Stake systems) or to create a *new space token specifically designed for this purpose*.

Using dedicated space tokens offers significant flexibility because they are independent of any restrictions related to other uses of stake, such as consensus or smart contracts. However, there is a risk of Denial-of-Service (DoS) attacks if a malicious party acquires almost all space tokens. We propose two methods to mitigate this risk, both of which offer additional advantages.

The first idea is to *partition the blockchain space into two fixed parts*: the “spot space” and the “tokenized space.” In the *spot space*, transactions are included using the usual mechanism (e.g., first-price auctions in Bitcoin, EIP-1559 in Ethereum). In the *tokenized space*, transactions are included using the proposed BST mechanism of the previous section with the space tokens. The fraction allocated to the tokenized space is fixed permanently or adjusted very slowly by blockchain governance to meet demand. We anticipate power users who issue many transactions will utilize the tokenized space, while regular users will primarily use the spot space. DeFi users may leverage both spaces depending on their needs for speed, cost, and predictability.

The second idea is to *limit the lifetime of each access token*. A lifespan of a few months makes it challenging and expensive for a malicious actor to control all tokens for an extended period. Of course, preventing a well-funded attacker from acquiring all tokens is impossible, but this risk exists in any system that allocates blockchain space through a transaction fee system.

A simple and effective way to distribute the space tokens is to sell them in an *auction conducted within the spot space*. This approach avoids bootstrapping difficulties. For example, the system can run a sealed-bid auction every  $T$  blocks for tokens expiring after  $L$  blocks,

where  $L$  can be a small multiple of  $T$  (e.g., a monthly auction for tokens with a 3-month lifespan). While sealed-bid auctions typically require significant space for registering bids and recording outcomes on the blockchain, we anticipate that only a few hundred power users will participate, making on-chain execution feasible. Otherwise, the auction can run off-chain, with only the results recorded on the blockchain. To prevent incentive issues, *bidders should submit encrypted bids along with collateral*. During the auction, bids are decrypted and the winners are determined. The collateral ensures bidders don't withdraw after the auction and should be set high enough to deter this behavior.

With sealed bids, there are several options for a *truthful multi-unit auction*, including the VCG auction with no reserve price, the VCG with a reserve price [18], or even more exotic auctions (such as Triage auctions [7]). Taking into account limitations in communication and blockchain space, the VCG auction with reserve price emerges as the most suitable choice. This reserve price can be calculated based on historical values of the tokenized space, transaction fees of the spot space, or set as a fixed value with adjustments by blockchain governance. An additional advantage of the VCG auction with reserve price is that it is also the optimal auction for maximizing revenue in Myerson's settings [17]. The revenue of the auction could be equally distributed to the block producers at the end of the period to eliminate any strategic considerations by block producers. While technically this is a repeated auction, the analysis of the myopic (single-shot) setting captures the key aspects given the relatively long intervals between auctions.

Once the auction concludes, the tokens become tradable like any other token until they expire. Power users can estimate their service needs at the auction and then buy or sell tokens to adjust their requirements throughout the period.

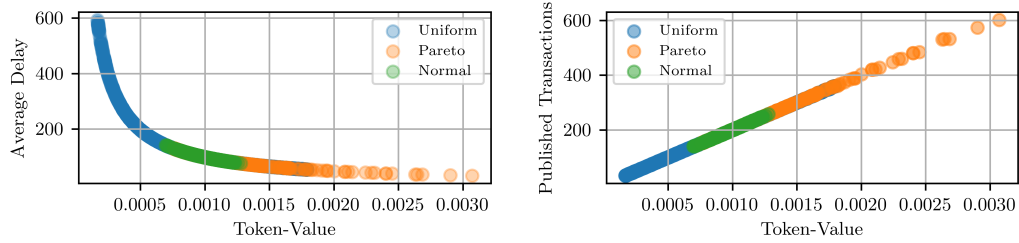
## 5 Deployment Considerations

Next, we focus on deployment considerations of mechanism  $\mathcal{M}_{\text{bst}}^{\text{th}}$ . We argue that the main component of the system, i.e., the procedure that computes the priorities of different transactions, can be efficiently and compactly implemented in the major blockchain architectures.

Firstly, our scheme does not require support of *any specialized cryptographic primitives*, such as VRFs, VDFs, ZK-SNARKs, etc.<sup>13</sup> Typically, implementing new cryptographic primitives is one of the major obstacles in quickly releasing new technology in the blockchain landscape. The main operation of the mechanism revolves around being able to efficiently determine the priority of different transactions and pick the ones that have the maximum priority, which basically amounts to suitable book-keeping.

In more detail, the mempool operator should maintain a data-structure  $DS$  containing the block that each token was last referenced by some transaction. To determine the priority of a transaction  $tx$ , it suffices to query  $DS$  about the age of the tokens referenced by  $tx$ , and then simply compute the priority value following the equation in Section 3.3. Using some kind of self-balancing binary search tree to implement  $DS$ , e.g., an AVL tree, allows retrieving and updating the token-age related information in  $O(\log(n))$  time in the worst-case, where  $n$  is the total number of tokens. Therefore, computing the priority of a transaction takes  $O(m \cdot \log(n))$  time, where  $m$  is the number of tokens referenced by the transaction, while updating the token related information takes  $O(l \cdot \log(n))$ , where  $l$  is the number of token values to be updated. Space-wise, the AVL tree takes about  $O(n)$  space.

<sup>13</sup>We note that this is not necessarily the case for the token-distribution part, where an auction has to be deployed.



■ **Figure 3** 1000 users for 20000 steps, where every user becomes active with  $p_{i,t} = 0.8$ .

To more efficiently use their tokens, users may be tempted to split them in smaller chunks. This way they can better control the priority injected into a transaction by including more or less of these small tokens. Such token-splitting would result in more load to the system, and is generally unwanted. To avoid such an issue, a transaction can explicitly state how much of the priority generated by each token referenced should be used, with the rest retained for later use. Such a change does not affect the security analysis presented in previous sections, and can be easily implemented:  $DS$ , in addition to the last-use information stored per token, also stores any remaining priority left from a previous use of the token. Again, transaction priority can be efficiently computed.

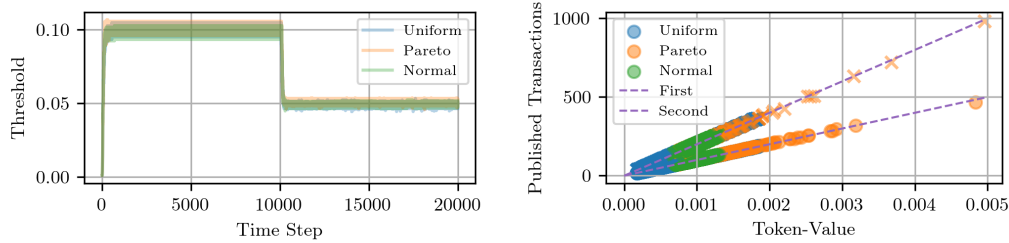
Finally, we describe two possible instantiations of the mechanism in the UTXO (used by Bitcoin) and the account model (used by Ethereum), respectively.

In the UTXO-based case, say in Bitcoin, we can introduce our space tokens using the ordinals mechanism [1] each satoshi (Bitcoin’s smallest denomination) can receive an inscription and afterwards it can be transferred as an NFT. The initial inscription can specify the value of the token in terms of priority, and subsequently, it is possible to consume only a fraction of the priority of a token by prescribing a value in  $[0, 1]$  and using the reinscription mechanism of ordinals – this deals with the token-splitting issue described earlier. To issue a transaction utilizing such a token, it is sufficient to post a transaction transferring the corresponding NFT to the change address of the posting user, while setting the transaction fee to 0. It is easy to see that this mechanism can be facilitated as a soft fork in the Bitcoin network (note that transaction relaying with 0-fees would need to be amended accordingly). On the other hand, in the account-based case, say in Ethereum, a smart contract can mint the tokens with their corresponding values and subsequently the priority consumed can be specified in each transaction. Note finally that a hard-fork would be required to allow transactions posted with zero fees that utilize a space token instead to become admissible into the ledger (as due to EIP-1559 it is imperative that a valid transaction comes with a minimum fee).

## 6 Simulations

We validate our theoretical results using experiments on synthetic blockchain traffic. Specifically, we are assuming that there are  $n$  independent transaction issuers, each of which has tokens of value  $v_i > 0$ , sampled independently for the same distribution  $F$  (and subsequently normalized so the total amount is 1). At every step, each user might be *active* or *inactive*. If they are inactive at time  $t - 1$ , they flip a coin and become active at time  $t$  with probability  $p_{i,t}$ . Once they are active, they submit a transaction and remain active until that transaction is published. Then, they become inactive again and the cycle continues. For simplicity, the users do not trade their tokens and there are no adversarially produced blocks. We use the BST mechanism with the parameters of Theorem 9. Each transaction has size 1 and a block can hold up to 20 transactions, with 10 being the target size for the threshold update rule.





■ **Figure 4** 1000 users for 20000 steps, where a random subset of half the users stops issuing new transactions halfway. Notice that the threshold drops to half its value and the users that remain active publish twice as many transactions given their token value. The user transactions published in the second half are denoted using the cross symbol, while the circles refer to the first half.

We simulate two scenarios, both of which consist of four runs with different token distributions. Specifically, we have:

- Uniform in  $[0, 1]$ .
- Pareto Type II with parameter 2.
- Truncated Normal with  $\mu = 100$  and  $\sigma^2 = 10$ .

In the first scenario, depicted in Figure 3, we have 1000 users for 20000 blocks. The activation probability of all users stays the same throughout. The average delays follow the worst case result from Theorem 9. The relation is much better depicted in the graph at the right, where the number of published transactions (which is the inverse of the delay multiplied by the number of blocks) is shown to be linear in the amount of tokens.

In the second scenario, depicted in Figure 4, we show that this mechanism has the ability to *adapt* to changes in demand. We vary the activation probability as follows: all users have  $p_{i,t} = 1$  for the first 10000 steps and then half of the users switch to 0. Notice how the threshold decreases, and also that in both cases we match the optimal worst-case delay (which is easier seen as the number of published transactions), thus showcasing that under normal operation conditions our mechanism indeed achieves optimal delays.

---

## References

- 1 Ordinal Theory Authors. Ordinal theory handbook. <https://docs.ordinals.com/>, 2024.
- 2 Soumya Basu, David Easley, Maureen O’Hara, and Emin Gün Sirer. Towards a functional fee market for cryptocurrencies. *arXiv preprint*, 2019. [arXiv:1901.06830](https://arxiv.org/abs/1901.06830).
- 3 Vitali Buterin. On inflation, transaction fees and cryptocurrency monetary policy, 2016. URL: <https://blog.ethereum.org/2016/07/27/inflation-transaction-fees-cryptocurrency-monetary-policy>.
- 4 Vitalik Buterin, Eric Conner, Rick Dudley, Matthew Slipper, and Ian Norden. Ethereum improvement proposal 1559: Fee market change for eth 1.0 chain, 2019. URL: <https://github.com/ethereum/EIPs/blob/master/EIPS/eip-1559.md>.
- 5 Xi Chen, David Simchi-Levi, Zishuo Zhao, and Yuan Zhou. Bayesian mechanism design for blockchain transaction fee allocation. *Available at SSRN 4413816*, 2023.
- 6 Hao Chung and Elaine Shi. Foundations of transaction fee mechanism design. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3856–3899. SIAM, 2023.
- 7 Shahar Dobzinski and Noam Nisan. Multi-unit auctions: Beyond roberts. *J. Econ. Theory*, 156:14–44, 2015.

- 8 Matheus VX Ferreira, Daniel J Moroz, David C Parkes, and Mitchell Stern. Dynamic posted-price mechanisms for the blockchain transaction-fee market. In *Proceedings of the 3rd ACM Conference on Advances in Financial Technologies*, pages 86–99, 2021.
- 9 Yotam Gafni and Aviv Yaish. Greedy transaction fee mechanisms for (non-) myopic miners. *arXiv preprint*, 2022. [arXiv:2210.07793](https://arxiv.org/abs/2210.07793).
- 10 Juan A Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. *Journal of the ACM*, 2015.
- 11 Andrew V Goldberg, Jason D Hartline, Anna R Karlin, Michael Saks, and Andrew Wright. Competitive auctions. *Games and Economic Behavior*, 55(2):242–269, 2006.
- 12 Benjamin Hindman, Andy Konwinski, Matei Zaharia, Ali Ghodsi, Anthony D Joseph, Randy Katz, Scott Shenker, and Ion Stoica. Mesos: A platform for {Fine-Grained} resource sharing in the data center. In *8th USENIX Symposium on Networked Systems Design and Implementation (NSDI 11)*, 2011.
- 13 Gur Huberman, Jacob D Leshno, and Ciamac Moallemi. Monopoly without a monopolist: An economic analysis of the bitcoin payment system. *The Review of Economic Studies*, 88(6):3011–3040, 2021.
- 14 Ron Lavi, Or Sattath, and Aviv Zohar. Redesigning bitcoin’s fee market. *ACM Transactions on Economics and Computation*, 10(1):1–31, 2022.
- 15 Stefanos Leonardos, Barnabé Monnot, Daniël Reijnders, Efstratios Skoulakis, and Georgios Piliouras. Dynamical analysis of the eip-1559 ethereum fee market. In *Proceedings of the 3rd ACM Conference on Advances in Financial Technologies*, pages 114–126, 2021.
- 16 Stefanos Leonardos, Daniël Reijnders, Daniël Reijnders, Barnabé Monnot, and Georgios Piliouras. Optimality despite chaos in fee markets. *arXiv preprint*, 2022. [arXiv:2212.07175](https://arxiv.org/abs/2212.07175).
- 17 Roger B. Myerson. Optimal auction design. *Math. Oper. Res.*, 6(1):58–73, 1981.
- 18 Noam Nisan, Tim Roughgarden, Éva Tardos, and Vijay V. Vazirani, editors. *Algorithmic Game Theory*. Cambridge University Press, 2007.
- 19 Oiler. The oiler network. <https://www.oiler.network/>. Accessed: 2024-10-16.
- 20 Rafael Pass, Lior Seeman, and abhi shelat. Analysis of the blockchain protocol in asynchronous networks. Cryptology ePrint Archive, Report 2016/454, 2016. URL: <http://eprint.iacr.org/2016/454>.
- 21 Rafael Pass, Lior Seeman, and Abhi Shelat. Analysis of the blockchain protocol in asynchronous networks. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017 – 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II*, volume 10211 of *Lecture Notes in Computer Science*, pages 643–673, 2017. doi:10.1007/978-3-319-56614-6\_22.
- 22 Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments. <https://lightning.network/lightning-network-paper.pdf>, January 2016.
- 23 Daniël Reijnders, Shyam Sridhar, Barnabé Monnot, Stefanos Leonardos, Stratis Skoulakis, and Georgios Piliouras. Transaction fees on a honeymoon: Ethereum’s eip-1559 one month later. In *2021 IEEE International Conference on Blockchain (Blockchain)*, pages 196–204. IEEE, 2021.
- 24 Tim Roughgarden. Transaction fee mechanism design. In Péter Biró, Shuchi Chawla, and Federico Echenique, editors, *EC ’21: The 22nd ACM Conference on Economics and Computation, Budapest, Hungary, July 18-23, 2021*, page 792. ACM, 2021. doi:10.1145/3465456.3467591.
- 25 Itay Tsabary, Alex Manuskin, Roi Bar-Zur, and Ittay Eyal. Ledgerhedger: Gas reservation for smart-contract security. *Cryptology ePrint Archive*, 2022.
- 26 Carl A Waldspurger and William E Weihl. Lottery scheduling: Flexible proportional-share resource management. In *Proceedings of the 1st USENIX conference on Operating Systems Design and Implementation*, pages 1–es, 1994.
- 27 Andrew Chi-Chih Yao. An incentive analysis of some bitcoin fee designs. *arXiv preprint*, 2018. [arXiv:1811.02351](https://arxiv.org/abs/1811.02351).

# Optimal RANDAO Manipulation in Ethereum

Kaya Alpturer  

Princeton University, NJ, USA

S. Matthew Weinberg  

Princeton University, NJ, USA

---

## Abstract

It is well-known that RANDAO manipulation is possible in Ethereum if an adversary controls the proposers assigned to the last slots in an epoch. We provide a methodology to compute, for any fraction  $\alpha$  of stake owned by an adversary, the maximum fraction  $f(\alpha)$  of rounds that a strategic adversary can propose. We further implement our methodology and compute  $f(\cdot)$  for all  $\alpha$ . For example, we conclude that an optimal strategic participant with 5% of the stake can propose a 5.048% fraction of rounds, 10% of the stake can propose a 10.19% fraction of rounds, and 20% of the stake can propose a 20.68% fraction of rounds.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Algorithmic game theory and mechanism design; Information systems  $\rightarrow$  Digital cash; Security and privacy  $\rightarrow$  Distributed systems security

**Keywords and phrases** Proof of Stake, Consensus, Blockchain, Ethereum, Randomness manipulation

**Digital Object Identifier** 10.4230/LIPIcs.AFT.2024.10

**Supplementary Material** *Software*: <https://github.com/kalpturer/randao-manipulation>  
archived at `swh:1:dir:4632247a5ed3767ade8c71d514997c9f99d52387`

**Funding** Supported by NSF CAREER CCF-1942497 and a grant from the Ripple University Blockchain Research Initiative.

**Acknowledgements** We are grateful to Noah Citron for introducing us to the problem, and helpful discussions in early phases of this work. We are also grateful to Yunus Aydın, István Seres, Aadityan Ganesh, and anonymous reviewers for feedback on earlier drafts of this work.

## 1 Introduction

Randomness is an essential component of blockchain protocols. With the invention of Proof of Work blockchains [20], a major innovation in Bitcoin was to use the randomness of the SHA256 function to select the next block proposer. In particular, a participant in the Bitcoin ecosystem is able to propose a block of their choice with probability proportional to their computational power. While this system satisfies many desirable properties, it is in many ways not desirable due to inefficiency. With the move to Proof of Stake blockchain protocols, the dependence on computation is replaced with stake in the digital currency itself. However, a new source of randomness is needed to select the next block proposer with probability *proportional to one's stake*. A major security requirement for this randomness is for it to be verifiable (i.e. everyone can verify that the block proposer lottery was not rigged) and unpredictable (i.e. before the lottery happens, no one can know the winner).

Several approaches exist to provide this source of randomness to Proof of Stake blockchain protocols. One approach is to use an external randomness beacon [13] which achieves similar guarantees as in Bitcoin. However, implementing such a beacon comes with trust centralization concerns. A more practical approach is to use protocols that rely on pseudorandom cryptographic primitives to select block proposers, which are adopted by Proof of Stake blockchains such as Ethereum.



© Kaya Alpturer and S. Matthew Weinberg;  
licensed under Creative Commons License CC-BY 4.0  
6th Conference on Advances in Financial Technologies (AFT 2024).

Editors: Rainer Böhme and Lucianna Kiffer; Article No. 10; pp. 10:1–10:21

Leibniz International Proceedings in Informatics



LIPIC Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 10:2 Optimal RANDAO Manipulation in Ethereum

While the system safety and liveness are not compromised by the randomness mechanism in current Proof of Stake blockchains, it is well-known that they are susceptible to manipulation (see [23, 7]). In particular, incentive-incompatibilities that result in block-withholding behavior exist in many Proof of Stake blockchain protocols – analogous to selfish mining for Bitcoin [10, 26, 22].

There is a recent line of work focusing on Proof of Stake incentive incompatibilities [4, 13, 15, 3, 14], some of which derive concrete bounds on optimally manipulating randomness for the Algorand protocol. However, while some analyses such as [23, 9] and simulation-based approaches such as [2] conclude that randomness manipulation is likely to be negligible for Ethereum, it is currently unknown how much more an adversary that *optimally* manipulates Ethereum can make. In this paper, we focus on answering this question and compute optimal strategies for randomness manipulation in Ethereum. Our approach relies on modeling the randomness manipulation game as a Markov decision process.

### 1.1 Brief overview of Proof of Stake Ethereum

We now briefly cover the relevant details of the Proof of Stake Ethereum protocol. In the Ethereum protocol, time is divided into epochs, each epoch is divided into 32 slots, and each slot is 12 seconds. Each epoch is assigned 32 block proposers (one for each slot) who can construct and broadcast a block to be added to the blockchain at that slot. If the block proposer fails to do so, the slot is *missed* (no block is added) and the blockchain moves on to the next slot.

Proof of Stake Ethereum provides randomness by a scheme that maintains a random value called the RANDAO (also known as `randao_reveal`) in each block [9]. As each block is proposed, the previous RANDAO value is mixed using the private key of the proposer. Since the private key is used to sign the epoch number and is mixed into the previous RANDAO value by the xor operation, the mixing is *verifiable*. Moreover, as the signature is assumed to be uniformly random and the private key is unknown to the public, it is *unpredictable*. These properties ensure that the only actions available to an adversary in influencing the RANDAO value is to choose between *broadcasting* or *withholding* a block.

At the end of each epoch, the RANDAO value is used to select a set of 32 new proposers for the next epoch<sup>1</sup>. For example, if an adversary controls multiple validators and happens to get assigned to propose in slot 30 and 31 (the last two slots of an epoch), after slot 29 passes, the adversary can use the RANDAO value at slot 29 to compute 4 different RANDAO outcomes for the next epoch. If the adversary withholds both 30 and 31, the RANDAO value remains the same. If the adversary withholds 30 and broadcasts 31, the RANDAO value is only mixed with the signature of the proposer of 31, and so on. By precomputing 4 possible outcomes, the adversary is able to select one of the 4 RANDAO values (at the cost of missing the relevant block rewards). Similarly, in general, an adversary that controls the last  $k$  proposers in an epoch is able to choose from  $2^k$  RANDAO values that determine the proposers for the next epoch. We call the longest contiguous adversarial slots at the end of an epoch the tail. With this scheme, it is conceivable that an adversary may strategically withhold their block at specific slots to win the right to produce *more* blocks in expectation.

---

<sup>1</sup> Ethereum actually skips an epoch in this process so the RANDAO value at the end of epoch  $i$  determines the 32 proposers for epoch  $i + 2$  – we discuss this in more detail in Section 3

## 1.2 Main contributions

Our main technical contributions in this paper are:

- We model and formalize the game that an adversary with  $\alpha < 1$  proportion of the total staked Ethereum plays in manipulating the RANDAO value.
- We show that the RANDAO manipulation game can be formulated as a Markov Decision Process, and we show how to significantly reduce the state space so that policy iteration on a laptop quickly converges.<sup>2</sup>
- We present precise answers to the fraction of slots a strategic player can propose after optimally manipulating Ethereum’s RANDAO.

## 1.3 Related Work

**Manipulating Ethereum’s RANDAO.** The name RANDAO comes from (and the scheme is inspired by) an earlier project [1], and its manipulability has been acknowledged and discussed in the Ethereum community [6, 5]. Work of [2] focuses on modeling the RANDAO mechanism<sup>3</sup> in probabilistic rewrite logic and evaluating greedy strategies (analogous to TAIL-MAX of Section 5.1). In evaluating the model, they follow a simulation based approach with epoch length 10 and report some biasability. In [23], some probabilistic analysis of how many blocks an adversary controlling the last two proposers in the current epoch can get in the next epoch is presented. In addition, it is demonstrated that in specific instances, some staking pools had the opportunity to control more than half the next epoch. Lastly, [9] shows that the number of proposers an adversary controls at the tail is expected to shrink as long as the adversarial stake is less than roughly 1/2. It also provides an strategy analysis that considers tails of length 0 and 1, concluding marginal improvement over the honest. These results are consistent with ours, and as expected the reported improvement in rewards is less than the optimal strategy we compute. For example, a 25% adversary with their single look-ahead strategy makes 2.99% more than the honest while we compute that the optimal strategy makes 4.09% more than the honest. In comparison to these works, our work nails down the optimal RANDAO manipulation, and via a principled framework that can accommodate slight modifications (such as epoch length) as well.

**Computing Optimal Manipulations.** The most related methodological papers are [22, 14], who also compute optimal strategic manipulations. [22] computes optimal manipulations in Bitcoin’s longest-chain protocol, and [14] computes optimal manipulations in Algorand’s cryptographic self-selection. Our work is methodologically similar, as we also formulate an MDP and use some technical creativity to solve it. On the methodological front, our state-space reduction in Section 4 is perhaps most distinct from prior work.

**Manipulating Consensus Protocols, generally.** There is a significant and rapidly-growing body of work on manipulating consensus protocols broadly [10, 22, 19, 8, 17, 16, 13, 11, 25, 24, 3, 14]. Aside from the aforementioned works, most of these do not compute *optimal* manipulations, but instead understand when profitable manipulations exist. For Ethereum’s RANDAO, it is already well-understood that profitable manipulations exist for arbitrarily small stakers, and so the key open problem is how profitable they are (which our work resolves).

---

<sup>2</sup> Our implementation is available here: <https://github.com/kalpturer/randao-manipulation>

<sup>3</sup> The RANDAO model in [2] is an earlier variation of the RANDAO mechanism that uses a commit-reveal scheme. The induced game, however, is quite similar.

## 1.4 Roadmap

In Section 2, we briefly cover the relevant background on Markov Decision Processes. In Section 3 we formalize the RANDAO manipulation process as a game. In Section 4, we formulate the RANDAO manipulation game as a Markov Decision Process and reduce the state space to a tractable regime. In Section 5 and 6 we describe how to evaluate and solve for optimal policies. Lastly, in Section 7 we conclude with a discussion on modeling assumptions, future work, and block miss rates.

## 2 Preliminaries: Markov Decision Processes

In this section, we quickly review the necessary background for Markov chains and Markov decision/reward processes. The material in this section is largely drawn from [18] and [21]. These texts can be consulted for a more detailed treatment.

A *Markov chain*  $C = (S, P)$  consists of a set of states  $S$  and transition probabilities  $P : S \times S \rightarrow \mathbb{R}$ . Given a current state  $s \in S$ , we transition to the next state with the probability distribution induced by  $P(s, \cdot)$ . Rewards can be added to this framework with a reward function  $R : S \times S \rightarrow \mathbb{R}$  such that if we transition from state  $s$  to  $s'$ , we get  $R(s, s')$  reward. A Markov chain with rewards is a *Markov reward process (MRP)*.

An agent navigating a Markovian system can be modelled using a *Markov decision process (MDP)*. An MDP is a tuple  $M = (S, A, \{P_a\}_{a \in A}, \{R_a\}_{a \in A})$  where  $S$  is a set of states,  $A$  is a set of actions,  $P_a : S \times S \rightarrow \mathbb{R}$  is a transition probability function representing the probability of individual transitions given an action  $a \in A$ , and  $R_a : S \times S \rightarrow \mathbb{R}$  represents the reward of transitioning between individual states with action  $a \in A$ .

A *policy*  $\pi : S \rightarrow A$  is a function specifying which actions to take given the current state. Once we fix a policy in an MDP, we get an MRP. We only consider deterministic policies since the standard results [21] show that in the models we consider, an optimal deterministic policy exists.

We will be modeling the RANDAO manipulation game as an MDP. We now introduce some definitions and properties that will be useful when we introduce our MDP.

### 2.1 Properties of Markovian systems

One important property concerns whether some states are visited infinitely often.

A state  $s$  in a Markov Chain is *recurrent* if, conditioned on currently being at state  $s$ , the probability of later returning to state  $s$  is 1. If a state is not recurrent, it is called *transient*. A *recurrent class* of states is a set of recurrent states  $\hat{S}$  such that, for all  $s \in \hat{S}$ , conditioned on being at state  $s$ : for all  $s' \in \hat{S}$ , the probability of visiting  $s'$  at a later time is  $> 0$ .

A Markov chain is *ergodic* if it consists of a single recurrent class of states. Similarly an MDP is ergodic if for every deterministic policy,<sup>4</sup> the Markov chain induced by the policy is ergodic. All MDPs we will consider will be ergodic so for the rest of this section we assume ergodicity.

A *stationary distribution*  $\sigma$  of a transition probability matrix  $\mathbf{P}$  in a Markov chain is defined to be a solution to the following:

$$\sigma = \sigma \mathbf{P} \quad \text{and} \quad \sum_i \sigma_i = 1$$

<sup>4</sup> We only work with stationary policies which are policies that do not change over time. For the processes we consider a stationary optimal policy is guaranteed to exist.

► **Proposition 1** ([21]). *If a Markov chain is ergodic, then there exists a unique stationary distribution.*

## 2.2 Reward criteria

Now we define the reward criteria for a Markov decision process  $M$ . For our application, *average reward* is more appropriate than *discounted reward* since we are concerned with the infinite behavior of the system.

Let  $\rho_{\pi,s}(m)$  be a random variable that is equal to the reward at time  $m$  while transitioning the state at time  $m$  to the state at time  $m + 1$  in some MDP when running policy  $\pi$  starting at state  $s$ .

The *average reward* of a policy  $\pi$  is defined as:

$$\Gamma_{\pi} = \lim_{N \rightarrow \infty} \mathbb{E} \left[ \frac{1}{N} \sum_{m=0}^N \rho_{\pi}(m) \right]$$

where we rely on the following result to ignore the initial state.

► **Proposition 2** ([21]). *The average reward for ergodic MDPs is initial state independent.*

Let  $q(s)$  be the expected reward of transitioning from state  $s$ . More formally  $q(s) = \sum_{s' \in S} R_{\pi(s)}(s, s') P_{\pi(s)}(s, s')$ . Then,  $\Gamma_{\pi}$  can be computed using the stationary distribution  $\sigma^{\pi}$  of the Markov chain induced by fixing policy  $\pi$ .

$$\Gamma_{\pi} = \sum_{s \in S} q(s) \sigma_s^{\pi}$$

**Value Functions.** In any recurrent process, it is a useful concept to understand the “value” of being in one state over another, due to the potential future rewards. With non-discounted rewards, this requires some subtlety to properly define (because the expected future reward from any state is infinite). One standard method is to define the value of a state as its *average adjusted sum of rewards*:

$$v_{\pi}(s) = \lim_{N \rightarrow \infty} \mathbb{E} \left[ \sum_{m=0}^N (\rho_{\pi,s}(m) - \Gamma_{\pi}) \right]$$

That is, the average adjusted sum of rewards captures the additive difference between an unbounded process starting from state  $s$  and iterating  $\pi$  and an unbounded process that earns  $\Gamma_{\pi}$  (the average per-round reward of  $\pi$ ) per round.

► **Lemma 3** ([18]). *For an ergodic MDP  $M$ ,*

$$v_{\pi}(s) + \Gamma_{\pi} = q(s) + \sum_{s' \in S} P_{\pi(s)}(s, s') v_{\pi}(s')$$

Since we can compute  $\Gamma_{\pi}$  first given a policy, this equation determines all  $v_{\pi}$  up to an additive constant which we can solve for after setting  $v_{\pi}(s) = 0$  for some  $s \in S$ .<sup>5</sup>

<sup>5</sup> The average reward  $\Gamma_{\pi}$  is sometimes called *gain* and what we call the *value*  $v_{\pi}$ , which is the average adjusted sum of rewards, is sometimes called *bias* in the literature.

## 10:6 Optimal RANDAO Manipulation in Ethereum

To find the optimal policy with respect to the average reward criterion, we can run the policy iteration algorithm of [18, 21]. Starting from an arbitrary policy  $\pi_0$ , evaluate the policy to compute  $\Gamma_{\pi_0}$  and  $v_{\pi_0}$ . Then, a policy improvement step is performed which defines  $\pi_{i+1}(s) := \arg \max_{a \in A} \{q_a(s) + \sum_{s' \in S} P_a(s, s') v_{\pi_i}(s')\}$ . The following Bellman optimality equation guarantees that once this process stabilizes, we have an average reward optimal policy.

► **Theorem 4** (Bellman optimality equation for average-reward MDPs, [18]). *If a policy  $\pi^*$  satisfies the following equations in an MDP, then  $\pi^*$  is average reward optimal.*

$$v_{\pi^*}(s) + \Gamma_{\pi^*} = \max_{a \in A} \left\{ q_a(s) + \sum_{s' \in S} P_a(s, s') v_{\pi^*}(s') \right\} \quad \forall s \in S$$

### 3 The RANDAO manipulation game

We now review RANDAO in more detail, and formulate the RANDAO manipulation game. RANDAO is a pseudorandom seed that updates every block, and is used to select Ethereum proposers. Below, we use the terminology  $R(b)$  to denote the RANDAO value after the  $b^{\text{th}}$  slot has finished.

**Updating RANDAO.** The process for updating  $R(b)$  is quite simple. If no one proposes a block during slot  $b$  of epoch  $x$ , then  $R(b) = R(b-1)$ . If a block is proposed during slot  $b$ , the proposer must also digitally sign the epoch number  $x$  and the hash of this digital signature is XORed with  $R(b-1)$  to produce  $R(b)$ . Note, in particular, that there is a unique private key eligible to propose a block during slot  $b$ , and therefore the only two possibilities for  $R(b)$  are either  $R(b-1)$  (if no block is proposed) or  $R(b-1)$  XOR hash(signature of  $x$  by proposer for slot  $b$ ).

**Using RANDAO to seed epochs.** The Ethereum blockchain consists of epochs, where each epoch contains 32 blocks. Within each epoch  $t$ , a seed  $S(t)$  determines which private keys are eligible to propose during each slot. That is, for each of the 32 slots in an epoch, the proposer of that slot is a deterministic function of  $S(t)$  (but  $S(t)$  is a pseudorandom number). Moreover, if  $S(t)$  is a uniformly random number, then each slot proposer is independently and uniformly randomly drawn proportional to stake.

$S(t)$  is set based on RANDAO. Specifically,  $S(t)$  is equal to the value of RANDAO at the end of epoch  $t-2$ . To be extra clear, there have been  $32(t-2)$  slots completed by the end of epoch  $t-2$ , so  $S(t) := R(32t-64)$ .

**Rewards.** In practice, proposer rewards involve transaction fees, Maximal Extractable Value (MEV), and any payments made in the Proposer-Builder-Separation (PBS) ecosystem. To streamline analysis, and to be consistent with an overwhelming majority of prior work, we focus on the *fraction of slots* where an adversary proposes.<sup>6</sup> That is, we consider an adversary who aims to maximize the fraction of slots where they propose.

---

<sup>6</sup> Of course, it is an appropriate direction for future work to instead explicitly model transaction fees, MEV, etc. Such modeling would only make an adversary stronger, as their strategy can now depend on the value of each slot [8].



**Ideal Cryptography.** It is widely-believed that digital signatures of a previously-unsigned message using an unknown private key (and hashes of previously-unhashed inputs, etc.) are indistinguishable from uniformly-random numbers by computationally-bounded adversaries. However, such cryptographic primitives do not generate truly uniformly-random numbers. For the sake of tractability, and in a manner that is consistent with all prior work studying strategic manipulations of consensus protocols, we consider a mathematical model based on idealized cryptographic primitives (i.e. that hashing a previously-unhashed input produces a uniformly random number, independent of all prior computed hashes).

**RANDAO Manipulation Game v1.** We now formally define the RANDAO Manipulation Game (v1). After defining the game, we note its connection to Ethereum’s RANDAO, and then proceed to simplify the game. Consistent with an overwhelming majority of prior work, we consider a single strategic manipulator optimizing against honest participants.<sup>7</sup>

► **Definition 5** (RANDAO Manipulation Game v1). *The RANDAO Manipulation Game proceeds in epochs  $1, 2, \dots, n, \dots$ . Each epoch has  $\ell := 32$  slots. The strategic player has an  $\alpha$  fraction of stake.*

- Initialize  $\text{REWARD} := 0$ .
- At all times, there is a RANDAO-generated list  $R := \langle R_1, \dots, R_{32} \rangle \in \{S, H\}^\ell$ .  $R$  denotes the list of  $\ell$  proposers based on the current value of RANDAO, and  $R_i$  denotes whether the strategic player ( $S$ ) or an honest player ( $H$ ) would propose in an epoch using the current value of RANDAO.
- Initialize  $R$  so that each coordinate of  $R$  is drawn iid, and equal to  $S$  with probability  $\alpha$ .
- For each epoch  $n := 1, \dots$ 
  - Store  $R^n := R$  and set the proposers for epoch  $n$  equal to  $R^n$ .
  - At all times during this epoch, for any set  $B$  of slots such that  $R_i^n = S$  for all  $i \in B$ , Strategic Player can compute  $R|_B$ , which represents how the list of 32 proposers would update if Strategic Player were to propose a block in exactly slots  $B$  and no other blocks are proposed, given that the current RANDAO induces  $R$ .
  - For each slot  $i := 1, \dots, \ell$ :
    - \* If  $R_i^n = H$ :
      - Update  $R$  to redraw each coordinate of  $R$  iid, and equal to  $S$  with probability  $\alpha$ .
      - For all  $B$ , update  $R|_B$  to redraw each coordinate of  $R$  iid, and equal to  $S$  with probability  $\alpha$ .
    - \* If  $R_i^n = S$ :
      - Strategic Player chooses whether to propose or not.
      - If they choose to propose: (a) Add +1 to  $\text{REWARD}$ , and (b) update  $R := R|_i$ , and  $R|_B := R|_{B \setminus \{i\}}$  for all  $B$ .
  - \* Store  $\text{REWARD}(n) := \text{REWARD}$ .

*Strategic Player’s reward is  $\liminf_{n \rightarrow \infty} \{\text{REWARD}(n)/(\ell n)\}$ .*

Let us now overview the game above, highlight why it captures RANDAO manipulation on Ethereum, and where we’ve made stylizing assumptions.

- First, observe that the epoch’s slot proposers are a deterministic function of RANDAO. We have skipped explicitly representing the RANDAO value, and focused only on the resulting proposers in  $R$ .

<sup>7</sup> An honest participant proposes a block during every round they are eligible.

## 10:8 Optimal RANDAO Manipulation in Ethereum

- Next, observe that every time RANDAO changes *due to an Honest digital signature*, we've randomly redrawn each proposer i.i.d. and equal to  $S$  with probability  $\alpha$ . This makes two stylizing assumptions.
  - First, we've assumed that uniformly RANDAO seed generates proposers iid proportional to stake. This may not be literally true, as Ethereum employs a more complicated sampling<sup>8</sup>. However, this stylizing assumption has negligible effect for the vast majority of real-world conditions<sup>9</sup>.
  - Second, we've assumed that the hash of an honest digital signature is distributed uniformly at random from the perspective of Strategic Player. This assumes an Ideal hash function and Ideal digital signature (as consistent with prior work [12, 11, 14]), although in practice it only holds that the distribution is indistinguishable from uniform to a computationally-bounded adversary.
- In our game, the RANDAO value relevant for epoch  $t$  is whatever RANDAO is at the end of epoch  $t - 1$ . In Ethereum proper, the RANDAO value relevant for epoch  $t$  is at the end of epoch  $t - 2$ . However, we claim our modeling choice is *almost* wlog. Specifically, observe that there are essentially two RANDAO Manipulation Games being played: one on odd epochs, and one on even epochs. That is, the RANDAO value at the end of epoch  $2t - 1$  determines the proposers for epoch  $2t + 1$  for all  $t$ , just as in our RANDAO Manipulation Game. The only distinction to our game is that the RANDAO value at the start of epoch  $2t + 1$  is *not* equal to the RANDAO value at the end of epoch  $2t - 1$  (whereas in our RANDAO Manipulation Game, it is) – the RANDAO value can change during round  $2t$ . However, *as long as there is at least one Honest proposer during round  $2t + 1$* , the RANDAO value at the start of epoch  $2t + 1$  doesn't matter anyway, because it will be reset to uniformly random (at least, from the Strategic Player's perspective).
- To elaborate on the previous bullet, *as long as an Honest player proposes in at least one slot in every epoch*, our RANDAO Manipulation Game v1 correctly models all odd Ethereum epochs, and separately correctly models all even Ethereum epochs.
- Finally, observe that our Strategic Player receives a reward of one exactly when they propose a block, and their reward is indeed equal to the time-averaged fraction of rounds in which they propose.
- To summarize, our stylized game captures RANDAO manipulation in Ethereum with three exceptions: (a) it assumes Ideal cryptography for simplicity of analysis, (b) it assumes proposers in each epoch are drawn i.i.d. proportional to fixed stake, (c) it assumes that every epoch contains at least one Honest proposer. (a) is a natural assumption consistent with prior works, and essentially abstracts strategic manipulation away from breaking cryptography. The impact of (b) is negligible, as the distinction with Ethereum's shuffling and iteration based approach is negligible for an essentially uniform<sup>10</sup> set of over one million validators. (c) is also negligible, as the probability that a Strategic Player could ever induce the next epoch to be the first with no Honest proposers is at most  $(2\alpha)^{32}$ .<sup>11</sup>

---

<sup>8</sup> Roughly speaking, for each slot Ethereum shuffles the set of active validators and starts iterating over the shuffled list. A validator is selected to be the proposer for this slot with probability equal to its effective balance over 32.

<sup>9</sup> As long as most validators have effective balance equal to the maximum, Ethereum essentially selects the proposer using a uniformly random sample. Ethereum's real world conditions match this assumptions since the vast majority of validators have maximum effective balance.

<sup>10</sup> Here, by uniform, we are referring to the *effective balance* of validators. For Ethereum's current validator set, almost all have maximum effective balance which is 32 ETH.

<sup>11</sup> To see this, observe that Strategic Player has at most  $2^{32}$  options to seed the subsequent epoch, and for each option the probability that it has no Honest proposers is  $\alpha^{32}$ . The calculation follows by a union bound. Observe that even for  $\alpha = 30\%$ , this is  $2^{-32}$ , meaning we would need to wait  $2^{32}$  epochs, or over 150 years.

**Manipulating RANDAO.** To build intuition, we first give an example of how and why one might manipulate RANDAO. First, imagine that the Strategic Player proposes in slots 25 and 30 during epoch 1. Observe that RANDAO will get reset to uniformly random during epoch 31, and the RANDAO value between rounds 25 and 30 has no impact on any future proposers. Therefore, the Strategic Player gains nothing by skipping these proposal slots.

On the other hand, imagine that the Strategic Player proposes in slots 31 and 32. The Strategic Player knows the RANDAO value going into slot 32, and therefore has two options to set the RANDAO for epoch 2. If they choose not to propose in slot 32, they miss out on a one-slot reward, but perhaps this leaves the RANDAO in a favorable place for epoch 2 as compared to the RANDAO value if they were to propose. In fact, the Strategic Player knows the RANDAO value going into slot 31, and has four choices between {propose twice, propose zero times, propose only in 31, propose only in 32}. Each of these will seed a different set of proposers for epoch 2, and forego a different number of rewards.

This example helps establish that the Strategic Player never benefits from foregoing a proposal before an Honest slot, but has  $2^k$  options for the next epoch's RANDAO when they propose the last  $k$  slots. We therefore call the largest number of slots  $k$  such that the adversary controls the last  $k$  slots of an epoch the *tail* of the epoch. The adversary can influence the next epoch only through these slots and intuitively these slots represent how much predictive power the adversary holds for the next epoch.

**Refining the RANDAO Manipulation Game.** Since the length of the tail fully captures the manipulation power of the Strategic Player, we further analyze this and refine our RANDAO Manipulation Game. We first observe that the length of the tail for a single RANDAO draw is distributed according to a roughly geometric distribution. A tail of length  $t$  occurs if we have  $t$  slots at the end of an epoch controlled by the strategic player which happens with probability  $\alpha^t$ , preceded by a single honest slot which happens with probability  $(1 - \alpha)$ . We call the remaining non-tail slots that the adversary controls the *count*. The count follows a binomial distribution conditioned on the length of the tail. Specifically:

- $geom'(\alpha)$  is the distribution of the tail given an adversary with stake  $\alpha$ . It is defined such that for  $T \sim geom'(\alpha)$ ,

$$\Pr(T = t) = \begin{cases} (1 - \alpha)\alpha^t & 0 \leq t < \ell \\ \alpha^\ell & t = \ell \end{cases}$$

- $Binom'(\ell - t - 1, \alpha)$  is the distribution of the remaining count (how many slots the adversary gets from the non-tail part of the epoch) given that the tail is  $t$ . For  $C \sim Binom'(\ell - t - 1, \alpha)$ ,

$$\Pr(C = c) = \begin{cases} \binom{\ell-t-1}{c} \alpha^c (1 - \alpha)^{\ell-t-1-c} & 0 \leq c < \ell - t \wedge 0 \leq t < \ell - 1 \\ 1 & c = 0 \wedge (t = \ell - 1 \vee t = \ell) \\ 0 & c \neq 0 \wedge (t = \ell - 1 \vee t = \ell) \end{cases}$$

- $\mathcal{F}$  is the distribution of  $(C, T)$  where we first sample  $T \sim geom'(\alpha)$  and then sample  $C \sim Binom'(\ell - T - 1, \alpha)$ .

Given our reasoning above, an optimal Strategic Player will always propose during any of the “count” rounds, and will only manipulate the “tail” rounds. In particular, this means that the Strategic Player need not know the full slate of proposers in an epoch, but *only the count and the tail*. With this in mind, we can now refine our RANDAO Manipulation Game v1 to an equivalent RANDAO Manipulation Game v2.

Using the definitions above, formally, the RANDAO manipulation game  $G$  is:

## 10:10 Optimal RANDAO Manipulation in Ethereum

► **Definition 6** (RANDAO Manipulation Game v2).

1. Initialize  $REWARD := 0$ , and  $ROUNDS := 0$ .
2. Initialize  $(c, t)$  drawn from  $\mathcal{F}$ .
3. For  $i = 1$  to  $t$ ,
  - For  $j = 1$  to  $\binom{t}{i}$ ,
    - Sample  $(c_{i,j}, t_{i,j})$  from  $\mathcal{F}$ .
4. The Strategic Player chooses an  $(i^*, j^*)$  pair.
5. Update  $t := t_{i^*, j^*}$ , add  $c_{i^*, j^*} + t_{i^*, j^*} - i^*$  to  $REWARD$  and  $\ell$  to  $ROUNDS$ .
6. Repeat from step 3.

The final payoff is  $\liminf\{REWARD/ROUNDS\}$ .

RANDAO Manipulation Game v2 is equivalent to RANDAO Manipulation Game v1, after assuming that the Strategic Player optimally proposes during all non-tail slots. Our method of counting the rewards observes that in the next epoch we will always propose during the “count” rounds (and hence just add them directly to our reward), and miss  $i$  tail proposals in order to influence the RANDAO (and hence get only  $t_{i,j} - i$  slot rewards from the tail).

Before we proceed with the analysis, we define two sets of interest. Let  $\mathcal{O}$  be the set of all possible values of  $(C_{i,j} - i, T_{i,j})$  when  $(C_{i,j}, T_{i,j})$  gets sampled from  $\mathcal{F}$  for all  $0 \leq i \leq t$ ,  $1 \leq j \leq \binom{t}{i}$  for some current tail  $t \in \{0, \dots, \ell\}$ . Given the range of the tail, count and the epoch length,  $\mathcal{O} = \{(\omega, t) : t \in [0..l] \wedge \omega \in [-l..l] \wedge ((t = l \wedge \omega \leq 0) \vee (t < l \wedge \omega \leq l - t - 1))\}$ . Let  $\Omega$  be the set of all possible multisets of observations. More formally,  $\Omega$  is the set of all multisets  $\{(C_{i,j} - i, T_{i,j}) : 0 \leq i \leq \ell, 1 \leq j \leq \binom{t}{i}\}$  for some tail length  $t$ . Note that this makes all observation multisets  $Obs \in \Omega$  have size equal to some power of 2.

### 4 MDP formulation

We can now directly formulate the RANDAO manipulation game as an MDP given the RANDAO Manipulation Game v2.

► **Definition 7** (RANDAO MDP  $M_G$ ). The Markov decision process representing the RANDAO manipulation game is  $M_G = (S, A, \{P_\pi\}_\pi, \{R_\pi\}_\pi)$  where

- $S = \{(t, Obs) : t \in \mathbb{N}, 0 \leq t \leq \ell, Obs \in \Omega\}$ . Each state represents the length of the tail  $t$  and the observations available to the adversary corresponding to the RANDAO samples.
- The action space  $A = \mathcal{O}$ , each action is selecting a future state from the given observations.
- $\pi \in \Pi$  is the policy space where  $\Pi$  is the set of all functions  $\pi : \Omega \rightarrow \mathcal{O}$  such that  $\pi(Obs) = (\omega, t) \in Obs$ .  $\pi$  chooses one of the sample in  $Obs$  to transition towards.
- $P_\pi$  and  $R_\pi$  are determined by the process in the game where given a state  $(t, Obs)$ , we transition to  $(t', Obs')$  such that  $\pi(Obs) = (\omega', t')$  and  $Obs'$  consists of  $2^{t'}$  pairs  $(c_{i,j} - i, t_{i,j})$  each sampled using  $\mathcal{F}$  as in step 2 of the game. The reward of this transition is  $\omega' + t'$ .

► **Lemma 8.**  $M_G$  is ergodic.

**Proof.** It suffices to observe for any policy  $\pi$ , we can transition from any state to any other state in three steps with non-zero probability. Consider the Markov chain induced by fixing  $\pi$  in MDP  $M_G$ . Suppose we are at state  $(t, Obs)$  and we consider  $(t', Obs')$ . Let  $t^* = \log(|Obs'|)$ . We then observe that the following sequence of transitions have non-zero probability:

$$(t, Obs) \rightarrow (t_1, Obs_1) \rightarrow (t^*, Obs_2) \rightarrow (t', Obs')$$

where  $Obs_1$  is the set of observations where all have tail equal to  $t^*$  and  $Obs_2$  is the set of observations where all have tail equal to  $t'$ . ◀

It is fairly straight-forward to see that this MDP formulation accurately captures the RANDAO Manipulation Game v2 – the state captures the point in time after drawing  $(c_{i,j}, t_{i,j})$  for all  $(i, j)$ , and our only action at this point is to choose one such  $(i, j)$  and transition, getting reward REWARD. Note also that every state transition corresponds to exactly an increase of  $\ell$  in ROUNDS, so the time-averaged reward in this MDP is exactly the payoff in RANDAO Manipulation Game v2.

Unfortunately, the state space of this MDP is enormous, and we have absolutely no hope of even writing it down (let alone solving it). Luckily since our MDP is ergodic, we can exploit the structure of the optimal policy and drastically simplify the state space.

### Reducing the state space

We now refine our formulation of the RANDAO manipulation MDP to make it tractable. We know that for each policy  $\pi$ , there exists a valuation  $v_\pi : \{0, \dots, \ell\} \rightarrow \mathbb{R}$ , the *average adjusted sum of rewards*. We also know from the Bellman optimality equation that the optimal policy will take the action maximizing the immediate reward plus the weighted sum of potential future states with their values. Hence, any plausibly optimal policy, given the set of samples  $\{(c_{i,j}, t_{i,j}) : 0 \leq i \leq t, 1 \leq j \leq \binom{t}{i}\}$ , will simply choose the one that maximizes  $c_{i,j} - i + t_{i,j} + v_\pi(t_{i,j})$ . Motivated by this observation we reformulate the RANDAO MDP  $M_G$  as the following reduced state space MDP  $M'_G$ .

Below, intuitively we no longer need to explicitly store all (count, tail) options, because any optimal policy can be fully specified by assigning a value to the tail. So our new state space is simply the tail, but it is now more complex to iterate a transition.

► **Definition 9** (RANDAO MDP  $M'_G$ ). *The reduced Markov decision process representing the RANDAO manipulation game is  $M'_G = (S, \Pi, \{P_\pi\}_\pi, \{R_\pi\}_\pi)$  where*

- $S = \{t \in \mathbb{N} : 0 \leq t \leq \ell\}$ . *Each state represents the length of the tail.*
- $\Pi$  *is the policy space where  $\Pi$  is the set of all total orders on  $\mathcal{O}$ .*
- *We treat the action space as the same as the policy space. In other words, we only consider constant strategies that pick a total order on  $\mathcal{O}$ .*
- $P_\pi$  *and  $R_\pi$  are determined as follows. Follow RANDAO Manipulation Game v2 in Steps 3-4 (drawing several  $(c, t)$ s and choosing one), where in Step 4 we choose the future state that is earliest in the total order according to  $\pi$ . We then transition according to the selected  $t$  (this defines  $P_\pi$ ) and accumulate reward according to  $c + t - i$  (this defines  $R_\pi$ ).<sup>12</sup>*

Intuitively, the key difference between  $M_G$  and  $M'_G$  is at which point in the process we pause and determine a state. In  $M_G$ , we pause after seeing a large set of (count, tail) pairs and declare this a state. We then make a very simple decision (pick a pair), a very simple reward update (plus count, plus tail, minus number of missed slots), and a fairly simple state transition (draw the new collection of pairs from a known distribution based on the chosen tail).

In  $M'_G$ , we instead pause after selecting the tail, and declare this a state. We then make a complex decision (pick a total ordering over all plausible pairs), a complex and randomized reward update (sample the set of pairs according to the known distribution based on the state, pick the highest in the total order, and take the reward), and a complex state transition (sample the set of pairs according to the known distribution based on the state, pick the

<sup>12</sup>In Section 5, we explicitly describe how these transition probabilities and rewards are computed.

## 10:12 Optimal RANDAO Manipulation in Ethereum

highest in the total order, and take the reward). That is,  $M_G$  has a very complicated state space but simple transitions, whereas  $M'_G$  has a very complicated action space but simple states. Moreover, we use the Bellman optimality principle to narrow down the plausibly optimal actions for consideration in  $M'_G$ . We now proceed to establish their equivalence formally.

► **Lemma 10.**  $M'_G$  is ergodic.

**Proof.** It suffices to observe that under any policy  $\pi \in \Pi$  the transition probability between any pair of states is positive. Suppose that we are running policy  $\pi$  and are at state  $t \in S$  and we consider destination tail  $t' \in S$ . Now, if all  $2^t$  sampled future states have tail  $t'$ , the next state is  $t'$ . Since this happens with small but non-zero probability, ergodicity holds. ◀

While we are no longer explicitly keeping track of individual observations in the state, the optimal policy for  $M'_G$  still achieves reward equal to the optimal reward in  $M_G$  (in expectation).

► **Proposition 11.** If  $\pi \in \Pi$  is an optimal policy for  $M_G$  and  $\pi' \in \Pi'$  is an optimal policy for  $M'_G$ , then  $\Gamma_\pi(M_G) = \Gamma_{\pi'}(M'_G)$ .

**Proof.** Suppose  $\pi \in \Pi$  is an optimal policy for  $M_G$  and  $\pi' \in \Pi'$  is an optimal policy for  $M'_G$ . We first show that given  $\pi \in \Pi$  for  $M_G$ , there exists a corresponding policy  $\pi^* \in \Pi'$  for  $M'_G$  such that  $\Gamma_\pi(M_G) = \Gamma_{\pi^*}(M'_G)$ . Since every optimal policy attains the same expected average reward, without loss of generality, assume that  $\pi$  satisfies the Bellman optimality equation. As a consequence,  $\pi$  selects the observation  $(\omega, t)$  that maximizes  $\omega + t + v_\pi(t)$ . This precisely defines a total order on  $(\omega, t)$  as  $v_\pi$  is fixed. Let  $\pi^*$  be the total order defined by maximizing  $\omega + t + v_\pi(t)$ . As both  $M_G$  and  $M'_G$  sample from the same distributions the same number of times,  $\Gamma_\pi(M_G) = \Gamma_{\pi^*}(M'_G)$ .

Hence, we know that given  $\pi_0^*$ , there exists a corresponding policy  $\pi_0^{*'}$  playing  $M'_G$  such that  $\Gamma_{\pi_0^*}(M_G) \leq \Gamma_{\pi_0^{*'}}(M'_G)$ . By the optimality of  $\pi_1^*$ , we also know that  $\Gamma_{\pi_0^{*'}}(M'_G) \leq \Gamma_{\pi_1^*}(M'_G)$ . Therefore,  $\Gamma_{\pi_0^*}(M_G) = \Gamma_{\pi_1^*}(M'_G)$  and the claim holds. ◀

## 5 Evaluating policies

In this section, we first analyze the policy that only cares about maximizing the tail length (TAIL-MAX) as an instructive example. Intuitively, this policy can be implemented by (a) for each subset of slots that the Strategic Player can choose to propose, computing the resulting RANDAO value and hence the next epoch proposer assignments, and (b) the Strategic Player choosing to propose in the subset of slots that results in the longest tail in the next epoch. Subsequently, we describe how to evaluate arbitrary policies in our Markov decision process  $M'_G$  formulation of the RANDAO manipulation game.

### 5.1 Analyzing the Tail-max policy

The TAIL-MAX policy can be defined as the policy  $\pi$  that given the current state  $t$  and  $2^t$  samples  $(C_{i,j}, T_{i,j}) \sim \mathcal{F}$ , picks  $(i, j)$  that maximizes  $T_{i,j}$ . Note that in case of ties, we pick the transition with higher reward (i.e. breaking ties in favor of higher  $C_{i,j} - i$ ).

To analyze TAIL-MAX, we are interested in computing the following quantities.

- $P_{tail}(t, t')$ : the probability of transitioning from state  $t$  to  $t'$  when running game  $G$  with TAIL-MAX.
- $R_{tail}(t)$ : the expected reward of transitioning from state  $t$  when running game  $G$  with TAIL-MAX.

Let  $\maxTail(t)$  be a random variable representing the maximum sampled tail in the current round of the game. More formally, it is defined as the following:

$$\maxTail(t) := \max_{\substack{(C_{i,j}, T_{i,j}) \sim \mathcal{F} \\ 0 \leq i \leq t \\ 1 \leq j \leq \binom{t}{i}}} \{T_{i,j}\}$$

which is identically distributed as  $\max_{\substack{T_i \sim \text{geom}'(\alpha) \\ 0 \leq i \leq 2^t - 1}} \{T_i\}$ . Then, we have the following probabilities.

$$\begin{aligned} P_{\text{tail}}(t, t') &= \Pr(\maxTail(t) = t') \\ &= \begin{cases} \Pr(\maxTail(t) \leq t') - \Pr(\maxTail(t) \leq t' - 1) & 0 < t' \leq \ell \\ \Pr(\maxTail(t) \leq t') & t' = 0 \end{cases} \end{aligned}$$

where we use the following:

► **Lemma 12.**  $\Pr(\maxTail(t) \leq T') = \begin{cases} (1 - \alpha^{t'+1})^{2^t} & 0 \leq t' < \ell \\ 1 & t' = \ell \end{cases}$

**Proof.** Using the fact that each  $\{T_i\}$  are i.i.d.,

$$\begin{aligned} \Pr(\maxTail(t) \leq T') &= \Pr_{\substack{T_i \sim \text{geom}'(\alpha) \\ 0 \leq i < 2^t}} \left( \bigwedge_{0 \leq i < 2^t} (T_i \leq t') \right) \\ &= \prod_{0 \leq i < 2^t} \Pr_{T_i \sim \text{geom}'(\alpha)} (T_i \leq t') \\ &= \left( \Pr_{T'' \sim \text{geom}'(\alpha)} (T'' \leq t') \right)^{2^t} \\ &= \begin{cases} (1 - \alpha^{t'+1})^{2^t} & 0 \leq t' < \ell \\ 1 & t' = \ell \end{cases} \quad \blacktriangleleft \end{aligned}$$

The transition reward can be computed in a similar way. Let  $\preceq \in \Pi$  of  $M'_G$  be defined as  $(t, v) \preceq (t', v')$  if and only if  $t < t'$  or  $(t = t' \wedge v \leq v')$ . Intuitively, these pairs represent choices that a policy can make where  $t$  is the tail and  $v$  is the amount of reward we get from the rest of the count. The TAIL-MAX policy picks the maximum such pair according to  $\preceq$ . Equality and strict ordering are defined in the usual way. Also let  $\text{prev}_{\preceq}(v, t)$  be defined as the previous pair in the ordering  $\preceq$  if it exists and  $\perp$  otherwise. Note that  $\Pr(\cdot \preceq \perp)$  is interpreted as 0.

Let  $\maxPair_{\preceq}(t)$  be a random variable representing the maximum tail, and the non-tail reward pair in the current round of the game according to the total order  $\preceq$ . More formally, it is defined as the following:

$$\maxPair_{\preceq}(t) := \max_{\substack{(C_{i,j}, T_{i,j}) \sim \mathcal{F} \\ 0 \leq i \leq t \\ 1 \leq j \leq \binom{t}{i}}} \{(T_{i,j}, C_{i,j} - i)\}$$

## 10:14 Optimal RANDAO Manipulation in Ethereum

Note that  $\preceq$  is a total order and we have the property that  $\text{maxPair}_{\preceq}(t) \preceq (a, b)$  if and only if  $(T_{i,j}, C_{i,j} - i) \preceq (a, b)$  for all  $0 \leq i \leq t, 0 \leq j \leq \binom{t}{i}$ . Therefore,

$$\begin{aligned} R_{\text{tail}}(t) &= \mathbb{E} [T' + V' \mid \text{maxPair}_{\preceq}(t) = (T', V')] \\ &= \sum_{\substack{0 \leq t' \leq \ell \\ -t \leq v' \leq \ell - t'}} \Pr(\text{maxPair}_{\preceq}(t) = (t', v'))(t' + v') \\ &= \sum_{\substack{0 \leq t' \leq \ell \\ -t \leq v' \leq \ell - t'}} (\Pr(\text{maxPair}_{\preceq}(t) \preceq (t', v)) - \Pr(\text{maxPair}_{\preceq}(t) \preceq \text{prev}_{\preceq}(t', v)))(t' + v) \end{aligned}$$

where we can use the following:

► **Lemma 13.** Let  $\preceq \in \Pi$  of  $M_G^t$  be the TAIL-MAX policy. For  $f(x) := \Pr_{T \sim \text{geom}'(\alpha)}(T < x)$ ,  $g(x) := \Pr_{T \sim \text{geom}'(\alpha)}(T = x)$ , and  $h(x, y) := \Pr_{C \sim \text{Binom}(\ell - x - 1, \alpha)}(C \leq y)$ ,

$$\Pr(\text{maxPair}_{\preceq}(t) \preceq (t', v)) = \prod_{0 \leq i \leq t} \left( \Pr_{(C,T) \sim \mathcal{F}}((T, C - i) \preceq (t', v)) \right)^{\binom{t}{i}}$$

and

$$\Pr_{(C,T) \sim \mathcal{F}}((T, C - i) \preceq (t', v)) = \begin{cases} f(t') & t' = \ell \wedge 0 > v' + i \\ f(t') + g(\ell) & t' = \ell \wedge 0 \leq v' + i \\ f(t') + g(t')h(t', v' + i) & \text{otherwise} \end{cases}$$

**Proof.** Using standard properties, we observe that

$$\begin{aligned} \Pr(\text{maxPair}_{\preceq}(t) \preceq (t', v)) &= \Pr_{\substack{(C_{i,j}, T_{i,j}) \sim \mathcal{F} \\ 0 \leq i \leq t \\ 1 \leq j \leq \binom{t}{i}}} \left( \bigwedge_{\substack{0 \leq i \leq t \\ 1 \leq j \leq \binom{t}{i}}} ((T_{i,j}, C_{i,j} - i) \preceq (t', v)) \right) \\ &= \prod_{\substack{0 \leq i \leq t \\ 0 \leq j \leq \binom{t}{i}}} \Pr_{(C,T) \sim \mathcal{F}}((T, C - i) \preceq (t', v)) \\ &= \prod_{0 \leq i \leq t} \left( \Pr_{(C,T) \sim \mathcal{F}}((T, C - i) \preceq (t', v)) \right)^{\binom{t}{i}} \end{aligned}$$

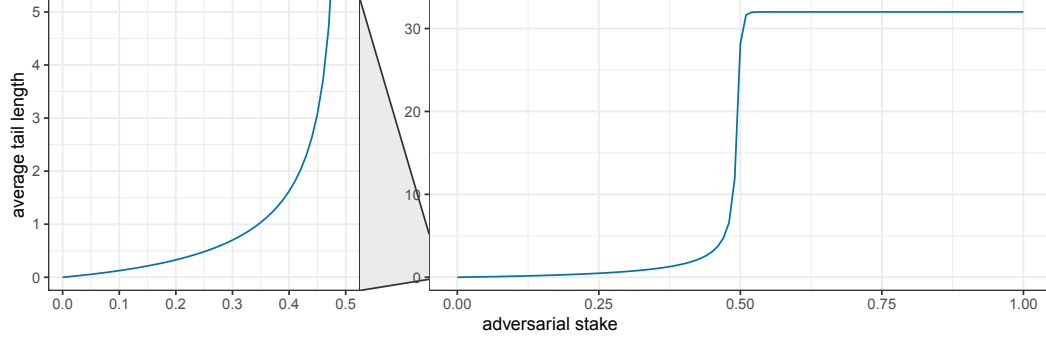
and

$$\begin{aligned} &\Pr_{(C,T) \sim \mathcal{F}}((T, C - i) \preceq (t', v)) \\ &= \Pr_{(C,T) \sim \mathcal{F}}(T < t' \vee (T = t' \wedge C - i \leq v')) \\ &= \Pr_{T \sim \text{geom}'(\alpha)}(T < t') + \Pr_{(C,T) \sim \mathcal{F}}(T = t' \wedge C \leq v' + i) \\ &= \Pr_{T \sim \text{geom}'(\alpha)}(T < t') + \Pr_{T \sim \text{geom}'(\alpha)}(T = t') \Pr_{(C,T) \sim \mathcal{F}}(C \leq v' + i \mid T = t') \\ &= \Pr_{T \sim \text{geom}'(\alpha)}(T < t') \\ &+ \begin{cases} 0 & t' = \ell \wedge 0 > v' + i \\ \Pr_{T \sim \text{geom}'(\alpha)}(T = \ell) & t' = \ell \wedge 0 \leq v' + i \\ \Pr_{T \sim \text{geom}'(\alpha)}(T = t') \Pr_{C \sim \text{Binom}(\ell - t' - 1, \alpha)}(C \leq v' + i) & \text{otherwise} \end{cases} \end{aligned}$$

◀



We can now compute the stationary distribution in order to directly compute average reward of the TAIL-MAX policy. This is a lower bound to the optimal reward ratio we can obtain from this game.<sup>13</sup>



■ **Figure 1** Average tail length attained for each  $\alpha$  when running TAIL-MAX. The adversary controls a larger tail value as  $\alpha$  rises as expected. There is a quick jump when approaching  $\alpha = 50\%$ , indicating that the adversary can propose almost all blocks.

## 5.2 Policy evaluation in the general case

We proceed similar to the TAIL-MAX analysis. Consider policy  $\preceq \in \Pi$  so it is some total order on  $\mathcal{O}$ . Recall that  $\maxPair_{\preceq}(t)$  is a random variable defined (in Subsection 5.1) to be the maximum (tail, non-tail reward) pair given that the adversary currently controls a tail of length  $t$ . Similar to the analysis of TAIL-MAX, we then have

$$P_{\preceq}(t, t') = \sum_{-\ell \leq v \leq \ell - t'} \Pr(\maxPair_{\preceq}(t) = (t', v))$$

$$R_{\preceq}(t) = \sum_{\substack{0 \leq t' \leq \ell \\ -t \leq v' \leq \ell - t'}} \Pr(\maxPair_{\preceq}(t) = (t', v))(t' + v)$$

Now, it suffices to describe how to compute the CDF of  $\maxPair_{\preceq}(t)$  since

$$\Pr(\maxPair_{\preceq}(t) = (t', v)) = \Pr(\maxPair_{\preceq}(t) \preceq (t', v)) - \Pr(\maxPair_{\preceq}(t) \preceq \text{prev}_{\preceq}(t', v))$$

► **Lemma 14.** *Let  $\preceq \in \Pi$  be an arbitrary policy in  $M'_G$ .*

$$\Pr(\maxPair_{\preceq}(t) \preceq (v, t')) = \prod_{0 \leq i \leq t} \left( \sum_{\forall (t^*, v^*) \preceq (t', v)} \Pr_{(C, T) \sim \mathcal{F}}((T, C) = (t^*, v^* + i)) \right)^{\binom{t'}{i}}$$

<sup>13</sup>Note that TAIL-MAX does not necessarily outperform Honest – it could be that in an attempt to increase the tail by one, TAIL-MAX misses several proposal slots, and yet also does not take good advantage of the increased tail. However, our results show that TAIL-MAX does outperform the honest strategy for all  $\alpha$ . See Figure 2 and 3 for a comparison with Honest and the optimal policy. The average tail length of TAIL-MAX, however, serves as an upper bound on the average tail length of any feasible strategy, including the optimum.

## 10:16 Optimal RANDAO Manipulation in Ethereum

**Proof.** Using standard properties of independent samples, we observe that:

$$\begin{aligned}
\Pr(\text{maxPair}_{\preceq}(t) \preceq (v, t')) &= \Pr_{\substack{(C_{i,j}, T_{i,j}) \sim \mathcal{F} \\ 0 \leq i \leq t \\ 1 \leq j \leq \binom{t}{i}}} \left( \bigwedge_{\substack{0 \leq i \leq t \\ 1 \leq j \leq \binom{t}{i}}} ((T_{i,j}, C_{i,j} - i) \preceq (t', v)) \right) \\
&= \prod_{\substack{0 \leq i \leq t \\ 1 \leq j \leq \binom{t}{i}}} \Pr_{(C,T) \sim \mathcal{F}} ((T, C - i) \preceq (t', v)) \\
&= \prod_{\substack{0 \leq i \leq t \\ 1 \leq j \leq \binom{t}{i}}} \sum_{\forall (t^*, v^*) \preceq (t', v)} \Pr_{(C,T) \sim \mathcal{F}} ((T, C - i) = (t^*, v^*)) \\
&= \prod_{0 \leq i \leq t} \left( \sum_{\forall (t^*, v^*) \preceq (t', v)} \Pr_{(C,T) \sim \mathcal{F}} ((T, C) = (t^*, v^* + i)) \right)^{\binom{t'}{i}} \blacktriangleleft
\end{aligned}$$

Therefore we can also evaluate arbitrary policies by computing the quantities above.

### 6 Solving for optimal strategies

We can now run policy iteration [18] in our policy space.

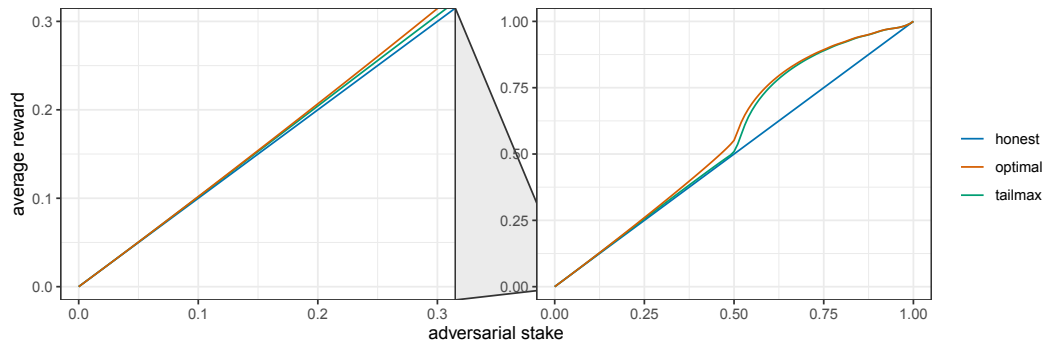
1. Start with an arbitrary policy  $\preceq \in \Pi$ .
2. Compute  $P_{\preceq}$  and  $R_{\preceq}$ .
3. Compute average reward  $\Gamma_{\preceq}$  using  $R_{\preceq}(t)$  and the stationary distribution of  $P_{\preceq}$ .
4. Determine  $v_t$  for each  $t \in \{0, \dots, \ell\}$  by setting  $v_0 = 0$  and solving the system of linear equations given by

$$\Gamma_{\preceq} + v_{\preceq}(t) = R_{\preceq}(t) + \sum_{t'=0}^{\ell} P_{\preceq}(t, t') v_{t'} \quad \text{for } t \in \{0, \dots, \ell\}.$$

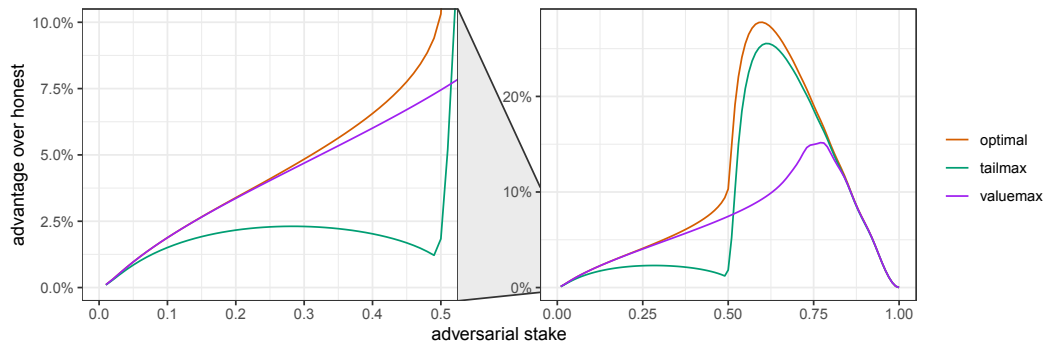
5. Let  $\preceq'$  be a new total order constructed by sorting each  $(\omega, t)$  pair by the quantities  $\omega + t + v_t$  in ascending order.
6. If  $\preceq$  is equal to  $\preceq'$ , we converged and  $\preceq$  is optimal. Otherwise let  $\preceq := \preceq'$  and repeat from step 2.

Note that this is equivalent to policy iteration as described in Section 2 since sorting by immediate reward plus future state value to get new policy  $\preceq$  is equivalent to picking the maximum at each state.

For  $\ell = 32$  and all  $\alpha$ , policy iteration always converged in less than 10 steps. This enables us to plot the following results on optimal manipulations for RANDAO:



■ **Figure 2** Average reward of the optimal policy and TAIL-MAX for  $\ell = 32$ . The figure on the left shows the  $0 < \alpha \leq 0.3$  range, the figure on the right show the entire range of  $0 < \alpha < 1$ .



■ **Figure 3** Percentage improvement of the optimal policy and TAIL-MAX over the honest policy for  $\ell = 32$ . Improvement is defined as  $(\text{policy average reward})/(\text{honest average reward}) - 1$ . We also analyze the strategy VALUE-MAX here which we define as the strategy that maximizes the reward in the next epoch (chooses the pair that maximizes  $c_{i,j} + t_{i,j} - i$ ).

■ **Table 1** Average reward of the optimal policy. In expectation, the honest reward is equal to  $\alpha$ . We see in the table that the optimal policy is strictly more profitable.

$\alpha$	optimal reward
1%	1.00107%
5%	5.04834%
10%	10.18807%
15%	15.39960%
20%	20.67770%
25%	26.02472%
30%	31.45164%
35%	36.97348%
40%	42.62435%
45%	48.49184%

## 10:18 Optimal RANDAO Manipulation in Ethereum

A key strength of our methodology is the fact that it is constructive, we explicitly construct the optimal policy for each  $\alpha$ . In order to implement the optimal policy we compute, the values can be used as in step 5 of the policy iteration routine described above. For instance, for  $\alpha = 0.2$ , the optimal policy assigns the the following values to tails of length 0 to 32 (rounded to two decimal points):

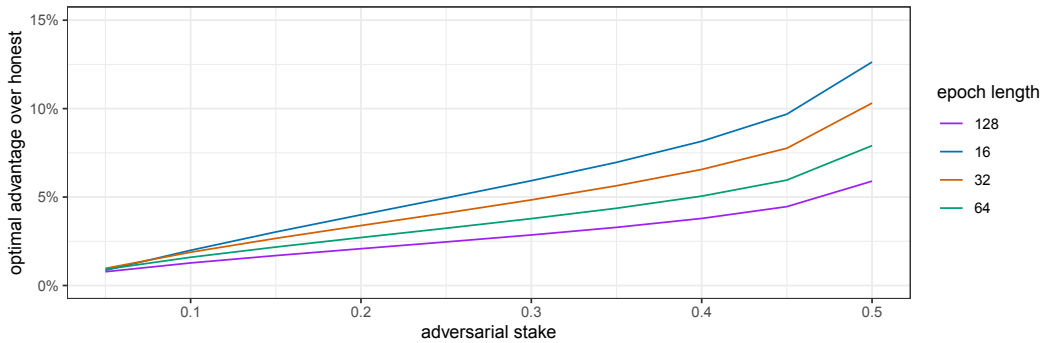
(0.00, 0.90, 1.66, 2.30, 2.86, 3.35, 3.79, 4.19, 4.55, 4.89, 5.21,  
5.50, 5.78, 6.05, 6.30, 6.55, 6.78, 7.00, 7.22, 7.43, 7.63, 7.82,  
8.01, 8.19, 8.37, 8.54, 8.71, 8.87, 9.03, 9.19, 9.34, 9.49, 9.64)

### Considering $\ell \neq 32$

One benefit of our approach is that it trivially extends to any  $\ell$ . This allows one to easily answer, for example, whether RANDAO would be more or less manipulable with different epoch lengths and by how much.

Policy iteration runs smoothly for  $\ell = 32$  and smaller. However, when we consider  $\ell = 64$  or 128, numerical instability becomes a concern, and our experiments are no longer *provably* accurate. In particular, 64-bit floats we use in our machine introduce precision error that explode when evaluating the expression in Lemma 14 with  $\ell > 32$ .

In order to improve the numerical stability of the expression in Lemma 14, when  $\ell > 32$  we evaluate the inner sum directly to 1 instead of taking the exponential when the sum reaches within  $10^{-14}$  of 1 since the error  $\epsilon$  introduced by the floating point representation cause  $(1 \pm \epsilon)^N$  to become 0 or a large constant for  $N \gg 1$ . The following figure shows running our evaluation for different  $\ell$ , using this modification.



■ **Figure 4** Percentage improvement over honest for  $\ell \in \{16, 32, 64, 128\}$ . Improvement is defined as  $(\text{optimal average reward})/(\text{honest average reward}) - 1$ .

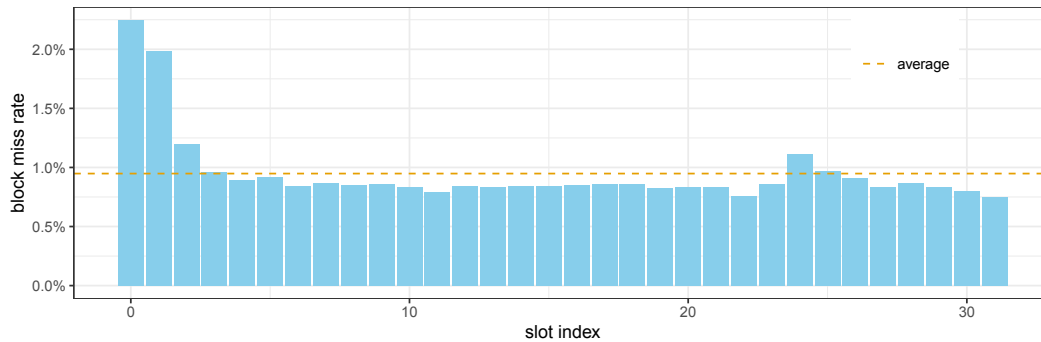
We conjecture that this plot is representative of how the results scale with  $\ell$ , although unlike our main results the experiments are not provably accurate due to the aforementioned numerical instability. If one desires provable numerical guarantees on these quantities, one would need an analysis of numerical error induced by floating point representations of the machines that run the evaluation.

## 7 Discussion

We model optimal RANDAO manipulation in Proof-of-Stake Ethereum and frame it as an MDP. Our main modeling contribution is getting from RANDAO manipulation in practice to RANDAO Manipulation Game v2, and our key technical insight is getting from there to the reduced RANDAO MDP  $M'_G$ . From here, simple policy iteration on a laptop suffices to analyze the optimal strategy. Our main results shed light on exactly how manipulable Ethereum’s RANDAO is. One could compare our results, for example, to those of [14] for Proof-of-Stake protocols based on cryptographic self-selection. For example, [14] establishes that a well-connected Strategic Player with 10% of the stake can propose between 10.08% and 10.15% of the rounds in cryptographic self-selection protocols, and our work establishes that a Strategic Player with 10% of the stake can propose a 10.19% fraction of rounds in Ethereum Proof-of-Stake. While our work introduces methodology to compute these numbers, we leave *interpretation* of their significance to the Ethereum community since many different factors come into play when designing the consensus mechanism.

A clear direction for future work would be to consider the impact of slot-varying rewards as in [8]. This will clearly increase the manipulability (as now the Strategic Player can use the value of a slot when deciding whether to propose), but it is not obvious by how much. A second direction would be to consider the impact of idiosyncratic details such as Ethereum’s sync committees (extra rewards every 256 blocks).

Lastly, we briefly discuss the empirical signature of randomness manipulation. The results immediately lead to the following question: are there any entities currently manipulating the RANDAO value? The signature of such an attack would affect the block miss rates especially around the tail. Some prior analyses suggest that while there has been ample opportunities that would result in short term gains for certain entities, none have been observed to manipulate RANDAO [23]. In the figure below, the block miss rates by epoch slot index is displayed from epoch 146876 to 272341.



■ **Figure 5** Block miss rate by slot index from epoch 146876 to 272341. The average block miss rate is 0.9482%.

Our interpretation of this data is that slot index 0, 1, and 2 is missed frequently since validators have less time to react to their proposer assignments and slot index 24 and 25 is missed with slightly higher frequency than the baseline due to votes crossing the 2/3 majority. We do not observe any significant elevation in block miss rates around the tail of the epoch. It is also an interesting direction for future work to examine whether *undetectable* profitable strategies exist for RANDAO manipulation (i.e. strategies that strictly outperform Honest, but produce the same miss rate for all slots).

---

**References**

---

- 1 Randao: A dao working as rng of ethereum, March 2019. URL: <https://github.com/randao/randao/>.
- 2 Musab A Alturki and Grigore Roşu. Statistical model checking of randao’s resilience to pre-computed reveal strategies. In *Formal Methods. FM 2019 International Workshops*, pages 337–349, Porto, Portugal, 2020. Springer.
- 3 Maryam Bahrani and S. Matthew Weinberg. Undetectable selfish mining. In *EC ’24: The 25th ACM Conference on Economics and Computation*. ACM, 2024.
- 4 Jonah Brown-Cohen, Arvind Narayanan, Alexandros Psomas, and S. Matthew Weinberg. Formal barriers to longest-chain proof-of-stake protocols. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, EC ’19, pages 459–473, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3328526.3329567.
- 5 Vitalik Buterin. Randao beacon exploitability analysis, round 2, May 2018. URL: <https://ethresear.ch/t/randao-beacon-exploitability-analysis-round-2/1980>.
- 6 Vitalik Buterin. Rng exploitability analysis assuming pure randao-based main chain, April 2018. URL: <https://ethresear.ch/t/rng-exploitability-analysis-assuming-pure-randao-based-main-chain/1825/1>.
- 7 Vitalik Buterin. Vitalik’s annotated ethereum 2.0 spec, 2020. URL: <https://notes.ethereum.org/@vbuterin/SkeyEI3xv#Time-parameters>.
- 8 Miles Carlsten, Harry A. Kalodner, S. Matthew Weinberg, and Arvind Narayanan. On the instability of bitcoin without the block reward. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 154–167, 2016. doi:10.1145/2976749.2978408.
- 9 Ben Edgington. Upgrading ethereum, 2023. Capella edition. URL: <https://eth2book.info/>.
- 10 Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography and Data Security*, pages 436–454. Springer, 2014.
- 11 Matheus V. X. Ferreira, Ye Lin Sally Hahn, S. Matthew Weinberg, and Catherine Yu. Optimal strategic mining against cryptographic self-selection in proof-of-stake. In David M. Pennock, Ilya Segal, and Sven Seuken, editors, *EC ’22: The 23rd ACM Conference on Economics and Computation, Boulder, CO, USA, July 11 - 15, 2022*, pages 89–114. ACM, 2022. doi:10.1145/3490486.3538337.
- 12 Matheus V. X. Ferreira and S. Matthew Weinberg. Credible, truthful, and two-round (optimal) auctions via cryptographic commitments. In Péter Biró, Jason D. Hartline, Michael Ostrovsky, and Ariel D. Procaccia, editors, *EC ’20: The 21st ACM Conference on Economics and Computation, Virtual Event, Hungary, July 13-17, 2020*, pages 683–712. ACM, 2020. doi:10.1145/3391403.3399495.
- 13 Matheus V. X. Ferreira and S. Matthew Weinberg. Proof-of-stake mining games with perfect randomness. In *Proceedings of the 22nd ACM Conference on Economics and Computation*, EC ’21, pages 433–453, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3465456.3467636.
- 14 Matheus V.X. Ferreira, Aadityan Ganesh, Jack Hourigan, Hannah Hu, S. Matthew Weinberg, and Catherine Yu. Computing optimal manipulations in cryptographic self-selection proof-of-stake protocols. In *EC ’24: The 25th ACM Conference on Economics and Computation*. ACM, 2024.
- 15 Matheus V.X. Ferreira, Ye Lin Sally Hahn, S. Matthew Weinberg, and Catherine Yu. Optimal strategic mining against cryptographic self-selection in proof-of-stake. In *Proceedings of the 23rd ACM Conference on Economics and Computation*, EC ’22, pages 89–114, New York, NY, USA, 2022. Association for Computing Machinery.
- 16 Amos Fiat, Anna Karlin, Elias Koutsoupias, and Christos H. Papadimitriou. Energy equilibria in proof-of-work mining. In *Proceedings of the 2019 ACM Conference on Economics and Computation, EC 2019, Phoenix, AZ, USA, June 24-28, 2019.*, pages 489–502, 2019. doi:10.1145/3328526.3329630.

- 17 Guy Goren and Alexander Spiegelman. Mind the mining. In *Proceedings of the 2019 ACM Conference on Economics and Computation, EC 2019, Phoenix, AZ, USA, June 24-28, 2019.*, pages 475–487, 2019. doi:10.1145/3328526.3329566.
- 18 R.A. Howard. *Dynamic programming and Markov processes*. Technology Press of Massachusetts Institute of Technology, 1960.
- 19 Aggelos Kiayias, Elias Koutsoupias, Maria Kyropoulou, and Yiannis Tselekounis. Blockchain mining games. In *Proceedings of the 2016 ACM Conference on Economics and Computation, EC '16, Maastricht, The Netherlands, July 24-28, 2016*, pages 365–382, 2016. doi:10.1145/2940716.2940773.
- 20 S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. URL: <http://www.bitcoin.org/bitcoin.pdf>.
- 21 M.L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley, 2014.
- 22 Ayelet Sapirshstein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in bitcoin. In *Financial Cryptography and Data Security - 20th International Conference, FC 2016, Christ Church, Barbados, February 22-26, 2016, Revised Selected Papers*, pages 515–532, 2016. doi:10.1007/978-3-662-54970-4\_30.
- 23 Toni Wahrstätter. Selfish mixing and randao manipulation, July 2023. URL: <https://ethresear.ch/t/selfish-mixing-and-randao-manipulation/16081>.
- 24 Aviv Yaish, Gilad Stern, and Aviv Zohar. Uncle maker: (time)stamping out the competition in ethereum. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023*, pages 135–149. ACM, 2023. doi:10.1145/3576915.3616674.
- 25 Aviv Yaish, Saar Tochner, and Aviv Zohar. Blockchain stretching & squeezing: Manipulating time for your best interest. In David M. Pennock, Ilya Segal, and Sven Seuken, editors, *EC '22: The 23rd ACM Conference on Economics and Computation, Boulder, CO, USA, July 11 - 15, 2022*, pages 65–88. ACM, 2022. doi:10.1145/3490486.3538250.
- 26 Roi Bar Zur, Ittay Eyal, and Aviv Tamar. Efficient mdp analysis for selfish-mining in blockchains. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies, AFT '20*, pages 113–131, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3419614.3423264.





# ***Bribe & Fork:*** **Cheap PCN Bribing Attacks via Forking Threat**

**Zeta Avarikioti** ✉

Department of Informatics, TU Wien, Austria

**Paweł Kędzior** ✉

University of Warsaw, Poland

**Tomasz Lizurej** ✉

NASK, Warsaw, Poland

University of Warsaw, Poland

**Tomasz Michalak** ✉

IDEAS NCBR, Warsaw, Poland

University of Warsaw, Poland

---

## **Abstract**

In this work, we reexamine the vulnerability of Payment Channel Networks (PCNs) to bribing attacks, where an adversary incentivizes blockchain miners to deliberately ignore a specific transaction to undermine the punishment mechanism of PCNs. While previous studies have posited a prohibitive cost for such attacks, we show that this cost can be dramatically reduced (to approximately \$125), thereby increasing the likelihood of these attacks. To this end, we introduce *Bribe & Fork*, a modified bribing attack that leverages the threat of a so-called feather fork which we analyze with a novel formal model for the mining game with forking. We empirically analyze historical data of some real-world blockchain implementations to evaluate the scale of this cost reduction. Our findings shed more light on the potential vulnerability of PCNs and highlight the need for robust solutions.

**2012 ACM Subject Classification** Security and privacy → Systems security

**Keywords and phrases** Blockchain, Payment Channels Networks, Timelock Bribing, Feather Forking

**Digital Object Identifier** 10.4230/LIPIcs.AFT.2024.11

**Related Version** *Full Version:* <https://arxiv.org/abs/2402.01363>

**Funding** This work was supported by the Austrian Science Fund (FWF) through the SFB SpyCode project F8512-N, the project CoRaF (grant agreement ESP 68-N), and by the WWTF through the project 10.47379/ICT22045.

This result is part of a project that received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 and Horizon Europe research and innovation programs (grant PROCONTRA-885666). This work was also partly supported by the National Science Centre, Poland, under research project No. 46339.

**Acknowledgements** We thank Paul Harrenstein for his help in defining the model presented in this work.

## **1 Introduction**

The financial world was transformed by blockchains such as Bitcoin [24] and Ethereum [32]. While blockchains offer a number of benefits, their scalability remains a significant challenge when compared to traditional centralized payment systems [10]. One promising solution to this issue is the so-called *payment channel networks (PCNs)* that move most of the transaction workload off-chain [15].



© Zeta Avarikioti, Paweł Kędzior, Tomasz Lizurej, and Tomasz Michalak;  
licensed under Creative Commons License CC-BY 4.0

6th Conference on Advances in Financial Technologies (AFT 2024).

Editors: Rainer Böhme and Lucianna Kiffer; Article No. 11; pp. 11:1–11:22

Leibniz International Proceedings in Informatics



LIPIC Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Several PCN proposals [1, 2, 3, 4, 5, 11, 12, 13, 14, 16, 25, 28] have been laid forward so far, each design offering some unique combination of features. Nonetheless, the core idea behind payment channels remains the same, that is to facilitate off-chain transactions among parties connected, either directly or indirectly via a path, on a network operating on top of the blockchain layer, the PCN.

To participate in a PCN, two parties can lock funds in a joint account on-chain, thereby opening a payment channel. Subsequently, the parties can transact off-chain by simply updating (and signing) the distribution of their funds. When either party wants to settle the account, or in other words close the payment channel with its counterparty, they can publish the last agreed distribution of the channel’s funds. However, each update constitutes a valid closure of the channel from the perspective of the blockchain miners. As a result, a malicious party may publish an outdated update to close the channel holding more than it currently owns. To secure the funds against such attacks, payment channels enforce a *dispute period*. During this period, the funds remain locked to allow the counterparty to punish any malicious behavior, and if so, claim all the funds locked in the channel.

Hence, the security of PCNs, like the most widely deployed Bitcoin Lightning Network [25], crucially relies on financial incentives. Specifically, during the dispute period, the punishment mechanism should enforce that a malicious party is always penalized and an honest party should never lose its funds. Unfortunately, this is not always the case, as argued by Khabbazian et al. [23] and Avarikioti et al. [6]. For instance, Lightning channels are susceptible to the so-called *timelock bribing attacks*. In such an attack, a malicious party posts an old update transaction on-chain, attempting to close its channel with more funds than it presently possesses. Concurrently, the party bribes the miners to ignore the punishment transaction of the counterparty for the duration of the dispute period. This bribe is typically offered to the miners in the form of high transaction fees.

Naturally, the success of this attack depends on the value of the bribe. Avarikioti et al. [6] showed that a bribing attack will be successful if the bribe is no smaller than:  $\frac{f_1 - f}{\lambda_{min}}$ , where  $\lambda_{min}$  is the fraction of the mining power controlled by the least significant miner in the underlying proof-of-work blockchain,  $f_1$  is the sum of the fees of a block containing the punishment transaction, and  $f$  is the sum of the fees of a block containing average transactions. We observe, however, that as  $\lambda_{min}$  can be arbitrarily small, the bribing amounts required can significantly exceed the funds typically locked in PCNs, rendering the bribing attack impractical. Moreover, even in blockchains with rather concentrated mining power, like in Bitcoin [18], the cost of a bribing attack is very high. For example, conservatively assuming that the smallest miner has  $10^{-4}$  of the total mining power<sup>1</sup>, the cost of the attack as analyzed in [6] would be at least 1 BTC, for  $f_1 - f \approx 10^{-4}$  BTC. As a result, it would be irrational to perform a bribing attack of this sort, as the average closing price for 1 BTC between, for instance, 2019 and 2022 was 23,530.92 USD<sup>2</sup>, which is more than 10-fold the current total value locked on average in a Lightning channel<sup>3</sup>. This naturally leads to questioning *whether there is potential to amplify such attacks to the extent they pose a genuine threat to the security of PCNs*.

<sup>1</sup> Details can be found in the full version of the paper. We experimentally show, that the mining power of the weakest miner in the system can be fairly assumed to be of magnitude  $10^{-12}$

<sup>2</sup> [statmuse.com](https://statmuse.com)

<sup>3</sup> <https://1ml.com/statistics>

## Our Contribution

In this work, we show that *bribing costs can be significantly reduced*, thereby making timelock bribing attacks a realistic threat. We do so by extending the bribing attack to leverage not only the structure of transaction fees but also a threat to fork the blockchain, known as a *feather fork attack* [19, 27]. In our context, a given miner executes a feather fork attack by announcing a self-penalty transaction  $tx_p$ . Whenever the self-penalty transaction appears on the blockchain, the miner is incentivized to fork the punishment transaction  $tx_1$  on the blockchain, i.e., the miner will try to extend the blockchain based on the predecessor of the block  $tx_1$  including some other block. Specifically, a feather-forking miner is bribed to commit collateral, betting that their fork will win the race. As a consequence, their threat of forking becomes considerably credible, incentivizing other miners to follow their fork. The collateral is of a similar magnitude as the bribe in [6], however, the miners only lock it *temporarily*. We call our attack *Bribe & Fork*. With the feather fork at hand, the bribing cost may now be reduced from  $\frac{f_1 - f}{\lambda_{min}}$  to approximately:

$$\frac{2\bar{f} + 2(f_1 - f)}{\lambda_s},$$

where  $\bar{f}$  is the average fee of a single transaction, and  $\lambda_s$  is the mining fraction of the most significant miner. Recall that  $f_1$  and  $f$  denote the sum of fees of a block containing the punishment transaction and only average transactions respectively. To demonstrate the cost reduction of *Bribe & Fork*, we reexamine the previous example for Bitcoin, with  $\bar{f} \approx 10^{-4}$  BTC,  $f_1 - f \approx \bar{f}$ , and  $\lambda_s \approx 20\%$ . Now,  $\lambda_s$  replaces in the denominator the previously presented  $\lambda_{min} \ll 10^{-4}$ , thus *yielding a bribe at least 1000 times smaller than the one in [6]*.

To derive this result, we present a formal model of mining games with forking, extending the conditionally timelocked games introduced in [6]. In the game with forks, miners may now choose, in each round, (a) which transactions to mine, and (b) whether they want to continue one of the existing chains or they intend to fork one of the chains. All miners know the choices of the winner of each block, as a feather-forking miner locks collateral on-chain.

To empirically estimate the cost reduction of *Bribe & Fork*, we analyze the historical data of real-world blockchain implementations. Among others, we analyze the average block rewards and fees, as well as the hash power present in the system and available to a single miner, primarily for Bitcoin in 2022. Given the officially available data, we observe that *the cost of our attack can be as cheap as \$125* (for 1 BTC  $\approx$  \$25,000). In general, the cost of our attack can be up to  $10^{10}$  times cheaper than the bribe required in [6] according to our findings. Hence, even considering a collateral of around \$30,000, *Bribe & Fork* is substantially more cost-efficient, and, by extension, more probable to occur.

## 2 Background

In this section, we first describe the necessary context required to understand *Bribe & Fork*.

### 2.1 Timelocked Bribing Attack

Whenever miners decide to create a new block, they select some set of transactions from all transactions posted on the mempool, which is a database of all publicly visible transactions. Mining pools and individual miners usually choose the transactions with the highest fees first, as they are part of their reward for a successfully mined block. Miners are aware that some

transactions may be dependent on each other. For instance, two transactions that spend the same Unspent Transaction Output (or UTXO) in Bitcoin, cannot be both published on-chain; the transaction that is validated first, i.e., is included on a block of the longest chain, immediately deems invalid the other transaction. If from two interdependent transactions, only one can be published directly, while the other is timelocked and thus can only be published after some time elapses, we refer to this pair of transactions as a *conditionally timelocked output*. This conditionally timelocked output is the target of timelock bribing attacks: the owner of the coins of the transaction that is valid only after the timelock expires attempts to bribe the miners to ignore the currently valid competing transaction. Thus, for the miners that observe transactions on the mempool, sometimes it may be profitable to censor one transaction in order to mine another one that provides a greater gain in the future.

## 2.2 Timelock Bribing in the Bitcoin Lightning Network

Next, we describe the timelock bribing attack in the context of the most widely deployed payment channel network, namely the Bitcoin Lightning Network (LN). In LN, a single on-chain transaction called the funding transaction, opens a channel between parties  $\mathcal{P}_1, \mathcal{P}_2$ . Next, parties exchange with each other signed messages off-chain which update the state of their accounts. If the parties are honest and responsive, they may close the channel in collaboration. To do so, the parties post a mutually signed transaction that spends the output of the funding transaction and awards each party their fair share of funds. However, if a dishonest party  $\mathcal{P}_2$  attempts to publish on-chain an old state that she profits from comparably to the latest agreed state, her funds will remain locked for the so-called dispute period. During this period, the other (rational) party  $\mathcal{P}_1$  will try to revoke the old state, by sending a transaction  $tx_1$  to the mempool called the revocation transaction (or breach remedy). Transaction  $tx_1$  awards all the channel funds to the cheated party  $\mathcal{P}_1$ . We denote by  $f_1$  the miner's fee to include a block with  $tx_1$ .

In this case, the malicious party  $\mathcal{P}_2$  can launch a timelocked bribing attack, attempting to bribe the miners to ignore  $tx_1$  for the dispute period  $T$  such that  $\mathcal{P}_2$  gains the additional funds. Specifically,  $\mathcal{P}_2$  may send in the mempool a block with fee  $f_2$  that includes transaction  $tx_2$ , with  $f_2 > f_1$ , that is only valid if no miner includes in their winning block containing  $tx_1$  within time  $T$ . Consequently, if the revocation transaction  $tx_1$  is not published on-chain within  $T$ ,  $\mathcal{P}_2$  can spend the funds of the old state and the next winning miner will be awarded  $f_2$ . The pair of transactions  $tx_1$  and  $tx_2$  is now a conditionally timelocked output.

Assuming that for an average block of transactions, the users get in total  $f$  fees, the following holds [6]: if  $f_2 - f > \frac{f_1 - f}{\lambda_{min}}$  then all rational miners will choose to wait for  $T$  rounds and publish a block containing  $tx_2$ .

## 2.3 Feather Fork Attack

A feather fork, as introduced in [20], is an attack on Bitcoin wherein a miner threatens to fork the chain if selected transactions are included. This intimidation mechanism aims to subtly alter the miners' block acceptance policy: the threatened miners may exclude the selected transactions in order to mitigate the risk of losing their mining reward [31]. As feather forking relies on economic incentives, the attacker may increase their probability of success by bribing other miners to follow their short-lived fork, e.g., committing to pay them the block rewards they may lose by censoring the selected transactions.

Unlike a “hard” fork, where miners exclusively mine their own chain version regardless of its length compared to other versions, a feather fork entails mining on the longest chain that excludes selected transactions and does not fall significantly behind its alternatives. Thus, a feather fork is less disruptive and more likely to be adopted by the network, making it a potentially powerful tool in the hands of a malicious actor. In this work, we employ this tool to enhance the likelihood of a successful timelocked bribing attack.

### 3 Bribe & Fork Attack

We now introduce our novel attack, termed *Bribe & Fork*, that combines the timelock bribing attack in LN with the feather fork attack. We assume the existence of a payment channel between parties  $\mathcal{P}_1$  and  $\mathcal{P}_2$ . Similarly to timelock bribing, we consider  $\mathcal{P}_2$  to be malicious and attempt to close the channel with  $\mathcal{P}_1$  in an old state using the transaction  $tx_2$ . Consequently,  $\mathcal{P}_1$  is expected to attempt to revoke the old state. To prevent the inclusion of the revocation transaction,  $\mathcal{P}_2$  bribes the miner  $N_s$  with the highest mining power  $\lambda_s$  to threaten others with a fork if they add the unwanted transaction  $tx_{s1}$  on-chain. This action is implemented through a self-penalty mechanism where the bribed miner temporarily locks collateral which can only be reclaimed in case a block  $tx_{s2}$  containing  $tx_2$  appears on-chain (and thus any block  $tx_{s1}$  containing  $tx_1$  is not included on-chain). In essence, the bribed miner “bets” that the revocation transaction will be censored, thus rendering the threat credible for the rest of the miners.

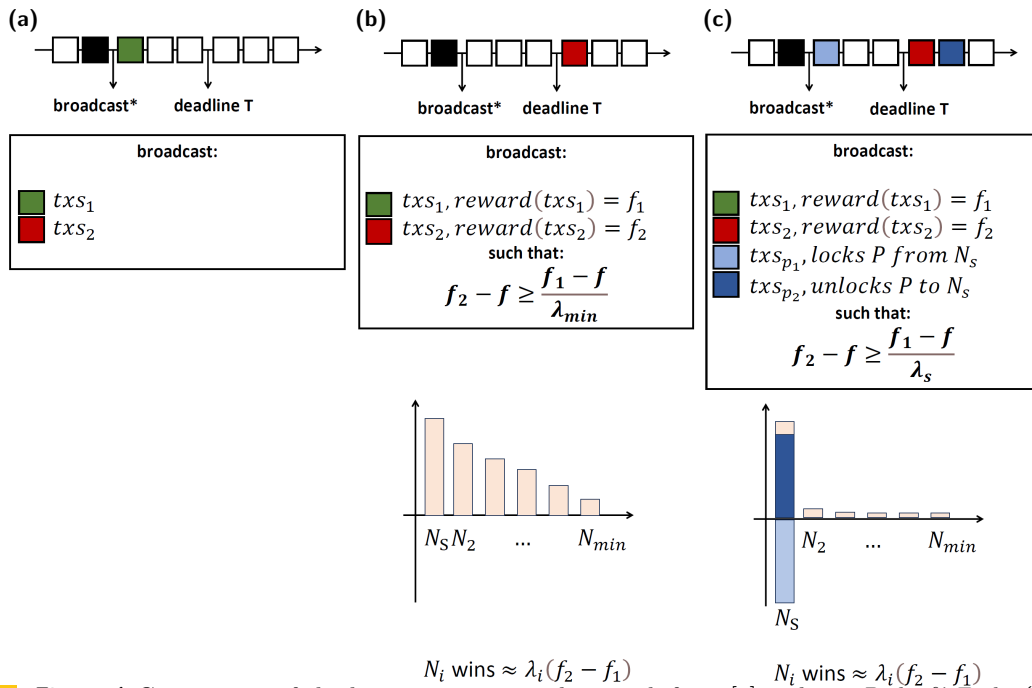
To realize *Bribe & Fork*, there are two mechanisms that should be implementable on-chain: a) the bribe transaction that should only be spendable if  $tx_{s2}$  is included on-chain, and b) the self-penalty mechanism that enables the bribed miner  $N_s$  to lock collateral  $P$  (with transaction  $tx_{p1}$ ) and then reclaim it only if  $tx_{s2}$  is included on-chain (with transaction  $tx_{p2}$ ). We implement the bribing and self-penalty mechanisms in Bitcoin script, using the conditioning enabled by the UTXO (Unspent Transaction Output) structure. We note that in Ethereum, preparing a smart contract that has access to the state of the closing channel suffices to implement the bribing and self-penalty mechanisms.

In detail, a single bribing transaction  $tx_b$  and two special transactions  $tx_{p1}, tx_{p2}$  are introduced. Let us assume that the cheating party  $\mathcal{P}_2$  is bribing the miners to launch the attack conditioned on the inclusion of its transaction  $tx_2$ . To do so,  $\mathcal{P}_2$  creates a bribe transaction  $tx_b$  with input the party’s money from  $tx_2$  and outputs three UTXOs, one given to the miner that mines this transaction, one dummy output owned by player  $N_s$  (i.e., the miner bribed to perform feather forking), and one that returns the rest of the money to  $\mathcal{P}_2$ . Now,  $N_s$  creates a transaction  $tx_{p1}$ , locking a deposit  $P$ , that is spendable via a multisignature of  $m$ -out-of- $n$  parties (e.g.,  $m = n/2$ ), one of which is  $N_s$ ’s signature. Then,  $tx_{p2}$  is created with two inputs: the output of  $tx_{p1}$  and the dummy output of  $tx_b$ . Consequently,  $tx_{p2}$  is spendable only if it is signed by at least  $m$  parties of the predefined set  $n$  and  $tx_2$  is validated on-chain. Upon receiving  $tx_{p2}$  signed by at least  $m - 1$  parties,  $N_s$  signs and posts it on-chain. Assuming that no subset of size  $m - 1$  of the rest  $n - 1$  parties will collude with the miner to spend the deposit, the deposit can be claimed by the miner only if transaction  $tx_2$  is included on-chain. The security of this scheme depends on the selection of the  $n - 1$  parties, which can be in principle conditioned on the honest majority assumption of the blockchain via subsampling.<sup>4</sup>

<sup>4</sup> One could also consider using an instantiation of a Trusted Execution Environment (TEE) that outputs  $tx_{p2}$  only when  $tx_{s2}$  appears on the blockchain. Additionally, note that on Ethereum, preparing a smart contract that has access to the state of the closing channel is sufficient to implement the penalty mechanism.

Now, the malicious party  $\mathcal{P}_2$  together with a selected miner  $N_s$  can launch the *Bribe & Fork* attack as follows: They can create special transactions  $tx_b$  (as a companion transaction for  $tx_2$ ) and the self-penalty transactions  $tx_{p_1}, tx_{p_2}$  to publicly announce a credible threat that the transaction  $tx_1$  will be forked once it appears on the blockchain. They publish these special transactions in the mempool as transaction sets  $txs_2$  (that contains  $tx_2$  and  $tx_b$ ) and  $txs_{p_1}, txs_{p_2}$  (that contain  $tx_{p_1}, tx_{p_2}$ , respectively).

We show later that this attack significantly reduces the cost of the required bribe from  $\frac{f_1 - f}{\lambda_{min}}$  to approximately  $\frac{2\bar{f} + 2(f_1 - f)}{\lambda_s}$ . The required collateral that is eventually reclaimed by the bribed miner is expected to be  $\lambda_s \cdot B$  (where  $B$  is a constant in Bitcoin block reward independent of the user fees), which is comparable to the original bribe needed in [6]. Figure 1 illustrates the comparison of *Bribe & Fork* with [6], while Figure 2 below depicts the details of *Bribe & Fork*.

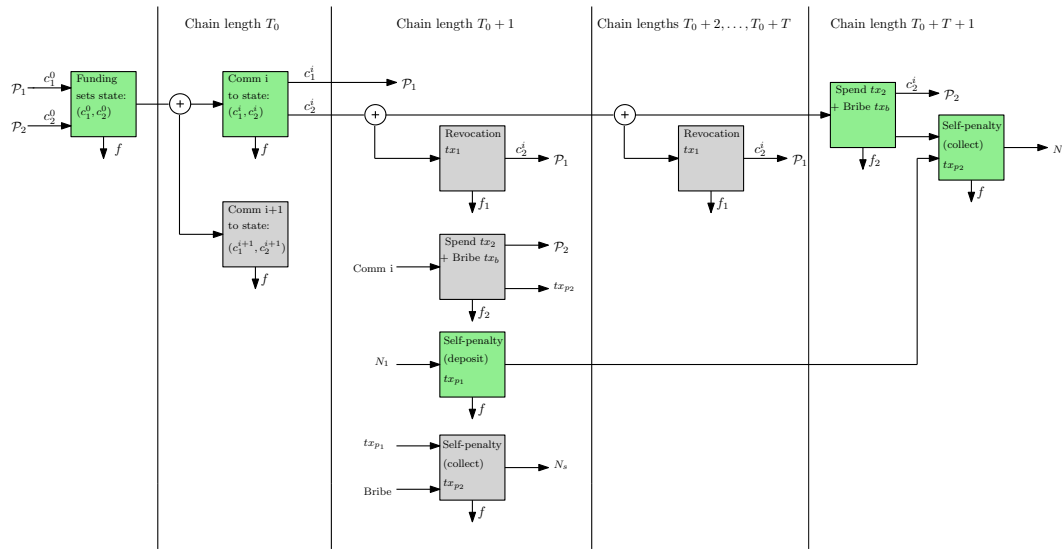


■ **Figure 1** Comparison of the honest execution, the attack from [6] and our *Bribe & Fork*. (a) **Honest execution:** once an old state appears on-chain (black rectangle),  $\mathcal{P}_1$  gets an option to revoke this state with a transaction  $tx_1$  (included in the block  $txs_1$  which is published in the first round). (b) **Attack in [6]:** the bribing party publishes  $tx_2$  and  $tx_b$  included in a single block  $txs_2$ , with a large miner fee (reversely proportional to the fraction of the mining power  $\lambda_{min}$  of the least significant miner). The miners skip mining  $txs_1$  in the first round, and mine  $txs_2$  in the last round. (c) ***Bribe & Fork*:** the bribing party publishes  $txs_2$  with a fee sufficient to bribe only the strongest miner (with  $f_2 - f$  reversely proportional to  $\lambda_s$ ). The strongest miner publishes the self-penalty transactions  $tx_{p_1}, tx_{p_2}$  that can be mined in transaction sets  $txs_{p_1}, txs_{p_2}$ . In the first round, the miner  $N_s$  locks  $P \approx \lambda_s \cdot B$  to the deposit transaction  $tx_{p_1}$ , thus threatening other miners that they will be forked once  $txs_1$  is mined before the deadline. After the deadline the transaction set  $txs_2$  is published and the miner  $N_s$  may collect back the deposit using  $txs_{p_2}$ .

### Implementation Details of *Bribe & Fork*

Figure 2 contains a diagram depicting the details of our attack. At first, a Lightning Channel is opened with a single funding transaction and allows parties  $\mathcal{P}_1, \mathcal{P}_2$  to make an arbitrary number of off-chain state transitions of their funds. Once one of the parties ( $\mathcal{P}_2$  in our

example) decides to publish an old state ( $comm_i$  in our example) at a chain of length  $T_0$ , the opportunity to manipulate the behaviour of miners' is opened. Before the chain reaches length  $T_0 + 1$ , the transactions  $tx_1$  and  $tx_2$  are published. Transaction  $tx_1$  allows  $\mathcal{P}_1$  to revoke a dishonestly committed state. Transaction  $tx_2$  allows  $\mathcal{P}_2$  to collect and manage dishonest funds after  $T$  rounds. Along with  $tx_1$  and  $tx_2$ , the bribing transaction  $tx_b$  and self-penalty transactions  $tx_{p_1}, tx_{p_2}$  are published. On the chain of length  $T_0$ , the miners may decide to mine the transaction  $tx_{p_1}$  that would lock some amount of coins of one of the miners (say  $N_s$ ). If so, all miners are threatened that they will be forked once  $tx_1$  appears on the blockchain until the chain reaches length  $T_0 + T$ . On the chain of length  $T_0 + T$ , the transaction  $tx_2$  along with bribing transaction  $tx_b$  may be published and the selected miner  $N_s$  may collect back the deposit with the transaction  $tx_{p_2}$ .



■ **Figure 2** The *Bribe & Fork* attack. The green boxes indicate the transactions that should be put on-chain to run a successful *Bribe & Fork* attack. The grey boxes indicate the transactions that should be published on the mempool before the chain reaches a specific length. For instance, Spend transaction  $tx_2$  has to be published on the mempool before the chain reaches length  $T_0 + 1$ , even though it can not be published on the blockchain until the chain reaches length  $T_0 + T$ . The arrows going into the boxes indicate the spending conditions of the transactions and the arrows going out of the boxes indicate how the funds of the boxes can be spent.

## 4 Our Model

In this section, we gradually define our game that models the process of mining that takes into account the forks. A summary of our notation can be found in the full version of this paper.

### 4.1 Preliminaries

Let us begin by recalling the conditionally-timelocked output definition from [6].

► **Definition 1** (Conditionally timelocked output [6]). *A conditionally timelocked transaction output  $txo(T_0, T, cond_1, cond_2)$  is a transaction output of a transaction  $tx$  with spending condition  $cond_1 \vee cond_2$ . Condition  $cond_1$  is not encumbered with any timelock and condition  $cond_2$  is encumbered with a timelock that expires  $T$  blocks after the block with height  $T_0$ , where  $tx$  was published.*

The game with forks is defined for a fixed set of players (miners)  $N = \{N_1, \dots, N_n\}$  with a tuple of mining powers  $\lambda = (\lambda_1, \dots, \lambda_n)$  and will last  $R$  rounds. Notice that we focus on proof-of-work blockchains that employ the so-called Nakamoto consensus, such as Bitcoin [24]. We assume that in such environments miners (players) tend to form mining pools to (a) bypass the task of verifying transactions – the pool’s manager dispatches a list of valid transactions for inclusion – and more importantly, (b) to guarantee a more stable income as individual mining carries substantial deviation. Empirical evidence supporting this assumption, drawn from Bitcoin, is detailed in the full version of the paper.

For the rest of this work, each miner is assumed to either mine independently or stick to a selected mining pool throughout the execution of the game, i.e., we treat the mining pools as single players in the game. As already mentioned, we assume that the game lasts for a fixed number of rounds. Alternatively, we could consider a scenario where the game lasts until the main chain reaches a predefined length. The first assumption is more suitable for the time periods when the block rate is constant. On the other hand, the second modeling approach is better for longer periods where the mining difficulty of blockchains is adjusted to achieve a given number of blocks within a given time unit.

#### 4.1.1 Global State Object

We introduce a global state object  $\mathcal{S} = \{S_1, \dots, S_{|S|}\}$  that describes a set of currently mined chains on the blockchain. Each  $S_i$  consists of a *list* (chain) of pairs  $S_i = [(\text{block}_1, W_1), \dots, (\text{block}_{|S_i|}, W_{|S_i|})]$  describing successfully mined blocks. In each pair  $(\text{block}_j, W_j) \in S_i$ ,  $\text{block}_j$  describes a set of transactions included in the block, and  $W_j \in N$  indicates a player that successfully mined the block.

#### 4.1.2 Allowed Actions

We define the classes of possible actions in our game:

- All chains in a state can be *continued*. When the operation *continue* is successful, a new pair  $(\text{block}, W)$  is appended to the continued chain in the global state object.
- Chains of length at least 1 in the state can be *forked*. Whenever one of the players successfully *forks*, the new (duplicate) chain is created in the global state object in the following manner:
  - the source fork is duplicated; and
  - a new block replaces the latest block in the duplicate.

For instance, let  $\mathcal{S} = \{[(\text{block}_1, W_1), (\text{block}_2, W_2)]\}$  be a current state with a single chain  $S_1$ . Then, after a successful fork of  $S_1$  with  $(B_3, W_3)$ , one gets  $\mathcal{S} = \{[(\text{block}_1, W_1), (\text{block}_2, W_2)], [(\text{block}_1, W_1), (\text{block}_3, W_3)]\}$ .

Notice that on existing blockchains, miners can fork a chain or mine on top of an arbitrary block in one of the existing chains. However, forking that starts at old blocks is less likely to outrun the longest chain. For that reason, we exclude this possibility from the game (following [20, 31, 26]). In other words, miners in our model can fork only the last transaction on one of the chains, and then either the original chain or the fork becomes stable whenever it reaches a length equal to the length of the original chain plus one. The forks can be modeled differently, e.g., assuming a longer fork length or using a finite automaton definition. We expect our results to hold in the alternative modeling as well, but with different parameters of our solution would change.



### 4.1.3 The Abandon Rule

Let us define the abandon rule  $\text{abandon} : \mathbb{S} \rightarrow \mathbb{S}$  that is later used to abandon old chains no longer useful in the game. As we allow forking only the newest block in a chain, our **abandon** rule will make each chain that outruns the length of other chains the only chain in the game. That is, for any  $S_i \in \mathcal{S}$ :  $\exists_{S_j \in \mathcal{S}} \text{len}(S_j) \geq \text{len}(S_i) + 1$  the abandon rule removes  $S_i$  from the state  $\mathcal{S}$ .

### 4.1.4 Types of Transaction Sets

Each block mined in the game (denoted as **block**) includes only one of the transaction sets listed below:

- An unlimited amount of unrelated transaction sets  $txs_u$  that contain average transactions  $tx_u$  unrelated to any special transactions listed below. These transaction sets can be mined at any point in the game.
- A transaction set  $txs_1$  that contains the transaction  $tx_1$  that spends money of  $txo$  under  $cond_1$ . As long as  $txo$  is not spent, this transaction set can be mined on a chain of any length. The rest of the transactions in this transaction set are unrelated transactions  $tx_u$ .
- A (bribing) transaction set  $txs_2$  that contains the transaction  $tx_2$  that spends money of  $txo$  under  $cond_2$  and a bribing transaction  $tx_b$ . As long as  $txo$  is not spent, this transaction set can be mined on a chain of length  $\geq T$ . The rest of the transactions in this transaction set are unrelated transactions  $tx_u$ .
- A special transaction set  $txs_{p_1}$ . In the first round of the game, one of the players (say  $N_1$ ) might decide to create a transaction set  $txs_{p_1}$  with a self-punishment transaction  $tx_{p_1}$  (see the description of the penalty mechanism in the Section 3). The player chooses the amount  $P$ , which he deposits to the transaction. The rest of the transactions in this transaction set are unrelated transactions  $tx_u$ .
- A special transaction set  $txs_{p_2}$  with transaction  $tx_{p_2}$ . The transaction  $tx_{p_1}$  assures that the player  $N_1$  that created the transaction  $tx_{p_1}$  may collect back the deposit  $P$  by publishing the transaction  $tx_{p_2}$ , but only after the transaction set  $txs_2$  is published on the blockchain (see the Section 3). The rest of the transactions in this transaction set are unrelated transactions  $tx_u$ .

### 4.1.5 Rewards

We assume that a miner, after successfully mining a transaction set  $txs_i$  on the main chain, gets a reward  $\text{reward}(txs_i)$  equal to  $B + f_i + P$ , where  $B$  is a constant block reward and  $f_i$  is a sum of user fees input by users posting transactions in the transaction set  $txs_i$ . Whenever  $txs_i$  contains a transaction that locks  $C$  coins from the miner's account, we set  $P$  to be equal to  $-C$ . Analogously, when a miner collects  $C$  as one of the transactions from  $txs_i$ , we set the parameter  $P$  to  $C$ . The reward for mining a block depends on the number of transactions within the block and their fees. The fee for a more complex transaction is typically higher as it occupies more space in a block. In this respect, we make the following assumptions that correspond to the current Bitcoin values (more details can be found in the full version of the paper):

- Each unrelated transaction set  $txs_u$  has on average  $m$  transactions, its reward

$$\text{reward}(txs_u) = B + f = B + m \cdot \bar{f},$$

where  $\bar{f}$  is an average user transaction fee. We also assume that  $\bar{f} < 10^{-4}B$ .

- For other transaction sets with an uncommon functionality, e.g.,  $txs_j$ , we assume it contains in total  $m - c_j$  unrelated (average) transactions, a special transaction  $tx_j$  that takes space of  $c_j$  average transactions, where  $c_j < m$ . In total,

$$\text{reward}(txs_j) = B + (m - c_j)\bar{f} + c_j\bar{f}_j + P = B + f_j + P,$$

where  $f_j = (m - c_j)\bar{f} + c_j\bar{f}_j$ . The interpretation of the parameter  $c_j$  is that it describes the number of transactions needed in a block to implement the uncommon functionality, each of them with fee  $\bar{f}_j$ .

In the full version of the paper, based on empirical data, we show how block rewards fluctuate in the real world. However, following [6], we assume that standard transactions have a constant (average) reward and that all blocks have a constant number of transactions. In the list above, we refer to each standard transaction as an average transaction.

#### 4.1.6 Mining Power Distribution

We assume the following mining power allocation  $\lambda = (\lambda_1, \dots, \lambda_n)$  among the players (see discussion in the full version of this paper).

- There exists a single “strong” player (say player  $s$ ) with mining power  $\lambda_s \geq 20\%$ . All other players have mining power smaller than  $\lambda_s$ .
- There exists a “relatively” strong player (say player  $i$ ) with mining power  $1\% < \lambda_i < 2\%$ .
- We assume that all players with mining power less than 1% have collective power at most 5%.
- The smallest mining power is of any miner in the game is  $\lambda_{min} > 10^{-100}$ .

#### 4.1.7 Players’ reluctance to believe a threat

In the mining process, players can threaten other players that they will fork their blocks, once these blocks appear on the blockchain, as in the feather forking attacks. However, without any additional assumptions, there exist multiple solutions for such a setting [17]. To derive a single solution in our game, we make a conservative assumption that the players do not conduct the forking action if it *can* result in financial losses to them. In other words, we accept only threats from a player who strictly profits from forking a selected transaction, i.e., the forking action is a dominating one for the player in this particular state.

#### 4.1.8 No Shallow Forks Conjecture

The Conjecture 1 below is a second assumption (together with the assumption that players are reluctant to believe a threat) that allows us to achieve a unique solution in the game with forks. In the conjecture, we assume that players have the option to fork a transaction only when they see an explicit opportunity of mining any other transaction with a *higher* miner’s fee<sup>5</sup>, initially blocked by the currently forked transaction. That is, we forbid shallow forks in the model.

---

<sup>5</sup> We denote that, alternatively to Conjecture 1, one could assume that the size of the mining fees in the game is limited, as excluding transactions with outstandingly high fees can also discourage the players from forking these transactions.

► **Conjecture 1** (No shallow forks). *At any point in the game  $\Gamma_F$ , the players will not start a fork of a chain ending with a transaction set  $txs_a$ , unless they see an explicit opportunity to mine a fork containing at some point  $txs_b$ , initially blocked by  $txs_a$  (e.g., by the conditionally timelocked output transaction mechanism, or the self-penalty mechanism). What is more, the players must be aware that  $\text{reward}(txs_b) > \text{reward}(txs_a)$ .*

Note that given the feasible transaction sets in the Section 4.1.4 and the reward structure defined in the Section 4.1.5, whenever  $\text{reward}(txs_2) > \text{reward}(txs_1)$  and  $\text{reward}(txs_1) > \text{reward}(txs_{p_1})$ , according to the Conjecture 1, the players in our game can attempt to fork only  $txs_1$  to get  $txs_2$  or  $txs_{p_1}$  to mine  $txs_1$ . In other words, whenever  $txs_u, txs_2$ , or  $txs_{p_2}$  appear on one of the chains, they will not be forked.

## 4.2 The Game

Finally, we describe a game that models the process of PoW blockchain mining taking into account the option to conduct a forking of a block. The game proceeds in rounds; in each round, the miners can choose whether they want to continue mining one of the chains or they want to fork one of the chains.

► **Definition 2** (Conditionally timelocked game with forks). *A conditionally timelocked game with forks  $\Gamma_F(N, R)$  is a game with a finite set of players (miners)  $N = \{N_1, \dots, N_n\}$ , where  $n = |N|$ , that lasts  $R$  rounds. We define a tuple of mining powers  $\lambda = (\lambda_1, \dots, \lambda_n)$  associated with the players, such that  $\sum_{\lambda_i \in \lambda} = 1$ . In the following, we will write  $\Gamma_F$ , instead of  $\Gamma_F(N, R)$ , when  $N, R$  are obvious from the context.*

*Given the global state object, the set of possible actions, the rewards structure and the mining power distribution defined above, the game is played as follows:*

1. *The game starts with the state  $\mathcal{S} = \{[\ ]\}$  which is updated exactly  $R$  times. All players are aware that this state is built upon a blockchain of height  $T_0$  which includes an unspent conditionally timelocked transaction output  $txo(T_0, T, \text{cond}_1, \text{cond}_2)$ , where  $T < R$ .*
2. *At each round  $1 \leq r \leq R$ , players  $N_i \in N$  choose which of the subchains  $\mathcal{S}_k \in \mathcal{S}$  they build upon, whether they will continue or fork this chain and which of the feasible blocks (built upon one of the transaction sets) they want to add in case they are declared as the winner. Let  $\Omega(\mathcal{S}, r)$  denote the set of all feasible actions for the state  $(\mathcal{S}, r)$  described as triplets  $(\mathcal{S}, \text{decision}, \text{transaction\_set})$ , where  $\mathcal{S} \in \mathcal{S}$ , and  $\text{decision} \in \{\text{continue}, \text{fork}\}$ . Based on  $\lambda = (\lambda_1, \dots, \lambda_n)$ , one player is declared as the winner in the round  $r$ , and the state object is modified accordingly.*
3. *After each round, the abandon rule `abandon` is run on the current state.*

*When the final round  $R$  of the game is over, it finishes in some state  $\mathcal{S}$ , and rewards are given to the players. By  $\mathcal{S}^*$ , let us denote the longest chain in the state  $\mathcal{S}$ . Whenever the state has multiple longest chains,  $\mathcal{S}^*$  denotes the oldest of the longest chains of  $\mathcal{S}$ . After the final round  $R$ , in the state  $\mathcal{S}$ , the reward given to a player  $N_i$  is:  $\text{reward}_i(\mathcal{S}) = \sum_{(\text{block}, W) \in \mathcal{S}^*: W=i} \text{reward}(\text{block})$ .*

### 4.2.1 Strategies

Notice that given the set of players  $N$ , the actions `continue` and `mine` defined above, and the set of transaction sets possible to mine, one can determine the set  $\mathbb{S}$  of all states that may happen in the game.

A strategy  $\sigma_i$  for a player  $N_i$  is given by a function mapping each pair  $(\mathcal{S}, r) \in \mathbb{S} \times [R]$  to a triplet feasible for this pair  $(\mathcal{S}, \text{decision}, \text{transaction\_set})$ .

Let  $\sigma$  denote a strategy profile of all players - a tuple of strategies of all players - i.e.,  $\sigma = (\sigma_1, \dots, \sigma_n)$ . Given a fixed index  $i$ , with  $\sigma_{-i}$ , we will denote a strategy profile of all players, but the selected  $N_i$ .

The distribution of mining power among the players  $\lambda = (\lambda_1, \dots, \lambda_n)$ , the strategy profile  $\sigma$ , current state  $\mathcal{S} \in \mathbb{S}$  and current round  $r \in [R]$  define a probability distribution function  $\mathbf{p}_{\lambda, \sigma, \mathcal{S}, r, r'} : \mathbb{S} \rightarrow [0, 1]$  that assigns a probability that a certain state  $\mathcal{S}' \in \mathbb{S}$  is activated after round  $r' \in [R]$ , where  $r' > r$ . Given  $\mathbf{p}_{\lambda, \sigma, \mathcal{S}, r, r'}$  and the reward function, we can define the utility (the expected reward)  $\mathbb{E}_i(\sigma)$  of each player  $i$ , when strategy  $\sigma$  is played. We say  $\sigma^* = (\sigma_1^*, \dots, \sigma_n^*)$  is a Nash Equilibrium if for all players  $N_i \in N$  it holds that

$$\mathbb{E}_i(\sigma_i^*, \sigma_{-i}^*) \geq \mathbb{E}_i(\sigma_i, \sigma_{-i}^*),$$

for all alternative strategies  $\sigma_i$  for the player  $i$ .

We denote by  $\Gamma_F^{\mathcal{S}, r}$  the subgame of the game  $\Gamma_F$  in a round  $r$ , at a state  $\mathcal{S}$ . We denote by  $\mathbb{E}_i(\sigma, \mathcal{S}, r)$  the utility of a player  $N_i$  in this subgame, which is the expected reward for this player once the game is over.

## 5 Analysis of Bribe & Fork

In this section, we formally analyze *Bribe & Fork* where a bribe transaction  $tx_2$  is published, large enough to bribe a chosen miner with the highest mining power, yet significantly smaller than the value required to directly persuade all miners to skip mining the transaction  $txs_1$ . The selected miner is then asked to threaten others with a fork if they add the unwanted transaction  $txs_1$  to the blockchain. To make these threats credible, we implemented the self-penalty mechanism (see Section 3).

### 5.1 About the proofs

In the proofs, we aim to find a *dominating strategy* for a player  $N_i$  in a given state  $\mathcal{S}$  and a round  $r$ , i.e., a strategy that outweighs other strategies of a selected player in the given state and round. As we will move from the very last round of the game till the first round of the game, we will be able to conclude our reasoning with a single NE of the full game. Whenever needed, we use the mathematical induction technique to show that some choice of strategy is optimal for a sequence of rounds. Usually, the base case is the last round of the game and the induction step proves that if a given strategy is dominating in a round  $k + 1$ , then it is also a dominating strategy in a round  $k$ .

When we compare how the player  $N_i$  benefits from taking two distinct actions  $A, B$  in a given state  $\mathcal{S}$  and a round  $r$ , we often say that there exists a constant  $C$  common for these strategies. To this end, we assume that action  $A$  refers to some strategy  $\sigma_a$  of the player  $N_i$ , and action  $B$  refers to some strategy  $\sigma_b$  of the player  $N_i$ , such that  $\sigma_a$  differs only in its definition from  $\sigma_b$  on the selected state  $\mathcal{S}$  and the selected round  $r$ . The utility of the player  $N_i$  is the same for both strategies whenever in the state  $\mathcal{S}$  and  $r$  someone else than  $N_i$  is selected as the winner of the round. With  $C$ , we denote the utility of player  $N_i$  multiplied by the probability of this event when player  $N_i$  is not the winner of the round. This reasoning gives us an easy-to-use method to compare utility between the strategies  $\sigma_a, \sigma_b$ . We can thus compare the utilities of the player  $N_i$  in the state  $\mathcal{S}$  and round  $r$  when the two distinct strategies  $\sigma_a, \sigma_b$  are selected as:

$$\begin{aligned} \mathbb{E}_i(\sigma_a, \mathcal{S}, r) &= \lambda_i(\text{utility of the player } N_i \text{ when action A was taken}) + C, \\ \mathbb{E}_i(\sigma_b, \mathcal{S}, r) &= \lambda_i(\text{utility of the player } N_i \text{ when action B was taken}) + C. \end{aligned}$$

## 5.2 Transaction Order in a Single Chain

Although, due to the definition of the player's utility function, the bribing transaction ( $txs_2$ ) may encourage the players not to skip mining transactions with high rewards during the dispute period (e.g.  $txs_1$ ), we first show that once timelock is over and  $txs_1$  was not mined, the players will follow the default strategy to mine transactions with highest rewards first.

► **Lemma 3.** *Let  $\Gamma_F(\mathcal{S}, T+1)$  be a subgame in a state  $\mathcal{S} = \{S\}$ , where the state  $\mathcal{S}$  contains a single chain  $S$  and the transaction set  $txs_{p_1}$  was mined in the first round. In the next  $T-1$  rounds, miners mined unrelated transactions sets  $txs_u$ . Furthermore, it holds that*

$$\text{reward}(txs_2) > \text{reward}(txs_1) > \text{reward}(txs_u) \text{ and } \text{reward}(txs_{p_2}) > \text{reward}(txs_u).$$

*Then the dominating strategy for all players in the subgame  $\Gamma_F(\mathcal{S}, T+1)$  is to mine transaction sets in the following order:  $txs_2, txs_{p_2}$ , and for the rest of rounds  $txs_u$ .*

**Proof.** As  $txs_{p_1}$  was mined in the first round, then in any round after the round  $T+1$  there are available to mine the following transaction sets:

- mutually exclusive  $txs_1$  and  $txs_2$  with expired timelock,
- one  $txs_{p_2}$  that can be mined only after  $txs_2$  appears on blockchain,
- and an unlimited amount of unrelated transaction sets  $txs_u$ .

Since only  $txs_1$  and  $txs_2$  are mutually exclusive and  $\text{reward}(txs_1) < \text{reward}(txs_2)$ , then by Conjecture 1, whenever  $txs_2, txs_{p_2}$  or  $txs_u$  appear on the blockchain, they will not be forked. Thus only  $txs_1$  may be forked in the subgame.

Once both  $txs_2, txs_{p_2}$  are on the chain, miners can not mine any special transaction sets, and all of the miners will mine  $txs_u$  till the end of the game.

Next we show that for any round  $r \in \{T+2, \dots, R\}$ , in a state  $\mathcal{S}'$  created by extending the chain in  $\mathcal{S}$  with  $txs_u$  and one  $txs_2$  at any point in this chain, the dominating strategy for all players is to mine  $txs_{p_2}$  first if it was not mined until this point. We will prove it by induction. The statement trivially holds in the last round  $R$ , because  $txs_{p_2} > txs_u$ . Now, assuming that it holds in round  $R-k$ , we prove that it also holds in round  $R-k-1$ . Any player  $N_i$  will be chosen with probability  $\lambda_i$  as the winner of the round. The utility of the player  $N_i$  following a strategy  $\sigma_{p_2}$  that first mines  $txs_{p_2}$  in the state  $\mathcal{S}'$  is:

$$\mathbb{E}_i(\sigma_{p_2}, \mathcal{S}', R-k-1) = \lambda_i(\text{reward}(\mathcal{S}') + f_{p_2} + B + \lambda_i k(f+B)) + C,$$

for some constant  $C$  that describes the expected reward of  $N_i$  in case he/she is not chosen as the winner of this round.

The utility of the player  $N_i$  following a strategy  $\sigma_u$  that first mines  $txs_u$  in the state  $\mathcal{S}'$  is:

$$\mathbb{E}_i(\sigma_u, \mathcal{S}', R-k-1) = \begin{cases} \lambda_i(\text{reward}(\mathcal{S}') + f_u + B) + C & \text{when } k=0 \\ \lambda_i(\text{reward}(\mathcal{S}') + f_u + B + \lambda_i(f_{p_2} + B) + \lambda_i(k-1)(f+B)) + C & \text{when } k \geq 1 \end{cases}$$

From the above it follows that

$$\mathbb{E}_i(\sigma_{p_2}, \mathcal{S}', R-k-1) > \mathbb{E}_i(\sigma_u, \mathcal{S}', R-k-1).$$

Next we show that for any  $r \in \{R, \dots, T+1\}$ , in a state  $\mathcal{S}''$  created by extending the chain in  $\mathcal{S}$  with  $txs_u$ , the dominating strategy for all players is to mine  $txs_2$  first if it was not mined until this point. Observe that in the state  $\mathcal{S}''$  the miners can only mine  $txs_u, txs_1$  or  $txs_2$ . The statement trivially holds in the last round. Now, assuming that it holds in

round  $R - k$ , we prove that it also holds in round  $R - k + 1$ . Any player  $N_i$  will be chosen with probability  $\lambda_i$  as the winner of the round, and with probability  $1 - \lambda_i$  someone else will be selected as the winner of the round. Then for some constant  $C$ , the utility of the player  $N_i$  in a strategy  $\sigma_2$  that first mines  $txs_2$  is

$$\mathbb{E}_i(\sigma_2, \mathcal{S}', R - k + 1) = \begin{cases} \lambda_i(\text{reward}_i(\mathcal{S}'') + f_2 + B) + C & \text{when } k = 0 \\ \lambda_i(\text{reward}_i(\mathcal{S}'') + f_2 + B + \lambda_i(f_{p_2} + B)) + C & \text{when } k = 1 \\ \lambda_i(\text{reward}_i(\mathcal{S}'') + f_2 + B + \lambda_i(f_{p_2} + B) + (k - 2)\lambda_i(f + B)) + C & \text{when } k \geq 2 \end{cases}$$

Whereas the utility of the player  $N_i$  in a strategy  $\sigma_1$  that first mines  $txs_1$  is

$$\mathbb{E}_i(\sigma_1, \mathcal{S}'', R - k + 1) \leq \begin{cases} \lambda_i(\text{reward}_i(\mathcal{S}'') + f_1 + B) + C & \text{when } k = 0 \\ \max\{\lambda_i(\text{reward}_i(\mathcal{S}'') + f_1 + B + \lambda_i(f + B)) + C, \\ \lambda_i(\text{reward}_i(\mathcal{S}'') + f_2 + B) + C\} & \text{when } k = 1 \\ \max\{\lambda_i(\text{reward}_i(\mathcal{S}'') + f_1 + B + 2\lambda_i(f + B)) + C, \\ \lambda_i(\text{reward}_i(\mathcal{S}'') + f_2 + B + \lambda_i(f_{p_2} + B)) + C\} & \text{when } k = 2 \\ \max\{\lambda_i(\text{reward}_i(\mathcal{S}'') + f_1 + B + k\lambda_i(f + B)) + C, \\ \lambda_i(\text{reward}_i(\mathcal{S}'') + f_2 + B + \lambda_i(f_{p_2} + B)) + (k - 3)\lambda_i(f + B)C\} & \text{when } k \geq 3 \end{cases}$$

It is again easy to see that  $\mathbb{E}_i(\sigma_2, \mathcal{S}'', R - k + 1) > \mathbb{E}_i(\sigma_1, \mathcal{S}'', R - k + 1)$ .  $\blacktriangleleft$

The details of the proof of the above Lemma imply the following result.

► **Lemma 4.** *Let  $\Gamma_F(\mathcal{S}, T + 1)$  be a subgame in a state  $\mathcal{S} = \{S\}$ , where the state contains a single chain  $S$  where in  $T$  rounds miners mined unrelated transaction sets  $txs_u$ . Furthermore, for all miners*

$$\text{reward}(txs_2) > \text{reward}(txs_1) > \text{reward}(txs_u).$$

*Then the dominating strategy for all players in the subgame  $\Gamma_F(\mathcal{S}, T + 1)$  will result in the following transactions order:  $txs_2$ , and for the rest of rounds  $txs_u$ .*

### 5.3 Decisions of an Individual Miner are Consistent

In this section, we show that without a high-cost reward  $f_2$ , once someone is successful with mining  $txs_1$ , the miner will continue mining this chain, as it might be too costly for the miner to lose the block reward that he already mined. As  $txs_1, txs_2$  is the only pair of conflicting transactions in the game whenever  $txs_{p_1}$  was not created, it follows from Conjecture 1 that the forks may occur only when one miner successfully mines  $txs_1$ , and the other player wants to profit from mining  $txs_2$ . Thus, in the following, we study the behavior of the players whenever one of the players decides to mine  $txs_1$ .

► **Theorem 1.** *Assuming subgame  $\Gamma_F(\mathcal{S}, r)$  in a state  $\mathcal{S}$  with a single chain of length  $r \leq T$ , formed until round  $r$  where player  $N_j$  mined  $txs_1$  in the last round, and  $txs_{p_1}$  is not on the chain, then the player  $N_j$  will continue to mine this chain unless  $f_2 - f \geq f_1 + B$ , even when other miners decide to fork the chain with  $txs_1$  and continue mining the new subchain created during the fork.*

**Proof.** At every point of the game, each player  $N_i$  can choose a strategy for the remaining  $M$  rounds to collect at least  $M\lambda_i(f + B)$  if he simply always chooses to mine  $txs_u$  from this point.

Thus, whenever  $txs_1$  was just mined by  $N_j$  and  $R - r$  rounds are left till the end of the game, then for some  $C$ :

- $N_j$  chooses a strategy  $\sigma_1$  where he continues the current chain of the state  $\mathcal{S}$ , thus:

$$\mathbb{E}_j(\sigma_1, \mathcal{S}, r+1) \geq \lambda_j(\text{reward}_j(\mathcal{S}) + f + B + (R - r - 1)\lambda_j(f + B)) + C.$$

- when the  $N_j$  “forks” himself, then at least one of the blocks  $txs_1$  or  $txs_u$  will be canceled out in the final chain, therefore for any strategy  $\sigma_2$  that involves forking  $txs_1$ :

$$\mathbb{E}_j(\sigma_2, \mathcal{S}, r+1) \leq \lambda_j(\text{reward}_j(\mathcal{S}) - (f_1 + B) + f_2 + B + (R - r - 1)\lambda_j(f + B)) + C.$$

In conclusion  $\mathbb{E}_j(\sigma_1, \mathcal{S}, r+1) > \mathbb{E}_j(\sigma_2, \mathcal{S}, r+1)$ , unless  $f_2 - f \geq f_1 + B$ .

Now, since  $N_j$  that already mined  $txs_1$  will not fork himself in the first round, it is easy to see that the same follows in the next round. ◀

## 5.4 Only a High-Cost Reward May Encourage Miners to Fork

The next result shows that it is not possible to credibly threaten with forks without high fees. In particular, we show that for any miner with mining power  $\lambda_j$  that *considers* mining the block  $txs_1$ , any forking threat in the game where  $txs_{p_1}$  was not created, will not be credible unless  $f_2 - f \geq \lambda_j(f + B)$ , as the miner that mined the transaction will continue to mine his transaction.

► **Theorem 2.** *Let  $\Gamma_F(\mathcal{S}, r+1)$  be a subgame in a state  $\mathcal{S}$  that contains only a single chain of length  $r$  consisting of  $r-1$  unrelated transaction sets  $txs_u$  and one (just mined)  $txs_1$  (mutually exclusive with  $txs_2$  with  $\text{reward}(txs_2) > \text{reward}(txs_1) > \text{reward}(txs_u)$ ) mined by some miner  $N_j$ . The  $txs_{p_1}$  was not created and  $r \leq T$ . Other miners will not fork  $txs_1$ , unless  $f_2 - f \geq \lambda_j(f + B)$ , where  $\lambda_j$  is the mining power of the miner  $N_j$ .*

**Proof.** For brevity, the proof of this statement was moved to the full version of this paper. ◀

## 5.5 Without a High-Cost Reward, All Players Mine $txs_1$

As we already observed, once  $txs_1$  is mined, it will not be forked unless the bribing fee is sufficient. We will show that for a sufficiently large number of rounds  $T$ , all of the players will mine transaction set  $txs_1$  in the first round. A similar result was introduced in [23], but we prove that this result still holds in the game with forks.

► **Theorem 3.** *Let  $\Gamma_F(\mathcal{S}, 1)$  be a subgame where none of the miners decides to create  $txs_{p_1}$  before the first round, and the bribing fee is not too high, i.e.  $f_2 - f < 10^{-2}(f + B)$ . What is more  $f_1 > f$ , and if we define  $Y = \sum_{j=i: \lambda_j > 0.01, f_2 - f < \frac{f_1 - f}{\lambda_j}} \lambda_j$ , then  $T, Y$  are big enough, such that  $(1 - 1.01(1 - Y)^T) > 0$ . Every miner with  $\lambda_j > 0.01$  will decide to mine  $f_1$  in the first round.*

**Proof.** In the game where none of the miners decides to create the transaction set  $txs_{p_1}$ , miners may choose to mine  $txs_u$  and  $txs_1$  in all rounds, or  $txs_2$  only after round  $T$ . Now, since the game contains only one pair of mutually exclusive transactions  $txs_1, txs_2$  with  $\text{reward}(txs_2) > \text{reward}(txs_1)$ , then by Conjecture 1 players can start to fork only when  $txs_1$  appears on the blockchain. What is more, since  $f_2 - f < 10^{-2}(f + B)$ , by Theorem 2, whenever some player with  $\lambda_j > 10^{-2}$  successfully mines  $txs_1$  in a chain of length  $\leq T$ , none of the players will decide to fork his block.

We prove that in the above game miners with collective mining power at least  $Y$  will decide to mine  $txs_1$  in rounds  $\{T, T-1, \dots, 1\}$  if not mined up to this point. Let's take any miner with  $\lambda_i > 10^{-2}$  that makes a decision in round  $T-k$ , for  $k \in \{0, \dots, T-1\}$ .

As already mentioned, once he successfully mines the block  $txs_1$ , it will not be forked. In round  $T$ , whenever the block  $txs_1$  was not mined, then the miners had only mined  $txs_u$  so far ending up in a state  $\mathcal{S}_T$ . Then for some constant  $C$ , the utility of the player  $N_i$  in all strategies  $\sigma_1$  that choose to mine  $txs_1$  in  $\mathcal{S}_T$ , and all strategies that choose to mine  $txs_u$  in  $\mathcal{S}_T$ :

$$\mathbb{E}_i(\sigma_1, \mathcal{S}_T, T) \geq \lambda_i(\text{reward}_i(\mathcal{S}_T) + f_1 + B + \lambda_i(R - T - 1)(f + B)) + C,$$

$$\mathbb{E}_i(\sigma_2, \mathcal{S}_T, T) \geq \lambda_i(\text{reward}_i(\mathcal{S}_T) + f + B + \lambda_i(f_2 + B) + (R - T - 2)\lambda_i(f + B)) + C.$$

Now,  $\mathbb{E}_i(\sigma_2, \mathcal{S}_T, T) < \mathbb{E}_i(\sigma_1, \mathcal{S}_T, T)$  only if  $(*)f_2 - f \geq \frac{f_1 - f}{\lambda_i}$ . This implies that miners with collective mining power at least  $Y$  will prefer to mine  $txs_1$  in this round.

In round  $T - k$ , where  $k > 0$ , whenever the block  $txs_1$  was not mined, then the miners had only mined  $txs_u$  so far, ending up in a state  $\mathcal{S}_{T-k}$ . Then for some constant  $C$ , the utility of the player  $i$  in all strategies  $\sigma_1$  that choose to mine  $txs_1$  in  $\mathcal{S}_T$ , and all strategies that choose to mine  $txs_u$  in  $\mathcal{S}_T$ :

$$\begin{aligned} \mathbb{E}_i(\sigma_1, \mathcal{S}_{T-k}, T - k) &= \lambda_i(\text{reward}_i(\mathcal{S}_{T-k}) + f_1 + B + \lambda_i(k - 1)(f + B) + \\ &\quad \lambda_i(R - T + k - 1)(f + B)) + C, \end{aligned}$$

$$\begin{aligned} \mathbb{E}_i(\sigma_2, \mathcal{S}_{T-k}, T - k) &\leq \lambda_i(\text{reward}_i(\mathcal{S}_{T-k}) + f + B + \lambda_i(k - 1)(f + B) + \lambda_i(f_2 + B) + \\ &\quad (R - T + k - 2)\lambda_i(f + B)) + C. \end{aligned}$$

Similiary, the above equation implies that at least  $Y$  miners will prefer to mine  $txs_1$  in this round.

Now, after the first round there are  $T$  rounds till the moment of mining  $txs_2$ , player's  $N_i$  benefit from mining  $txs_1$  (with  $\lambda_i > 0.01$ ) and not waiting for  $txs_2$  is at least  $benefit = \lambda_i(f_1 + B) - \lambda_i(1 - Y)^T(f_2 + B)$ , and since  $f_2 - f < 10^{-2}(f + B)$ , then  $benefit \geq \lambda_i(f_1 + B - (1 - Y)^T(1.01f + 1.01B))$ . Now, assuming that  $f_1 > f$  we have  $benefit \geq \lambda_i(f(1 - 1.01(1 - Y)^T) + B(1 - 1.01(1 - Y)^T))$ . This implies that whenever  $(1 - 1.01(1 - Y)^T) > 0$ , then all miners with  $\lambda_i > 0.01$  will mine  $txs_1$  in the first round. ◀

A similar results holds in any state where sufficiently large number of  $T - r + 1$  rounds are left till the round  $T$ ,  $txs_{p_1}$  was not created in the game (or  $txs_u$  was mined in the first round), and  $txs_1$  was not mined yet. We leave it as a lemma without a proof.

► **Lemma 5.** *Given a game with forks  $\Gamma_F(\mathcal{S}, r)$  with  $r < T$ , where none of the miners decides to create  $txs_{p_1}$  before the first round (or  $txs_u$  is mined in the first round), and the bribing fee is not too high, i.e.  $f_2 - f < 10^{-2}(f + B)$  and given  $Y = \sum_{j=i:\lambda_j > 0.01, f_2 - f < \frac{f_1 - f}{\lambda_j}} \lambda_j$ ;  $T - r + 1, Y$  are big enough, such that  $(1 - 1.01(1 - Y)^{T-r+1}) > 0$ , every miner with  $\lambda_j > 0.01$  will decide to mine  $f_1$  in this round.*

## 5.6 Discouraging Miners to Mine $txs_1$

In the previous sections, we have shown that it is rather expensive to force the players not to mine  $txs_1$  in the first round, even when the players can fork this transaction. In this section, we leverage the self-penalty mechanism introduced in Section 3. The proof is inductive, and its base case starts in round  $T$ . For each round, we first show that the miner  $N_s$  with the highest mining power  $\lambda_s$  will not mine  $txs_1$ , as we assume that  $f_2 - f > \frac{f_1 - f}{\lambda_s}$ . Next, given a



sufficiently large penalty  $P > \lambda_s(f + B)$ , we show that the selected player  $N_s$  will fork the transaction  $tx_{s_1}$ , once it appears on the blockchain, even though it poses a risk of losing the block reward. Finally, we show that in this round all players other than the player  $N_s$  are afraid to mine  $tx_{s_1}$ , when the self-penalty transaction is on the chain.

► **Theorem 4.** *Let  $\Gamma_F(\mathcal{S}, 2)$  be a subgame where  $tx_{s_{p_1}}$  defined by a player  $N_s$  with mining power  $\lambda_s$  was mined in the first round with  $P > \lambda_s(f + B)$ . What is more  $f_2 - f > \frac{f_1 - f}{\lambda_s}$ , and  $\frac{f+B}{f_1+B} > 1 - \lambda_s^2$ . None of the miners will decide to mine  $tx_{s_1}$  in rounds  $2, \dots, T$ .*

**Proof.** For brevity, the proof of the theorem was moved to the full version of this paper. ◀

## 5.7 Encouraging the Strongest Miner to Use the Penalty Mechanism

Finally, we observe the benefit that comes from using the penalty mechanism. First for the miner with the strongest mining power  $\lambda_s$ , we observe that using the self-penalty mechanism and threatening others to mine the transaction  $tx_{s_1}$ , once it appears on the blockchain is beneficial for him whenever  $f_2 - f > \frac{2\bar{f} + 2(f_1 - f)}{\lambda_s} + \bar{f}$ . Next, for any miner with a smaller mining power, we show that merely the fact that he is threatened to mine  $tx_{s_1}$  can force them to skip mining this transaction.

► **Theorem 5.** *In the game with forks  $\Gamma_F$  that starts with an empty state  $\mathcal{S}$ , whenever  $f_2 - f > \frac{2\bar{f} + 2(f_1 - f)}{\lambda_s} + \bar{f}$ ,  $\frac{f+B}{f_1+B} > 1 - \lambda_s^2$ ,  $f_1 > f$ ,  $f_{p_2} > f$ ,  $\lambda_{min} > 0.05^{T/2}$ ,  $f_2 - f < 10^{-2}(f+B)$  and given  $Y = \sum_{j=i:\lambda_j > 0.01, f_2 - f < \frac{f_1 - f}{\lambda_j}} \lambda_j$ , it holds that  $(1 - 1.01(1 - (1 - Y)^{T/2})) > 0$ , the dominating strategy for all players in the game  $\Gamma_F$  is to mine  $tx_{s_{p_1}}$  with  $P > \lambda_s(f + B)$  created in the first round by the strongest player  $N_s$  with the mining power  $\lambda_s$ , then mine  $tx_{s_u}$  until round  $T$ , then  $tx_{s_2}$ ,  $tx_{s_{p_2}}$ , and  $tx_{s_u}$  until the end.*

**Proof.** By Theorems 3 and 4, utility of the player  $N_s$  that chooses to create  $tx_{s_{p_1}}$ ,  $tx_{s_{p_2}}$  and mine  $tx_{s_{p_1}}$  in the first round<sup>6</sup> (strategy  $\sigma_p$ ) is at least:

$$\mathbb{E}_s(\sigma_p, \mathcal{S}, 1) \geq -\lambda_{p_1} c_{p_1} \bar{f}_{p_1} + \lambda_s((m - c_{p_1})\bar{f} + B) + (\lambda_{p_1} + \lambda_s)F_2' + (1 - \lambda_s - \lambda_{p_1})F_1',$$

where  $F_1' = (\lambda_s(T - 1)(f + B) + \lambda_s(R - T)(f + B))$ ,

$$F_2' = (\lambda_s((T - 1)(f + B)) + \lambda_s(f_2 + B) - \lambda_{p_2} c_{p_2} \bar{f}_{p_2} + \lambda_s((m - c_{p_2})\bar{f} + B) + \lambda_i(R - T - 2)(f + B)).$$

Recall that we assume that all players with mining power less than 1% have collective power at most 5%. As the players with mining power more than 0.01 will prefer to mine  $tx_{s_1}$  in the first place when  $tx_{s_{p_1}}$  is not created, the utility of the player  $N_s$  that does not decide to create  $tx_{s_{p_1}}$  (strategy  $\sigma_1$ ) is at most (by Lemma 5 and the Theorem 4):

$$\mathbb{E}_s(\sigma_1, \mathcal{S}, 1) \leq ((1 - 0.05^{T/2})F_1 + 0.05^{T/2}F_2),$$

where  $F_1 = F_1' + \lambda_s(f_1 + B)$ ,  $F_2 = F_2' + \lambda_s(f + B)$ . Further, if  $\lambda_{p_1} c_{p_1} \bar{f}_{p_1} + \lambda_s(f + B) - \lambda_s c_{p_1} \bar{f} + F_1' + (\lambda_{p_1} + \lambda_s)(F_2' - F_1') > F_1 + 0.05^{T/2}(F_2 - F_1)$ , then  $\mathbb{E}_s(\sigma_p, \mathcal{S}, 1) > \mathbb{E}_s(\sigma_1, \mathcal{S}, 1)$ . This condition holds whenever:

$$(\lambda_{p_1} + \lambda_s)[\lambda_s f_2 - \lambda_s f_1] > 0.05^{T/2}[\lambda_s f_2 - \lambda_s f_1] + \lambda_{p_1} c_{p_1} \bar{f}_{p_1} + \lambda_s c_{p_1} \bar{f} + (\lambda_{p_1} + \lambda_s)(\lambda_{p_2} c_{p_2} \bar{f}_{p_2} + \lambda_s c_{p_2} \bar{f}) + \lambda_s(f_1 - f).$$

<sup>6</sup> In the analysis we omit the strategy where the player  $N_s$  creates  $tx_{s_{p_1}}$ ,  $tx_{s_{p_2}}$  and does not decide to mine  $tx_{s_{p_1}}$

What concludes that the following bribe is enough to encourage the strong miner to wait for  $txs_2$ :

$$f_2 - f > \frac{c_{p_1} \bar{f}_{p_1} + c_{p_1} \bar{f} + c_{p_2} \bar{f}_{p_2} + f_1 - f}{\lambda_s - 0.05^{T/2}} + c_{p_2} \bar{f}.$$

Now, if the  $txs_{p_1}, txs_{p_2}$  are created then every player  $N_i$  other than the player  $N_s$  may mine  $txs_{p_1}$ , once it is published (strategy  $\sigma_{p_1^*}$ ). When  $txs_{p_1}$  is successfully mined in the first round, then all miners will be encouraged to wait until  $txs_2$  may be mined after the  $T$ 'th round.  $1 - \lambda_{p_1} - \lambda_i$  miners may decide to mine  $txs_u$  (or  $txs_1$ ) in the first round. In this case, when the  $txs_u$  is mined, all other players will be able to mine at least  $f + B$  for the rest of the rounds  $\mathbb{E}_i(\sigma_{p_1^*}, \mathcal{S}, 1) \geq \lambda_i(f_{p_1} + B) + (\lambda_{p_1} + \lambda_i)F_2 + (1 - \lambda_{p_1} - \lambda_i)F$ , where  $F_2 = (T - 1)\lambda_i(f + B) + \lambda_i(f_2 + B) + \lambda_i(f_{p_2} + B) + (R - T - 2)\lambda_i(f + B)$  and  $F = (R - 1)\lambda_i(f + B)$ .

On other hand the players may first decide to mine either  $txs_u$  or  $txs_1$  in the first round (strategy  $\sigma_{1^*}$ ). In the worst case scenario the block with  $txs_1$  is not forked. What is more, whenever  $1 - \lambda_{p_1}$  miners decide to mine  $txs_u$  in the first round, then all miners with mining power more than 0.01 will make an attempt to mine  $txs_1$ . In conclusion, by Lemma 5 and the Theorem 4:

$$\mathbb{E}_i(\sigma_{1^*}, \mathcal{S}, 1) \leq \lambda_i(f_1 + B) + \lambda_{p_1}F_2 + (1 - \lambda_{p_1} - \lambda_i)((1 - 0.05^{T/2})F + 0.05^{T/2}F_2),$$

where  $F$  and  $F_2$  are defined as previously.  $\mathbb{E}_i(\sigma_{p_1^*}, \mathcal{S}, 1) > \mathbb{E}_i(\sigma_{1^*}, \mathcal{S}, 1)$  holds whenever:

$$\lambda_i(f_{p_1} + B) + \lambda_i(F_2 - F) > \lambda_i(f_1 + B) + (1 - \lambda_{p_1})0.05^{T/2}(F_2 - F)$$

Which holds for any  $f_{p_1} \geq f_1$  and  $\lambda_i > 0.05^{T/2}$ .

Now, by setting  $c_{p_1} = 1, c_{p_2} = 1, \bar{f}_{p_1} = f_1 - f, \bar{f}_{p_2} = \bar{f}$ , we get a condition  $f_2 - f > \frac{2\bar{f} + 2(f_1 - f)}{\lambda_s - 0.05^{T/2}} + \bar{f}$ , what for  $\lambda_s \approx 20\%$  and sufficiently large  $T/2$  concludes  $f_2 - f \gtrsim \frac{2\bar{f} + 2(f_1 - f)}{\lambda_s} + \bar{f}$ . ◀

## 6 Example Evaluation

Using the real-world data analysis of Bitcoin fees and hashpower distribution in major PoW blockchains (see discussion in the full version of this paper), we visualize the improvement our bound  $f_2 - f > \frac{2\bar{f} + 2(f_1 - f)}{\lambda_s} + \bar{f}$  brings compared to the previous result from [6], namely  $f_2 - f \geq \frac{f_1 - f}{\lambda_{min}}$ . Additionally, the Theorem 5 requires that  $f_2 - f < 10^{-2}(f + B)$  and there exists a player  $N_j$  with mining power  $\lambda_j > 0.01$  for which  $f_2 - f < \frac{f_1 - f}{\lambda_j}$ .

For example, let us assume that  $f_1 - f \approx \bar{f}$ , and set  $T > 110$ . Now, since  $\lambda_{min}$  can be fairly estimated to be  $\lambda_{min} < 10^{-12}$ , we can see that the attack without forking threats could cost in practice around  $10^{12}\bar{f}$ . On the other hand, the new bound requires only  $f_2 - f > \frac{2\bar{f} + 2(f_1 - f)}{\lambda_s} + \bar{f}$ , for  $\lambda_s \approx 0.2$ , this costs around  $f_2 - f > 21\bar{f}$ . The only condition left is that for some miner with  $\lambda_j > 0.01$ , the following condition must hold  $f_2 - f < \frac{f_1 - f}{\lambda_j}$ , but the data shows that miners that control approximately 1.5% – 2% of the total mining power usually exist, thus for a miner with mining power  $1\% < \lambda_j < 2\%$  it holds that  $f_2 - f < 50 \cdot \bar{f}$ . In summary, if we take any  $f_2$  that is larger than  $f$  by 21 up to 50 times, then the default strategy for all miners is to wait for the bribing transaction.

## 7 Related Work and Countermeasures

In the landscape of constructing financially stable systems on blockchain [21, 7], our work falls into the class of incentive manipulation attacks which have been widely applied to undermine blockchain’s security assumptions [22]. To the best of our knowledge, we are the first to combine feather forking attacks [8] with timelock bribing attacks on payment channel networks and to achieve a bribing cost that is approximately only constant times larger than the cost of an average transaction fee.

Incentive manipulation attacks on timelocked puzzles were introduced with the so-called timelock bribing attack [23]. Later, Avarikioti et al. [6] applied timelock bribing attacks in payment channel networks, such as the Lightning Network and Duplex Micropayment Channels, and proposed countermeasures. Our work extends [6], modifying the timelock bribing attack for payment channels to facilitate a miner bribing strategy that incorporates feather forking. As a result, our work reduces the cost of bribing attacks significantly in comparison to [6], i.e., the cost is now inversely proportional to the mining power of the largest miner instead of the smallest miner which results in at least 1000 times smaller bribes. Our model is similar to the one in [17] that introduced forks, but we were able to craft reasonable assumptions for the PoW blockchains which secured a *unique* NE solution. In particular, we restrict the strategy of the miner by forbidding him to conduct shallow forks and allowing him to fork only in a case when the player strictly profits from conducting the fork action (compare Sections 4.1.7, 4.1.8).

The bribing strategies for the payment channels are similar in their nature to the bribing strategies for the HTLC mechanism. Perhaps the closest to our work is [31], where the authors introduced a way to bribe HTLCs, leveraging the power of smart contracts and feather forking. The cost of the attack in [31] is, however, proportional to the sum of the fees ( $\gtrsim \sum_{i=1}^T f \cdot \lambda_{max}$ ) of all blocks before the deadline  $T$ . In contrast, we achieve a cost proportional to the cost of fees of a single block ( $\gtrsim \frac{f_1 - f}{\lambda_s}$ ).

Furthermore, MAD-HTLC [29] underlined the vulnerability of HTLCs to bribing attacks, achieving the same attack cost as [6], specifically  $\approx \frac{f_1 - f}{\lambda_{min}}$ . MAD-HTLC presupposes that the minimum fraction of mining power controlled by a single user,  $\lambda_{min}$ , is at least 0.01, to achieve low bribing costs. This is, however, an impractical assumption, as the analysis of the actual data (see discussion in the full version of this paper) shows that  $\lambda_{min}$  can be reasonably estimated to be less than  $10^{-12}$ , making the bribing attack exceedingly expensive. The reduction of the bribing costs *Bribe & Fork* achieves in comparison to MAD-HTLC is similar to that of [6] analyzed above.

MAD-HTLC additionally proposed a countermeasure for bribing attacks where miners are allowed to claim the locked coins in the HTLC in case a party misbehaves, similar to [6]. Later, He-HTLC [30] pointed out that MAD-HTLC is susceptible to counter-bribing attacks. In particular, one party may (proactively) collude with the miners to cooperatively steal the coins of the counterparty in the MAD-HTLC construction. He-HTLC also proposed a modification on MAD-HTLC to mitigate the counter-bribing attack: now the coins are partially burned in case of fraud instead of being fully awarded to the miners. Recently, Rapidash [9] revisited the counter-bribing attack and proposed yet another improvement on He-HTLC. These works are orthogonal to ours as the proposed attacks apply only to the specific MAD-HTLC construction and not to Lightning Channels that are the focus of this work. Furthermore, our focus is not on designing countermeasures against timelocked bribing attacks. Instead, we demonstrate how employing feather forking can make timelocked bribing attacks very cheap for the attacker, therefore highlighting the need for robust mitigating strategies.

Nonetheless, it is crucial to acknowledge that the previously mentioned countermeasures can be used to defend against *Bribe & Fork*—inheriting their respective vulnerabilities. For example, one can employ the mitigation technique for timelocked bribing on the Lightning Network proposed by Avarikioti et al. [6]. In our model, this countermeasure ensures that announcing  $txs_2$  also involves revealing a secret that anyone can use to claim the money before time  $T$ . This implies that if  $txs_2$  is announced in the mining pool before time  $T$ , all the money to be collected only after time  $T$  can be immediately claimed by another party. We assert, without proof, that the same countermeasure mechanism remains effective even in a model that considers forks. Intuitively, the “strong” miner in our analysis does not benefit from waiting for the bribing transaction if it is not announced, thus preventing the creation of the self-penalty transaction. Conversely, if the transaction is announced and the secret is revealed, any (winning) miner could claim the reward.

## 8 Conclusions and Future Work

In conclusion, our work reexamines the vulnerability of PCNs to bribing attacks and introduces a modified attack leveraging the threat of forking. We introduce a formal model of a mining game with forking extending the conditionally timelocked games introduced by Avarikioti et al. [6]. In particular, in our extended model, miners not only choose which transactions to mine in each round but also decide whether to continue existing chains or initiate forks. In this model, we demonstrate that the cost of the bribing attack can be significantly reduced compared to the previous analysis. In more detail, it can be reduced from  $\frac{f_1 - f}{\lambda_{min}}$  to approximately  $\frac{2\bar{f} + 2(f_1 - f)}{\lambda_s}$ , where  $\bar{f}$  represents the cost of an average fee for a single transaction and  $\lambda_s$  denotes the reduction factor compared to significantly smaller  $\lambda_{min}$  calculated in previous work [6]. To validate our findings, we empirically analyze the historical data of real-world blockchain implementations. This analysis confirms that staging a bribing attack on a PCN is significantly less costly (approximately 125%) than considered previously.

The results of our study have implications for the design and implementation of PCNs, as well as for the broader applications of timelocked contracts, e.g., atomic swaps. Our findings underscore the need for proactive measures to mitigate the risk of bribing attacks.

Possible avenues for future research include exploring whether our penalty mechanism implementation can be implemented without the honest majority assumption or whether our results still hold in the presence of more general abandon rules. Another interesting question is whether our results extend in a Proof-of-Stake setting.

---



## References

- 1 Lukas Aumayr, Ozgur Ersoy, Andreas Erwig, Sebastian Faust, Kristina Hostakova, Matteo Maffei, Pedro Moreno-Sanchez, and Sabrina Riah. Generalized bitcoin-compatible channels. *Cryptology ePrint Archive*, 2020:476, 2020. URL: <https://eprint.iacr.org/2020/476>.
- 2 Lukas Aumayr, Ozgur Ersoy, Andreas Erwig, Sebastian Faust, Kristina Hostakova, Matteo Maffei, Pedro Moreno-Sanchez, and Sabrina Riah. Bitcoin-compatible virtual channels. In *IEEE Symposium on Security and Privacy*, 2021. URL: <https://eprint.iacr.org/2020/554.pdf>.
- 3 Lukas Aumayr, Sri AravindaKrishnan Thyagarajan, Giulio Malavolta, Pedro Moreno-Sanchez, and Matteo Maffei. Sleepy channels: Bi-directional payment channels without watchtowers. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 179–192, 2022.

- 4 Zeta Avarikioti, Eleftherios Kokoris Kogias, Roger Wattenhofer, and Dionysis Zindros. Brick: Asynchronous incentive-compatible payment channels. In *International Conference on Financial Cryptography and Data Security*, 2021. URL: <https://fc21.ifca.ai/preproceedings/50.pdf>.
- 5 Zeta Avarikioti, Orestis S. T. Litos, and Roger Wattenhofer. Cerberus channels: Incentivizing watchtowers for bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 346–366. Springer, 2020. URL: [https://link.springer.com/chapter/10.1007/978-3-030-60276-7\\_18](https://link.springer.com/chapter/10.1007/978-3-030-60276-7_18).
- 6 Zeta Avarikioti and Orfeas Stefanos Thyfronitis Litos. Suborn channels: Incentives against timelock bribes. In *Financial Cryptography and Data Security - 26th International Conference, FC 2022, Grenada, May 2-6, 2022, Revised Selected Papers*, volume 13411 of *Lecture Notes in Computer Science*, pages 488–511. Springer, 2022. doi:10.1007/978-3-031-18283-9\_24.
- 7 Iddo Bentov and Ranjit Kumaresan. How to use bitcoin to design fair protocols. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference*, volume 8617 of *Lecture Notes in Computer Science*, pages 421–439. Springer, 2014. doi:10.1007/978-3-662-44381-1\_24.
- 8 Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A. Kroll, and Edward W. Felten. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In *2015 IEEE Symposium on Security and Privacy*, pages 104–121, 2015. doi:10.1109/SP.2015.14.
- 9 Hao Chung, Elisaweta Masserova, Elaine Shi, and Sri AravindaKrishnan Thyagarajan. Rapi-dash: Foundations of side-contract-resilient fair exchange. *Cryptology ePrint Archive*, Paper 2022/1063, 2022. URL: <https://eprint.iacr.org/2022/1063>.
- 10 Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, Dawn Song, and Roger Wattenhofer. On scaling decentralized blockchains. In *International Conference on Financial Cryptography and Data Security*, pages 106–125. Springer, 2016.
- 11 Christian Decker, Rusty Russell, and Olaoluwa Osuntokun. eltoo: A simple layer2 protocol for bitcoin. <https://blockstream.com/eltoo.pdf>, 2019.
- 12 Christian Decker and Roger Wattenhofer. A fast and scalable payment network with bitcoin duplex micropayment channels. In *Stabilization, Safety, and Security of Distributed Systems*, pages 3–18. Springer, 2015.
- 13 Stefan Dziembowski, Lisa Eckey, Sebastian Faust, and Daniel Malinowski. Perun: Virtual payment hubs over cryptocurrencies. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 344–361. IEEE, 2019.
- 14 Stefan Dziembowski, Sebastian Faust, and Kristína Hostáková. General state channel networks. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 949–966. ACM, 2018.
- 15 Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick McCorry, and Arthur Gervais. Sok: Layer-two blockchain protocols. In Joseph Bonneau and Nadia Heninger, editors, *Financial Cryptography and Data Security - 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10-14, 2020 Revised Selected Papers*, volume 12059 of *Lecture Notes in Computer Science*, pages 201–226. Springer, 2020. doi:10.1007/978-3-030-51280-4\_12.
- 16 Michael Jounen, Nicolas Larangeira, and Koji Tanaka. Lightweight virtual payment channels. In *Cryptology and Network Security*, pages 365–384. Springer International Publishing, 2020.
- 17 Dimitris Karakostas, Aggelos Kiayias, and Thomas Zacharias. Blockchain bribing attacks and the efficacy of counterincentives, 2024. arXiv:2402.06352.
- 18 Sishan Long, Soumya Basu, and Emin Gün Sirer. Measuring miner decentralization in proof-of-work blockchains. *arXiv preprint arXiv:2203.16058*, 2022.
- 19 Antonio Magnani, Luca Calderoni, and Paolo Palmieri. Feather forking as a positive force: incentivising green energy production in a blockchain-based smart grid. In *Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems*, pages 99–104, 2018.

- 20 Andrew Miller. Feather-forks: enforcing a blacklist with sub-50% hash power. URL: <https://bitcointalk.org/index.php?topic=312668.0>.
- 21 Andrew Miller and Iddo Bentov. Zero-collateral lotteries in bitcoin and ethereum, 2017. [arXiv:1612.05390](https://arxiv.org/abs/1612.05390).
- 22 Michael Mirkin, Yan Ji, Jonathan Pang, Arian Klages-Mundt, Ittay Eyal, and Ari Juels. Bdos: Blockchain denial of service, 2020. [arXiv:1912.07497](https://arxiv.org/abs/1912.07497).
- 23 Tejaswi Nadahalli, Majid Khabbazi, and Roger Wattenhofer. Timelocked bribing. In *Financial Cryptography and Data Security - 25th International Conference, FC*, volume 12674 of *Lecture Notes in Computer Science*, pages 53–72. Springer, 2021. doi:10.1007/978-3-662-64322-8\_3.
- 24 Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. URL: <http://bitcoin.org/bitcoin.pdf>.
- 25 Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments. <https://lightning.network/lightning-network-paper.pdf>, January 2016.
- 26 Yahya Shahsavari, Kaiwen Zhang, and Chamseddine Talhi. A theoretical model for fork analysis in the bitcoin network. In *IEEE International Conference on Blockchain, Blockchain 2019, Atlanta, GA, USA, July 14-17, 2019*, July 2019. doi:10.1109/Blockchain.2019.00038.
- 27 Santhi Shalini and H Santhi. A survey on various attacks in bitcoin and cryptocurrency. In *2019 International Conference on Communication and Signal Processing (ICCSP)*, pages 0220–0224. IEEE, 2019.
- 28 Joseph Spilman. Anti dos for tx replacement. <https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2013-April/002433.html>, 2013. Accessed: 2020-11-22.
- 29 Itay Tsabary, Matan Yechieli, Alex Manuskin, and Ittay Eyal. MAD-HTLC: because HTLC is crazy-cheap to attack. In *42nd IEEE Symposium on Security and Privacy, SP*, pages 1230–1248. IEEE, 2021. doi:10.1109/SP40001.2021.00080.
- 30 Sarisht Wadhwa, Jannis Stoeter, Fan Zhang, and Kartik Nayak. He-htlc: Revisiting incentives in HTLC. In *30th Annual Network and Distributed System Security Symposium, NDSS*. The Internet Society, 2023. URL: <https://www.ndss-symposium.org/ndss-paper/he-htlc-revisiting-incentives-in-htlc/>.
- 31 Fredrik Winzer, Benjamin Herd, and Sebastian Faust. Temporary censorship attacks in the presence of rational miners. In *2019 IEEE European Symposium on Security and Privacy Workshops, EuroS&P Workshops*, pages 357–366. IEEE, 2019. doi:10.1109/EuroSPW.2019.00046.
- 32 Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 2014.

# Payment Censorship in the Lightning Network Despite Encrypted Communication

Charmaine Ndolo  

Dresden University of Technology, Germany

Florian Tschorsch  

Dresden University of Technology, Germany

---

## Abstract

The Lightning network (LN) offers a solution to Bitcoin’s scalability limitations by providing fast and private off-chain payments. In addition to the LN’s long known application-level centralisation, recent work has highlighted its centralisation at the network-level which makes it vulnerable to attacks on privacy by malicious actors. In this work, we explore the LN’s susceptibility to censorship by a network-level actor such as a malicious autonomous system. We show that a network-level actor can identify and censor all payments routed via their network by just examining the packet headers. Our results indicate that it is viable to accurately identify LN messages despite the fact that all inter-peer communication is end-to-end encrypted. Additionally, we describe how a network-level observer can determine a node’s role in a payment path based on timing, direction of flow and message type, and demonstrate the approach’s feasibility using experiments in a live instance of the network. Simulations of the attack on a snapshot of the Lightning mainnet suggest that the impact of the attack varies from mild to potentially dramatic depending on the adversary and type of payments that are censored. We analyse countermeasures the network can implement and come to the conclusion that an adequate solution comprises constant message sizes as well as dummy traffic.

**2012 ACM Subject Classification** Networks → Network privacy and anonymity; Security and privacy → Software and application security; Security and privacy → Distributed systems security

**Keywords and phrases** Lightning network, payment channel networks, censorship resistance

**Digital Object Identifier** 10.4230/LIPIcs.AFT.2024.12

### Supplementary Material

*Software:* <https://github.com/tud-dud/lightning-censorship-simulator>

## 1 Introduction

Bitcoin [25] and similar blockchain-based payment systems continue to enjoy significant popularity. While Bitcoin is to date the most popular based on its market capitalisation, it suffers from grave constraints with respect to scalability, which limit its ability to compete with traditional (centralised) payment systems. Layer 2 solutions are gaining traction as a feasible solution to the scalability challenges by enabling off-chain transactions. One such solution is the Lightning network (LN) [30] – a peer-to-peer (P2P) payment channel network (PCN) enabling fast, low-cost and private Bitcoin payments. It is a network of off-chain bilateral channels in which funds can move in either direction between the two channel partners. LN also implements multi-hop payments such that payments can be routed over multiple intermediate channels in cases where the sender and beneficiary of a payment do not have a direct channel. In order to offer a degree of payment privacy, all P2P communication subsequent to connection establishment is encrypted using the Noise [29] protocol framework. Furthermore, LN uses the Sphinx mix format [10] to implement onion routing of payments across the network. This means that, among others, routing nodes only know their predecessor and successor when forwarding a payment, but do not know if either is the source or destination of the payment.



© Charmaine Ndolo and Florian Tschorsch;  
licensed under Creative Commons License CC-BY 4.0  
6th Conference on Advances in Financial Technologies (AFT 2024).

Editors: Rainer Böhme and Lucianna Kiffer; Article No. 12; pp. 12:1–12:24

Leibniz International Proceedings in Informatics



LIPIC Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

A compelling selling point of decentralised P2P systems is their censorship-resistant nature due to their fundamental design, i.e., there is no single point of failure or owner. Nonetheless, blockchain-based cryptocurrencies such as Bitcoin have been subject to state-wide bans enforced via legal frameworks and/or technical means such as aggressive protocol blocking.

While the broader topic of Internet censorship is by no means a new one, it remains highly relevant due to some of the censorship currently imposed across the globe, e.g., in China [15], India [46], Iran [4] or Turkmenistan [28]. Blockchain networks such as Bitcoin and Ethereum have also been shown to be vulnerable to censorship despite their design [22,44]. State censors employ different techniques such as high prices, notoriously low broadband speeds [17,35] or various network-level techniques [15,46] to restrict access to services. Powerful tools such as Geneva [8] and OONI [13] are able to detect or even evade censorship based on a censor's identified strategies. If a censor using network-level tactics intends on bypassing common censorship-detection tools, the need for subtle, difficult to reproduce yet effective measures arises. Additionally – and aside from ethical concerns – such restrictions are detrimental to their national and international image and have the potential to spark unrest. Thus, a censor may instead look to discretely implement such a ban such that it either goes unnoticed or initial blame is put on other actors, e.g., application issues. Assuming that a certain level of operation (within the censor's area of jurisdiction) can be maintained, the censor may even be able to plausibly deny the fact that they are indeed tampering with network traffic.

In this work, we explore the Lightning network's susceptibility to censorship by a network-level actor such as a malicious autonomous system (AS). For the previously detailed reasons, we assume that the censor's goal is not to disrupt the entire network but to control participation in the LN within their domain. In doing so, the censor seeks to limit their impact on the day-to-day operations of the greater LN and avoid collateral damage. The censor strives to remain undetected as much as possible such that from an observer's perspective, e.g., a user issuing payments or network explorer, it is difficult to recognise that a given node is under attack but aims to maximise their impact on the censored nodes.

Our work expands on previous work by von Arx et al. [43] in which they showed that application messages can be identified based on traffic analysis. We first confirm that a network-level adversary is able to accurately classify LN traffic using the header data and flow direction of transmitted packets by implementing a rule-based classification program for live LN traffic. Our results indicate that it is possible to accurately identify LN messages in real time despite the fact that all P2P communication in the network is end-to-end encrypted. Based on this, we show how a network-level actor can censor all payments routed via their network using a simple state machine to determine if a packet should be dropped. All other LN operations, e.g., channel management, remain unaffected. Due to the atomic nature of the payment process in the LN, dropping select messages eventually results in payment failure without attempting alternative routing options. This result may not be adequate for a censor who does not want to tamper with third-party activity that just so happens to be traversing their network. Thus, we then show that it is possible for a network-level observer to determine a node's role in a payment path based on the timing and direction of flow as well as the message type. We use the information to enhance the attack such that a censor can selectively block payments, e.g., block intra-AS payments but permit inter-AS payments.

We implemented the attack as an efficient `netfilter` program and validated the attack's feasibility and performance in a private network as well as in the public testnet. Our experiments show that for rates of up to 1 payment per second, we are able to correctly determine a node's role in a payment path. While this rate may sound underwhelming, it exceeds the currently estimated payment rates in Lightning by five orders of magnitude.



Furthermore, simulations of the attack on the mainnet’s channel graph show that the impact on the broader network is almost non-existent in the case of selective censorship. Based on reviewing current state-of-the-art traffic fingerprinting protection measures, we discuss and verify possible mitigation strategies for the LN. We come to the conclusion that an adequate solution entails implementing some form of cover traffic and constant-size messages in the network similar to what is implemented in the Tor network.

To summarise, the following are our main contributions:

1. We show that network packets can accurately be mapped to the corresponding LN message types using the payload length and sequence of messages in real time;
2. We exploit timing information and type of message to identify an on-path node’s role in a payment path;
3. Based on the preceding contributions, we present a censorship attack on the LN that is founded on selectively dropping network packets identified to be related to payments;
4. We implement the attack, deploy it in a private Lightning network and report on our findings. We evaluate the attack using our implementation and simulations; and
5. We analytically discuss possible countermeasures the LN can implement and derive recommendations for the network.

The remainder of this paper is structured as follows. We provide a pertinent introduction to the LN and present our system model in Section 2. The core of this work is Section 3 in which we describe a censorship attack on the LN and evaluate it comprehensively in Section 4. We discuss countermeasures for this attack vector in Section 5 and provide an overview of related work in Section 6. We conclude this work and discuss avenues for future work in Section 7.

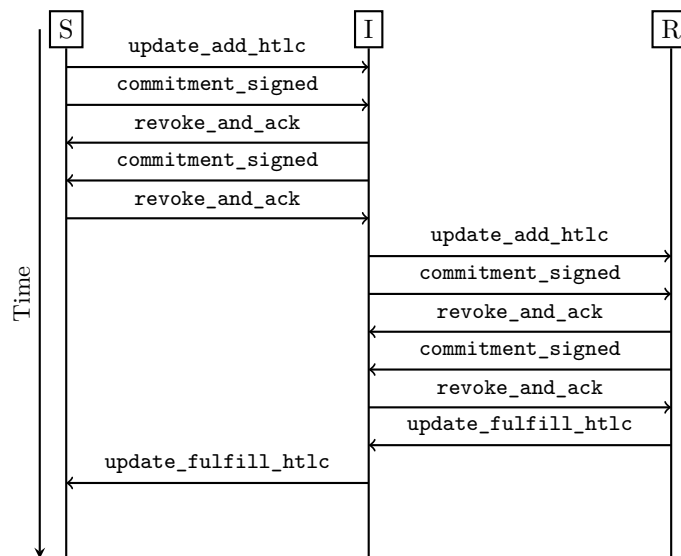
## 2 Background and system model

We provide the reader with a pertinent introduction to the Lightning network in Section 2.1; we refer the reader to [1, 30, 31] for a comprehensive introduction beyond the scope of this work. We briefly analyse the network topology in Section 2.2 and describe our threat model in Section 2.3.

### 2.1 The Lightning network

The Lightning network (LN) is a peer-to-peer (P2P) network of bilateral off-chain payment channels, i.e., a payment channel network (PCN). A payment channel signifies a financial relationship between a pair of nodes in which a set number of funds (the channel’s *capacity*) is committed via a transaction on the Bitcoin blockchain. Lightning payments alter the distribution of the channel’s capacity (*balances*) between the two endpoints. Payments in the LN can be routed via multiple hops for a fee that is independently set by each node. In order to route payments securely over multiple hops, payments are secured by Hashed Timelock Contracts (HTLCs), which guarantee that payments are made atomically, i.e., a payment either succeeds at all hops or fails at all hops. An HTLC is basically a conditional payment that can be claimed by producing a preimage that is revealed by the payment’s beneficiary. During channel establishment, each node defines and announces how long they are willing to wait for an HTLC to be resolved – the *time lock*. If the *time lock* expires before the HTLC is resolved, the HTLC expires and is settled on-chain which requires a forceful closure of the affected channel.

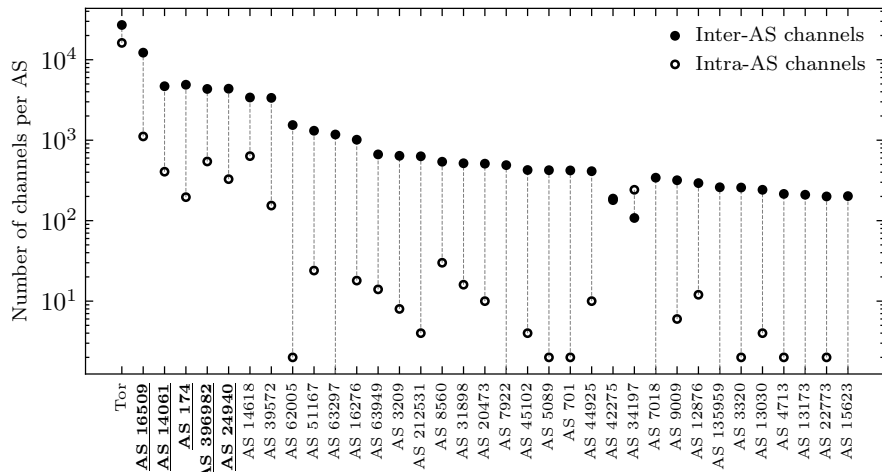
## 12:4 Payment Censorship in the Lightning Network Despite Encrypted Communication



■ **Figure 1** The sequence of messages exchanged for a payment between a sender (*S*) and recipient (*R*) routed via an intermediate hop (*I*). The `update_fulfill_htlc` message is only sent in the case of a successful payment and is replaced by an `update_fail_htlc` message otherwise.

Once a payment channel has been established, an arbitrary number of payments can be made over the channel. Finding a suitable path for a payment is an essential part of LN and is delegated to the sender of a payment. Based on the public channel graph, the sender tries to find a path connecting them to the recipient of the payment. For the sake of illustration, the following assumes a payment from node *S* to node *R* made via an intermediate node *I*. *S* begins by encoding the calculated path in an onion packet using the Sphinx message format [10], i.e., a packet with multiple layers of encryption that each identify the next hop on the path. Forwarding nodes along the path hence only know their predecessor and successor on the path, but do not know if either is the payment's source or destination. *S* initiates the payment by constructing an HTLC and sending it in an `update_add_htlc` message with the onion packet to *I*. Upon receipt, *I* decrypts the topmost layer to receive its payload and prepares to forward the `update_add_htlc` message to the next hop. However, *I* will only forward it to the next hop after the new conditional payment is reflected in the *S* – *I* channel's state. The state update must also be irrevocably committed by both nodes using a handshake of `commitment_signed` and `revoke_and_ack` messages as shown in Figure 1.

Once the state updates have been successfully completed, *I* forwards the remaining onion packet to *R* in a new `update_add_htlc` message. *I* and *R* then negotiate the new state in the same way *S* and *I* did (cf. Figure 1). As soon as *I* and *R* conclude the handshake necessary for the channel update, *R* sends an `update_fulfill_htlc` message to *I*. The message contains the preimage needed by each hop to settle the HTLC with its channel partner. The `update_fulfill_htlc` is propagated to all hops along the path in reverse order such that each hop can redeem the conditional payment. In the event of an error along the way, e.g., due to insufficient balances or time lock, a node will immediately send an `update_fail_htlc` to its predecessor instead, which will be propagated to all preceding nodes as well.



■ **Figure 2** The sum of channels where both endpoints belong to different ASs and the sum of channels where both belong to the same AS for the 35 ASs with the highest number of channels. The top five ASs in the network w.r.t. to the number of channels are underlined and printed in bold.

## 2.2 Topology

It is well-known that the application level of the LN is highly centralised [21, 34, 41]. That is, while the network is considerably large in terms of the number of nodes and channels, most payments are routed via a small subset of available channels. This has been shown to be detrimental to the network’s privacy goals and overall resilience [19, 26]. Recent work [43] revealed that the network layer is similarly centralised, with just a handful of ASs technically being able to compromise payment privacy. For instance, 80% of all Lightning channels are hosted at just five ASs.

As gaining a deeper understanding of the topological structure may prove to be useful to discover potential censorship strategies, we examined the distribution of channels to ASs based on a snapshot of the mainnet’s channel graph on 12 January 2024. The network comprised 12,781 nodes and 112,958 channels after reducing it to its largest strongly connected component. We pruned all nodes (and their channels) that had not announced at least one public network address from the obtained channel graph, which leaves approximately 22% of the nodes in the channel graph. We then mapped each node’s announced address to the corresponding AS using the GeoLite2 database.<sup>1</sup> We examined the distribution of node degrees across AS and find that all high-degree ( $> 500$  channels) nodes belong to different ASs. Further, we analysed the share of channels in which both endpoints belong to the same AS and depict the results in Figure 2. The figure shows the total number of channels that are shared by two different ASs and the total number of channels that belong to the AS alone. Except for nodes connecting to the network over Tor and a handful of ASs, e.g., AS 34197 or AS 42275, most channels in Lightning are between a pair of ASs and not within the same AS.

<sup>1</sup> Available at <https://www.maxmind.com> (accessed on 12 January 2024).

### 2.3 Adversary Model

We explore the feasibility of imposing censorship of the LN in this work. As previously mentioned, we presume that the censor aims to impose (from their perspective) effective censorship within their area of jurisdiction. In other words, the censor is not interested in disrupting the greater LN, but only controlling Lightning’s operation in their sphere of influence. Additionally, the censor wants to maintain plausible deniability and hence looks to implement the ban such that a certain level of operation is upheld within their domain despite the ongoing censorship. This is why applying less sophisticated methods such as port and IP blocking are out of the question for the censor.

Similar to multiple related works [27, 43], we assume a powerful yet malicious network-level adversary such as an AS or a party cooperating with an AS. While the attack can be executed by any adversary with access to network-level traffic, e.g., an operator of a Lightning node, the impact and significance of the attack is directly related to the adversary’s scope of influence. The adversary’s foremost goal is to control activity in and access to the LN within their area of influence. For instance, this may be to enforce a controversial ban on cryptocurrencies.

The adversary expects that all inter-peer communication is end-to-end encrypted as per the Lightning specifications [1]. The adversary is only interested in LN nodes using a clearnet address because of the fixed-size cells transmitted by the Tor network. Furthermore, we assume that all nodes are operating on the default port: Transmission Control Protocol (TCP) port 9735 [1]. In case a node is using a non-default port, the adversary may use publicly-available data to trivially identify the port in use. Similarly, the adversary can refer to such data to learn which client implementation a node is running or infer the client [24]. Knowledge of the client implementation in use is, however, strictly not necessary.

We focus on an adversary that fully controls at least one AS network. The network-level adversary can observe and inspect all communication sent over their network; it is however encrypted by the application layer. As the adversary wants to minimise the risk of detection, blocking all traffic on port 9735 would be self-defeating. Instead, and in order to maintain a level of operation and plausible deniability, the adversary is capable of executing refined filtering techniques such as selective packet dropping.

### 2.4 Ethical Considerations

We would like to emphasise that the primary goal of this work is to contribute to further developing and improving the network for all Lightning users. Uncovering, presenting, and fixing potential issues in the network is a core part of that process. We do not see this work as an instruction manual for adversaries and strongly disapprove of any misappropriation of our work. It is for this reason that we have decided to not make our proof-of-concept implementation of the attack available to the public. We believe that this paper contains enough information and details for the reader to reproduce with their own implementation. We made the code available during the peer review process and will consider doing the same to researchers upon request.

As far as the practical evaluation of the presented attack is concerned, we followed the guidelines of the Menlo report [5] and general security research best practices. In particular, with the exception of obtaining a network snapshot from our own node, we did not interact with the public mainnet in any way. We deployed a modified version of our proof-of-concept implementation to the testnet in order to validate the feasibility of the attack’s preliminary phase. However, at no point did we actually mount the attack in the testnet. All adverse

■ **Table 1** Comparison of expected message sizes (in bytes) as specified in [1] and actual captured message sizes. The sizes refer to messages containing exactly one HTLC. The node running LND sent all messages in two packets – an 18B payload followed by the remainder.

Message	Specifications	Message size (bytes)	
		LND v0.17.2-beta	CLN v23.11.2
<code>update_add_htlc</code>	1450	18 + 1468	1486
<code>commitment_signed</code>	162	18 + 116	134
<code>revoke_and_ack</code>	97	18 + 115	133
<code>update_fulfill_htlc</code>	72	18 + 90	108

experiments were conducted in our private network comprising only nodes we set up for the precise purpose. In order to evaluate the potential impact of our work on the main network, we followed a simulation-based approach using the obtained snapshot. The simulation mocks payment routing in the network by reconstructing the topology locally.

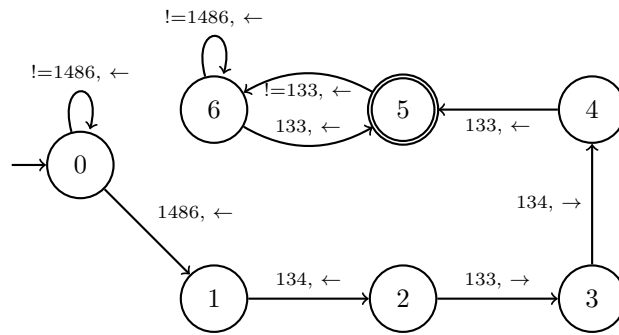
### 3 Censorship Attack

In the following, we present a novel censorship attack on a set of nodes in the LN. The attack leverages the fixed message sizes defined in the Lightning specifications [1] as well as its overall protocol design. This allows an adversary to accurately classify encrypted application traffic based on network-layer data without much effort in real time. Subsequently, we show how a network-level attacker can censor payments and enhance the attack with knowledge on a payment source and destination.

#### 3.1 Message Classification

A recent work [43] presented an attack on privacy in the LN based on monitoring network-layer traffic. The first step of the attack is to map TCP packets to application messages based on the payload lengths in combination with the sequence of observed packets. The censorship attack we demonstrate in Section 3.2 makes use of the same shortcoming. To that end, we take a closer look at identifying LN messages based on network-level observations. In what follows, we use HTLCs to exemplify the procedure. It, however, applies to other message types analogously.

Figure 1 illustrates the type and sequence of messages exchanged between two channel partners during the payment process. By generating and capturing LN packets in a private network in order to validate the feasibility of matching network packets to application messages, we established that none of the captured TCP payload lengths corresponds to the sizes defined in the BOLTs. Table 1 shows the actual message sizes for the two most popular clients [24, 47] – Lightning Network Daemon (LND) and Core Lightning (CLN). We observed that nodes running on LND sent each application message in two TCP packets, the first of which was always 18B. While the sizes of the messages sent by these clients differ from what is expected, they remain constant and hence allow us to identify the application messages based on the size, order of arrival and direction. The direction is not actually strictly necessary but it provides additional insights on the packet origin that we make use of to refine the adversary’s strategy. Additionally, the adversary should know which client software is running due to the slight differences in payload size. Inferring the implementation, however, can be done with reasonable effort by, e.g., analysing the transmitted payloads



■ **Figure 3** A state machine implementing the censorship rules for incoming payments. State transitions are defined by the (sum of the) TCP payload lengths and direction of flow ( $\rightarrow$  for egress traffic,  $\leftarrow$  for ingress traffic). All omitted transitions reset the state machine to its initial state.

(cf. Table 1) or using the public channel graph [24, 47]. For example, a packet with a  $134B$  TCP payload from  $A$  to  $B$  that is preceded by a  $1486B$  payload in the same direction and succeeded by  $18B$  and  $115B$  payloads in the reverse direction, is bound to have been a `commitment_signed` message. Furthermore,  $A$  is likely running CLN whereas  $B$  is almost certainly running LND.

### 3.2 Payment censorship in the LN

As per Section 2.3, the adversary wants the attack to go largely unnoticed and is indifferent towards third parties. This is why simply blocking or interfering with all LN traffic is not a viable strategy. However, given that an adversary is capable of identifying LN application messages by monitoring the network traffic, they can selectively interfere with the traffic passing their network.

In the following, we show how an adversary such as an AS can censor all payments involving nodes in their network while maintaining a degree of plausible deniability by preserving LN functionality in their network. Consequently, the adversary does not interfere with any messages pertaining to node management and channel management, e.g., open and close channel messages. By allowing nodes to operate Lightning channels, neither the affected nodes nor other observers have credible reason to put blame on the AS when issues with payments start to surface. For instance, a (suspicious) user inspecting the LN topology using a network explorer will not recognise that a malicious AS is suppressing its nodes' participation in the network.

However, the adversary pays close attention to all TCP traffic on port 9735 that is assumed to be payment-related using the method described in Section 3.1. The adversary must then interrupt the payment process in order to provoke application failures. The adversary prompts such failures by dropping select packets following the state machine in Figure 3 for each pair of source and destination. State 0 is the initial state in which the adversary waits for an `update_add_htlc` message which means that a payment is underway. The state machine's transitions are defined by the payload lengths of the series of messages exchanged between two nodes when a payment is being made. We choose to have the adversary drop the first `revoke_and_ack` message that is sent from the source to the recipient (cf. Figure 1). This is identified by arriving at the accepting state, state 5, after a series of messages. Although the adversary could drop the other payment messages, we opted for the `revoke_and_ack` message due its terminal position in the series of exchanged messages. We thus expect that

the adversary will make “correct” decisions. Note that Figure 3 depicts the rules applied by an adversary for incoming payments only. Outgoing payments can be censored analogously by reversing the direction of flow in the transition rules.

As discussed in Section 2.1, the `revoke_and_ack` message is sent in response to a state update; it revokes the previous state and acknowledges the new one. The payment process can therefore not proceed until either the recipient receives the `revoke_and_ack` message or the payment’s timelock expires. At this point, a node will no longer attempt to route a payment via an alternative path until the payment conclusively fails. Lightning clients thus initiate retransmissions of the unacknowledged `revoke_and_ack` message for as long as the payment is valid. This is why the adversary needs state 6 in Figure 3, i.e., to block all subsequent `revoke_and_ack` messages from getting to the recipient. Note that the effect is similar when the first `commitment_signed` from the sender to recipient is not acknowledged.

### 3.3 Selective censorship

So far, the adversary is able to monitor network traffic and block all payments routed via their network by dropping all `revoke_and_ack` messages. This is not yet quite satisfactory because the adversary’s goal is to remain largely unnoticed and minimise the collateral damage. The current strategy, however, defeats this objective. We thus refine the adversary’s packet dropping criteria by showing how to determine a node’s role in a payment based on network-level observations. The adversary can then selectively drop packets depending on the censored node’s position in the path. Besides contributing to the adversary remaining undetected, the ability to selectively drop LN messages using knowledge of a node’s position allows them to block payments based on origin and/or destination. For example, a malicious AS could let all payments pass that neither originate from nor are destined for their network, or allow all incoming payments but block outgoing payments.

An on-path node in the LN can occupy one of three roles for a given payment: *sender*, *intermediary* or *recipient*. When forwarding a payment in the network, intermediate nodes are not aware of other nodes’ or even their own positions in the path. While determining a node’s role has been subject of previous work [19], we are, to the best of our knowledge, the first to do so based on live network traffic. The adversary is hence able to use the node’s role for their decision on whether or not to block a packet. Based on the combination of packet direction, message type and position in the sequence of transmitted messages, it is possible to determine a node’s role as follows:

1. *sender*: a node is the initiator of a payment if it sends an outgoing `update_add_htlc` message “out of the blue”. In other words, if a sufficient amount of time  $t$  has passed since the last incoming `revoke_and_ack` message, we conclude that the current `update_add_htlc` message belongs to a separate payment. Due to the symmetric exchange of messages during payment routing (cf. Figure 1), an intermediate hop will always receive a `revoke_and_ack` message before offering an HTLC to the next hop in the path. If there is no such `revoke_and_ack` message, the purpose of the `update_add_htlc` message must be to initiate a new payment.
2. *intermediary*: if that less than  $t$  time has passed since receipt of a `revoke_and_ack` message when an `update_add_htlc` message is sent, i.e., an incoming `revoke_and_ack` was observed within time  $t$  before the outgoing `update_add_htlc`, the node is an intermediary.
3. *recipient*: when a node sends an `update_fulfill_htlc` message, it is the final destination of the payment if the previous (incoming) message was a `revoke_and_ack` message. We can conclude this because an intermediate hop will always send a new `update_add_htlc` message after receiving a `revoke_and_ack` so as to offer an HTLC to the next hop (cf.

## 12:10 Payment Censorship in the Lightning Network Despite Encrypted Communication

```
iptables -I INPUT -p tcp --dport 9735 -j NFQUEUE --queue-num 1
iptables -I OUTPUT -p tcp --dport 9735 -j NFQUEUE --queue-num 1
```

■ **Figure 4** The iptables rule set to direct all ingress and egress TCP traffic destined for port 9735 to queue 1.

Figure 1). Note that we can only know with certainty that a node is the destination during the settlement of the HTLC. This means that all HTLC offers need to be delivered in order to determine if a node is the recipient. The `update_fulfill_htlc` message only applies to successful payments but similar logic can be applied for failed payments when an `update_fail_htlc` message is sent. Using the aforementioned rules, the adversary can augment their attack and apply selective censorship.

### 3.4 Implementation

There are generally several feasible options to implement the attack. However, bearing the following properties in mind, we chose to implement the attack using the `netfilter`<sup>2</sup> framework.

1. *performance*: as the adversary only wants to interfere with relevant traffic, an efficient implementation is crucial. The filter thus needs to be capable of making in-flight decisions in a very efficient manner;
2. *scalability*: due to the LN's network-level centralisation, it is safe to assume that such a malicious AS observes up to thousands of channels concurrently. Furthermore, increasing the complexity of the state machine, e.g., to accommodate other message types, should not come at the cost of performance; and
3. *generalisability*: an implementation that does not rely on the specifics of an adversary's infrastructure.

While a hardware-level firewall, i.e., on the network interface card (NIC), may be attractive from a performance and scalability standpoint, the functionality generally depends on the specific NIC. In contrast, the `netfilter` project has been readily available in the Linux kernel since version 2.4 and provides, among others, the iptables module.

We implemented Figure 3 as a user space program using the `nfq-rs`<sup>3</sup> library in Rust. In order for the program to receive packets, we must first define an iptables rule set that is responsible for directing all TCP packets on port 9735 to a `netfilter` queue. The relevant rule set is shown in Figure 4. Note that we implemented the state machine in Figure 3 without regard for the source address as correlating independent TCP streams is out of the scope of this work. In other words, our program does not maintain state for different source addresses, and assumes that multiple unrelated payments are not received concurrently. If a packet is determined to be an incoming `revoke_and_ack` message, the program returns an `NF_DROP` verdict, i.e., the packet is discarded. All other packets are allowed to traverse the network stack by issuing the `NF_ACCEPT` verdict.

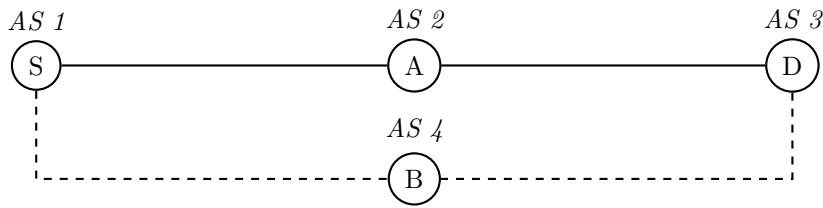
A notable alternative to the `netfilter` project is eXpress Data Path (XDP) [2, 16] – a framework that enables packet processing within extended Berkeley Packet Filter (eBPF) programs. XDP has been available in the Linux kernel as of Linux 4.8 and requires neither specialised hardware nor kernel bypass. It is an integrated fast path in the kernel stack

---

<sup>2</sup> <https://www.netfilter.org>

<sup>3</sup> <https://github.com/nbdd0121/nfq-rs>





■ **Figure 5** The private network environment used to validate the attack’s practicability. We assume that each node belongs to a different AS and  $A$ ’s AS is malicious. The solid path represents the preferred path between  $S$  and  $D$  w.r.t. the path selection parameters whereas the dashed path represents an alternative path.

and works together with the TCP/IP stack. Packet processing happens before meta-data structures are allocated by the kernel leading to high processing speeds [33]. As the majority of LN traffic is not dropped by the adversary, we do not expect to gain a significant improvement in performance from XDP. Nonetheless, and for the sake of comparison, we also implemented the attack as an eBPF program that makes use of XDP to process the incoming packets following Figure 3. However, as XDP inspects just ingress traffic, this implementation only features the “base” attack described in Section 3.2.

## 4 Analysis

We evaluated the attack using the proof-of-concept implementation on deployed Lightning nodes. We first describe our evaluation setup then analyse the attack in various scenarios as well as its impact on the greater network based on conducted simulations.

### 4.1 Evaluation Setup

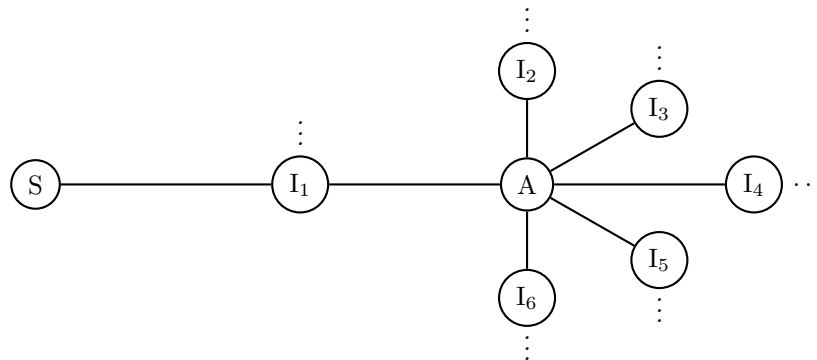
Due to the potentially destructive nature of the attack on the public network, we did not perform any measurements on the mainnet. All experiments were conducted either in the testnet or in a private network depending on the potential for harm and interference with other nodes. Both setups are described in the following.

#### 4.1.1 Regtest

We set up a private Bitcoin network in regression test (regtest) mode which allowed us to deploy the complete attack code without interacting or interfering with other nodes in the public networks. Furthermore, regtest mode allows users to create a private blockchain and mine blocks instantaneously as the mining difficulty is set to zero. We configured four different Lightning nodes in the network as shown in Figure 5 running on different machines within the same network.  $S, A, B$  were all running LND v0.17.2-beta while  $D$  was running CLN v23.11.2. We assume that all four nodes are in different ASs and that AS 2 is adversarial. The attack code is therefore attached to  $A$ ’s network interface.

#### 4.1.2 Testnet

In order to validate the attack in a realistic environment, we set up a node,  $A$ , running LND v0.17.2-beta in the public testnet. We strategically opened six balanced channels with moderate capacities between 300k sat and 500k sat to six nodes nodes,  $I_1, I_2, I_3, I_4, I_5, I_6$ ,



■ **Figure 6** Our two nodes, the sender  $S$  and the adversary  $A$ , in the testnet with public channels to other nodes  $I_i$  in the network. Three dots symbolise the intermediate nodes' channels with other unrelated nodes. The attack is deployed on  $A$ 's machine.

in the network and configured zero-fee routing policies. The channels were positioned to connect previously unconnected differently-sized hubs to each other in the hope of receiving routing requests.<sup>4</sup>

We deployed a modified version of the `netfilter` program on  $A$ 's machine and attached it to  $A$ 's only network interface. The program was modified such that instead of dropping a packet when the relevant state is reached, an entry is written to a log file notifying us that the packet would have been dropped. We also modified  $A$ 's LND source code to log whenever a `revoke_and_ack` message is received – no other changes were made to the client software. We reiterate that no harm was caused to other nodes or the network in general.

We also set up a second Lightning node,  $S$ , with a channel to one of the nodes  $A$  was connected to as illustrated in Figure 6. We abstain from a direct channel between our two nodes in order to route payments over the Internet. We then generated random payments worth 1100 sat from  $S$  to the nodes  $I_1, \dots, I_6$  using the `sim-ln` tool.<sup>5</sup> We chose this amount as its the lowest amount satisfying the minimum payment amount all involved nodes were willing to forward. Due to the topology, all of the payments coming from  $S$  could only be routed via  $A$ . This resulted in a total of 71 payments in the span of 24 hours that were all delivered successfully.

## 4.2 Feasibility

In the following, we look at the message classification efficacy of the approach described in Section 3.1. Hereafter, we discuss the practicability of the attack described in Section 3.2.

### 4.2.1 Accuracy

In order to evaluate how well message identification works when packets are sent via the Internet, we used our testnet setup and deployed the code in the public Lightning testnet.

We compared the ground-truth LN message and our program's output using the generated logs, and calculated commonly used classification metrics for the `revoke_and_ack` message type: precision and recall. We recorded a precision of 1.0 and recall of 1.0. This means

<sup>4</sup> At the time of writing, approximately 2 months since joining the testnet, we are yet to receive any routing requests.

<sup>5</sup> <https://github.com/bitcoin-dev-project/sim-ln>

that the program returns neither false positives nor false negatives. These results are not unexpected and emphasise the exact problem brought by the highly deterministic nature of communication in the LN. We discuss the accuracy for higher payment rates in Section 4.3.

### 4.2.2 Practicability

Given the confidence that we can correctly identify encrypted LN messages, we sought to verify that the adversary can actually censor nodes in its network. We thus performed all of the following tests in our private regtest network.

In the first experiment, *S* tried to send 10k sat to *D*. As a result of the fees and timelock advertised by *A* and *B*, the most attractive path for payments from *S* to *D* was via *A*. After receiving the HTLC offer from *S*, *A* offered an HTLC to *D* by sending an `update_add_htlc` message immediately. The attack code thus correctly identified that the payment is not destined for *A* and does not drop any packets.

In the second experiment, *S* attempted to send 10k sat to *A* via their shared channel. We run the “base” attack on *A*’s interface as we know it is the recipient and can only otherwise determine the recipient as the HTLC is being settled (cf. Section 3.3). We evaluate identifying a node’s role in a payment path in an ensuing analysis. Once the program got to state 5 of Figure 3, the `revoke_and_ack` message was dropped. *S* retransmitted the packet as it is not acknowledged by *A* before closing the TCP connection. This left the channel in a temporarily inactive state and the payment in a pending state. After an exponential backoff period, *S* reestablished the connection and sent the `revoke_and_ack` message again. Note that *S* can still open a P2P connection as the code only drops `revoke_and_ack` messages. All subsequent `revoke_and_ack` received on *A*’s interface were dropped which triggers the connection close, reestablishment and retransmission loop. We advanced the blockchain manually by mining blocks until the time lock elapsed. At that point, the payment attempt failed permanently, and *S* forcefully closed the channel as well as the TCP connection. We reversed the direction of payment and observed similar behaviour on the CLN node with a few minor differences mainly with respect to retransmissions.

In summary, we confirmed that it is possible to execute the attack and block payments based on network-level observations. Furthermore, we verified that the adversary is able to selectively censor payments and thus leave third-party payments intact.

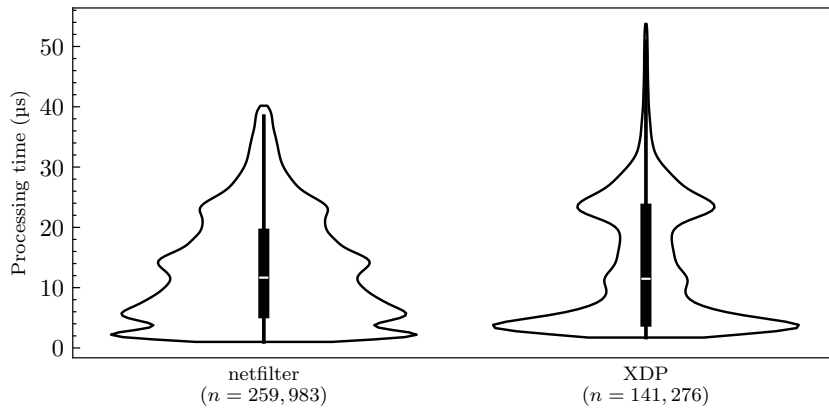
## 4.3 Performance

Subsequent to the feasibility analysis, we studied the implementations’ performance in regard to the induced delays, throughput and accuracy at different payment rates. The sole fact that the attack can be executed is not sufficient if such is not possible efficiently.

### 4.3.1 Latency

In the first of the three performance-related measurements, we examined the delay added to each TCP packet received on or directed to port 9735 by both the `netfilter` and XDP implementations. Figure 7 shows the time required to process TCP packets in microseconds by both implementations, i.e., the duration from the program first accessing a packet to a decision being made on the packet. The data was collected during the 24-hour time frame in which the measurements in Section 4.2.1 were performed and is depicted as a violin plot.

The results indicate that both implementations are quite efficient and issue a verdict on packets within the same median time of  $\approx 11\mu\text{s}$  per packet. The mean processing time is  $13\mu\text{s}$  per packet and  $14\mu\text{s}$  per packet for the `netfilter` and XDP programs respectively. That



■ **Figure 7** The packet processing times in microseconds for the `netfilter` and XDP programs based on packets received in the testnet over a period of 24h. The difference in the observed number of packets is due to the fact that XDP only receives ingress traffic.

equates to a mean throughput of  $\approx 76,923pps$  and  $\approx 71,4428pps$  respectively regardless of packet size. It may be surprising that the XDP program does not outperform the `netfilter` implementation despite XDP’s superiority to iptables in respect to speed [7,8]. However, these measurements were performed on the user-space code attached to either of the subsystems and do not reflect the underlying technologies’ throughput capabilities. In summary, we conclude that the delays induced by the additional filtering layer are negligible and do not hamper the feasibility of the attack. Such delays in the range of tens of microseconds are likely to go unnoticed by LN users or even routing nodes.

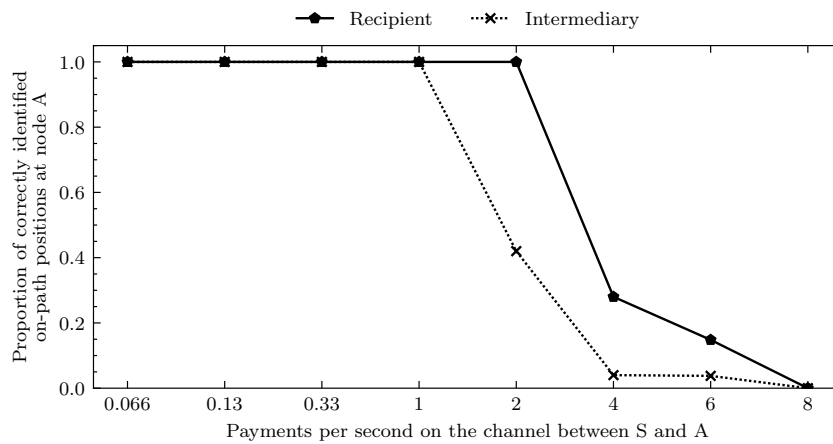
### 4.3.2 Throughput

We studied the maximum rate at which packets may be received by the `netfilter` program before they start being dropped due to congestion in the queue. The maximum queue length defaults to 1024 packets; all packets will be dropped as long as the target queue is full.

As per the previous measurements, the `netfilter` program achieves a mean throughput of  $\approx 76,923pps$ . Hence, in order to have 1024 queued packets, the program must receive packets at a rate roughly 1000 times higher than 76,923, i.e.,  $76,923 \cdot 10^3pps$ . Based on the traffic we observed in the testnet, we strongly believe that it is highly unlikely for a single Lightning node to generate and/or receive packets at speeds remotely close to that.

The largest AS (with respect to the number of nodes) in the mainnet is AS 14618 (Amazon.com) with 298 nodes as of 12 January 2024. Let us assume that, hypothetically, AS 14618 wants to execute the attack using a single instance of the program, i.e., all LN packets to/from the 298 nodes are processed sequentially by the same instance of the program. Further, we assume that the number of packets at each node is even.<sup>6</sup> This means that each node must pass approximately  $3 \cdot 10^5pps$  to the program in order to achieve a combined rate of  $76,923 \cdot 10^3pps$ . Similarly, we do not consider such rates to be feasible in the LN. While we have made simplifying assumptions, these results indicate that the censorship attack can be executed in a large-scale manner using `netfilter`.

<sup>6</sup> This is a reasonable assumption to make as the amount of traffic at central nodes and less central nodes probably balance each other out.



■ **Figure 8** The success rate of identifying the monitored node’s role in a payment, i.e., whether the node is the recipient or an intermediate routing hop.

### 4.3.3 Position identification

The concluding performance-related measurements studied the outcome of identifying a node’s position in a payment path for different transaction rates. While there are no studies on the network’s throughput, the channel-wise transaction rate is estimated to be rather low, e.g., 0.000019 payments per second based on reports by a central routing node [43]. To that end, we sent payments at different rates that we believe to be realistically achievable in the LN. The payments were issued in the private network due to the high volume.

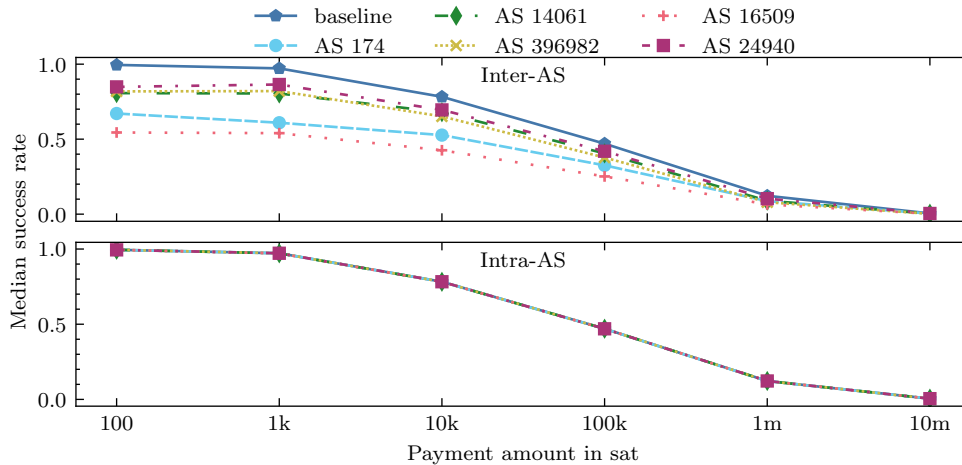
For each of the different transaction rates under study,  $S$  issued 50 payments to  $A$  and 50 payments to  $D$ . We recorded  $A$ ’s true position in each payment’s path as well as the `netfilter` program’s verdict on the its position. We omit the classification of the different messages as it remains possible even at higher transaction rates without significant effort.

The proportion of routing positions correctly identified using the methodology described in Section 3.3 is depicted in Figure 8 for a varying number of payments per second between nodes  $S$  and  $A$ . At a rate of up to 1 payment per second the program correctly identified  $A$ ’s position in a payment path in all cases. However, as the rate increases beyond 1 payment per second, the accuracy gradually declines and ultimately falls to zero at 8 payments per second. This is because of the shorter intervals between messages which make it harder to distinguish whether messages are related or not. It is worth noting that correctly uncovering a node’s position when it is the recipient is slightly more robust to higher payment rates. A transaction rate of 1 payment per second is indeed very low, however, we remark that it is still significantly higher than current estimates of LN’s throughput.

These results show that, as long as the network’s throughput does not increase drastically, the attack can be executed accurately.

## 4.4 Global impact

Naturally, we did not perform any measurements on the public network. Instead, and similar to multiple previous works [6, 26, 43], we simulated the attack using a snapshot of the channel graph obtained from a fully-synced LND node on 12 January 2024. We extended the LN simulator from [26] with some networking logic in order to map nodes to their corresponding



■ **Figure 9** The median success rate when each of the top five ASs forbid either all inter-AS payments or all intra-AS payments.

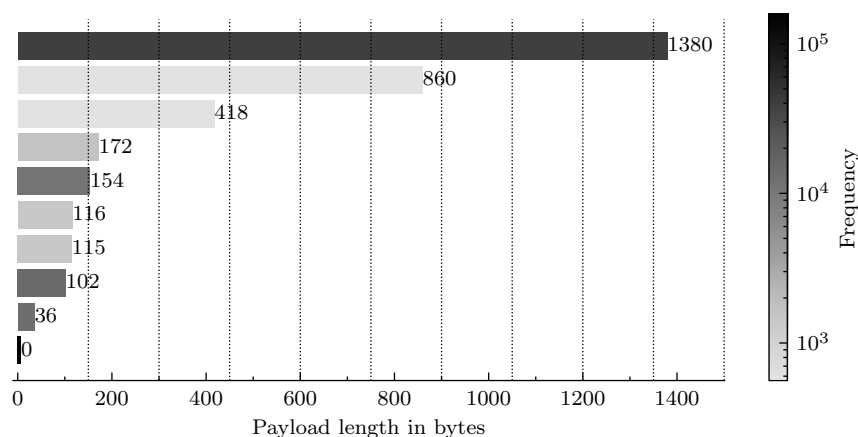
AS<sup>7</sup>, as well as the ability to simulate node failures. The code is publicly available on GitHub.<sup>8</sup> As actual payment volumes in LN are unknown, we simulated various payment volumes following the categorisation in [12] ranging from 100 sat to 10m sat. We simulated a set of 1,000 payments between random sender-receiver pairs for each of the selected amounts. In order to measure the impact of censorship by a malicious AS on the LN, we simulated sending the payments in two different adversarial scenarios: when an AS allows either only local payments, i.e., intra-AS payments, or only payments involving at least one other AS, i.e., inter-AS payments. As shown in Section 4.2.1, a network-level adversary is able to determine a node’s role in payment path, and we can thus simulate selective censorship. The ASs were ranked based on the number of channels and the top five were selected. We repeated each simulation scenario ten times with different seeds for the random number generator, i.e., for each set of 1,000 sender-receiver pairs and for each AS, the channel graph was reinitialised before simulating payment delivery for each of the selected amounts.

The median success rate, i.e., the ratio of successful payments and the total number of payments, for all conducted simulations is shown in Figure 9. Besides observing what is already known in regard to the inverse relation between the success rate and payment amount [6, 26], the results clearly suggest that most payments in the LN are made between different ASs. Bearing the low proportion of intra-AS channels in mind (cf. Figure 2), it is not surprising that a significant amount of payments are affected by inter-AS censorship. The impact of the attack varies depending on the choice of malicious AS, e.g., AS 16509 causes a decrease of up to 45% while AS 24940 results in a drop of “only” up to 18% in the success rate. In contrast, when an AS only blocks payments within their network, the difference in the success rates is minimal suggesting that the impact on the greater network is negligible.

These results indicate that an adversary can block payments within their area of jurisdiction without causing significant harm to the wider network. On the other hand, the effects of a malicious adversary blocking payments being routed via their network would

<sup>7</sup> We used the GeoLite2 data from MaxMind, available at <https://www.maxmind.com>.

<sup>8</sup> <https://github.com/tud-dud/lightning-censorship-simulator>



**Figure 10** The ten most frequent Lightning payload lengths transmitted to/from an LND node in the testnet over 24 hours. The  $18B$  packets LND nodes send are not included. The vertical lines illustrate the resulting number of packets if the payloads are padded and chunked to  $150B$ .

be very adverse for the network. The results of the other studied metrics such as fees and path lengths generally show little to no variation to the baseline simulation regardless of the applied dropping strategy. The charts have thus been omitted due to space constraints.

## 5 Countermeasures

In what follows, we discuss different measures the LN can implement in order to impede and/or mitigate network-level monitoring attacks. The authors of [43] propose a shift from a default port in LN to a pairwise-negotiated port in order to thwart port-based traffic filtering. Deviating from port 9735 is a stopgap which does not provide a suitable mitigation, but adds a layer of complexity to the attack that must be overcome. We argue that this alone is not adequate as, assuming the adversary is able to determine the new port, the attack can still be executed without any change. The new port can, for instance, be discovered using public crawl data, or by simply operating an LN node as each node stores the current topology locally. They also propose to “avoid adversarial ASs” by using third-party network services, e.g., Tor and VPNs, and implementing AS-aware routing. We argue that a VPN does not offer sufficient protection as it simply transfers the risk from one AS to another.

Briefly recapped, the core of the attack presented in this work exploits two side channels – payload size and timing information – to allow a network-level adversary to identify the different LN messages despite encryption. Arx et al. suggest hiding the lengths of the application data but do not provide specifics on a plausible padding strategy [43]. The reasoning behind employing a length-hiding scheme is that the network-level classification attacks rely on the TCP payload lengths to identify messages. It is, however, not clear which strategy is best suited for the LN.

### 5.1 Weighing the options

We recorded the lengths of all the TCP packets on port 9735 during the 24-hour time period in which the measurements in Section 4 were performed and depict the observations in Figure 10. As evident in Figure 10, the payload sizes of LN messages differ wildly. Consequently, finding a common length is not trivial. Simply padding all payloads to the

maximal length would result in a significant waste of bandwidth. Instead, we could chunk the data following a block-length padding strategy [14, 23], i.e., padding to the closest multiple of  $x$  bytes. As a result, a network-level adversary would only observe constant-length LN payloads. Nevertheless, as LN messages have a specified length, the adversary can still make use of the other side channel – the timing information – to classify the encrypted TCP payloads. All the attacker needs to do is map the messages sizes in Table 1 to multiples of  $x$ . Observing the direction of flow, number of packets and sequence still gives clear indications of the underlying messages in transit. For the sake of argumentation, let us assume that  $x$  is somewhat arbitrarily set to  $150B$ , i.e., all LN messages are sent in  $150B$  chunks. As visualised in Figure 10, all but the first message exchanged during the HTLC commitment phase would be identical on the network layer (cf. Table 1). Identifying the application messages is then no longer possible by simply inspecting the observed packet sizes. However, as we know both the type and order of messages involved in the process, we know how many packets correspond to each of the messages. An adversary must therefore additionally keep a count of packets which adds a minimal layer of complexity to the attack. The perhaps most obvious telltale sign is the `update_add_htlc` message ( $1450B$ ) that would be sent in ten packets followed by two packets for the `commitment_signed` ( $162B$ ) in the same direction. Regardless of this weakness, other message types not discussed in this work would also need to be taken into consideration in order to define a meaningful chunk size.

If we turn our attention to the timing information, we realise that it is even more delicate. For instance, we cannot simply reorder messages while conforming to the protocol specifications. Techniques such as adaptive padding [36] which inject dummy packets into the packet flow thus become relevant. This destroys timing fingerprints without any additional latency. However, adaptive padding on its own is not an adequate countermeasure for the LN as the other side channel – message size – remains unaddressed. For similar reasons, transmitting packets at a constant rate [11] is not sufficient on its own either. Currently, Tor implements a variation of adaptive padding as a defence against website fingerprinting (WF) attacks derived from the Website Traffic Fingerprinting Protection with Adaptive Defence (WTF-PAD) [18] mechanism. In summary, WTF-PAD sends dummy data such that an attacker cannot tell real data apart from fake data based on expected packet inter-arrival times. Furthermore, since all traffic in Tor is padded to  $514B$  cells, WTF-PAD impedes the effectiveness of WF attacks in Tor by obfuscating timing patterns.

## 5.2 Towards a solution

We examined whether the variant of padding that is implemented in Tor would provide sufficient protection in the LN against the attack at hand. We did so by configuring two LND nodes to connect to each other over Tor and opening a channel between them. The purpose of doing so is to utilise Tor’s implementation of WTF-PAD and not for Tor’s privacy properties. We issued payments in both directions, closed the channel and finally the TCP connection. Not only did all packets have the same packet length (as is expected when using Tor), but the flow of transmitted packets included packets that did not originate from the application. Consequently, we were not able to detect which packets belonged to which Lightning message by manually inspecting the capture. The rule-based state machine is therefore no longer capable of distinguishing application messages based on the network traces alone. In fact, we conjecture that this approach offers a high degree of protection for the LN against more sophisticated fingerprinting techniques by network-level adversaries as basically all size and timing features are destroyed.



Although we have established that the mechanisms implemented in Tor offer sufficient protection, the question of how much this protection costs remains unanswered. In order to get an approximation of the cost of using Tor, we captured all packets while executing the above operations in a thirty-minute time frame. In addition to the aforementioned deliberate activity, the time frame also includes periods in which only control messages were sent by the nodes, e.g., when the blockchain advances or health checks. Specifically, we concurrently captured the packets sent locally between the LND node and the Tor SOCKS5 proxy, as well as the packets sent between the Tor process and Tor network. The former provides data on the packets that actually come from the application while the latter provides data on what a network-level attacker would observe. The captures show a total of 14,824 bytes transmitted in 379 TCP packets to/from LND and 929,596 bytes in 3191 TCP packets to/from the Tor network. This equates to an increase of  $\approx 6170\%$  in bandwidth when using Tor. The captures also show a peak rate of 0.116 Mbit/s when using Tor, which clearly should not cause any problems for LN nodes while maintaining their current hardware configurations. However, we note that these are overestimations of the actual overhead to expect in the LN as they include traffic in Tor that is not actually relevant to mitigating network-level message identification in the LN, e.g., circuit management. We therefore do not consider the universal usage of Tor in the LN to be the solution; the overhead of a standalone implementation of WTF-PAD in the LN is expected to be much lower. Besides, Tor nodes are susceptible to other potential threats [20, 27, 39] and using Tor implies higher latency in order to provide features that may not be required by all nodes in the LN.

An effective mitigation strategy for the LN must omit both the timing and size information. Obfuscating either properties is further complicated by the fact that crucial LN operations, e.g., channel opening or HTLC commitment, must follow an order defined in the protocol. This means that message flows between two Lightning nodes often follow deterministic patterns. In view of the preceding discussion, we recommend that the LN adopts a form of adaptive padding similar to Tor as a defence against network-layer monitoring attacks. That is, not only must we conceal all packet sizes on the network layer, we must also obfuscate the timing patterns in the P2P communication. Our assessments of the attack's feasibility over Tor demonstrate that fixed-length packets in conjunction with cover traffic effectively hamper the attack. While this solution will necessarily introduce a degree of overhead, the LN may be facing a technical version of pick your poison.

## 6 Related Work

### 6.1 Censorship

Internet censorship has been the subject of multiple works, e.g., [4, 45, 46], due to some of the extensive censorship currently imposed in various parts of the world. It is thus a highly relevant topic. P2P networks are generally considered to be more resistant to censorship than classic server-client networks as a result of their fundamental architectural differences. Nonetheless, there have been reports on the feasibility of imposing censorship in blockchain-based P2P networks such as Bitcoin [22] and Ethereum [44] by, for instance, exploiting application-level protocol designs. A prominent example of a state-imposed censorship in the realm of digital payment networks is the complete trading and mining ban in China. In a recent work by Sridhar et al., the authors present a censorship attack in the InterPlanetary File System (IPFS) [38] – a popular P2P content delivery network.

## 6.2 Network-level Attacks in the Lightning network

To the best of our knowledge, the LN's network layer has not received significant attention so far. In one of the few works, Casas et al. [9] analysed the P2P network and found that a significant number of nodes connects to the LN through Tor. An analysis conducted in [47] established a degree of geographic clustering among the nodes. The authors of [43] study attacks on the LN's network layer and show that it is possible to decipher encrypted LN messages via traffic analysis. Besides pursuing a different goal, our work not only confirms their findings but also refines the information an adversary can gain from traffic analysis. This additional information is what enables an adversary to impose selective censorship based on the payment's source and/or destination. Furthermore, our presented attack is based on real-time traffic monitoring and execution in contrast to [43]. There has also been research on AS-level side channel attacks on privacy and routing in the broad spectrum of cryptocurrency networks [3, 32, 37, 42] and anonymisation networks such as Tor [27, 39].

## 6.3 Lightning network Topology and Simulations

Numerous works have studied the structural properties of the Lightning channel graph and demonstrated that it is highly centralised [6, 34, 41] at the application level, e.g., a small number of nodes function as essential routing nodes due to their high centrality in the graph. As a result, it is susceptible to a variety of attacks on privacy and security [19, 31, 40]. The analysis of the channel graph's network level in [43] revealed that it is equally centralised and vulnerable to attacks on payment privacy. Our topological analysis of the channel graph complements existing ones and provides new insights on its network-level structure, e.g., most channels in the network are between distinct pairs of ASs.

A broad range of research on LN takes a simulation-based approach, e.g., [6, 19, 31], to analyse their studies' significance for the public mainnet. Simulations are often necessary in order to not interact with third-party nodes in the public network. Due to the availability of multiple open-source LN simulators, we did not develop a new simulator but instead extended an existing one [26] with the relevant functionality for this work.

## 7 Conclusion and Future Work

We studied potential censorship attacks in the Lightning network founded on monitoring network-level traffic. Furthermore, we demonstrated that it is feasible to determine a node's position in a payment path based on the observed traffic. In doing so, our work highlights the threat powerful adversaries such as autonomous systems pose to the Lightning network which is further heightened by the network-level centralisation. Based on our analysis of potential countermeasures, we conclude that an effective mitigation strategy in the LN inevitably implies some bandwidth overhead.

The attack presented in this work exploits two side channels at the network layer – payload size and timing patterns. We think that studying effective and efficient mitigation strategies is an interesting and relevant research question. Complementary to mitigation strategies, developing mechanisms to detect censorship is a similarly relevant question for future work. Additionally, and like in multiple other previous works, estimates used in this work with respect to the network's throughput relied on the occasional reports provided by node operators. Acknowledging that measuring throughput in a public P2P network is not straightforward, we believe that future research on Lightning would benefit from well-founded assessments of the network's throughput.

---

**References**


---

- 1 Bolt: Basis of lightning technology (lightning network specifications). URL: <https://github.com/lightning/bolts>.
- 2 XDP - IO Visor Project. <https://www.iovisor.org/technology/xdp>, 2016. [Accessed 01/02/2024].
- 3 Maria Apostolaki, Cedric Maire, and Laurent Vanbever. Perimeter: A network-layer attack on the anonymity of cryptocurrencies. In *Financial Cryptography and Data Security - 25th International Conference, FC 2021, Virtual Event, March 1-5, 2021, Revised Selected Papers, Part I*, volume 12674 of *Lecture Notes in Computer Science*, pages 147–166. Springer, 2021. doi:10.1007/978-3-662-64322-8\_7.
- 4 Simurgh Aryan, Homa Aryan, and J. Alex Halderman. Internet censorship in iran: A first look. In *3rd USENIX Workshop on Free and Open Communications on the Internet (FOCI 13)*, Washington, D.C., 2013. USENIX Association. URL: <https://www.usenix.org/conference/foci13/workshop-program/presentation/aryan>.
- 5 Michael D. Bailey, David Dittrich, Erin Kenneally, and Douglas Maughan. The menlo report. *IEEE Secur. Priv.*, 10(2):71–75, 2012. doi:10.1109/MSP.2012.52.
- 6 Ferenc Béres, István András Seres, and András A. Benczúr. A cryptoeconomic traffic analysis of bitcoins lightning network. *CoRR*, 2019. arXiv:1911.09432.
- 7 Gilberto Bertin. Xdp in practice: integrating xdp into our ddos mitigation pipeline. In *Technical Conference on Linux Networking, Netdev*, volume 2, pages 1–5. The NetDev Society, 2017.
- 8 Kevin Bock, George Hughey, Xiao Qiang, and Dave Levin. Geneva: Evolving censorship evasion strategies. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2199–2214. Association for Computing Machinery, 2019. doi:10.1145/3319535.3363189.
- 9 Pedro Casas, Matteo Romiti, Peter Holzer, Sami Ben Mariem, Benoit Donnet, and Bernhard Haslhofer. Where is the light(ning) in the taproot dawn? unveiling the bitcoin lightning (IP) network. In *10th IEEE International Conference on Cloud Networking, CloudNet 2021, Cookeville, TN, USA, November 8-10, 2021*, pages 87–90. IEEE, 2021. doi:10.1109/CLOUDNET53349.2021.9657121.
- 10 George Danezis and Ian Goldberg. Sphinx: A compact and provably secure mix format. In *30th IEEE Symposium on Security and Privacy (SP 2009), 17-20 May 2009, Oakland, California, USA*, pages 269–282. IEEE Computer Society, 2009. doi:10.1109/SP.2009.15.
- 11 Kevin P. Dyer, Scott E. Coull, Thomas Ristenpart, and Thomas Shrimpton. Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail. In *2012 IEEE Symposium on Security and Privacy*, pages 332–346, 2012. doi:10.1109/SP.2012.28.
- 12 Oguzhan Ersoy, Stefanie Roos, and Zekeriya Erkin. How to profit from payments channels. In *Financial Cryptography and Data Security - 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10-14, 2020 Revised Selected Papers*, volume 12059 of *Lecture Notes in Computer Science*, pages 284–303. Springer, 2020. doi:10.1007/978-3-030-51280-4\_16.
- 13 Arturo Filastò and Jacob Appelbaum. OONI: Open observatory of network interference. In *2nd USENIX Workshop on Free and Open Communications on the Internet (FOCI 12)*. USENIX Association, 2012. URL: <https://www.usenix.org/system/files/conference/foci12/foci12-final12.pdf>.
- 14 Daniel Kahn Gillmor. Empirical dns padding policy. <https://dns.cmrg.net/ndss2017-dprive-empirical-DNS-traffic-size.pdf>, 2017. [Accessed 03/04/2024].
- 15 Nguyen Phong Hoang, Arian Akhavan Niaki, Jakub Dalek, Jeffrey Knockel, Pellaon Lin, Bill Marczak, Masashi Crete-Nishihata, Phillipa Gill, and Michalis Polychronakis. How great is the great firewall? measuring china’s DNS censorship. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 3381–3398. USENIX Association, 2021. URL: <https://www.usenix.org/conference/usenixsecurity21/presentation/hoang>.

- 16 Toke Høiland-Jørgensen, Jesper Dangaard Brouer, Daniel Borkmann, John Fastabend, Tom Herbert, David Ahern, and David Miller. The express data path: fast programmable packet processing in the operating system kernel. In *Proceedings of the 14th International Conference on emerging Networking EXperiments and Technologies, CoNEXT 2018, Heraklion, Greece, December 04-07, 2018*, pages 54–66. ACM, 2018. doi:10.1145/3281411.3281443.
- 17 OpenNet Initiative. Turkmenistan, December 2010. URL: <https://opennet.net/research/profiles/turkmenistan>.
- 18 Marc Juarez, Mohsen Imani, Mike Perry, Claudia Díaz, and Matthew Wright. Toward an efficient website fingerprinting defense. In *Computer Security - ESORICS 2016 - 21st European Symposium on Research in Computer Security, Heraklion, Greece, September 26-30, 2016, Proceedings, Part I*, volume 9878 of *Lecture Notes in Computer Science*, pages 27–46. Springer, 2016. doi:10.1007/978-3-319-45744-4\_2.
- 19 George Kappos, Haaron Yousaf, Ania M. Piotrowska, Sanket Kanjalkar, Sergi Delgado-Segura, Andrew Miller, and Sarah Meiklejohn. An empirical analysis of privacy in the lightning network. In *Financial Cryptography and Data Security - 25th International Conference, FC 2021, Virtual Event, March 1-5, 2021, Revised Selected Papers, Part I*, volume 12674 of *Lecture Notes in Computer Science*, pages 167–186. Springer, 2021. doi:10.1007/978-3-662-64322-8\_8.
- 20 Albert Kwon, Mashaal AlSabah, David Lazar, Marc Dacier, and Srinivas Devadas. Circuit fingerprinting attacks: Passive deanonymization of tor hidden services. In *24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015*, pages 287–302. USENIX Association, 2015. URL: <https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/kwon>.
- 21 Jian-Hong Lin, Kevin Primicerio, Tiziano Squartini, Christian Decker, and Claudio J. Tessone. Lightning network: a second path towards centralisation of the bitcoin economy. *CoRR*, 2020. arXiv:2002.02819.
- 22 Angelique Faye Loe and Elizabeth Anne Quaglia. You shall not join: A measurement study of cryptocurrency peer-to-peer bootstrapping techniques. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 2231–2247. ACM, 2019. doi:10.1145/3319535.3345649.
- 23 A. Mayrhofer. Padding policies for extension mechanisms for dns (edns(0)). <https://datatracker.ietf.org/doc/html/rfc8467>, 2018. [Accessed 03/04/2024].
- 24 Ayelet Mizrahi and Aviv Zohar. Congestion attacks in payment channel networks. In *Financial Cryptography and Data Security - 25th International Conference, FC 2021, Virtual Event, March 1-5, 2021, Revised Selected Papers, Part II*, volume 12675 of *Lecture Notes in Computer Science*, pages 170–188. Springer, 2021. doi:10.1007/978-3-662-64331-0\_9.
- 25 Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. URL: <https://nakamotoinstitute.org/bitcoin/>.
- 26 Charmaine Ndolo and Florian Tschorsch. On the (not so) surprising impact of multi-path payments on performance and privacy in the lightning network. In *Computer Security. ESORICS 2023 International Workshops - CyberICS, DPM, CBT, and SECPRE, The Hague, The Netherlands, September 25-29, 2023, Revised Selected Papers, Part I*, volume 14398 of *Lecture Notes in Computer Science*, pages 411–427. Springer, 2023. doi:10.1007/978-3-031-54204-6\_25.
- 27 Rishab Nithyanand, Oleksii Starov, Phillipa Gill, Adva Zair, and Michael Schapira. Measuring and mitigating as-level adversaries against tor. In *23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016*. The Internet Society, 2016. URL: <http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2017/09/measuring-mitigating-as-level-adversaries-against-tor.pdf>.
- 28 Sadia Nourin, Van Tran, Xi Jiang, Kevin Bock, Nick Feamster, Nguyen Phong Hoang, and Dave Levin. Measuring and evading turkmenistan’s internet censorship: A case study in large-scale measurements of a low-penetration country. In *Proceedings of the ACM Web Conference 2023, WWW ’23*, pages 1969–1979. Association for Computing Machinery, 2023. doi:10.1145/3543507.3583189.

- 29 Trevor Perrin. The noise protocol framework. <https://noiseprotocol.org/noise.pdf>, 2018. [Accessed 05/04/2024].
- 30 Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments, January 2016. URL: <https://lightning.network/lightning-network-paper.pdf>.
- 31 Elias Rohrer and Florian Tschorsch. Counting down thunder: Timing attacks on privacy in payment channel networks. In *AFT '20: 2nd ACM Conference on Advances in Financial Technologies, New York, NY, USA, October 21-23, 2020*, pages 214–227. ACM, 2020. doi:10.1145/3419614.3423262.
- 32 Muhammad Saad and David Mohaisen. Three birds with one stone: Efficient partitioning attacks on interdependent cryptocurrency networks. In *44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, May 21-25, 2023*, pages 111–125. IEEE, 2023. doi:10.1109/SP46215.2023.10179456.
- 33 Dominik Scholz, Daniel Raumer, Paul Emmerich, Alexander Kurtz, Krzysztof Lesiak, and Georg Carle. Performance implications of packet filtering with linux ebpf. In *30th International Teletraffic Congress, ITC 2018, Vienna, Austria, September 3-7, 2018 - Volume 1*, pages 209–217. IEEE, 2018. doi:10.1109/ITC30.2018.00039.
- 34 István András Seres, László Gulyás, Dániel A. Nagy, and Péter Burcsi. Topological analysis of bitcoin’s lightning network. In *Mathematical Research for Blockchain Economy, 1st International Conference, MARBLE 2019, Santorini, Greece, May 6-9, 2019*, pages 1–12. Springer, 2019. doi:10.1007/978-3-030-37110-4\_1.
- 35 RFE/RL’s Turkmen Service. Internet in turkmenistan, already the world’s slowest, faces further restrictions, January 2022. URL: <https://www.rferl.org/a/turkmenistan-internet-slowest-restrictions/31652467.html>.
- 36 Vitaly Shmatikov and Ming-Hsiu Wang. Timing analysis in low-latency mix networks: Attacks and defenses. In *Computer Security - ESORICS 2006, 11th European Symposium on Research in Computer Security, Hamburg, Germany, September 18-20, 2006, Proceedings*, volume 4189 of *Lecture Notes in Computer Science*, pages 18–33. Springer, 2006. doi:10.1007/11863908\_2.
- 37 Paulo Silva. Impact of geo-distribution and mining pools on blockchains: A study of ethereum - practical experience report and ongoing phd work. In *50th Annual IEEE-IFIP International Conference on Dependable Systems and Networks, DSN 2020, Valencia, Spain, June 29 - July 2, 2020 - Supplemental Volume*, pages 73–74. IEEE, 2020. doi:10.1109/DSN-S50200.2020.00039.
- 38 Srivatsan Sridhar, Onur Ascigil, Navin V. Keizer, François Genon, Sébastien Pierre, Yiannis Psaras, Etienne Rivière, and Michal Król. Content censorship in the interplanetary file system. *CoRR*, 2023. doi:10.48550/arXiv.2307.12212.
- 39 Yixin Sun, Anne Edmundson, Laurent Vanbever, Oscar Li, Jennifer Rexford, Mung Chiang, and Prateek Mittal. RAPTOR: routing attacks on privacy in tor. In *24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015*, pages 271–286. USENIX Association, 2015. URL: <https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/sun>.
- 40 Sergei Tikhomirov, Pedro Moreno-Sanchez, and Matteo Maffei. A quantitative analysis of security, anonymity and scalability for the lightning network. In *IEEE European Symposium on Security and Privacy Workshops, EuroS&P Workshops 2020, Genoa, Italy, September 7-11, 2020*, pages 387–396. IEEE, 2020. doi:10.1109/EUROSPW51379.2020.00059.
- 41 Saar Tochner, Stefan Schmid, and Aviv Zohar. Hijacking routes in payment channel networks: A predictability tradeoff. *CoRR*, 2019. arXiv:1909.06890.
- 42 Florian Tramèr, Dan Boneh, and Kenny Paterson. Remote side-channel attacks on anonymous transactions. In *29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020*, pages 2739–2756. USENIX Association, 2020. URL: <https://www.usenix.org/conference/usenixsecurity20/presentation/tramer>.

- 43 Theo von Arx, Muoi Tran, and Laurent Vanbever. Revelio: A network-level privacy attack in the lightning network. In *8th IEEE European Symposium on Security and Privacy, EuroS&P 2023, Delft, Netherlands, July 3-7, 2023*, pages 942–957. IEEE, 2023. doi:10.1109/EUROSP57164.2023.00060.
- 44 Anton Wahrstätter, Jens Ernstberger, Aviv Yaish, Liyi Zhou, Kaihua Qin, Taro Tsuchiya, Sebastian Steinhorst, Davor Svetinovic, Nicolas Christin, Mikolaj Barczentewicz, and Arthur Gervais. Blockchain censorship. In *Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, May 13-17, 2024*, pages 1632–1643. ACM, 2024. doi:10.1145/3589334.3645431.
- 45 Xueyang Xu, Zhuoqing Morley Mao, and J. Alex Halderman. Internet censorship in china: Where does the filtering occur? In *Passive and Active Measurement - 12th International Conference, PAM 2011, Atlanta, GA, USA, March 20-22, 2011. Proceedings*, volume 6579 of *Lecture Notes in Computer Science*, pages 133–142. Springer, 2011. doi:10.1007/978-3-642-19260-9\_14.
- 46 Tarun Kumar Yadav, Akshat Sinha, Devashish Gosain, Piyush Kumar Sharma, and Sambuddho Chakravarty. Where the light gets in: Analyzing web censorship mechanisms in india. In *Proceedings of the Internet Measurement Conference 2018, IMC '18*, pages 252–264. Association for Computing Machinery, 2018. doi:10.1145/3278532.3278555.
- 47 Philipp Zabka, Klaus-Tycho Förster, Stefan Schmid, and Christian Decker. Node classification and geographical analysis of the lightning cryptocurrency network. In *ICDCN '21: International Conference on Distributed Computing and Networking, Virtual Event, Nara, Japan, January 5-8, 2021*, pages 126–135. ACM, 2021. doi:10.1145/3427796.3427837.



# Musketeer: Incentive-Compatible Rebalancing for Payment Channel Networks

Zeta Avarikioti   

TU Wien, Vienna, Austria  
Common Prefix, Vienna, Austria

Stefan Schmid  

TU Berlin, Germany  
Fraunhofer SIT, Berlin, Germany  
Weizenbaum Institute, Berlin, Germany

Samarth Tiwari  

Centrum Wiskunde & Informatica, Amsterdam, The Netherlands

---

## Abstract

In this work, we revisit the severely limited throughput problem of cryptocurrencies and propose a novel rebalancing approach for Payment Channel Networks (PCNs). PCNs are a popular solution for increasing the blockchain throughput, however, their benefit depends on the overall users' liquidity. Rebalancing mechanisms are the state-of-the-art approach to maintaining high liquidity in PCNs. However, existing opt-in rebalancing mechanisms exclude users that may assist in rebalancing for small service fees, leading to suboptimal solutions and under-utilization of the PCNs' bounded liquidity.

We introduce the first rebalancing approach for PCNs that includes *all users*, following a “*all for one and one for all*” design philosophy that yields optimal throughput. The proposed approach introduces a double-auction rebalancing problem, which we term MUSKETEER, where users can participate as buyers (paying fees to rebalance) or sellers (charging fees to route transactions). The desired properties tailored to the unique characteristics of PCNs are formally defined, including the novel game-theoretic property of *cyclic budget balance* that is a stronger variation of strong budget balance.

Basic results derived from auction theory, including an impossibility and multiple mechanisms that either achieve all desiderata under a relaxed model or sacrifice one of the properties, are presented. We also propose a novel mechanism that leverages time delays as an additional cost to users. This mechanism is provably truthful, cyclic budget balanced, individually rational and economic efficient but only with respect to liquidity.

**2012 ACM Subject Classification** Computing methodologies → Distributed algorithms; Theory of computation → Distributed algorithms; Theory of computation → Algorithmic mechanism design

**Keywords and phrases** Blockchains, Payment Channel Networks, Rebalancing, Game Theory

**Digital Object Identifier** 10.4230/LIPIcs.AFT.2024.13

**Funding** This work was partially funded by the German Research Foundation (DFG) Schwerpunktprogramm (SPP 2378, ReNO) 2023-2027, by the Austrian Science Fund (FWF) through the SFB SpyCode project F8512-N, the project CoRaF (grant agreement ESP 68-N), and by the WWTF through the project 10.47379/ICT22045.

## 1 Introduction

### 1.1 Motivation

Bitcoin and other cryptocurrencies are significantly transforming the financial landscape [35, 50]. However, a well-known issue of the celebrated Nakamoto consensus introduced with Bitcoin, is that it inherently prohibits high transaction throughput which in turn hinders the



© Zeta Avarikioti, Stefan Schmid, and Samarth Tiwari;  
licensed under Creative Commons License CC-BY 4.0

6th Conference on Advances in Financial Technologies (AFT 2024).

Editors: Rainer Böhme and Lucianna Kiffer; Article No. 13; pp. 13:1–13:22

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

widespread adoption of blockchain technologies [18]. For example, Bitcoin can process at most 7 transactions per second [18], while Visa processes tens of thousands of transactions per second. Furthermore, blockchains are evidently environments for-profit, therefore user-incentive design is critical. Although several works have studied blockchain-related topics under the lens of game theory, e.g., [13, 21, 16, 27, 14], there is still much to be explored, particularly concerning scaling protocols. *In this work, we model and investigate incentive-compatible mechanisms that can enhance the limited transaction throughput of blockchains like Bitcoin.*

Specifically, we focus on one of the most prominent and well-studied scalability solutions for blockchains, called *payment channels* [38]. With payment channels, users can transact off-chain at far lower costs and faster speeds. The core idea is that any two users can lock their coins in a “joint account” on-chain, namely the payment channel. Thereby, the channel parties may perform arbitrarily many off-chain transactions with each other by signing messages with the new distribution of coins in their joint account. To close the payment channel, the parties can publish on-chain the last update on the distribution of their coins. Naturally, each channel is limited by the coins locked by each party (liquidity), dictating the maximum amount that can be sent between them. For example, in a channel with Alice and Bob currently holding 3 and 5 coins respectively, Alice can send at most 3 coins to Bob, and Bob can send at most 5 coins to Alice. In short, the coins can be moved on the channel from Alice to Bob or vice versa, much like moving balls from one side of an abacus to the other.

Multiple payment channels operating on the same underlying blockchain, comprise a *payment channel network (PCN)*. PCNs allow users, who have at least one payment channel open, to route transactions through the network to other users with whom they do not share a direct payment channel. To successfully route a transaction, a path of channels with sufficient liquidity for all senders must exist. For example, if Alice wants to send 3 coins to Carol through Bob, Alice must have 3 coins available in her channel with Bob, and Bob must have 3 coins available in his channel with Carol. The intermediaries (e.g., Bob) that offer to use their channel liquidity to route another user’s transaction typically ask for a routing service fee. If a channel in the selected path is depleted (i.e., has low liquidity) in the desired direction, all the transfers in the path will be reverted and the transaction will fail. *The liquidity of individual payment channels is, therefore, a crucial factor in the effectiveness of PCNs as a scaling solution.* It determines the ability to route transactions and impacts the overall efficacy of PCNs in enhancing the transaction throughput.

To maintain high liquidity in PCNs, parties have two options: either lock a significant amount of coins initially or use an on-chain transaction to top up their channels. However, both options have their drawbacks. Locking a substantial amount of coins incurs an opportunity cost as these coins cannot be used for other on-chain operations. On the other hand, using on-chain transactions to top up channels hinders the scaling capabilities of the underlying blockchain.

*Rebalancing mechanisms* are an attractive alternative solution to improve liquidity within PCNs [26, 10, 1]. These mechanisms aim to identify cycles of depleted edges (channels) and route transactions across them in a way that ensures each node in the network has an equal amount of coins at the end of the process. By leveraging cycles within the PCN, parties with depleted channels can rebalance their channels by utilizing two of their channels – one as a source to send coins and another as a destination to receive coins.

However, the deployed local rebalancing algorithms [1] may be practically insufficient for two main reasons. Firstly, they only involve parties interested in rebalancing, thereby excluding channels that may route transactions for low or no routing fees; after all, intermediaries



are indifferent to whether the routed payment concerns a payment (path) or rebalancing (cycle). Secondly, local searching algorithms may miss optimization opportunities leading to poor outcomes.

To address the latter limitation, Revive [26] proposed globally coordinated channel rebalancing, therefore, achieving optimal outcomes. Hide & Seek [10] recently improved on Revive by enabling global rebalancing in a decentralized and privacy-preserving manner. However, in both algorithms, the rebalancing subgraph only includes the parties that wish to rebalance while the vast majority of channels of the PCN that may route transactions for low or no fees are neglected. *Thus, even with globally coordinated rebalancing, the limited rebalancing subgraph still impacts the optimality of the overall solution, and subsequently the PCN’s scaling capability, i.e., how many transactions can succeed off-chain given a bounded overall liquidity.*

## 1.2 Our Contribution

We propose a novel approach to rebalancing that *involves all PCN users* in order to maximize the liquidity utilization and subsequently the transaction throughput. Our approach allows all users to submit their liquidity and bid for every one of their channels. The liquidity in this setting captures the number of coins they are willing to use for routing/rebalancing while the bid encapsulates how much they are willing to pay per coin for rebalancing the specific channel. So positive bids express the desire of buyers to rebalance, whereas negative (and zero) bids the desire of sellers to sell their routing service. Now, modeling this problem reveals a major challenge: *how can we design an incentive-compatible rebalancing mechanism for both buyers and sellers?*

To the best of our knowledge, we are the first to examine user incentives in the context of rebalancing mechanisms for PCNs. Our goal is twofold: First, to formally model the problem, capturing the unique characteristics present in PCNs; second, to discover satisfactory solutions, exploring different trade-offs. To achieve our objectives, we extend Hide & Seek [10] to accommodate both buyers and sellers of rebalancing liquidity. This approach leads to a double-auction problem with several challenges stemming either from traditional auction theory or from the individual needs of PCNs. In modeling our problem, we pinpoint *channel depletion* as a distinct feature, setting it apart from other network mechanism designs like routing games [22]. Channel depletion signifies that transactions can *permanently* lower an edge’s capacity (here, liquidity) until counteracted by an opposite flow. Unlike railway networks where trains need tracks only temporarily, flows in our model can *compensate* for each other. Thus, existing results do not directly apply.

To determine the desiderata of our mechanism, we revisit conventional requirements from auction theory: (1) *economic efficiency*, i.e., maximizing the social welfare which captures that channels are prioritized for rebalancing based on their bids, (2) *truthfulness*, meaning users submit their true value, and (3) *individual rationality*, i.e., non-negative utility for rebalancing participants. However, our problem encounters an idiosyncrasy rooted in the payment channel primitive itself, affecting the budget-balancedness of the mechanism, i.e., the mechanism does not incur a deficit (nor a surplus). Specifically, coins cannot be burned in a payment channel because intuitively channel updates must always benefit one party; if there exists a coin distribution where both parties in the channel can benefit from changing, then there is no way to enforce it. For instance, we cannot enforce a distribution of 3 coins to Alice and Bob each and 2 coins burned, because the parties will cooperatively update their channel to hold 4 coins each. This implies that the mechanism cannot have either a surplus or a deficit, rendering (weakly) budget-balanced mechanisms infeasible. What’s more,

## 13:4 Incentive-Compatible Rebalancing for PCNs

rebalancing itself occurs via individual cycles in the PCN. As a result, our setting demands a stronger notion of budget balance, which we term (4) *cyclic budget balance*, i.e., each cycle must be strongly budget balanced independently.

Unfortunately, the above four desired properties cannot be simultaneously achieved by any mechanism. We prove this by applying the classic Myerson-Satterthwaite impossibility result for double auctions [33]. We further emphasize the significance of the cyclic budget balance property in shaping potential solutions: The output of a rebalancing mechanism consists of a set of rebalancing circulations, which are global solutions where user preferences in one segment of the graph can impact the rebalancing cycles in distant segments of the graph. While in VCG-type mechanisms users are compensated for the global effects of their channels, the constraint of cyclic budget balance prevents this approach.

To provide satisfactory solutions, we apply standard techniques such as the renowned VCG mechanism and first-price auctions to the problem at hand. In particular, we showcase a mechanism that satisfies all the desired properties but is only applicable when all users are aware of the potential maximum and minimum fees they might pay or earn for their participation. Subsequently, we present a VCG-type mechanism that also satisfies all the desiderata exclusively for buyers, under the assumption that sellers are not treated as strategic agents. We then provide a mechanism that also considers sellers but, similarly to first-price auctions, sacrifices truthfulness. Finally, we propose a novel mechanism that introduces *time delays* as a natural characteristic of this problem, with the aim of incentivizing users to actively and truthfully participate in the rebalancing process while optimizing the outcome. The inclusion of time delays allows us to navigate around the impossibility and maintain our objective of maximizing rebalanced liquidity, in exchange for losing economic efficiency in terms of time delays and liquidity combined.

## 2 Preliminaries and Model

In this section, we first provide the necessary background on the rebalancing of payment channel networks, which we subsequently use to introduce our setting and problem definition, termed MUSKETEER. We further present an overview of MUSKETEER. For the rest of the paper, we use the terms users and players interchangeably.

### 2.1 Rebalancing PCNs

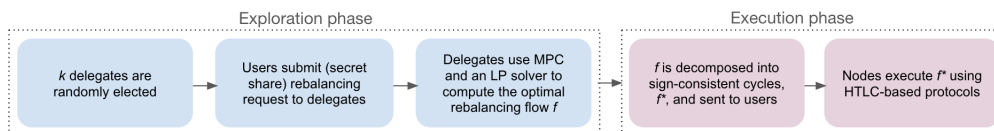
Rebalancing mechanisms are currently the only approach that allows users to restore their channel balances off-chain. In a nutshell, rebalancing mechanisms search the payment network for depleted channels that users wish to top-up off-chain until they identify a cycle of channels with enough liquidity. For instance, suppose Alice has one depleted channel with Bob, which she wants to top-up for 3 coins, and another channel with Carol where she has plenty of coins. Now, if Bob and Carol share a channel with at least 3 coins available for Carol, Alice can send 3 coins to Carol in their channel, Carol 3 coins to Bob, and Bob 3 coins to Alice. This way all users end up with the same total amount of coins. We stress that coins locked in a channel cannot be transferred to any other channel, much like the balls in different rows of an abacus.

Rebalancing mechanisms fall into two categories: local and global. Local rebalancing, currently deployed on the Lightning Network [38], has each party searching individually the network for other channels that want rebalancing; if a cycle is identified then the party can rebalance its channel. This approach may not find the optimal solution for rebalancing and it is very inefficient. Global rebalancing, introduced with Revive [26] and subsequently

optimized with Hide & Seek [10], finds the globally optimal solution for the users that directly and personally benefit from rebalancing their channels by leveraging coordination. Our solution extends this approach to further include users who are indifferent to rebalancing or may be willing to participate for a very small service routing fee. Considering routing fees are orders of magnitude smaller than the typical fee paid to the blockchain to top-up the channel balance, it is cost-effective for users to pay intermediaries to facilitate their rebalancing, similarly to transaction routing in PCNs – instead of paths, they route in cycles. We detail below the Hide & Seek mechanism that underpins our solutions.

## Hide & Seek [10]

The protocol proceeds in two phases: the exploration phase which identifies rebalancing cycles, and the execution phase which ensures their atomic execution in a secure and incentive-compatible fashion. The exploration phase begins with the random selection of  $k$  delegates among the users, e.g., using cryptographic sortition [23]. Then, the users submit their rebalancing requests, i.e., how many coins they wish to rebalance, to the delegates using secret sharing. Thereafter, the delegates use multi-party computation to calculate the optimal rebalancing flow on the network. To preserve users' privacy, each user receives only their specific flow. The optimization problem is modeled as a linear program that maximizes the rebalancing flow. The execution phase initiates by decomposing this flow into simple sign-consistent cycles, meaning that each channel only shares cycles with flow in that same direction. As a result, the channel owners are incentivized to execute all channels, and not select a subset thereof. The execution of the cycles occurs atomically, i.e., either all transactions succeed or all fail, using HTLC-based solutions [38, 47, 49]. Figure 1 illustrates the Hide & Seek protocol flow.



■ **Figure 1** The protocol flow of Hide & Seek.

## 2.2 Musketeer Overview

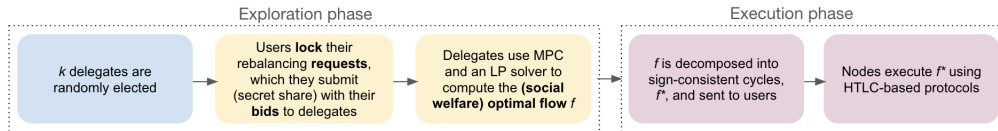
In MUSKETEER, each PCN channel may participate in the rebalancing process either as a depleted or as an indifferent edge. Depleted edges are channels owned by players that wish to rebalance their channels (i.e., act as *buyers*), while indifferent edges are owned by players that sell their routing services (i.e., act as *sellers*). We model this problem as a double auction: each player submits their (non-negative or non-positive) bid for each channel they are part of, which indicates the maximum or minimum amount they are willing to pay or receive per unit coin for rebalancing or routing through that channel, respectively.

Additionally, for each channel, the users submit their liquidity, i.e., the number of coins available to the rebalancing mechanism. These coins may be available because buyers want to rebalance their channels or because sellers may want to earn fees for their service. With this knowledge, we extract the rebalancing subgraph, which is a directed graph with capacities capturing each channel's liquidity.

## 13:6 Incentive-Compatible Rebalancing for PCNs

The resulting combinatorial problem can be modeled as a max-flow problem, where the goal is to maximize the total number of coins (flow) weighted by the buyer’s bids. In other words, we calculate the flow that maximizes social welfare, respecting the channel capacities. We then decompose the flow in simple independent cycles that may be executed atomically [10]. Our main problem is pricing each cycle separately, awarding fees to sellers paid by the buyers.

MUSKETEER’s participants are required to pre-lock the coins intended for rebalancing prior to the mechanism revealing the individual cycles. This design decision is primarily to prevent buyers from choosing whether to proceed with rebalancing after the output of the mechanism is known, as this could potentially incentivize dishonest strategies. From a different perspective, if buyers have the option to abort the mechanism in hindsight, the effectiveness of the mechanism may be severely hindered as a cycle can only be executed only if all players choose to participate and lock their coins. Figure 2 illustrates MUSKETEER integrated into the Hide & Seek protocol flow.



■ **Figure 2** The backbone of MUSKETEER, integrated into the Hide & Seek protocol flow.

## 2.3 Model and Notation

### 2.3.1 Payment Channel Network (PCN)

A payment channel network can be modeled as an undirected graph, with a vertex for every user and an edge connecting users  $u, v$  whenever they jointly own a payment channel, as depicted in Figure 3(a). At any point in time, the (bidirectional) capacities of the payment channel and its current distribution of coins can be encoded as follows: the capacity of edge  $e = (u, v)$  in the direction from  $u$  to  $v$  is the maximum amount of money that can be transferred given the channel’s current coin distribution.

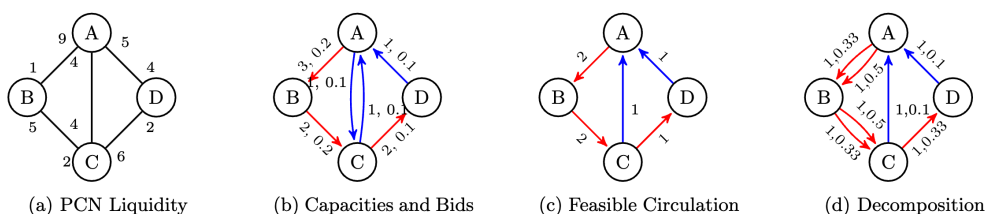
### 2.3.2 Rebalancing amounts as network flows

First, all users submit capacities for their channels in both directions. These requests from both buyers and sellers are encoded as a directed capacitated graph  $G = (V, E)$ . For a node  $u$ , the outgoing edges express the channels that  $u$  wishes to send coins to its counterparty – either because the counterparty wishes to rebalance their channel or because  $u$  wants to gain routing fees as a seller. Symmetrically, the incoming edges express the channels that node  $u$  wishes to receive coins – either because  $u$  wishes to rebalance its channel as a buyer or because its counterparty wants to gain routing fees as a seller. We note that it is, therefore, possible to have both directed edges  $(u, v)$  and  $(v, u)$  in  $E$ . The capacities of each edge  $c(e)$ ,  $e \in E$ , represent the maximum amount of flow that the owners of the channel are willing to dedicate to rebalancing. Consequently, the rebalancing problem is now transformed into a network flow problem, e.g., maximizing the rebalancing liquidity is equivalent to maximizing the flow in  $G$ , as illustrated in Figure 3(b).

From the perspective of one user, rebalancing simply transfers their liquidity from one channel to another, possibly depleted, channel. Rebalancing by itself must not result in any monetary gain or loss for any user, a property also known as *balance conservation* [10]. This does not include the fees associated with rebalancing, which may very well lead to a surplus or deficit for users. This requirement of balance conservation characterizes possible rebalancing flows as circulations. A circulation is a flow  $\mathbf{f} = (f(u, v))_{(u, v) \in E}$  such that the net flow through each vertex is zero:  $\sum_{v \in V} f(u, v) = \sum_{v \in V} f(v, u), \forall u \in V$ .

Two circulations  $\mathbf{f}_1, \mathbf{f}_2$  can be added to get yet another circulation:  $\mathbf{f}_1 + \mathbf{f}_2 = (f_1(u, v) + f_2(u, v))_{(u, v) \in E}$ . A cycle is a sequence of vertices  $v_1, v_2 \dots v_k$  such that  $(v_i, v_{i+1}) \in E, \forall 1 \leq i \leq k-1$  and  $(v_k, v_1) \in E$  as well. We equivalently refer to this cycle as  $(e_1, e_2 \dots e_k)$  where  $e_i = (v_i, v_{i+1}), \forall 1 \leq i \leq k-1$  and  $e_k = (v_k, v_1)$ . We call  $k$  the length of this cycle. A cycle flow  $\mathbf{f}$  of weight  $w$  on cycle  $C$  is a circulation where  $f(e) = w, \forall e \in C$  and  $f(e) = 0$  otherwise (cf. Figure 3(c)).

Although all circulations represent possible rebalancings, rebalancing in practice is executed through cycle flows. First, a so-called *sign-consistent cycle decomposition* of a circulation is computed, and these cycles are individually executed [10]. A sign consistent cycle decomposition of a circulation  $\mathbf{f}$  is a set of cycles  $\mathbf{f}_1, \mathbf{f}_2 \dots \mathbf{f}_k$  such that  $\mathbf{f} = \sum_i \mathbf{f}_i$  and all the cycles share the same orientation (cf. Figure 3(d)). To be precise, if two cycles  $\mathbf{f}_i, \mathbf{f}_j$  route non-zero flow through an edge  $(u, v)$ , they do so in the same direction:  $\mathbf{f}_i(u, v) > 0$  and  $\mathbf{f}_j(v, u) > 0$  cannot hold simultaneously. A standard result of network flow theory is that any circulation may be expressed as a sum of at most  $|E|$  sign-consistent cycles [2]. We are only interested in the space of feasible circulations  $\mathbf{f}$  that satisfy every capacity constraint:  $\mathbf{f} \leq \mathbf{c}$ .



■ **Figure 3** We illustrate the rebalancing process of MUSKETEER: (a) Given a PCN with specific liquidity per channel (indicated by the numbers of each node on each edge), (b) the players may submit capacities and bids (the first number indicated the submitted capacity, the arrow indicated the direction they wish to rebalance, while the second number indicates the fees they are willing to pay). Then, (c) the rebalancing circulation is calculated (the number refer to the number of coins to be transfer and the direction is indicated by the arrow), and (d) subsequently decomposed to sign-consistent cycles which are then priced (the multiple arrows indicate that the flow is divided into multiple cycles; the first number is the number of coins to be transferred and the second the fee to be paid). Depleted edges are shown in red and indifferent edges in blue. All numbers are indicative.

### 2.3.3 User valuations

In a two-party channel, rebalancing is not symmetrically beneficial. We define the utilities resulting from rebalancing flows below.

Associated with each user  $u \in V$  is a valuation function  $\mathbf{v}_u$  on the set of flows in  $G$ . We first assume this valuation to be a linear function of  $\mathbf{f}$ , so that by abuse of notation we may treat  $\mathbf{v}_u$  as a function as well as a vector:  $\mathbf{v}_u(\mathbf{f}) = \mathbf{v}_u \cdot \mathbf{f}$ . For an edge  $e$ , we denote by  $\mathbf{v}_u(e)$  the  $e$ -th coordinate of the vector  $\mathbf{v}_u$ .

If channel  $e = (u, v)$  is depleted in the direction from  $u$  to  $v$ , rebalancing should occur from  $v$  to  $u$ . This would benefit user  $u$ , thus  $u$  has a positive valuation for flow along  $e$ :  $\mathbf{v}_u(e) > 0$ . Any incurred fees are also paid by  $u$ , making  $u$  the buyer in this case. However, if a channel is not depleted, it is termed *indifferent*. Flow in either direction is allowed to aid in rebalancing the network but has a non-positive valuation for the channel owners:  $\mathbf{v}_u(e) \leq 0$ . Flow from  $u$  to  $v$  requires the authorization of the  $u$ , thus fees earned through this flow are paid to  $u$ , termed the seller.

We assume these valuations are local, meaning that the utility of users is not impacted by the flow along non-adjacent channels:  $\mathbf{v}_u(v, w) = 0$  for distinct users  $u, v, w$ . We further presume each user has a probabilistic knowledge of other users' valuations. Finally, we assume that the utility derived from rebalancing by a unit flow along any channel is bounded, encapsulated by  $\|\mathbf{v}_u\|_\infty < 0.1$ . In other words, no user is willing to pay a fee rate greater than 10%, nor can a user demand greater fees for its indifferent edges. A similar concept is already implemented in the Bitcoin Lightning Network for multi-hop payments (approximately equal to 0.03%). We stress our mechanisms function with any maximum fee rate lower than 100%, and the 10% bound is merely indicative.

### 2.3.4 User bids

Similarly to traditional auctions, user valuations are private and they may submit a different bid. Indeed, we assume all players are rational utility-maximizing agents. We call the bids *valid* when they satisfy the above assumptions on valuations.

In our problem, users submit bids  $\mathbf{b}_u$  for their channels reflecting their self-interests, as shown in Figure 3(b). Buyers submit positive bids while sellers submit negative ones, expressing the maximum/minimum amount of fees they are willing to pay/receive, respectively, per unit flow along their channels during rebalancing. Both users in an indifferent channel may participate as sellers. In depleted channels, however, one party can serve as a buyer while the counterparty is precluded from being a seller to avoid necessitating payment from  $u$  to  $v$  for routing flow. Although we distinguish between buyers and sellers for simplicity, note that users may possess multiple depleted and indifferent channels simultaneously. It is more precise to view each user as a strategic agent with specific utilities derived from their edges.

### 2.3.5 Social welfare and utility functions

Recall that individual user valuations  $\mathbf{v}_u$  are local by assumption and are nonzero only for adjacent directed edges. Let  $\mathbf{v}$  be the aggregate valuation function  $\sum_{u \in V} \mathbf{v}_u$ . Given a feasible circulation  $\mathbf{f}$ , the social welfare generated by  $\mathbf{f}$  under  $\mathbf{v}$  is defined as  $\text{SW}(\mathbf{v}, \mathbf{f}) := \mathbf{v} \cdot \mathbf{f}$ . As usual, user utilities are considered quasi-linear: if  $u$  is charged price  $\mathbf{p}$  for participation in a circulation  $\mathbf{f}$  of valuation  $\mathbf{v}_u(\mathbf{f})$ , then the player's utility is  $\mathbf{u}_u(\mathbf{f}, \mathbf{p}) := \mathbf{v}_u(\mathbf{f}) - \mathbf{p}_u$ . An example of pricing a circulation can be seen in Figure 3(d).

For a vertex  $v$ , we use the subscript “ $-v$ ” to denote the situation where  $v$  is removed from consideration.  $G_{-v}$  refers to the subgraph of  $G$  with  $v$  and all edges adjacent to  $v$  removed.  $\mathbf{v}_{-v}$ ,  $\mathbf{b}_{-v}$  denote valuation and bid vectors with coordinates for edges adjacent to  $v$  removed. Finally,  $\mathbf{u}_{-v}$ ,  $\mathbf{p}_{-v}$  denote utilities and prices of all players except  $v$ . These vectors may also be considered as elements of the larger class when it is clear from the context.

### 2.3.6 Rebalancing Game

We define here the rebalancing problem termed MUSKETEER.

► **Definition 1** (MUSKETEER). Consider a game consisting of  $n$  players, one for each of the vertices of a capacitated directed graph  $G(V, E)$  each with valuation (vector)  $\mathbf{v}_v$ ,  $v \in V$ . The coordinates correspond to the channels of player  $u$ . A Rebalancing Mechanism  $\mathcal{M} : (G, \mathbf{c}, \mathbf{b}) \mapsto (\mathbf{f}_i, \mathbf{p}_i)_{1 \leq i \leq k}$  receives edge capacities  $\mathbf{c}(e)$  and valid valuations  $\mathbf{b}_v$  as bids from each player.  $\mathcal{M}$  computes a feasible circulation  $\mathbf{f}$  as a sign-consistent cycle decomposition  $\mathbf{f}_1, \dots, \mathbf{f}_k$ . For each cycle flow  $\mathbf{f}_i$ , it computes a price vector  $\mathbf{p}_i = (\mathbf{p}_i(v))_{(v \in V)}$  that each user must pay. Each cycle  $\mathbf{f}_i$  yields a utility of  $\mathbf{u}_v(\mathbf{f}_i, \mathbf{p}_i)$  for player  $v$  as given by  $\mathbf{u}_v(\mathbf{f}_i, \mathbf{p}_i) = \mathbf{v}_v(\mathbf{f}_i) - \mathbf{p}_i(v)$ , and the total utility for player  $v$  is  $\mathbf{u}_v(\mathbf{f}) = \sum_i \mathbf{u}_v(\mathbf{f}_i)$ .

The following properties should hold:

1. **Economic Efficiency:** The cycle decomposition  $\mathbf{f}$  maximizes the social welfare under the given bids,  $\mathbf{f} = \arg \max \text{SW}(\mathbf{b}, \mathbf{f})$ .
2. **Cyclic Budget Balance:** The prices per cycle must sum zero  $\sum_v \mathbf{p}_i(v) = 0$ .
3. **Individual Rationality:** Any cycle flow  $\mathbf{f}_i$  yields non-negative utility to every truthful player,  $\mathbf{u}_u(\mathbf{f}_i) \geq 0$ .
4. **Truthfulness:** Regardless of other players' actions, the best response for utility-maximizing players is to bid truthfully,  $\mathbf{b}_v = \mathbf{v}_v$ .

In the context of our problem, *economic efficiency* refers to the maximization of the total flow weighted by users' bids, resulting in the most beneficial effect of rebalancing. Additionally, this property encompasses the prioritization of channels for rebalancing based on their respective bids.

*Individual rationality*, on the other hand, demands that each user pays no more than their bid for each channel. By adhering to rationality, users ensure that every rebalancing cycle yields non-negative utility for all players. Therefore, executing all suggested cycles, rather than selectively performing only those that are optimal to their self-interest, is necessary to achieve the maximal beneficial effect of rebalancing.

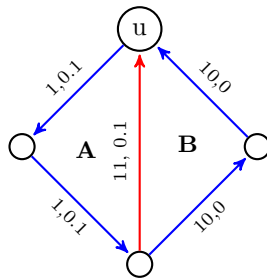
*Truthfulness* is another crucial aspect of the mechanism, whereby each player should bid their truthful valuation for each channel to ensure that no one can benefit from misreporting their valuations.

Lastly, *cyclic budget balance* is a novel property tailored to our problem. It is a more restrictive variation of the strong budget balance property and demands that there is no deficit or surplus for each cycle produced by the mechanism. There are two reasons we opt for the cyclic budget balance, both of which stem from PCN technicalities: Firstly, the rebalancing circulation is preferably decomposed into cycles. As posited in Hide & Seek, executing small cycles is faster, more robust and requires less communication among nodes[10]. In contrast, executing the entire circulation simultaneously demands complex protocols (such as that of [3]) with high network overhead, which are more likely to fail. Secondly, payment channel constructions do not allow the burning of coins, or in other words, a surplus for the mechanism. This is because the two users may cooperatively update the channel state later in order to split the burned coins, effectively reversing the "burn". Therefore, each cycle must be priced in a way that all coins are distributed among the players in the cycle. However, we showcase below that attaining cyclic budget balance is strictly harder than strong budget balance.

### Hardness of Cyclic Budget Balance

Let us demonstrate the increased complexity of attaining cyclic budget balance in comparison to strong budget balance (in conjunction with individual rationality). The following example (Figure 4) shows that *the feasible region for strong budget balance exceeds that of cyclic budget balance*: Suppose player  $u$  submits a bid of 0.1 per unit flow for his depleted channel

## 13:10 Incentive-Compatible Rebalancing for PCNs



■ **Figure 4** Depleted edges are depicted with red and indifferent edges with blue. The numbers on each edge indicate the rebalancing capacities and bids, while the rebalancing directions are indicated by the arrows.

$e$  with a rebalancing capacity of 11.  $u$  participates in two cycles,  $A$  and  $B$ .  $A$  consists of two indifferent edges bidding  $-0.1$  each (total  $-0.2$  per unit flow) with capacity 1, while  $B$  is composed of two indifferent edges with 0 bids and capacity 10. Regardless of the chosen budget balance property, cycle  $B$  can be selected. However, cycle  $A$  fails to satisfy cyclic budget balance as any rational pricing would result in a deficit of  $-0.1$  per unit flow. However, strong budget-balanced solutions may include both cycles  $A$  and  $B$ , having  $u$  pay  $0.2/11 < 0.1$  fees per unit flow *on average*. Thus, cyclic budget balance restricts the solution space more than strong budget balance.

### 3 Towards Truthful Rebalancing

In this section, we explore how to provide incentive-compatible rebalancing in various settings using auction theory, yielding a flurry of results. In particular, we first prove that satisfying all the desired properties of the MUSKETEER is impossible by applying the classic Myerson-Satterthwaite impossibility result for double auctions (Section 3.1).

To circumvent the impossibility, we present a variety of mechanisms, all of which relax the notion of economic efficiency by restricting the set of possible bids we consider when maximizing social welfare. In particular, we first consider the limited setting where buyers and sellers choose to participate in the mechanism knowing upfront the maximum and minimum fees they would potentially pay or gain, respectively (Section 3.2). The presented algorithm is fairly simple but restricts the choices for participants.

To expand our results to the broader context where players are allowed to submit bids, we relax our model to a single auction, solely considering the buyer's incentives. Specifically, we assume players are willing to forward flow through their indifferent edges hoping to earn some fees in the process, but without a guarantee on the fees. Under this assumption, we present a VCG-type mechanism, satisfying incentive compatibility for buyers (Section 3.3).

Next, we present a double-auction mechanism that takes into account the bids of both buyers and sellers, albeit sacrificing truthfulness, similarly to a first price auction (Section 3.4).

Finally, we leverage time delays to navigate around the impossibility result and design a novel double auction that satisfies all the desiderata in exchange for some costs that users incur in the form of time delays (Section 3.5).

In the following, we provide a high-level description of the various mechanisms, named after the Four Musketeers, highlighting their different design choices and trade-offs. The algorithm facilitating the cycle decomposition is abstracted from the exposition of these mechanisms, and the protocol implementing the atomic execution of these cycles is likewise not detailed. Indicative algorithms that realize these functions can be found in [10] as well as in Section 3.6 for completeness.



### 3.1 Impossibility Result

► **Theorem 2.** *No mechanism can simultaneously satisfy all the desired properties of MUSKETEER, namely economic efficiency, individual rationality, truthfulness, and cyclic budget balance.*

**Proof.** We formulate the double auction problem as a rebalancing game, thus showing that if all properties are satisfied in MUSKETEER then that would be true also for the double auction problem, hence the impossibility of Myerson-Satterthwaite does not hold.

Suppose  $A$  wishes to sell an item and  $B$  wishes to buy it, with individual valuations  $V_a, V_b$  respectively. Each player knows their own valuation with certainty but the valuation of the other player only probabilistically. Without loss of generality, we normalize the valuations to lie in  $[0, 1]$ .

Now construct the following instance of MUSKETEER: the graph  $G = (V, E)$  consists of  $V = \{a, b, c\}$ ,  $E = \{(a, c), (c, b), (b, a)\}$  and with  $\mathbf{c}(e) = 1, \forall e \in E$ . For a flow  $\mathbf{f} = (f_1, f_2, f_3)$  - that is,  $f_1$  units going from  $a$  to  $c$ ,  $f_2$  from  $c$  to  $b$ , and  $f_3$  from  $b$  to  $a$  - set the valuations  $\mathbf{v}_a(\mathbf{f}) = -V_a f_1$ ,  $\mathbf{v}_b(\mathbf{f}) = V_b f_2$ ,  $\mathbf{v}_c(\mathbf{f}) = 0$ . We suppose that the players submitted bids  $\mathbf{b}_a, \mathbf{b}_b, \mathbf{b}_c$  respectively, and in particular that  $c$  was honest:  $\mathbf{b}_c = \mathbf{v}_c = 0$ .

The only non-zero feasible circulation is  $\mathbf{f} := (1, 1, 1)$ , so that the mechanism must decide solely between  $\mathbf{f}$  and 0. It must also choose a price vector  $\mathbf{p}$  satisfying cyclic budget balance:  $\mathbf{p}_a + \mathbf{p}_b + \mathbf{p}_c = 0$ .

We interpret choosing  $\mathbf{f}$  as a trade occurring between  $A$  and  $B$ , and choosing 0 as no trade. An efficient mechanism must output  $\mathbf{f}$  if  $V_b > V_a$  (the buyer values the commodity more than the seller). This corresponds to Pareto Efficiency. Individual rationality of players  $a, b$  directly corresponds to individual rationality of  $A$  and  $B$ . Next, individual rationality of  $c$  (the ‘‘auctioneer’’) demands that  $\mathbf{p}_c \leq \mathbf{b}_c = 0$ , which corresponds to Weak Budget Balance. Truthfulness in our setting matches that of Myerson-Satterthwaite: in both cases, we require the truthful bid to be the best response.

In this manner, a solution to MUSKETEER can be used to simulate a single buyer single seller trade as studied by Myerson and Satterthwaite [33]. As a result, all four desired properties cannot be concurrently realized without additional assumptions. ◀

### 3.2 Athos: A Mechanism for Fixed Fees

In this section, we present a straightforward approach for incorporating fees into rebalancing. To circumvent the aforementioned impossibility, the input to the mechanism is restricted. Users do not submit bids. Instead, a predetermined fee rate of  $\hat{p}$  is made publicly known (such as the most commonly chosen fee rate<sup>1</sup>). All flow through indifferent channels will be paid at this fee rate. There is an additional parameter  $k$  that bounds the maximum fee rate for buyers: flow through depleted edges will be charged at a fee rate  $\leq k\hat{p}$ .

Given these parameters, users can decide upfront if they want to participate in the mechanism. Instead of bidding, they specify which of their channels are depleted.  $D \subseteq E$  denotes the set of depleted edges, and the rest are considered indifferent edges, denoted by  $I = E \setminus D$ . The rebalancing flow is chosen to optimize:  $\sum_{e \in D} k\mathbf{f}(e) - \sum_{e \in I} \mathbf{f}(e)$ . The rebalancing is then decomposed into sign-consistent cycles, and a separate price vector is computed for each cycles that achieves cyclic budget balance. This way we achieve all the desiderata but under a restricted setting.

<sup>1</sup> Bitcoin Lightning fees: [https://www.reddit.com/r/lightningnetwork/comments/tm1k1c/bmonthly\\_ln\\_fee\\_report/](https://www.reddit.com/r/lightningnetwork/comments/tm1k1c/bmonthly_ln_fee_report/)

## 13:12 Incentive-Compatible Rebalancing for PCNs

This structure leads to a simple mechanism (ATHOS), a natural evolution of Hide & Seek including fees. We observe however that certain rebalancing cycles are not considered: given the parameter  $k$ , any rebalancing cycle must contain at least one depleted edge for every  $k$  indifferent edges.

### ATHOS: Fixed fees

*Input:* Channel capacities  $\mathbf{c}$  and the set of depleted edges  $D \subseteq E$ .

1. Let  $I = E \setminus D$  be the set of indifferent edges.
2. Compute the optimal rebalancing  $\mathbf{f} := \arg \max_G \sum_{e \in D} \hat{p} \mathbf{f}(e) - \sum_{e \in I} \hat{p} \mathbf{f}(e)$ .
3. For this flow, the total cost incurred is  $C = \hat{p} \sum_{e \in I} \mathbf{f}(e)$ .
4. Consider a sign-consistent cycle decomposition  $\mathbf{f}_1, \mathbf{f}_2 \dots \mathbf{f}_k$  of  $\mathbf{f}$ , and define the cost incurred per cycle as  $C_i = \hat{p} \sum_{e \in I} \mathbf{f}_i(e)$ .
5.  $C_i$  is distributed to the depleted edges in  $\mathbf{f}_i$ . Notice that every cycle  $\mathbf{f}_i$  must contain at least one depleted edge per  $k - 1$  indifferent edges, otherwise remove  $\mathbf{f}_i$  from  $\mathbf{f}^*$  to get a more optimal solution.
6. If the  $i$ th cycle  $\mathbf{f}_i$  contains  $n_i$  depleted edges, then each depleted edge is charged at fee rate  $C_i/n_i$  during the execution of  $\mathbf{f}_i$ . All indifferent edges earn fees at rate  $\hat{p}$ .

*Output:* Cycle flows with prices  $(\mathbf{f}_i, \mathbf{p}_i)$ , each released only to involved players.

► **Theorem 3.** *ATHOS:  $(G, \mathbf{c}, D) \mapsto (\mathbf{f}_i, \mathbf{p}_i)_{1 \leq i \leq k}$ ,  $D \subseteq E$  expressing the set of depleted edges, satisfies economic efficiency, individual rationality, and cyclic budget balance. It also provides sellers with a fee of  $\hat{q} \leq k\hat{p}$  per unit flow along their edges.*

**Proof.** This mechanism assumes bids of  $k\hat{p}, \hat{p}$  for depleted and indifferent edges resp., and selects a circulation maximizing social welfare under these bids, thus achieving economic efficiency.

Step 3 clearly indicates that sellers receive a fixed fee for each unit of flow. The parameters  $\hat{p}, k$  are publicly known in advance, hence a user can decide a priori whether it is beneficial to participate in ATHOS based on their private valuations. Individual rationality of player is thus implicit in their participation in the mechanism, along with the fact that all indifferent edges earn fees at rate  $\hat{p}$ , and depleted edges are charged at rate  $\leq k\hat{p}$ .

The fee computation in Step 4 is cyclic budget balanced by design: since we consider the cycle decomposition of  $\mathbf{f}^*$  and charge fees per cycle, the fees charged to depleted edges are identical to the fees levied by the indifferent edges. ◀

### 3.3 Porthos: A Truthful Single Auction

The impossibility of Section 3.1 indicates that achieving all the desiderata is not possible for both buyers and sellers in the original setting. In particular, in our setting, the cyclic budget balance property is critical since burning coins is not possible in payment channels. For this reason, the most straightforward way to circumvent the aforementioned impossibility is to either restrict our setting, as in Section 3.2 where the bids were fixed and known a priori, or revert to a single auction by assuming that sellers will accept any reward that is non-negative.

We, thereby, present here a single auction mechanism where only non-negative bids are permitted: positive bids for depleted channels, and zero for indifferent channels. Instead, all the users of a PCN may participate in the rebalancing process hoping to receive some fees from the mechanism.

We construct a VCG-type mechanism to determine the price vector of buyers based on their impact on social welfare, achieving incentive compatibility for buyers. Charging these prices would result in some surplus for the mechanism, which is instead redistributed to owners of indifferent channels to achieve cyclic budget balance.

**PORTHOS: A VCG-type single auction**

*Input:* Channel capacities  $\mathbf{c}$  and non-negative player bids  $\mathbf{b}_v \geq 0$ .

1. Compute the optimal rebalancing  $\mathbf{f} := \arg \max_G SW(\mathbf{b}, \mathbf{f})$ .
2. Compute an alternative rebalancing for every player  $v$ ,  $\mathbf{f}_{-v} := \arg \max_G SW(\mathbf{b}_{-v}, \mathbf{f})$ .
3. Charge  $v$  the price  $\mathbf{p}(v) := SW(\mathbf{b}_{-v}, \mathbf{f}_{-v}) - SW(\mathbf{b}_{-v}, \mathbf{f})$ .
4. Let  $\mathbf{f}_1, \dots, \mathbf{f}_k$  be a sign-consistent cycle decomposition of  $\mathbf{f}$ . Non-zero prices  $\mathbf{p}(v)$  are split into  $\mathbf{p}_i(v)$  for each  $\mathbf{f}_i$  proportional to  $v$ 's valuation of  $\mathbf{f}_i$ :  

$$\mathbf{p}_i(v) := \mathbf{p}(v) \frac{SW(\mathbf{b}_v, \mathbf{f}_i)}{SW(\mathbf{b}_v, \mathbf{f})}.$$
5. The total fees per cycle  $\mathbf{f}_i$  are  $q_i = \sum \mathbf{p}_i(v)$  for every buyer in  $\mathbf{f}_i$ .
6. If  $\mathbf{f}_i$  has  $m$  sellers  $u_1, u_2, \dots, u_m$ , then  $p_i(u_j) := -\frac{\sum q_i}{m}$ .

*Output:* Cycle flows with prices  $(\mathbf{f}_i, \mathbf{p}_i)$ , each released only to involved players.

► **Theorem 4.** *PORTHOS:  $(G, \mathbf{c}, \mathbf{b}) \mapsto (\mathbf{f}_i, \mathbf{p}_i)_{1 \leq i \leq k}$  assuming  $\mathbf{b} \geq 0$ , satisfies economic efficiency, individual rationality, and cyclic budget balance. Users' bids for depleted edges are truthful.*

**Proof.** A feasible circulation  $\mathbf{f}$  that maximizes social welfare under  $\mathbf{b}$  achieves economic efficiency. For a player  $v$ , let  $\mathbf{f}_{-v}$  be a feasible circulation on  $G$  maximizing social welfare under bids  $\mathbf{b}_{-v}$ . We set  $\mathbf{p}(v) := SW(\mathbf{b}_{-v}, \mathbf{f}_{-v}) - SW(\mathbf{b}_{-v}, \mathbf{f})$ .

It is sufficient to show truthfulness under the pricing  $\mathbf{p}'(v) := -SW(\mathbf{b}_{-v}, \mathbf{f})$ , since  $\mathbf{p}$  and  $\mathbf{p}'$  are revenue equivalent: meaning that their difference  $\mathbf{p} - \mathbf{p}'$  is a function of  $\mathbf{b}_{-v}$  and  $G_{-v}$ , and this function crucially does not depend on player  $v$ 's bid or valuation.

Under  $\mathbf{p}'$ , player  $v$  is incentivized to bid truthfully regardless of every other player's action. Consider  $\mathbf{b} = (\mathbf{v}_v, \mathbf{b}_{-v})$ ,  $\mathbf{b}' = (\mathbf{v}'_v, \mathbf{b}_{-v})$  for any other valuation  $\mathbf{v}'_v \neq \mathbf{v}_v$ . When  $v$  reports valuation honestly, the mechanism selects  $\mathbf{f}$  maximizing social welfare under  $\mathbf{b}$ , and player  $v$ 's utility is given by  $\mathbf{v}_v(\mathbf{f}) - \mathbf{p}'(v) = \mathbf{v}_v(\mathbf{f}) + SW(\mathbf{b}_{-v}, \mathbf{f}) = SW(\mathbf{b}, \mathbf{f})$ . In the second case, the mechanism selects a possibly different  $\mathbf{f}'$  maximizing social welfare under  $\mathbf{b}'$  and the utility for player  $v$  is:  $\mathbf{v}_v(\mathbf{f}') + SW(\mathbf{b}_{-v}, \mathbf{f}') = SW(\mathbf{b}', \mathbf{f}')$ . Since  $SW(\mathbf{b}, \mathbf{f}') \leq SW(\mathbf{b}, \mathbf{f})$  by definition of  $\mathbf{f}$ , we have that bidding honestly always achieves the maximum possible utility regardless of other players' actions. In other words, both pricings  $\mathbf{p}, \mathbf{p}'$  are Nash-equilibrium incentive-compatible.

Finally, we show individual rationality, or that buyer utilities are non-negative under price  $\mathbf{p}$ . Buyer  $v$ 's utility is  $\mathbf{u}_v = SW(\mathbf{b}, \mathbf{f}) - SW(\mathbf{b}_{-v}, \mathbf{f}_{-v})$  which must be non-negative as  $SW(\mathbf{b}_{-v}, \mathbf{f}_{-v}) \leq SW(\mathbf{b}, \mathbf{f}_{-v}) \leq SW(\mathbf{b}, \mathbf{f})$  by definition of  $\mathbf{f}$ . We note that the first inequality only holds for non-negative bids  $\mathbf{b}_v$ .

## 13:14 Incentive-Compatible Rebalancing for PCNs

By the computation in Step 5, the fees charged to depleted edges are equally distributed to all indifferent edges for each cycle. In other words, PORTHOS satisfies cyclic budget balance. ◀

### 3.4 Aramis: A non-truthful Double Auction

As a stepping stone to Section 3.5, we present a straightforward double-auction mechanism (Mechanism ARAMIS) that accepts both positive and negative bids, and satisfies all properties but truthfulness. The rationale of Algorithm 3 resembles that of a first-price auction.

#### ARAMIS: A Double Auction

*Input:* Channel capacities  $\mathbf{c}$  and player bids  $\mathbf{b}_v$ .

1. Compute the optimal rebalancing  $\mathbf{f} := \arg \max_G SW(\mathbf{b}, \mathbf{f})$ .
2. Let  $\mathbf{f}_1, \dots, \mathbf{f}_k$  be a sign-consistent cycle decomposition of  $\mathbf{f}$ .
3. Suppose  $\mathbf{f}_i$  is a cycle flow of length  $n_i$ . The price  $\mathbf{p}_i(v)$  for  $v$ 's participation in  $\mathbf{f}_i$  is:  

$$\mathbf{p}_i(v) := \mathbf{b}_v(\mathbf{f}_i) - \frac{SW(\mathbf{b}, \mathbf{f}_i)}{n_i} \quad (\mathbf{p}_i(v) = 0 \text{ when } v \text{ is not part of } \mathbf{f}_i).$$

*Output:* Cycle flows with prices  $(\mathbf{f}_i, \mathbf{p}_i)$ , each released only to involved players.

► **Theorem 5.** *ARAMIS:  $(G, \mathbf{c}, \mathbf{b}) \mapsto (\mathbf{f}_i, \mathbf{p}_i)_{1 \leq i \leq k}$  satisfies economic efficiency, individual rationality, and cyclic budget balance, but not truthfulness.*

**Proof.** The feasible circulation  $\mathbf{f}$  maximizes social welfare under  $\mathbf{b}$  and thus achieves economic efficiency. The social welfare of each cycle  $\mathbf{f}_i$  under  $\mathbf{b}$  must be non-negative, else the circulation  $\mathbf{f} - \mathbf{f}_i$  would have greater social welfare than  $\mathbf{f}$ , contradicting its optimality. Intuitively, the social welfare per cycle is shared uniformly by all involved vertices.

For a truthful player  $v$ , their utility under a cycle  $\mathbf{f}_i$  of length  $n_i$  is given by  $\mathbf{u}_{i,v}(\mathbf{f}_i) = \mathbf{v}_{i,v}(\mathbf{f}_i) - \mathbf{p}_{i,v}(\mathbf{f}_i) = \frac{SW(\mathbf{b}, \mathbf{f}_i)}{n_i} \geq 0$ . This proves individual rationality per cycle. From the price calculation in Step 3, we can readily confirm that the sum of the prices along each cycle is zero:

$$\sum_{j=1}^{n_i} \mathbf{p}_{i,v_j}(\mathbf{f}_i) := \sum_{j=1}^{n_i} \mathbf{b}_{v_j}(\mathbf{f}_i) - SW(\mathbf{b}, \mathbf{f}_i) = 0. \quad \blacktriangleleft$$

**Remark.** Players' incentives mirror first-price auctions: They are incentivized to bid higher to ensure their participation in the rebalancing circulation over other competing players. But for a given rebalancing circulation, players are incentivized to bid lower to maximize utility.

### 3.5 d'Artagnan: A Truthful Double Auction with Time Delays

Mechanism ARAMIS is straightforward but lacks the crucial property of truthfulness. To mitigate this issue, we introduce time delays into the rebalancing cycles (mechanism D'ARTAGNAN). The basic concept is that cycles with lower social welfare will be released later in time. Consequently, users who attempt to save on fees by underbidding will experience an undesirable delay in rebalancing. This concept is akin to that of opportunity cost, where users face potential losses from the inability to use their locked funds.

D'ARTAGNAN first computes an optimal rebalancing circulation and decomposes into sign-consistent cycle flows with prices  $(\mathbf{f}_i, \mathbf{p}_i)$ , similar to ARAMIS. Moreover, D'ARTAGNAN selects a time  $t_i \in [0, 1]$  for every flow  $\mathbf{f}_i$  (e.g.,  $t = 1$  represents an 1 hour delay). We assume delaying execution until time  $t_i \leq 1$  gives player  $v$  a utility of  $\mathbf{u}_v = \mathbf{v}_v(\mathbf{f}) - \mathbf{p}_v + d(1 - t)$ . Users join the mechanism with the implicit assumption that rebalancing cycles are released at time  $t = 1$ . Any earlier rebalancing improves the utility of a player at the rate  $d$ , a configurable parameter of our mechanism that depicts the estimated opportunity costs of players.

**D'ARTAGNAN: A Double Auction with delays**

*Input:* Channel capacities  $\mathbf{c}$ , player bids  $\mathbf{b}_v$ , and global *delay factor*  $d$ .

1. Compute the optimal rebalancing  $\mathbf{f} := \arg \max_G SW(\mathbf{b}, \mathbf{f})$ .
2. Let  $\mathbf{f}_1, \dots, \mathbf{f}_k$  be a sign-consistent cycle decomposition of  $\mathbf{f}$ .
3. Suppose  $\mathbf{f}_i$  is a cycle flow of length  $n_i$ . The price  $\mathbf{p}_i(v)$  for  $v$ 's participation in  $\mathbf{f}_i$  is:  

$$\mathbf{p}_i(v) := \mathbf{b}_v(\mathbf{f}_i) - \frac{SW(\mathbf{b}, \mathbf{f}_i)}{n_i}$$
.  $\mathbf{p}_i(v)$  is set to zero when  $v$  is not part of the cycle flow.
4. Let  $n_i$  be the length of the cycle flow  $\mathbf{f}_i$ . Define the delay of  $\mathbf{f}_i$  as  

$$t_i = 1 - \left(1 - \frac{1}{n_i}\right) \frac{SW(\mathbf{b}, \mathbf{f}_i)}{d}$$
.

*Output:* The  $i$ th pair  $(\mathbf{f}_i, \mathbf{p}_i)$  is released to involved players at time  $t_i$ .

► **Theorem 6.** *D'ARTAGNAN:  $(G, \mathbf{c}, \mathbf{b}, d) \mapsto (\mathbf{f}_i, \mathbf{p}_i)_{1 \leq i \leq k}$  where  $d$  is an additional delay parameter, satisfies economic efficiency, truthfulness, cyclic budget balance, and individual rationality.*

**Proof.** Cyclic budget balance and economic efficiency follow as in Mechanism PORTHOS since Steps 1 – 3 are identical in both PORTHOS and D'ARTAGNAN. To analyze individual rationality and truthfulness, let us compute the utility of a player  $v$ . Due to the sign consistency of cycles,  $v$ 's utility can be expressed as the sum of utilities induced by each of the  $k$  cycles:  $\mathbf{u}_v = \sum_i \mathbf{u}_v(\mathbf{f}_i)$ .

The utility of  $v$  per cycle  $\mathbf{f}_i$  is:

$$\begin{aligned} \mathbf{u}_v(\mathbf{f}_i) &= \mathbf{v}_v(\mathbf{f}_i) - \left( \mathbf{b}_v(\mathbf{f}_i) - \frac{SW(\mathbf{b}, \mathbf{f}_i)}{n_i} \right) + d - dt_i \\ &= \mathbf{v}_v - \mathbf{b}_v + \frac{SW(\mathbf{b}_v, \mathbf{f}_i)}{n_i} + \left(1 - \frac{1}{n_i}\right) SW(\mathbf{b}, \mathbf{f}_i) = (\mathbf{v}_v, \mathbf{b}_{-v}) \cdot \mathbf{f}_i \end{aligned}$$

simplifying to  $SW((\mathbf{v}_v, \mathbf{b}_{-v}), \mathbf{f}_i)$ . Since  $v$ 's utility is independent of their bid, D'ARTAGNAN is truthful.

In fact, this utility matches the social welfare if bids were honest:  $\mathbf{u}_i(\mathbf{f}_i) = SW(\mathbf{b}, \mathbf{f}_i)$ . The social welfare of  $\mathbf{f}_i$  cannot be negative. If  $SW(\mathbf{b}, \mathbf{f}_i) < 0$ , then  $\mathbf{f}$  is not an optimal solution: as  $\mathbf{f}_1, \dots, \mathbf{f}_k$  is a sign consistent cycle decomposition, removal of  $\mathbf{f}_i$  from the circulation  $\mathbf{f}$  leads to a feasible solution that is strictly better. This proves individual rationality. ◀

**Remark.** To guarantee both truthfulness and individual rationality, the users lock their coins to the mechanism a priori for the maximum time delay. Otherwise, buyers may benefit from participating in the mechanism even when the maximum time delay supersedes their

## 13:16 Incentive-Compatible Rebalancing for PCNs

true valuations: buyers might only participate in the execution phase (i.e., the sale) if they are quoted a favorable price. This is undesirable behavior as it affects all other users in the cycle. Hence, we enforce the execution of cycles according to the mechanism [38]. However, this hinders economic efficiency, as there may be buyers with ex-ante utility (i.e., utility of player before the output of the mechanism is known) less than their ex-post utility (i.e., utility of player after the output of the mechanism is known). As a result, there may be buyers who would have participated in the mechanism but chose not to, therefore leading to suboptimal outcomes.

### 3.6 Additional Algorithms

In the following, we present for completeness indicative protocols that can implement the cycle decomposition and the atomic execution of these cycles.

#### Sign-Consistent Cycle Decomposition

We first outline the algorithm for the cycle decomposition, as introduced in [10]. Algorithm 1 leverages depth-first search to identify cycles and then applies cycle flows to them.

■ **Algorithm 1** Depth-first Search Cycle Decomposition.

---

```
input : Circulation  $\mathbf{f}$  on directed graph  $G = (V, E)$ 
output : A set of cycle flows  $\mathcal{S}$  that sum to  $\mathbf{f}$ 
initialize  $i = 1$ 
initialize  $R \leftarrow \{e \in E : f(e) \neq 0\}$  set of active edges
while  $R \neq \emptyset$  do
  pick an edge  $e_1 \in R$ 
  run depth first search to find a cycle  $C_i = (e_1, e_2, \dots, e_k)$  in  $R$ 
   $w_i \leftarrow \min_{e \in C_i} f(e)$ 
  initialize  $\mathbf{f}_i \leftarrow 0$ 
  for  $e \in C_i$  do
     $f_i(e) = w_i$ 
     $f(e) \leftarrow f(e) - f_i(e)$ 
    if  $f(e) = 0$  then
       $\perp$  delete  $e$  from  $R$ 
   $i \leftarrow i + 1$ 
return  $\mathcal{S} = \{\mathbf{f}_1, \mathbf{f}_2 \dots \mathbf{f}_i\}$ 
```

---

#### Atomic Execution of Rebalancing Cycles

Next, we present an algorithm that ensures the secure atomic execution of the rebalancing cycles, taking place after the output of each respective rebalancing mechanism, e.g. D'ARTAGNAN.

Provided a set of (sign-consistent) rebalancing cycles, Algorithm 2 randomly selects one user for each cycle responsible for initiating the execution. This user selects a random number  $r_c$  and sends its cryptographic digest  $h_c = H(r_c)$  to the other users in its cycle. The initiator and the next user have their timelock set to the cycle's length, while the transaction value is the cycle's weight  $w_c$ . Each user in the sequence reduces the timelock by 1, identifies the next user in the cycle for HTLC creation based on vertex order, and sets up an HTLC with the updated timelock.

■ **Algorithm 2** HTLC creation for cycles.

---

```

input :  $\mathcal{S}$  set of directed cycles
for  $c \in \mathcal{S}$  do
    select starting user  $u_c$  at random from users in  $c$ 
    timelock  $t_c \leftarrow \text{len}(c)$ 
     $u_c$  chooses random secret  $r_c$  and creates hash  $h_c = H(r_c)$ 
    for  $e_c = (u, v) \in c$  starting from  $u_c$  do
         $u$  creates  $HTLC(u, v, w_c, h_c, t_c)$ 
        decrement  $t_c$  by 1

```

---

Algorithm 2 follows [10] and is only indicative. It can be replaced by any other protocol that achieves atomic execution of multi-hop payments in PCNs, e.g., [49, 47, 46]. For example, MAPPCN [46] can be leveraged to preserve user anonymity, while MAD-HTLC [47] or He-HLTC [49] can be employed to ensure security even when the blockchain miners can be bribed to enable fraud (so-called timelock bribing attacks [34]).

## 4 Limitations, Extension, and Future Work

Our work leaves open several interesting research avenues which we outline below.

### 4.1 Minimum Fees for Sellers in Aramis

The primary limitation of PORTHOS is that buyer prices rely on the graph structure, resulting in seller's fees being contingent on the number of possible rebalancing cycles in the graph, e.g., if the graph has only one feasible cycle, sellers earn no fees.

A key question is whether it is feasible to guarantee a minimum fee per unit flow through indifferent edges in the mechanism. For a seller, rebalancing is comparable to a typical transaction in the PCN, wherein the seller forwards coins through their channels and earns service fees. Thus, a seller's earnings generally rely on their highly connected position in the network and the amount of capital they have invested. The fee per unit flow (i.e, transfer of one coin) is determined by the intermediary, i.e., the node that sends the coin to the counterparty in their channel. As mentioned earlier, most intermediaries select the same fee per unit flow for forwarding transactions. We thus inquire whether it is feasible to design a novel VCG-style mechanism based on Mechanism PORTHOS, where the graph is modified to guarantee a sufficiently large surplus. Note that the fee earned by sellers is essentially a redistribution of the surplus, and a surplus that is large enough guarantees a minimum fee for every seller.

### 4.2 Incentives

The binary classification of truthfulness is an oversimplification. Future research could aim to quantify and lower bound the potential benefits of misrepresenting bids, such as the gains achieved by underbidding a certain amount instead of truthfully reporting one's valuation.

### 4.3 Variable Delay Costs

In our primary mechanism D'ARTAGNAN, we assume a uniform time delay factor for all players. However, this assumption may not be realistic since different players may experience time delays differently, leading to distinct levels of utility loss. Our model can incorporate

this variation by allowing for different delay factors ( $d$ ) for each player. The delay factor can also be construed as the opportunity cost of unused capital in depleted edges, i.e., the potential gain from fees had the player rebalanced his channel. We conjecture that this opportunity cost is quantifiable if buyers furnish their proposed fees for routing since these fees are typically determined by evaluating this loss. Therefore, we can expand our model and require all players, including both buyers and sellers, to submit their anticipated fees. Nevertheless, incorporating this alteration into our model is not a straightforward task. Buyers could potentially manipulate their combined bid by taking into account both the maximum time delay and fees they are willing to incur, consequently violating incentive compatibility.

#### **4.4 Repeated Games**

A pertinent inquiry stemming from the repeated utilization of rebalancing in PCNs is whether the expected behavior of players would be altered if they were aware that the rebalancing mechanism would occur frequently. Specifically, we ask how would the mechanism design be impacted if we shift our game to a repeated setting. We hypothesize that if the rebalancing game occurs with sufficient frequency, underbidding may be beneficial as the opportunity to rebalance would be reduced but not entirely eliminated. Conversely, if the rebalancing game is infrequent, players may miss their chance to rebalance. We thus anticipate that integrating frequency-dependent utility losses may significantly alter the results of the rebalancing game.

#### **4.5 Group Strategy-Proof Mechanisms**

Both PORTHOS and D'ARTAGNAN are strategy-proof but not group strategy-proof. While a single user's misreported bids cannot improve their utility, in certain cases, two users can manipulate their bids to jointly increase their utilities. Consider for instance the parties  $u, v$  of a depleted channel in PORTHOS. If the channel is depleted from  $u$  to  $v$ , then an honest  $u$  would truthfully report a positive bid from  $v$  to  $u$ , thus prohibiting  $v$  from gaining routing fees for the  $u, v$  channel. However, both  $u$  and  $v$  may gain by  $u$  misreporting a zero bid for the channel. This misrepresentation converts the channel's status from depleted to indifferent, enabling the potential for  $v$  to gain routing fees while precluding the possibility that  $u$  pays any fees. Given this example, an intriguing open problem is designing group strategy-proof mechanisms specifically tailored to counter collusion between a channel's joint owners.

### **5 Additional Related Work**

#### **5.1 Blockchain Scalability & Payment Channel Networks**

Improving the blockchain transaction throughput has garnered interest since the inception of Bitcoin [35]. Proposed solutions include increasing the block size, sharding the blockchain, or moving the workload off-chain leveraging so-called layer-2 protocols such as sidechains, channels, and rollups (see [25, 24] for recent surveys). Among these solutions, payment channel networks, such as the Bitcoin Lightning Network [38], have attracted substantial attention because they enable instant, low-cost off-chain transactions.

A large body of research has emerged focusing on various aspects of PCNs, such as efficient and privacy-preserving routing, e.g. [43, 39, 37, 31, 4, 45], and algorithmic analysis of the PCN topology [6, 5]. In the intersection of PCNs and game theory, there are several



works, mainly focusing on network topology leveraging network creation games [19, 7, 5], and incentive-compatible outsourcing of channels' dispute resolution [32, 9, 8, 30]. All these works are orthogonal and complementary to ours as they ignore channel depletion.

Perhaps the most relevant work to ours is Merchant [48], employing fee functions as a mechanism to avert channel depletion by guiding routing paths. By allowing intermediaries to impose varying fees for distinct routes, users are incentivized to prefer specific routes over others, ultimately mitigating channel depletion. This method presents a complementary approach to our work, in which we propose an opt-in rebalancing protocol to address channel depletion.

The problem of channel rebalancing has been studied in several recent works [26, 10, 1], which we build upon and extend. Our work is the first to consider user incentives in the context of rebalancing mechanisms for PCNs.

## 5.2 Game-theoretic Analysis of Blockchains

Numerous studies have investigated incentives in the context of the consensus layer of blockchains. For instance, Pass and Shi introduced an innovative incentive-compatible consensus protocol called FruitChains [36]. Additionally, several works focused on a rational analysis of Bitcoin's consensus: exploring when rational miners follow the protocol [13, 27], devising attacks that showcase Bitcoin is not incentive-compatible, e.g., [21, 29, 41, 44], investigating the impact of block rewards and mining pools, e.g., [16, 15, 20, 42]. On the other hand, Babaioff et al. [12] explored the network layer of blockchains and proposed an incentive-compatible scheme for information propagation within Bitcoin's peer-to-peer network. However, these works address different issues from ours, as they focus on the consensus and network layers of blockchains, while our research investigates incentives on layer-2 networks that build upon the other layers.

## 5.3 Mechanism Design on Networks

The rebalancing problem fits into the well-established research area of mechanism design on networks, with an impact on computer science, economics, and operations research. The most relevant examples to our problem include Stackelberg routing, selfish routing in capacitated networks, and optimal oblivious routing, e.g., [17, 11, 40, 28]. Our work differs from the existing literature in several ways. First, we focus on a novel problem domain – rebalancing mechanisms for PCNs – which has not been previously studied from a game-theoretic perspective. Second, we deal with a unique set of constraints due to the nature of payment channels and PCNs, such as channel depletion and cyclic budget balance. These constraints lead to novel challenges in the design of incentive-compatible mechanisms, addressed in this work.

## 6 Conclusion

In this paper, we revisited the challenge of rebalancing in payment channel networks (PCNs) from a mechanism design perspective, introducing a novel approach that takes into account users' incentives. By incorporating both buyers and sellers of channel liquidity in our proposed rebalancing mechanism, we introduced the double-auction rebalancing problem MUSKETEER, which aims to optimize the throughput in PCNs.

Our work demonstrates that the unique characteristics of PCNs, particularly the cyclic budget balance property, pose significant challenges in designing a mechanism that simultaneously satisfies all the desiderata. Our results, grounded in auction theory, revealed an impossibility result, leading us to develop a variety of mechanisms that balance the various desiderata. Notably, we introduced a novel mechanism that employs time delays to overcome the impossibility result, successfully meeting all desired properties, albeit at the expense of economic efficiency in terms of time delays and liquidity combined.

---

## References

- 1 Rebalance plugin. <https://github.com/lightningd/plugins/tree/master/rebalance>.
- 2 R. Ahuja, T. Magnanti, and J. Orlin. *Network flows - theory, algorithms and applications*. Prentice Hall, 1993.
- 3 Lukas Aumayr, Kasra Abbaszadeh, and Matteo Maffei. Thora: Atomic and privacy-preserving multi-channel updates. *IACR Cryptol. ePrint Arch.*, page 317, 2022. URL: <https://eprint.iacr.org/2022/317>.
- 4 Lukas Aumayr, Pedro Moreno-Sanchez, Aniket Kate, and Matteo Maffei. Blitz: Secure Multi-Hop payments without Two-Phase commits. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 4043–4060. USENIX Association, August 2021. URL: <https://www.usenix.org/conference/usenixsecurity21/presentation/aumayr>.
- 5 Georgia Avarikioti, Gerrit Janssen, Yuyi Wang, and Roger Wattenhofer. Payment network design with fees. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, pages 76–84. Springer, 2018.
- 6 Georgia Avarikioti, Yuyi Wang, and Roger Wattenhofer. Algorithmic channel design. In *29th International Symposium on Algorithms and Computation, Jiaoxi, Yilan County, Taiwan, 2018*.
- 7 Zeta Avarikioti, Lioba Heimbach, Yuyi Wang, and Roger Wattenhofer. Ride the lightning: The game theory of payment channels. In *International Conference on Financial Cryptography and Data Security*, 2020.
- 8 Zeta Avarikioti, Eleftherios Kokoris Kogias, Roger Wattenhofer, and Dionysis Zindros. Brick: Asynchronous incentive-compatible payment channels. In *International Conference on Financial Cryptography and Data Security*, 2021.
- 9 Zeta Avarikioti, Orfeas Stefanos Thyfronitis Litos, and Roger Wattenhofer. Cerberus channels: Incentivizing watchtowers for bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 346–366. Springer, 2020.
- 10 Zeta Avarikioti, Krzysztof Pietrzak, Iosif Salem, Stefan Schmid, Samarth Tiwari, and Michelle Yeo. Hide and seek: Privacy-preserving rebalancing on payment channel networks. In *Proc. Financial Cryptography and Data Security (FC)*, 2022.
- 11 Yossi Azar, Edith Cohen, Amos Fiat, Haim Kaplan, and Harald Racke. Optimal oblivious routing in polynomial time. In *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing, STOC '03*, pages 383–388, New York, NY, USA, 2003. Association for Computing Machinery. doi:10.1145/780542.780599.
- 12 Moshe Babaioff, Shahar Dobzinski, Sigal Oren, and Aviv Zohar. On bitcoin and red balloons. In *EC*, 2012. doi:10.1145/2229012.2229022.
- 13 Christian Badertscher, Juan Garay, Ueli Maurer, Daniel Tschudi, and Vassilis Zikas. But why does it work? a rational protocol design treatment of bitcoin. In *Eurocrypt*, 2018. doi:10.1007/978-3-319-78375-8\_2.
- 14 Burak Can, Jens Leth Hougaard, and Mohsen Pourpouneh. On reward sharing in blockchain mining pools. *Games and Economic Behavior*, 136:274–298, 2022. doi:10.1016/j.geb.2022.10.002.
- 15 Miles Carlsten, Harry Kalodner, S Matthew Weinberg, and Arvind Narayanan. On the instability of bitcoin without the block reward. In *CCS*, 2016. doi:10.1145/2976749.2978408.

- 16 Xi Chen, Christos Papadimitriou, and Tim Roughgarden. An axiomatic approach to block rewards. In *AFT*, 2019. doi:10.1145/3318041.3355470.
- 17 José R Correa, Andreas S Schulz, and Nicolás E Stier-Moses. Selfish routing in capacitated networks. *Mathematics of Operations Research*, 33(4):961–976, 2008.
- 18 Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, Dawn Song, and Roger Wattenhofer. On scaling decentralized blockchains. In *International Conference on Financial Cryptography and Data Security*, pages 106–125. Springer, 2016.
- 19 Oğuzhan Ersoy, Stefanie Roos, and Zekeriya Erkin. How to profit from payments channels. In *FC*, 2020. doi:10.1007/978-3-030-51280-4\_16.
- 20 Ittay Eyal. The miner’s dilemma. In *IEEE S&P*, 2015. doi:10.1109/SP.2015.13.
- 21 Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *International conference on financial cryptography and data security*, pages 436–454. Springer, 2014.
- 22 Joan Feigenbaum, Christos H Papadimitriou, Rahul Sami, and Scott Shenker. A bgp-based mechanism for lowest-cost routing. *Distributed Computing*, 18(1):61–72, 2005.
- 23 Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nikolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 51–68. ACM, 2017.
- 24 Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick McCorry, and Arthur Gervais. Sok: Layer-two blockchain protocols. In *International Conference on Financial Cryptography and Data Security*, pages 201–226. Springer, 2020.
- 25 Abdelatif Hafid, Abdelhakim Senhaji Hafid, and Mustapha Samih. Scaling blockchains: A comprehensive survey. *IEEE Access*, 8:125244–125262, 2020. doi:10.1109/ACCESS.2020.3007251.
- 26 Rami Khalil and Arthur Gervais. Revive: Rebalancing off-blockchain payment networks. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 439–453, 2017.
- 27 Aggelos Kiyias, Elias Koutsoupias, Maria Kyropoulou, and Yiannis Tselekounis. Blockchain mining games. In *EC*, 2016. doi:10.1145/2940716.2940773.
- 28 Y.A. Korilis, A.A. Lazar, and A. Orda. Achieving network optima using stackelberg routing strategies. *IEEE/ACM Transactions on Networking*, 5(1):161–173, 1997. doi:10.1109/90.554730.
- 29 Yujin Kwon, Dohyun Kim, Yunmok Son, Eugene Vasserman, and Yongdae Kim. Be selfish and avoid dilemmas: Fork after withholding (faw) attacks on bitcoin. In *CCS*, 2017. doi:10.1145/3133956.3134019.
- 30 Joshua Lind, Oded Naor, Ittay Eyal, Florian Kelbert, Emin Gün Sirer, and Peter R. Pietzuch. Teechain: a secure payment network with asynchronous blockchain access. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, pages 63–79, 2019.
- 31 Giulio Malavolta, Pedro Moreno-Sanchez, Aniket Kate, and Matteo Maffei. Silentwhispers: Enforcing security and privacy in decentralized credit networks. In *24th Annual Network and Distributed System Security Symposium*, 2017.
- 32 Patrick McCorry, Surya Bakshi, Iddo Bentov, Sarah Meiklejohn, and Andrew Miller. Pisa: Arbitration outsourcing for state channels. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, pages 16–30. ACM, 2019.
- 33 Roger B Myerson and Mark A Satterthwaite. Efficient mechanisms for bilateral trading. *Journal of Economic Theory*, 29(2):265–281, 1983. doi:10.1016/0022-0531(83)90048-0.
- 34 Tejaswi Nadahalli, Majid Khabbazi, and Roger Wattenhofer. Timelocked bribing. In *Financial Cryptography and Data Security*, Lecture Notes in Computer Science, 2021.
- 35 Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.
- 36 Rafael Pass and Elaine Shi. Fruitchains: A fair blockchain. In *PODC*, 2017. doi:10.1145/3087801.3087809.

- 37 Krzysztof Pietrzak, Iosif Salem, Stefan Schmid, and Michelle Yeo. Lightpir: Privacy-preserving route discovery for payment channel networks. In Zheng Yan, Gareth Tyson, and Dimitrios Koutsonikolas, editors, *IFIP Networking Conference, IFIP Networking 2021, Espoo and Helsinki, Finland, June 21-24, 2021*, pages 1–9. IEEE, 2021. doi:10.23919/IFIPNetworking52078.2021.9472205.
- 38 Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments, 2015.
- 39 Stefanie Roos, Pedro Moreno-Sanchez, Aniket Kate, and Ian Goldberg. Settling payments fast and private: Efficient decentralized routing for path-based transactions. *arXiv preprint arXiv:1709.05748*, 2017.
- 40 Tim Roughgarden. Selfish routing and the price of anarchy. *MIT press*, 2005.
- 41 Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in bitcoin. In *FC*, 2016. doi:10.1007/978-3-662-54970-4\_30.
- 42 Okke Schrijvers, Joseph Bonneau, Dan Boneh, and Tim Roughgarden. Incentive compatibility of bitcoin mining pool reward functions. In *FC*, 2016. doi:10.1007/978-3-662-54970-4\_28.
- 43 Vibhaalakshmi Sivaraman, Shaileshh Bojja Venkatakrishnan, Kathleen Ruan, Parimarjan Negi, Lei Yang, Radhika Mittal, Giulia Fanti, and Mohammad Alizadeh. High throughput cryptocurrency routing in payment channel networks. In *17th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 20)*, pages 777–796, 2020.
- 44 Jason Teutsch, Sanjay Jain, and Prateek Saxena. When cryptocurrencies mine their own business. In *FC*, 2016. doi:10.1007/978-3-662-54970-4\_29.
- 45 Samarth Tiwari, Michelle Yeo, Zeta Avarikioti, Iosif Salem, Krzysztof Pietrzak, and Stefan Schmid. Wiser: Increasing throughput in payment channel networks with transaction aggregation. *CoRR*, abs/2205.11597, 2022. doi:10.48550/arXiv.2205.11597.
- 46 Somanath Tripathy and Susil Kumar Mohanty. Mappcn: Multi-hop anonymous and privacy-preserving payment channel network. In *International Conference on Financial Cryptography and Data Security*, pages 481–495. Springer, 2020.
- 47 Itay Tsabary, Matan Yechieli, and Ittay Eyal. MAD-HTLC: because HTLC is crazy-cheap to attack. *IEEE S&P*, 2021. URL: <https://arxiv.org/abs/2006.12031>.
- 48 Yuup Van Engelshoven and Stefanie Roos. The merchant: Avoiding payment channel depletion through incentives. In *2021 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*, pages 59–68. IEEE, 2021.
- 49 Sarisht Wadhwa, Jannis Stoeter, Fan Zhang, and Kartik Nayak. He-HTLC: Revisiting Incentives in HTLC. In *30th Annual Network and Distributed System Security Symposium, NDSS 2023, San Diego, California, USA, February 27 - March 3, 2023*. The Internet Society, 2023. URL: <https://www.ndss-symposium.org/ndss-paper/he-htlc-revisiting-incentives-in-htlc/>.
- 50 Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 2014.

# SoK: Zero-Knowledge Range Proofs

Miranda Christ  

Columbia University, New York, NY, USA

Foteini Baldimtsi  

Mysten Labs, Palo Alto, CA, USA

George Mason University, Fairfax, VA, USA

Konstantinos Kryptos Chalkias  

Mysten Labs, Palo Alto, CA, USA

Deepak Maram  

Mysten Labs, Palo Alto, CA, USA

Arnab Roy  

Mysten Labs, Palo Alto, CA, USA

Joy Wang  

Mysten Labs, Palo Alto, CA, USA

---

## Abstract

Zero-knowledge range proofs (ZKRPs) allow a prover to convince a verifier that a secret value lies in a given interval. ZKRPs have numerous applications: from anonymous credentials and auctions, to confidential transactions in cryptocurrencies. At the same time, a plethora of ZKRP constructions exist in the literature, each with its own trade-offs. In this work, we systematize the knowledge around ZKRPs. We create a classification of existing constructions based on the underlying building techniques, and we summarize their properties. We provide comparisons between schemes both in terms of properties as well as efficiency levels, and construct a guideline to assist in the selection of an appropriate ZKRP for different application requirements. Finally, we discuss a number of interesting open research problems.

**2012 ACM Subject Classification** Security and privacy → Cryptography

**Keywords and phrases** Range proofs, zero knowledge

**Digital Object Identifier** 10.4230/LIPIcs.AFT.2024.14

**Related Version** *Full Version:* <https://eprint.iacr.org/2024/430> [27]

## 1 Introduction

Zero-knowledge (ZK) proofs have received much attention in recent years, with an abundance of generic protocols being developed using various assumptions and techniques. Although these generic protocols are becoming very efficient and easier to implement, there are still cases for specific types of statements, where customized ZK protocols are preferable.

Zero-knowledge range proofs (ZKRPs) are a subclass of zero-knowledge proofs that proves a structured kind of set membership. A ZKRP allows a prover to convince a verifier that a secret, committed value lies in a given (integer) interval. Brickell et al. [15] introduced the first type of zero-knowledge range proof as a building block in a protocol for revealing a secret discrete logarithm bit-by-bit. Since their introduction, ZKRPs have been used in various applications such as private e-cash protocols [25] (to verify non-negative transaction amounts), anonymous credentials systems [22, 6, 24] (to prove that a secret credential attribute, i.e. user age, falls in a specific range) as well as private voting [43], auctions [3] and privacy preserving federated learning [7] and so on. Additionally, ZKRPs are often used as building blocks for more complex cryptographic schemes. For instance, they have been used to



© Miranda Christ, Foteini Baldimtsi, Konstantinos Kryptos Chalkias, Deepak Maram, Arnab Roy, and Joy Wang;

licensed under Creative Commons License CC-BY 4.0

6th Conference on Advances in Financial Technologies (AFT 2024).

Editors: Rainer Böhme and Lucianna Kiffer; Article No. 14; pp. 14:1–14:25

Leibniz International Proceedings in Informatics



Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

construct ZK proofs of non-membership [55] and ZK proofs of certain polynomial relations over the integers [23, 21], and they have also been used to prove well-formedness of RLWE ciphertexts [35, 52] and well-formedness of shares in secret-sharing schemes [45, 42].

At the same time, with the rise of decentralized systems and cryptocurrencies, range proofs have received increased attention due to their use in mechanisms that preserve the privacy of transactions posted on the blockchain. For instance, ZKRPs are a key ingredient in confidential transactions [59, 16, 64] – which hide the amount of each transaction posted on the blockchain. The transaction amounts are stored in a committed fashion, and to ensure validity of the transaction the sender must prove that the sum of the output amounts does not exceed the sum of the input amounts. For this check to be sound, the sender must also prove that all output amounts are positive (else an adversarial sender could commit to negative output amounts and create coins out of thin air). For commitments in a group, such as Pedersen commitments, this positivity check also involves showing that the committed value is much less than the order of the group. This check essentially amounts to showing that the committed value is in some integer range  $[0, 2^k - 1]$  and is done via a ZKRP. Additionally, ZKRPs are heavily used in protocols for blockchain auditing and solvency solutions [32, 20, 48, 26] to show that transactions or reserves of an organization satisfy certain policies.

This increased interest in ZKRPs has also resulted in a growing number of proposed constructions with different characteristics and properties. With numerous ZKRP constructions available, selecting the suitable scheme for a specific application can be challenging. The goals of this SoK are to organize the space on the various techniques used to construct range proofs, compare their properties in a systematic way, identify open research questions, and provide a guideline to select the appropriate protocol for each type of application.

**Our contributions and organization.** We start by defining the necessary background on cryptographic schemes and computational assumptions in Section 2. In Section 3, we provide a taxonomy of general approaches used in the construction of zero-knowledge range proofs. Concretely, we identify three underlying methods used in the constructions of known ZKRP schemes: (a) square decomposition, (b) binary/ $n$ -ary decomposition and (c) hash-chain approach. We describe each method in detail, and for  $n$ -ary decomposition we present an abstraction that allows us to synthesize the several techniques used. Our abstraction is of independent interest, and could potentially lead to new insights. Then, in Section 4, we collect the set of properties beyond the standard soundness and zero-knowledge that are desirable in certain application scenarios of ZKRPs, such as aggregation, transparent setup and efficiency considerations. In Sections 5-7 we classify all known (to the best of our knowledge) ZKRP constructions under the three methods we identified in Section 3. For each method, we provide an analytical list of known protocols and we compare all protocols based on the desirable properties listed in Section 4. In Section 8, we provide a guideline for how to select the best type of ZKRP construction based on the desired properties and then in Section 9, we report storage and computation (verifier/prover time) costs of the most popular ZKRP constructions using existing and new benchmarks. (We provide a more detailed list of known ZKRP applications in the full version of the paper [27].) Finally, we identify a series of research gaps relevant to ZKRP which we believe can serve as a starting point for future research works in Section 10.

**Comparison with prior work.** We compare our paper with the previous survey of range proofs by Morais, Koens, van Wijk, and Koren [60]. The technical portion of [60] focuses largely on Boudot’s four-square decomposition construction [14], the signature-based construction of

CCs [22], and Bulletproofs [17]. It omits or does not go into detail on many other works, such as the line of code-based constructions, the newer and more efficient square-decomposition constructions, the polynomial commitment-based constructions, the hash chain constructions and lattice based constructions. In particular, many of the most efficient schemes such as Sharp [29] and BFGW [11] are not covered in their survey. Their work also provides a comparison only of the three schemes that it focuses on. Our SoK is significantly more comprehensive, and here is a summary of how our work goes beyond [60]. First, to the best of our knowledge, we provide a complete description of techniques and schemes in the ZKRP category and we extensively compare all such schemes based on their techniques, assumptions, and other properties. Additionally, we observe a useful abstraction for binary decomposition-based range proofs, breaking such proofs into two components, and presenting the techniques used for each of these components. An important aspect for our work, especially for practitioners who will use our SoK to determine the most suitable ZKRP for their application, is that we provide new benchmarks and assemble existing benchmarks for easier comparison. We plan to open-source the code used for our benchmarks. Finally, we include open questions and research gaps, and a flowchart to help identify the most appropriate range proof construction family for various applications.

## 2 Preliminaries

We use boldface, like  $\mathbf{a} = (a_1, \dots, a_n)$ , to denote a vector, and we let  $\text{wt}(\mathbf{a})$  denote its Hamming weight. We use  $\circ$  to denote the Hadamard product, i.e.,  $\mathbf{a} \circ \mathbf{b} = (a_1 b_1, \dots, a_n b_n)$ . For a nonzero value  $a$ , we use  $\mathbf{a}^n$  to denote the vector  $(1, a, a^2, \dots, a^{n-1})$ . We let  $\mathbf{0}^n$  denote the length- $n$  vector  $(0, \dots, 0)$ . For two vectors  $\mathbf{x}, \mathbf{y}$ , we let  $\mathbf{x}^{\mathbf{y}} = (x_1^{y_1}, \dots, x_n^{y_n})$  denote element-wise exponentiation. We use  $\lambda$  to denote the security parameter,  $\mathcal{A}$  to denote an adversary,  $\mathbb{Z}$  to denote the integers, and  $\text{negl}(\cdot)$  to denote a negligible function. We use the word *efficient*, or *p.p.t.*, to mean probabilistic polynomial time.

► **Definition 1** (Commitment scheme [50]). *A commitment scheme is a pair of efficiently computable algorithms  $(\text{Gen}, \text{Com})$  where:*

- $\text{Gen}(1^\lambda)$  is an efficient randomized algorithm that outputs public parameters  $\mathbf{p}$ .
- $\text{Com}(\mathbf{p}, m, r)$  is an efficient deterministic function that takes as input the public parameters, a message  $m$ , and randomness  $r$ . It outputs a commitment to  $m$ .

*A commitment scheme must be binding and hiding, defined as follows:*

*A commitment scheme is binding if for all p.p.t. adversaries  $\mathcal{A}$ , it is infeasible to come up with two different messages corresponding to a given commitment.*

$$\Pr_{\mathbf{p} \leftarrow \text{Gen}(1^\lambda)} \left[ \begin{array}{l} (m_0, r_0), (m_1, r_1) \leftarrow \mathcal{A}(1^\lambda, \mathbf{p}) \wedge \\ (m_0 \neq m_1) \wedge \\ \text{Com}(\mathbf{p}, m_0, r_0) = \text{Com}(\mathbf{p}, m_1, r_1) \end{array} \right] = \text{negl}(\lambda)$$

*A commitment scheme is computationally (resp., statistically) hiding if for all p.p.t. (resp., unbounded) adversaries  $\mathcal{A}$ , it is infeasible to distinguish whether a commitment corresponds to any  $m_0$  or  $m_1$  known to  $\mathcal{A}$ . That is, for all  $m_0, m_1$ :*

$$\Pr_{r \leftarrow \mathbb{S}} \left[ \begin{array}{l} c \leftarrow \text{Com}(\mathbf{p}, m_0, r) \\ \mathcal{A}(1^\lambda, \mathbf{p}, c, m_0, m_1) = 1 \end{array} \right] \approx \Pr_{r \leftarrow \mathbb{S}} \left[ \begin{array}{l} c \leftarrow \text{Com}(\mathbf{p}, m_1, r) \\ \mathcal{A}(1^\lambda, \mathbf{p}, c, m_0, m_1) = 1 \end{array} \right]$$

A commitment scheme is *homomorphic* if

$$\text{Com}(\mathbf{p}, m_0, r_0) + \text{Com}(\mathbf{p}, m_1, r_1) = \text{Com}(\mathbf{p}, m_0 + m_1, r_0 + r_1).$$

## 14:4 SoK: Zero-Knowledge Range Proofs

Next we define zero-knowledge proof and non-interactive zero-knowledge proof (NIZK). Most of the ZKRPs in this SoK are in fact non-interactive. In the following sections, we will skip mention of the non-interactive aspect, unless not clear from context. We provide informal definitions next, while deferring the formal definition of NIZK and its properties to our full version [27].

► **Definition 2 (Zero-knowledge proof).** *Let  $L$  be a language in NP and  $R$  be a polynomially verifiable relation, such that  $x \in L \iff \exists w : R(x, w)$ . A zero-knowledge proof system for  $L$  is a tuple of efficient interactive algorithms (Prover, Verifier, Simulator), such that the following properties hold:*

- *Completeness.* Given  $(x, w) \in R$ , the honest execution of the Prover (given  $x, w$ ) and the Verifier (given only  $x$ ) result in the Verifier outputting 1.
- *Soundness.* Given  $x \notin L$ , a malicious Prover interacting with the Verifier can only make it output 1 with negligible probability.
- *Zero-Knowledge.* Given  $x \in L$ , the Simulator can produce an interaction transcript of an honest Prover with a (possibly) malicious Verifier, that is computationally indistinguishable from an actual execution transcript of the Prover with the Verifier. Note that the Simulator doesn't get  $w$ , while the Prover gets  $w$ .

A *non-interactive zero-knowledge (NIZK)* proof system is a zero-knowledge proof system, where the Prover, given  $(x, w)$  just sends one message  $\pi$  to the Verifier and the Verifier outputs 0/1 based on  $(x, \pi)$ . A NIZK has an additional setup algorithm *CRSGen*, which outputs a common reference string (CRS) used by all the proofs and verifications. Instead of a CRS, some NIZKs can also specify a random oracle. The Simulator algorithm is allowed to keep trapdoors about the CRS, or be able to simulate the random oracle.

A zero-knowledge *proof of knowledge* requires that an adversary which produces a valid proof for a statement also knows a valid witness. This is formally captured by requiring the existence of an extractor, which can run the adversary's code and produce the witness.

► **Definition 3 (ZKRP).** *A zero-knowledge range proof (ZKRP) is a zero-knowledge proof of knowledge for the following relation:*

$$RP_{\mathbf{p}} = \{((y, u, v), (m, r)) : y = \text{Com}(\mathbf{p}, m, r) \wedge u \leq m \leq v\}$$

where  $\mathbf{p}, y, u$ , and  $v$  are known to the verifier, and *Com* is some particular commitment scheme.

A question may arise since  $\mathbf{p}$  is hard-coded in the language definition: what if a malicious prover samples  $\mathbf{p}$  badly and thus renders the NIZK-soundness property vacuous? We note that most applications require both commitment security and NIZK-soundness. These requirements enforce that the attacker of the application's security cannot badly sample  $\mathbf{p}$ .

**Pedersen commitments.** Most range proofs use *Pedersen commitments* [63] as the underlying commitment scheme. Let  $\mathbb{G}$  be a cyclic group of prime order and  $g$  and  $h$  be generators of that group, where the discrete logarithm relationship between  $g$  and  $h$  is not known. The Pedersen commitment  $\text{Com}(x, r)$  for a value  $x \in \mathbb{G}$  with randomness  $r$  is  $g^x h^r$ .

Pedersen commitments are statistically hiding, and their binding property is based on the hardness of the discrete logarithm assumption.

► **Definition 4 (Discrete Logarithm Assumption).** *Let  $\mathbb{G}$  be a group of order  $p$  and let  $g$  be a generator of  $\mathbb{G}$ . A challenger samples a random  $x \leftarrow \mathbb{Z}_p$  and sends  $g^x$  to an adversary. The Discrete Logarithm Assumption states that it is infeasible for the adversary to output  $x$ , given  $(\mathbb{G}, g, g^x)$ .*



Apart from the Discrete Logarithm setting, we will also describe schemes based on the hardness of the RSA problem, as well as lattices.

► **Definition 5** (RSA Assumption). *A challenger samples primes  $p$  and  $q$  and sets  $N = pq$ . It picks a quantity  $e$  co-prime to  $\phi(N)$ , where  $\phi(N) = (p-1)(q-1)$  is Euler's totient function. Then it randomly samples  $z \leftarrow [1, N]$  and sends  $(N, e, z)$  to the adversary. The adversary outputs  $y$ . The RSA Assumption states that the probability of  $y^e = z \pmod{N}$  is negligible.*

► **Definition 6** (Strong RSA Assumption). *The Strong RSA Assumption states that the RSA problem is intractable even when the adversary is allowed to choose the public exponent  $e$  (for  $e \geq 3$ ).*

► **Definition 7** (SIS Assumption). *Let  $q, n, m \in \mathbb{Z}^+$ ,  $\beta \in \mathbb{R}^+$  be given, where  $\beta \ll q$ . A challenger samples a random matrix  $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$ . The SIS Assumption states that it is infeasible for an adversary to find a nonzero  $m$ -vector  $\mathbf{e}$ , such that  $\mathbf{A}\mathbf{e} = \mathbf{0} \pmod{q}$  and  $\|\mathbf{e}\|_2 \leq \beta$ .*

### 3 General Approaches

Efficient zero-knowledge range proofs typically use three classes of approaches: square decomposition,  $n$ -ary decomposition, and hash chains. We present these approaches below, then explore specific instantiations of these approaches in more detail in their respective sections. We also mention the approach of using generic zero-knowledge proofs.

We describe these approaches for proving that a committed value lies in a range of the form  $[0, n^k - 1]$ , or that a committed value is positive in the case of square decomposition. Most works consider ranges of this form, which may seem at a first glance to be a relaxed version of the problem. However, when the commitments used are homomorphic, it turns out to be sufficient for constructing more general range proofs with only a small amount of work to translate.

Assume that we have the ability to prove that any committed value is in the interval  $[0, n^k - 1]$ . To prove that  $z$  is in some interval  $[u, v]$ , one can show first that  $(z - u) \in [0, n^k - 1]$  and then that  $(v - z) \in [0, n^k - 1]$ . Thus,  $z \geq u$  and  $z \leq v$ . Certain constructions from integer commitments (e.g., CKLR [30]) can combine these checks into proving a single equation:  $(z - u)(v - z) \geq 0$ . It is easy to obtain commitments for  $(z - u)$  and  $(v - z)$  homomorphically, given a commitment to  $z$ . For non-homomorphic commitments, one can do this translation by creating a commitment  $c$  to  $z - u$ , proving in zero knowledge that  $c$  indeed commits to  $z - u$ , and performing this range proof with respect to  $c$ .

#### 3.1 Square decomposition

The square decomposition method involves writing the committed integer as a sum of squares in order to prove that it is positive. A common version of this method, the four-square decomposition method, uses Lagrange's four-square theorem. This theorem states that for any integer  $z \in \mathbb{Z}_{\geq 0}$ , there exist  $x_1, x_2, x_3, x_4 \in \mathbb{Z}$  such that

$$z = x_1^2 + x_2^2 + x_3^2 + x_4^2 \tag{1}$$

Thus, to prove that a committed value  $z$  is non-negative, it suffices to prove knowledge of  $x_1, \dots, x_4$  such that Equation 1 holds. However, it is crucial that the relation of Equation 1 holds *over the integers* since it may hold for a negative  $z$  if we are working in some group rather than over  $\mathbb{Z}$ . For example, in  $\mathbb{Z}_5$  it is possible that  $z = -1$  and  $0^2 + 1^2 + 2^2 + 2^2 = 9 = z \pmod{5}$ . To avoid such problems, this approach requires a special type of commitment called an integer commitment.

### Integer commitments

An *integer commitment scheme* is a commitment scheme where binding holds *over*  $\mathbb{Z}$ . That is, for all p.p.t. adversaries  $\mathcal{A}$ ,

$$\Pr_{\mathbf{p} \leftarrow \text{Gen}(1^\lambda)} \left[ \begin{array}{l} (m_0, r_0), (m_1, r_1) \leftarrow \mathcal{A}(1^\lambda, \mathbf{p}) \\ \wedge (m_0 \neq_{\mathbb{Z}} m_1) \\ \wedge \text{Com}(\mathbf{p}, m_0, r_0) = \text{Com}(\mathbf{p}, m_1, r_1) \end{array} \right] = \text{negl}(\lambda)$$

where  $m_0 \neq_{\mathbb{Z}} m_1$  denotes that  $m_0$  and  $m_1$  are not equal *over the integers*. Bounded integer commitments (used in [30]) satisfy the same binding property, but are weaker in that the message space is restricted to some bounded interval, e.g.,  $\{x \in \mathbb{Z} : |x| \leq B\}$ . For constructing range proofs, this boundedness is not an issue as long as the ranges in question are well within the bounds.

Pedersen commitments, for example, are not integer commitments as their message space is  $\mathbb{Z}_p$ , and any messages that are equivalent  $(\text{mod } p)$  result in the same commitment given the same randomness:  $g^m h^r = g^{m+p} h^r$  over a cyclic group of order  $p$ . This attack against binding fails if the order of the group is unknown, and indeed many integer commitment schemes (e.g., Fujisaki-Okamoto commitments, and constructions of [30, 29]) operate in groups of unknown order.

**Fujisaki-Okamoto commitments [40].** We recall an overview of FO commitments but refer the reader to [40] for details. FO commitments operate over a group of unknown order  $\mathbb{Z}_N^*$ .  $g$  and  $h$  are generators of large subgroups of  $\mathbb{Z}_N^*$ , whose relation is unknown. The commitment to  $x \in \mathbb{Z}$  is

$$\text{Com}_{FO}(\mathbf{p}, x, r) := g^x h^r$$

This commitment is computationally hiding when  $r$  is chosen uniformly in the interval  $[2^{-\lambda} \cdot N + 1, \dots, 2^\lambda \cdot N - 1]$ . Fujisaki-Okamoto commitments are computationally binding under the factoring assumption.

### 3.2 $n$ -ary decomposition

The  $n$ -ary decomposition method involves committing to the digits of the committed value  $z$  in some base  $n$ . For simplicity, assume for this explanation that we use base 2, although certain approaches can be generalized to other bases. Thus, if the prover wishes to show that  $z \in [0, 2^k - 1]$ , the prover writes  $z = z_0 \cdot 2^0 + z_1 \cdot 2^1 + \dots + z_{k-1} \cdot 2^{k-1}$  and generates commitments to  $z_0, \dots, z_{k-1}$ . The prover then shows that both of the following properties hold, which we present as predicates:

**Digit validity (DV( $\mathbf{z}$ )):**  $\text{DV}(\mathbf{z}) = 1$  if and only if  $z_i \in \{0, 1\}$  for all  $i \in [0, k - 1]$ .

**Representativeness (Rep( $\mathbf{z}, z$ )):**  $\text{Rep}(\mathbf{z}, z) = 1$  if and only if  $z = \sum_{i=0}^{k-1} z_i \cdot 2^i$ .

In terms of these predicates, the  $n$ -ary decomposition method proves membership in the following relation:

$$R_{\text{decomp}} = \{(\mathbf{p}, (c_1, c_2, n, k), (z, \mathbf{z}, r, \mathbf{r})) : c_1 = \text{Com}(\mathbf{p}, z, r) \\ \wedge c_2 = \text{Com}(\mathbf{p}, \mathbf{z}, \mathbf{r}) \wedge \text{DV}(\mathbf{z}) \wedge \text{Rep}(\mathbf{z}, z)\}$$

We note that here, we slightly abuse notation and use  $\text{Com}$  to commit to a vector  $\mathbf{z}$  with a vector of randomness  $\mathbf{r}$ .

There are (at least) four common tools used to show that the digits are valid for the desired base; i.e., for binary decomposition they all lie in  $\{0, 1\}$ . These tools include zero-knowledge *set membership arguments*, *product arguments*, *inner product arguments*, and *polynomial commitments*. These strategies are primarily applicable for base 2, with the exception of set membership, which easily extends to any arbitrary base.

### 3.2.1 Set membership

A set membership proof shows that a committed value lies in some publicly known set  $\Phi$ ; that is, it is a proof of knowledge for the following relation:

$$SM = \{(\mathbf{p}, (\Phi, y), (m, r)) : y = \text{Com}(\mathbf{p}, m, r) \wedge m \in \Phi\}$$

Although one could define a set membership proof with respect to a *private* committed set, in our application the set is determined by the publicly known base.

**Digit validity.** Set membership arguments are useful for instances of  $R_{\text{decomp}}$  where the commitment scheme used for  $\mathbf{z}$  commits to its components individually; that is,

$$c_2 = (\text{Com}(\mathbf{p}, z_0, r_0), \dots, \text{Com}(\mathbf{p}, z_{k-1}, r_{k-1}))$$

for some scheme  $\text{Com}$ . Then, one can show digit validity by providing a set membership proof for each element of  $c_2$ , with respect to the set  $\Phi = \{0, 1, \dots, n-1\}$ . However, such protocols require commitments and range proofs of length at least linear in  $k$ .

**Representativeness.** There is no general way to show representativeness using set membership proofs; schemes using this construction (e.g., [22]) rely on properties of the specific commitment scheme used.

### 3.2.2 Product arguments

A product argument is a proof system for showing that the product of two committed values  $a$  and  $b$  is some value  $c$ . Typically, this equality holds in the group underlying the commitment scheme. For example, for Pedersen commitments in a group of prime order  $p$ , this argument shows that  $ab \equiv c \pmod{p}$ . For integer commitments, we have the stronger property that this equality holds over the integers:  $ab = c$ .

**Digit validity.** Product arguments are useful for proving digit validity base 2, if as with set membership  $c_2$  consists of individual bit commitments. To show that a committed bit  $b$  is in  $\{0, 1\}$ , the prover can commit to a value  $a$  and prove that  $ab = 0$  and  $a + b = 1$ . Observe that if  $b \neq 0$ ,  $a$  must be 0 to satisfy the first equation. Then the second equation implies that  $b = 1$ . Thus,  $b$  must be 0 or 1. Furthermore, the prover can always find a satisfying  $a$ ; if  $b = 0$ ,  $a = 1$ , and if  $b = 1$ ,  $a = 0$ . Inner product arguments, which we present next, allow the prover to simultaneously show many product relations more efficiently.

**Representativeness.** As is the case with set membership proofs, product arguments are primarily useful for showing digital validity rather than representativeness.

### 3.2.3 Inner product arguments

An inner product argument (IPA) is a proof system for showing that the inner product of two committed vectors is some value. The inner product used in Bulletproofs [17] shows the following relation, using Pedersen commitments, where  $\mathbb{G}$  denotes a group of prime order:

$$\{(\mathbf{g}, \mathbf{h} \in \mathbb{G}^k, P \in \mathbb{G}, z \in \mathbb{Z}_p; \mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n) : P = \mathbf{g}^{\mathbf{a}} \mathbf{h}^{\mathbf{b}} \wedge z = \langle \mathbf{a}, \mathbf{b} \rangle\}$$

Here,  $P$  is a binding (but not hiding) commitment to the vectors  $\mathbf{a}$  and  $\mathbf{b}$ . Therefore, Bulletproofs introduces blinding factors to make this argument zero-knowledge. Bulletproofs also constructs an argument for the Hadamard product relation (i.e.,  $\mathbf{c} = \mathbf{a} \circ \mathbf{b}$ ) from their inner product argument, though we do not present the details here.

**Digit validity.** A useful fact used when constructing zero-knowledge range proofs from inner product arguments is that with overwhelming probability, the inner product of a nonzero vector  $\mathbf{a}$  and a random vector  $\mathbf{b}$  is nonzero. Thus, the prover can convince the verifier that  $\mathbf{a}$  is  $\mathbf{0}^k$  by showing that its inner product with a random challenge vector is 0. Using the same idea as for product arguments, the prover can commit to the binary representation of the given value as a vector  $\mathbf{z}$ , then use an inner product argument to show simultaneously that *all* components of this vector are indeed bits. That is, the prover commits to a vector  $\mathbf{z}' = \mathbf{1}^k - \mathbf{z}$ , and shows for a random  $\mathbf{x}$  that:

$$\langle \mathbf{z}' - (\mathbf{1}^k - \mathbf{z}), \mathbf{x} \rangle = 0 \text{ and } \mathbf{z}' \circ \mathbf{z} = \mathbf{0}^k$$

The lattice-based scheme [4] uses this approach as well.

**Representativeness.** Although we presented an inner product relation where the value  $z$  is a public input, many inner product arguments, such as that of Bulletproofs, work also when  $z$  is secret and the public input includes only a commitment to  $z$ . One shows representativeness by a single application of this inner product argument, showing  $\langle \mathbf{z}, \mathbf{2}^k \rangle = z$ .

Bulletproofs combines some of these checks for greater efficiency and uses blinding factors to make their argument zero-knowledge.

### 3.2.4 Polynomial commitments

A polynomial commitment scheme allows a prover to commit to a polynomial  $p(\cdot)$  over a finite field  $\mathbb{F}_p$ , such that a verifier can query a point  $x$  to the prover, which can respond with  $p(x)$  and a proof  $\pi$  that this evaluation is correct. The scheme should be hiding in that the commitment reveals nothing about the polynomial, and the evaluation proofs reveal no extra information beyond the evaluations themselves. Polynomial commitments are binding in that it is computationally infeasible to produce a verifying proof for an incorrect evaluation of the committed polynomial. A useful property of polynomial commitments is that it is easy for a prover to show that a committed polynomial is identically zero, by providing a proof that its evaluation at a random point is zero. By binding and the Schwartz-Zippel lemma, this occurs with only negligible probability if the polynomial is nonzero.

The following approach, which we describe at a high level, was introduced in BFGW [11] and is detailed nicely in [67]. Suppose that we are given a commitment to  $z$  in the form of a polynomial commitment to  $f$  such that  $f(1) = z$ . In constructing a range proof for  $z \in [0, 2^k - 1]$ , it is useful to work over a subgroup  $H = \{1, \omega, \omega^2, \dots, \omega^{k-1}\}$  and use polynomials whose evaluations over  $H$  encode the binary representation of  $z$ . That is, the prover computes a polynomial  $g$  such that:

$$\begin{aligned} g(\omega^{k-1}) &= z_{k-1} \\ g(\omega^i) &= 2g(\omega^{i+1}) + z_i \quad \forall i \in \{0, \dots, k-2\} \end{aligned}$$

Another useful property of polynomial commitments is that one can show that a polynomial  $g(X)$  is zero on all of  $H$  by committing to a related polynomial  $g'(X)$  and proving that  $g'(X)$  is identically zero over  $\mathbb{F}_p$ .

**Digit validity.** The prover shows that the following two polynomials are zero over all of  $H$ :

$$\begin{aligned} w_2 &= g \cdot (1 - g)(X - 1)(X - \omega) \cdots (X - \omega^{k-2}) \\ w_3 &= [g(X) - 2g(X\omega)] \cdot [1 - g(X) + 2g(X\omega)] \cdot (X - \omega^{k-1}) \end{aligned}$$

$w_2$  has zeros at  $1, \omega, \dots, \omega^{k-2}$  by construction. It is zero at  $\omega^{k-1}$  if and only if  $g(\omega^{k-1}) \in \{0, 1\}$ . For  $w_3$ , observe that  $g(X) - 2g(X\omega)$  is exactly  $z_i$  when evaluated at  $\omega^i$ . Therefore,  $w_3$  is zero at  $\{1, \dots, \omega^{k-2}\}$  if and only if  $z_i \in \{0, 1\}$ .

**Representativeness.** The prover shows that the following polynomial is zero over all of  $H$ :

$$w_1 = (g - f)(X - \omega)(X - \omega^2) \cdots (X - \omega^{k-1})$$

As [11] notes, this approach can be instantiated with any polynomial commitment scheme that is hiding, binding, and additively homomorphic.

### 3.3 Hash chains

Hash chains can be used to prove that a committed value is at least some threshold. In the hash chain approach, a commitment to a value  $z$  is  $C_z = H^z(r)$ , the output of a hash function applied  $z$  times to a random  $r$ . The proof that  $z$  exceeds some threshold  $t$  is  $\pi = H^{z-t}(r)$ . A verifier can check that  $H^t(\pi) = C_z$ ; if  $z < t$ , then  $z - t$  is negative and it is infeasible for a cheating prover to compute a preimage of  $r$  under  $H$ .

This simple hash chain requires prover and verifier time that is exponential in  $k$  for ranges  $[0, 2^k - 1]$ . However, using decomposition techniques, HashWires [24] constructs a hash chain-based range proof requiring only  $O(k)$  work.

### 3.4 Generic zero-knowledge

There are many efficient generic zk-SNARKs, such as [44, 41, 9, 18]. These proof systems can be used to construct range proofs. However, because they are generic and do not leverage the structure of the range proof relation, they are less efficient than the tailored range proofs we explore. In Section 9, we include efficiency benchmarks for Groth16 [44], one of the most popular generic zk-SNARKs used in practice.

It is worth noting that practical benefits may outweigh these efficiency losses. In particular, because of their wide-ranging applications, generic zk-SNARKs offer convenient, well-engineered, and optimized libraries. For example, we used Circom [8] and rapidsnark [47] for our Groth16 benchmarks. Even so, the prover and verifier times for Groth16 are roughly an order of magnitude larger than the more tailored range proofs. Furthermore, if range proofs are required in a larger system that already uses a generic zk-SNARK elsewhere, using this zk-SNARK for the range proof as well may be practically convenient.

## 4 Desirable properties

All zero-knowledge range proofs must satisfy the standard notions of soundness, completeness, and zero knowledge. All ZKRPs that we cover in this SoK are non-interactive. In this section, we discuss some additional nice features that might be desirable in some settings.

**Efficiency.** Unsurprisingly, it is desirable for ZKRPs to be efficient. In blockchain applications, where a transactor must pay for the storage cost and the amount of computation done by validators, it is especially important to minimize proof size and verifier time. The proof size should be at most linear in  $k$  for intervals  $[0, 2^k - 1]$ , and several schemes offer even constant-sized proofs. Though proof size and verifier time are often priorities, prover time also should not be prohibitively large. Since it is hard to directly compare efficiency of the constructions we discuss in Sections 5 - 7 (even in the asymptotic setting the different parameters make one-to-one comparison very hard), we instead opt to provide a *concrete* comparison of some of the most popular ZKRPs in Section 9.

**Transparent setup.** Some range proofs require public parameters that are generated using secret randomness. It is crucial for the security of the proofs that this randomness is not known to the prover. For example, several square decomposition range proofs use RSA-based integer commitments, which require an RSA modulus. Importantly, this modulus  $N$  must be generated in such a way that no party know the factorization of  $N = pq$ . Similarly, BFGW [11] instantiated with KZG commitments [49] requires a powers-of-tau common reference string, which consists of a series of values  $g^{\tau^i}$ , where no party knows  $\tau$ . Protocols that require secrecy of the randomness used in parameter generation are said to require *trusted setup*. Trusted setup does not necessarily require a trusted party, as many trusted setup procedures can be conducted by distributed multi-party protocols. Such protocols (often called ceremonies) exist for many common trusted setup procedures, such as generation of RSA moduli and powers-of-tau [39, 12, 62].

Ideally, protocols should have a *transparent* setup procedure that does not require secret randomness. For example, the parameters could be generated by applying a hash function to some public randomness, e.g., to generate a random group element or random matrix.

Note that trusted setup is different from having a trusted issuer responsible for distributing the proper commitments to users, e.g., a Pedersen commitment corresponding to that user's account balance. Any protocol needs to assume that the prover and verifier agree on the commitment at hand.

**Aggregation.** Aggregation allows multiple range proofs to be compressed into a single succinct proof. That is, a single prover holding  $m$  commitments to values in the same range  $[0, 2^k - 1]$  can efficiently generate a short aggregate proof  $\pi$  proving all of these range statements simultaneously. For this aggregation property to be nontrivial,  $\pi$  should be shorter than the concatenation of  $\pi_1, \dots, \pi_m$ . For example, for Bulletproofs, Bulletproofs+, and Bulletproofs++ [17, 28, 36], the aggregate proof for  $m$  values in  $[0, 2^k - 1]$  consists of only  $O(\log(m \cdot k))$  group elements. As the concatenation of  $m$  proofs would require  $O(m \cdot \log(k))$  group elements, aggregation results in considerable space savings.

In the notion of aggregation considered so far, a single prover knows the openings of all commitments that are being aggregated. A stronger notion of *multi-prover* aggregation allows one to combine range proofs generated by multiple provers, who wish to hide their openings from one another. Bulletproofs enables such aggregation via an MPC protocol run by the parties holding the commitments [17]. Multi-prover aggregation is harder to achieve, and is less well studied than single-prover aggregation.

Aggregation is especially useful for confidential transactions, where minimizing the amount of space used on-chain decreases gas costs. Since range proofs are used to show non-negativity, all range proofs typically prove membership in the same interval.

■ **Table 1** Properties of square decomposition-based range proofs.

Square Decomposition-Based Range Proofs					
Scheme	Commitment Scheme	Assumptions	Transp. Setup	Proof Aggregation	Batched Ver.
Boudot [14]	F-O [40]	Strong RSA	N	N	N
Lipmaa [56]	RDF integer comm.*	Strong RSA	N	N	N
Groth [43]**	RDF integer comm.*	Strong RSA	N	N	N
CKLR [30]	Ped***	DLOG (optionally DSLE)	N	N	N
CKLR [30]	ElGamal variant [30]	DXDH, ORD	Y (class groups)	N	N
Sharp <sub>CS</sub> , Sharp <sub>SO</sub> <sup>PO</sup> [29]†	Pedersen	DLOG, SEI	Y	Y	Y
Sharp <sub>HO</sub> [29]†	Pedersen	1/2-fROOT	N (RSA), Y (class groups)	Y	Y

- An extension of the Dámgaard-Fujisaki commitment [33] that [56] constructs.
- \*\*[43] is not exactly a new scheme; its contribution is observing a trick that can be applied to make [56] more efficient. Integers of a certain form can be written as a sum of *three* squares, and one can quickly find this decomposition.
- \*\*\* A bounded integer commitment scheme based on Pedersen commitments.
- † Sharp is only a *relaxed* range proof and not sufficient for all applications. [29] has a thorough discussion; it is sufficient for anonymous credentials and can be used for some but not all proofs in anonymous transactions, with some modifications. Sharp<sub>HO</sub> refers to a scheme where Sharp<sub>CS</sub> or Sharp<sub>SO</sub><sup>PO</sup> is modified using an additional commitment requiring an RSA group or class group in order to achieve improved soundness.

**Batch verification.** A related property is batch verification, where there exists a process for verifying many proofs together that is more efficient than verifying each proof individually. Batch verification is especially useful in blockchain applications, where a block proposer can aggregate the range proofs for its block and other validators can batch verify this proof more efficiently. Bulletproofs provides batch verification [17], using an observation that verifying many statements of the form  $g^x = 1$  can be done by carefully combining them into a single equation requiring fewer exponentiations.

Aggregated range proofs often naturally enable batch verification, as some of the work is effectively done by the aggregator. However, neither aggregation nor batch verification in general implies the other.

**Compatibility with homomorphic commitments.** A commitment scheme Com is homomorphic if  $\text{Com}(m_0, r_0) + \text{Com}(m_1, r_1) = \text{Com}(m_0 + m_1, r_0 + r_1)$ . It is convenient for applications such as confidential transactions for the underlying commitments to be homomorphic; in particular, homomorphism makes it easier to prove that the sum of transaction output amounts is at least the sum of input amounts.

Most ZKRPs use Pedersen commitments, which are homomorphic. Some exceptions are HashWires [24] and various lattice-based constructions such as KTX [51], which often achieve weaker homomorphism.

## 5 Square Decomposition Constructions

Recall that the square decomposition method involves writing the committed value as the sum of four squares and proving that this equality holds over the integers. Integer commitments, which were discussed in greater detail in Section 3, are a useful tool here. (Recall: An integer commitment scheme is a commitment scheme for which binding holds over the integers: it is computationally infeasible for an adversary to find messages  $m_0, m_1$  and randomness  $r_0, r_1$  such that  $\text{Com}(m_0, r_0) = \text{Com}(m_1, r_1)$ , where  $m_0 \neq m_1$  over  $\mathbb{Z}$ .) Below we discuss different approaches in this class and also compare them in Table 1. Our comparison is done in terms of the properties discussed in Section 4 except efficiency which as explained above, will be treated separately in Section 9.

Approaches in this class combine integer commitment schemes with a way to prove in zero knowledge that, given a commitments  $\text{Com}_x$  and  $\text{Com}_y$ , the committed values satisfy  $x^2 = y$ . This implies that  $y$  is non-negative. One can generalize this argument to work not just for squares  $y$ , but for all non-negative integers.

Boudot [14] introduced the approach of proving that a committed value is positive by representing an arbitrary integer as a sum of squares (although not four squares). It uses Fujisaki-Okamoto commitments [40], which require a group of unknown order such as an RSA group. Damgård-Fujisaki commitments [33] are slightly more efficient integer commitments used in subsequent work [56] which refined Boudot's idea and used Lagrange's four square theorem [46, Theorem 369] (which states that every integer can be written as the sum of the squares of four integers). In order to do so, it also introduced an efficient algorithm for finding this four-square decomposition. [43] similarly followed this approach and improved its efficiency by observing that  $x$ 's of a certain form can be written as the sum of only three squares rather than four. [31] further improved the efficiency and showed that the RSA assumption (rather than the strong RSA assumption, as previously shown) is sufficient to show the security of Damgård-Fujisaki commitments.

The integer commitments used by all of [14, 56, 43] require a group modulus whose factorization is unknown, and therefore require trusted setup. A newer line of work [30, 29] develops new integer commitment schemes, some of which do not require a trusted setup. These schemes also yield much better efficiency, though Bulletproofs and subsequent binary-decomposition-based proofs are still more efficient in practice due to compatibility with available optimized libraries.

CKLR [30] build a *bounded integer commitment* by modifying Pedersen commitments; their scheme essentially enforces that the Pedersen commitment can only be opened to values within some bounded range. They then use this bounded integer commitment to construct their ZKRP following the square decomposition approach. However, their commitment scheme operates over *rationals* rather than integers; while honest openers round these rationals to integers, malicious openers may open to rationals instead which can be problematic for some applications and results in a relaxed notion of soundness. Sharp [29] improves upon CKLR in several ways. In addition to improving over the efficiency of CKLR, Sharp is compatible with standard Pedersen commitments. This is because Sharp effectively moves CKLR's modifications of Pedersen commitments to the *proof* rather than modifying the commitment itself. Two variants of Sharp ( $\text{Sharp}_{\text{GS}}$ ,  $\text{Sharp}_{\text{SO}}^{\text{PO}}$ ), like CKLR, achieve a relaxed notion of soundness. However, they show how to boost soundness by adding an additional commitment using a hidden-order group such as an RSA group or class group; the resulting variants  $\text{Sharp}_{\text{HO}}$  achieve standard soundness but require longer proofs. The RSA version also requires a trusted setup. Class groups are hidden-order groups that can be instantiated without a trusted setup, though they are less well-supported than RSA groups from an engineering standpoint. Finally, Sharp improves over CKLR by also offering batching capabilities.

## 6 Binary Decomposition Constructions

CCs [22] introduced the  $n$ -ary decomposition paradigm to zero-knowledge range proofs. CCs [22] operates over Pedersen commitments and constructs a zero-knowledge set membership protocol by having the verifier publish a signature of each element in the set. The prover then shows in zero knowledge that it knows a signature of its committed value  $x$  under the verifier's secret key; by unforgeability this is only possible if the value is in this set. By choosing this set to be  $\{0, \dots, n - 1\}$  for base  $n$ , the prover can commit to the



digits of  $x$  and prove that they are valid digits under that base. CCs then uses properties of Pedersen commitments to show that the committed digits indeed represent  $x$ . The size of the proof is linear in  $\log_n 2^k$ , where  $n$  is the base used and the range is of size  $2^k$ . By optimizing the choice of the base  $n$ , this results in a slightly sublinear (in  $k$ ) proof size for a range  $[0, 2^k - 1]$ . This scheme requires a trusted setup for the signature generation, and it does not offer aggregation.

Subsequent constructions (which we call “Bulletproofs-style” and detail in the next subsection) improve on the efficiency of CCs to avoid this near-linear dependence on  $k$ . They use inner product arguments or polynomial commitment schemes in clever ways to avoid showing individually that each bit is in  $\{0, 1\}$ ; instead, they are able to roll all of these checks into a shorter proof.

There are also several newer lattice- and code-based constructions that use binary decomposition, such as [61, 4]. While these schemes are less efficient and have very large proofs, their main merit is that they are plausibly post-quantum secure. Additionally, they do offer transparent setup. Developing more practical lattice-based ZKRP is an interesting research direction as we discuss in more details in Section 10.

When surveying binary decomposition constructions, we separate them into two categories: Bulletproofs-style constructions, which are very practical; and lattice-based constructions, which are primarily of theoretical interest. We provide an overview of all the bulletproof style constructions described below in Table 2.

## 6.1 Bulletproofs-Style Constructions

Bulletproofs [17], arguably considered the state-of-the-art range proof scheme, uses the binary decomposition technique.

Bulletproofs combines the binary decomposition technique with an inner product argument to enable the prover to send only  $O(\log k)$  elements. Bulletproofs improves and uses their improvement of an inner product argument (IPA) of [13] where the prover sends only  $O(\log k)$  group elements for an IPA over length- $k$  vectors. The key idea in Bulletproofs is that the prover can use this IPA to execute the binary decomposition approach more efficiently; we give intuition for this idea here.

We write  $x = a_0 \cdot 2^0 + a_1 \cdot 2^1 + \dots + a_{k-1} \cdot 2^{k-1}$  and let  $\mathbf{a}_L = [a_0, a_1, \dots, a_{k-1}]$ . We let  $\mathbf{2}^k := [2^0, 2^1, \dots, 2^{k-1}]$ . The prover shows that it knows a vector  $\mathbf{a}_R$  such that:

$$(1) \mathbf{a}_L \circ \mathbf{a}_R = \mathbf{0}^k, \quad (2) \mathbf{a}_L - \mathbf{a}_R = \mathbf{1}^k, \quad (3) \mathbf{a}_L \circ \mathbf{2}^k = x$$

Conditions (1) and (2) show that each component of  $\mathbf{a}_L$  is in  $\{0, 1\}$ , using the standard inner product strategy described in Section 3. Condition (3) shows that indeed  $\mathbf{a}_L$  contains the binary decomposition of  $x$ .

These three checks can be combined into a single invocation of the IPA. The IPA used employs a technique that reduces each IPA of length- $n$  vectors to an equivalent IPA over length- $\frac{n}{2}$  vectors. Using this IPA results in a proofs size of  $O(\log_2 k)$ .

Subsequent works [28, 68, 69] slightly optimize Bulletproofs but keep the scheme and its properties (in particular, its transparent setup and aggregation properties) largely the same. Bulletproofs+ [28] slightly optimizes the Bulletproofs argument to reduce the number of group elements sent by the prover. Bulletproofs++ [36] further improves efficiency by reducing both prover and verifier time. All of these Bulletproofs derivatives maintain the same aggregation properties.

■ **Table 2** Properties of Bulletproofs-style proofs (all support aggregation and batched verification).

Bulletproofs-Style Range Proofs (all DLOG-based)		
Scheme	Commitment Scheme	Transparent Setup
Bulletproofs [17]	Pedersen	Y
Bulletproofs+ [28]	Pedersen	Y
Bulletproofs++ [36]	Pedersen	Y
Flashproofs [68]	Pedersen	Y
SwiftRange [69]	Pedersen	Y
DRZ [34]	Pedersen	N
ZZT+ [72]	Pedersen	N
Libert [52]	Pedersen	N
BFGW [11] + $KZG_{\text{Ped}}$	Pedersen	N
BFGW [11] + DARKs [19]	DARK [19]	Y with class groups; N with RSA

Bulletproofs++ [36] extends the recursive-style argument of Bulletproofs to work for any base, yielding asymptotic and concrete efficiency improvements. They do so using a *lookup argument*, which shows that committed values lie in some predefined table. Bulletproofs++ applies this lookup argument to show digit validity in arbitrary bases, allowing them to improve the proof size from Bulletproofs'  $O(\log_2 k)$  to  $O(\log_2 k / \log_2 \log_2 k)$ .

BFGW [11] takes a different approach to the binary decomposition idea, using a polynomial commitment scheme. We detail this approach in Section 3. This scheme assumes that the commitment to a value  $x$  is formed as commitment to a polynomial  $f$  such that  $f(1) = x$ . For some polynomial commitment schemes, such a commitment is nonstandard; conveniently, there is a version of KZG commitments for which this is a Pedersen commitment.

BFGW works with any hiding and binding polynomial commitment scheme, yielding different properties based on the scheme used. Notably, when instantiated with KZG commitments [49], BFGW has constant-sized proofs and is competitive efficiency-wise with Bulletproofs. Though KZG commitments require a trusted setup, this setup ceremony is perhaps one of the most commonly run, and some blockchains such as Ethereum have run a KZG ceremony.<sup>1</sup> In Section 9, we provide the first efficiency (prover and verifier time) benchmarks that we know of for BFGW + KZG. If the Pedersen variant of KZG commitments is used, BFGW + KZG is compatible with Pedersen commitments. BFGW can also be instantiated with DARKs [19], which do not require a trusted setup. Both BFGW + KZG and BFGW + DARKs are aggregatable.

## 6.2 Lattice- and code-based constructions

There are several lattice- and code- based zero knowledge range proof schemes. These schemes have the advantages that they are plausibly post-quantum secure and have a transparent setup. However, they are concretely much less efficient than the discrete logarithm-based schemes such as Bulletproofs. In particular, they have very long proofs. Thus, one worthwhile research direction is to improve the efficiency of these lattice-based protocols, such as [4, 37, 58]. One area for improvement is in the repetition required to achieve negligible soundness error. Most of these schemes build on protocols with constant soundness and must repeat the protocol  $\Omega(\lambda)$  times to achieve  $\lambda$  bits of security. When made non-interactive, this amplification results in large proofs.

<sup>1</sup> <https://blog.ethereum.org/2023/01/16/announcing-kzg-ceremony>

Lattice- and code-based schemes typically use the binary decomposition approach, where the prover already holds a commitment to the bits  $b_0, \dots, b_{k-1}$  of the value in question. The prover wants to show that  $\sum_{i=0}^{k-1} 2^i \cdot b_i \leq \beta$  for some  $\beta$ . This condition can be written equivalently as a system of equations over the bits modulo 2. Such systems of equations can be proven in zero-knowledge using *Stern-like* protocols [66].

In this section, we present several ideas involved in lattice-based schemes. We first present a lattice-based commitment scheme, KTX [51], that is used in some of these ZKRP. In doing so, we emphasize several challenges common to many lattice-based schemes. We then give a high-level description of Stern-like protocols, a standard technique for lattice-based zero-knowledge proofs. We also include a table with newer lattice-based schemes that offer constructions tailored to range proofs. We do not include all generic lattice-based zero-knowledge proof constructions.

**KTX commitment scheme ([51]).** The KTX commitment scheme is based on the hardness of the Short Integer Solution (SIS) problem. Let  $\lambda$  be the security parameter,  $L$  be the number of bits to be committed to, and  $q$  be a prime modulus of size  $O(\lambda\sqrt{L})$ . Let  $m = 2\lambda\lceil\log q\rceil$ . The scheme uses public parameters  $(\mathbf{A}, \mathbf{B})$  chosen uniformly from  $\mathbb{Z}_q^{\lambda \times L} \times \mathbb{Z}_q^{\lambda \times m}$ . The commitment to a bit vector  $\mathbf{x} \in \{0, 1\}^L$  is the vector

$$\mathbf{c} = \mathbf{A} \cdot \mathbf{x} + \mathbf{B} \cdot \mathbf{r} \pmod{q}$$

where  $\mathbf{r}$  is sampled uniformly from  $\{0, 1\}^m$ . This scheme is statistically hiding and computationally binding assuming that the public parameters are sampled uniformly.

Note that KTX commitments are only approximately homomorphic. While it holds that:

$$\mathbf{A} \cdot \mathbf{x}_1 + \mathbf{B} \cdot \mathbf{r}_1 + \mathbf{A} \cdot \mathbf{x}_2 + \mathbf{B} \cdot \mathbf{r}_2 = \mathbf{A}(\mathbf{x}_1 + \mathbf{x}_2) + \mathbf{B}(\mathbf{r}_1 + \mathbf{r}_2) \pmod{q},$$

note that  $(\mathbf{x}_1 + \mathbf{x}_2)$  and  $(\mathbf{r}_1 + \mathbf{r}_2)$  may not be 0/1 vectors. Therefore,  $A(x_1 + x_2) + B(r_1 + r_2)$  is not necessarily a valid commitment to a message in the message space. Many commitment schemes used by schemes in this section have similar limited homomorphism.

Note also that KTX commitments do not require a trusted setup to generate the public parameters  $\mathbf{A}$ ,  $\mathbf{B}$ , and  $q$ , as these matrices are uniformly random and  $q$  can be publicly known. Many lattice-based commitment schemes similarly use random matrices as the public parameters. All of the range proofs in this section offer transparent setup.

**Stern-like protocols.** Stern's original protocol [66] proves in zero knowledge that a committed bit vector has a certain Hamming weight; that is, it is a zero-knowledge argument of knowledge for the following relation:

$$\{((\mathbf{H}, \mathbf{y}, w), \mathbf{s}) \in \mathbb{Z}_2^{n \times m} \times \mathbb{Z}_2^n \times \mathbb{Z} \times \mathbb{Z}_2^m : (\text{wt}(\mathbf{s}) = w) \wedge (\mathbf{H} \cdot \mathbf{s} = \mathbf{y})\}$$

The key idea behind Stern's protocol is that the prover permutes the bits of  $\mathbf{s}$  to obtain  $\mathbf{s}'$  which it reveals to the verifier. It also convinces the verifier that  $\mathbf{s}'$  is indeed a permutation of  $\mathbf{s}$  under some  $\pi$ .  $\mathbf{s}'$  has the same Hamming weight as  $\mathbf{s}$ , and the distribution of  $\mathbf{s}'$  is identical for any  $\mathbf{s}$  satisfying the relation – therefore,  $\mathbf{s}'$  reveals no information about  $\mathbf{s}$ . At a high level, the prover samples a random blinding factor  $\mathbf{r}$  and constructs three commitments, which it sends to the verifier, as follows:

$$c_1 = \text{Com}(\pi, \mathbf{H} \cdot \mathbf{r}), \quad c_2 = \text{Com}(\pi(\mathbf{r})), \quad c_3 = \text{Com}(\pi(\mathbf{r} \oplus \mathbf{s}))$$

Here,  $\pi(\mathbf{v})$  denotes the vector obtained by permuting the components of  $\mathbf{v}$  under  $\pi$ . We now run one of three randomized checks: the verifier sends the prover  $b \in \{0, 1, 2\}$ . In each of these tests, the prover opens a different combination of the commitments and sends some

## 14:16 SoK: Zero-Knowledge Range Proofs

additional information, e.g.,  $\pi(\mathbf{s})$  for  $b = 2$ . The cheating prover cannot pass all of these tests simultaneously and therefore fails with probability at least  $1/3$ . Note that running all of these tests at once would reveal information about  $\mathbf{s}$ .

This permute-then-reveal strategy can be used for other relations with similar properties. [61] provides an abstraction of such relations, in terms of some set **VALID**, which in Stern's original protocol was  $\mathbf{VALID} = \{\mathbf{s} : \text{wt}(\mathbf{s}) = w\}$ :

$$R = \{((\mathbf{H}, \mathbf{y}), \mathbf{s}) \in \mathbb{Z}_2^{n \times m} \times \mathbb{Z}_2^n \times \mathbf{VALID} : \mathbf{H} \cdot \mathbf{x} = \mathbf{y}\}$$

**Correctness under permutation:** For all  $((\mathbf{H}, \mathbf{y}), \mathbf{s}) \in \mathbb{Z}_2^{n \times m} \times \mathbb{Z}_2^n \times \mathbb{Z}_2^m$  and all permutations  $\pi$  over  $[m]$ ,

$$\mathbf{s} \in \mathbf{VALID} \iff \pi(\mathbf{s}) \in \mathbf{VALID}$$

**Hiding under permutation:** For all  $\mathbf{s} \in \mathbf{VALID}$ , the distribution of  $\pi(\mathbf{s})$  where  $\pi$  is a random permutation over  $[m]$  is uniform over the set **VALID**

Even given a relation that does not fit the above requirements, one can sometimes construct an associated relation (e.g., using a common technique called *extension*) that does fall into this paradigm and allows one to construct the desired argument.

Other relations that can be proven under Stern's paradigm include proving knowledge of one secret bit that may appear in multiple equations [54], or proving the knowledge of the product of two secret bits [53]. Stern-like techniques underlie many older lattice- and code-based zero-knowledge protocols. However, recall that due to the randomized tests, Stern's original protocol has soundness error  $2/3$ . In general, Stern-like protocols have constant soundness error and thus require roughly  $\lambda$  repetitions for  $\lambda$  bits of security. Thus, once made non-interactive via Fiat-Shamir, these protocols result in long proofs.

Only recently have techniques emerged for avoiding Stern-like protocols in constructing lattice-based ZKRPs, whose state-of-the-art is thus not reflected in the previous ZKRP survey [60]. These new techniques resulted in a surge of lattice-based constructions with greatly improved efficiency, with proofs on the order of 10,000 KB rather than 100,000 KB. However, this efficiency still lags behind many non-lattice-based constructions with 500-byte proofs, as seen in Table 4. Improving lattice-based schemes remains a fruitful research direction.

[37] proposes techniques for avoiding the repetition that Stern-like protocols require for soundness. Their *one-shot* protocol saves a factor of  $\lambda$  computation time over repeated Stern-like protocols, though the proofs are still quite long as shown in Table 4. One-shot approaches are a fruitful direction for developing a more practical (in terms of both communication and computation) lattice-based ZKRP.

ALS [4] uses an inner product argument in the  $n$ -ary decomposition approach, which results in significantly shorter proofs compared to other lattice-based constructions; see Table 4. Its proofs are roughly an order of magnitude larger than those of the most efficient non-lattice schemes, such as Bulletproofs. Another barrier to practical efficiency is that the proofs of ALS cannot be aggregated.

## 7 Hash chain constructions

Payword [65] was the first to use hash chains to construct a range proof for electronic payments, and HashWires [24] more recently revisited this idea with great efficiency improvements. In this approach, the core idea is that a commitment  $C_x$  to a value  $x$  is the output of a hash function evaluated  $x$  times on a random value. That is,  $C_x = H^x(r)$  for a random  $r$ . The proof that  $x$  is at least some threshold  $t$  is a value  $\pi = H^{x-t}(r)$  such that applying the

■ **Table 3** Properties of lattice- and code-based range proofs. CKLR supports proof aggregation and batch verification, while it is unclear if the other schemes natively do so.

Lattice- and Code-Based Range Proofs			
Scheme	Commitment Scheme	Assumptions	Transp. Setup
LLNW [55]	KTX [51]	SIVP	Y
ESLL [37]	UMC, HMC [10, 5, 38]	Module-SIS, Module-LWE	Y
YAZ+ [71]	KTX [51]	LWE, SIS	Y
ALS [4]	BDLOP [5]	Module-SIS, Module-LWE	Y
CKLR [30]†	BDLOP [5], as modified by [71]	LWE, SIS	Y
LNS [58]*	BDLOP [5]	Module-SIS, Module-LWE	Y
LNP [57]	ABDLOP [1, 5]	Module-SIS, Module-LWE	Y
Code-based [61]	[61]	2-RNSD	Y

- †CKLR [30] uses the square decomposition approach, but one of their constructions is lattice-based.
- \*In addition to their standard range proof, LNS [58] also constructs an *approximate* range proof, showing that  $z \in [0, n \cdot 2^k - 1]$  for some small  $n$ . While relaxed, this kind of approximate range proof is sufficient for showing smallness of vectors, which is an application they target. Its efficiency does not depend on  $k$ .

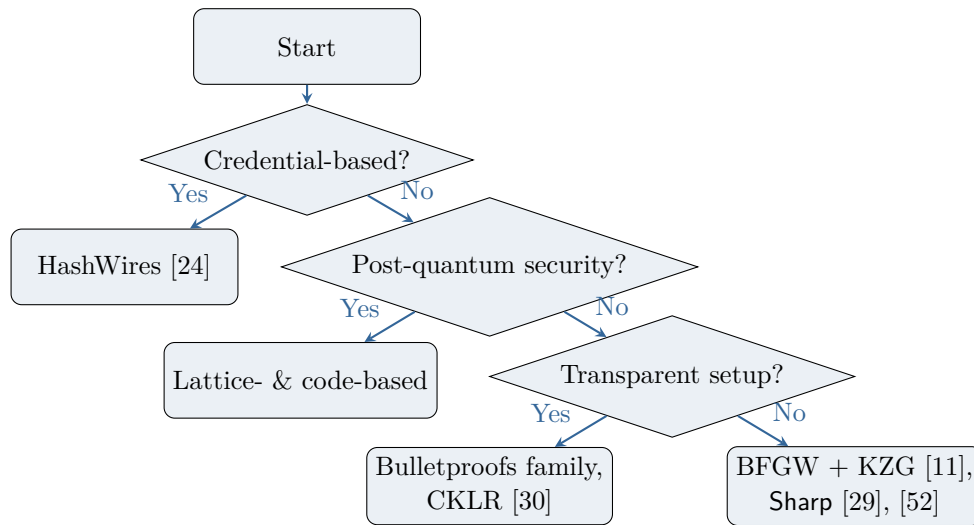
hash function  $t$  more times to  $\pi$  yields  $C_x$ ; that is,  $H^t(\pi) = C_x$ . Since the hash function is hard to invert, if  $x - t$  is negative it should be hard for the prover to find an accepting  $\pi$ . Importantly, though,  $C_x$  must be well-formed to ensure soundness. Thus, the setting where hash chain constructions can be used is slightly more restricted.

HashWires [24] defines a relaxation of zero-knowledge range proofs called *credential-based range proofs* (CBRPs). This notion is weaker than general ZKRPs in that the commitment is assumed to be well-formed. Soundness is shown only under this assumption, which is motivated by a setting where a trusted authority distributes commitments to parties that later prove that their committed values exceed some threshold. For example, the trusted authority may be a government, and the commitments might be used for credentials including citizens' ages. If a commitment is signed by this trusted authority, a verifier can be confident that the commitment is properly formed. Technically, this implies that when defining soundness for CBRPs, the adversary cannot produce the commitment (as defined in the statement of Def. 3, but instead is honestly generated (the full definition of CBRPs can be found in [24]).

As described, the time to generate  $\pi$  and  $C_x$  is linear in  $x$ , and the verifier time is linear in  $t$ . This is very expensive if we wish to prove that  $x$  is in some large range  $[0, 2^k - 1]$ ; ideally, these costs should grow at most linearly with  $k$ . HashWires achieves this by observing that  $x$  can be written in some base  $u$ , and the proof can be broken into several sub-chains to greatly improve this efficiency (they called this a *minimum dominating partition*). This base can be chosen to trade off between proof size and prover/verifier efficiency. In our later discussion of efficiency, we include benchmarks for a variety of bases. We will see in Section 9 that HashWires is extremely concretely efficient, in terms of both verifier time and prover time. Its proof sizes are also competitive with other constructions.

## 8 Choosing the construction family for your application

As there are dozens of ZKRP constructions, choosing the appropriate scheme for a particular application can be challenging. In Figure 1, we give a flowchart for narrowing down the class of range proofs depending on constraints. The next section gives an efficiency comparison to help choose a scheme within this class.



■ **Figure 1** Flowchart for choosing a range proof based on desired properties.

HashWires [24] are concretely quite efficient and use only hash functions; thus, they are plausibly post-quantum secure and do not require a trusted setup. However, they’re in a more stringent trust model (they are *credential-based range proofs* as defined in Section 7), where there is a trusted issuer distributing commitments; that is, soundness holds only if the commitment is well-formed. If the desired use case does have this type of trusted issuer, HashWires is likely the most efficient scheme.

Among the remaining constructions, only the lattice-and code-based constructions are plausibly post-quantum secure, and thus if this is a requirement this class is the only option. These schemes have relatively large proof sizes (on the order of 10KB). Hash-based generic zero-knowledge proof systems may be considered as well.

If trusted setup is allowed, there are several schemes with very short proofs and efficient verifier and prover. BFGW + KZG [11], Sharp [29], and Libert’s DLOG-based scheme [52] all have constant-sized proofs.

If trusted setup is undesired, the Bulletproofs family is recommended. Although many lattice- and code-based constructions do not require a trusted setup, all Bulletproofs-style constructions have much shorter proofs. Even if a trusted setup is allowed, Bulletproofs-style constructions may still be worth considering depending on how much one values short proofs. Though their proof sizes are not constant, they seem to be the most commonly used in practice. We list CKLR [30] as well because it has comparable efficiency to Bulletproofs on paper and also does not require trusted setup. However, it has several drawbacks: it does not allow batching, it is less efficient in practice due to its incompatibility with optimized libraries for common elliptic curves, and it offers a more relaxed notion of security. For certain applications where these drawbacks are less important, CKLR may be worth considering.

## 9 Efficiency Comparison

This section includes an efficiency comparison of various ZKRPs. In Table 4, we compile both concrete and asymptotic proof sizes for schemes of particular interest. The concrete proof sizes have been extracted directly from the schemes’ respective papers, as the proof sizes are largely the same across machine configurations. Groth16 has the shortest range proofs for a 64-bit range at 192 bytes whereas HashWires has the shortest range proofs at 177 bytes for a 32-bit range.

In Table 5, we record prover and verifier times for various schemes. We add many of our own benchmarks to ensure that configurations are normalized. In particular, we add a benchmark for Groth16 [44] that was absent in prior work. The configurations for benchmarks that we pull from other papers are noted below.

**Other benchmarks.** The Sharp paper’s [29] benchmark was run on a MacBook Pro with a 2.3 GHz Intel core i7 processor and uses the library libsecp256k1 [70]. The HashWires paper [24] includes a benchmark for Bulletproofs which is significantly faster than ours. They used an AVX2 backend which significantly speeds up curve arithmetic. We include this benchmark in addition to ours, to reflect the speedup possible with their configuration.

**Our benchmarks.** We add our own benchmarks for Hashwires (base 16 and base 256), Bulletproofs, BFGW + KZG, and Groth16. In all cases, we record the median running time over 100 runs. We plan to open source all of our benchmarks for reproducibility.

For Groth16, we implement range proofs with two versions of the commitment scheme: the well-established Pedersen commitments and the new zk-friendly Poseidon commitments. We’ve used Circom [8] for writing circuits and rapidsnark [47] for generating and verifying the Groth16 proofs.

The implementations for Hashwires, Bulletproofs and BFGW + KZG are in Rust. All the benchmarks were run on a AMD EPYC 7443P 24-Core with 512GB of RAM (a c3.large.x86 machine hosted by latitude.sh). We explicitly chose a non-Mac machine because rapidsnark leverages Intel Assembly to speed up Groth16 proof generation.

Hashwires has the fastest proof generation and verification times. Both BFGW + KZG and Groth16 have constant-sized proofs but they are less computationally efficient than others. Groth16 has the longest proof generation times. This is expected because we are instantiating range proofs within a general-purpose zk proof system.

It is worth noting that in practice the availability of a reliable library may outweigh mild efficiency gains. Bulletproofs is the most widely used range proof in practice and is likely a good choice. Groth16, though not tailored to range proofs, is one of the most popular general-purpose zero-knowledge proof systems and offers several well supported libraries; we use Circom [8] and rapidsnark [47]. From our benchmarks, one can see the efficiency gains offered by tailored range proof solutions over generic solutions, which can be seen especially in the long prover times required for Groth16 relative to the other range proofs.

## 10 Research Gaps

► **Research Gap 1.** *Practical transparent constant-sized range proofs.*

No zero-knowledge range proofs are practical, transparent, and have constant-sized proofs. Bulletproofs and its close relatives have transparent setup but have proofs of size  $O(\log k)$  for a  $k$ -bit range. BFGW + KZG has constant-sized proofs but requires a trusted setup; BFGW + DARKs has a transparent setup but requires  $O(\log k)$ -sized proofs. CKLR has a transparent setup and has constant-sized proofs but achieves only a relaxed notion of soundness. Furthermore, its proofs are not as practically efficient as the above schemes because they use less common curves that optimized libraries do not support.

► **Research Gap 2.** *Shorter (even amortized) lattice- or code-based ZKRP.*

## 14:20 SoK: Zero-Knowledge Range Proofs

■ **Table 4** Proof sizes in bytes for 64- and 32-bit ranges. The benchmark for each of these schemes is from that scheme’s original paper, except where otherwise noted.

Scheme	Proof size (bytes)		Proof size (asymptotic)
	32-bit range	64-bit range	$k$ -bit range
Bulletproofs	610	675	$O(\log k)$
BFGW + KZG †	576	576	$O(1)$
Sharp <sub>CS</sub>	318	360	$O(1)$
Sharp <sub>SO</sub> <sup>Po</sup>	335	389	$O(1)$
Sharp <sub>RSA</sub>	751	793	$O(1)$
HashWires (Base 16) †	231	263	$O(\log k)$
HashWires (Base 256) †	167	199	$O(\log k)$
Groth16 [44] <sup>§</sup>	192	192	$O(1)$
Lattice-based ALS [4]**	5,900	-	$O(k)$
Lattice-based ESSL [37]	58,000	93,000	$\Omega(k)^*$
Lattice-based LNS [58]**	11,800	-	$o(k)^\ddagger$

- † Our own benchmark.
- § Benchmark from HashWires [24], over the BLS12-381 curve.
- \*See [37] for the exact expression, which includes several other parameters not described here. It is  $\Omega(k)$  and is large relative to the other schemes.
- \*\*The proof sizes for 64-bit ranges were not included in [4, 58]. Note that [4] has linear growth, so extrapolating from its 5,900-bit proof for 32-bit ranges, its proof for 64-bit ranges would be large.
- ‡ See [58] for the exact expression, which is complicated; it is sublinear in  $k$ .

■ **Table 5** Verifier and prover times. †Our own benchmark.

Scheme	Verifier Time (ms)		Prover Time (ms)	
	32-bit range	64-bit range	32-bit range	64-bit range
Bulletproofs †	1.37	2.51	6.32	11.96
Sharp <sub>SO</sub> <sup>Po</sup>	0.74	0.75	0.97	1.17
Bulletproofs AVX2 (HashWires benchmark)	-	0.938	-	6.516
HashWires base 16 †	0.002	0.002	0.003	0.061
HashWires base 256 †	0.009	0.01	0.083	0.194
BFGW + KZG †	5.653	5.682	9.572	12.569
Groth16-Poseidon †	4	4	34.23	34.46
Groth16-Pedersen †	4	4	31.18	33.57

The proofs of lattice-based and code-based ZKRPs are concretely quite long, as shown in Table 4. For blockchain applications where one must pay for the space used on-chain, this length is problematic, especially as these constructions do not support aggregation. In order to be competitive with constructions using other techniques shown in Table 4, the proof size must be under 1 KB.

► **Research Gap 3.** *Lattice- or code-based ZKRPs with multi-prover aggregation.*

Lattice-based ZKRPs with short proofs are desirable for confidential transactions, as blockchains transition to post-quantum security. In such settings, this size issue may be mitigated by multi-prover aggregation. Each block would then contain only an aggregate range proof for all included transactions. However, this aggregation must be multi-prover as these transactions may be made by many different parties, each holding commitments to private values. Lattice- and code-based ZKRPs with multi-prover aggregation have not yet been constructed, leading us to the this related research gap.

► **Research Gap 4.** *Un-replayable credential-based range proofs.*



For credential applications, one might want an interactive range proof that cannot be replayed. Suppose that Alice has a commitment of her age signed by a trusted credential issuer. Alice should be able to visit the DMV and prove in zero knowledge that her committed age is above 16. An observer Bob should not be able to copy Alice's commitment and re-use the transcript of the protocol to prove (possibly falsely) that his age is above 16. If this range proof is non-interactive, Bob can simply copy the proof and re-use it. This re-use might be avoided if the protocol is public-coin interactive, and the DMV issues a random challenge that requires knowledge of the committed value to respond to.

Can we make hash-chain-based range proofs that are un-replayable in this way? As credentials are a primary motivation for HashWires, un-replayability would be a nice property to add.

► **Research Gap 5.** *Integer commitments with full soundness with transparent setup.*

CKLR [30] and Sharp [29] construct integer commitments with a relaxed notion of soundness. In order to be used for confidential transactions, they must be augmented with additional proof elements from an RSA group or class group. The RSA version requires a trusted setup, and the class group solution is not compatible with existing optimized libraries. Rather than patching soundness issues by adding these extra elements, it would be preferred to construct practically efficient integer commitments with full soundness and transparent setup.

► **Research Gap 6.** *Efficient post-quantum ZKRPs compatible with LWE-based ciphertexts.*

Zero-knowledge range proofs can be used to build verifiable LWE-based encryption schemes as discussed in our full version [27]. However, existing verifiable LWE-based encryption schemes constructed using ZKRPs [35, 52] use discrete logarithm-based ZKRPs. Thus, while they obtain privacy against quantum adversaries due to the LWE-based encryption used, they lack soundness in verification due to the DLOG-based range proofs. If there were efficient post-quantum range proofs compatible with LWE-based ciphertexts, one could obtain verifiable encryption with soundness against quantum adversaries as well. While a lattice-based zkSNARK (e.g., [2]) may work in theory, it may not be efficient (yielding long ciphertexts and heavy computation). An efficient lattice-based ZKRP that is compatible with lattice-based encryption would be more satisfactory.

---

## References

- 1 Miklós Ajtai. Generating hard instances of lattice problems. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 99–108, 1996.
- 2 Martin R Albrecht, Valerio Cini, Russell WF Lai, Giulio Malavolta, and Sri Aravinda Krishnan Thyagarajan. Lattice-based snarks: Publicly verifiable, preprocessing, and recursively composable. In *Annual International Cryptology Conference*, pages 102–132. Springer, 2022.
- 3 Sebastian Angel and Michael Walfish. Verifiable auctions for online ad exchanges. In *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, pages 195–206, 2013.
- 4 Thomas Attema, Vadim Lyubashevsky, and Gregor Seiler. Practical product proofs for lattice commitments. In *Annual International Cryptology Conference*, pages 470–499. Springer, 2020.
- 5 Carsten Baum, Ivan Damgård, Vadim Lyubashevsky, Sabine Oechsner, and Chris Peikert. More efficient commitments from structured lattice assumptions. In *International Conference on Security and Cryptography for Networks*, pages 368–385. Springer, 2018.
- 6 Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Randomizable proofs and delegatable anonymous credentials. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16–20, 2009. Proceedings*, volume 5677 of *Lecture Notes in Computer Science*, pages 108–125. Springer, 2009. doi:10.1007/978-3-642-03356-8\_7.

- 7 James Bell, Adrià Gascón, Tancrede Lepoint, Baiyu Li, Sarah Meiklejohn, Mariana Raykova, and Cathie Yun. {ACORN}: Input validation for secure aggregation. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 4805–4822, 2023.
- 8 Marta Bellés-Muñoz, Miguel Isabel, Jose Luis Muñoz-Tapia, Albert Rubio, and Jordi Baylina. Circom: A circuit description language for building zero-knowledge applications. *IEEE Transactions on Dependable and Secure Computing*, 2022.
- 9 Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. *Cryptology ePrint Archive*, 2018.
- 10 Fabrice Benhamouda, Stephan Krenn, Vadim Lyubashevsky, and Krzysztof Pietrzak. Efficient zero-knowledge proofs for commitments from learning with errors over rings. In *European symposium on research in computer security*, pages 305–325. Springer, 2015.
- 11 D Boneh, B Fisch, A Gabizon, and Z Williamson. A simple range proof from polynomial commitments, 2020. URL: <https://hackmd.io/@dabo/B1U4kx8XI>.
- 12 Dan Boneh and Matthew Franklin. Efficient generation of shared rsa keys. In *Advances in Cryptology – CRYPTO’97: 17th Annual International Cryptology Conference Santa Barbara, California, USA August 17–21, 1997 Proceedings 17*, pages 425–439. Springer, 1997.
- 13 Jonathan Bootle, Andrea Cerulli, Pyrrhos Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In *Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8–12, 2016, Proceedings, Part II 35*, pages 327–357. Springer, 2016.
- 14 Fabrice Boudot. Efficient proofs that a committed number lies in an interval. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 431–444. Springer, 2000.
- 15 Ernest F Brickell, David Chaum, Ivan B Damgård, and Jeroen van de Graaf. Gradual and verifiable release of a secret. In *Advances in Cryptology – CRYPTO’87: Proceedings 7*, pages 156–166. Springer, 1988.
- 16 Benedikt Bünz, Shashank Agrawal, Mahdi Zamani, and Dan Boneh. Zether: Towards privacy in a smart contract world. In *International Conference on Financial Cryptography and Data Security*, pages 423–443. Springer, 2020.
- 17 Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE symposium on security and privacy (SP)*, pages 315–334. IEEE, 2018.
- 18 Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent snarks from dark compilers. In *Advances in Cryptology–EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part I 39*, pages 677–706. Springer, 2020.
- 19 Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent snarks from dark compilers. In *Advances in Cryptology–EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part I 39*, pages 677–706. Springer, 2020.
- 20 Philippe Camacho. Secure protocols for provable security, 2014. URL: <https://www.slideshare.net/philippecamacho/protocols-for-provable-solvency-38501620>.
- 21 Jan Camenisch, BRICS, and Markus Michels. Separability and efficiency for generic group signature schemes. In *Annual International Cryptology Conference*, pages 413–430. Springer, 1999.
- 22 Jan Camenisch, Rafik Chaabouni, and Abhi Shelat. Efficient protocols for set membership and range proofs. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 234–252. Springer, 2008.
- 23 Jan Camenisch and Markus Michels. Proving in zero-knowledge that a number is the product of two safe primes. In *Advances in Cryptology – EUROCRYPT’99: International Conference on the Theory and Application of Cryptographic Techniques Prague, Czech Republic, May 2–6, 1999 Proceedings 18*, pages 107–122. Springer, 1999.

- 24 Konstantinos Chalkias, Shir Cohen, Kevin Lewi, Fredric Moezina, and Yolán Romailler. Hashwires: Hyperefficient credential-based range proofs. *Proceedings on Privacy Enhancing Technologies*, 4:76–95, 2021.
- 25 Agnes Hui Chan, Yair Frankel, and Yiannis Tsioumis. Easy come – easy go divisible cash. In Kaisa Nyberg, editor, *Advances in Cryptology – EUROCRYPT ’98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 – June 4, 1998, Proceeding*, volume 1403 of *Lecture Notes in Computer Science*, pages 561–575. Springer, 1998. doi:10.1007/BFb0054154.
- 26 Panagiotis Chatzigiannis and Foteini Baldimtsi. Miniledger: Compact-sized anonymous and auditable distributed payments. In Elisa Bertino, Haya Shulman, and Michael Waidner, editors, *Computer Security – ESORICS 2021 – 26th European Symposium on Research in Computer Security, Darmstadt, Germany, October 4-8, 2021, Proceedings, Part I*, volume 12972 of *Lecture Notes in Computer Science*, pages 407–429. Springer, 2021. doi:10.1007/978-3-030-88418-5\_20.
- 27 Miranda Christ, Foteini Baldimtsi, Konstantinos Kryptos Chalkias, Deepak Maram, Arnab Roy, and Joy Wang. SoK: Zero-knowledge range proofs. *Cryptology ePrint Archive*, Paper 2024/430, 2024. URL: <https://eprint.iacr.org/2024/430>.
- 28 Heewon Chung, Kyoohyung Han, Chanyang Ju, Myungsun Kim, and Jae Hong Seo. Bulletproofs+: Shorter proofs for a privacy-enhanced distributed ledger. *IEEE Access*, 10:42081–42096, 2022.
- 29 Geoffroy Couteau, Dahmun Goudarzi, Michael Kloob, and Michael Reichle. Sharp: Short relaxed range proofs. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 609–622, 2022.
- 30 Geoffroy Couteau, Michael Kloob, Huang Lin, and Michael Reichle. Efficient range proofs with transparent setup from bounded integer commitments. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 247–277. Springer, 2021.
- 31 Geoffroy Couteau, Thomas Peters, and David Pointcheval. Removing the strong rsa assumption from arguments over the integers. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 321–350. Springer, 2017.
- 32 Gaby G Dagher, Benedikt Bünz, Joseph Bonneau, Jeremy Clark, and Dan Boneh. Provisions: Privacy-preserving proofs of solvency for bitcoin exchanges. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 720–731, 2015.
- 33 Ivan Damgård and Eiichiro Fujisaki. An integer commitment scheme based on groups with hidden order. *Cryptology ePrint Archive*, 2001.
- 34 Vanesa Daza, Carla Ràfols, and Alexandros Zacharakis. Updateable inner product argument with logarithmic verifier and applications. In *Public-Key Cryptography–PKC 2020: 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4–7, 2020, Proceedings, Part I 23*, pages 527–557. Springer, 2020.
- 35 Rafaël Del Pino, Vadim Lyubashevsky, and Gregor Seiler. Short discrete log proofs for the and ring-lwe ciphertexts. In *IACR International Workshop on Public Key Cryptography*, pages 344–373. Springer, 2019.
- 36 Liam Eagen. Bulletproofs++. *Cryptology ePrint Archive*, 2022.
- 37 Muhammed F Esgin, Ron Steinfeld, Joseph K Liu, and Dongxi Liu. Lattice-based zero-knowledge proofs: new techniques for shorter and faster constructions and applications. In *Annual International Cryptology Conference*, pages 115–146. Springer, 2019.
- 38 Muhammed F Esgin, Ron Steinfeld, Amin Sakzad, Joseph K Liu, and Dongxi Liu. Short lattice-based one-out-of-many proofs and applications to ring signatures. In *Applied Cryptography and Network Security: 17th International Conference, ACNS 2019, Bogota, Colombia, June 5–7, 2019, Proceedings 17*, pages 67–88. Springer, 2019.
- 39 Yair Frankel, Philip D MacKenzie, and Moti Yung. Robust efficient distributed rsa-key generation. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 663–672, 1998.

- 40 Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *Advances in Cryptology – CRYPTO’97: 17th Annual International Cryptology Conference Santa Barbara, California, USA August 17–21, 1997 Proceedings 17*, pages 16–30. Springer, 1997.
- 41 Ariel Gabizon, Zachary J Williamson, and Oana Ciobotaru. Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. *Cryptology ePrint Archive*, 2019.
- 42 Craig Gentry, Shai Halevi, and Vadim Lyubashevsky. Practical non-interactive publicly verifiable secret sharing with thousands of parties. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 458–487. Springer, 2022.
- 43 Jens Groth. Non-interactive zero-knowledge arguments for voting. In *Applied Cryptography and Network Security: Third International Conference, ACNS 2005, New York, NY, USA, June 7-10, 2005. Proceedings 3*, pages 467–482. Springer, 2005.
- 44 Jens Groth. On the size of pairing-based non-interactive arguments. In *Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II 35*, pages 305–326. Springer, 2016.
- 45 Jens Groth. Non-interactive distributed key generation and key resharing. *Cryptology ePrint Archive*, 2021.
- 46 G.H. Hardy, E.M. Wright, D.R. Heath-Brown, and J. Silverman. *An Introduction to the Theory of Numbers*. Oxford mathematics. OUP Oxford, 2008.
- 47 iden3. rapidsnark. <https://github.com/iden3/rapidsnark>, 2023.
- 48 Yan Ji and Konstantinos Chalkias. Generalized proof of liabilities. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 3465–3486, 2021.
- 49 Aniket Kate, Gregory M Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In *Advances in Cryptology-ASIACRYPT 2010: 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings 16*, pages 177–194. Springer, 2010.
- 50 Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography: principles and protocols*. Chapman and hall/CRC, 2007.
- 51 Akinori Kawachi, Keisuke Tanaka, and Keita Xagawa. Concurrently secure identification schemes based on the worst-case hardness of lattice problems. In *Advances in Cryptology-ASIACRYPT 2008: 14th International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, Australia, December 7-11, 2008. Proceedings 14*, pages 372–389. Springer, 2008.
- 52 Benoît Libert. Vector commitments with short proofs of smallness. *Cryptology ePrint Archive*, 2023.
- 53 Benoît Libert, San Ling, Fabrice Mouhartem, Khoa Nguyen, and Huaxiong Wang. Zero-knowledge arguments for matrix-vector relations and lattice-based group encryption. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 101–131. Springer, 2016.
- 54 Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. Zero-knowledge arguments for lattice-based accumulators: logarithmic-size ring signatures and group signatures without trapdoors. In *Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II 35*, pages 1–31. Springer, 2016.
- 55 Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. Lattice-based zero-knowledge arguments for integer relations. In *Annual International Cryptology Conference*, pages 700–732. Springer, 2018.



- 56 Helger Lipmaa. On diophantine complexity and statistical zero-knowledge arguments. In *Advances in Cryptology-ASIACRYPT 2003: 9th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, November 30–December 4, 2003. Proceedings 9*, pages 398–415. Springer, 2003.
- 57 Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Maxime Plançon. Lattice-based zero-knowledge proofs and applications: shorter, simpler, and more general. In *Annual International Cryptology Conference*, pages 71–101. Springer, 2022.
- 58 Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Practical lattice-based zero-knowledge proofs for integer relations. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, pages 1051–1070, 2020.
- 59 Greg Maxwell. Confidential transactions, 2016. URL: [https://people.xiph.org/~greg/confidential\\_values.txt](https://people.xiph.org/~greg/confidential_values.txt).
- 60 Eduardo Morais, Tommy Koens, Cees van Wijk, and Aleksei Koren. A survey on zero knowledge range proofs and applications. *CoRR*, abs/1907.06381, 2019. [arXiv:1907.06381](https://arxiv.org/abs/1907.06381).
- 61 Khoa Nguyen, Hanh Tang, Huaxiong Wang, and Neng Zeng. New code-based privacy-preserving cryptographic constructions. In *Advances in Cryptology-ASIACRYPT 2019: 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8–12, 2019, Proceedings, Part II 25*, pages 25–55. Springer, 2019.
- 62 Valeria Nikolaenko, Sam Ragsdale, Joseph Bonneau, and Dan Boneh. Powers-of-tau to the people: Decentralizing setup ceremonies. In *International Conference on Applied Cryptography and Network Security*, pages 105–134. Springer, 2024.
- 63 Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Annual international cryptology conference*, pages 129–140. Springer, 1991.
- 64 Andrew Poelstra. Mumblewimble, 2016.
- 65 Ronald L Rivest and Adi Shamir. Payword and micromint: Two simple micropayment schemes. In *International workshop on security protocols*, pages 69–87. Springer, 1996.
- 66 Jacques Stern. A new paradigm for public key identification. *IEEE Transactions on Information Theory*, 42(6):1757–1768, 1996.
- 67 Alin Tomescu. Range proofs from polynomial commitments, re-explained, March 2020. URL: <https://decentralizedthoughts.github.io/2020-03-03-range-proofs-from-polynomial-commitments-reexplained/>.
- 68 Nan Wang and Sid Chi-Kin Chau. Flashproofs: Efficient zero-knowledge arguments of range and polynomial evaluation with transparent setup. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 219–248. Springer, 2022.
- 69 Nan Wang, Sid Chi-Kin Chau, and Dongxi Liu. Swiftrange: A short and efficient zero-knowledge range argument for confidential transactions and more. *Cryptology ePrint Archive*, 2023.
- 70 Pieter Wuille. libsecp256k1, 2018. URL: <https://github.com/bitcoin/secp256k1>.
- 71 Rupeng Yang, Man Ho Au, Zhenfei Zhang, Qiuliang Xu, Zuoxia Yu, and William Whyte. Efficient lattice-based zero-knowledge arguments with standard soundness: construction and applications. In *Advances in Cryptology-CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part I 39*, pages 147–175. Springer, 2019.
- 72 Zibo Zhou, Zongyang Zhang, Hongyu Tao, Tianyu Li, and Boyu Zhao. Efficient inner product arguments and their applications in range proofs. *IET Information Security*, 17(3):485–504, 2023.



# Privacy Comparison for Bitcoin Light Client Implementations

Arad Kotzer  

The Department of Computer Science, Technion, Haifa, Israel

Ori Rottenstreich  

The Department of Computer Science and the Department of Electrical and Computer Engineering, Technion, Haifa, Israel

---

## Abstract

Light clients implement a simple solution for Bitcoin’s scalability problem, as they do not store the entire blockchain but only the state of particular addresses of interest. To be able to keep track of the updated state of their addresses, light clients rely on full nodes to provide them with the required information. To do so, they must reveal information about the addresses they are interested in. This paper studies the two most common light client implementations, SPV and Neutrino with regards to their privacy. We define privacy metrics for comparing the privacy of the different implementations. We evaluate and compare the privacy of the implementations over time on real Bitcoin data and discuss the inherent privacy-communication tradeoff. In addition, we propose general techniques to enhance light client privacy in the existing implementations. Finally, we propose a new SPV-based light client model, the aggregation model, evaluate it, and show it can achieve enhanced privacy than in the existing light client implementations.

**2012 ACM Subject Classification** Networks → Network measurement

**Keywords and phrases** Blockchain, Privacy, Light Clients, Bloom filter

**Digital Object Identifier** 10.4230/LIPIcs.AFT.2024.15

## 1 Introduction

Blockchain networks like Bitcoin [41] are composed of a decentralized blockchain, structured as an immutable chain of data blocks. Starting from the first block, each block includes the output of a cryptographic hash function computed over the content of the previous block, making it impossible to alter a block without changing all subsequent blocks. Blockchain blocks are lists of transactions bundled together due to the high cost of the consensus protocol. Blocks are composed of two main components: block header and transactions. The block header stores only metadata, a hash of the previous block and the root of the Merkle tree of the block transactions [36]. The increasing amount of memory required to maintain the full Bitcoin state together with rapid growth in the volume of transactions that must be processed imply a large overhead on full Bitcoin nodes. Hence, not all nodes participating in a blockchain store and process the entire blockchain.

A light client is a node variant that can verify only part of a block, without locally maintaining the complete network state. While *full nodes* process the entire block (both header and transactions), *light clients* process only partial block information. Light clients are connected to full nodes and receive relevant information through them. To do so, light clients reveal in various forms information about the addresses of their interest [23]. This paper focuses on two common light client implementations SPV [41] and Neutrino [44], covering the different approaches used by most existing light client solutions.



© Arad Kotzer and Ori Rottenstreich;  
licensed under Creative Commons License CC-BY 4.0  
6th Conference on Advances in Financial Technologies (AFT 2024).

Editors: Rainer Böhme and Lucianna Kiffer; Article No. 15; pp. 15:1–15:23

Leibniz International Proceedings in Informatics



LIPIC Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Our main contributions are as follows:

- We overview the main existing light client implementations: SPV and Neutrino.
- We define privacy metrics for measuring the privacy of the implementations.
- We perform both theoretical and empirical analysis, and compare the privacy of the different light clients over time on real data.
- We discuss how light clients may improve their privacy.
- We present a new model that further improves light client privacy.

We structure the paper as follows. Section 2 overviews methods for set representation, in addition to previous work on light client privacy. Section 3 presents the paper’s threat model and Section 4 describes metrics used to measure the privacy of light clients. Next, we provide a theoretical analysis of the privacy of SPV and Neutrino in Section 5. In subsections 6.1-6.3 we conduct an empirical analysis of the privacy of SPV and Neutrino, based on real data. Subsection 6.4 presents the privacy-communication tradeoff and compare the different implementations. These results are further discussed in Subsection 6.5. Then, we propose a new light client model that improves privacy in Section 7. Section 8 concludes the paper.

## 2 Background

### 2.1 Memory-efficient Methods for Set Representation

**Bloom Filters.** The Bloom filter [10, 13] is a popular data structure widely used in many networking device algorithms [12, 34], in fields as diverse as packet classification, routing, filtering, caching, and accounting, as well as beyond networking in areas like verification and spell checking. The Bloom filter is used for set representation, supporting element insertion and answering membership queries. There are two kinds of errors in membership queries: a false positive (when an element  $x \notin U$  is reported as a member of a represented set  $U$ ) and a false negative (when an element  $x \in U$  is reported as a nonmember of  $U$ ). The Bloom filter encounters false positives and has no false negatives. It is built as an array of bits, where hash functions map elements to bits in the array. With initial values of zero bits, the elements of  $U$  are first inserted into the filter, setting the bits pointed by the hash functions. Upon a query, bits mapped by the queried element are examined and a positive answer is returned only when the bits are all set.

**Golomb-Coded Set (GCS).** *GCS* is a data structure similar to Bloom filters, though it has a more compact in-memory representation that comes at the expense of having a slower query time (compared to Bloom filters) [24]. Given a hash function,  $N$  the number of items to be inserted into the set and an expansion parameter  $M$ , GCS works as follows:

- (i) Hash all items using hashing function  $H$  to integers in the range  $[0, N \cdot M)$ .
- (ii) Sort hashed values (in ascending order).
- (iii) Calculate the differences between each value and the previous one.
- (iv) Write the differences sequentially, compressed with Golomb coding [24].

Similar to Bloom filters, GCS is a probabilistic data structure that may contain false positives as a tradeoff with the memory size. Assuming the hashing function  $H$  has a uniform distribution, elements that were not inserted into the set have a probability of  $\frac{N \cdot 1}{N \cdot M} = \frac{1}{M}$  to have the same hash value of an element in the set (after step (i)), and thus to appear in the set. Small  $M$  values reduce the GCS size but increase the false positive rate. Table 1 summarizes the main notations of the paper.



■ **Table 1** Summary of main notations.

Symbol	Meaning	relevant to
$A$	address space	SPV, Neutrino
$ c $	number of addresses associated with client $c$	SPV, Neutrino
$p_i$	probability that an address $i$ belongs to $c$	SPV, Neutrino
$H$	client entropy	SPV, Neutrino
$R$	detection ratio	SPV, Neutrino
$X$	number of addresses guessed correctly among $ c $ guessed addresses	SPV, Neutrino
$E_c$	expected number of correctly guessed addresses	SPV, Neutrino
$x$	a target probability sum	SPV, Neutrino
$N$	number of inserted elements in a set	Bloom filter, GCS
$F$	number of false positives in a set	Bloom filter (SPV)
$M$	expansion parameter	GCS (Neutrino)
$p_{FP}$	probability of a downloaded block being a false positive	Neutrino
$K$	number of addresses in a block	Neutrino
$k$	number of non-eliminated addresses in a block	Neutrino
$A'$	address space excluding all eliminated addresses	Neutrino
$z$	number of blocks an address appears in	Neutrino
$p_i^z$	probability that non-eliminated address $i$ appearing in $z$ blocks downloaded by client $c$ is associated with $c$	Neutrino
$G$	number of groups client $c$ participates in	Aggregation Model
$S$	number of clients in each group	Aggregation Model
$l$	group leader	Aggregation Model
$p_{colluding}$	probability light clients might collude with a full node	Aggregation Model

## 2.2 Light client implementations

**Bitcoin Simplified Payment Verification (SPV).** *SPV* clients were first suggested in Bitcoin’s original white paper [41]. Since light clients do not keep track of the entire network state but only of several addresses in their interest, to be familiar with the balance of these addresses, SPV clients request all relevant transactions from a full node. To preserve privacy regarding the addresses associated with each client, light clients do not send an explicit list of relevant addresses but send a filter containing these addresses implicitly. Among all transactions that appear in a block, a full node only forwards those transactions that match the SPV filter, potentially with some false positives (namely transactions beyond the interest of the SPV client). Together with the particular transactions of interest, the full node also provides Merkle-tree-based proofs, demonstrating their inclusion in the block. Once the SPV client receives and validates the transactions (using the Merkle proof), it updates its state according to the transactions.

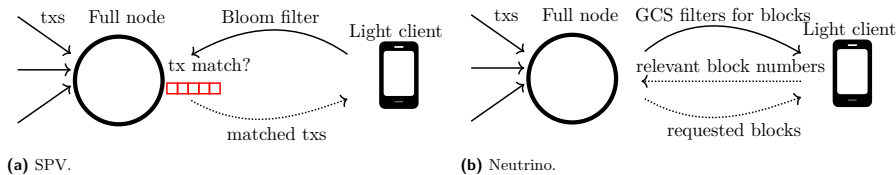
A *Bitcoin Improvement Proposal (BIP)* is a formal proposal to change Bitcoin suggested by the Bitcoin community (recall Bitcoin does not have one centralized leader). In BIP-37 [37], Bloom Filters were suggested as light client address filters. Using Bloom filters allows SPV clients to in-explicitly express the set of addresses they are interested in. The filter length can be selected based on a required false positive probability. Fig. 1a illustrates the process of an SPV client requesting transactions from a full node.

Though multiple Bitcoin light client implementations exist, most implementations, such as Electrum [51], Bither [47] and Mycelium [1], are all SPV-based. Moreover, [15] covers the main blockchain systems supporting light clients, and though these systems differ from each other, and accordingly the light client implementation they support, almost all of the light client implementations, including Binance light client [5], Cosmos - InterBlockchain

## 15:4 Privacy Comparison for Bitcoin Light Client Implementations

■ **Table 2** A high-level comparison between SPV and Neutrino.

	SPV [37]	Neutrino [3]
Set representation method	Bloom filter	Golomb-coded set
Data downloaded by light client	Transactions	Blocks
Who performs most computation?	Full node	Light client
Privacy achieved by	False positives	Downloading blocks



■ **Figure 1** (a) Bitcoin SPV Clients: The light client reports its set of relevant addresses through a Bloom filter. (b) Neutrino Clients: A full node sends GCS (Golomb-Coded Sets) filters of the addresses in newly generated blocks, the light client reports relevant blocks it would like to download and the full node sends these blocks.

Communication [11] and ZCash’s Flyclient [14] light client, are all SPV-based or have a similar implementation to SPV. Ethereum light clients such as Helios [4], Kevlar [50] and Lodestar [6] use Bloom filters too for gathering information transactions from a full node. Cuckoo filter [22] based light clients [48] are similar too, having a client using a filter to request transactions when updating its state. The advantages of SPV clients are that they store locally only a small amount of data, in addition to very little data that is sent as part of the communication with the full node. Additionally, SPV clients perform minimal computations as the full node is the one to process newly generated blocks for finding relevant transactions. However, this light client implementation suffers from several drawbacks. First, SPV clients may observe low privacy as full nodes can infer what addresses are related to the SPV client. Additionally, since the heavy computation is performed on the full node, it is not very rewarding for the full node. This also makes full nodes vulnerable to DOS (Denial of service) attacks. To improve privacy, SPV clients use Bloom filters with a high rate of false positives to make it harder to infer what addresses are associated with the client.

**Neutrino.** To overcome the privacy drawbacks of SPV light clients, Neutrino clients suggest a different approach: Rather than requesting specific transactions from the full node, light clients download specific blocks, containing the relevant transactions. BIP-157 [44] implements this approach, using Client-Side Block Filtering. Whenever a new block is generated, full nodes create and broadcast a filter of all addresses that appear in the block. When a light client receives a filter, it checks if any of the addresses in the block are relevant. If so, the light client downloads the entire block from the full node and updates its state. BIP-158 [43] suggests using *Golomb-Coded Sets* (GCS) filters since a GCS filter is typically smaller than a Bloom filter with the same amount of elements inserted and the same false positive rate. *Neutrino* [3] is a light client implementation with Client-Side Block Filtering using GCS. Fig. 1b illustrates the process of updating the state of Neutrino clients. Since Neutrino light clients download the entire block and not specific transactions, full nodes have much less information regarding the light client’s addresses. Additionally, as full nodes with Neutrino implementations perform fewer computations than full nodes with SPV implementations, Neutrino full nodes are less vulnerable to DOS attacks. On the other hand, Neutrino requires

more computations on the light client's side when a client checks the relevancy of a received filter. Additionally, the network communication of Neutrino clients is higher than the SPV clients as Neutrino clients download the entire block and not only specific transactions. Moreover, as full nodes know what blocks were downloaded by clients, as we show in this paper Neutrino clients also have privacy concerns. Table 2 summarizes high-level differences between SPV and Neutrino clients.

## 2.3 Related Work

Security and privacy in Bitcoin have been widely discussed in the literature. Conti et al. [18] perform a comprehensive study on the security and privacy of Bitcoin, reviewing de-anonymization methods by analyzing blockchain data. [39] use a forensics perspective, analyzing Bitcoin wallets for iOS and Android, recovering information such as metadata, installation data, timestamps, and usage traces. Multiple papers [21, 9, 58, 40] try to cluster Bitcoin addresses. [32] link between fungibility and anonymity of cryptocurrencies and put forward a framework to measure the fungibility and anonymity of cryptocurrencies, using Shannon entropy. Additionally, [53] uses a Transaction Directed Acyclic Graph (TDAG) to capture blockchain privacy notions (PDAG) and compare Monero and Zcash, the two most prominent privacy-preserving blockchains. All of these papers though, do not focus on light clients. [31] provides a taxonomy of cryptocurrency wallets, including light clients such as SPV and Neutrino, evaluating their performance and security. The privacy evaluation is given at a high level without formalized privacy or empirical evaluation.

The privacy and anonymity of Bitcoin light clients have been discussed widely [23, 29, 57, 8]. The first to analyze and formalize the privacy issues of light clients using Bloom filters was a study by Geravis [23]. Later, in [26], the BIP-37 proposers who implemented Bloom filters in SPV clients expanded on these privacy issues and discussed the difficulties of solving them. [29] further continue the analysis of [23]. Let  $N$  be the total number of addresses inserted in a Bloom filter and  $F$  the number of false positives. [23] show the probability of an address with a Bloom filter positive indication actually belonging to  $c$  is  $p_i = \frac{N}{N+F}$ . Accordingly, the probability of guessing  $j$  addresses with a positive Bloom filter indication and having them all associated with  $c$  is  $\prod_{i=0}^{j-1} \frac{N-i}{N+F-i}$ . We note that in our paper, we base our initial analysis on these probabilities. Similarly, [29] presents a metric called  $\gamma$ -deniability. They refer to a Bloom filter member  $x \in S$  as deniable if for  $i \in \{1, \dots, n\}$  there is a nonmember  $y_i$  such that  $Hash(x) = Hash(y_i)$ . Then, a Bloom filter is  $\gamma$ -deniable if an address is deniable with probability  $\gamma$ . That work indicates that privacy is affected not only by the false positive rate of the Bloom filter but also by the number of real addresses. They describe a method for estimating the number of active addresses through a linear regression model. A similar observation about the efficiency of Bloom filters was described in a more general context [46]. Later, [27] provides an evaluation of the privacy of SPV clients using multiple bloom filters with the  $\gamma$ -deniability metric, in addition to an entropy measurement of the Bloom filters. Unlike SPV, the privacy of Neutrino clients is not discussed much as it is generally considered much higher than SPV [45].

To provide better privacy than SPV, several approaches were suggested, such as Neutrino light clients [3], PIR-based light clients and [16, 45, 56] and using trusted execution environment-based light clients [35, 42, 49, 54], and In this paper, we do not focus on the latter approach since it requires suitable special hardware which makes it unusable for most current Bitcoin users, hence it is rarely implemented in practice.

### 3 Threat Model

In blockchain systems, user privacy is a major concern. However, the relation between Bitcoin transactions and addresses can be used to analyze Bitcoin’s privacy information, seriously jeopardizing Bitcoin anonymity [59]. An adversary may find an association between Bitcoin transactions and addresses using address clustering, further associating groups of addresses with the same entity [25, 28, 59, 52, 33]. Moreover, although the recommendation in Bitcoin is to generate a new address for each transaction, as this results in a large overhead of generating and managing addresses, most Bitcoin users do not generate a new address each time. Similarly to previous work [8, 23, 29, 57], in this paper, the goal of an attacker is to *reveal what addresses typically belong to the wallet of light client  $c$* . This allows the attacker to track every transaction performed by  $c$ , and keep track of  $c$ ’s financial status. Likewise, we assume the light clients are connected to the Bitcoin P2P network, and receive information regarding the network transaction through full Bitcoin nodes. That attacker gains information from the filters used by  $c$  (the Bloom filters for SPV clients and the blocks downloaded for Neutrino clients). Moreover, we assume all relevant filters can be tracked to the same IP address (thus the adversary knows what filters are related).

We argue this model currently represents a real-life scenario. First, for an SPV client  $c$  all the attacker needs is a filter of the addresses of  $c$ , which is sent to any full node  $c$  communicates with. As for Neutrino clients, often a Neutrino client continues to communicate with the same full nodes over and over, allowing them to gain information regarding all of the blocks  $c$  is interested in.  $c$  communicates with the same nodes for several reasons: First, as the process of full node seeking is DNS-similar and might be time-costly, light clients keep a cache of full nodes they discovered, and communicate with cached full nodes rather than search for new full nodes every time. Additionally, as this threat model is passive,  $c$  is not aware of the attacker’s gain of information. An attacker might act as a fast and reliable node, encouraging  $c$  to continue using it rather than search for other full nodes. Moreover, if  $c$  disconnects from the network and later rejoins, to reconstruct the previous state  $c$  might request all relevant blocks at once from the same full node. In addition, Neutrino clients are encouraged to communicate with multiple full nodes and validate the consistency of the data received to lower the risk of data leakage attacks [35]. On top of that,  $c$  might communicate with colluding full nodes, sharing information. Hence, Neutrino clients are at risk of having a full node with information regarding all of the blocks  $c$  was interested in.

### 4 Privacy Metrics

In this section, we present light client privacy evaluation metrics. Previous papers [23, 29] have already analyzed the privacy of SPV clients, though they did not use privacy evaluation methods that can be compared to other light client implementations but SPV. We expand the privacy analysis of SPV clients and present different privacy metrics that can be used to analyze the privacy of Neutrino clients as well. These metrics also allow us to compare the privacy of SPV and Neutrino clients. Although Neutrino clients are thought to have much higher privacy since they download full blocks rather than specific transactions [3, 57, 31], as we show in this paper over time Neutrino clients may suffer from severe privacy issues too. To the best of our knowledge, we are the first to measure the privacy of Neutrino clients. Throughout the paper, the *privacy* of light client  $c$  refers to the situation where the specific details of the account addresses of  $c$  should be hidden from external parties.  $p_i$  denotes the probability that an address  $i$  is among the addresses of interest of client  $c$ . Denote by  $A$  the blockchain address space such that  $c \subseteq A$ . In Section 5 we show how to evaluate  $p_i$  for each address in both SPV and Neutrino clients. We present the two following privacy metrics:

**(i) Light Client Entropy.** Entropy is a metric used to measure the uncertainty or disorder of a dataset or a random variable. The higher the uncertainty regarding each element in the dataset, the higher the entropy gets. When all addresses have the same probability,  $p_i = \frac{|c|}{|A|}$ , it is the most difficult to distinguish between addresses associated with  $c$  and other addresses. Additionally, when some addresses have a higher probability compared to others, there is less uncertainty regarding which addresses belong to  $c$ . Assuming the sum of probabilities  $p_i$  of all addresses in  $A$  is  $|c|$  (as  $c$  has  $|c|$  addresses), the number of addresses needed to cover some probability sum of  $x < |c|$  can indicate the uncertainty of the network. Hence, we define  $T(x)$  as the minimal number of addresses needed to cover some probability sum  $x$ . This minimal number of addresses can be derived based on the addresses with the highest probabilities. On the one hand, a wide range of  $x$  values gives a better indication. On the other hand, lower  $x$  values distinguish addresses better (for instance  $x = |c|$  will always return the number of addresses with  $p_i \neq 0$ , which is less informative). Hence Definition 1, presenting the *light client entropy* of some light client  $c$ , sums  $T(x \cdot |c|)$  for  $x$  values in the range  $[0, 1]$ , and gives higher weights for lower  $x$  values. Finally, a  $\ln$  operation is applied for convenience to scale the entropy (and is not mandatory). We note that addresses that are necessarily not associated with  $c$  (satisfying  $p_i = 0$ ) decrease the value of entropy  $H$  as they often lead to higher probabilities of other addresses being associated with  $c$ , thus decreasing  $T(x \cdot |c|)$ . Moreover, it is easy to see that the minimal entropy value is achieved when there are  $|c|$  addresses with probability  $p_i = 1$  (namely, when all addresses associated with  $c$  are known), as  $T(x \cdot |c|)$  returns the minimal value possible for all  $x$  values.

Our definition of entropy differs from the classic Shannon entropy equation,  $Entropy = -\sum_j p(j) \cdot \log p(j)$ . The entropy values are maximized when the  $p_i$  values get closer to 0.5. As  $p_i$  describes the probability of an address being associated with  $c$ , and since there are many more addresses that are not associated with  $c$  (namely,  $|c| \ll |A| - |c|$ ), lower  $p_i$  values, rather values closer to 0.5, are values indicating higher uncertainty. Hence, the classic Shannon entropy equation (while considering the probabilities for addresses in the address space) is less suitable for measuring uncertainty regarding  $c$ 's addresses.

► **Definition 1.** The *light client entropy* of light client  $c$  is defined as

$$H(c) = \ln \int_0^1 (1-x) \cdot T(x \cdot |c|) dx$$

where  $T(x \cdot |c|)$  is the minimal number of addresses needed to cover a probability sum of  $x \cdot |c|$ .

**(ii) Detection Ratio.** As the number of addresses of  $c$  an adversary can guess correctly is a privacy concern, the detection ratio metric measures the ratio between the number of correctly guessed addresses and the total number of guesses. Assuming an adversary guesses  $|c|$  addresses,  $E_c$  denotes the expected number of correctly guessed addresses.

► **Definition 2.** The *detection ratio*  $R(c)$  of client  $c$  is defined as  $R(c) = \frac{1}{|c|} \cdot E_c$ , assuming out of  $|c|$  addresses guessed, the expected number of correctly guessed addresses is  $E_c$ .

Intuitively, for lower values of the detection ratio  $R(c)$  there is less certainty about which addresses are associated with  $c$ , and the privacy of  $c$  is higher. We present a simple lemma related to the detection ratio.

► **Lemma 3.** The expected number of correctly guessed addresses  $E_c$  equals the sum of probabilities of guessed addresses:  $E_c = \sum_{i \in \Omega} p_i$ , where  $\Omega$  is the set of guessed addresses.

**Proof.** Assume an adversary guesses a set  $\Omega$  of  $|\Omega| = |c|$  addresses. Now, let  $X$  be the number of addresses in  $\Omega$  that were guessed correctly. Let  $X_i$  be an indicator for a guessed address  $i \in \Omega$  to be indeed in  $c$ . In that case,  $E_c = E[X] = E[\sum_{i=1}^n X_i] = \sum_{i=1}^n E[X_i]$ . Since the expectation on an indicator  $i$  is equal to the probability that  $X_i = 1$ , i.e.  $E[X_i] = P(X_i = 1)$ , then  $E_c = \sum_{i \in \Omega} p_i$  equals the sum of the probabilities of the guessed addresses. ◀

By Lemma 3, the value  $E_c$  equals the sum of the probabilities of the guessed addresses. We note the minimal detection ratio is achieved when each address  $i \in A$  has the probability  $p_i = \frac{|c|}{|A|}$  of being associated with  $c$ . In such a case, the detection ratio is  $R(c) = \frac{|c|}{|A|}$ .

We note that while entropy measures the uncertainty on the entire network and is affected by all addresses, the detection ratio measures the more prominent addresses and is affected mainly by the high-probability addresses in the network.

## 5 Privacy Theoretical Analysis

### 5.1 SPV privacy

When evaluating the privacy of SPV clients, the information regarding the addresses of client  $c$  is inferred from the Bloom filter  $c$  creates. As most SPV-based implementations use a constant filter for a long time, the privacy of SPV clients does not change over time since the information regarding the addresses in  $c$  is inferred from the filter alone. We start with a simple property regarding addresses that do not appear in the filter:

► **Property 4.** *Addresses with a negative indication in a filter are not associated with  $c$ .*

Property 4 states that the probability of every address with a negative indication being related to  $c$  is  $p_i = 0$  since  $c$  inserts all of its addresses to the filter that has no false negatives. The probability of address  $i$  with a positive indication belonging to  $c$  was analyzed in [23], and depends on the number of addresses of  $c$  and the number of false positives  $F$ :  $p_i = \frac{|c|}{|c|+F}$ . We take the analysis of SPV privacy a step forward, trying to numerically evaluate its privacy in a way that could be compared to other light client implementations. Lemma 5 evaluates the entropy of light client  $c$ , showing  $H(c) = \ln \frac{|c|+F}{6}$ . Lemma 6 shows the number of addresses guessed correctly out of guessing  $|c|$  addresses of an SPV client is hypergeometric distributed with parameters  $(N, |c|, |c|)$ , allowing Lemma 7 to evaluate the detection ratio of light client  $c$  as  $R(c) = \frac{|c|}{|c|+F}$ .

► **Lemma 5.** *For an SPV client with  $|c|$  addresses and a Bloom filter containing  $F$  false positives, the light client entropy value is  $H(c) = \ln \frac{|c|+F}{6}$ .*

**Proof.** We first note that there are  $|c| + F$  addresses with a positive filter indication, all with probability  $p_i = \frac{|c|}{|c|+F}$  of being associated with  $c$ . By property 4, all other addresses in address space  $A$  have a probability of  $p_i = 0$ . Next, we note that there are  $x \cdot (|c| + F)$  addresses needed to achieve a probability sum of  $x \cdot |c|$ , hence for  $0 \leq x \leq 1$ ,  $T(x \cdot |c|) = x \cdot (|c| + F)$ . Thus,  $H(c) = \ln \int_0^1 (1-x) \cdot T(x \cdot |c|) dx = \ln \int_0^1 (1-x) \cdot x \cdot (|c| + F) dx = \ln \frac{|c|+F}{6}$ . ◀

► **Lemma 6.** *Let  $X$  be the number of addresses guessed correctly out of guessing  $|c|$  addresses of an SPV client (all guesses are of addresses with a positive Bloom filter indication).  $X$  has a hypergeometric distribution with parameters  $(N, |c|, |c|)$ .*

**Proof.** Recall SPV clients create a Bloom filter with a positive indication for  $N = |c| + F$  addresses, out of them  $|c|$  indeed belong to  $c$ .  $X$  is the number of addresses that belong to client  $c$  that are guessed correctly. The guessing order does not matter, and each guess reduces the address space to guess from (since  $|c|$  different addresses are guessed). We note this case

is identical to the classic hypergeometric distribution problem: Given a bin with  $N$  balls, out of them  $n = |c|$  are black and the rest are white. Let  $X$  count the number of black balls drawn (with no returning the balls and with no meaning to the drawing order). Since there are  $|c|$  black balls, the maximal number of black balls drawn is  $D = |c|$ . Therefore,  $X$  has a hypergeometric distribution with parameters  $(N, D, n)$ , that is  $X \sim HG(|c| + F, |c|, |c|)$ . ◀

► **Lemma 7.** *For an SPV client with  $|c|$  addresses and a Bloom filter containing  $F$  false positives, the light client detection ratio is  $R(c) = \frac{|c|}{|c|+F}$ .*

**Proof.** By Lemma 6  $X \sim HG(N, |c|, |c|)$ . Hence, the expected number of correctly guessed addresses is  $E_c = \frac{|c|^2}{|c|+F}$ . Therefore, by the definition of the light client detection ratio  $R(c) = E_c \cdot \frac{1}{|c|} = \frac{|c|^2}{|c|+F} \cdot \frac{1}{|c|} = \frac{|c|}{|c|+F}$ . ◀

## 5.2 Neutrino privacy

As Neutrino clients receive the GCS (Golomb-coded set) filter from a full node and do not download specific transactions but full blocks, it seems like there is much less information that can be inferred by a full node hence Neutrino clients' privacy is generally considered much higher than of SPV [45]. Matetic et al. [35] show that a Neutrino client  $c$  might be at risk of an attack revealing information on its addresses if it receives GCS block filters from a single entity, though privacy is considered high when  $c$  communicates with multiple servers (full nodes) and validates the data is consistent between the servers. *This motivates Neutrino clients to request the same information from multiple servers.* We show there yet exists a major privacy issue if some server knows the exact blocks  $c$  downloaded. Additionally, in contrast to SPV light clients, we show the privacy of Neutrino clients decreases over time. To the best of our knowledge, we are the first to address and formalize this problem. We assume there exists a server with information about what exact blocks  $c$  downloaded and can thus infer what blocks were not downloaded too.

Recall GCS filters have false positives, possibly making clients download blocks without addresses associated with them. Given the probability for the block containing at least one address the client is  $L$ ,  $|c|$  is the number of addresses associated with client  $c$  and an expansion parameter  $M$  for the GCS filter, Lemma 9 presents the probability of this block being false positive for  $c$ . Upon assuming that in address space  $A$  all addresses have the same probability to appear in a block,  $L$  can be evaluated as shown in Lemma 8.

► **Lemma 8.** *Assume each block contains  $K > |c|$  addresses, selected uniformly at random from the address space  $A$ . The probability  $L$  for the block to contain at least one address of client  $c$  is  $L = 1 - \binom{|A|-|c|}{K} / \binom{|A|}{K}$ .*

**Proof.** Given a block, let  $Y$  be the number of addresses associated with client  $c$  that appear in the block. As all addresses have the same probability to appear in the block, when choosing  $K$  addresses (out of them  $|c|$  addresses are associated with  $c$ ) for the block the maximal number of addresses associated with  $c$  that may appear in a block is  $\min\{|c|, K\} = |c|$ . Hence, similarly to  $X$  in Lemma 6,  $Y$  has a hypergeometric distribution, i.e.  $Y \sim HG(|A|, |c|, K)$ . Thus  $P(Y = 0) = \binom{D}{0} \cdot \binom{N-D}{n-0} / \binom{N}{n} = \binom{|A|-|c|}{K} / \binom{|A|}{K}$  and  $L = 1 - P(Y = 0) = 1 - \binom{|A|-|c|}{K} / \binom{|A|}{K}$ . ◀

► **Lemma 9.** *Consider a Neutrino client  $c$  with  $|c|$  addresses and let  $M$  be the expansion parameter of the GCS filter representation of a block. Let  $L$  be the probability of a block containing at least one address of  $c$ . The probability of a block being a false positive block (i.e. being downloaded by  $c$  without actually containing any address of  $c$ ) is*

$$p_{FP} = (1 - L) \cdot \left(1 - \left(1 - \frac{1}{M}\right)^{|c|}\right) = \binom{|A| - |c|}{K} / \binom{|A|}{K} \cdot \left(1 - \left(1 - \frac{1}{M}\right)^{|c|}\right).$$

## 15:10 Privacy Comparison for Bitcoin Light Client Implementations

**Proof.** For a block to be a false positive for client  $c$ , two conditions must be met:

- (i) The block does not contain any address of  $c$  (event  $B$ ).
- (ii) Client  $c$  has at least one address with a positive filter indication (event  $D$ ).

As  $L$  is the probability that at least one address associated with  $c$  will appear in the block, the probability for the first condition to be met is  $P(B) = 1 - L$ . We compute the probability for event  $D$  given that event  $B$  holds. Recall the probability of a non-associated address to have a positive filter indication is  $\frac{1}{M}$ , implying that the probability of a non-associated address having a false filter indication is  $1 - \frac{1}{M}$ . Since client  $c$  has  $|c|$  addresses, the probability that all of addresses in  $c$  have a false filter indication is  $(1 - \frac{1}{M})^{|c|}$  and at least one address has a positive filter indication with probability  $P(D|B) = (1 - (1 - \frac{1}{M})^{|c|})$ . Thus,  $p_{FP} = P(B) \cdot P(D|B) = (1 - L) \cdot (1 - (1 - \frac{1}{M})^{|c|})$ . ◀

Given that a block was downloaded by a client and is not a false positive, the probability of guessing a single address is  $p_i = \frac{1}{K}$  where  $K$  is the number of addresses in a block. Since an address appearing on a block may belong to  $c$  only if the block is not a false positive, Property 10 is intuitive:

▶ **Property 10.** *Given a block with  $K$  addresses that was downloaded by  $c$ , with one address that belongs to  $c$  and with no other knowledge, the probability of each address being associated with  $c$  is  $p = (1 - p_{FP}) \cdot \frac{1}{K}$ .*

Similarly to Property 4, a full node can infer that addresses in blocks not downloaded by  $c$  are not associated with it, as stated in Property 11. For every such address, the probability of being associated with  $c$  is  $p = 0$ . An address in a block downloaded by  $c$  that also appears in an earlier block not downloaded by  $c$  has a probability of  $p = 0$ .

▶ **Property 11.** *Addresses appearing in blocks the light client did not download are not associated with the client.*

▶ **Definition 12.** *For some light client  $c$  and a block  $B$  downloaded by  $c$ , **non-eliminated addresses** are addresses that appear in  $B$  and do not appear in any earlier block that was not downloaded by  $c$ .*

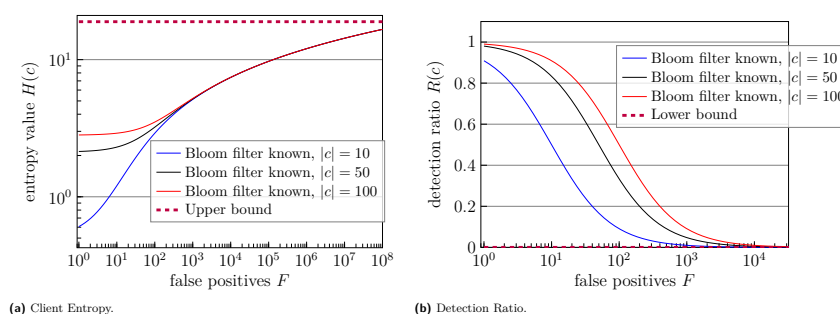
Therefore, Property 13 states that only the amount of *non-eliminated addresses*  $k$  should be considered when evaluating  $p_i$ .

▶ **Property 13.** *Given a block downloaded by  $c$ , with one address that belongs to  $c$ , knowing the block contains  $k$  non-eliminated addresses, the probability of each address being associated with  $c$  is  $p = (1 - p_{FP}) \cdot \frac{1}{k}$ .*

## 6 Light Client Privacy Measurement

After presenting the privacy analysis of each implementation, we now evaluate and compare the analysis of the different implementations. We examine the main parameters affecting the privacy of each implementation and later empirically evaluate the privacy based on real Bitcoin data. As Bitcoin uses the RIPEMD-160 [20] hash function, the number of possible addresses is  $2^{160}$ . However, in practice, the Bitcoin blockchain consists of about a billion addresses as of April 2023. Hence, the address space is of size  $|A| = 10^9$ . If only the number of addresses  $|c|$  is known,  $p_i = \frac{|c|}{|A|}$ . As all addresses have the same  $p_i$ , this case is equivalent to having a filter using  $F = |A| - |c|$  addresses, and by Lemmas 5 and 7 the entropy is





■ **Figure 2** (a) The entropy of an SPV client  $c$  compared to the number of positives. The graph also contains the entropy values of  $C$  when the Bloom filter is unknown. (b) The number of addresses and the detection ratio compared to the number of positives. For convenience, the detection ratio is presented as a percentage (scaled by 100). Both graphs are calculated for light clients with having  $|c| = 10, 50$  and  $100$  addresses.

$H(c) = \ln \frac{|A|-|c|+|c|}{6} = \ln \frac{10^9}{6} = 18.93$ , and the detection ratio is  $R(c) = \frac{1}{\lfloor 10^9 \rfloor}$ . We now show that by receiving additional information from the client, like a Bloom filter (SPV) client and what blocks  $c$  downloaded (Neutrino) the privacy decreases.

## 6.1 Data

To evaluate the privacy of the different implementations on real-life data, we downloaded all mined Bitcoin blocks during April 2023. A total of 4161 blocks were mined, averaging 137 mined blocks per day. Though the maximal size of a Bitcoin block is 4 MB, the average block size was around 3.2 MB. Since achieving information regarding real-life wallets and the addresses associated with them is not an easy task (as Bitcoin wallets are private), to simulate a Neutrino client wallet, we sampled random addresses appearing in the blocks mined on April 1st for wallets of size 10, 50 and 100. In each analysis, the address sampling and privacy measuring were performed 100 times to neutralize noises. As there is no simple formula for calculating the entropy of a Neutrino client, to evaluate the entropy of Neutrino clients in Section 6.3, we sorted all of the probabilities in the network, implemented  $T(x)$  and approximated the entropy value using calculating  $H(c) \approx \ln \frac{1}{1000} \cdot \sum_{x \in [0.001, 0.002, \dots, 1]} (1-x) \cdot T(x \cdot |c|)$ . To assure uniformity between all entropy measurements, SPV entropy was calculated using both the formula presented in Lemma 5 (derived from Definition 1) and by implementing  $T(x)$  and calculating the value of  $\ln \frac{1}{1000} \cdot \sum_{x \in [0.001, 0.002, \dots, 1]} (1-x) \cdot T(x \cdot |c|)$ . For all SPV entropy experiments, the entropy values of both calculations were at least 99.999% similar, showing the sampling of the entropy evaluation of  $\ln \frac{1}{1000} \cdot \sum_{x \in [0.001, 0.002, \dots, 1]} (1-x) \cdot T(x \cdot |c|)$  provides a close enough approximation.

## 6.2 SPV Measurement

For SPV light clients the main parameter affecting privacy is the number of positives  $F$  in the Bloom filter. Fig. 2a presents the entropy of client  $c$  compared to the number of positives, for different values of  $|c|$ . The figure additionally contains an upper bound of the privacy metrics which is achieved when the adversary does not have the client Bloom filter. The larger  $F$  is, the higher the entropy is. For instance, for a client with  $|c| = 50$  addresses,  $F = 104$  false positives imply an entropy of  $H(c) = 3.25$ . The entropy increases to  $H(c) = 7.51$  for a larger amount of false positives,  $F = 10^5$ . Up to some point (in our case around  $F = 40 \cdot 10^3$ ), an entropy increase can be achieved not only by increasing  $F$ , but also by increasing the

number of addresses used by  $c$ . For instance, for  $|c| = 10$  addresses and  $F = 500$  the entropy is equal to  $H(c) = 4.52$ , while for the same amount of false positives, the entropy increases to  $H(c) = 4.69$  if  $c$  uses  $|c| = 100$  addresses instead. This is because as entropy measures the uncertainty in the network, it is affected by all addresses, and the more addresses with  $p_i \neq 0$  there are, the higher entropy gets. More specifically, for SPV clients it is very easy to see from Lemma 5 how the number of addresses in the filter affects entropy  $H(c)$ , hence for larger  $|c|$  values the filter contains more addresses and  $H(c)$  increases accordingly. However, from around  $F = 40 \cdot 10^3$  all three sizes of  $|c|$  achieve similar entropy, as the size of  $c$  becomes quite negligible compared to  $F$ .

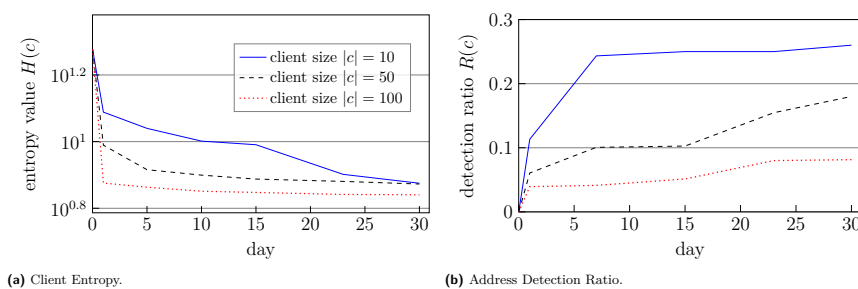
Fig. 2b presents the address detection ratio of client  $c$  compared to the number of positives. As expected, the detection ratio decreases as  $F$  increases. Though, unlike the entropy, more addresses associated with  $c$  result in a higher detection ratio: Having  $F = 10$  false positives, the detection ratio is  $R(c) = 0.5$  if  $c$  uses  $|c| = 10$  addresses, namely 50% of the guessed addresses are associated with  $c$ . For  $|c| = 100$  the detection ratio is  $R(c) = 0.91$ . To achieve a detection ratio of  $R(c) = 0.5$  an amount of  $F = 100$  false positives is needed. This is because the detection ratio is affected mainly by the probability  $p_i$  of each address, and for the same amount of false positives used in a filter, higher  $c$  values have a higher percentage of addresses that belong to  $c$  in the filter, hence probability  $p_i$  for address increases and as a result  $R(c)$  increases too. Similar to the entropy, when using around  $F = 40 \cdot 10^3$  false positives,  $|c|$  becomes relatively negligible compared to  $F$ , in addition to a very low detection ratio, hence the differences between the different  $|c|$  values become very small.

### 6.3 Neutrino Measurement

For Neutrino light clients, privacy is affected mainly by the block size, the amount of non-eliminated addresses  $k$  in each block, the number of blocks downloaded by  $c$  and the number of blocks each address appears in. Since the privacy of Neutrino clients depends on various parameters and the probability of an address being associated with  $c$  varies between the different addresses, Neutrino privacy evaluation is more difficult than SPV clients. We now show that although Neutrino clients are considered to have high privacy, they may suffer privacy issues over time.

Following the analysis in Section 5.2, we analyze the probability of an address appearing in  $z$  blocks being associated with  $c$ . Intuitively, when the false-positive ratio is low, the more blocks an address appears in the higher the chances it is associated with  $c$ . Assuming (for simplicity)  $c$  is interested in (at most) one address from each downloaded block, since in a non false-positive block the probability of a non-eliminated address not being associated with client  $c$  is  $p_i = \frac{k-1}{k}$  (for a false positive block  $p_i = 0$ ). This is assuming independence between blocks, meaning the probability that an address that appears in  $z$  blocks is not associated with  $c$  is  $P = \prod_{i=1}^z \frac{k_i-1}{k_i}$ , where  $k_i$  is the number of non-eliminated addresses and 1 is the number of addresses associated with  $c$  in a non false-positive block. Hence, if all blocks are non false-positive the probability that the address is associated with  $c$  is  $p = 1 - \prod_{i=1}^z \frac{k_i-1}{k_i}$ . Since the probability for  $z$  blocks not to be false positive is  $P = (1 - p_{FP})^z$ , the probability of a non-eliminated address appearing in  $z$  blocks downloaded by  $c$  being associated with  $c$  is  $p_i^z(c) = (1 - p_{FP})^z \cdot \left(1 - \prod_{i=1}^z \frac{k_i-1}{k_i}\right)$ .

Fig. 3 presents the average entropy and detection ratio over time. We consider probability  $p = 0$  for eliminated addresses, and for each non-eliminated address  $i \in A'$  that did not appear yet in a block, we consider probability  $p = \frac{|c|}{|A'|}$ , where  $A'$  is the address space excluding all eliminated addresses. When evaluating the entropy, all address probabilities were normalized to ensure the sum of probabilities equals  $c$ .



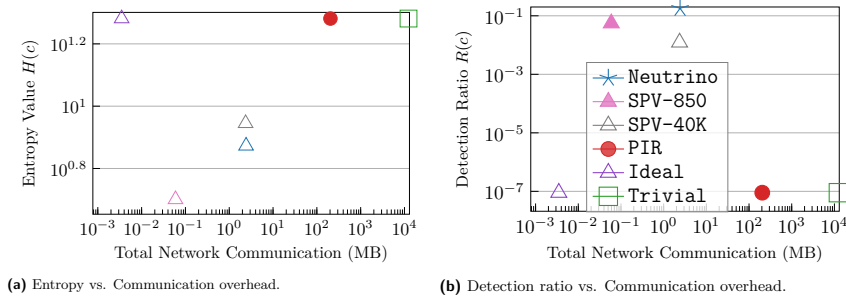
■ **Figure 3** (a) Entropy of a Neutrino client  $c$  over a month. (b) Detection ratio of a Neutrino client  $c$  over a month. Both graphs were calculated for client size of  $|c| = 10, 50$  and  $100$  addresses.

Fig. 3a shows the more blocks are mined the more information there is regarding the addresses of  $c$  and the entropy decreases. For instance, while for  $|c| = 50$  the entropy was  $H(c) = 9.75$  at the end of the first day (after 139 mined blocks), by the 30th day the entropy decreased to  $H(c) = 7.46$ . This indicates that the more blocks are mined the more certainty there is regarding what addresses belong to  $c$ . We additionally notice there is a large drop in entropy on the first day. This is because before a full node has any information on  $c$ , all addresses are candidates of being associated with  $c$ . After a day, many addresses become eliminated addresses hence  $|A'|$  decreases by much, and as the entropy is affected by all addresses, eliminating that many addresses decreases the entropy by much. Moreover, we see the slopes of the graphs become milder over time. This is because over time there are fewer and fewer addresses that are eliminated, hence the entropy decreases slower over time.

The detection ratio evaluation shows how the detection ratio increases over time too, indicating a privacy decrease. As shown in Fig. 3b, after one day the detection ratio for  $|c| = 10$  was  $R(c) = 0.11$ , meaning an adversary could identify correctly an average of 11% of the addresses of  $c$ . By the 30th day, the detection ratio increased to  $R(c) = 0.26$ . Unlike the entropy, there is a difference in the detection ratio for different  $|c|$  values, as the more addresses  $c$  uses the lower the chances of guessing correctly the addresses that belong to  $c$ . For instance, on the 30th day with  $|c| = 100$  the detection ratio is only  $R(c) = 0.08$ , more than three times lower than  $|c| = 10$ . Moreover, the graphs show that for each day, for larger  $c$  values the entropy and detection ratio have lower values, whereas for smaller  $c$  values they increase. This is because larger  $c$  values result in downloading more blocks. The main addresses that contribute to the entropy values are the non-eliminated addresses that did not appear in any block. Hence, as for larger  $c$  values more blocks are downloaded, fewer addresses have not appeared yet in a block and thus the entropy is relatively larger. As the detection ratio is mostly affected by unique addresses that appear in blocks  $c$  downloaded more than others, when more blocks are downloaded the higher the chances of addresses appearing, thus reducing the chances of unique addresses that significantly appear more than other addresses. To conclude, for all  $c$  values the entropy decreases over time while the detection ratio increases, both indicating a privacy decrease over time in Neutrino.

## 6.4 Privacy and Communication Overhead

We now analyze the privacy and network communication of SPV and Neutrino. Additionally, we compare this analysis to a PIR light client, which provides maximal privacy. *Private Information Retrieval* (PIR) protocols, introduced by [17], allow clients to query a server and retrieve data from the server's database without revealing information regarding the data the client was interested in. There have been several suggestions for light client implementations using PIR [55, 56, 45]. We note that PIR implementations require a change in the current



■ **Figure 4** Privacy compared to communication overhead comparison between different light client implementations, for a client with  $|c| = 50$  addresses. (a) Entropy vs. communication overhead. (b) Detection ratio vs. communication overhead.

Bitcoin full node protocol, in addition to much higher increased computational requirements on both the server and clients. As light clients should stay light (usually running on devices with small computation power), they are not yet popular as a light client solution. Hence, we do not focus on PIR client implementations and use them as a solution providing ideal privacy while improving the trivial solution of downloading the entire blockchain. We analyze the PIR-based light client protocol presented in [45], provided in Percy++ [2], an open-source library. Since we assume the privacy of PIR-based clients is ideal, the probability  $p_i$  of each address for PIR implementations is  $p_i = \frac{|c|}{|A|}$  with  $|A|$  addresses that appear in the blockchain.

In our analysis, we generate clients with  $|c| = 50$  random addresses similarly to Section 6.3. We note the communication overhead of the filters used by SPV and Neutrino clients is negligible compared to the downloaded transactions' data. Fig. 4 presents the privacy of the different implementations, in addition to the *Trivial* (downloading the entire blockchain) and the *Ideal* implementations (achieving maximal privacy with downloading minimal data), compared to the communication overhead after 30 days. For the *SPV* implementation, we used two filter sizes with  $F = 850$  and  $F = 40 \cdot 10^3$  false positives, denoted as *SPV-850* and *SPV-40K*, respectively. We observe in Fig. 4a that the *Ideal*, *Trivial* and *PIR* solutions reach an entropy of  $H(c) = 18.91$ . While *Trivial* requires downloading the entire blockchain of size 12.3 GB, *PIR* reduces this overhead to 204 MB. An ideal solution, though, would require only 0.004 MB. *SPV-40K* achieves an entropy of  $H(c) = 8.81$  for only 2.36 MB, and *Neutrino* achieves an entropy of  $H(c) = 7.46$ , higher only than *SPV-850*, having an overhead of 2.4 MB. The lowest entropy implementation was *SPV-850* with an entropy of  $H(c) = 5.02$ , though having a small overhead of 0.06 MB. Fig. 4b presents the detection ratio of each solution. As we have already seen in Fig. 3, the detection ratio of *Neutrino* increases over time, making it the highest detection ratio implementation, with  $R(c) = 0.18$  after 30 days. *SPV-850* and *SPV-40K* achieve a detection ratio of  $R(c) = 0.055$  and  $0.012$ , respectively, while *PIR* has a detection ratio of approximately zero.

## 6.5 Discussion: Additional Insights from the Measurement and Analysis

We now compare the privacy of the light client implementations, suggesting insights to improve privacy in the existing light client implementations.

### 6.5.1 Privacy Comparison

Subsection 6.4 shows that in general, there is a privacy and communication overhead tradeoff for light clients. For example, *PIR* provides better privacy than *SPV* and *Neutrino*, though with a much higher communication overhead. That said, several implementations are

comparable to others and some light clients perform better than others. Our analysis shows that, unlike the common conception that Neutrino clients preserve more privacy compared to SPV, under our threat model, *SPV clients provide better privacy* than Neutrino: While SPV-40K has the same communication overhead as Neutrino, it provides a higher entropy value and a detection ratio almost 15 times lower than Neutrino clients. SPV clients have an additional advantage over other implementations, as there is only one main parameter that determines privacy and communication overhead- the number of false positives  $F$ . This allows SPV clients the opportunity to easily tune  $F$  based on what is more important for their use- privacy or low network communication. Similarly, PIR light clients achieve similar privacy as Trivial, having a much lower communication overhead, and can thus perform better than SPV in the extreme case  $c$  wants maximal privacy.

## 6.5.2 Improving Light Client Privacy

Both SPV and Neutrino clients can improve their privacy by increasing the false-positive rate of their filters. This results in additional transactions downloaded in SPV, and additional blocks downloaded for Neutrino clients. We now suggest other efficient techniques to improve light client privacy.

**Neutrino.** Recall by Section 6.3 that the probability  $p_i$  of address  $i$  being associated with  $c$  mainly depends on two parameters:  $z$ , the number of blocks an address appears in, and  $k$ , the number of non-eliminated addresses. Increasing the number of addresses used by  $c$  and thus decreasing  $z$  for many addresses will result in higher entropy and lower detection ratio, indicating privacy improvement without any communication increase. Additionally, we suggest methods for decreasing the chances of our threat model occurring. Recall Neutrino clients are motivated to request information from multiple servers to reduce changes of some privacy attacks [35]. We suggest Neutrino light clients should both *limit themselves to a non-large amount of full nodes* and additionally *divide the blocks download between multiple servers* rather than querying the same full nodes every time, especially when rejoining the network. Moreover, we suggest Neutrino clients should keep track of the full nodes they approached for information, and try being as diverse as possible when choosing a full node to communicate with. In addition, occasionally clearing the full node cache will help avoid reaching the same full nodes every time.

**SPV.** The main parameter determining the privacy of SPV clients is the number of false positives,  $F$ . Hence, besides increasing  $F$ , section 7 suggests an SPV-based light client model, that improves privacy for a similar communication overhead.

## 7 Proposal: The Aggregation Model (AM)

### 7.1 Overview

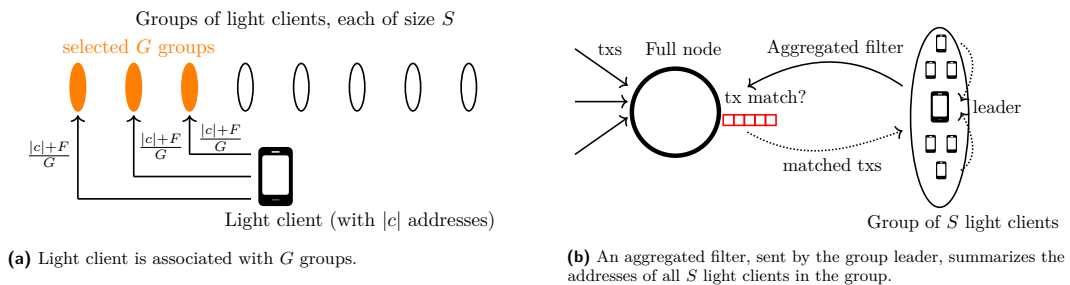
As existing light client implementations suffer from privacy issues, we suggest a new SPV-based light client model, *the aggregation model (AM)*, to potentially improve the privacy of light clients, for a similar communication overhead. In short, we suggest light clients should group up with other light clients, create an aggregated filter for all the clients and have one client representative that communicates with the full node. We suggest light client  $c$  should join  $G$  groups (each of size  $s$  light clients), equally split its addresses between these groups and additionally add some false positives to each group filter.

## 15:16 Privacy Comparison for Bitcoin Light Client Implementations

We assume each light client  $c$  uses  $F$  false positives. Additionally,  $c$  joins  $G$  groups, each group of size  $S$ , meaning there are  $S$  light clients in each group. The following steps are performed when a light client wants to download transactions from a full node:

1.  $c$  creates  $G$  non-intersecting filters, composed of  $\frac{|c|}{G}$  addresses of  $c$  and  $\frac{F}{G}$  false positives.
2.  $c$  joins  $G$  groups, each of size  $S$ .
3. For each group, the following is performed:
  - a. A leader  $l$  is randomly chosen (this can be done using several distributed algorithms such as [7, 30, 38]).
  - b. All clients send  $l$  their (bloom) filters.
  - c.  $l$  creates an aggregated Bloom filter and sends it to a full node.
  - d.  $l$  receives transactions from the full node and sends each client its transactions (with their Merkle-proofs), as received from the full node.

The implementation is illustrated in Fig. 5.



■ **Figure 5** Illustration of the aggregation model (a) Light client is associated with  $G$  groups and sends  $\frac{|c|+F}{G}$  addresses to each group. (b) An aggregated filter, sent by the group leader, summarizes the addresses of all  $S$  light clients in the group.

When a new Bitcoin node connects to the network, it queries several DNS servers (which are operated by volunteer nodes and provide a random selection of bootstrap nodes that are active in the Bitcoin network). Once connected, the joining node learns about other nodes by asking their neighbors for known addresses and listening for spontaneous advertisements of new addresses [19]. Hence, we suggest light clients advantage of this mechanism to form  $G$  groups of  $S$  clients each. We note the main advantage of the aggregation model is that only the leader exposes itself to a full node, providing anonymity for other clients. This way light clients receive transactions without any full node gathering information about non-leader clients. Additionally, even at the worst scenario, where all nodes in the groups  $c$  participates in collude with full nodes, using the aggregation model *the privacy of light client  $c$  is at least as high as the original SPV model*.

### 7.2 Privacy analysis

In the aggregated model, both the full nodes and the leader gain information regarding the light client addresses. As long as there are no colluding light clients, full nodes cannot infer anything about the group light clients besides information concerning the leader. Hence, assuming there are no colluding clients in the network, when evaluating the privacy of light client  $c$ , the addresses in address space  $A$  can be separated into two groups: addresses in filters of groups where  $c$  is the leader and all other addresses. We assume there are  $G$  groups that  $c$  participates in, each of size  $S$ , and all clients have the same amount of addresses  $|c|$  and use the same amount of false positives  $F$ , equally split between  $G$  groups. Property 14 calculates the expected number of leader groups. For each such group, Property 15 evaluates

the number of addresses in the aggregated filter  $c$  creates and the probability  $p_i$  of each address in the filter being associated with  $c$ . As address space  $A$  is much larger than  $|c|$ , Property 16 evaluated the total number of addresses associated with  $c$  in groups where  $c$  is the leader, and Property 17 evaluates the probability  $p_i$  of each address that does not appear in a group  $c$  is the leader of being associated with  $c$ .

► **Property 14.** *Given there are  $G$  groups  $c$  participates in, each of size  $S$ . The expected number of groups  $c$  will be the leader of is  $\frac{G}{S}$ .*

► **Property 15.** *Given there are  $G$  groups  $c$  participates in, each of size  $S$ , assuming all clients have the same amount of addresses  $|c|$  and use the same amount of false positives  $F$ , equally split between  $G$  groups. In this case, the group leader creates an aggregated filter containing  $\frac{|c| \cdot S}{G} + \frac{F \cdot S}{G} = \frac{(|c|+F) \cdot S}{G}$  addresses. As the group leader  $l$  has  $\frac{|c|}{G}$  addresses on the aggregated filter associated with him, the probability  $p_i$  of any address in the aggregated filter being associated with  $l$  is  $\frac{|c|}{G} / \frac{(|c|+F) \cdot S}{G} = \frac{|c|}{(|c|+F) \cdot S}$ .*

► **Property 16.** *Assume the addresses in  $c$  are equally split between  $G$  groups, each of size  $S$ . As  $c$  is the leader of  $\frac{G}{S}$  groups (Property 14) and there are  $\frac{|c|}{G}$  addresses associated with  $c$  in each group, the total number of addresses associated with  $c$  in groups  $c$  is the leader of is  $\frac{G}{S} \cdot \frac{|c|}{G} = \frac{|c|}{S}$ .*

► **Property 17.** *Assuming there are  $|c'| \leq |c|$  addresses that appear in groups  $c$  is the leader of ( $c'$  is evaluated in Property 16), and assuming  $|A| \gg |c|$ , when there are no colluding light clients the probability  $p_i$  of an address that did not appear in a group  $c$  is the leader of is  $\frac{|c'|}{|A|}$ .*

We note that the previous analysis relies on the light client being honest. However, some light clients might be adversaries too, and collude with each other or with a full node. We denote by  $p_{colluding}$  the percentage of colluding light clients in the network (or similarly the probability of a light client colluding). In such a case, the addresses in address space  $A$  can be separated into three groups: addresses in filters of groups where  $c$  is the leader, addresses in groups where  $c$  is not the leader and leader  $l$  is colluding, and all other addresses. Property 18 calculates the expected number of groups with a colluding leader, and Property 19 evaluates the probability  $p_i$  of all addresses used by  $c$  in this group. Similar to Property 17, assuming there are  $|c'| < |c|$  addresses of  $c$  that appear in groups  $c$  is the leader of (evaluated in Property 16), and  $|c''| < |c|$  addresses of  $c$  that appear in a group with a colluding leader (evaluated in Property 20), the probability  $p_i$  of any other address is  $\frac{|c|-|c'|-|c''|}{A}$ . We note we assume here the strictest colluding assumption, where some full node has the entire information gained by all colluding nodes.

► **Property 18.** *Given there are  $G$  groups  $c$  participates in, each of size  $S$ , and a colluding probability of  $p_{colluding}$  for each node that is not  $c$ . The number of groups  $c$  participates in with a colluding leader is  $p_{colluding} \cdot \frac{G \cdot (S-1)}{S}$ .*

► **Property 19.** *Assume client  $c$  has  $|c|$  addresses and uses a total of  $F$  false positives, equally split between  $G$  groups. For each group that  $c$  is not the leader of,  $c$  sends a filter to the leader. The probability  $p_i$  of each address in the filter being associated with  $|c|$  is  $p_i = \frac{|c|}{G} / \frac{|c|+F}{G} = \frac{|c|}{|c|+F}$ .*

► **Property 20.** *Assume the addresses in  $c$  are equally split between  $G$  groups, each of size  $S$ . As by Property 18 the number of groups  $c$  is not the leader of is  $G \cdot \frac{S-1}{S}$ , each having  $\frac{|c|}{G}$  addresses associated with  $c$ . With a probability of  $p_{colluding}$  of the leader colluding, the number of addresses associated with  $c$  in groups where  $c$  is not the leader and there is a colluding leader is  $p_{colluding} \cdot G \cdot \frac{|c|}{G} \cdot \frac{S-1}{S} = p_{colluding} \cdot |c| \cdot \frac{S-1}{S}$ .*

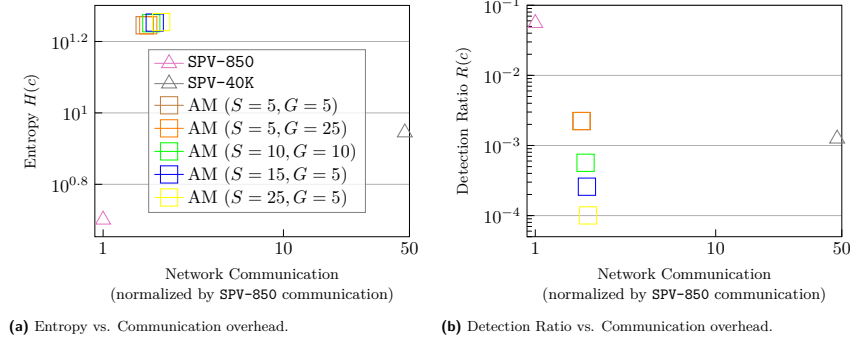
► **Property 21.** *When all nodes communicating with  $c$  are colluding, by Property 19 the probability  $p_i$  of each address used in a group  $c$  is not the leader of is  $p_i = \frac{|c|}{|c|+F}$ . Additionally, for groups  $c$  is the leader of, as all addresses that do not belong to  $c$  are known to the leader, the probability of each address being associated with  $c$  is  $p_i = \frac{|c|}{G} / \frac{|c|+F}{G} = \frac{|c|}{|c|+F}$ . Thus, as  $c$  uses  $|c| + F$  addresses in total each with probability  $\frac{|c|}{|c|+F}$  being associated with  $c$ , the privacy when all nodes collude equals the privacy of the original SPV model using  $F$  false positives.*

Using these properties, we evaluated the entropy and detection ratio of the aggregated transaction proposed model, similar to the previous privacy evaluation of the other light client implementations. In the entropy calculation evaluation, we evaluated the probability of each address in the network, implemented  $T(x)$  and calculated  $H(c) \approx \ln \frac{1}{1000} \cdot \sum_{x \in [0.001, \dots, 1]} (1 - x) \cdot T(x \cdot |c|)$ . Fig. 6 evaluates the privacy of the aggregation model using several  $S$  and  $G$  parameters, compared to the communication overhead (normalized by the communication overhead of SPV-850) assuming there are no colluding nodes. Additionally, the entropy and detection ratio of SPV-850 and SPV-40K are added to show the privacy improvement. Fig. 6a presents the entropy compared to the communication overhead. As we see, the aggregation model using all  $S$  and  $G$  parameters has a similar entropy of around  $H(c) = 17.5$ , while having a communication overhead 1.8 – 1.96 times higher than SPV-850, and 25 times lower than SPV-40K. Recall, SPV-850 and SPV-40K have an entropy of  $H(c) = 8.81$  and  $H(c) = 5.02$ , respectively. The communication network overhead is derived directly from the ratio  $\frac{G}{S}$ , as the communication overhead increases when the number of groups  $c$  is the leader of increases. The entropy of the aggregation model is very similar for all  $S, G$  parameters since the entropy evaluates the knowledge regarding all addresses, and when assuming there are no colluding nodes a full node does not have information about all of the filters used by  $c$ , hence all addresses are eligible to be associated with  $c$ . The increase in the probabilities of the addresses that appear in groups  $c$  is the leader of (and a full node has information about) are relatively negligible to all other addresses. Hence, the entropy values are high and similar for all  $S, G$  values.

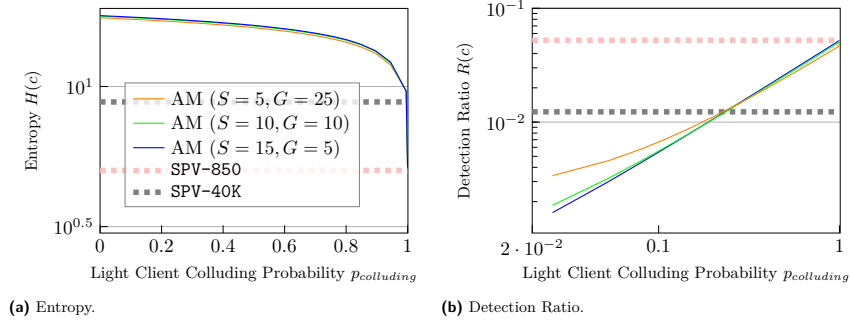
In the detection ratio, shown in Fig. 6b, there are some differences between parameters. While for using  $S, G = (5, 5)$  the detection ratio is  $R(c) = 0.0022$ , having a communication overhead 1.8 times higher than SPV-850, for  $S, G = (25, 5)$  the detection ratio decreases to  $R(c) = 0.0001$ , though with a communication overhead 1.96 times higher than SPV-850. As the detection ratio is mainly affected by the high-probability addresses a full node knows about (rather than of all addresses), the differences in the detection ratio values are derived from the relation between  $S$  and  $G$ , which affects the probability of  $c$  being the leader of the group. The lower the ratio  $\frac{G}{S}$  is, the lower the expected number of times  $c$  is a leader of the group, and the detection ratio decreases accordingly.

Fig. 7 presents the entropy and detection ratio of the aggregation model, using  $S, G = (5, 25), (10, 10), (15, 5)$ , compared to  $p_{colluding}$ , the probability the leader colluding with full nodes. While all three parameter sets have a similar entropy, using  $S, G = (5, 15)$  achieves higher entropy for lower  $p_{colluding}$  values: As for  $p_{colluding} = 0.05$   $S, G = (5, 15)$  has an entropy of  $H(c) = 17.79$ , the entropy of  $S, G = (10, 10)$  and  $S, G = (5, 25)$  is  $H(c) = 17.71$  and  $H(c) = 17.49$ , respectively. This is because for smaller  $\frac{G}{S}$  values the expected number of groups  $c$  is the leader of is lower, hence  $c$  is exposed to the full node less times. Yet, recall by Properties 15 and 19, when  $c$  is the leader a full node receives a filter with more addresses than when  $c$  is not the leader and the leader colludes. Hence, when  $p_{colluding}$  is high, full nodes get more information from groups  $c$  is not the leader of, thus the entropy is higher when  $c$  is the leader of more groups, i.e. when  $\frac{G}{S}$  is larger. For instance, when  $p_{colluding} = 1$ ,  $H(c) = 9.65$  for  $S, G = (5, 25)$ , compared to  $H(c) = 9.63$  and  $H(c) = 9.62$  for  $S, G = (10, 10)$





■ **Figure 6** The entropy and detection ratio of SPV-850, SPV-40K and the aggregation model (AM) using  $S, G = (5, 5), (10, 10), (15, 5), (25, 5), (5, 25)$ , compared to the network communication overhead normalized by the communication of SPV-850, assuming there are no colluding nodes.



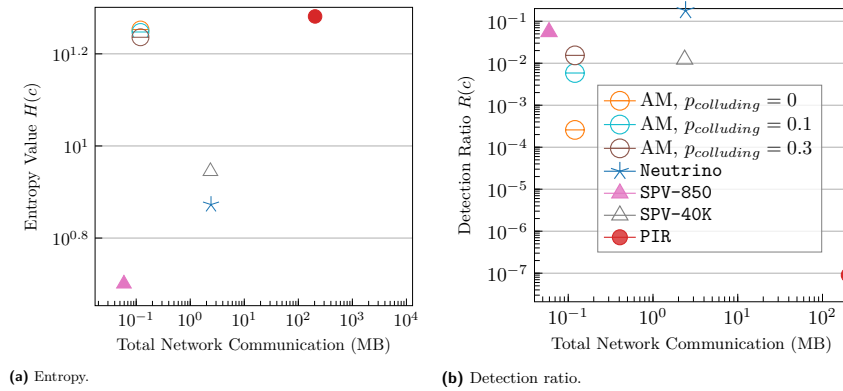
■ **Figure 7** The entropy and detection ratio of the aggregation model (AM) using  $S, G = (10, 10), (15, 5), (5, 25)$ , compared to the light client colluding probability  $p_{colluding}$ .

and  $(15, 5)$ , respectively. We note there is an entropy dropdown when  $p_{colluding} = 1$  to  $H(c) = 5.02$ , as when  $p_{colluding} = 1$  all addresses used in filters  $c$  sent are known, hence the address space  $A$  is decreased to only 900 addresses (the number of addresses used in filters of  $c$ ).

The detection ratio increases too as  $p_{colluding}$  increases. As for  $S, G = (10, 10)$  the detection ratio is  $R(c) = 0.0006$  when there are no colluding nodes, when 50% of the nodes collude the detection ratio increases to  $R(c) = 0.026$ , and when all nodes collude the detection ratio increases to  $R(c) = 0.05$ , similar to the detection ratio of SPV using 950 false positives (as derived by Lemma 7). For the same reasons as the entropy, for small  $p_{colluding}$  values, the detection ratio is lower (implying better privacy) when  $\frac{G}{S}$  is smaller, and for a high  $p_{colluding}$ , the detection ratio is lower when  $\frac{G}{S}$  is larger. Finally, following Property 21, the figure shows the privacy of the aggregation model is always *at least as good as the privacy of SPV-850*, as the entropy is higher and the detection ratio is lower. We additionally note that compared to SPV-40K, the entropy of AM was higher except for the case where  $p_{colluding} = 1$ . Additionally, when  $p_{colluding} \leq 0.23$ , the detection ratio of SPV-40K is higher than the AM.

### 7.3 Aggregation Model Discussion and Limitations

Section 7.2 shows the great potential of the aggregation model, as a privacy increase is achieved with very low communication overhead. Fig. 8 summarizes the entropy and detection ratio of the different implementations compared to the communication overhead. The figure includes



■ **Figure 8** Privacy compared to communication overhead comparison between the different light client implementations and the *AM* model using ( $S = 15, G = 5$ ) and  $F = 850$  false positives.

*AM* using parameters ( $S = 15, G = 5$ ) and  $F = 850$  false positives, for  $p_{colluding} = 0, 0.1, 0.3$ . Though the *AM* communication overhead is 20 times lower than *SPV-40K*, for all three  $p_{colluding}$  values, the entropy is higher. Moreover, for  $p_{colluding} = 0, 0.1$ , the detection ratio was lower too. Assuming the percentage of colluding nodes is lower than 23%, the *AM* model is only second to the *PIR* model in its privacy (having a communication overhead 1700 times smaller), with a communication overhead of approximately only twice the communication overhead of *SPV-850*.

We acknowledge the drawbacks of this model, as it might suffer from a time delay when creating the client groups, and when waiting for an answer from the leader. Moreover, in the scenario where all clients participating in groups with  $c$  are colluding, there is no privacy improvement. That said, the aggregation model, the best advantage of the aggregation model is it does not require any changes in the Bitcoin network protocols, and can work side-by-side with the original *SPV* protocol and full nodes. Clients preferring time over privacy can always use the original *SPV* protocol, as at any point light client  $c$  can send a filter to the full node. Property 21 states the *AM* model guarantees privacy at least as good as the original *SPV* model even when all other clients collude with full nodes. We additionally note that the *AM* protocol is designed to be very simple and light to run, as light clients are not meant to run heavy computations. As this paper mainly focuses on the privacy issue of light clients, the aggregation model presented here is not complete, and future work will expand this model. However, it contains the main building blocks of a new model that has great potential in increasing light client privacy.

## 8 Conclusion

In this work, we analyzed and compared the privacy of the *SPV* and *Neutrino* light client implementations. We defined two metrics to evaluate the privacy of light clients: light client entropy and detection ratio. Based on these metrics we analyzed and evaluated the privacy of the different implementations on real data and evaluated the privacy-communication tradeoff, comparing the different implementations and discussing these results and ways to improve privacy. Finally, we suggested a new *SPV*-based light client model that improves privacy. In future work, we intend to deepen the privacy analysis, including cases when an adversary with a full node has only partial information. We aim to further enhance the technical details and the analysis of the aggregation model, dealing with issues such as disconnecting nodes

and other malicious behaviors.

---

## References


- 1 Mycelium - Bitcoin wallet. <https://mycelium.com/index.html>.
- 2 Percy++ / PIR in C++. <https://percy.sourceforge.net/readme.php>.
- 3 Neutrino: Privacy-preserving Bitcoin light client. <https://github.com/lightninglabs/neutrino>, 2017.
- 4 Helios. <https://github.com/a16z/helios>, 2023.
- 5 Run a light client to join binance chain. <https://docs.binance.org/light-client.html>, 2023.
- 6 Lodestar ethereum consensus implementation. <https://github.com/ChainSafe/lodestar>, 2024.
- 7 Ittai Abraham, Danny Dolev, and Joseph Y Halpern. Distributed protocols for leader election: A game-theoretic perspective. In *International Symposium on Distributed Computing (DISC)*, 2013.
- 8 André Augusto, Rafael Belchior, Miguel Correia, André Vasconcelos, Luyao Zhang, and Thomas Hardjono. Sok: Security and privacy of blockchain interoperability. *Authorea Preprints*, 2023.
- 9 Alex Biryukov and Sergei Tikhomirov. Transaction clustering using network traffic analysis for Bitcoin and derived blockchains. In *IEEE INFOCOM Workshops*, 2019.
- 10 Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, 1970.
- 11 Sean Braithwaite, Ethan Buchman, Ismail Khoffi, Igor Konnov, Zarko Milosevic, Romain Ruetschi, and Josef Widder. A tendermint light client. *arXiv preprint arXiv:2010.07031*, 2020.
- 12 Andrei Z. Broder and Michael Mitzenmacher. Network applications of Bloom filters: A survey. *Internet Mathematics*, 1(4):485–509, 2003.
- 13 Jehoshua Bruck, Jie Gao, and Anxiao Jiang. Weighted Bloom filter. In *IEEE International Symposium on Information Theory (ISIT)*, 2006.
- 14 Benedikt Bünz, Lucianna Kiffer, Loi Luu, and Mahdi Zamani. Flyclient: Super-light clients for cryptocurrencies. In *IEEE Symposium on Security and Privacy (SP)*, 2020.
- 15 Panagiotis Chatzigiannis, Foteini Baldimtsi, and Konstantinos Chalkias. SoK: Blockchain light clients. In *International Conference on Financial Cryptography and Data Security (FC)*, 2022.
- 16 Archana Chhabra, Rahul Saha, Gulshan Kumar, and Tai-Hoon Kim. Navigating the maze: Exploring blockchain privacy and its information retrieval. *IEEE Access*, 2024.
- 17 Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private information retrieval. *Journal of the ACM (JACM)*, 45(6):965–981, 1998.
- 18 Mauro Conti, E Sandeep Kumar, Chhagan Lal, and Sushmita Ruj. A survey on security and privacy issues of Bitcoin. *IEEE Communications Surveys & Tutorials*, 20(4):3416–3452, 2018.
- 19 Christian Decker and Roger Wattenhofer. Information propagation in the Bitcoin network. In *IEEE International Conference on Peer-to-Peer Computing (P2P)*, 2013.
- 20 Hans Dobbertin, Antoon Bosselaers, and Bart Preneel. RIPEMD-160: A strengthened version of RIPEMD. In *International Workshop on Fast Software Encryption*, 1996.
- 21 Dmitry Ermilov, Maxim Panov, and Yury Yanovich. Automatic Bitcoin address clustering. In *IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2017.
- 22 Bin Fan, Dave G Andersen, Michael Kaminsky, and Michael D Mitzenmacher. Cuckoo filter: Practically better than Bloom. In *ACM International on Conference on emerging Networking Experiments and Technologies (CoNext)*, 2014.
- 23 Arthur Gervais, Srdjan Capkun, Ghassan O. Karame, and Damian Gruber. On the privacy provisions of Bloom filters in lightweight Bitcoin clients. In *ACM Annual Computer Security Applications Conference (ACSAC)*, 2014.
- 24 Solomon Golomb. Run-length encodings (corresp.). *IEEE transactions on information theory*, 12(3):399–401, 1966.
- 25 Xi He, Ketai He, Shenwen Lin, Jinglin Yang, and Hongliang Mao. Bitcoin address clustering method based on multiple heuristic conditions. *IET Blockchain*, 2(2):44–56, 2022.

- 26 Mike Hearn. Bloom filter privacy and thoughts on a newer protocol. <https://groups.google.com/forum/#!msg/bitcoinj/Ys13qkTwcNg/9qxnHwnkeoIJ>, 2015.
- 27 Kilan M Hussein and MF Al-Gailani. Evaluation performance of bloom filter in blockchain network. *Iraqi Journal of Information and Communication Technology*, 6(2):17–30, 2023.
- 28 MJ Jeyasheela Rakkini and K Geetha. Detection of Bitcoin miners by clustering crypto address with google bigquery open dataset. In *Soft Computing: Theories and Applications (SoCTA)*. 2022.
- 29 Kota Kanemura, Kentaroh Toyoda, and Tomoaki Ohtsuki. Design of privacy-preserving mobile Bitcoin client based on  $\gamma$ -deniability enabled Bloom filter. In *Int. Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 2017.
- 30 Bruce M Kapron, David Kempe, Valerie King, Jared Saia, and Vishal Sanwalani. Fast asynchronous byzantine agreement and leader election with full information. *ACM Transactions on Algorithms (TALG)*, 6(4):1–28, 2010.
- 31 Kostis Karantias. SoK: A taxonomy of cryptocurrency wallets. *Cryptology ePrint Archive*, 2020.
- 32 Domokos Miklós Kelen and István András Seres. Towards measuring the fungibility and anonymity of cryptocurrencies. *arXiv preprint arXiv:2211.04259*, 2022.
- 33 Feng Liu, Zhihan Li, Kun Jia, Panwei Xiang, Aimin Zhou, Jiayin Qi, and Zhibin Li. Bitcoin address clustering based on change address improvement. *IEEE Transactions on Computational Social Systems*, 10:1–13, 2023.
- 34 Lailong Luo, Deke Guo, Richard T. B. Ma, Ori Rottenstreich, and Xueshan Luo. Optimizing Bloom filter: Challenges, solutions, and comparisons. *IEEE Communications Surveys and Tutorials*, 21(2):1912–1949, 2019.
- 35 Sinisa Matetic, Karl Wüst, Moritz Schneider, Kari Kostianen, Ghassan Karame, and Srdjan Capkun. Bite: Bitcoin lightweight client privacy using trusted execution. In *USENIX Security Symposium*, 2019.
- 36 R. C. Merkle. Secrecy, authentication, and public key systems. *PhD thesis, Stanford*, 1979.
- 37 Matt Corallo Mike Hearn. Connection Bloom filtering. <https://github.com/bitcoin/bips/blob/master/bip-0037.mediawiki>, 2012.
- 38 Deepanjan Mitra, Agostino Cortesi, and Nabendu Chaki. ALEA: An anonymous leader election algorithm for synchronous distributed systems. In *Progress in Image Processing, Pattern Recognition and Communication Systems (CORES, IP&C, ACS Conference)*, 2021.
- 39 Angelica Montanez. Investigation of cryptocurrency wallets on iOS and Android mobile devices for potential forensic artifacts. *Forensic Science Center, Marshall University*, 2014.
- 40 Malte Möser and Arvind Narayanan. Resurrecting address clustering in Bitcoin. In *International Conference on Financial Cryptography and Data Security (FC)*, 2022.
- 41 Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, 2008.
- 42 Yukun Niu, Chi Zhang, Lingbo Wei, Yankai Xie, Xia Zhang, and Yuguang Fang. An efficient query scheme for privacy-preserving lightweight Bitcoin client with Intel SGX. In *IEEE GLOBECOM*, 2019.
- 43 Alex Akselrod Olaoluwa Osuntokun. Compact block filters for light clients. <https://github.com/bitcoin/bips/blob/master/bip-0158.mediawiki>, 2017.
- 44 Jim Posen Olaoluwa Osuntokun, Alex Akselrod. Client side block filtering. <https://github.com/bitcoin/bips/blob/master/bip-0157.mediawiki>, 2017.
- 45 Kaihua Qin, Henryk Hadass, Arthur Gervais, and Joel Reardon. Applying private information retrieval to lightweight Bitcoin clients. In *Crypto Valley Conference on Blockchain Technology (CVCBT)*, 2019.
- 46 Ori Rottenstreich and Isaac Keslassy. The Bloom paradox: When not to use a Bloom filter. *IEEE/ACM Trans. Netw.*, 23(3):703–716, 2015.
- 47 Maroufi Saeid and Nemat Moein. Go bither. <https://github.com/bitherhq/go-bither>, 2017.


- 48 Sehrish Shafeeq, Sherali Zeadally, Masoom Alam, and Abid Khan. Curbing address reuse in the iota distributed ledger: A cuckoo-filter-based approach. *IEEE Transactions on Engineering Management*, 67(4):1244–1255, 2019.
- 49 Ke Shao, Wei Lv, and Yu Li. Addressing blockchain privacy and efficiency challenges in mobile environments: an optimization strategy for lightweight clients and full nodes. *Advances in Engineering Technology Research*, 7(1):1–1, 2023.
- 50 Apostolos Tzinas Angel Leon Dimitris Lamprinos Ardis Lu shrestha agrawal, Justin Martin. Kevlar. <https://github.com/lightclients/kevlar>, 2022.
- 51 Thomas Voegtlin. Electrum - lightweight Bitcoin client. <https://github.com/spesmilo/electrum>, 2018.
- 52 Kai Wang, Maike Tong, Changhao Wu, Jun Pang, Chen Chen, Xiapu Luo, and Weili Han. Exploring unconfirmed transactions for effective Bitcoin address clustering. *arXiv preprint arXiv:2303.01012*, 2023.
- 53 François-Xavier Wicht. Blockchain privacy notions using the transaction graph model. *Master Thesis, University of Fribourg*, 2023.
- 54 Karl Wüst, Sinisa Matetic, Moritz Schneider, Ian Miers, Kari Kostiainen, and Srdjan Čapkun. ZLite: Lightweight clients for shielded Zcash transactions using trusted execution. In *International Conference on Financial Cryptography and Data Security (FC)*, 2019.
- 55 Yankai Xie, Chi Zhang, Lingbo Wei, Yukun Niu, and Faxing Wang. Private transaction retrieval for lightweight Bitcoin client. In *IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2019.
- 56 Yankai Xie, Chi Zhang, Lingbo Wei, Yukun Niu, Faxing Wang, and Jianqing Liu. A privacy-preserving Ethereum lightweight client using PIR. In *IEEE/CIC International Conference on Communications in China (ICCC)*, 2019.
- 57 Yiyin Zhang. SoK: Anonymity of lightweight clients in cryptocurrency systems. In *IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2023.
- 58 Yuhang Zhang, Jun Wang, and Jie Luo. Heuristic-based address clustering in Bitcoin. *IEEE Access*, 8:210582–210591, 2020.
- 59 Baokun Zheng, Liehuang Zhu, Meng Shen, Xiaojiang Du, and Mohsen Guizani. Identifying the vulnerabilities of Bitcoin anonymous mechanism based on address clustering. *Science China Information Sciences*, 63:1–15, 2020.



# CrudiTEE: A Stick-And-Carrot Approach to Building Trustworthy Cryptocurrency Wallets with TEEs

Lulu Zhou ✉ 


Yale University, New Haven, CT, USA

Zeyu Liu ✉ 

Yale University, New Haven, CT, USA

Fan Zhang ✉ 

Yale University, New Haven, CT, USA

Michael K. Reiter ✉ 

Duke University, New Haven, CT, USA

---

## Abstract

Cryptocurrency introduces usability challenges by requiring users to manage signing keys. Popular signing key management services (e.g., custodial wallets), however, either introduce a trusted party or burden users with managing signing key shares, posing the same usability challenges. TEE (Trusted Execution Environment) is a promising technology to avoid both, but practical implementations of TEEs suffer from various side-channel attacks that have proven hard to eliminate.

This paper explores a new approach to side-channel mitigation through *economic incentives* for TEE-based cryptocurrency wallet solutions. By taking the cost and profit of side-channel attacks into consideration, we designed a Stick-and-Carrot-based cryptocurrency wallet, CrudiTEE<sup>1</sup>, that leverages penalties (the stick) and rewards (the carrot) to disincentivize attackers from exfiltrating signing keys in the first place. We model the attacker’s behavior using a Markov Decision Process (MDP) to evaluate the effectiveness of the bounty and enable the service provider to adjust the parameters of the bounty’s reward function accordingly.

**2012 ACM Subject Classification** Security and privacy → Authorization; Security and privacy → Side-channel analysis and countermeasures

**Keywords and phrases** Cryptocurrency wallet, blockchain

**Digital Object Identifier** 10.4230/LIPIcs.AFT.2024.16

**Related Version** *Full Version*: <https://arxiv.org/abs/2407.16473>

**Supplementary Material** *Software*: [https://github.com/luluzhou1/MDP\\_for\\_Bounty\\_Evaluation](https://github.com/luluzhou1/MDP_for_Bounty_Evaluation) archived at [swh:1:dir:ff2054e363714e71e597d4d669d602df3f60f5dc](https://swh.1:dir:ff2054e363714e71e597d4d669d602df3f60f5dc)

**Funding** This work was funded in part by NSF grant 2207214 and an Ethereum Academic Grant.

## 1 Introduction

As cryptocurrencies [8, 59] gain popularity, the daunting task of key management—the process of keeping cryptographic keys secure from attacks and loss—has become an everyday task for *end users*. With inexperienced users often struggling with lost or leaked keys, a natural tendency is to outsource the task to specialized service providers. For example, 11% of the entire cryptocurrency marketization is stored in custody by a single service provider (Coinbase [61]). This is undesirable security-wise, as the secrecy of keys (thus the safety of the funds) relies on the trustworthiness of a centralized party.

---

<sup>1</sup> Crudite is a salad with carrots and (other) vegetable sticks.



To provide stronger security guarantees (and to reduce liability), a cryptocurrency wallet service provider can generate users' signing keys in Trusted Execution Environments (TEEs, such as Intel SGX [3, 37], AMD SEV [2], Nvidia H100 [28]) and serve signing requests in TEE without ever seeing the signing keys in plaintext. However, the naive adoption of TEEs does not provide a meaningful secrecy guarantee *to users*, because the service provider may be able to exfiltrate signing keys through side-channel attacks [43]. While side-channel mitigation has been extensively studied in the literature (e.g., see [52] for a survey), side channels are notoriously hard to eliminate, due to the complexity of modern processor design (e.g., TEEs often share physical resources with untrusted processes, such as caches).

Our work is motivated by the observation that the operator of TEEs is the primary actor capable of mounting side-channel attacks, since most attacks [57, 43, 34, 49, 35, 42] require root access to the host. For wallet key management services, the TEE operator is the service provider. This observation gives us additional leverage to prevent side-channel attacks, as the service provider can be held responsible (using techniques to be presented later) if a wallet key is leaked or accessed without user authorization. Using a proper penalty mechanism, we can eliminate the service provider's gains from a successful side-channel attack, thus removing the incentive to attack in the first place.

With the TEE operator striving to avoid key leakage, the possibility of side-channel attacks by non-local, unprivileged attackers is significantly reduced (e.g., the service provider is motivated to employ heightened security measures). To further discourage such attacks, our idea is to reward the attackers for partial success. For example, if a signing key is distributed cross  $N$  TEEs using secret sharing, we give the attacker a substantial reward if he successfully exfiltrated *any* share. With a proper reward function, this early reward can serve as a strong incentive for the attacker to *stop early*, giving the system administrator time to react to partial compromise before a full key is exfiltrated.

## 1.1 CrudiTEE: The Cryptocurrency Wallet with Stick and Carrot

Based on the above two principles, we propose CrudiTEE, a TEE-based cryptocurrency wallet that can defend against TEE side channels by privileged and unprivileged attackers, using penalties (stick) and rewards (carrot), respectively. Furthermore, CrudiTEE strives to achieve user-friendliness (i.e., users do not need to store keys locally). CrudiTEE first requires that the signing keys be generated inside TEE and never exported in plaintext. Assuming correct implementation, this implies that signing key leakage is impossible except for through side-channel attacks.

We classify potential actors capable of mounting side channel attacks into *insider attackers* and *outsider attackers*. The insiders are privileged attackers, such as service providers, who have full control over the TEE including physical access. Insiders have powerful attacking capabilities required by most side-channel attacks (such as root privilege) like the ones needed in [31, 20, 47]. In contrast, the outsiders are all the attackers who can exfiltrate the secrets in the TEEs only through less-privileged means like remote time-based attacks [32, 12, 1]. We refer readers to Section 2.2 for more examples. As introduced above, CrudiTEE consists of the stick (penalties), to discourage insider attackers, and the carrot (rewards), to encourage outsider attackers to stop early.

Note that to perform such punishment or distribute the bounty, we need an automated but also trustworthy and publicly accessible mechanism. Smart contracts [59] (autonomous programs executed on blockchains) are the perfect tool for this purpose. Thus, below, when discussing the stick and the carrot, we use the smart contract as an important building block.



### 1.1.1 The stick

Due to the power of the service provider, preventing it from mounting side channels via a technical way seems infeasible. Instead, CrudiTEE requires the service provider to put down *collateral*, which will be confiscated if signing keys safeguarded by the TEEs are used for unauthorized signatures or if legit service requests from users are denied.

To realize the stick of CrudiTEE, the key is to enable a user to generate publicly verifiable proof if her TEE-generated keys are illegally accessed. First, as mentioned, raw keys stay in the TEE and are never exported outside. Second, each key corresponds to a wallet owner and can only be used by the owner through well-defined APIs (e.g., an API could allow the owner to sign messages with the key using a carefully implemented signature algorithm). Third, to access a key, a signed authorization from its owner must be present and checked by TEEs, thus making the authorization process *accountable* (i.e., if the user disputes a signature, the service provider can present proof that the signature was authorized by the user). Users can verify TEE attestations to ensure the prerequisites are met before signing up for the service.

In order not to burden the user with signing key management while making the authorization process accountable, we use the OAuth protocol (Section 3.3). The token signed by the OAuth provider is used as proof of authorization.

The service provider sets up a smart contract to implement the insurance (denoted  $SC_{ins}$ ) with the following logic and makes an initial deposit. If a user discovers any unauthorized signature, she can submit a request to  $SC_{ins}$ . The service provider must prove that the user had authorized such key use within a specific period. Failing to provide such proof results in the insurance smart contract automatically compensating the user.

### 1.1.2 The carrot

Without the help of any insiders, outside attacks become unlikely, but still not impossible. To limit potential exposure to external attacks, we employ the threshold signing protocol such as [29], where the signing key is stored as key shares across multiple independent TEEs (e.g., hosted in different clouds) and refresh secret shares periodically. This way, even if an outside attacker can exfiltrate a few shares, he needs all shares to exfiltrate the entire key. However, the security of such proactive secret sharing method as a defense is “black or white” – unless the attacker can break a sufficient number of TEEs and cause a catastrophic breach, partial breaches cannot be detected and therefore cannot inform the service provider to take proper action to prevent those catastrophic breaches. By exploiting economic incentives, we can elicit such information from the attacker. Specifically, CrudiTEE enhances a proactive secret-sharing scheme with an alerting mechanism so that when partial breaches happen (e.g., TEEs deployed in one cloud are vulnerable, but not others), the attacker is encouraged to alert the service provider in exchange for a *bounty*. This allows the service provider to take proper action before full breaches happen.

Designing a bounty reward function that induces the desired behavior of the attackers is the main technical challenge. Specifically, we aim to formulate a reward function that motivates attackers to promptly alert the service provider without generating any illegal signature or selling the acquired signing key shares, while minimizing the defender’s cost (i.e., the service provider’s cost). We employ a 2-step methodology in the reward function design: we start with the attacker with a fixed known cost first and then deal with the one whose attacking process is non-deterministic and whose cost cannot be accurately estimated.

**Step 1.** We provide the following toy example to illustrate the challenge in reward function design under a deterministic setting. We start with a key (worth \$3 in total) stored as three secret shares, each of which is worth \$1 (assuming a share can be sold on the market for \$1). To steal one share, the attacker’s cost is \$0.8. Furthermore, assume that \$0.01 is the smallest unit of money for simplicity. Without a bounty, the attacker will keep attacking until he gets 3 shares and sells them on the market for \$3, making a profit of \$0.6.

To protect against such an attacker, there are two naive but natural solutions. The first solution is to simply have the reward function be a constant function of \$3.01 (i.e., the attacker obtains \$3.01 for any amount of shares he steals). In this case, an attacker always submits the share as soon as he obtains the first share, but then the defender costs more than the key value itself. The second solution is setting the function to be \$1.01 per share (i.e., a function linear in the number of shares). However, in this case, an attacker would instead try to obtain all three shares and claim a total reward of \$3.03, which costs even more.

The optimal solution is to set the reward function to be a constant function of \$1.41 (i.e., the attacker is awarded with \$1.41 for submitting any amount of shares): the attacker will stop attacking and turn in the key shares whenever he obtains 1 key share, making a profit of \$0.61. This reward function not only encourages the attacker to submit as soon as getting one share but also minimizes the defender’s cost. Note that it is indeed the least the service provider can pay, as if the reward is less than \$1.41, the attacker will sell the key for a higher profit instead (assuming w.l.o.g. that the attacker sells the secret when the profit from the bounty is tied with selling the key).

**Step 2.** The reward function in the toy example, however, is based on a simplified assumption of deterministic attack costs and requires the defender to accurately know the attacker’s cost. Our design instead aims to address real-world situations where the attacker’s attack process is non-deterministic, and the cost of attacking cannot be accurately known in advance.

To design the reward function in this setting, we first turn the desired properties of the reward function into numerical metrics. Then we capture the non-deterministic attacking process as an “optimal stopping” game and use Markov Decision Process (MDP) to analyze the attacker’s optimal strategy. We propose a reward function for non-deterministic attackers and optimize it using the metrics as an objective function, based on the defender’s budget and estimation of the attacker’s cost and success rate. We further show that the reward function not only has good performance for the attacker with an accurately estimated cost but also for attackers with different costs. We provide the defender with the performance of the optimized reward function for attackers with a wide range of costs and success rates. The defender can use such a strategy to assess how the reward function she obtains performs for a range of attackers. If she is not satisfied with the result, she could raise their budget and generate another function.

To realize the bounty, the service provider creates a smart contract  $SC_{\text{bounty}}$  that accepts proofs of knowledge (PoK) of TEE-managed key shares and remits rewards accordingly. Valid PoK submissions to  $SC_{\text{bounty}}$  raise a flag, pausing operations until the keys are rotated and the flag is reset. To ensure that the attacker did not use the breached key for unauthorized signings, users are requested to check for unauthorized signatures during the shutdown period. If any are found, the attacker’s reward is forfeited.

## Contribution

We summarize our contributions as follows:

1. We introduce a new approach to building a cryptocurrency wallet: *CrudiTEE* that leverages *economic incentives* to defend against side-channel attacks from insiders and outsiders.
2. *CrudiTEE* involves a novel automatic insurance system (Section 5), allowing users to receive compensation if their wallet signing key is used for signing transactions without their authorization.
3. We develop a reward function for the bounty in *CrudiTEE* (Section 6) that encourages attackers to submit key shares to the bounty immediately while minimizing the defender's cost. We use the Markov Decision Process (MDP) to model the non-deterministic nature of side-channel attacks and optimize the reward function against numerical metrics. We evaluate and show the optimized reward function is effective not only for attackers with precisely estimated costs but also for attackers with variable costs. The service provider may adjust her budget to cover a wider range of attackers the reward function can effectively defend against based on the evaluation.

## 2 Related Work

### 2.1 Cyber Bounty

Setting up bug bounties is a popular way to defend against hackers [36]. However, a fair exchange of bugs and money is difficult without trust. Breidenbach et al. [10] proposed that smart contracts be deployed to guarantee that the attacker gets paid once a valid bug is submitted. Their game-theoretic analysis showed that the attacker is incentivized to submit the bug as soon as possible because of competition from other honest hackers. However, this is not always the case for side-channel attacks: a malicious attacker may be the only one to discover a zero-day<sup>2</sup> side channel. That is why we take the submission time into consideration in our reward function, i.e., to incentivize attackers to submit the leaked signing key (share) immediately upon acquiring it.

### 2.2 Side Channels

Side-channel attacks against cryptographic systems usually take one of three forms. *Time-driven* side-channel attacks expose key information by monitoring *total* execution times of cryptographic operations with a fixed key, which can reflect interactions among the value of the key, the structure of the cryptographic implementation, and system-level effects such as cache evictions (e.g., [32, 12, 1, 58]). *Trace-driven* side-channel attacks observe a time-series signal reflecting a device's cryptographic operation *throughout its execution*, e.g., by monitoring the device's power draw during the operation (e.g., [31]) or its electromagnetic emanations (e.g., [20, 47]). Finally, in an *access-driven* side-channel attack, the attacker executes a program on the same computer where the cryptographic operation is taking place, using this vantage point to monitor the operation's use of microarchitectural components on the platform (e.g., [45, 27, 26]). Time-driven and trace-driven attacks are largely agnostic to the encapsulation of the cryptographic operation within a TEE. In contrast, much effort has been expended to adapt access-driven attacks to attack a cryptographic operation executed within TEE from outside, with considerable success (e.g., [57, 43, 34]).

Using the terminology of Section 1, we consider *outsiders* to be less privileged and thus limited to time-driven and some access-driven attacks, that can be performed remotely (i.e., without any physical access to the TEE). Any attacks available to an outsider, however, must

---

<sup>2</sup> A zero-day is a vulnerability in software or hardware that is unknown to its vendor.

incur costs to conduct over time, e.g., to achieve and maintain co-residency on the same physical computer as the victim computation [56] (possibly despite defenses to make this difficult, e.g., [41]) and to perform attack computations. In contrast, *insiders* are permitted to conduct *any* time-driven, trace-driven, or access-driven attacks, and so are considerably more powerful. In particular, we design CrudiTEE in anticipation of insiders capable of extracting keys from TEEs easily. Outsiders, on the other hand, are assumed to require more time and costs to mount their attacks.

## 2.3 TEE Side-channel Defense

A recent concurrent and independent work, Sting [7], proposes to use SC as a bug bounty, which is set up to encourage anyone who has access to a leaked secret to submit proof. The proof of leakage is acquired in this way: first, a prover-owned TEE generates a secret, without disclosing it to the prover. Second, the secret is directly sent to the secret management service provider (without exposing the secret to the prover). Finally, the prover acquires the secret using a side-channel attack, sends it back to the prover-owned TEE, and gets proof of leakage from the TEE. Sting focuses more on the proof generation rather than the bounty design, however. This is different from our bounty as we encourage attackers (*without* physical access to the machine) to stop recovering the secret and submit a bounty claim without recovering the whole secret via economic incentives.

Numerous techniques other than bug bounty could be applied to side-channel defense, including ORAM [16], code hardening [11], data location randomization [9]. However, defenses introduce performance overheads and usually defend against only specific types of attacks. Another problem is that a service provider might not have enough incentive to apply these defensive technologies expeditiously. Therefore, motivating the service providers to keep their TEEs safe from attack is crucial to the real-world use of TEEs.

## 2.4 Existing Wallet Solutions

Some companies provide the service like a centralized bank for cryptocurrency [15], holding users' funds in company-owned accounts. Such centralized service deviates from the decentralized nature of cryptocurrency and increases risk to user funds. On the other hand, there are products to enable users to store their signing keys in a protected area of an offline device, named hardware wallet [51]. This approach raises costs and complicates transactions, and users usually have to trust the software provided by the hardware manufacturer for signing transactions. A keyless wallet was constructed using witness encryption [63]. To access the money, the user only needs to provide a short one-time password of 6 alphanumeric characters generated from an offline device. Since Witness Encryption is currently impractical, however, the scheme is largely theoretical.

# 3 Background and Preliminaries

## 3.1 Trusted Execution Environments

TEEs (Trusted Execution Environments) are secure and isolated execution environments that provide confidentiality and integrity guarantees and the ability for a party to remotely verify the status of a TEE through remote attestation. Prominent examples of TEEs include Intel SGX [3, 37], AMD SEV [2], and Nvidia H100 [28]. A major practical limitation of TEEs is side channel attacks (Section 2.2) that could break the confidentiality guarantee.

## 3.2 Smart Contracts

To create elaborate economic incentive structures, CrudiTEE uses smart contracts, autonomous programs running on top of blockchains, to remit payments under specific events. We follow the standard assumption that smart contracts are correct (i.e., the security assumptions required by the blockchain protocol are met) and available (i.e., all parties in our protocols can access the smart contract and request submitted to the smart contract is executed within a time limit).

## 3.3 OAuth

CrudiTEE uses the OpenID Connect feature in OAuth (Open Authorization) 2.0 [25, 46] to enable users to make signing requests without possessing a signing key. OpenID is an authentication protocol that allows users to use an existing account from an OpenID provider (denoted as “OAuth provider”), such as Google, to authenticate themselves on other applications. Furthermore, during authentication, a user can embed a customized message in the “nonce” field of the signed ID token [25] (looking ahead, this allows the user to put a description of her request in this field).

## 3.4 Cryptographic Primitives

We provide a brief description of the threshold signing scheme.

Threshold signature allows  $N > 1$  parties to share a secret signing key, such that each party obtains a share of the signing key. Only when  $m$  parties owning a sharing,  $1 \leq m < N$ , together can sign a message. Knowledge of  $< m$  shares leaks no information about the secret signing key. Furthermore, when the secret shares are updated to  $N$  new shares, even  $m_1 < m$  old shares and  $m_2 < m$  new shares where  $m_1 + m_2 \geq m$  together leak no information about the secret. We use it to allow multiple TEEs to share the signing key, such that only if  $\geq m$  shares are leaked, the secret is leaked.

## 3.5 Markov Decision Process

A Markov decision process (MDP) is a mathematical model that captures decision-making under uncertain situations. A Markov state is a state  $S_t$  at time  $t > 0$  satisfying  $\Pr[S_t|S_{t-1}] = \Pr[S_t|S_{t-1}, \dots, S_1]$  (i.e., the previous state captures the entire history states). The MDP consists of a sequence of Markov states and an associated state transition matrix. This matrix represents the probabilities of transitioning from one state to another based on the player’s actions. The player’s optimal strategy in MDP can be computed using tools like [13].

# 4 Threat Model and Roadmap

## 4.1 Threat Model

The purpose of the techniques in CrudiTEE is to mitigate the side-channel attacks that break the privacy of the TEEs but not the integrity. We assume TEE integrity (i.e. the data and code in the TEE cannot be modified by any attacker) to hold and remote attestation to be secure, following a common assumption (c.f., [53, 14]), as the attestation key is only used through a limited interface, unlike application-generated secrets. The side-channel attacks that are strong enough to compromise the attestation key [55] are out of scope for this work, as such incidents have historically been rare.

We assume that the integrity and liveness of smart contracts are enforced by the blockchain. Furthermore, we assume the OAuth providers are trusted, but note that any user can choose her own set of OAuth providers to trust (i.e., the user can choose a subset of a predefined set of OAuth providers). Finally, we assume that both the service provider and the outsider attacker are rational entities aiming to maximize their profits. We do not consider non-financial incentives, and the agent who attacks the system as a mere malicious intruder is out of our scope.

## 4.2 Wallet Design Overview

In our wallet service, each client registered with the wallet service provider has a wallet whose signing key is stored in the service provider's TEE. Our goal is to defend side-channels against such signing keys.

We categorize side-channel attacks into two types: insider attacks, which require physical access and/or root privileges, and outsider attacks which can be executed remotely without such privileges (Section 2.2). In our wallet design, the service provider, who controls the TEEs, is classified as an insider, whereas all other attackers, including users, are categorized as outsiders. We defend the insiders using the insurance (the stick) and the outsiders using the bounty (the carrot).

The side-channel mitigation in CrudiTEE thus consists of three main components:

1. The accountable signing key management service (Section 5.1) enables the users to register for the service and authorize the service provider to sign a transaction when needed.
2. The insurance (Section 5.2) ensures the service provider provides the desired service, and otherwise is punished.
3. The bounty (Section 6) aims to incentivize the outsider attacker to submit the key shares acquired through the remote side channel to the bounty (smart contract) rather than using them to make unauthorized signatures or selling them.

Both the insurance and the bounty are initiated using smart contracts ( $SC_{ins}$  and  $SC_{bounty}$ ). In addition, to make sure that the service provider answers all the service requests (instead of ignoring those requests), the smart contract  $SC_{avail}$  is also deployed. During setup, the service provider needs to build the TEE program and publish the attestation. Then, the service provider deploys the aforementioned smart contracts on the blockchain.

To use the service, the user first chooses the OAuth provider(s) she trusts and creates a new account with her OAuth token (signed by that OAuth provider(s)). The service provider will execute the threshold key-generation protocol among the TEEs, register the OAuth account and key mapping, and then provide the public key to the user. It is essential that the signing key is generated within the TEEs and remains within the TEEs (i.e. cannot be exported in plaintext format). This is because if the users learn the key, it becomes ambiguous whether the responsibility for any unauthorized signature lies with the users or the service provider. After the generation of the signing key, a smart contract wallet  $SC_{wallet}$  will be deployed for the user.  $SC_{bounty}$  will also be updated so that the new key is also protected by the bounty. The proof-of-publication<sup>3</sup> scheme is employed to ensure that the smart contract update is done properly.

The service provider replies to the user's transaction signing requests with authentication via OAuth providers (Figure 1). The signing is conducted using the threshold signature scheme, with the signing key secret-shared among several TEEs. When the service provider

---

<sup>3</sup> Proof of publication is a way for the TEE to verify that a state change is updated on the blockchain.

is not responding to a signing request, the user can send the request through  $SC_{avail}$  and force the service provider to respond. If the user realizes that an unauthorized signature exists, she can submit a claim to  $SC_{ins}$  and get compensated (Figure 2).

Finally, if an outsider attacker steals the signing key (shares) from a remote side channel, he can submit it to  $SC_{bounty}$  and get rewarded based on the submission time and number of shares he submits (Figure 2). Any valid  $SC_{bounty}$  or  $SC_{ins}$  submission will trigger a flag to signify that some of the TEEs have been breached. CrudiTEE requires that all wallet transactions cease until the service provider rotates all the signing keys and clears the flag. If the full key is leaked, the TEE will generate a fresh key pair, update the OAuth account and wallet key mapping, and transfer the money in the smart contract wallet to the new wallet while the red flag is on. Transactions during the red flag period can only be triggered by a message signed by the TEE attestation key. The reward for the attacker will be held for a specified period, during which the user of the affected keys will be asked to check whether there exists any unauthorized transactions and the reward will not be given to the attacker if such transactions are found.

### 4.3 Reward Function Design Roadmap

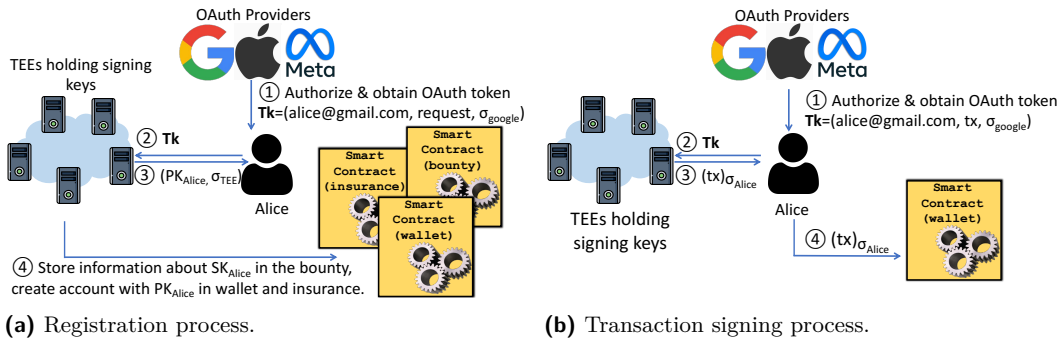
The attacker’s reward is determined by a reward function designed to incentivize them to claim the bounty immediately upon obtaining a single key share from the TEEs, while minimizing the defender’s cost (Section 6.3). Since the reward function design is particularly challenging among other components of the wallet, we discuss our roadmap here. We employ a 2-step methodology here: First, we deal with attackers with known deterministic costs (a simplified case). Then, we employ the ideas from this simplified case together with other more advanced mechanisms to develop the reward function for the attacker with non-deterministic and unknown costs.

In more detail, we begin with a case study assuming the attacker operates under a deterministic cost function known by the defender. However, in the real world, the side-channel attacking process is non-deterministic, and the cost of the attack is hard to estimate accurately. Building on insights gained from the case study, we propose a reward function for attackers with non-deterministic behavior. We model the non-deterministic attacking process as the “optimal stopping” game [54, 50, 24] and employ Markov Decision Processes to calculate the best strategies for the attackers. By translating the desired properties of this reward function into quantitative metrics used as the objective function, we optimize the parameters in the reward function (based on the defender’s budget and her estimation of the capability of the attacker). Finally, we evaluate the effectiveness of our proposed reward function when the attacker’s ability (parameterized by his cost and success rate) is different from the estimations. Based on the evaluation of the attacker, the defender can further raise her budget and recompute the function to get a more satisfying range of attackers the function can defend against.

## 5 The Stick

In this section, we first provide more details about the wallet workflow (Section 5.1), which outlines the responsibilities of the service provider. Then, we specify the “stick” part which holds the service provider responsible (Section 5.2).

## 16:10 CrudiTEE: A Stick-And-Carrot Cryptocurrency Wallet



■ **Figure 1** Registration and Transaction Signing Workflow.

### 5.1 Authorization and Signing Transactions

We start by elaborating on how we make the authorization of the transactions accountable and describe how a user registers for an account and requests signed transactions.

**Accountable authorization.** As mentioned in the Section 1.1.1, an authorization process is *accountable* if it leaves a signed evidence that can be used to prove the validity of the signing key usage later. Meanwhile, it should not burden the user with additional key management.

Our solution leverages a feature in OAuth 2.0 called OpenID Connect (OIDC) [46, 25]. Specifically, OIDC-enabled OAuth providers issue signed identity tokens (called `ID_token` [25]) that include a user identifier (such as email addresses) and a nonce set by users. Many mainstream OAuth providers enable the user application to specify the nonce in the ID token (e.g., Google [25], Microsoft[39], etc.).

Every time the signing key is used, we require the user to provide an ID token signed by the OAuth provider(s), which is uniquely linked to that specific signing request by including the request hash in the nonce field. TEE verifies the token of the corresponding OAuth provider(s) keys accordingly. The public key of the OAuth providers is hardcoded in TEE and verified by the user through attestation. This method not only provides a log-in process that most users are familiar with, but also delegates authorization to a third party (or a set of third parties) that they trust, providing signed OAuth token(s) as proof of authorization.

**Registration.** As shown in Figure 1 (a), when registering for a new account, the user runs a protocol to determine the future authentication process with the service provider. Specifically, the user first chooses a set of OAuth provider(s) she trusts. Next, she puts the hash of the account registration request (e.g. the hash of “CrudiTEE account registration”) in the “nonce” field of the ID token, authenticates it with the OAuth provider, and asks the OAuth provider to sign it. Then, the user sends the account registration request to the service provider along with the token(s). TEE verifies the token(s) and generates a fresh key pair for signing. The TEE creates a TEE-signed receipt with the newly generated verification key (to verify the signed transactions for this user’s wallet) and the OAuth ID(s) associated with it. Lastly, a smart contract wallet is created for the user.

**Transaction signing request.** As shown in Figure 1 (b), when the user wants to sign a transaction, she generates a signing request. Then, she acquires a signed token from the OAuth provider(s) with the hash of the transaction included in the token(s). Once receiving the signing request and token(s), the service provider should input it into the TEEs. The



TEE will check the validity of the request by verifying the token(s) and respond accordingly (we discuss how to enforce the TEEs to respond in Section 5.2.1). If the request is valid, the TEE will reply with the signature of the transaction, generated with the signing key associated with the user’s OAuth ID(s). If not, the TEE will reply with a message saying that the request is invalid, signed with its attestation key. We require TEEs to store the (valid) tokens and requests in case of any future insurance claim (Section 5.2.2). The signed transaction will be submitted by the user to the wallet smart contract  $SC_{\text{wallet}}$ . The wallet smart contract will check the signature and execute the transaction.

**Threshold signing.** CrudiTEE use a threshold signature scheme (e.g., [22]) for signing. Specifically, the key-management service provider secret-shares each key into  $N$  secret shares using a  $(m, N)$ -threshold-signature scheme (where  $m \leq N$ ), stores them in independent TEEs, and rotates them every  $T$  units of time. This approach not only serves to complicate the execution of side-channel attacks but also establishes the foundation for the bounty scheme described in Section 6.

## 5.2 The stick: hold service provider responsible

Based on the accountable signing process described in the previous subsection, the “stick” aims to establish mechanisms to punish the service provider when it misbehaves. The goal is that *any* rational service provider would not choose to misbehave (e.g., steal the secret and produce an unauthorized signature).

### 5.2.1 Ensure Availability of TEE

We start by discussing how to ensure that service providers process requests using TEE (with the expected inputs), guaranteeing TEE’s availability <sup>4</sup>. The service provider sets up  $SC_{\text{avail}}$  and makes the initial deposit. If the service provider refuses to process a signing request directly submitted to the service provider, the user submits the request to  $SC_{\text{avail}}$ . The service provider monitors the SC, processes any request from the SC, and forwards the request to the TEE. The TEE then generates a reply, which is either the requested signature or indicates that the request is invalid. The reply, along with the user’s request, must be signed by the TEE’s attestation key. After receiving the reply,  $SC_{\text{avail}}$  checks whether the reply is signed by the TEE’s attestation key and the request is included in the signed message.<sup>5</sup> If it is,  $SC_{\text{avail}}$  records the reply. If the service provider does not submit a valid reply within a time limit, its deposit gets burnt (destroyed).<sup>6</sup>

### 5.2.2 Insurance for unauthorized transactions

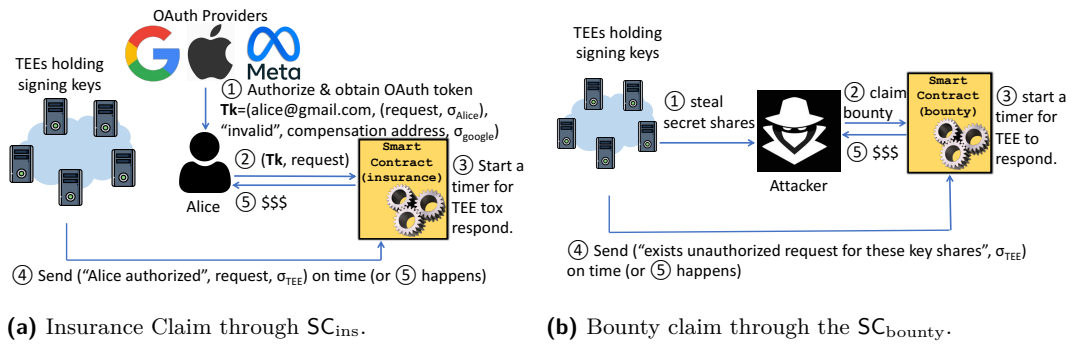
In this part, we develop a mechanism that enables users to report unauthorized transactions. As shown in Figure 2 (a), the user submits the signature to request a message, signed by the TEE’s attestation key, stating that the signature is authorized by the user. When the

<sup>4</sup> The idea of using incentives to make a service available is not new, though. A similar method is used in blockchain Layer2 to prevent transaction censorship [5].

<sup>5</sup> Attestation key is hardcoded to the smart contract.

<sup>6</sup> Note that one may consider a DoS-attack: initiating many small transactions using  $SC_{\text{avail}}$ . To avoid this, the service provider can set a corresponding transaction fee to use  $SC_{\text{avail}}$  paid by the user. If the user, however, needs to use such a service, the user may consider the service provider as malicious, thus withdrawing all the money and stop using the service. Thus, a rational service provider would avoid letting the user make transactions via  $SC_{\text{avail}}$ .

## 16:12 CrudiTEE: A Stick-And-Carrot Cryptocurrency Wallet



■ **Figure 2** Insurance and bounty workflow.

service provider is unable to provide such a message, the user is automatically compensated. Since the user initiates the insurance claim, they are responsible for monitoring transactions and submitting complaints for unauthorized transactions, similar to most systems based on staking and slashing [33].

We instantiate the insurance using a smart contract ( $SC_{ins}$ ). This smart contract specifies the necessary ground truth requirements, such as the attestation key of the TEEs, and the conditions under which users are eligible for compensation. A predefined quantity of deposits is deposited in it, serving as potential compensation for the user.

An insurance claim is initiated by the submission of an unauthorized transaction to  $SC_{ins}$  together with the proof of ownership of the key. The proof of ownership is a message stating the ownership of the key signed by the TEE, which could be requested using the user’s OAuth token.  $SC_{ins}$  checks whether the claim for the transaction has not yet been made before. If yes, the claim will be rejected. The service provider monitors  $SC_{ins}$  and sends the request to the TEE once it is published on the blockchain. The TEE looks for the authentication token(s) associated with this request (recall that the valid requests are stored). If no valid token(s) in question are found, the TEE will sign a message stating that the signature was unauthorized with its attestation key. Otherwise, a message stating that the signature was authorized will be signed. The service provider submits the reply to  $SC_{ins}$ .  $SC_{ins}$  checks whether the message signed by the TEE attestation key states that the signature was authorized. If not,  $SC_{ins}$  compensates the user (for some predetermined value that depends on the application) and records this claim (e.g., on the chain) for future reference. If the service provider fails to submit the requisite proof within the specified timeframe, the user automatically gets compensated from the smart contract.

**Security analysis.** We briefly analyze how the initial goal was achieved with the design of the “stick”. For any attack, the service provider can earn at most the total value of all the accounts. Therefore, as long as the collateral required to be put down is larger than this total amount<sup>7</sup>, a service provider has no incentive to misbehave, as each misbehavior costs more than what it gains.

<sup>7</sup> We believe that a 100% deposit is reasonable because the cost to the service provider is the potential interest they could have earned on the deposit, not the deposit itself.

## 6 The Carrot

In this section, we describe how we design the bounty (the carrot in CrudiTEE) to defend against the outsider attacker. The goal is to encourage the outsider attacker to report the wallet signing key breach to the service provider without abusing the signing key.

Throughout this section, we refer to the service provider as the defender, using these two terms interchangeably.

### 6.1 Desired properties of the Bounty

Distributing signing key shares across multiple TEEs with a threshold signature key generation procedure can lower the chance of signing key breaches caused by outsiders as used in [29]. However, it is not fully resolved. In this section, we further mitigate the risk of unauthorized signatures resulting from side-channel attacks by external attackers with a bounty. The bounty enables the service provider to take appropriate actions before any catastrophic security breaches occur.

The two technical difficulties in the design of the bounty are: (1) how can the attacker and the service provider perform an atomic exchange of the key share and the reward; and (2) how to give the attacker just enough incentive to claim the bounty, while saving the defender's cost. In detail, a good bounty should achieve the following goals:

1. An attacker gets the reward from the service provider if and only if he submits valid proof that convinces the service provider that he has obtained the key share.
2. The construction itself does not leak any knowledge about the key share other than what has already been obtained by the attacker.
3. An attacker prefers submitting the key share(s) to bounty over selling them in the market.
4. An attacker submits the key share as soon as he gets the first key share, instead of continuing the attack.
5. The defender's cost is minimized.

We suggest using smart contract bounty (Section 6.2) to satisfy the goal 1-2. Goals 3-5 are achieved by carefully designing a reward function for submitting key shares for a bounty claim.

### 6.2 The Smart Contract Bounty

To realize the atomic exchange of the key share and the reward, we initiate the bounty using a smart contract  $SC_{\text{bounty}}$ .

As a defense against the outsider attacker, the signing keys are rotated every  $T$  units of time. Following each key shares rotation, each TEE computes the hash of all the shares they hold and outputs the hash values to the service provider. The service provider then publishes them in the  $SC_{\text{bounty}}$ . The problem arises when the service provider publishes the hash values that do not match the ones generated by the TEEs, making the bounty unable to be claimed. To ensure that the hashes of the key shares are successfully published on the blockchain, we use the proof of publication scheme [14]. In other words, after each rotation or restart, the TEE will verify that the hash of the key shares they are using is the same as the latest version published on the blockchain (via proof of publication). Only then will it use the current key shares to sign the user's requests.

To claim the bounty, the attacker submits the share(s) he finds as proof of knowledge. To prevent front-running, proofs are submitted following a commit-and-reveal scheme [62]. We model this hash function as a random oracle so that it does not leak any information about the key shares themselves.

Upon receiving the key share, the smart contract  $SC_{\text{bounty}}$  checks whether the hash of the share is included in the smart contract. If it is,  $SC_{\text{bounty}}$  puts the reward on hold for a designated period and immediately invalidates all the current secret shares (such that the attacker cannot sell the shares or produce unauthorized signatures after submitting to the bounty). At the same time, the service provider asks the user of the affected accounts to submit insurance in case there exists an unauthorized signature. The attacker gets the reward if there is no insurance claim for the signing key whose shares they are submitting. The amount of the reward is determined by the reward function specified in Section 6.3.

### 6.3 Reward Function Design

In this subsection, we apply a two-step methodology to the design of the reward function. First, we present a case study focused on the reward function for a deterministic attacker (Section 6.3.2). Then, we broaden the scope to more general scenarios involving non-deterministic attacks (Section 6.3.4 to Section 6.3.7), using observations and insights gained from the simpler case.

#### 6.3.1 Notation and Definition

In this section, we address two types of attackers: the deterministic attacker and the non-deterministic attacker. The deterministic attacker has a fixed deterministic cost function  $C(k)$ , which is analyzed in Section 6.3.2. The non-deterministic attacker has a fixed cost  $c_a$  of attacking one TEE at one step with a certain probability  $p_s$  of obtaining one share of the key from the TEE at that step. We deal with them in Section 6.3.4 to Section 6.3.7.

In the smart contract bounty, the reward given to the attacker is determined by a reward function  $R(k, t)$ , where  $k$  is the number of shares that the proof is trying to prove against (i.e., the number of shares obtained by the attacker), and  $t$  is the submission time (which is the blockchain timestamp of the inclusion of the bounty-claiming transaction). Essentially, at time  $t$ , the attacker provides evidence of having acquired  $k$  shares. Since the signing key is rotated every  $T$  units of time and the signing key is secret-shared into  $N$  shares, we have  $t \in [0, T]$  and  $k \in [0, N]$ .

Recall that we use a  $(m, N)$  signature scheme. The service provider has  $N$  secrets shares, with  $\geq m$  of them together having value  $v$  for some  $m \leq N$ , and  $k < m$  of them have value  $v \cdot k/m$ .<sup>8</sup> Since  $m$  shares are enough to recover the key, the value of  $m$  or more shares is the wallet value (i.e.  $V(m) = V(m + 1) = \dots = V(N) = v$ ).

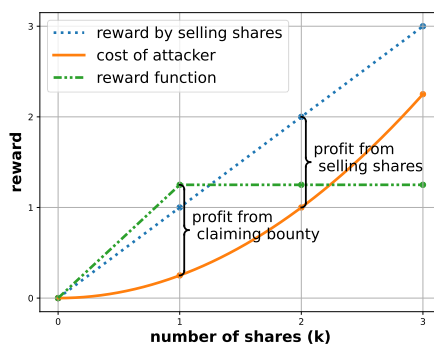
#### 6.3.2 Case study for deterministic attacker

We first provide a case study with respect to a simpler attacker: he has a deterministic cost function  $C(k)$ , which is non-decreasing in  $k$ , the number of acquired shares.

**Naive solution.** We start with a naive solution as briefly discussed in Section 1: the linear reward function. In other words,  $R(k, t) = V(k) + \eta_1$  for some  $\eta_1 > 0$ . This is a natural solution: it gives a bit more than how much the share(s) are worth. However, as mentioned, this naive solution can only achieve the goal (3), but not (4) or (5) proposed in Section 6.1. As analyzed, the attacker would continue to attack for more shares and only submit when he has all the key shares.

---

<sup>8</sup> Note that in some cases, it may also make sense that having  $k < m$  of them has no value. For generality, we consider them to have some partial value.



■ **Figure 3** Example of reward function in simplified case.

**A starting point.** Therefore, we propose first a simple solution that can achieve the goals 3-5 under such a deterministic attack (as the starting point for our real reward function):

$$R(k) = \max_{0 < k \leq N} (V(k) - C(k)) + C(1) + \eta_0 + (1 - t/T)\delta_0,$$

where  $\eta_0$  and  $\delta_0$  are small constant numbers serving as bonus. This reward function straightforwardly satisfies our goals. For goal (3): Submitting to the bounty provides the attacker with at least  $\eta_0$  more than selling the shares when the attacker submits with only one share. Consequently, there is no incentive for the attacker to sell the share. For goal (4): Since the adversary achieves maximum profit from the bounty by obtaining just one share  $\max_{0 \leq k \leq N} (V(k) - C(k)) + \eta_0 + (1 - t/T)\delta_0$ , and given that the bonus  $\delta_0$  decreases over time, the attacker is incentivized to submit the share to the bounty upon acquiring the first share (and since the adversary needs one share to submit,  $C(1)$  is used to compensate this cost). For goal (5): the defender's cost is minimized since the defender cannot spend less. If she reduces her expenditure by  $\eta_0$ , the adversary's gain from the reward might equal the profit from selling the key at point  $i$ , where the profit  $(V(k) - C(k))$  is maximized. This could lead the attacker to opt for selling the key. As a side property, the attacker also saves cost, as its total cost is always non-decreasing.

A concrete example is depicted in Figure 3. Here, the cost of attack is  $C(k) = \frac{1}{4}k^2$ , and the value of key shares is  $V(k) = k$ . The maximum profit for the attacker is  $\max_{0 \leq k \leq N} (V(k) - C(k)) = V(2) - C(2) = 1$ . We set  $\eta_0 = \delta_0 = 0.1$ . Therefore, the optimal reward function in this scenario is  $R(k) = C(1) + (V(2) - C(2)) + \eta_0 = 1.25 + \eta_0$ . By structuring the reward function in this way, we not only incentivize the attacker to submit the key share as soon as they get one share but also reduce the defense cost.

Let's compare the reward function we proposed with two baselines: a zero function  $R_0(k) = 0$  and a linear reward function  $R_l(k) = k + \eta_0$ . With  $R_0$ , the attacker accumulates 2 shares and sells them in the market, which violates goals 3 and 4. With  $R_l$ , the defender pays  $2 + \eta_0$  to prevent the attacker from selling 2 shares, which violates the goal 3 and costs more than our reward function.

The main observation from the case study is that giving the attacker more reward at first share is not only a good way to persuade the attacker not to further exploit the key, but also saves the defender's cost.

Of course, here, the context is greatly simplified: the attacker's cost is a known deterministic function of the number of key shares gained. If the attacker's cost is a probabilistic function, the reward function does not always achieve the goals. Also, even for a deterministic

attacker with a slightly different cost function, the reward function may not work anymore (e.g., if the attacker costs 10% less per share). Thus, we propose a more complete reward function in Section 6.3.4.

### 6.3.3 Metrics for Reward Function

While for the deterministic attacker, the simple reward function satisfies all the goals, it becomes more complicated for a non-deterministic attacker, and also when we want to protect against a wider range of attackers. There is a trade-off between goals 3-5 in Section 6.1. For example, it would cost more if we wanted to encourage the attacker to turn in the key shares to the bounty earlier. To address this, we turn the goals into numerical metrics and balance them using a weighted average.

We developed three metrics to evaluate how well the reward function meets each of the three specified goals. The first metric is the probability of key shares being sold, denoted as  $p_e$  (goal (3)). The second metric is the average holding time,  $t_h$ , representing the average time between the attacker finding the first share and the termination of the game (goal (4)). The third metric, the cost to the defender, is denoted as  $c_d$  (goal (5)). The cost of the defender is the max between the value the attacker gets by selling the  $k$  shares (i.e.,  $V(k)$ ) and the amount of the bounty claimed (recall that an attacker can only do one of the two instead of both). To combine these metrics into a score, denoted as  $f$ , we introduce parameters  $\alpha_1$  and  $\alpha_2$  to compute a weighted average.

$$f = \alpha_1 \cdot p_e + \alpha_2 \cdot \frac{t_h}{T} + (1 - \alpha_1 - \alpha_2) \cdot \frac{c_d}{v} \quad (1)$$

In Equation (1), the holding time is normalized by the time period  $T$  and the defender's cost is normalized by the value of the key  $v$ .

### 6.3.4 Propose reward function for non-deterministic attacker

We now propose a reward function designed to achieve the objectives outlined in Section 6.1 for a non-deterministic attacker. The optimization and evaluation of this proposed reward function will be detailed in the subsequent parts of this subsection.

To achieve goal (3) in Section 6.1, we need to give more reward to the attacker than the value of the shares. For an attacker with  $k$  shares of secret, he can gain  $V(k)$  units of money. Thus, to encourage the attacker to submit to the bounty, we give out more than the amount they should have received by selling the key shares. A non-deterministic attacker, however, may get lucky in some cases and get more than one share at a low cost. So our proposed function should have the property  $R(k, t) > V(k)$  for all  $k \in [1, N]$ .

Formally, we give a reward of  $V(N)^\epsilon \cdot V(k)^{1-\epsilon} + \eta$  (recall that  $dV/dk \geq 0$  for all  $k \in [N]$ ), for some  $\epsilon \in [0, 1], \eta > 0$ . As long as  $\epsilon \geq 0, \eta > 0$ , we have  $V(N)^\epsilon \cdot V(k)^{1-\epsilon} + \eta > V(k)$  for all  $k > 0$ . Note that when  $\epsilon$  increases, we give more reward when  $k = 1$ , which could potentially reduce the defender's cost (achieving the goal (5)) according to the case study above.

Finally, we need to encourage the adversaries to submit earlier to achieve the goal (4) in Section 6.1. Similarly, we set the ‘‘extra bonus’’ decreasing over time. Formally, let  $g(k) := V(N)^\epsilon \cdot V(k)^{1-\epsilon} - V(k)$  denote the extra reward we paid to the attacker. We reduce this gain by time, adding a term  $-g(k) \cdot t/T$ . The reward function we suggest is:

$$R(k, t) := V(N)^\epsilon \cdot V(k)^{1-\epsilon} + \eta - g(k) \cdot t/T \quad (2)$$

To model the real-world constraint of the defender's budget, we also introduce an additional parameter,  $\alpha_{\text{cap}}$ , into the reward function. This parameter represents the maximum amount of

money that the bounty can afford, expressed as a percentage of the secret’s value. Specifically, we add a bound  $\alpha_{\text{cap}} \cdot V(N)$  to our reward function  $R(k, t)$  (Equation (2)), and the resulting new reward function is:

$$\tilde{R}(k, t) = \begin{cases} R(k, t) & \text{if } R(k, t) < \alpha_{\text{cap}} \cdot V(N) \\ \alpha_{\text{cap}} \cdot V(N) & \text{if } R(k, t) \geq \alpha_{\text{cap}} \cdot V(N) \end{cases} \quad (3)$$

where  $t$  is the submission time and  $k$  is the number of submitted shares ( $t \in [0, T], k \in [0, N]$ ).

### 6.3.5 Modelling the non-deterministic attacker

To evaluate our function, we first need to model how an attacker behaves. To do this, we first describe the behavior of the attacker that can be modeled as the optimal stopping game. Then, we further find the optimal attacker strategy using a Markov decision process (MDP).

Moreover, with this evaluation result, the defender can quantitatively understand what range of attackers can be effectively prevented using this reward function. She can then change the parameters (e.g., the attacker’s ability to begin with and the budget) to modify the function accordingly.

**Attacker behavior.** We give a detailed description of the attacker’s decision process as follows. As in the preceding sections, we exclusively consider a single signing key that is shared among  $N$  TEEs. The time period during which the secret remains valid is divided into  $T$  discrete time steps. Each time step is further divided into two sub-steps, during which the attacker makes distinct choices: In the first sub-step, the attacker selects the number of TEEs to target during that step. In the second sub-step, the attacker decides whether to terminate the game (sell the shares or claim the bounty) or proceed to the next step. If an attacker decides to target a TEE in a given step, they have a success probability of  $p_s$  to acquire a key share from it, while incurring a fixed cost of  $c_a$ .

**Optimal stopping game.** We model an adversary as a player of an “optimal stopping” game [54, 50, 24]. Essentially, the optimal stopping game states the following: there is a sequence of random variables  $X_1, X_2, \dots$  whose distribution is assumed to be known; and there is a sequence of gain functions  $(Y_i)_{i \geq 1}$  which take the first  $i$  random variables as inputs (i.e.,  $Y_i(x_1, \dots, x_i)$  is a function over  $x_1 \leftarrow X_1, \dots, x_i \leftarrow X_i$ ). Then, the player observes the sequence of random variables one at a time, and for each step  $i$ , the player can either stop observing and claim the gain  $Y_i(x_1, \dots, x_i)$  or continue. The goal of the player is to optimize the expected gain. Note that this setting is essentially the same as our setting, where the random variables are the shares gotten by the adversary (e.g. if an attacker can obtain a share with probability  $p$  at step  $i$ ,  $X_i$  is a Bernoulli random variable returning 1 with probability  $p$  and 0 with probability  $1 - p$ ). Then,  $y_i$  is the profit the attacker can gain from all the shares he has obtained up to step  $i$ , which is the maximum between the value of the bounty and the value of selling these shares, less his cost up to step  $i$ . Although some specific forms of optimal stopping games have closed-form solutions (e.g., the secretary problem [19]), for more complex scenarios like ours, a typical approach to find the player’s optimal strategy is to model the game with Markov Decision Process (MDP) [54, 50].

**MDP.** We model the attacking process as an MDP, structuring it into discrete steps. At each step, the attacker decides the number of TEEs to target. The attacker also needs to determine the optimal time to end the attack and obtain their reward: after each step,

he must choose to either cease the attack and get the reward or continue attacking in the subsequent step.

We specify the state transition function and the reward function of the MDP as follows. The state of the MDP is defined by the tuple of the number  $k$  of shares gained by the attacker, the time slot  $t$ , and the sub-step in each time slot  $d \in \{0, 1\}$ . At state  $(t, k, 0)$ , the attacker needs to choose the number of TEEs (denoted as  $n$ ) to attack in this time slot. The state transitions to  $(t, k + \Delta k, 1)$ , where  $\Delta k$  is the number of key shares gained in this time slot. The number of newly gained key shares depends on the success rate  $p_s$  and the number of TEEs the attacker chooses to attack in that particular step. Specifically, the probability that the attacker gets  $i$  new shares in this time slot is  $Pr(\Delta k = i) = \binom{n'}{i} p_s^{n'} (1 - p_s)^{n' - i}$ , where  $n' = \max(n, m - k)$ . At state  $(t, k, 1)$ , the attacker faces a decision: either end the game by selling the key shares or submitting them to the bounty, or wait until the next time slot. If the attacker chooses to wait until the next time slot, the state will transition to state  $(t + 1, k, 1)$ . If the attacker chooses to sell the key shares or submit them to the bounty, the next state will be the termination state. When the time slot reaches the maximum time  $T$  at state  $(t, T, 1)$ , the next state will be the termination state.

At each step of the process, the attacker incurs a negative reward of  $-c_a \cdot n$ , representing the cost of the attacking  $n$  TEEs. The attacker gains a positive reward  $R(k, t)$  if he submits the key shares to the bounty. Alternatively, if he decides to sell the key shares, he gets  $V(k)$ . A summary of the transition and reward function of the decision problem is in Table 1.

■ **Table 1** Description of the state transition and reward matrix.

State $\times$ Action	State	Probability	Reward
$(k, t, 0) \times$ attack $n$ TEEs	$(k + i, t, 1)$	$Pr(\Delta k = i)$	$-n \cdot c_a$
$(k, t, 1) \times$ wait ( $t < T$ )	$(k, t + 1, 0)$	1	0
$(k, t, 1) \times$ wait ( $t = T$ )	termination	1	0
$(k, t, 1) \times$ turn in	termination	1	$R(k, t)$
$(k, t, 1) \times$ selling key	termination	1	$V(k)$

Utilizing the MDP solver [38], we are able to compute the attacker’s optimal strategy for a specific reward function. By examining this optimal strategy, we can obtain the metrics defined in Section 6.3.3 ( $f$  score). The  $f$  score then serves as the objective for optimizing the parameters within the reward function.

### 6.3.6 Optimize the Reward Function Parameter

In this part, we describe the methodology for deciding the optimal  $\epsilon$  within the reward function in Equation (2), with  $\alpha_{\text{cap}}$  as described in Equation (3).

Recall that our reward function  $\tilde{R}$  is determined by  $\alpha_{\text{cap}}$  (bounty cap) and  $\epsilon$  (determining the starting point of the reward). We assume  $\alpha_{\text{cap}}$  is some constant predefined by the defender, according to her budget.

We now explain our approach for identifying the optimal value of  $\epsilon$  with regard to the performance metric  $f$ . As the defender aims to minimize the cost of the defender, the probability that the attacker will sell the key on the market, and the holding time, the objective is to minimize the score  $f$ . When defending against an attacker, the service provider must first decide the parameters used in  $f$  ( $\alpha_1$  and  $\alpha_2$ ) and estimate the ability of the attacker by specifying  $p_s$  and  $c_a$ . Using the estimated parameters, an optimal  $\epsilon$  could be numerically computed. Specifically, we discretize  $[0, 1]$  into a sequence of evenly spaced



numbers, calculate a score for each  $\epsilon$ , and select the one corresponding to the lowest score.<sup>9</sup>

Upon determining the optimal  $\epsilon$  with estimated parameters, we examine how attackers of various abilities respond to the computed  $\epsilon$  in the next part. Specifically, these attackers might have different  $p_s, c_a$  compared to the initial estimates used for  $\epsilon$  optimization, representing a range of adversaries stronger or weaker than the initial expectation.

### 6.3.7 Evaluation Results

We compare the score  $f$  of different reward functions, including our reward function, the linear reward function (see below), and no bounty (reward function equals 0).

The linear reward function is a solution that satisfies the goal 3 without considering the cost. Recall that we introduced this naive solution in Section 1 and Section 6.3.2: in the linear reward function, the bounty claimer gets the exact value of share(s) plus a small bonus  $\eta_1$  to encourage turning in key share(s). We additionally set a time bonus  $\delta_1$  that decays with time and encourages early turn-in for the purpose of this case study (to break ties for attacker decisions in MDP), formally given as follows:  $R_l(k, t) = V(k) + (1 - t/T)\delta_1 + \eta_1$ . In our experiment,  $\delta_1 = 0.1$  and  $\eta_1 = 0.01$ . For our proposed reward function,  $\eta = 0.01$  as well.

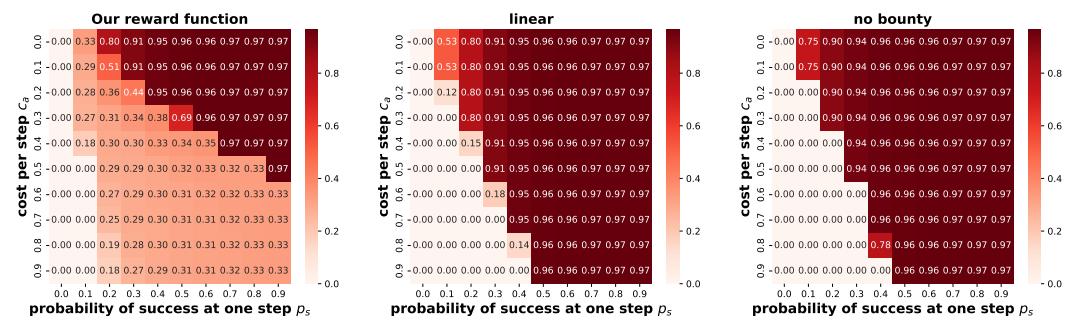


Figure 4 f score for different reward functions.  $\alpha_{\text{cap}} = 0.8$ .  $\alpha_1 = \alpha_2 = 1/3$ ,  $c_a = 0.4$ ,  $p_s = 0.4$ ,  $N = 3$ ,  $v = 6$ . Optimal  $\epsilon = 0.95$ .

In the evaluation, we set the estimation as  $c_a = 0.4$  and  $p_s = 0.4$ . We set the total number of key shares as  $N = 3$  and the value of the key as  $v = 6$ , which means the value per share is 2. In expectation, the cost incurred by the attacker to obtain one share is 1 (cost per step / probability of success), resulting in a positive expected profit of 1 for each share acquired. We set  $\alpha_1 = \alpha_2 = 1/3$  which means each metric has equal importance. The parameters can be replaced with real-world values when the wallet is implemented in practice. The optimal  $\epsilon$  we get is 0.95 given the parameters above. Then, we use the optimal parameter to derive the score for attackers with variant cost  $c_a$  and success rate  $p_s$ .

We show how this function behaves when facing different attackers in Figure 4, where each cell within the heatmap shows the  $f$  score corresponding to a specific configuration of the attacker’s capabilities, denoted by the parameters  $c_a$  and  $p_s$ . When the cost is low and the success rate is high (located in the upper right region of the heatmap), the attacker is considered strong. Conversely, when the cost is high and the success rate is low (positioned in the lower left area of the heatmap), the attacker is perceived as weak.

As we can see in the heatmap, when  $\alpha_{\text{cap}} = 80\%$ , the performance of the reward function we proposed (state of the art) is better than the baseline (no bounty and linear reward

<sup>9</sup> The precision is affected by how many intervals  $[0, 1]$  is discretized into.

function) in most cases. For most attackers, regardless of the ability, our reward function generates a smaller score. The figure demonstrates that our reward function has great performance not just for attackers whose abilities are equal to our estimations ( $c_a = 0.4$  and  $p_s = 0.4$ ), but it also works well for stronger attackers. As shown in the figure, essentially for *any*  $p_s$ , as long as  $c_a \geq 0.4$ , the  $f$  score is at most 0.3. Similar flexibility on  $c_a$  can also be seen in the graph. These results indicate that even without precise attacker ability estimations, our reward function outperforms the alternative reward functions and shows decent effectiveness in preventing outsider attacks.

As mentioned, the defender can then use the heatmap to determine the effectiveness of the reward function given the current attacker’s ability estimation and the budget. She may increase her budget to find a reward function that effectively defends against a broader spectrum if needed.

## 7 Case Study

We briefly discuss how to choose the parameters for the bounty in CrudiTEE using a simple case study. Recall that we need to set time  $T$ , the expected return given the number of shares  $V(k)$ , and the cost function  $C(k)$ . The calculation below assumes using a (10,20)-threshold signature scheme (i.e., 10 shares are enough to recover the secret) and  $T = 30$ .

To set the rest of the parameters, we first examine the state-of-the-art side-channel attacks against ECDSA. ECDSA [30] is the most commonly used signature scheme for blockchains like Bitcoin [8], and thus we use it as an example. To our knowledge, all the side-channel attacks without root privilege in recent years against the most popular ECDSA library (OpenSSL [44]) show that they require at least  $2^{12}$  traces to recover a secret [60, 21, 4]. Then, we let the service provider cap the number of signatures a user can make. According to [18], a regular user makes 68 bank transactions per month, which means  $\sim 2.3$  transactions per day. To be lenient, assume the victim makes 230 transactions per day (which is 100x the average number of transactions per day). Since recovering a key share requires at least  $2^{12}$  signatures, which takes  $\sim 17.8$  days. For  $V(k)$ , recall that we have a rate limit  $v$  for each wallet (i.e., the amount of money in each wallet). According to [23], each transaction’s average value is 36 dollars for a debit card. We thus set  $v = 36000$ , again 100x larger than the average transaction value. Each key share has equal value, and  $m = 10$  shares are enough to recover a key, we set  $V(k) = \min(\lceil v \cdot k/m \rceil, v)$ .

Lastly, we discuss the cost function. The cost function is the most tricky one, since it should capture all the possible costs of an attacker, including operational costs, the risk of being caught, the side channel being mitigated, and so on. Thus, we propose a conservative function (i.e., the minimum cost an attacker can have). Note that for an outsider, the minimum requirement is essentially getting to obtain the traces remotely. The most common way is residing on the same virtual machine as the victim program, as discussed in [48]. Thus, we estimate the cost using the cost of renting the same cloud machine as the service provider. Suppose that it costs  $c_{\text{cloud}}$  dollars per unit of time (e.g., c5.metal from AWS, a commonly used server instance, costs  $\sim \$97.9$  per day [6]). Thus, we have  $C(k) = c_{\text{cloud}} \cdot k \cdot 17.8$ .

These numbers give us that to recover a key with a value of 36000 dollars, the cost of the attacker is at least  $\sim 17426$  dollars (based on 17.8 days per share, a total of 10 shares, and 97.9 dollars per day for VM). We can come up with a reward function accordingly given all these numbers, along with their budget limit. More accurate numbers can be obtained for a specific service provider by analyzing their own transaction data.

## 8 Discussion

In this section, we discuss CrudiTEE’s performance, limitations and extension application.

**Performance Analysis.** Reasonable signing performance is required to make the scheme practical. A potential bottleneck of performance may be caused by the secret sharing between different TEEs. In this part, we analyze its concrete performance to show that the multi-TEE ECDSA signing will not be a bottleneck.

For the threshold ECDSA scheme proposed by Gennaro and Goldfeder [22],<sup>10</sup> the benchmark for the signature generation time among  $m$  participants is  $29 + 24m$  milliseconds. As benchmarked in [40], the highest overhead of TEE is  $19.31 \times$  in all the tasks tested. Therefore, a conservative signature generation time is around  $560 + 463m$  milliseconds. The protocol requires five rounds of communication and we estimate the communication delay for each round as 100 milliseconds [17]. Consequently, the total time for generating a threshold signature is about  $1060 + 463m$  milliseconds, which is generally acceptable for cryptocurrency wallets. Additionally, to accommodate high transaction volumes, we can employ multiple sets of TEEs in parallel.

**Limitations of insurance.** Our techniques provide a technical basis for penalizing the service provider when an attack succeeds against it, providing an incentive for it to properly safeguard its TEEs from outside attackers and a transparent and measurable guarantee to end users. These are significant improvements over the current status quo. Ensuring that the company deposits assets sufficient to satisfy claims against it is a matter for insurance regulators; today, insurance regulators in most jurisdictions require companies to maintain *statutory reserves*, i.e., an amount of cash and readily marketable securities that it can use to pay its foreseeable claims. As with other insurance in real life (e.g., property insurance), users in our system may not be compensated if these reserves (i.e., the company’s deposits) are depleted by other claims. Our technical solutions presented here cannot entirely eliminate the need for legal recourse in such situations. Nevertheless, our design provides a stronger foundation for reducing trust in a service provider and for reducing the risk to clients.

**Limitations on the type of assets.** Note that in most blockchains today, each wallet is tied to a specific private key. Thus, key updates after leakage can cause the assets in the wallet to be non-retrievable. In our paper, we require the asset to be tied to a smart-contract-based wallet, allowing the key updates to work as expected. How to extend our idea to support a wallet without such support remains open.

## 9 Conclusion

In this paper, we introduced CrudiTEE, a solution designed to mitigate side channels in TEE-based cryptocurrency wallets by leveraging economic incentives. Our wallet authentication system utilizes OAuth to ensure both accountability and user-friendliness. Additionally, we designed a combination of stick (insurance) and carrot (bounty) to safeguard against both insider and outsider attacks. Finally, we evaluated our approach and showed its effectiveness.

---

<sup>10</sup>This scheme considers malicious participants, so there are unnecessary steps in the protocol if we assume all the participants are honest, which is true in our case.

---

**References**

---

- 1 O. Aciicmez, W. Schindler, and Ç. K. Koç. Cache based remote timing attack on the AES. In *Topics in Cryptology – CT-RSA 2007, The Cryptographers’ Track at the RSA Conference 2007*, pages 271–286, February 2007.
- 2 AMD secure encrypted virtualization (SEV). URL: <https://www.amd.com/en/developer/sev.html>.
- 3 Ittai Anati, Shay Gueron, Simon Johnson, and Vincent Scarlata. Innovative technology for CPU based attestation and sealing. In *2nd International Workshop on Hardware and Architectural Support for Security and Privacy*, 2013.
- 4 Diego F. Aranha, Felipe Rodrigues Novaes, Akira Takahashi, Mehdi Tibouchi, and Yuval Yarom. Ladderleak: Breaking ecdsa with less than one bit of nonce leakage. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, CCS ’20*, pages 225–242, New York, NY, USA, 2020. Association for Computing Machinery. doi: 10.1145/3372297.3417268.
- 5 The sequencer and censorship resistance. URL: <https://docs.arbitrum.io/sequencer/#unhappyuncommon-case-sequencer-isnt-doing-its-job>.
- 6 AWS price calculator. <https://calculator.aws/>, 2023.
- 7 Kushal Babel, Nerla Jean-Louis, Mahimna Kelkar, Yunqi Li, Carolina Ortega Perez, Aditya Asgoankar, Sylvain Bellemare, Ari Juels, and Andrew Miller. The Sting framework (SF), 2023. URL: <https://initc3org.medium.com/the-sting-framework-sf-ef00702c88c7>.
- 8 Bitcoin core 25.0. <https://github.com/bitcoin/bitcoin>, 2023.
- 9 Ferdinand Brasser, Srdjan Capkun, Alexandra Dmitrienko, Tommaso Frassetto, Kari Kostainen, and Ahmad-Reza Sadeghi. Dr. SGX: Automated and adjustable side-channel protection for SGX using data location randomization. In *35th Annual Computer Security Applications Conference*, pages 788–800, 2019.
- 10 Lorenz Breidenbach, Phil Daian, Florian Tramèr, and Ari Juels. Enter the hydra: Towards principled bug bounties and exploit-resistant smart contracts. In *27th USENIX Security Symposium*, pages 1335–1352, 2018.
- 11 Ernie Brickell, Gary Graunke, and Jean-Pierre Seifert. Mitigating cache/timing attacks in AES and RSA software implementations. In *RSA Conference*, 2006.
- 12 D. Brumley and D. Boneh. Remote timing attacks are practical. *Computer Networks*, 48(5):701–716, 2005.
- 13 Iadine Chadès, Guillaume Chapron, Marie-Josée Cros, Frédéric Garcia, and Régis Sabbadin. Mdptoolbox: a multi-platform toolbox to solve stochastic dynamic programming problems. *Ecography*, 37(9):916–920, 2014.
- 14 Raymond Cheng, Fan Zhang, Jernej Kos, Warren He, Nicholas Hynes, Noah Johnson, Ari Juels, Andrew Miller, and Dawn Song. Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contracts. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 185–200. IEEE, 2019.
- 15 Coinbase. <https://www.coinbase.com/>.
- 16 Manuel Costa, Lawrence Esswood, Olga Ohrimenko, Felix Schuster, and Sameer Wagh. The pyramid scheme: Oblivious RAM for trusted processors. *arXiv preprint arXiv:1712.07882*, 2017.
- 17 Luca De Vito, Sergio Rapuano, and Laura Tomaciello. One-way delay measurement: State of the art. *IEEE Transactions on Instrumentation and Measurement*, 57(12):2742–2750, 2008.
- 18 Federal Reserve Bank of Atlanta. Survey of consumer payment choice 2020, 2020. URL: <https://www.atlantafed.org/-/media/documents/banking/consumer-payments/survey-of-consumer-payment-choice/2020/2020-survey-of-consumer-payment-choice.pdf>.
- 19 Thomas S. Ferguson. Who Solved the Secretary Problem? *Statistical Science*, 4(3):282–289, 1989. doi:10.1214/ss/1177012493.

- 20 K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic analysis: Concrete results. In *Cryptographic Hardware and Embedded Systems – CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261, May 2001.
- 21 Daniel Genkin, Lev Pachmanov, Itamar Pipman, Eran Tromer, and Yuval Yarom. Ecdsa key extraction from mobile devices via nonintrusive physical side channels. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS '16*, pages 1626–1638, New York, NY, USA, 2016. Association for Computing Machinery. doi:10.1145/2976749.2978353.
- 22 Rosario Gennaro and Steven Goldfeder. Fast multiparty threshold ECDSA with fast trustless setup. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1179–1194, 2018.
- 23 Geoffrey Gerdes, Claire Greene, Xuemei (May) Liu, Emily Massaro, Ambika Nair, Zach Proom, Nancy Donahue, Lisa Gillispie, Mary Kepler, Doug King, Susan Krupkowski, Ellen Levy, Dave Lott, Mark Manuszak, David Mills, Laura Reiter, Stephanie Scuiletti, Susan Stawick, Catherine Thaliath, Jessica Washington, and Julius Weyman. The 2019 federal reserve payments study. URL: <https://www.federalreserve.gov/paymentsystems/2019-December-The-Federal-Reserve-Payments-Study.htm>.
- 24 Alexander V. Gnedin and Ulrich Krengel. A stochastic game of optimal stopping and order selection. *Annals of Applied Probability*, 5:310–321, 1995. URL: <https://api.semanticscholar.org/CorpusID:122457776>.
- 25 Google LLC. Using OAuth2.0 with OpenID Connect in Google. URL: <https://developers.google.com/identity/openid-connect/openid-connect>.
- 26 Daniel Gruss, Clémentine Maurice, Klaus Wagner, and Stefan Mangard. Flush+flush: A fast and stealthy cache attack. In *DIMVA 2016: Detection of Intrusions and Malware, and Vulnerability Assessment*, volume 9721 of *Lecture Notes in Computer Science*, pages 279–299, 2016.
- 27 D. Gullasch, E. Bangerter, and S. Krenn. Cache games – bringing access-based cache attacks on AES to practice. In *32nd IEEE Symposium on Security & Privacy*, pages 490–505, 2011.
- 28 H100 tensor core GPU | NVIDIA. URL: <https://www.nvidia.com/en-us/data-center/h100/>.
- 29 Amir Herzberg, Stanislaw Jarecki, Hugo Krawczyk, and Moti Yung. Proactive secret sharing or: How to cope with perpetual leakage. In *Advances in Cryptology – CRYPTO '95*, volume 963 of *Lecture Notes in Computer Science*, pages 339–352, 1995.
- 30 Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ecdsa). In *International Journal of Information Security*. Association for Computing Machinery, July 2001. doi:10.1007/s102070100002.
- 31 P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397, August 1999.
- 32 P. C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Advances in Cryptology – CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113, 1996.
- 33 Jiasun Li. On the security of optimistic blockchain mechanisms. Available at SSRN 4499357, 2023.
- 34 Mengyuan Li, Yinqian Zhang, Huibo Wang, Kang Li, and Yueqiang Cheng. CIPHERLEAKS: Breaking constant-time cryptography on AMD SEV via the ciphertext side channel. In *30th USENIX Security Symposium*, August 2021.
- 35 Moritz Lipp, Andreas Kogler, David Oswald, Michael Schwarz, Catherine Easdon, Claudio Canella, and Daniel Gruss. Platypus: Software-based power side-channel attacks on x86. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 355–371, 2021. doi:10.1109/SP40001.2021.00063.

- 36 Suresh S Malladi and Hemang C Subramanian. Bug bounty programs for cybersecurity: Practices, issues, and recommendations. *IEEE Software*, 37(1):31–39, 2019.
- 37 Frank McKeen, Ilya Alexandrovich, Alex Berenzon, Carlos V. Rozas, Hisham Shafi, Vedvyas Shanbhogue, and Uday R. Savagaonkar. Innovative instructions and software model for isolated execution. In *2nd International Workshop on Hardware and Architectural Support for Security and Privacy*, HASP '13, page 1, New York, NY, USA, June 2013. Association for Computing Machinery. doi:10.1145/2487726.2488368.
- 38 Markov decision process (mdp) toolbox. URL: <https://pymdptoolbox.readthedocs.io/en/latest/api/mdptoolbox.html>.
- 39 ID tokens in the Microsoft identity platform. URL: <https://learn.microsoft.com/en-us/azure/active-directory/develop/id-tokens>.
- 40 Saeid Mofrad, Fengwei Zhang, Shiyong Lu, and Weidong Shi. A comparison study of intel sgx and amd memory encryption technology. In *Proceedings of the 7th International Workshop on Hardware and Architectural Support for Security and Privacy*, pages 1–8, 2018.
- 41 S.-J. Moon, V. Sekar, and M. K. Reiter. Nomad: Mitigating arbitrary cloud side channels via provider-assisted migration. In *22nd ACM Conference on Computer and Communications Security*, pages 1595–1606, October 2015.
- 42 M. Morbitzer, S. Proskurin, M. Radev, M. Dorfhuber, and E. Salas. Severity: Code injection attacks against encrypted virtual machines. In *2021 IEEE Security and Privacy Workshops (SPW)*, pages 444–455, Los Alamitos, CA, USA, May 2021. IEEE Computer Society. doi:10.1109/SPW53761.2021.00063.
- 43 Alexander Nilsson, Pegah Nikbakht Bideh, and Joakim Brorsson. A survey of published attacks on Intel SGX. *arXiv preprint arXiv:2006.13598*, 2020.
- 44 OpenSSL. <https://www.openssl.org/>, 2023.
- 45 D. A. Osvik, A. Shamir, and E. Tromer. Cache attacks and countermeasures: The case of AES. In *Topics in Cryptology – CT-RSA 2006*, volume 3860 of *Lecture Notes in Computer Science*, pages 1–20, 2006.
- 46 Aaron Parecki. OAuth 2.0 basic information. URL: <https://developers.google.com/identity/openid-connect/openid-connect>.
- 47 J.-J. Quisquater and D. Samyde. Electromagnetic analysis (EMA): Measures and countermeasures for smart cards. In *Smart Card Programming and Security, International Conference on Research in Smart Cards, E-smart 2001*, volume 2140 of *Lecture Notes in Computer Science*, pages 200–210, September 2001.
- 48 Thomas Ristenpart, Eran Tromer, Hovav Shacham, and Stefan Savage. Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS '09, pages 199–212, New York, NY, USA, 2009. Association for Computing Machinery. doi:10.1145/1653662.1653687.
- 49 Carlton Shepherd, Konstantinos Markantonakis, Nico van Heijningen, Driss Aboulkassimi, Clément Gainé, Thibaut Heckmann, and David Naccache. Physical fault injection and side-channel attacks on mobile devices: A comprehensive analysis. *Computers & Security*, 111:102471, 2021. doi:10.1016/j.cose.2021.102471.
- 50 Albert N. Shiryaev. *Optimal Stopping Rules*, pages 1032–1034. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. doi:10.1007/978-3-642-04898-2\_433.
- 51 Saurabh Suratkar, Mahesh Shirole, and Sunil Bhirud. Cryptocurrency wallet: A review. In *2020 4th international conference on computer, communication and signal processing (ICCCSP)*, pages 1–7. IEEE, 2020.
- 52 J. Szefer. Survey of microarchitectural side and covert channels, attacks, and defenses. *Journal of Hardware and Systems Security*, 3:219–234, September 2019.
- 53 Florian Tramèr, Fan Zhang, Huang Lin, Jean-Pierre Hubaux, Ari Juels, and Elaine Shi. Sealed-Glass Proofs: Using Transparent Enclaves to Prove and Sell Knowledge. In *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 19–34, April 2017. doi:10.1109/EuroSP.2017.28.

- 54 J.N. Tsitsiklis and B. van Roy. Optimal stopping of markov processes: Hilbert space theory, approximation algorithms, and an application to pricing high-dimensional financial derivatives. *IEEE Transactions on Automatic Control*, 44(10):1840–1851, 1999. doi:10.1109/9.793723.
- 55 Stephan van Schaik, Andrew Kwong, Daniel Genkin, and Yuval Yarom. SGAXe: How SGX fails in practice. <https://sgaxeattack.com/>, 2020.
- 56 Venkatanathan Varadarajan, Yinqian Zhang, Thomas Ristenpart, and Michael Swift. A placement vulnerability study in multi-tenant public clouds. In *24th USENIX Security Symposium*, August 2015.
- 57 Wenhao Wang, Guoxing Chen, Xiaorui Pan, Yinqian Zhang, XiaoFeng Wang, Vincent Bind-schaedler, Haixu Tang, and Carl A. Gunter. Leaky cauldron on the dark land: Understanding memory side-channel hazards in SGX. In *24th ACM Conference on Computer and Communications Security*, October 2017.
- 58 M. Weiß, B. Heinz, and F. Stumpf. A cache timing attack on AES in virtualization environments. In *16th International Conference on Financial Cryptography and Data Security*, February 2012.
- 59 Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.
- 60 Yuval Yarom and Naomi Benger. Recovering openssl ecdsa nonces using the flush+reload cache side-channel attack. *Cryptology ePrint Archive*, Paper 2014/140, 2014. URL: <https://eprint.iacr.org/2014/140>.
- 61 Martin Young. Coinbase custodies 11% of entire crypto capitalization. URL: <https://cointelegraph.com/news/coinbase-custodies-11-of-entire-crypto-capitalization>.
- 62 Zainan Victor Zhou and Matt Stam. Rc-5732: Commit interface: A simple but general commit interface to support commit-reveal scheme. <https://eips.ethereum.org/EIPS/eip-5732>, September 2022.
- 63 Dionysis Zindros. Hours of Horus: Keyless cryptocurrency wallets. *Cryptology ePrint Archive*, 2021.





# Cornucopia: Distributed Randomness at Scale

Miranda Christ  

Columbia University, New York, NY, USA

Kevin Choi  

New York University, NY, USA

Joseph Bonneau  

New York University, NY, USA

a16z crypto research, New York, NY, USA

---

## Abstract

We propose Cornucopia, a protocol framework for distributed randomness beacons combining accumulators and verifiable delay functions. Cornucopia generalizes the Unicorn protocol, using an accumulator to enable efficient verification by each participant that their contribution has been included. The output is unpredictable as long as at least one participant is honest, yielding a scalable distributed randomness beacon with strong security properties. Proving this approach secure requires developing a novel property of accumulators, *insertion security*, which we show is both necessary and sufficient for Cornucopia-style protocols. We show that not all accumulators are insertion-secure, then prove that common constructions (Merkle trees, RSA accumulators, and bilinear accumulators) are either naturally insertion-secure or can be made so with trivial modifications.

**2012 ACM Subject Classification** Security and privacy → Cryptography

**Keywords and phrases** Randomness beacons, accumulators

**Digital Object Identifier** 10.4230/LIPIcs.AFT.2024.17

**Related Version** *Full Version*: <https://eprint.iacr.org/2023/1554> [20]

**Funding** Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the United States Government, DARPA, Andreessen Horowitz, or any other supporting organization.

*Miranda Christ*: Supported by NSF grants CCF-2107187 and CCF-2212233, by LexisNexis Risk Solutions, and by the Algorand Centres of Excellence programme managed by Algorand Foundation.

*Kevin Choi*: Supported by DARPA Agreement HR00112020022 and NSF grant CNS-2239975.

*Joseph Bonneau*: Supported by DARPA Agreement HR00112020022, NSF grant CNS-2239975, and a16z crypto research.

**Acknowledgements** The authors thank Noemi Glaeser for suggesting the name Cornucopia.

## 1 Introduction

The goal of distributed randomness beacons (DRBs) is to enable  $n$  participants to jointly compute a random output (which we denote  $\Omega$ ) that cannot be predicted or biased by a malicious subset of the participants. Among many important applications of DRBs are cryptographically verifiable lotteries and leader election in consensus protocols.

A classic approach to constructing DRBs is *commit-reveal* [8]. First, all participants publish a cryptographic commitment to a random contribution  $r_i$ . Participants then reveal their  $r_i$  values and the result is  $\Omega = \text{Combine}(r_1, \dots, r_n)$  for some suitable combination function (such as exclusive-or or a cryptographic hash). Commit-reveal protocols are simple, efficient, and secure as long as any one participant chooses a random  $r_i$  and all participants open their commitments. However, the output can be biased via a so-called *last-revealer*



© Miranda Christ, Kevin Choi, and Joseph Bonneau;  
licensed under Creative Commons License CC-BY 4.0  
6th Conference on Advances in Financial Technologies (AFT 2024).

Editors: Rainer Böhme and Lucianna Kiffer; Article No. 17; pp. 17:1–17:23

Leibniz International Proceedings in Informatics



LIPIC Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

*attack*, in which a participant observes all other  $r_i$  values during the reveal phase and drops out if the impending value of  $\Omega$  is not to their liking. The protocol must either finish without the missing  $r_i$ , or restart completely. Either way, the attacker obtains 1 bit of bias on  $\Omega$ .

Most approaches to avoiding last-revealer attacks enable a majority coalition to recover a withholding participant's contribution. However, this downgrades the security model from requiring only honest participant to requiring an honest majority (to prevent a malicious coalition from privately computing  $\Omega$  early). Such protocols also typically require communication and computation superlinear in  $n$  (though some amortize this over multiple rounds).

A fundamentally different approach constructs DRBs uses time-based cryptography, specifically *delay functions*, to prevent manipulation. The simplest example is Unicorn [38], a one-round protocol in which each participant directly publishes (within a fixed time window) their contribution  $r_i$  to a public bulletin board. The result is computed as  $\Omega = \text{Delay}(\text{Combine}(r_1, \dots, r_n))$ . By assumption, a participant cannot compute the Delay function before the deadline to publish their  $r_i$  and therefore cannot choose  $r_i$  in such a way as to manipulate the output  $\Omega$ . This protocol retains the strong security model of commit-reveal, but with no last-revealer attacks. It is remarkably simple and, using modern verifiable delay functions [9], the result can be efficiently verified. The downside is that  $\Theta(n)$  contributions must be posted to the public bulletin board per protocol run.

**Improving efficiency with accumulators.** Unicorn is simple and robust, but requires publishing  $\Theta(n)$  data (one contribution per participant) on the public bulletin board. To reduce this cost to  $O(1)$ , we can instead publish a succinct commitment to all users' contributions using a cryptographic accumulator (for example, a Merkle tree). We formalize this approach as *Cornucopia*:

- Each participant sends their contribution  $r_i$  to a *coordinator* before a time deadline  $T_0$ .
- The coordinator accumulates all contributions into a succinct commitment  $R$  and publishes it to the bulletin board. It sends each user a proof  $\pi_i$  that their value  $r_i$  is included in  $R$ .
- After time  $t$  passes, the result  $\Omega = \text{Delay}(R)$  is published as well as a proof  $\pi_\Omega$ .
- Each user  $i$  checks both that their contribution  $r_i$  was included in  $R$  and that  $\Omega$  was properly computed from  $R$ .

While this is a small change to Unicorn, it is powerful. Since security requires only one honest participant there is no risk to allowing more participants. Honest participants need only verify that *they themselves participated in the protocol* (assuming they trust that their own device has not been compromised) and need not know about the full set of participants. The only downside to additional participants is performance, and Cornucopia's sub-linear verification cost means the approach is feasible for *open-participation* randomness protocols at planetary scale (i.e. millions or billions of participants). For example, every user buying a lottery ticket or every player in a massively multi-player online (MMO) game might contribute randomness and be convinced the process was fair.

A malicious coordinator and any number of other malicious participants in the protocol cannot manipulate the DRB output. A malicious coordinator might exclude all honest users from participating, but these users can easily see that they have been excluded and know not to trust the DRB output. For this reason, the coordinator can be viewed as *semi-trusted*; it is trusted for liveness but not for security. We could also consider the coordinator *malicious-but-cautious* [49], in that undermining liveness would be publicly detectable but biasing the DRB output would not be. In Section 7.1 we discuss extending Cornucopia to a multi-coordinator model with stronger liveness guarantees.

Performance-wise, the coordinator does face at least linear costs ( $\Omega(n)$ ) to compute the accumulator and per-user proofs, but for certain accumulators [52, 55], the coordinator can efficiently batch compute all users' witnesses.

**Related work.** There is a large and growing literature on randomness beacons, dating to the seminal proposal by Rabin [47] and foundational work on *distributed coin tossing* [21, 3, 4, 31, 23, 34, 33]. Several recent surveys cover modern DRBs [48, 19, 35]. Most of this work is orthogonal, as protocols without delay functions either require an honest majority [54, 17, 14, 7, 30, 32, 51, 22, 6, 24, 2] or offer only economic security [1, 46, 57].

Unicorn [38] introduced delay-based DRBs. Several extensions to Unicorn work in a similar model. Bicorn [18] extends Unicorn with a fast optimistic case, avoiding the delay function if *all* participants are honest. RandRunner [50] also enables avoiding a delay function per beacon output although it does not support flexible participation and allows a withholding leader to affect the protocol.

HeadStart [37] is the most conceptually similar approach to Cornucopia, using Merkle trees to scale Unicorn by combining many users' contributions in a succinct commitment in a multi-round, pipelined protocol. Cornucopia can be seen as a generalization of HeadStart, offering flexibility to use any accumulator and formalizing precise security notions required of accumulators for use in DRBs.

#### Our contributions.

- We formalize combining a VDF with an accumulator as Cornucopia (Section 3).
- We prove (in Section 4) that this approach is secure when instantiated with *any* VDF and *any* accumulator satisfying a natural security notion that we develop, *insertion security*.
- We prove (in Section 5) that the most common accumulator constructions either naturally feature insertion security (Merkle trees) or achieve it with trivial modifications (RSA accumulators, bilinear accumulators, and accumulators from vector commitments), meaning Cornucopia is practical to build from standard cryptographic assumptions and implementations. We also show that we can construct an insertion-secure accumulator generically from any universal accumulator (Section 5.6).
- We compare performance of different accumulators which can be used to instantiate Cornucopia in Section 6. No accumulator is clearly best in all settings, as different options offer different trade-offs of communication and computation cost.
- Finally, we discuss several natural extensions, including the multi-coordinator model to ensure liveness (Section 7.1) and a notarized model to provide verifiability to passive observers (Section 7.2).

## 2 Preliminaries

We use  $\lambda$  to denote a security parameter, and  $\text{poly}(\lambda)$  and  $\text{negl}(\lambda)$  to denote polynomial and negligible functions of  $\lambda$ , respectively. We let  $[k]$  denote the set  $\{1, \dots, k\}$ . We use  $\xleftarrow{\$}$  (or  $\xrightarrow{\$}$ ) to denote the output of a randomized algorithm, or sampling uniformly at random from a range. We use  $\alpha$  to denote an advice string passed from a precomputation algorithm to a later online algorithm. We assume all adversaries are limited to running in probabilistic polynomial time (PPT) in the security parameter  $\lambda$ ; some adversaries are further limited to running in  $\sigma(t)$  steps on at most  $p(t)$  parallel processors, as defined for VDF sequentiality [9]. Both VDFs [9] and accumulators [5] rely on public parameters  $\text{pp}$  which all functions require implicitly, though we will typically omit this for brevity.

$\mathcal{G}_{\mathcal{A}_0, \mathcal{A}_1, t, \text{VDF}}^{\text{sequential}}(\lambda)$ $\text{pp} \xleftarrow{\$} \text{VDF.Setup}(\lambda, t)$ $\alpha \xleftarrow{\$} \mathcal{A}_0(\text{pp})$ $x \xleftarrow{\$} U$ $\tilde{y} \xleftarrow{\$} \mathcal{A}_1(\alpha, x)$ $y, \pi \leftarrow \text{VDF.Eval}(\text{pp}, x)$ $\text{return } \tilde{y} = y$
---

■ **Figure 1** VDF sequentiality game.

## 2.1 Verifiable delay functions

► **Definition 1** (Verifiable delay function [9]). A verifiable delay function (VDF) is a tuple of algorithms (Setup, Eval, Verify) where:

**VDF.Setup**( $\lambda, t$ )  $\rightarrow$  **pp** takes as input  $\lambda$  and a time parameter  $t$  and outputs public parameters **pp**.

**VDF.Eval**(**pp**,  $x$ )  $\rightarrow$  ( $y, \pi$ ) takes as input  $x$  and produces an output  $y$  and optional proof  $\pi$ . This function should run in  $t$  sequential steps.

**VDF.Verify**(**pp**,  $x, y, \pi$ )  $\rightarrow$  {true, false} takes an input  $x$ , output  $y$ , and optional proof  $\pi$ , and returns true if ( $y, \pi$ ) is a genuine output of Eval.

VDFs must satisfy the following three properties:

**Verifiability.** The verification algorithm is efficient (at most polylogarithmic in  $t$  and  $\lambda$ ) and always accepts when given a genuine output from VDF.Eval.

**Uniqueness.** VDF evaluation must be a function, meaning that VDF.Eval is a deterministic algorithm and it is computationally infeasible to find two pairs  $(x, y), (x, y')$  with  $y \neq y'$  that VDF.Verify will accept.

**Sequentiality.** VDFs must impose a computational delay. Roughly speaking, computing a VDF successfully with non-negligible probability over a uniformly distributed challenge  $x$  should be impossible without executing  $t$  sequential steps. Formally (adapted from [9]):

► **Definition 2** (VDF sequentiality [9]). A VDF is  $(p, \sigma)$ -sequential if for all randomized algorithms  $\mathcal{A}_0$  which run in total time  $O(\text{poly}(t, \lambda))$ , and  $\mathcal{A}_1$  which run in parallel time  $\sigma(t)$  on at most  $p(t)$  processors:

$$\Pr \left[ \mathcal{G}_{\mathcal{A}_0, \mathcal{A}_1, t, \text{VDF}}^{\text{sequential}}(\lambda) = 1 \right] \leq \text{negl}(\lambda)$$

where  $\mathcal{G}_{\mathcal{A}_0, \mathcal{A}_1, t, \text{VDF}}^{\text{sequential}}(\lambda)$  is defined in Figure 1.

## 2.2 Accumulators

► **Definition 3** (Accumulator [5, 13]). Given a data universe  $U$ , an accumulator is a tuple of algorithms (Setup, Accumulate, GetMemWit, MemVer) where:

**Acc.Setup**( $\lambda$ )  $\rightarrow$  **pp** takes as input  $\lambda$  and outputs public parameters **pp**.

**Acc.Accumulate**( $S$ )  $\rightarrow$   $A$  takes as input a set  $S \subseteq U$  to be accumulated. It outputs  $A$ , an accumulator value for  $S$ .

**Acc.GetMemWit**( $S, A, x$ )  $\rightarrow$   $w$  takes as input a set  $S \subseteq U$ , an accumulator value  $A$  for  $S$ , and an element  $x \in S$ . It outputs a membership witness  $w$  for  $x$ .

**Acc.MemVer**( $A, x, w$ )  $\rightarrow$  {true, false} takes as input an accumulator value  $A$ , an element  $x$ , and a membership proof (membership witness)  $w$ . It outputs true if  $x$  is included in the accumulated set represented by  $A$  and false otherwise.

$\mathcal{G}_{\mathcal{A}, \text{Acc}}^{\text{acc}}(\lambda)$ $\text{pp} \xleftarrow{\$} \text{Acc.Setup}(\lambda)$ $S, x, w \xleftarrow{\$} \mathcal{A}(\text{pp})$ $A \leftarrow \text{Acc.Accumulate}(S)$ <b>return</b> $\text{Acc.MemVer}(A, x, w) \wedge x \notin S$
---

■ **Figure 2** Accumulator security game.

We describe here only the accumulator functionality necessary for our purposes. Accumulators generally also support an incremental **Update** function to add additional elements to the accumulated set and *dynamic* accumulators support a **Delete** function to remove elements [13]. Cornucopia does not require either capability; we assume in each run of the protocol the coordinator collects all randomness contributions (the set being accumulated), accumulates them in one batch operation and never deletes.

An accumulator is *correct* if **MemVer** always accepts for elements included in honestly accumulated sets. An accumulator is *computationally correct* if it is computationally infeasible to find a set such that an honestly generated inclusion proof for an element in that set does not verify. The key security property of an accumulator is that for an honestly generated accumulator value for some set  $S$ , it is infeasible to find a membership proof for an element not in  $S$ :

► **Definition 4** (Accumulator security [13]). *An accumulator  $\text{Acc}$  is secure if no PPT adversary  $\mathcal{A}$  can succeed with non-negligible probability in  $\mathcal{G}_{\mathcal{A}, \text{Acc}}^{\text{acc}}(\lambda)$  as defined in Figure 2.*

A *universal accumulator* [39] also supports non-membership proofs; that is, it supports two additional functions:

**Acc.GetNonMemWit**( $S, A, x'$ )  $\rightarrow w'$  takes as input a set  $S \subseteq U$ , an accumulator value  $A$  for  $S$ , and an element  $x' \notin S$ . It outputs a non-membership witness  $w'$  for  $x'$ .

**Acc.NonMemVer**( $A, x', w'$ )  $\rightarrow \{\text{true}, \text{false}\}$  takes as input an accumulator value  $A$ , an element  $x'$ , and a non-membership proof (non-membership witness)  $w'$ . It outputs **true** if  $x'$  is *not* included in the accumulated set represented by  $A$  and **false** otherwise.

For Cornucopia itself, a universal accumulator is not required as there is no reason for the coordinator to prove to that any value is *not* included. However, in Section 5.6 we show a generic transformation from any universal accumulator to an insertion-secure accumulator.

A universal accumulator is *correct* if, in addition to **MemVer** accepting for all included elements, **NonMemVer** accepts for all non-included elements. Security requires that no adversary can find valid membership and non-membership proofs for the same element:

► **Definition 5** (Universal accumulator security [39]). *A universal accumulator  $\text{Acc}$  is secure if for all PPT adversaries  $\mathcal{A}$ :*

$$\Pr \left[ \begin{array}{l} \text{pp} \xleftarrow{\$} \text{Acc.Setup}(\lambda) \\ A, x, w, w' \xleftarrow{\$} \mathcal{A}(\text{pp}) \\ \text{Acc.MemVer}(A, x, w) \wedge \text{Acc.NonMemVer}(A, x, w') \end{array} \right] \leq \text{negl}(\lambda)$$

### 2.3 Vector commitments

We present only the functionality of vector commitments necessary for our applications.

## 17:6 Cornucopia: Distributed Randomness at Scale

► **Definition 6** (Vector commitment [15]). Given a message space  $\mathcal{M}$ , a vector commitment is a tuple of algorithms including:

**KeyGen** $(\lambda, s) \rightarrow \mathbf{pp}$  takes in the security parameter  $\lambda$  and the size  $s$  of the committed vector, and outputs public parameters  $\mathbf{pp}$ .

**Com** $(m_1, \dots, m_s) \rightarrow C, \mathbf{aux}$  takes as input a vector of  $s$  messages in  $\mathcal{M}$ , and outputs a commitment  $C$  and some auxiliary information  $\mathbf{aux}$ .

**Open** $(m, i, \mathbf{aux}) \rightarrow \pi_i$  takes as input a message  $m \in \mathcal{M}$ , an index  $i$ , and some auxiliary information  $\mathbf{aux}$ . It outputs a proof  $\pi_i$  that the  $i^{\text{th}}$  component of the committed vector is  $m$ .

**Ver** $(C, m, i, \pi_i) \rightarrow \{\mathbf{true}, \mathbf{false}\}$  takes as input a commitment, a message  $m$ , an index  $i$ , and a proof that the  $i^{\text{th}}$  component of the committed vector is  $m$ . It outputs **true** if and only if the proof verifies.

A vector commitment must satisfy *correctness*, which requires that honestly generated proofs for correct components of honestly generated vector commitments verify, as well as *position binding*, which requires that an adversary cannot produce a (possibly maliciously formed) commitment and two proofs of distinct values for the same component.

► **Definition 7** (Position binding [15]). A vector commitment satisfies position binding if for all  $i \in [s]$  and for all PPT adversaries  $\mathcal{A}$ :

$$\Pr \left[ \begin{array}{l} \mathbf{pp} \xleftarrow{\$} \text{Acc.Setup}(\lambda) \\ C, m, m', i, \pi_i, \pi'_i \xleftarrow{\$} \mathcal{A}(\mathbf{pp}) \\ \text{Ver}(C, m, i, \pi_i) \wedge \text{Ver}(C, m', i, \pi'_i) \wedge m \neq m' \end{array} \right] \leq \text{negl}(\lambda)$$

### 3 Timed DRBs: Definitions and Constructions

We first define timed DRBs using a generalized syntax, building on the definitions of [18].<sup>1</sup>

► **Definition 8** (Timed DRBs). A *timed DRB protocol* is a tuple of algorithms (Setup, Prepare, Post, Finalize, Verify):

**Setup** $(\lambda, t) \xrightarrow{\$} \mathbf{pp}$ : The setup algorithm can be run once and outputs public parameters  $\mathbf{pp}$  used for multiple protocol runs.

**Prepare** $(\mathbf{pp}) \xrightarrow{\$} r_i$ : The prepare algorithm is run by each participant to produce a randomness contribution  $r_i$ . This contribution is submitted during the contribution phase, which is bounded in length by the time parameter  $t$ .

**Post** $(\{r_i\}) \rightarrow (R, \{\pi_i\})$ : The post algorithm is run by a coordinator immediately after the end of the contribution phase, producing a commitment  $R$  to all users' contributions and (optionally) a list of user-specific proofs  $\pi_i$ . Typically, this value  $R$  will be posted to a public bulletin board, whereas  $\pi_i$  will be made privately available.

**Finalize** $(\mathbf{pp}, R) \rightarrow (\Omega, \pi_\Omega)$ : The finalize algorithm is run after the post algorithm, evaluating a delay function on  $R$  to produce a final DRB output  $\Omega$  and (optionally) a proof  $\pi_\Omega$ . It is a deterministic algorithm running in time  $(1 + \epsilon)t$  for some small  $\epsilon$ .

**Verify** $(\mathbf{pp}, R, \Omega, \pi_\Omega, r_i, \pi_i) \rightarrow \{\mathbf{true}, \mathbf{false}\}$ : Individual users should verify both the final DRB output  $\Omega$  as well as that their contribution  $r_i$  was correctly included, possibly with the help of an auxiliary user-specific proof  $\pi_i$ .

<sup>1</sup> Note that our syntax here is specific to one-round timed DRBs. Some timed DRBs such as Bicorn [18] have an optional second communication round.

$\mathcal{G}_{\mathcal{A},t,b,\text{DRB}}^{\text{indist}}(\lambda)$ $\text{pp} \xleftarrow{\$} \text{Setup}(\lambda, t)$ $r_1 \xleftarrow{\$} \text{Prepare}(\text{pp})$ $\alpha_0 \xleftarrow{\$} \mathcal{A}_0(\text{pp})$ $\alpha_1, R, \pi_1 \xleftarrow{\$} \mathcal{A}_1(\alpha_0, r_1)$ $\Omega_0, \pi_0 \leftarrow \text{Finalize}(\text{pp}, R)$ $\Omega_1 \xleftarrow{\$} U$ $b' \xleftarrow{\$} \mathcal{A}_2(\alpha_1, \Omega_b)$ $\text{return } b = b'$ $\wedge \text{Verify}(\text{pp}, R, \Omega_0, \pi_0, r_1, \pi_1)$	$\mathcal{G}_{\mathcal{A},t,\text{DRB}}^{\text{unpred}}(\lambda)$ $\text{pp} \xleftarrow{\$} \text{Setup}(\lambda, t)$ $r_1 \xleftarrow{\$} \text{Prepare}(\text{pp})$ $\alpha_0 \xleftarrow{\$} \mathcal{A}_0(\text{pp})$ $\tilde{\Omega}, \pi_{\tilde{\Omega}}, R, \pi_1 \xleftarrow{\$} \mathcal{A}_1(\alpha_0, r_1)$ $\text{return Verify}(\text{pp}, R, \tilde{\Omega}, \pi_{\tilde{\Omega}}, r_1, \pi_1)$
---	---

■ **Figure 3** Security games for  $(p, \sigma)$ -indistinguishability (left) and  $(p, \sigma)$ -unpredictability (right).

A timed DRB has the following security properties (shown in Figure 3):

► **Definition 9** ( $(p, \sigma)$ -unpredictability). *The  $(p, \sigma)$ -unpredictability game tasks an adversary with predicting the final output  $\Omega$  exactly, allowing it control of all but a single honest participant (which publishes first). This adversary's computation is broken into two phases. In the precomputation phase, before the adversary sees the honest contribution  $r_1$ , it may run an algorithm  $\mathcal{A}_0$  that runs in time  $\text{poly}(\lambda, t)$ . This algorithm outputs some advice string. After seeing  $r_1$ , the adversary is limited to running for  $\sigma(t)$  steps on at most  $p(t)$  parallel processors, exactly like the adversary for VDF sequentiality (Definition 2). The adversary's advantage is:  $\text{Adv}_{\mathcal{A},t,\text{DRB}}^{\text{unpred}}(\lambda) = \Pr[\mathcal{G}_{\mathcal{A},t,\text{DRB}}^{\text{unpred}}(\lambda) = 1]$ .*

The  $(p, \sigma)$ -unpredictability property only guarantees the DRB output cannot be predicted exactly. We can define a stronger  $(p, \sigma)$ -indistinguishability property in which the adversary must distinguish a DRB output from random, again allowing the adversary control of all-but-one participants:

► **Definition 10** ( $(p, \sigma)$ -indistinguishability). *The  $(p, \sigma)$ -indistinguishability game is exactly like the  $(p, \sigma)$ -unpredictability game, except with an extra input bit  $b$ . The challenger provides the adversary the genuine output of `Finalize` if  $b = 0$  and a random output if  $b = 1$ . The adversary must, after running for at most  $\sigma(t)$  steps on at most  $p(t)$  parallel processors, output a guess  $b'$  for which output it received. We define the adversary's advantage as:*

$$\text{Adv}_{\mathcal{A},t,\text{DRB}}^{\text{indist}}(\lambda) = |\Pr[\mathcal{G}_{\mathcal{A},t,1,\text{DRB}}^{\text{indist}}(\lambda) = 1] - \Pr[\mathcal{G}_{\mathcal{A},t,0,\text{DRB}}^{\text{indist}}(\lambda) = 1]|$$

As observed by Boneh et al. [9], we can convert any timed DRB which satisfies  $(p, \sigma)$ -unpredictability into one with  $(p, \sigma)$ -indistinguishability by applying a random oracle to the output. Our main result (Theorem 14) shows Cornucopia is unpredictable, indistinguishability thus immediately follows in the random oracle model (Corollary 15).

### 3.1 Unicorn

As a warm-up, we succinctly describe Unicorn [38] as a timed DRB in our framework in Figure 4.<sup>2</sup> Intuitively, Unicorn is secure because every user can check that their value is included in the posted set  $\{r_i\}$ . A VDF is evaluated on a hash of this set. A single honest

<sup>2</sup> Note that the the original Unicorn proposal used the delay function Sloth, which computes modular square roots modulo a prime. We describe Unicorn here using a modern VDF instead [9].

$\text{Setup}(\lambda, t) \xrightarrow{\$} \text{pp}$ $\text{pp} \leftarrow \text{VDF.Setup}(\lambda, t)$ $\text{Prepare}() \xrightarrow{\$} r_i$ $r_i \xleftarrow{\$} U$ $\text{Post}(\{r_i\}) \rightarrow (R, \emptyset)$ $R \leftarrow \{r_i\}$ $\text{Finalize}(R) \rightarrow (\Omega, \pi_\Omega)$ $\Omega, \pi_\Omega \leftarrow \text{VDF.Eval}(H(R))$ $\text{Verify}(\text{pp}, R, \Omega, \pi_\Omega, r_i, \pi_i) \rightarrow \{\text{true}, \text{false}\}$ $\text{return } r_i \in R \wedge \text{VDF.Verify}(H(R), \Omega, \pi_\Omega)$	$\text{Setup}(\lambda, t) \xrightarrow{\$} \text{pp}$ $\text{pp} \leftarrow (\text{VDF.Setup}(\lambda, t), \text{Acc.Setup}(\lambda))$ $\text{Prepare}() \xrightarrow{\$} r_i$ $r_i \xleftarrow{\$} U$ $\text{Post}(\{r_i\}) \rightarrow (R, \{\pi_i\})$ $R \leftarrow \text{Acc.Accumulate}(\{r_i\})$ $\pi_i \leftarrow \text{Acc.GetMemWit}(\{r_j\}, R, r_i)$ $\text{Finalize}(R) \rightarrow (\Omega, \pi_\Omega)$ $\Omega, \pi_\Omega \leftarrow \text{VDF.Eval}(H(R))$ $\text{Verify}(\text{pp}, R, \Omega, \pi_\Omega, r_i, \pi_i) \rightarrow \{\text{true}, \text{false}\}$ $\text{return VDF.Verify}(H(R), \Omega, \pi_\Omega)$ $\wedge \text{Acc.MemVer}(R, r_i, \pi_i)$
--	--

■ **Figure 4** The Unicorn timed DRB protocol [38] (left) and the Cornucopia protocol (right).

user is enough to ensure this hashed value cannot have been precomputed by the adversary. Unicorn’s security is directly implied by our security proof for Cornucopia in Theorem 14, as Unicorn is a special case using the trivial “concatenation accumulator”.<sup>3</sup> The primary downside of Unicorn is the fact that  $|R| = \Theta(n)$ . The goal of Cornucopia is to achieve the same security as Unicorn while storing only  $\Theta(1)$  data on the public bulletin board.

### 3.2 Cornucopia

Cornucopia, shown in Figure 4, improves on Unicorn by having the coordinator accumulate all user contributions into a succinct commitment  $R$  using a cryptographic accumulator scheme (see Section 2). Because  $|R|$  does not grow with the number of participants, Cornucopia easily scales to many users with constant publishing costs. Our indistinguishability and unpredictability definitions ensure that the protocol is secure as long as a single honest user contributes, so any honest user can be convinced the final result is random as long as they are convinced that their contribution was included.

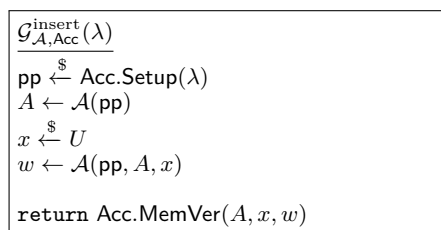
Note that our Cornucopia presentation and security definitions focus on security against manipulation and not on *liveness*; the coordinator can trivially block individual participants or even prevent the protocol from running at all. In Section 7.1 we revisit this and introduce the multi-coordinator model to ensure liveness even if all-but-one coordinators act maliciously.

## 4 Cornucopia Security

The security of Cornucopia relies on the adversary’s inability to predict the output of the VDF. This also requires that the adversary cannot produce an accumulator value satisfying an honest participant before seeing that participant’s randomness contribution. If it were able to do so, it could precompute the output of the VDF applied to this accumulator value and predict the output of the randomness beacon. However, the participant would still receive a valid proof that their contribution was included and believe that the randomness beacon was unpredictable. A trivial attack would be to accumulate the entire data universe, ensuring any user contribution could be proven “included.” To formalize this requirement, we define a novel security property for accumulators, called *insertion security*. We then prove that Cornucopia is secure when instantiated with any insertion-secure accumulator.

<sup>3</sup> Lenstra and Wesolowski prove security of Unicorn in a slightly different model [38].





■ **Figure 5** Insertion security game.

## 4.1 Insertion Security

Intuitively, an accumulator is insertion-secure if it is infeasible for any efficient adversary to accumulate a non-negligible fraction of the data universe. We formalize this property using an insertion security game, shown in Figure 5. To win the insertion security game, the adversary must produce an accumulator value  $A$  such that it can supply a membership proof for a randomly chosen element with non-negligible probability. Note that the adversary is not limited to producing  $A$  via the normal `Accumulate` function; it can compute  $A$  using any procedure at all. Using this game, insertion security is defined as follows:

► **Definition 11** (Insertion Security). *An accumulator is insertion-secure if for any PPT algorithm  $\mathcal{A}$ , the probability of  $\mathcal{A}$  winning the insertion security game (Figure 5) is negligible:*

$$\Pr [\mathcal{G}_{\mathcal{A}, \text{Acc}}^{\text{insert}}(\lambda) = 1] \leq \text{negl}(\lambda)$$

Although insertion security is (to our knowledge) a novel property of accumulators, it turns out that many constructions are naturally insertion-secure, as we will show in Section 5.

**Necessity of insertion security.** We will show that insecurity security is sufficient for Cornucopia in Theorem 14. We can also show insertion security is *necessary*. To see why, suppose that the underlying accumulator is not insertion-secure. The adversary is therefore able to produce some  $A$  such that with noticeable probability, it can efficiently compute a membership proof for a random element with respect to  $A$ . The adversarial coordinator precomputes  $\Omega = \text{VDF.Eval}(H(A))$  and predicts that this will be the beacon output. The coordinator then accepts randomness contributions from the participants, and in the `Post` protocol outputs  $A$  regardless of the values of these contributions. Now, consider some honest participant. With noticeable probability, the adversary is able to produce a membership proof with respect to  $A$  for their randomness contribution. Therefore, this honest participant accepts. However, this breaks security, as the adversary correctly predicted the output  $\Omega$ . Combined with our proof of Theorem 14, this shows that our definition is tight – insertion security is both necessary and sufficient.

**Incomparability with standard accumulator security.** We can show that insertion security is incomparable to standard accumulator security (Definition 4). Given any secure accumulator scheme  $\text{Acc}$ , one can construct an accumulator  $\text{Acc}'$  which is not insertion-secure, but otherwise satisfies the standard security definitions of an accumulator. One approach is to add a special symbol  $\epsilon$  which is defined as the accumulation of the entire data universe  $U$ .  $\text{Acc}'.\text{MemVer}(A, x, w)$  is defined to be 1 if  $A = \epsilon$  (regardless of the value of  $x$  or  $w$ ), and otherwise is equal to  $\text{Acc.MemVer}(A, x, w)$ . The scheme  $\text{Acc}'$  can be used exactly as  $\text{Acc}$  in normal operation, with the extra property that  $\epsilon$  is a “shortcut” to computing an accumulation of the entire data universe. We show later in Section 5 that some common schemes such as RSA and bilinear accumulators naturally feature this shortcut.

On the other hand, insertion security does not imply standard accumulator security. Recall that an accumulator is secure if an adversary cannot produce an honestly computed commitment  $A$  to a set  $S$ , an element  $x \notin S$ , and a valid membership proof for  $x$  with respect to  $A$ . Now, consider modifying an insertion-secure accumulator so that for a special element  $x^*$ , any witness is accepted; that is,  $\text{MemVer}(A, x^*, w)$  outputs true for all  $A$  and  $w$ . This resulting accumulator is still insertion-secure, as  $x^*$  is chosen as the challenge element with only negligible probability; however, it does not satisfy standard accumulator security as it is possible to provide a valid proof for  $x^*$  even if it was not in the genuinely accumulated set.

## 4.2 Security of Cornucopia

Before proving our main result (Theorem 14), we first prove two useful lemmas. The first is that if Cornucopia is constructed using an insertion-secure accumulator, an adversary cannot guess a satisfactory  $R$  before seeing the contribution  $r_1$  of the sole honest participant. Insertion security implies that it is difficult to precompute an accumulator value for which one can provide a membership proof of a random element. The second lemma states that if the adversary does not query  $R$  to the random oracle in its precomputation phase, it cannot output  $\tilde{\Omega} = \text{VDF.Eval}(H(R))$ . This is because after the precomputation phase, the adversary is  $(p, \sigma)$ -sequential and therefore cannot evaluate the VDF; thus, to prove this lemma we invoke VDF sequentiality. Together, these lemmas make it straightforward to prove that Cornucopia (CC for short) is secure given any insertion-secure accumulator and secure VDF.

► **Lemma 12.** *Let  $\mathcal{E}_1$  be the event that  $\mathcal{G}_{\mathcal{A},t,\text{CC}}^{\text{unpred}}(\lambda) = 1$  and  $\mathcal{A}_0$  queried  $R$  to the random oracle. If CC is instantiated with an insertion-secure accumulator, then  $\Pr[\mathcal{E}_1] \leq \text{negl}(\lambda)$ .*

**Proof.** Suppose for the sake of contradiction that for some constant  $c > 0$ ,

$$\Pr\left[\mathcal{G}_{\mathcal{A},t,\text{CC}}^{\text{unpred}}(\lambda) = 1 \wedge \mathcal{A}_0 \text{ queried } R \text{ to the random oracle}\right] \geq \frac{1}{\lambda^c}$$

We define an adversary  $\mathcal{B}$  that breaks insertion security of the accumulator scheme by simulating the challenger in  $\mathcal{G}_{\mathcal{A},t,\text{CC}}^{\text{unpred}}$  and using  $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ .  $\mathcal{B}$  first receives  $\text{Acc.pp}$  in  $\mathcal{G}_{\mathcal{B},\text{Acc}}^{\text{insert}}(\lambda)$ . It samples  $\text{VDF.pp} \leftarrow \text{VDF.Setup}(\lambda, t)$  and passes  $\text{pp} = (\text{Acc.pp}, \text{VDF.pp})$  to  $\mathcal{A}_0$ .  $\mathcal{B}$  simulates the challenger in  $\mathcal{G}_{\mathcal{A},t,\text{CC}}^{\text{unpred}}(\lambda)$  and records the queries  $q_1, \dots, q_k$  that  $\mathcal{A}_0$  makes to the random oracle.  $\mathcal{B}$  also receives  $\alpha_0$  as the output of  $\mathcal{A}_0$ .  $\mathcal{B}$  then chooses some query  $q_i$  uniformly at random from the queries made by  $\mathcal{A}_0$  and outputs  $A = q_i$  as its accumulator value in  $\mathcal{G}_{\mathcal{B},\text{Acc}}^{\text{insert}}(\lambda)$ .  $\mathcal{B}$  then receives  $x$  from the challenger in  $\mathcal{G}_{\mathcal{B},\text{Acc}}^{\text{insert}}(\lambda)$ , and it continues simulating the  $\mathcal{G}_{\mathcal{A},t,\text{CC}}^{\text{unpred}}(\lambda)$  challenger by passing  $\alpha_0$  and  $r_1 = x$  to  $\mathcal{A}_1$ .  $\mathcal{B}$  receives  $(\tilde{\Omega}, R, w_1)$  as the output of  $\mathcal{A}_1$ .

Since  $\mathcal{A}$  succeeds with at least probability  $\frac{1}{\lambda^c}$ ,

$$\Pr[\text{MemVer}(R, x, w_1) = \text{true} \wedge \mathcal{A}_0 \text{ queried } R \text{ to the random oracle}] \geq \frac{1}{\lambda^c}$$

Let  $q(\lambda)$  be some polynomial upper bounding the number of queries that  $\mathcal{A}_0$  makes to the random oracle; this polynomial must exist since  $\mathcal{A}_0$  runs in polynomial time. Since  $\mathcal{B}$ 's random choice of  $q_i$  is independent of  $\mathcal{A}$ ,  $\Pr[\text{MemVer}(R, x, w_1) = \text{true} \wedge A = R] \geq \frac{1}{\lambda^c} \cdot \frac{1}{q(\lambda)}$  which is non-negligible. Thus, with non-negligible probability,  $\mathcal{G}_{\mathcal{B},\text{Acc}}^{\text{insert}}(\lambda) = 1$ . ◀

► **Lemma 13.** *Let  $\mathcal{E}_2$  be the event that  $\mathcal{G}_{\mathcal{A},t,\text{CC}}^{\text{unpred}}(\lambda) = 1$  and  $\mathcal{A}_0$  did not query  $R$  to the random oracle. If CC is instantiated with an insertion-secure accumulator and a  $(p, \sigma)$ -sequential VDF, then  $\Pr[\mathcal{E}_2] \leq \text{negl}(\lambda)$ .*

**Proof.** Suppose for the sake of contradiction that for some constant  $c > 0$ ,

$$\Pr \left[ \mathcal{G}_{\mathcal{A},t,\text{CC}}^{\text{unpred}}(\lambda) = 1 \wedge \mathcal{A}_0 \text{ did not query } R \text{ to the random oracle} \right] \geq \frac{1}{\lambda^c}$$

We define an adversary  $\mathcal{B} = (\mathcal{B}_0, \mathcal{B}_1)$  that breaks  $(p, \sigma)$ -sequentiality of the VDF by simulating the challenger and random oracle in  $\mathcal{G}_{\mathcal{A},t,\text{CC}}^{\text{unpred}}$  and using  $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ . When  $\mathcal{A}$  evaluates the hash function it must query  $\mathcal{B}$ .  $\mathcal{B}$  responds in a way that is indistinguishable (to  $\mathcal{A}$ ) from a random function.

$\mathcal{B}_0$  first receives  $(\lambda, \text{VDF.pp}, t)$  from the VDF challenger in  $\mathcal{G}_{\mathcal{B}_0, \mathcal{B}_1, t, \text{VDF}}^{\text{sequential}}(\lambda)$ .  $\mathcal{B}_0$  samples  $\text{Acc.pp} \leftarrow \text{Acc.Setup}(\lambda)$  and passes  $\text{pp} = (\text{VDF.pp}, \text{Acc.pp})$  to  $\mathcal{A}_0$ .  $\mathcal{B}_0$  answers  $\mathcal{A}_0$ 's random oracle queries using uniformly random values. It records these queries and their responses in a list  $Q$ . If any query is repeated,  $\mathcal{B}_0$  answers consistently with its previous response in  $Q$ .  $\mathcal{A}_0$  outputs an advice string  $\alpha_0$ , which  $\mathcal{B}_0$  outputs as part of its advice string  $\alpha = (\alpha_0, Q)$ .

Now, the VDF challenger samples a random input  $x$  which is passed to  $\mathcal{B}_1$  along with  $\text{VDF.pp}$  and  $\alpha$ .  $\mathcal{B}_1$  passes  $\alpha_0$  and a randomly-generated value  $r_1 \xleftarrow{\$} \text{Prepare}(\text{pp})$  to  $\mathcal{A}_1$ .  $\mathcal{B}_1$  then simulates the random oracle for  $\mathcal{A}_1$ , with one key modification:  $\mathcal{B}_1$  chooses an index  $i \leq p(t) \cdot t$  uniformly at random<sup>4</sup> and answers  $\mathcal{A}_1$ 's  $i^{\text{th}}$  random oracle query  $q_i$  with  $x$  (provided that  $q_i$  has not been previously queried, otherwise it responds with the appropriate value from  $Q$ ). It answers any future repeated queries  $q_i$  similarly. For all other queries,  $\mathcal{B}_1$  answers randomly the first time and then consistent with its stored responses in  $Q$ . When  $\mathcal{A}_1$  outputs  $(\tilde{\Omega}, R, w_1)$ ,  $\mathcal{B}_1$  outputs  $\tilde{\Omega}$ .

**$\mathcal{B}$  properly simulates the random oracle.** Since  $x$  is a uniformly random value and all other queries receive random responses,  $\mathcal{B}_1$  does not change the output distribution of the random oracle and hence does not affect  $\mathcal{A}_1$ 's behavior.

**If  $\mathcal{A}$  succeeds,  $\mathcal{B}$  succeeds with non-negligible probability.** We now argue that if  $\mathcal{A}$  wins  $\mathcal{G}_{\mathcal{A},t,\text{CC}}^{\text{unpred}}$ ,  $\mathcal{B}$  wins  $\mathcal{G}_{\mathcal{B}_0, \mathcal{B}_1, t, \text{VDF}}^{\text{sequential}}(\lambda)$  with non-negligible probability. First, recall that if  $\mathcal{A}$  wins,  $\text{DRB.Verify}$  holds. By uniqueness of the VDF, the probability that  $\mathcal{A}_1$  outputs a proof  $\pi_\Omega$  such that  $\text{VDF.Verify}(\text{VDF.pp}, H(R), \tilde{\Omega}, \pi_\Omega) = 1$  yet  $\tilde{\Omega} \neq \text{VDF.Eval}(H(R))$  is negligible. Thus, since  $\text{DRB.Verify}$  holds,  $\mathcal{A}_1$  must have output  $\tilde{\Omega} = \text{VDF.Eval}(H(R))$ .

We now show that the fact that  $\mathcal{A}_1$  outputs  $\text{VDF.Eval}(H(R))$  implies that  $\mathcal{B}$  breaks  $(p, \sigma)$ -sequentiality of the VDF. Because the index  $i$  of the query to be replaced was chosen uniformly and independently of  $\mathcal{A}_1$ ,  $q_i$  was chosen to be the first instance that  $R$  was queried by  $\mathcal{A}_1$  with probability at least  $\frac{1}{p(t) \cdot t}$ . Since  $\mathcal{A}_0$  did not query  $R$ , we can indeed make this replacement. Therefore, with non-negligible probability  $\mathcal{B}_1$  simulates the random oracle to answer  $R$  with  $x$ , and  $\tilde{\Omega} = \text{VDF.Eval}(x)$  as desired.

Thus, for  $(\tilde{\Omega}, R, w_1)$  output by  $\mathcal{A}_1$ , it holds that

$$\Pr \left[ \tilde{\Omega} = \text{VDF.Eval}(H(R)) \wedge \mathcal{A}_0 \text{ did not query } R \text{ to the RO} \right] \geq \frac{1}{\lambda^c}$$

In the above, we assumed that  $\mathcal{A}_1$  queried  $R$  to the random oracle. If  $\mathcal{A}_1$  did not query  $R$  to the random oracle, it has anyways succeeded in computing the VDF output on  $H(R)$  which is a random value and identically distributed to  $x$ . ◀

Given these lemmas, we can now succinctly prove our main result:

<sup>4</sup> We use  $p(t) \cdot t$  as a generous upper bound on the number of random oracle queries made by  $\mathcal{A}_1$ , if every processor queries the oracle in every time step.

## 17:12 Cornucopia: Distributed Randomness at Scale

► **Theorem 14** (Unpredictability of Cornucopia). *Cornucopia is  $(p, \sigma)$ -unpredictable when instantiated with an insertion-secure accumulator, a  $(p, \sigma)$ -sequential VDF, and a hash function modeled as a random oracle.*

**Proof.** Let  $\mathcal{E}_1$  be the event that  $\mathcal{G}_{\mathcal{A},t,\text{CC}}^{\text{unpred}}(\lambda) = 1$  and  $\mathcal{A}_0$  queried  $R$  to the random oracle. Let  $\mathcal{E}_2$  be the event that  $\mathcal{G}_{\mathcal{A},t,\text{CC}}^{\text{unpred}}(\lambda) = 1$  and  $\mathcal{A}_0$  did not query  $R$  to the random oracle.

Observe that  $\Pr[\mathcal{G}_{\mathcal{A},t,\text{CC}}^{\text{unpred}}(\lambda) = 1] = \Pr[\mathcal{E}_1] + \Pr[\mathcal{E}_2]$ . By Lemma 12,  $\Pr[\mathcal{E}_1] \leq \text{negl}(\lambda)$ . By Lemma 13,  $\Pr[\mathcal{E}_2] \leq \text{negl}(\lambda)$ . Therefore,  $\Pr[\mathcal{G}_{\mathcal{A},t,\text{CC}}^{\text{unpred}}(\lambda) = 1] \leq \text{negl}(\lambda)$ . ◀

► **Corollary 15.** *Cornucopia is  $(p, \sigma)$ -indistinguishable when a random oracle is applied to its output.*

### 5 Insertion-secure accumulators

We now turn to the question of instantiating accumulators satisfying insertion security (Definition 11).

#### 5.1 Accumulators without insertion security

Recall from Section 4.1 that one can construct accumulators that have a shortcut  $\epsilon$  that accumulates the entire data universe. RSA accumulators naturally feature such a shortcut:  $\epsilon = 1$ . A valid membership witness for any  $x$  is  $w = 1$ , since  $w^x = 1^x = 1$ . Although we will prove RSA accumulators can easily be made insertion-secure by disallowing an accumulator value of 1, technically they are not insertion-secure as commonly specified. Bilinear accumulators have the same shortcut, which we remove with the same modification.

A second example, potentially of practical interest, is a *range accumulator*. A range accumulator can be defined from any accumulator scheme and for any data universe with a known total ordering (for example, any fixed subset of the integers such as  $\{0, 1\}^k$ ). With a range accumulator, the value  $H(x, y)$  can be accumulated, which is interpreted as adding a range  $[x, y]$  (the value  $H(x, x)$  can be accumulated to add a single element  $x$ ). Given any value  $z$ , proving membership can be achieved by providing a witness  $w' = (w, x, y)$  where  $w = \text{Acc.GetMemWit}(S, A, H(x, y))$  for  $x \leq z \leq y$ . This concept is quite natural and efficient, though it is also trivially not insertion-secure: an adversary can win  $\mathcal{G}_{\mathcal{A},\text{Acc}}^{\text{insert}}(\lambda)$  with probability 1 by accumulating the value  $H(x_{\min}, x_{\max})$  for the smallest and largest data elements in  $U$ , effectively accumulating the entire data universe in constant time.<sup>5</sup>

#### 5.2 Merkle trees

► **Lemma 16.** *A Merkle tree of bounded depth  $k = \text{poly}(n)$  is insertion-secure in the random oracle model.*

**Proof.** We work in the random oracle model, supposing that the Merkle tree uses a random oracle  $\mathcal{O} : \{0, 1\}^{2^n} \rightarrow \{0, 1\}^n$ . Let  $A$  be the accumulator output by an adversary  $\mathcal{A}$  in  $\mathcal{G}_{\mathcal{A},\text{Acc}}^{\text{insert}}(\lambda)$ . We show that for a uniform  $x \in \{0, 1\}^n$ , the adversary can provide a verifying witness  $w = (w_1, \dots, w_k)$  for  $x$  with only negligible probability. For a verifying witness, it must hold that  $\mathcal{O}(w_k || \dots || \mathcal{O}(w_2 || \mathcal{O}(w_1 || x))) = A$ . We'll show that with overwhelming probability (over choice of  $x$ ), no query to  $\mathcal{O}$  involved in the witness verification was made by the adversary in step 2 of  $\mathcal{G}_{\mathcal{A},\text{Acc}}^{\text{insert}}(\lambda)$ .

---

<sup>5</sup> The adversary can in fact win with non-negligible probability by accumulating any range whose size is a constant fraction of  $|U|$ .

This can be shown by induction. Let  $a_1, \dots, a_\ell$  be the adversary's queries to the random oracle in step 2. Let  $b_1, \dots, b_k$  be the queries to the random oracle in the Merkle membership proof verification; that is,  $b_i = w_i || \mathcal{O}(w_{i-1} || \dots)$ . Let  $p(\lambda)$  be a polynomial upper bound on the total number of queries made by the adversary to the random oracle throughout the game. Observe first that  $\Pr[b_1 = a_j \text{ for some } j] = \frac{\ell}{2^\lambda}$  since  $b_1 = w_1 || x$  and  $x$  is chosen at random. Assume that the probability that  $b_i$  is equal to any  $a_j$  is at most  $\frac{i\ell \cdot p(\lambda)}{2^\lambda}$ . If this event does not occur, then  $\mathcal{O}(b_{i+1}) = \mathcal{O}(w_{i+1} || \mathcal{O}(b_i))$  is a freshly random value, and the probability that  $b_{i+1} = a_j$  for any  $j$  is at most  $\frac{\ell \cdot p(\lambda)}{2^\lambda}$  (since  $\mathcal{A}$  can try up to  $p(\lambda)$  values for  $w_{i+1}$ ).

$$\begin{aligned} \Pr[b_{i+1} = a_j \text{ for some } j] &\leq \frac{\ell \cdot p(\lambda)}{2^\lambda} \Pr[b_i \neq a_j \text{ for all } j] \\ &\quad + \Pr[b_i = a_j \text{ for some } j] \\ &\leq \frac{\ell \cdot p(\lambda)}{2^\lambda} + \frac{i\ell \cdot p(\lambda)}{2^\lambda} \\ &= \frac{(i+1)\ell \cdot p(\lambda)}{2^\lambda} \end{aligned}$$

since  $\Pr[b_i = a_j \text{ for some } j] \leq \frac{i\ell \cdot p(\lambda)}{2^\lambda}$  by assumption. Therefore, the probability that any of the (polynomially bounded)  $k$  queries involved in witness verification was queried in step 2 is at most  $\frac{k\ell \cdot p(\lambda)}{2^\lambda} \leq \text{negl}(\lambda)$ .

In order for witness verification to pass, the last query must match the root; that is,  $\mathcal{O}(w_k || \dots || \mathcal{O}(w_2 || \mathcal{O}(w_1 || x))) = A$ . Since the above argument shows that  $(w_k || \dots || \mathcal{O}(w_2 || \mathcal{O}(w_1 || x)))$  was never queried in step 2, at the end of which  $\mathcal{A}$  outputs  $A$ , for each choice of  $w_k$ ,  $\mathcal{O}(w_k || \dots || \mathcal{O}(w_2 || \mathcal{O}(w_1 || x)))$  is a uniformly random value independent of  $A$  and equals  $A$  with only negligible probability. ◀

### 5.3 RSA accumulators

In a standard RSA accumulator [13, 40],  $\text{Setup}(\lambda)$  generates a random group of unknown order and a generator  $g$  for this group using some group generation algorithm  $\text{GenGroup}$ . The data universe is  $\Pi_\lambda$ , the set of all  $\lambda$ -bit primes. The accumulator value for a set  $S$  is  $A = g^{\prod_{x \in S} x}$ , and the witness  $w$  for an element  $x$  for the value  $A$  is  $w = g^{\prod_{x' \in S \setminus \{x\}} x'} = A^{1/x}$ .  $\text{Add}(A_t, x)$  outputs  $A_{t+1} = A_t^x$ . Thus, the accumulator value for a set  $S$  can be obtained by starting with the value  $A_0 = 1$  and adding each  $x_i \in S$  to  $A_{i-1}$  to obtain  $A_i$ , repeating until we reach  $A_{|S|}$ .  $\text{UpdWit}(A_t, x, w'_t)$  outputs  $w'_{t+1} = (w'_t)^x$ .  $\text{MemVer}(A, x, w)$  outputs 1 if and only if  $w^x = A$ . A non-membership witness for  $x$  with respect to  $A = g^{\prod_{s \in S} s}$  is  $\{a, B\}$  where  $a$  and  $b$  are Bézout coefficients for  $(x, \prod_{s \in S} s)$ , and  $B = g^b$ .  $\text{NonMemVer}(A, \{a, B\}, x)$  outputs 1 if and only if  $A^a B^x = g$ .

To make RSA accumulators insertion-secure, we add a second condition to  $\text{MemVer}(A, x, w)$ : It now outputs 1 if and only if  $w^x = A$  and  $A \neq 1$ . Note that our requirement that  $A \neq 1$  is necessary to reduce insertion security to the Adaptive Root Assumption.

► **Assumption 17** (Adaptive Root Assumption [10]).

$$\Pr \left[ \begin{array}{l} \mathbb{G} \xleftarrow{\$} \text{GenGroup}(\lambda) \\ (v, st) \leftarrow \mathcal{A}_0(\mathbb{G}) \\ u^l = v \neq 1 : \quad l \xleftarrow{\$} \Pi_\lambda = \text{Primes}(\lambda) \\ u \leftarrow \mathcal{A}_1(v, l, st) \end{array} \right] \leq \text{negl}(\lambda)$$

► **Lemma 18.** *Suppose a standard RSA accumulator is modified so that the algorithm  $\text{MemVer}(A, x, w)$  outputs 1 if and only if  $w^x = A$  and  $A \neq 1$ . The modified RSA accumulator is insertion-secure if the Adaptive Root Assumption holds for the group generation algorithm  $\text{GenGroup}$ .*

**Proof.** Suppose that there exists a PPT adversary  $\mathcal{A}$  that wins  $\mathcal{G}_{\mathcal{A}, \text{Acc}}^{\text{insert}}(\lambda)$  with probability at least  $\frac{1}{\text{poly}(\lambda)}$  when the data universe is  $\Pi_\lambda$ , the set of all  $\lambda$ -bit primes. We construct a pair of adversaries  $\mathcal{B}_0, \mathcal{B}_1$  that uses  $\mathcal{A}$  to break the Adaptive Root Assumption.  $\mathcal{B}_0$  draws  $\mathbb{G} \xleftarrow{\$} \text{GenGroup}(\lambda)$ .  $\mathcal{B}_0$  passes  $\mathbb{G}$  to  $\mathcal{A}$  and obtains an accumulator value  $A$ .  $\mathcal{B}_0$  outputs  $v = A$  and  $st$  as its current state.  $\mathcal{B}_1$  draws a random  $l \xleftarrow{\$} \Pi_\lambda$  and passes  $x = l$  to  $\mathcal{A}$ .  $\mathcal{A}$  outputs an alleged witness  $w_x$  which  $\mathcal{B}_1$  outputs directly as  $u$  in the Adaptive Root Game.

Recall that if  $\mathcal{A}$  wins  $\mathcal{G}_{\mathcal{A}, \text{Acc}}^{\text{insert}}(\lambda)$ , it means that  $\text{MemVer}(A, x, w_x) = \text{true}$ . For RSA accumulators,  $\text{MemVer}(A, x, w_x) = \text{true}$  if and only if  $(w_x)^x = A$  and  $A \neq 1$ . This implies that  $u^l = v$  where  $v \neq 1$ , and  $(\mathcal{B}_0, \mathcal{B}_1)$  win the Adaptive Root Game. Since  $\mathcal{A}$  wins with probability at least  $\frac{1}{\text{poly}(\lambda)}$ ,  $(\mathcal{B}_0, \mathcal{B}_1)$  win with probability at least  $\frac{1}{\text{poly}(\lambda)}$ , violating the Adaptive Root Assumption. ◀

► **Corollary 19.** *The modified RSA accumulator is insertion-secure in the Algebraic Group Model (AGM), since the Adaptive Root Assumption holds in the AGM [25].*

## 5.4 Bilinear accumulators

We show that bilinear accumulators [42, 53] with a small modification are insertion-secure in the AGM, under the Bilinear  $q$ -Strong Diffie-Hellman Assumption. The standard bilinear accumulator was defined by Nguyen [42], and we follow [44] in its presentation. Let  $\mathbb{G}, \mathcal{G}$  be cyclic multiplicative groups of prime order  $p$ , and let  $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathcal{G}$  be a bilinear pairing. Let  $s \xleftarrow{\$} \mathbb{Z}_p^*$ , and let  $g$  be a generator of  $\mathbb{G}$ . Let  $\text{srs} = [g, g^s, \dots, g^{s^q}]$  be the structured reference string, where  $q$  is an (polynomial in  $\lambda$ ) upper bound on the number of accumulated elements. The public parameters are  $(p, \mathbb{G}, \mathcal{G}, e, g, \text{srs})$ . Note that  $s$  must be kept secret even to the coordinator, and therefore a trusted setup is required.

This accumulator has data universe  $U = \mathbb{Z}_p^* \setminus \{-s\}$ . To accumulate a set  $X \subset U$ , where  $|X| \leq q$ , one computes  $A = g^{\prod_{x_i \in X} (x_i + s)}$ . The witness for an element  $x \in X$  is  $W = g^{\prod_{x_i \in (X \setminus \{x\})} (x_i + s)}$ . To verify a witness, one checks that  $e(W, g^{s+x}) = e(A, g)$ . To make this accumulator insertion-secure, we also check that  $A \neq 1$ .

In the Algebraic Group Model (AGM) [28], the adversary is constrained to perform only algebraic operations within the given group. That is, the adversary is given some group elements as input, and for any element that it outputs, it must provide a description of the operations used to obtain that element. In our setting, the algebraic adversary is given as input  $[1, g, g^s, \dots, g^{s^q}]$ . For any group element  $h$  that the adversary outputs, it must provide a scalar vector  $v \in \mathbb{Z}_p^*$  such that  $h = \prod_{i=0}^q g^{v_i \cdot s^i}$ . We refer the reader to [28, 29] for a more formal definition. Observe that the  $v_i$ 's can be interpreted as the coefficients of a polynomial of degree  $q$  evaluated at  $s$ . We use this interpretation in the following proof.

► **Assumption 20** ( $q$ -Discrete Logarithm Assumption ( $q$ -DLOG) [28]). *The  $q$ -DLOG assumption holds in a group  $\mathbb{G}$  if for every p.p.t. adversary  $\mathcal{A}$ ,*

$$\Pr_{s \leftarrow \mathbb{Z}_p^*} \left[ \mathcal{A} \left( g, g^s, \dots, g^{s^q} \right) \rightarrow s \right] \leq \text{negl}(\lambda).$$

► **Lemma 21.** *The bilinear accumulator of [42] is insertion-secure in the AGM, under the  $q$ -DLOG Assumption.*

**Proof.** Let  $\mathcal{A}$  be an algebraic adversary that takes  $\text{srs}$  as input and outputs  $A$  such that with non-negligible probability,  $\mathcal{A}$  can produce a verifying witness  $W$  for a randomly chosen  $x \in \mathbb{Z}_p^*$ . Since  $\mathcal{A}$  is algebraic, it must output vectors which we interpret as polynomials  $\alpha(S), w(S)$  of degree at most  $q$  such that  $A = g^{\alpha(s)}$  and  $W = g^{w(s)}$ . Since the witness verifies,  $e(W, g)^{(s+x)} = e(g^{\alpha(s)}, g)$ ; that is,  $e(g, g)^{w(s)(s+x)} = e(g, g)^{\alpha(s)}$ . Furthermore,  $\alpha(S)$  is a nonzero polynomial since verification requires that  $A \neq 1$ .

Observe that since  $x$  is chosen randomly from an exponentially large set, and  $\alpha$  is a nonzero polynomial of polynomially bounded degree,  $(S+x)$  divides  $\alpha(S)$  with only negligible probability by the Schwartz-Zippel lemma. Therefore,  $w(S)(S+x) - \alpha(S)$  is a nonzero polynomial that has  $s$  as a root. The adversary can factor  $w(S)(S+x) - \alpha(S)$  in polynomial time to find  $s$ . ◀

## 5.5 From vector commitments

Vector commitments (VCs) [15] can be used to construct an insertion-secure accumulator for sets of bounded size  $\leq k$  for any  $k$  polynomial in  $\lambda$ . Let the message space  $\mathcal{M}$  underlying our VC have size exponential in  $\lambda$ , and assume there is some total ordering over  $\mathcal{M}$ . To accumulate a set  $S \subseteq \mathcal{M}$ , we order this set to obtain a vector and commit to this vector. The witness for an element  $x \in S$  is an index  $i \leq k$  and a VC opening proof for that index. To verify this witness, one verifies the opening proof. This scheme is detailed below:

**Setup**( $\lambda$ ): Output  $\text{pp} \leftarrow \text{VC.Setup}(\lambda)$ .

**Accumulate**( $S$ ): Interpret  $S$  as an ordered list  $s_1, \dots, s_{|S|}$ , and let  $v = [s_1, \dots, s_{|S|}, 0, \dots, 0]$  be a vector of length  $k$ . Compute  $C, \text{aux} \leftarrow \text{VC.Commit}(v)$ .

**GetMemWit**( $S, A, x$ ): Compute  $C, \text{aux}$  from  $S$  as above. Let  $i$  be such that  $x = s_i$ . Compute  $\pi_i \leftarrow \text{VC.Open}(x, i, \text{aux})$  and output  $(i, \pi_i)$ .

**MemVer**( $A, x, (i, \pi_i)$ ): Output  $\text{VC.Ver}(A, x, i, \pi_i)$ .

*Position binding* of vector commitments says that it is infeasible for a PPT adversary to produce *any* (possibly maliciously-generated)  $A$ , distinct values  $x, x'$ , an index  $i$ , and accepting proofs  $\pi_i, \pi'_i$  that the vector committed to by  $A$  has  $x$  and  $x'$  respectively as its  $i^{\text{th}}$  component. We prove insertion security by showing that an adversary that breaks insertion security of this accumulator can be used to break position binding of the underlying VC scheme.

► **Theorem 22.** *When constructed with a vector commitment over an exponentially large data universe, this accumulator scheme is insertion-secure.*

**Proof.** Suppose that  $\Pr \left[ \mathcal{G}_{\mathcal{A}, \text{Acc}}^{\text{insert}}(\lambda) = 1 \right]$  is non-negligible. Let  $\mathcal{E}_i$  denote the event that  $\mathcal{A}$  outputs a proof for index  $i$ . Then there must be some accumulator  $A$  and index  $i$  such that

$$\Pr_{\substack{\text{pp} \leftarrow \text{Setup}(\lambda) \\ A \leftarrow \mathcal{A}(\text{pp})}} \left[ \Pr \left[ \mathcal{G}_{\mathcal{A}, \text{Acc}}^{\text{insert}}(\lambda) = 1 \wedge \mathcal{E}_i \mid \text{pp}, A \right] \geq \frac{1}{\lambda^{c_1}} \right] \geq \frac{1}{\lambda^{c_2}}$$

for some constants  $c_1, c_2 > 0$ .

Consider drawing  $\text{pp} \leftarrow \text{Setup}(\lambda)$  and running  $\mathcal{A}(\text{pp})$  to obtain  $A$ . As stated above, with non-negligible probability, there exists some  $i$  such that with non-negligible probability given this choice of  $\text{pp}$ ,  $A$  the adversary produces a verifying proof for index  $i$ . Consider running  $\mathcal{A}$  twice from this point, for two independently drawn  $x_1, x_2 \leftarrow U$ . With probability at least  $\frac{1}{\lambda^{2c_1}}$ ,  $\mathcal{A}$  produces verifying opening proofs  $\pi_1, \pi_2$  that the  $i^{\text{th}}$  index of the committed vector equals  $x_1$  and  $x_2$  respectively. Since  $U$  is exponentially large,  $x_1 \neq x_2$  with overwhelming

probability. Therefore, we have found a vector commitment  $A$  and proofs  $\pi_1, \pi_2$  that the same component takes on two distinct values, contradicting position binding of the vector commitment.  $\blacktriangleleft$

## 5.6 From generic universal accumulators

Finally, we show how to construct an insertion-secure accumulator  $\text{Acc}'$  from any universal accumulator  $\text{Acc}$ . The core idea is to map each element  $x$  to two pseudorandom sets  $(S_x^+, S_x^-)$ , each a subset of the data universe  $U$ . Proving membership of  $x$  for  $\text{Acc}'$  requires showing *inclusion* of all elements of  $S_x^+$  in  $\text{Acc}$  and *exclusion* of all elements of  $S_x^-$  in  $\text{Acc}$ . Intuitively, breaking insertion security by accumulating the entire data universe in  $\text{Acc}$  does not work because it will make the required non-membership proofs impossible. The best attacker strategy is to accumulate a random subset of half the elements of  $U$ , but this will mean that each item in  $S_x^+$  is wrongly excluded with probability  $\frac{1}{2}$  and each item in  $S_x^-$  is wrongly included with probability  $\frac{1}{2}$ . By setting ensuring the sizes of  $S_x^+, S_x^-$ , we can amplify security to ensure such an adversary has only a negligible probability of correctly showing inclusion of a random element.

In more detail, let  $\text{Acc}$  be a universal accumulator scheme for data universe  $U$ . Here, we let the data universe for  $\text{Acc}'$  be  $U' = \{0, 1\}^\lambda$ . Let  $H : [\lambda] \times U' \rightarrow U$  be a hash function that we will model as a random oracle. For any  $x \in U'$ , let  $S_x^+ := \{y : H(i, x) = y \text{ for } i \in [\frac{\lambda}{2}]\}$ , and let  $S_x^- := \{y : H(i, x) = y \text{ for } i \in \{(\frac{\lambda}{2} + 1), \dots, \lambda\}\}$  (assume for convenience that  $\lambda$  is even). We specify the functions of  $\text{Acc}'$  as follows:

**Setup:** uses the same setup function as  $\text{Acc}$ .

**Accumulate( $S'$ ):** Let  $S = \bigcup_{x \in S'} S_x^+$ . Outputs  $A = \text{Acc.Accumulate}(S)$ .

**GetMemWit( $S', \mathbf{A}, \mathbf{x}$ ):** Outputs a vector of witnesses  $\mathbf{w}$  of length  $\lambda$  where:

- For  $i \leq \frac{\lambda}{2}$ ,  $w_i = \text{Acc.GetMemWit}(S, A, H(i, x))$  is a membership proof for  $H(i, x)$
- For  $i > \frac{\lambda}{2}$ ,  $w_i = \text{Acc.GetNonMemWit}(S, A, H(i, x))$  is a non-membership proof for  $H(i, x)$

**MemVer( $A, x, \mathbf{w}$ ):** Outputs true if and only if the following holds for all  $i \in [\lambda]$ :

- For  $i \leq \frac{\lambda}{2}$ ,  $\text{Acc.MemVer}(A, H(i, x), w_i) = \text{true}$ .
- For  $i > \frac{\lambda}{2}$ ,  $\text{Acc.NonMemVer}(A, H(i, x), w_i) = \text{true}$ .

► **Lemma 23.** *If  $\text{Acc}$  is a secure universal accumulator and  $H$  is modeled as a random oracle,  $\text{Acc}'$  is insertion-secure.*

**Proof.** Suppose for the sake of contradiction that  $\text{Acc}'$  is not insertion-secure, and let  $\mathcal{A}$  be an adversary that wins the insertion game with probability at least  $\frac{1}{\lambda^c}$  for some constant  $c > 0$ , conditioned on the event that it does not query  $x$  before it outputs  $A$ . (Since  $\mathcal{A}$  is polynomially-bounded, this event fails to occur with only negligible probability). Thus, treating  $H$  as a random oracle,  $H(x)$  is a  $\lambda$ -length tuple of truly random independent values  $y_i \in U$ , where  $y_1, \dots, y_{\frac{\lambda}{2}}$  should be included, and  $y_{\frac{\lambda}{2}+1}, \dots, y_\lambda$  should be excluded.

Equivalently, we can think of drawing  $\mathbf{y} = y_1, \dots, y_\lambda$  (uniform and i.i.d. from  $U$ ) and subsequently drawing a uniformly random vector  $\mathbf{b}$  of Hamming weight  $\frac{\lambda}{2}$ , where  $y_i$  should be included if and only if  $b_i = 1$ .

By an averaging argument, we must have that for a non-negligible fraction of  $\mathbf{y} \in X$ ,  $\mathcal{A}$  succeeds with non-negligible probability over subsequent choice of  $\mathbf{b} \in \{0, 1\}^\lambda$ . Let  $\mathcal{E}[A, \mathbf{y}, \mathbf{b}, \mathbf{w}]$  denote the event that  $\text{Acc.MemVer}(A, y_i, w_i) = \text{true}$  for all  $i$  such that  $b_i = 1$ ,



and  $\text{Acc.NonMemVer}(A, y_i, w_i) = \text{true}$  for all  $i$  such that  $b_i = 0$ . The success of  $\mathcal{A}$  in  $\mathcal{G}_{\mathcal{A}, \text{Acc}'}^{\text{insert}}(\lambda)$  implies that  $\mathcal{E}[A, \mathbf{y}, \mathbf{b}, \mathbf{w}]$  occurs for its choice of  $A$  and  $\mathbf{w}$ , and the random choice of  $\mathbf{y}, \mathbf{b}$ . Thus,

$$\Pr_{\substack{\mathbf{pp} \leftarrow \text{Setup}(\lambda) \\ \mathbf{y}}} \left[ \Pr_{\mathbf{b}} \left[ \mathcal{A} \text{ outputs } A \text{ such that} \right. \right. \\ \left. \left. \Pr_{\mathbf{w}} [\mathbf{w} \leftarrow \mathcal{A} \wedge \mathcal{E}[A, \mathbf{y}, \mathbf{b}, \mathbf{w}]] \geq \frac{1}{\lambda^c}] \geq \frac{1}{\lambda^c} \right] \geq \frac{1}{\lambda^c}$$

We now construct an adversary  $\mathcal{B}$  that breaks universal security of  $\text{Acc}$  by producing an accumulator value, an element, and both membership and non-membership proofs for that element. Let  $\mathcal{B}$  first generate setup parameters and run  $\mathcal{A}$  on these parameters to obtain an accumulator value  $A$ . Let  $\mathcal{B}$  choose  $\mathbf{y}$  as above and  $\mathbf{b}_1, \mathbf{b}_2$  uniformly random vectors of Hamming weight  $\frac{\lambda}{2}$ .  $\mathcal{B}$  runs  $\mathcal{A}$  on inputs  $(\mathbf{y}, \mathbf{b}_1)$  and  $(\mathbf{y}, \mathbf{b}_2)$  to obtain  $\mathbf{w}_1$  and  $\mathbf{w}_2$  respectively. With probability at least  $\frac{1}{\lambda^c}$ ,  $\mathcal{B}$  chose  $\mathbf{pp}$  and  $\mathbf{y}$  such that  $\Pr_{\mathbf{b}} [\mathbf{w} \leftarrow \mathcal{A} \wedge \mathcal{E}[A, \mathbf{y}, \mathbf{b}, \mathbf{w}]] \geq \frac{1}{\lambda^c}$ . In this event, the probability that both  $\mathbf{w}_1$  and  $\mathbf{w}_2$  verify is at least  $\frac{1}{\lambda^{2c}}$ . As  $\mathbf{b}_1 = \mathbf{b}_2$  with only negligible probability (since  $\binom{n}{n/2} \geq 2^{n/2}$ ), with overwhelming probability there is some  $i$  such that  $(b_1)_i \neq (b_2)_i$ . However, we have (without loss of generality) both that  $\text{Acc.MemVer}(A, y_i, (w_1)_i) = \text{true}$  and  $\text{Acc.NonMemVer}(A, y_i, (w_2)_i) = \text{true}$ . This happens with probability at least  $\frac{1}{\lambda^c} \cdot \frac{1}{\lambda^{2c}} \cdot (1 - \frac{1}{2^\lambda})$ , which is non-negligible. This contradicts universal security of  $\text{Acc}$ .  $\blacktriangleleft$

**Correctness.** Accumulators typically require *correctness*, which says that given an honestly-generated accumulator value for a set, honestly-generated membership proofs for elements in that set should verify under  $\text{MemVer}$ ; similarly, honestly-generated non-membership proofs for elements not in that set should verify under  $\text{NonMemVer}$ . We note that  $\text{Acc}'$  has only *computational* correctness, since there may be some  $x_1, x_2$  for which the same  $y$  is included in  $S_{x_1}^+$  and  $S_{x_2}^-$ . This is problematic, since the membership proofs for  $x_1, x_2$  would require a membership proof *and* a non-membership proof for  $y$  (with respect to  $\text{Acc}$ ), which should be difficult by security of  $\text{Acc}$ , and hence  $x_1$  and  $x_2$  cannot both be included in the accumulator. In Cornucopia, if one user chose  $x_1$  and another user chose  $x_2$ , the coordinator could not satisfy both users.

Fortunately, collision resistance of  $H$  ensures that actually finding such  $x_1, x_2$  is computationally hard: finding  $x_1, x_2$  such that  $y \in S_{x_1}^+$  and  $y \in S_{x_2}^-$  would involve finding  $i_1 \neq i_2$  such that  $y = H(i_1, x_1) = H(i_2, x_2)$ , which yields a collision of  $H$ . Computational correctness is sufficient for use in Cornucopia (and most other applications), as polynomially-bounded users would not be able to find  $x_1$  and  $x_2$  resulting in the above issue.

## 6 Efficiency comparison of accumulator constructions

Cornucopia can be constructed from any insertion-secure accumulator. In Table 1 we compare efficiency trade-offs between Merkle trees, RSA accumulators, bilinear accumulators, and a construction from a vector commitment called Hyperproofs. All of these schemes require only  $O(1)$  space on the public bulletin board, regardless of the number of participants, though the concrete size varies. No accumulator construction offers obviously superior performance, each offers different trade-offs which might be attractive for different practical applications. For very large deployments (e.g. millions or billions of users) the performance bottleneck is likely inclusion proof generation by the coordinator.

## 17:18 Cornucopia: Distributed Randomness at Scale

■ **Table 1** Comparison of accumulator options for Cornucopia, at a security level of  $\lambda = 128$  bits. Witness generation time is the time required to compute all  $n$  witnesses.

†RSA accumulators can be instantiated using class groups [41], which do not require trusted setup. We report numbers here for the classic RSA group  $\mathbb{Z}_N^*$ .

Scheme	Trusted setup?	commitment  (bytes)	Witness size		Public params  (asympt.)	Witness gen. time (asympt.)
			(asympt.)	(bytes)		
Merkle tree	no	32	$O(\log n)$	$32 \cdot \lceil \log n \rceil$	$O(1)$	$O(n \log n)$
RSA Accumulator	yes <sup>†</sup>	384	$O(1)$	384	$O(1)$	$O(n^2)$
Bilinear Accumulator	yes	48	$O(1)$	48	$O(n)$	$O(n \log n)$
Hyperproofs [52]	yes	48	$O(\log n)$	$48 \cdot \lceil \log n \rceil$	$O(n)$	$O(n \log n)$

**Merkle trees.** Merkle trees are optimal in terms of the commitment size (32 bytes), require no trusted setup or public parameters and are naturally post-quantum secure. They are also the most efficient for the coordinator to compute witnesses, both in asymptotic and concrete terms. The only downside of Merkle trees is logarithmic witness sizes. Overall, we expect this to be the simplest and best approach for many applications, unless clients are extremely bandwidth-limited or the number of users is very large.

**RSA accumulators.** By contrast, RSA accumulators offer constant witness sizes, potentially offering the capability to scale to more users without imposing extra bandwidth requirements on clients. However, we note that the large size of RSA groups considered to offer 128-bit security (3072 bit moduli) means that Merkle proofs are shorter in practice with fewer than  $\approx 2^{12}$  users participating. RSA proofs also require computing modular exponentiation on large integers. This is relatively poorly supported by today’s smart contract platforms like EVM, but we observe that these only ever need to be verified off-chain by users. Still, proof verification is expected to be roughly an order of magnitude slower than Merkle proofs which only require hashing (though both are very efficient in concrete terms).

Furthermore, the size of the public commitment is over 10 times larger than for Merkle trees. This cost can be significant if the public bulletin board is an L1 blockchain such as Ethereum, where every 32-byte word stored on-chain costs over US\$2 at today’s gas prices. RSA accumulators also impose the highest costs on the coordinator ( $O(n^2)$ ) to compute witnesses, which may limit scalability.

RSA accumulators also require a trusted setup. This can be done for traditional RSA groups  $\mathbb{Z}_N^*$  as a multiparty ceremony [16]. Deployments may also use class groups of imaginary quadratic order [12, 41], which avoid the need for trusted setup but have higher concrete overhead and lack well-understood security parameters.

Finally, we note that there may be interesting optimizations when combining RSA accumulators with RSA-based VDFs [45, 56], such as offering a combined proof of inclusion and VDF evaluation.

**Bilinear accumulators.** Bilinear accumulators can offer the combination of small (48 byte) commitments and constant-sized membership proofs (48 bytes) along with the same asymptotic efficiency as Merkle trees for computing membership proofs ( $O(n \log n)$ ). Bilinear accumulators offer higher concrete overhead than for Merkle trees. In particular, they require pairing operations which are relatively expensive compared to hashing (though still cheap in concrete terms). However, only a single pairing operation by verifiers is required.

The downside is that bilinear accumulators require a trusted setup of an  $O(n)$ -sized structured reference string. This powers-of-tau string is common to many protocols and there are many approaches to generating it in a distributed manner [36, 43]. For example,

the Filecoin setup generated  $2^{27}$  powers of tau which can be used in a bilinear accumulator with up to  $2^{27} \approx 130$  million participants [26]. Ethereum generated a smaller string with  $2^{12}$  powers of tau in a community setup [27]. While the coordinator must store this entire string, participants need only access  $O(1)$  terms to verify that their contributions were included.

**Hyperproofs.** Finally, Hyperproofs [52] is a vector commitment scheme with the feature that witnesses can be generated in batch very efficiently – generating all  $n$  witnesses takes  $O(n \log n)$  time. Concretely, computing all  $n$  witnesses takes 0.7 hours for  $n = 2^{22}$  and 2.7 hours for  $n = 2^{24}$  as implemented in [52]. Verifying witnesses takes on the order of milliseconds. This efficiency is immediately inherited by the accumulator constructed using our approach in Subsection 5.5. The drawback of Hyperproofs is that it requires linear-sized public parameters that must be generated using a trusted setup. Merkle trees and bilinear accumulators also allow all witnesses to be batch computed in  $O(n \log n)$  time.

## 7 Concluding Discussion

Cornucopia is a simple but powerful framework for VDF-based DRBs, using accumulators to construct open-participation randomness beacon protocols at massive scale. Our work shows that this paradigm is secure, and it can be instantiated with efficient accumulators which are already in common practical use (see Section 6). We discuss important practical extensions to the Cornucopia framework, leaving a complete analysis to future work.

### 7.1 The multi-coordinator model

Basic Cornucopia is entirely dependent on the (single) coordinator to achieve liveness; a malicious coordinator could prevent targeted individuals from contributing to the protocol (censorship), or even withhold the commitment  $R$  and prevent the protocol from finishing at all. This does not undermine our DRB security definitions (Section 3) since the coordinator cannot do so conditionally based on the impending outcome, but they can arbitrarily bias the outcome if they successfully block all honest participants.

A natural way to ensure liveness is to allow  $k > 1$  coordinators, each of which posts an accumulator value  $R_i$ . The final beacon output is then computed as  $\Omega = \text{VDF.Eval}(H(R_1 || \dots || R_k))$ . In the limit, every user might be their own coordinator ( $k = n$ ), in which case the protocol is exactly the original Unicorn proposal [38]. Any number of malicious coordinators cannot undermine security of the protocol as long as at least one honest contributor submits a value to one honest coordinator.

If *any* coordinator is honest, the protocol will finish, hence we can achieve liveness if any of  $k$  coordinators is honest. Users can submit contributions to multiple coordinators and trust the final output  $\Omega$  as long as at least one coordinator includes their contribution. Combined, we can achieve 1-out-of- $k$  liveness and 1-out-of- $n$  security (for  $n$  contributors and  $k$  coordinators) for  $k \leq n$ .

While security and liveness are maximal with  $k = n$ , we note that in blockchain deployments the on-chain cost is  $O(k)$ , hence choosing  $k \ll n$  is likely required for efficiency considerations. Furthermore, the consequences of a security failure are more severe than a liveness failure, and a liveness failure will be visible on-chain whereas manipulating a randomness beacon is typically impossible to detect.

In a blockchain setting, there is no need to fix the set of coordinators; any party can act as a coordinator as long as they are willing to pay the cost (e.g. gas) of posting their accumulation  $R_i$  to the bulletin board. Coordinators can even use different accumulators

with different efficiency trade-offs. For example, a user participating across many epochs may prioritize shorter witnesses and prefer a coordinator using a bilinear accumulator with constant-sized witnesses but a trusted setup. Another user who participates only once may opt for a coordinator using a Merkle tree, requiring an  $O(\log n)$ -sized witness but avoiding the need for a trusted setup.

## 7.2 Public verifiability with notaries

As proposed, Cornucopia only offers meaningful security guarantees to participants who have contributed randomness to the protocol. Passive observers will have no idea if the coordinator actually included any honest participants' values in the published commitment. We can provide a notion of verifiability to purely passive observers by introducing a subset of *notarized participants* with some public reputation for honesty.

Notaries may be organizations such as nonprofits or government bodies who commit to participating in the protocol regularly. Each notary, after verifying its inclusion proof from the coordinator, signs the accumulator value. These signatures might be published by the coordinator or posted to the public bulletin board. To save space, they can be compressed using a succinct *multi-signature* scheme such as BLS [11], resulting in only  $O(1)$  additional overhead.

Observers can now verify the set of notarized participants who have contributed to the beacon output. As long as *one* of an observer's trusted notaries is honest and has signed the accumulator value, the final output  $\Omega$  can be trusted. In practice, using BLS multi-signatures, this would be about as efficient to verify as a threshold-signature-based protocol like *drand* [24], while offering much stronger security (any honest notarized participant vs. a majority of honest nodes in *drand*).

## 7.3 Incentivizing participation

Analyzing incentives in public randomness generation is an important open problem for DRBs in general. For Cornucopia specifically, we must incentivize coordinator(s) to provide a highly reliable service and expend non-trivial effort computing inclusion proofs. This is somewhat similar to incentivizing nodes to participate in an honest-majority DRB such as *drand*. In general, randomness beacons are a *public good* in that they are non-rivalrous (their value is not decreased as more users rely on them) and non-excludable (it is difficult to prevent anybody from utilizing them for their own purposes). Standard economic theory predicts that public goods are susceptible to free-riding: users may not want to contribute to funding a coordinator if they can rely on the efforts of others to do so and still utilize the randomness beacon. We hope that the relatively low costs of running a coordinator means it might attract corporate sponsorship for publicity, be run by a foundation, or receive government support.

Second, in Cornucopia we must incentivize users to regularly contribute randomness and to ensure their local machine is uncompromised and generating randomness correctly. The potentially large scale of Cornucopia instances might paradoxically decrease user motivation: if the protocol is secure as long as at least one other user is honest, why expend the effort to contribute at all? This is a version of the *bystander effect*, whereby opening participation to more parties which can contribute security means all of them may figure somebody else will do it. Hopefully, the open nature of Cornucopia may provide a new type of incentive, as by participating users themselves gain trust that the result is secure.

---

**References**

---

- 1 Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski, and Lukasz Mazurek. Secure Multiparty Computations on Bitcoin. In *IEEE Security & Privacy*, 2014.
- 2 Renas Bacho, Christoph Lenzen, Julian Loss, Simon Ochsenschreier, and Dimitrios Papachristoudis. GRandLine: First Adaptively Secure DKG and Randomness Beacon with (Almost) Quadratic Communication Complexity. *Cryptology ePrint Archive*, Paper 2023/1887, 2023.
- 3 Michael Ben-Or and Nathan Linial. Collective coin flipping, robust voting schemes and minima of banzhaf values. In *FOCS*, 1985.
- 4 Michael Ben-Or and Nathan Linial. Collective coin flipping. *Advances in Computing Research*, 1989.
- 5 Josh Benaloh and Michael De Mare. One-way accumulators: A decentralized alternative to digital signatures. In *Eurocrypt*, 1993.
- 6 Adithya Bhat, Aniket Kate, Kartik Nayak, and Nibesh Shrestha. OptRand: Optimistically responsive distributed random beacons. *Cryptology ePrint Archive*, Paper 2022/193, 2022.
- 7 Adithya Bhat, Nibesh Shrestha, Aniket Kate, and Kartik Nayak. RandPiper – Reconfiguration-Friendly Random Beacons with Quadratic Communication. *Cryptology ePrint Archive*, Paper 2020/1590, 2020.
- 8 Manuel Blum. Coin flipping by telephone a protocol for solving impossible problems. *ACM SIGACT News*, 1983.
- 9 Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable Delay Functions. In *CRYPTO*, 2018.
- 10 Dan Boneh, Benedikt Bünz, and Ben Fisch. A Survey of Two Verifiable Delay Functions. *Cryptology ePrint Archive*, Paper 2018/712, 2018.
- 11 Dan Boneh, Manu Drijvers, and Gregory Neven. Compact multi-signatures for smaller blockchains. In *Asiacrypt*, 2018.
- 12 Johannes Buchmann and Safuat Hamdy. A survey on IQ cryptography. In *Public-Key Cryptography and Computational Number Theory*, 2011.
- 13 Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *CRYPTO*, 2002.
- 14 Ignacio Cascudo and Bernardo David. Albatross: publicly attestable batched randomness based on secret sharing. In *Asiacrypt*, 2020.
- 15 Dario Catalano and Dario Fiore. Vector commitments and their applications. In *PKC*, 2013.
- 16 Megan Chen, Carmit Hazay, Yuval Ishai, Yuriy Kashnikov, Daniele Micciancio, Tarik Riviere, Abhi Shelat, Muthu Venkatasubramanian, and Ruihan Wang. Diogenes: Lightweight Scalable RSA Modulus Generation with a Dishonest Majority. In *IEEE Security & Privacy*, 2021.
- 17 Alisa Cherniaeva, Iliia Shirobokov, and Omer Shlomovits. Homomorphic encryption random beacon. *Cryptology ePrint Archive*, Paper 2019/1320, 2019.
- 18 Kevin Choi, Arasu Arun, Nirvan Tyagi, and Joseph Bonneau. Bicorn: An optimistically efficient distributed randomness beacon. In *Financial Crypto*, 2023.
- 19 Kevin Choi, Aathira Manoj, and Joseph Bonneau. Sok: Distributed randomness beacons. In *IEEE Security & Privacy*, 2023.
- 20 Miranda Christ, Kevin Choi, and Joseph Bonneau. Cornucopia: Distributed randomness beacons at scale. *Cryptology ePrint Archive*, 2023.
- 21 Richard Cleve. Limits on the security of coin flips when half the processors are faulty. In *TOC*, 1986.
- 22 Sourav Das, Vinith Krishnan, Irene Miriam Isaac, and Ling Ren. Spurt: Scalable distributed randomness beacon with transparent setup. In *IEEE Security & Privacy*, 2022.
- 23 Yevgeniy Dodis. Impossibility of black-box reduction from non-adaptively to adaptively secure coin-flipping. In *ECCE*, 2000.
- 24 Drand. <https://drand.love/>.
- 25 Dankrad Feist. RSA Assumptions. [rsa.cash/rsa-assumptions/](https://rsa.cash/rsa-assumptions/), 2022.

## 17:22 Cornucopia: Distributed Randomness at Scale

- 26 FileCoin. Trusted setup complete!, 2020. URL: <https://filecoin.io/blog/posts/trusted-setup-complete/>.
- 27 Ethereum Foundation. Proto-danksharding, 2023. URL: <https://www.eip4844.com/>.
- 28 Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In *CRYPTO*, 2018.
- 29 Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge. Cryptology ePrint Archive, Paper 2019/953, 2019.
- 30 David Galindo, Jia Liu, Mihai Ordean, and Jin-Mann Wong. Fully distributed verifiable random functions and their application to decentralised random beacons. In *Euro S&P*, 2021.
- 31 Shafi Goldwasser, Yael Tauman Kalai, and Sunoo Park. Adaptively secure coin-flipping, revisited. In *ICALP*, 2015.
- 32 Zhaozhong Guo, Liucheng Shi, and Maozhi Xu. SecRand: A Secure Distributed Randomness Generation Protocol With High Practicality and Scalability. *IEEE Access*, 2020.
- 33 Iftach Haitner and Yonatan Karidi-Heller. A tight lower bound on adaptively secure full-information coin flip. In *FOCS*, 2020.
- 34 Yael Tauman Kalai, Ilan Komargodski, and Ran Raz. A lower bound for adaptively-secure collective coin flipping protocols. *Combinatorica*, 41(1), 2021.
- 35 Alireza Kavousi, Zhipeng Wang, and Philipp Jovanovic. SoK: Public Randomness. Cryptology ePrint Archive, Paper 2023/1121, 2023.
- 36 Thomas Kerber, Aggelos Kiayias, and Markulf Kohlweiss. Mining for Privacy: How to Bootstrap a Snarky Blockchain. In *Financial Crypto*, 2021.
- 37 Hsun Lee, Yuming Hsu, Jing-Jie Wang, Hao Cheng Yang, Yu-Heng Chen, Yih-Chun Hu, and Hsu-Chun Hsiao. HeadStart: Efficiently Verifiable and Low-Latency Participatory Randomness Generation at Scale. In *NDSS*, 2022.
- 38 Arjen K. Lenstra and Benjamin Wesolowski. A random zoo: sloth, unicorn, and trx. Cryptology ePrint Archive, Paper 2015/366, 2015.
- 39 Jiangtao Li, Ninghui Li, and Rui Xue. Universal accumulators with efficient nonmembership proofs. In *ACNS*, 2007.
- 40 Helger Lipmaa. Secure accumulators from Euclidean rings without trusted setup. In *ACNS*, 2012.
- 41 Lipa Long. Binary quadratic forms. <https://github.com/Chia-Network/vdf-competition/blob/master/classgroups.pdf>, 2018.
- 42 Lan Nguyen. Accumulators from bilinear pairings and applications. In *CT-RSA*, 2005.
- 43 Valeria Nikolaenko, Sam Ragsdale, Joseph Bonneau, and Dan Boneh. Powers-of-tau to the people: Decentralizing setup ceremonies. In *ACNS*, 2024.
- 44 Charalampos Papamanthou. *Cryptography for efficiency: new directions in authenticated data structures*. PhD thesis, Brown University, 2011.
- 45 Krzysztof Pietrzak. Simple Verifiable Delay Functions. In *ITCS*, 2018.
- 46 Youcai Qian. Randao: Verifiable random number generation. [randao.org/whitepaper/Randao\\_v0.85\\_en.pdf](https://randao.org/whitepaper/Randao_v0.85_en.pdf), 2017.
- 47 Michael O. Rabin. Transaction protection by beacons. *Journal of Computer and System Sciences*, 1983.
- 48 Mayank Raikwar and Danilo Gligoroski. SoK: Decentralized randomness beacon protocols. In *Australasian Conference on Information Security and Privacy*, 2022.
- 49 Mark Ryan. Enhanced Certificate Transparency and End-to-End Encrypted Mail. In *NDSS*, 2014.
- 50 Philipp Schindler, Aljosha Judmayer, Markus Hittmeir, Nicholas Stifter, and Edgar Weippl. RandRunner: Distributed Randomness from Trapdoor VDFs with Strong Uniqueness. In *NDSS*, 2023.
- 51 Philipp Schindler, Aljosha Judmayer, Nicholas Stifter, and Edgar Weippl. Hydrand: Efficient continuous distributed randomness. In *IEEE Security & Privacy*, 2020.

- 52 Shравan Srinivasan, Alexander Chepurnoy, Charalampos Papamanthou, Alin Tomescu, and Yupeng Zhang. Hyperproofs: Aggregating and maintaining proofs in vector commitments. In *USENIX Security*, 2022.
- 53 Shравan Srinivasan, Ioanna Karantaidou, Foteini Baldimtsi, and Charalampos Papamanthou. Batching, aggregation, and zero-knowledge proofs in bilinear accumulators. In *ACM CCS*, 2022.
- 54 Ewa Syta, Philipp Jovanovic, Eleftherios Kogias, Nicolas Gailly, Linus Gasser, Ismail Khoffi, Michael J Fischer, and Bryan Ford. Scalable bias-resistant distributed randomness. In *IEEE Security & Privacy*, 2017.
- 55 Weijie Wang, Annie Ulichney, and Charalampos Papamanthou. {BalanceProofs}: Maintainable Vector Commitments with Fast Aggregation. In *USENIX Security*, 2023.
- 56 Benjamin Wesolowski. Efficient Verifiable Delay Functions. In *Eurocrypt*, 2019.
- 57 David Yakira, Avi Asayag, Ido Grayevsky, and Idit Keidar. Economically viable randomness. *CoRR*, 2020.





# Loss-Versus-Fair: Efficiency of Dutch Auctions on Blockchains

Ciamac C. Moallemi ✉ 🏠 

Graduate School of Business, Columbia University, New York, NY, USA

Dan Robinson ✉

Paradigm, San Francisco, CA, USA

---

## Abstract

---

Milionis et al. (2023) studied the rate at which automated market makers leak value to arbitrageurs when block times are discrete and follow a Poisson process, and where the risky asset price follows a geometric Brownian motion. We extend their model to analyze another popular mechanism in decentralized finance for onchain trading: Dutch auctions. We compute the expected losses that a seller incurs to arbitrageurs and expected time-to-fill for Dutch auctions as a function of starting price, volatility, decay rate, and average interblock time. We also extend the analysis to gradual Dutch auctions, a variation on Dutch auctions for selling tokens over time at a continuous rate. We use these models to explore the tradeoff between speed of execution and quality of execution, which could help inform practitioners in setting parameters for starting price and decay rate on Dutch auctions, or help platform designers determine performance parameters like block times.

**2012 ACM Subject Classification** Applied computing → Online auctions

**Keywords and phrases** Dutch auctions, blockchain, decentralized finance

**Digital Object Identifier** 10.4230/LIPIcs.AFT.2024.18

**Funding** *Ciamac C. Moallemi*: The first author is supported by the Briger Family Digital Finance Lab at Columbia Business School, and is an advisor to Paradigm and to fintech companies.

**Acknowledgements** The authors thank Eric Budish, Max Resnick, Anthony Zhang for helpful comments.

## 1 Introduction

Two of the most popular mechanisms for smart contracts to trade tokens are automated market makers (AMMs) – in which the price is determined by the contract’s reserves – and Dutch auctions – in which the price is determined by the current time.

When block times are discrete, both of these mechanisms leak some value to arbitrageurs. [13] studied the rate of this value leakage for AMMs, which is closely related to the concept of “loss-versus-rebalancing,” or LVR [14]. We apply a similar analysis to Dutch auctions, deriving a closed form for their “loss-versus-fair” (LVF) – the expected loss to the seller relative to selling their asset at its contemporaneous fair price – as well as their expected time-to-fill. We also do a similar analysis for gradual Dutch auctions, a variation on Dutch auctions that supports selling tokens at a constant rate over an extended period of time.

We hope this analysis can help inform practitioners in parameterizing these auctions (e.g., choosing the initial price and decay rate) to trade off execution quality with speed of execution, as well as helping spur research on how to design variants of Dutch auctions that are more resistant to LVF.

**Dutch auctions.** Also known as descending price auctions, Dutch auctions are auctions in which an item is listed at a high price that is gradually decreased over time until a bidder accepts.



© Ciamac C. Moallemi and Dan Robinson;

licensed under Creative Commons License CC-BY 4.0

6th Conference on Advances in Financial Technologies (AFT 2024).

Editors: Rainer Böhme and Lucianna Kiffer; Article No. 18; pp. 18:1–18:17

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 18:2 Loss-Versus-Fair: Efficiency of Dutch Auctions on Blockchains

Dutch auctions are a commonly-used mechanism in blockchain-based applications, thanks to their simplicity and efficiency in an environment with high transaction costs, limited privacy, and pseudonymous identities. Unlike ascending-price or sealed bid auctions, Dutch auctions typically require only one transaction after they start – the winning bid, with price as a function of the block number. This means that failed bidders typically do not need to pay transaction fees such as “gas” for their bids or leak any information about their intents.<sup>1</sup> Similarly, the seller needs only consider the single, winning bid, resulting in a significant reduction in communication and computation complexity versus other auction formats such as first- or second-price auctions. Dutch auctions are also strongly shill-proof [9]: the seller has no incentive to submit any fake or shill bids to change the auction outcome.

For these reasons, Dutch auctions have been used for a variety of applications in decentralized finance:

- Liquidations in peer-to-pool lending protocols like Maker [12] or Ajna [17]
- Rolling loans and discovering interest rates in peer-to-peer lending protocols like Blend [7]
- Routing trades in request-for-quote (RFQ) protocols like UniswapX [2] and 1inch Fusion [1]
- Collecting and converting fees in protocols like Euler [5]

Dutch auctions can be used both for price discovery of highly illiquid assets like NFTs, and for automated execution between liquid assets. Here, we focus on the latter case, and specifically on auctions between highly liquid but volatile pairs of tokens, such as between ETH and stablecoins like USDC. In particular, we assume a *common value* setting where all potential buyers agree on the value of the asset being sold at any point in time (because, for example, the asset may be liquidly traded in other off-chain markets), and assume the price of the asset obeys *geometric Brownian motion*.

**Gradual Dutch auctions.** Gradual Dutch auctions (GDAs) are a variation on Dutch auctions that were introduced by [6] as a mechanism for selling NFTs or tokens at a constant target rate over an extended time period. “Continuous gradual Dutch auctions” (CGDAs), the kind we consider in this paper, could be thought of as a series of infinitesimal Dutch auctions of a fungible token, with new auctions being initiated at a linear rate over time, and each auction independently decaying in price at an exponential rate.

**Arbitrage profits.** Dutch auctions have some drawbacks when implemented on a blockchain. In particular, since blocks only arrive at discrete times, the true market price of the asset at the time a block is created may be higher than the price offered by the auction, due to the decay of the Dutch auction price and the drift and volatility of the asset. This means the seller should expect to sell the asset at a discount to the market price or fair value at the time of sale, with the profits going to arbitrageurs or whoever is able to capture value from ordering transactions in the block – a type of maximal extractable value (MEV).

This type of loss is similar to the “quote-sniping” losses of market makers in high-frequency trading models [4], or the “loss-versus-rebalancing” suffered by liquidity providers on automated market makers, a concept introduced by [14]. In [13], LVR was extended to incorporate discrete blocks. For analytic tractability, block generation times are assumed to be from a Poisson process, an assumption we also make here.

---

<sup>1</sup> One exception is that if other bidders attempt to submit a bid at around the same time as the winning bid, their transaction may be publicized and/or included on chain after the winning transaction, possibly paying fees.

**Contributions.** In this paper, we apply a similar model to Dutch auctions and gradual Dutch auctions. Given certain parameters for a Dutch auction – volatility, drift, starting price, decay rate, and average block arrival times – we derive closed-form expressions for both the losses to fair value and expected time-to-fill. We also extend the analysis to gradual Dutch auctions, showing how expected losses to arbitrageurs and expected sales rate vary as a function of these parameters.

For both Dutch auctions and gradual Dutch auctions, as long as the auction starts above the current price, LVF is given by the following expression (where  $\delta$  is the decay rate of the auction plus the asset’s drift in log space,  $\sigma$  is the volatility of the asset, and  $\Delta t$  is the mean interblock time):

$$\text{LVF}_+ = \frac{1}{1 + \frac{\delta}{\sigma^2} \left( \sqrt{1 + \frac{2\sigma^2}{\delta^2 \Delta t}} - 1 \right)}.$$

For example, if volatility is 5% per day (0.017% per second), decay rate is 0.01% per second, and average block time is 12 seconds,  $\text{LVF}_+$  is about 0.13%. This would mean that for every \$100 worth of tokens that they sell, the seller should expect to get about \$99.87.

For regular Dutch auctions, the amount of time to fill if the starting price of the auction is higher than the current price is given by the following formula, in which  $z_0$  is the (log) difference between the starting price and the current price:

$$\text{FT}(z_0) = \frac{z_0}{\delta} + \frac{\Delta t}{2} \left( 1 + \sqrt{1 + \frac{2\sigma^2}{\delta^2 \Delta t}} \right).$$

For example, with the same parameters as above, and with starting price 0.1% higher than the current price, the expected time to fill is about 23.3 seconds. For gradual Dutch auctions, we find a closed form expression for the rate at which the asset is sold over time.

We also find closed forms for LVF and FT in the cases where the starting price of the auction is below the current price.

These models show how changing the decay rate of the auction affects both speed of execution and expected loss, helping inform practitioners who want to trade off between those values when choosing auction parameters such as initial price and decay rate. They also show how the characteristics of the blockchain – particularly average block time – affect the efficiency of Dutch auctions. For example, the formula for  $\text{LVR}_+$  above satisfies the lower bound

$$\text{LVF}_+ \geq \frac{1}{1 + \frac{1}{\sigma \sqrt{\Delta t/2}}} \approx \sigma \sqrt{\Delta t/2},$$

where the approximation holds for  $\Delta t$  small (the “fast block” regime). This suggests that if a platform wants to support Dutch auctions that lose less than 2 basis points for assets with daily volatility of 5%, it will need to have block times of less than 2.75 seconds.

## Literature Review

Dutch auctions have been analyzed extensively in the auction theory and mechanism design literature, since at least the work of [19], who showed the strategic equivalence of Dutch auctions and first-price sealed-bid auctions under certain assumptions.

Our approach is related to barrier-diffusion approaches [8] to limit order pricing. For example, in [11], the time-to-fill for a limit order is modelled as the first-passage time for a geometric Brownian motion with drift, and solve for the distribution of this time.

Mathematically, this is equivalent to a continuous time version of our model.<sup>2</sup> Crucially, they do not consider loss-versus-fair for a limit order, since this quantity would be zero in continuous time. [15] consider frictions introduced by latency in submitting limit orders, also using a barrier-diffusion model. In that setting, latency acts as a friction that limits the ability of an agent to update their limit orders in a timely fashion, in reaction to changing market conditions. The central novelty of the present paper is the blockchain setting: we analyze frictions restricting the ability to trade in the auction introduced by the discrete block generation process. To our knowledge, no prior work has modeled the behavior of Dutch auctions for geometric Brownian motion assets with discrete block generation times.

The idea of gradual Dutch auctions was proposed by [6]. [18] proposed an extension on the idea, variable rate GDAs, in which the target sales rate could vary as a function of time. Gradual Dutch auctions could be thought of as similar to automated market makers (AMMs) for which the price impact function is an exponential function, the fee to buy is 0, the fee to sell is infinite, and the asset price has a negative drift. In this way, we build on the setting of [13] in computing arbitrage profits for AMMs with fees.

A version of the GDA mechanism was studied by [10]. That work considers the use of discrete GDAs for illiquid NFTs where buyers depend on private signals for valuation, rather than continuous GDAs for highly liquid fungible tokens driven by common valuations.

## 2 Model

We imagine a scenario where an agent is selling<sup>3</sup> a risky asset via a descending price Dutch auction in a common value setting. Following the model of [13], we assume there exists the common fundamental value or price  $P_t$  at time  $t$  that follows a geometric Brownian motion price process,

$$\frac{dP_t}{P_t} = \mu dt + \sigma dW_t, \quad (1)$$

where  $\{W_t\}$  is a Brownian motion, and the process is parameterized by drift  $\mu$  and volatility  $\sigma > 0$ .

The agent is progressively willing to lower their offered price. Let  $A_t$  denote the lowest price the agent is willing to sell at, at time  $t$ , i.e., the best ask price. We assume  $A_t$  decreases exponentially according to<sup>4</sup>

$$\frac{dA_t}{A_t} = -\lambda dt, \quad (2)$$

<sup>2</sup> In our setting, the drift arises from descending price of the Dutch auction, while for [11], the limit order is at a static price and the drift arises from the underlying asset price process.

<sup>3</sup> While we focus on the case of an agent selling the asset via a Dutch auction, our model also applies to the case of an agent buying via an ascending price Dutch auction-style mechanism. In that case, the mechanism would have a steadily increasing bid price at which it is willing to buy the asset, and analogous formulas could be obtained. Note that over longer time horizons, the two cases are not completely symmetric because of the positivity of prices and the inherent asymmetry of geometric Brownian model. In particular, for example, for a seller LVF is bounded above by 100% since the sale price will always be bounded below by zero, while LVF is unbounded above for a buyer, since the buy price is unbounded above.

<sup>4</sup> The spirit here is to model a Dutch auction where the price is decreasing at a constant rate. We specifically choose exponentially decreasing prices (i.e., prices that are decreasing at a constant relative rate) because it matches well with the geometric Brownian motion price dynamics (1). An alternative choice would be to assume the ask price decreases linearly and that the price process is an arithmetic Brownian motion. On the short timescales of practical interest, this would yield similar results both quantitatively and qualitatively to the model here. To see this, note that, under our model,  $A_t = A_0 e^{-\lambda t} \approx A_0 (1 - \lambda t)$ , for  $t$  small.

with decay constant  $\lambda > 0$ . Define the log mispricing process  $z_t \triangleq \log(A_t/P_t)$ , so that, applying Itô's lemma,

$$dz_t = - \underbrace{\left(\lambda + \mu - \frac{1}{2}\sigma^2\right)}_{\triangleq \delta} dt + \sigma dW_t.$$

We assume that blocks are generated according to a Poisson process<sup>5</sup> of rate  $\Delta t^{-1}$ , where  $\Delta t > 0$  is the mean interblock time. We assume there is a population of “arbitrageurs”, or traders informed about the fundamental price  $P_t$ , who will buy from the agent at any discount to this price. However, these agents can only act at block generation times.

Thus, if  $\tau$  is a block generation time, and<sup>6</sup>  $A_{\tau-} < P_{\tau-}$ , arbs trade until there is no marginal profit, and so that the ask price updates with  $A_\tau = P_\tau$  and  $z_\tau = 0$ . Thus, we have  $z_\tau = \max(0, z_{\tau-})$ . Then,  $\{z_t\}$  is a Markov jump diffusion process. Since it involves the interaction of a diffusive process  $\{z_t\}$  with a barrier ( $z_t = 0$ ), our model falls into the general class of barrier-diffusion models for market microstructure [8].

This model is grounded in the typical way that blocks are built on decentralized blockchains today, in which each block is built by a “proposer” or “miner”. We imagine that the Dutch auction is implemented via a smart contract that sets the minimum acceptable price as a function of either block time or the block height. Within a block, the first transaction willing to pay the price will succeed. In the case that there are multiple buyers (as might be expected if the publicly observable fair value exceeds the limit price of the auction at the time), they would compete for earlier block position by offering priority fees to the proposer. We assume that each block proposer is independent and short-term profit-maximizing,<sup>7</sup> and hence they would include and prioritize the top priority-fee-paying transaction in the block, allowing that buyer to win the trade. In this case some or all of the arbitrage profits may accrue not to the buyer, but instead to the proposer in the form of priority fees. Our focus in this paper, however, is on quantifying the loss to the seller and not how it is distributed.

We will make the following assumption:

► **Assumption 1.** Assume that  $\delta \triangleq \lambda + \mu - \frac{1}{2}\sigma^2 > 0$ .

This assumption is sufficient to ensure that trade occurs with probability 1, and necessary so that the expected time to trade is finite. It can be satisfied by the agent making a sufficiently large choice of the decay rate  $\lambda$ . Under Assumption 1, the following lemma gives the stationary distribution  $\pi(z)$  of  $z_t$ :<sup>8</sup>

► **Lemma 2.** If  $\delta > 0$ , the process  $z_t$  is an ergodic process on  $\mathbb{R}$ , with unique invariant distribution  $\pi(\cdot)$  given by the density

$$p_\pi(z) = \begin{cases} \pi_+ \times p_{\zeta_+}^{\text{exp}}(z) & \text{if } z \geq 0, \\ \pi_- \times p_{\zeta_-}^{\text{exp}}(-z) & \text{if } z < 0, \end{cases}$$

<sup>5</sup> For a proof-of-work blockchain, Poisson block generation is a natural assumption [16]. However, modern proof-of-stake blockchains typically generate blocks at deterministic times. In these cases, we will view the Poisson assumption as an approximation that is necessary for tractability.

<sup>6</sup> We assume that the processes  $\{A_t, P_t\}$  are right continuous with left limits, and define  $A_{\tau-}$  and  $P_{\tau-}$  to be the left limits, i.e., the values immediately before the time  $\tau$ .

<sup>7</sup> Note that if this assumption is violated – such as if a single proposer controls multiple blocks in a row – they may be able to extract additional profit at the expense of the seller. As of this writing, extraction of this kind of “multi-block MEV” is generally believed to be rare on major blockchains like Ethereum, although there are reasons to be concerned that it could increase in the future.

<sup>8</sup> While applied in a different context, Lemma 2 is a special case of Theorem 7 of [13] up to a sign change, with  $\gamma_- \rightarrow \infty$ . For completeness, a standalone proof is provided in Appendix A.

## 18:6 Loss-Versus-Fair: Efficiency of Dutch Auctions on Blockchains

for  $z \in \mathbb{R}$ . Here,  $p_\zeta^{\text{exp}}(z) \triangleq \zeta e^{-\zeta z}$  is the density of an exponential distribution over  $z \in \mathbb{R}_+$  with parameter  $\zeta > 0$ . The parameters  $\{\zeta_\pm\}$  are given by

$$\zeta_- \triangleq \frac{\delta}{\sigma^2} \left( \sqrt{1 + \frac{2\sigma^2}{\delta^2 \Delta t}} - 1 \right), \quad \zeta_+ \triangleq \frac{2\delta}{\sigma^2}.$$

The probabilities  $\{\pi_\pm\}$  are given by

$$\pi_- \triangleq \pi((-\infty, 0)) = \delta \Delta t \zeta_-, \quad \pi_+ \triangleq \pi([0, +\infty)) = 1 - \delta \Delta t \zeta_-.$$

### 3 Regular Dutch Auctions

We first consider the case of a discrete quantity of the risk asset for sale, initially at ask price  $A_0$ , or, alternatively, initial log mispricing  $z_0 \triangleq \log(A_0/P_0)$ , with the ask price  $A_t$  decreasing exponentially at rate  $\lambda$  according to (2). Suppose the order is traded at fill time  $\tau_F$ , i.e.,  $\tau_F$  is the earliest block generation time which satisfies  $z_{\tau_F} \leq 0$ . Then, the order will sell at price  $A_{\tau_F}$  when the fundamental value is  $P_{\tau_F}$ . We are interested in the expected relative loss versus the fundamental price or fair value, i.e.,

$$\frac{P_{\tau_F} - A_{\tau_F}}{P_{\tau_F}} = 1 - e^{z_{\tau_F}}.$$

**Loss-versus-fair and time-to-fill.** We are interested in the expected relative loss, which we call “loss-versus-fair” (LVF), i.e.,

$$\text{LVF}(z_0) \triangleq \mathbb{E}[1 - e^{z_{\tau_F}} | z_0].$$

We are also interested in the expected time-to-fill, i.e.,

$$\text{FT}(z_0) \triangleq \mathbb{E}[\tau_F | z_0].$$

The following theorem characterizes these quantities:

► **Theorem 3.** *If  $z_0 \geq 0$ , the expected relative loss and time-to-fill are given by*

$$\text{LVF}(z_0) = \frac{1}{1 + \zeta_-} = \frac{1}{1 + \frac{\delta}{\sigma^2} \left( \sqrt{1 + \frac{2\sigma^2}{\delta^2 \Delta t}} - 1 \right)} \triangleq \text{LVF}_+, \quad (3)$$

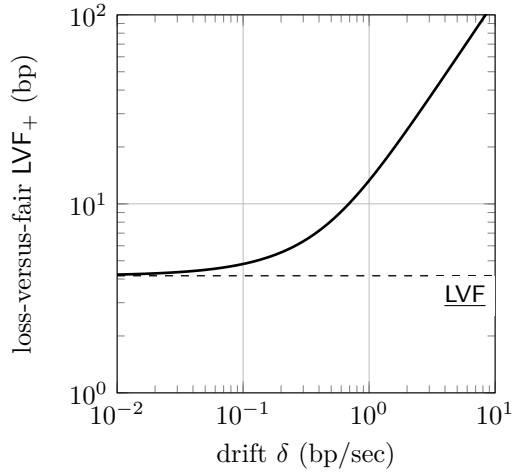
$$\text{FT}(z_0) = \frac{z_0}{\delta} + \frac{\Delta t}{2} \left( 1 + \sqrt{1 + \frac{2\sigma^2}{\delta^2 \Delta t}} \right). \quad (4)$$

If  $z_0 < 0$ , then

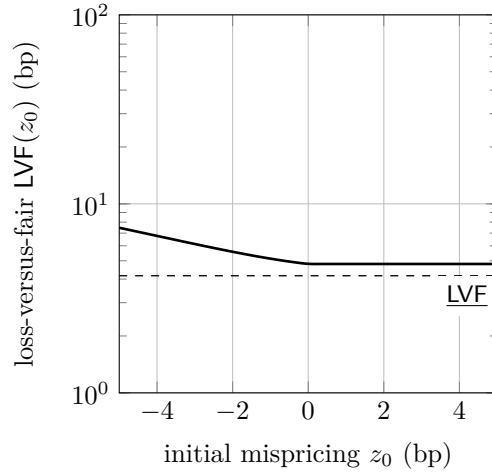
$$\text{LVF}(z_0) = 1 - \frac{e^{z_0}}{1 + \Delta t \left( \delta - \frac{1}{2}\sigma^2 \right)} + \left( \frac{1}{1 + \zeta_-} - \frac{\Delta t \left( \delta - \frac{1}{2}\sigma^2 \right)}{1 + \Delta t \left( \delta - \frac{1}{2}\sigma^2 \right)} \right) e^{\zeta_- z_0}, \quad (5)$$

$$\text{FT}(z_0) = \frac{\Delta t}{2} \left( 2 + \left( \sqrt{1 + \frac{2\sigma^2}{\delta^2 \Delta t}} - 1 \right) e^{\zeta_- z_0} \right). \quad (6)$$

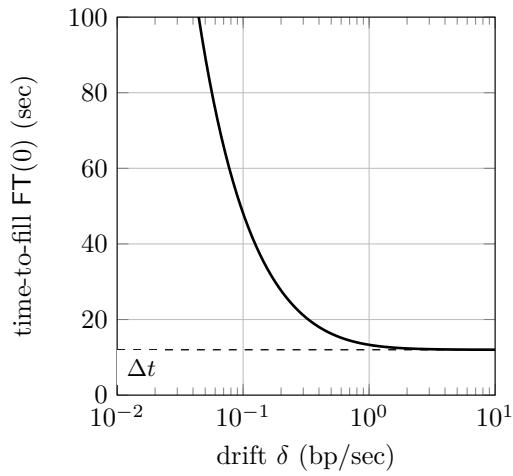
The formulas of Theorem 3 are illustrated for representative parameter choices in Figure 1.



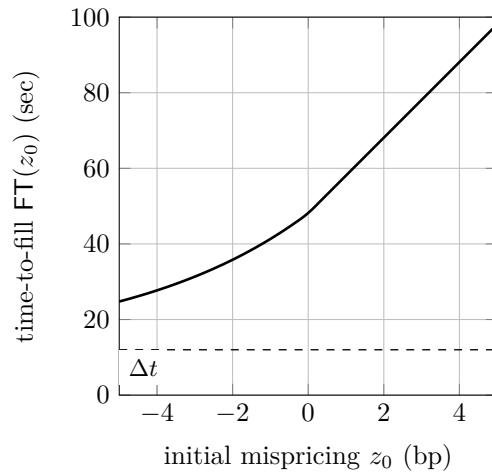
(a)  $LVF_+ = LVF(0)$  as a function of  $\delta$ .



(b)  $LVF(z_0)$  as a function of  $z_0$ , assuming a fixed value of  $\delta = 0.1$  (bp/sec).



(c)  $FT(0)$  as a function of  $\delta$ .



(d)  $FT(z_0)$  as a function of  $z_0$ , assuming a fixed value of  $\delta = 0.1$  (bp/sec).

■ **Figure 1** Comparison of LVF and FT for different parameter choices. These figures assume  $\sigma = 5\%$  (daily) and  $\Delta t = 12$  (sec). The dashed lines correspond to the lower bounds of (7) and (9).

**Discussion of loss-versus-fair.** Observe that, for  $z_0 \geq 0$ , the loss is given by  $\text{LVF}(z_0) = \text{LVF}_+$  and does not depend on the initial mispricing  $z_0$ . This is because, starting at  $z_0 \geq 0$ , the mispricing process must first pass through the boundary  $z_t = 0$ , since it is continuous. If we denote by  $\tau_0$  the first passage time of that boundary, because  $\{z_t\}$  is a Markov process and block generation times are memoryless, we have that  $\text{LVF}(z_0) = \mathbb{E}[\text{LVF}(z_{\tau_0})] = \text{LVF}(0) = \text{LVF}_+$ . For  $z_0 < 0$ ,  $\text{LVF}(z_0)$  is strictly decreasing in  $z_0$ . This is intuitive: the more the asset is initially underpriced, the larger the expected losses experienced upon the eventual sale.

Now, consider properties of the loss  $\text{LVF}_+$ . Observe that this is a strictly increasing function of the mispricing  $\delta$ , so that it is minimized when  $\delta = 0$ , and we have the lower bound

$$\underline{\text{LVF}} \triangleq \frac{1}{1 + \frac{1}{\sigma\sqrt{\Delta t/2}}} \leq \text{LVF}_+ \leq \text{LVF}(z_0). \quad (7)$$

In general, setting as small a value of the drift  $\delta$  as possible minimizes losses. However, the left side of (7) yields a lower bound on the loss that is due intrinsic volatility and discrete blocks. Indeed, in the fast block regime, when the average interblock time  $\Delta t$  is small, this lower bound takes the form

$$\underline{\text{LVF}} \triangleq \frac{1}{1 + \frac{1}{\sigma\sqrt{\Delta t/2}}} \approx \sigma\sqrt{\Delta t/2},$$

which is the standard deviation of changes in the mispricing process over half of a typical interblock time. This is a minimum, unavoidable level of loss, no matter what choice of auction parameters  $(z_0, \delta)$  is made.

We can also consider the behavior of  $\text{LVF}_+$  as a function of the volatility  $\sigma$ . It is straightforward to see that  $\text{LVF}_+$  is increasing in  $\sigma$ , and hence is lower bounded by the value as  $\sigma \rightarrow 0$ , i.e.,

$$\text{LVF}_+ \geq \lim_{\sigma \rightarrow 0} \text{LVF}_+ = \frac{1}{1 + \frac{1}{\delta\Delta t}} \approx \delta\Delta t, \quad (8)$$

where the final approximation holds in the fast block regime when  $\Delta t$  is small. The lower bound on the right side of (8) is the price decay over a single block. In the fast block regime, this is a lower bound on  $\text{LVF}_+$ , which also includes the impact of volatility.

**Discussion of time-to-fill.** For the time-to-fill, observe that  $\text{FT}(z_0)$  is a strictly increasing function of the initial mispricing  $z_0$  and a strictly decreasing function of the drift  $\delta$ , and that

$$\text{FT}(z_0) \geq \lim_{z \rightarrow -\infty} \text{FT}(z) = \Delta t. \quad (9)$$

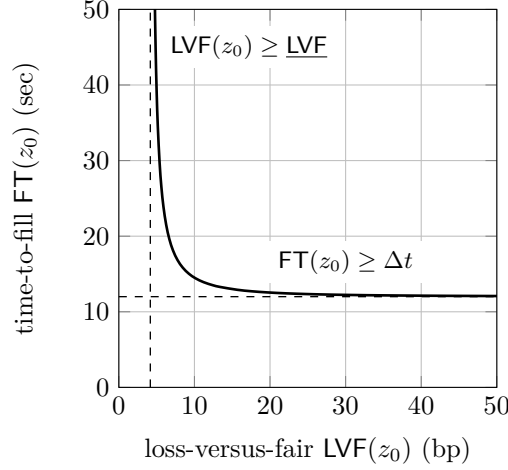
This lower bound is intuitive: by the memoryless nature of the Poisson process, the time-to-fill is always lower bounded by the mean interblock time.

$\text{FT}(z_0)$  is also increasing as a function of the volatility  $\sigma$ , hence we have the lower bound

$$\text{FT}(z_0) \geq \lim_{\sigma \rightarrow 0} \text{FT}(z_0) = \frac{z_0}{\delta} + \Delta t.$$

This bound is also intuitive: absent volatility, the ask price must first drift to the fair price (in time  $z_0/\delta$ ), and then wait for the next block (in time  $\Delta t$ ).





■ **Figure 2** The efficient frontier trading off loss-versus-fair and time-to-fill. This figure assumes  $\sigma = 5\%$  (daily) and  $\Delta t = 12$  (sec). The dashed lines correspond to the lower bounds of (7) and (9).

**Parameter optimization (known value).** Theorem 3 can be applied to optimize the initial auction price  $A_0$  at time  $t = 0$  and the decay rate  $\lambda$ . When the initial value  $P_0$  is known, we will parameterize this decision with the variables  $z_0 \triangleq \log(A_0/P_0)$  and  $\delta \triangleq \lambda + \mu - \frac{1}{2}\sigma^2 > 0$ . Then, the seller can solve the optimization problem

$$\underset{z_0, \delta \geq 0}{\text{minimize}} \quad \text{LVF}(z_0) + \theta \cdot \text{FT}(z_0).$$

Here,  $\theta \geq 0$  is a parameter that captures the trade off between minimizing loss and time-to-fill. The efficient frontier of Pareto optimal outcomes with these two objectives can be generated by varying  $\theta$ . An example of such an efficient frontier is illustrated in Figure 2.

Note that, in this setting, it is never optimal to pick  $z_0 > 0$ . This is because such a choice of  $z_0$  is Pareto dominated by setting  $z_0 = 0$ : in this case lowering the value of  $z_0$  strictly decreases  $\text{FT}(z_0)$ , without increasing  $\text{LVF}(z_0)$ . Indeed, with the representative parameter choices of Figure 2, setting  $z_0 \approx 0$  is typically optimal, i.e., the auction should be started at the current fundamental value (when it is known).

**Parameter optimization (unknown value).** Another setting of interest is where the buyer is uncertain of the value  $P_0$  when determining the auction parameters. We describe this uncertainty with a lognormal Bayesian prior: assume the seller believes that  $P_0 \sim \hat{P}_0 e^{-\frac{1}{2}\sigma_0^2 + \sigma_0 Z}$ , where  $Z \sim N(0, 1)$ ,  $\hat{P}_0 = \mathbb{E}[P_0]$  is the mean of the prior belief, and  $\sigma_0 > 0$  is the volatility of the prior belief. Then, we have  $z_0 = \log A_0/\hat{P}_0 + \frac{1}{2}\sigma_0^2 - \sigma_0 Z$ . Then, the seller can compute the loss-versus-fair and time-to-fill efficient frontier by solving the optimization problem

$$\underset{A_0, \delta \geq 0}{\text{minimize}} \quad \mathbb{E} \left[ \text{LVF} \left( \log A_0/\hat{P}_0 + \frac{1}{2}\sigma_0^2 - \sigma_0 Z \right) \right] + \theta \cdot \mathbb{E} \left[ \text{FT} \left( \log A_0/\hat{P}_0 + \frac{1}{2}\sigma_0^2 - \sigma_0 Z \right) \right],$$

for varying values of  $\theta \geq 0$ . Note that the expectations in the objective function can be computed in closed form, these formulas are provided in Appendix B.

## 4 Gradual Dutch Auctions

In this section, we develop stationary, steady-state analogs of the results of Section 3 in the context of gradual Dutch auctions. Introduced by [6], the continuous gradual Dutch auctions we consider here continuously emit the risky asset for sale at a rate per unit time given by  $r > 0$ . Each emission is in turn sold through a Dutch auction where the price decreases exponentially with decay rate  $\lambda > 0$ . Our goal will be to compute the steady-state rate at which such auctions leak value to arbitrageurs, as well as the rate of trade. We will see a similar tradeoff as in Section 3.

In our stationary, steady-state setting, we will imagine that the seller has been continuously emitting auctions since time  $t = -\infty$ . At any time  $t$ , if an auction has age  $u$ , the auction price is given by  $ke^{-\lambda u}$ , for some constant  $k > 0$ . When the age of the oldest available auction is  $T$ , this auction defines the best ask price by  $A_t = ke^{-\lambda T}$ . Hence, if an agent wishes to purchase a total quantity  $q$  at time  $t$ , and the age of the oldest available auction is  $T$ , the total cost is given by

$$C_t(q) = \int_{T-q/r}^T ke^{-\lambda u} \cdot r dt = \frac{kr}{\lambda} \frac{e^{\lambda q/r} - 1}{e^{\lambda T}} = A_t \cdot \frac{r}{\lambda} \left( e^{\lambda q/r} - 1 \right).$$

Denote the block generation times by  $0 < \tau_1 < \tau_2 < \dots$ . When a block is generated at each time  $t = \tau_i$ , arbitrageurs can trade against the auctions, and will myopically seek to do so to maximize their instantaneous profit, assuming they value the risky asset at the current fundamental price  $P_t$ . The following lemma characterizes this behavior:

► **Lemma 4.** *Suppose a block is generated at time  $\tau$ , with current fundamental price given by  $P \triangleq P_\tau$ , and mispricing (immediately before block generation) given by  $z \triangleq z_{\tau-}$ . Then, if  $\lambda > 0$ , the optimal arbitrage trade quantity of the risky asset is given by*

$$q^*(z) \triangleq -\frac{r}{\lambda} z \mathbb{I}_{\{z \leq 0\}},$$

with optimal arbitrage profits (or, equivalently, the total loss experienced by the auction seller relative to selling at the current fair fundamental price  $P$ )

$$A^*(P, z) \triangleq \frac{Pr}{\lambda} \{e^z - 1 - z\} \mathbb{I}_{\{z \leq 0\}}.$$

**Proof.** The arbitrageur faces the maximization problem

$$\underset{q \geq 0}{\text{maximize}} \quad P_\tau q - C_{\tau-}(q) = P \left\{ q - \frac{e^z r}{\lambda} \left( e^{\lambda q/r} - 1 \right) \right\},$$

where we use the fact that  $A_{\tau-} = P_\tau e^{z_{\tau-}}$ . The result follows from straightforward analysis of the first order and second order conditions for this optimization problem. Note that  $\lambda > 0$  is required for the second order conditions (concavity). ◀

Denote by  $N_T$  the total number of block generated over the time interval  $[0, T]$ . Suppose an arbitrageur arrives at time  $\tau_i$ , observing external price  $P_{\tau_i}$  and mispricing  $z_{\tau_i-}$ . From Lemma 4, the arbitrageur profit is given by  $A^*(P_{\tau_i}, z_{\tau_i-})$  and the trade size is given by  $q^*(z_{\tau_i-})$ . We can write the total arbitrage profit and total quantity traded (measured in the numéraire) paid over  $[0, T]$  by summing over all arbitrageurs arriving in that interval, i.e.,

$$\text{ARB}_T \triangleq \sum_{i=1}^{N_T} A^*(P_{\tau_i}, z_{\tau_i-}), \quad \text{VOL}_T \triangleq \sum_{i=1}^{N_T} P_{\tau_i} q^*(z_{\tau_i-}).$$

Clearly these are non-negative and monotonically increasing jump processes. The following theorem characterizes their instantaneous expected rate of growth or intensity:<sup>9</sup>

► **Theorem 5** (Rate of Arbitrage Profit and Volume). *Define the intensity, or instantaneous rate of arbitrage profit and volume, by*

$$\overline{\text{ARB}} \triangleq \lim_{T \rightarrow 0} \frac{\mathbb{E}[\text{ARB}_T]}{T}, \quad \overline{\text{VOL}} \triangleq \lim_{T \rightarrow 0} \frac{\mathbb{E}[\text{VOL}_T]}{T}.$$

*Given initial price  $P_0 = P$ , suppose that  $z_{0-} = z$  is distributed according to its stationary distribution  $\pi(\cdot)$ . Then, the instantaneous rate of arbitrage profit and volume are given by*

$$\overline{\text{ARB}} = \frac{\mathbb{E}_\pi[A^*(P, z)]}{\Delta t} = \frac{Pr\delta}{\delta - \mu + \frac{1}{2}\sigma^2} \times \frac{1}{1 + \zeta_-}, \quad (10)$$

$$\overline{\text{VOL}} = \frac{\mathbb{E}_\pi[Pq^*(z)]}{\Delta t} = \frac{Pr\delta}{\delta - \mu + \frac{1}{2}\sigma^2}. \quad (11)$$

Comparing the instantaneous rate of arbitrage profit and volume given by (10)–(11) with Theorem 3, we have that

$$\overline{\text{ARB}} = \overline{\text{VOL}} \times \text{LVF}_+. \quad (12)$$

This expression highlights the fact a gradual Dutch auction can be viewed as a continuum of many regular Dutch auctions, each of infinitesimal size, and each at a different price. From Theorem 3, we know that the seller will incur the same expected relative loss per dollar sold,  $\text{LVF}_+$ , in each of these auctions. This loss is the same irrespective of the different prices because all of the auctions start out-of-the-money ( $z_0 \geq 0$ ). Equation (12) intuitively decomposes the total arb profits per unit time as the product of the dollar volume sold per unit time and the loss per dollar sold.

**Parameter optimization.** As in Section 3, we can leverage Theorem 5 to optimize parameter choice in a gradual Dutch auction. In particular, a gradual Dutch auction is parameterized by the choice of emission rate  $r \geq 0$  and the choice of drift  $\delta \triangleq \lambda + \mu - \frac{1}{2}\sigma^2$  satisfying  $\delta \geq 0$  and  $\lambda = \delta - \mu + \frac{1}{2}\sigma^2 > 0$  (the second condition is required for concavity in Lemma 4). This choice can be made to minimize the losses incurred while maximizing the rate of trade. For example, consider the optimization problem

$$\underset{r > 0, \delta \geq \max(0, \mu - \frac{1}{2}\sigma^2)}{\text{minimize}} \quad \text{LVF}_+ - \theta \cdot \overline{\text{VOL}},$$

where  $\theta \geq 0$  is a tradeoff parameter.

## 5 Conclusion and Future Work

While there has been an increasing amount of academic interest in studying, designing, and formalizing automated market makers for liquid assets in the blockchain context, there has been somewhat less attention paid to Dutch auctions, despite their popularity with

<sup>9</sup> Mathematically,  $\overline{\text{ARB}}$  is the intensity of the compensator for the monotonically increasing jump process  $\text{ARB}_T$  at time  $T = 0$ , similarly  $\overline{\text{VOL}}$  is the intensity of the compensator for  $\text{VOL}_T$ .

protocol implementers. This paper was an attempt to bring the theoretical understanding of Dutch auctions in the setting of discrete block generation times closer to the current level of understanding that has been reached around automated market makers, particularly in [14] and [13].

The paper also sought to provide a guide for application designers in setting parameters for Dutch auctions, including deriving formulas that map the tradeoff between speed of execution and quality of execution. The paper may also be helpful for platform designers in determining performance parameters like block times. For example, the rule of thumb in Equation (7) suggests that if a platform wants to support Dutch auctions that lose less than 2 basis points for assets with daily volatility of 5%, it will need to have block times of less than 2.75 seconds.

The model in this paper shared some of the limitations of the model in [13], including not taking into account fixed transaction fees such as “gas” and use of a Poisson model for block generation as opposed to deterministic block generation, which is more relevant for modern proof-of-stake blockchains. Further, a purely diffusive, continuous process (geometric Brownian motion) has been used to model innovations in the fundamental price process, while jumps are known to be an important component of high-frequency price dynamics. This paper also assumes that block proposers are independent (and thus short-term profit-maximizing), rather than considering a model in which a single block proposer could acquire control over multiple blocks in a row and use that to extract “multi-block MEV.” Finally, while this work quantified the losses inherent in Dutch auctions, it does not explore possible alternative designs for Dutch auctions that might mitigate those losses without reducing the speed of execution. We hope further work can explore this area.

---

## References

- 1 1inch. Fusion, 2022. URL: <https://docs.1inch.io/docs/fusion-swap/introduction/>.
- 2 Hayden Adams, Noah Zinsmeister, Mark Toda, Emily Williams, Xin Wan, Matteo Leibowitz, Will Pote, Allen Lin, Eric Zhong, Zhiyuan Yang, Riley Campbell, Alex Karys, and Dan Robinson. Uniswapx, 2023. URL: <https://uniswap.org/whitepaper-uniswapx.pdf>.
- 3 Pierre Brémaud. *Markov chains: Gibbs fields, Monte Carlo simulation, and queues*, volume 31. Springer Science & Business Media, 2nd edition, 2020.
- 4 Eric Budish, Peter Cramton, and John Shim. The high-frequency trading arms race: Frequent batch auctions as a market design response. *The Quarterly Journal of Economics*, 130(4):1547–1621, 2015.
- 5 Euler. Fee flow: reverse dutch auctions for fees, 2024. URL: <https://docs.euler.finance/euler-v2-lite-paper/#fee-flow-reverse-dutch-auctions-for-fees>.
- 6 Frankie, Dan Robinson, Dave White, and andy8052. Gradual Dutch auctions. Technical report, Paradigm, 2022. URL: <https://www.paradigm.xyz/2022/04/gda>.
- 7 Galaga, Pacman, Toad, Dan Robinson, and Transmissions11. Blend: Perpetual lending with nft collateral, 2023. URL: <https://www.paradigm.xyz/2023/05/blend>.
- 8 Joel Hasbrouck. *Empirical market microstructure: The institutions, economics, and econometrics of securities trading*. Oxford University Press, 2007.
- 9 Andrew Komo, Scott Duke Kominers, and Tim Roughgarden. Shill-proof auctions. *arXiv preprint arXiv:2404.00475*, 2024.
- 10 Kshitij Kulkarni, Matheus V. X. Ferreira, and Tarun Chitra. Credibility and incentives in gradual dutch auctions, 2023. URL: <https://people.eecs.berkeley.edu/~ksk/files/GDA.pdf>.
- 11 Andrew W. Lo, A.Craig MacKinlay, and June Zhang. Econometric models of limit-order executions. *Journal of Financial Economics*, 65(1):31–71, 2002.

- 12 MakerDAO. Liquidation 2.0 module, 2022. URL: <https://docs.makerdao.com/smart-contract-modules/dog-and-clipper-detailed-documentation>.
- 13 Jason Milionis, Ciamac C Moallemi, and Tim Roughgarden. Automated market making and arbitrage profits in the presence of fees. *arXiv preprint arXiv:2305.14604*, 2023.
- 14 Jason Milionis, Ciamac C. Moallemi, Tim Roughgarden, and Anthony Lee Zhang. Automated market making and loss-versus-rebalancing, 2022. doi:10.48550/arXiv.2208.06046.
- 15 C. C. Moallemi and M. Sağlam. The cost of latency in high-frequency trading. *Operations Research*, 61(5):1070–1086, 2013.
- 16 Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Technical report, 2008.
- 17 Akash Patel, Ed Noepel, Gregory Di Prisco, Howard Malzberg, Ian Harvey, Joseph Quintilian, Matthew Cushman, and Michael Hathaway. Ajna protocol: Automated lending markets, 2023. URL: [https://www.ajna.finance/pdf/Ajna\\_Protocol\\_Whitepaper\\_01-11-2024.pdf](https://www.ajna.finance/pdf/Ajna_Protocol_Whitepaper_01-11-2024.pdf).
- 18 Transmissions11, Frankie, and Dave White. Variable rate GDAs. Technical report, Paradigm, 2022. URL: <https://www.paradigm.xyz/2022/08/vrgda>.
- 19 William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of Finance*, 16(1):8–37, 1961.

## A Proofs

**Proof of Lemma 2.** Note that  $\{z_t\}$  is a Markov jump diffusion process, with infinitesimal generator

$$\mathcal{A}f(z) = \frac{1}{2}\sigma^2 f''(z) - \delta f'(z) + \Delta t^{-1} [f(0) - f(z)] \mathbb{I}_{\{z < 0\}},$$

given a test function  $f: \mathbb{R} \rightarrow \mathbb{R}$ .

The invariant distribution  $\pi(\cdot)$  must satisfy

$$\mathbb{E}_\pi[\mathcal{A}f(z)] = \int_{-\infty}^{+\infty} \mathcal{A}f(z) \pi(dz) = 0, \quad (13)$$

for all test functions  $f: \mathbb{R} \rightarrow \mathbb{R}$ . We will guess that  $\pi(\cdot)$  decomposes according to two different densities over the positive and negative half line, and then compute the conditional density on each segment via Laplace transforms using (13).

Consider the test function

$$f_-(z) \triangleq \begin{cases} e^{\alpha z} & \text{if } z < 0, \\ 1 + \alpha z & \text{if } z \geq 0. \end{cases}$$

Then,

$$\mathcal{A}f_-(z) = \frac{1}{2}\sigma^2 \alpha^2 e^{\alpha z} \mathbb{I}_{\{z < 0\}} - \delta \alpha (e^{\alpha z} \mathbb{I}_{\{z < 0\}} + \mathbb{I}_{\{z \geq 0\}}) + \Delta t^{-1} [1 - e^{\alpha z}] \mathbb{I}_{\{z < 0\}},$$

so that

$$\begin{aligned} 0 &= \mathbb{E}_\pi[\mathcal{A}f_-(z)] \\ &= \frac{1}{2}\sigma^2 \alpha^2 \pi_- \mathbb{E}_\pi[e^{\alpha z} | z < 0] - \delta \alpha (\pi_- \mathbb{E}_\pi[e^{\alpha z} | z < 0] + \pi_+) + \Delta t^{-1} \pi_- (1 - \mathbb{E}_\pi[e^{\alpha z} | z < 0]). \end{aligned}$$

Then,

$$\mathbb{E}_\pi[e^{\alpha z} | z < 0] = \frac{\delta \alpha \frac{\pi_+}{\pi_-} - \Delta t^{-1}}{\frac{1}{2}\sigma^2 \alpha^2 - \delta \alpha - \Delta t^{-1}}.$$

## 18:14 Loss-Versus-Fair: Efficiency of Dutch Auctions on Blockchains

Observe the denominator has a single negative root. Then,  $\pi(-z|z < 0)$  must be exponential with parameter  $\zeta_- \triangleq (\sqrt{\delta^2 + 2\Delta t^{-1}\sigma^2} - \delta) / \sigma^2$ . Also, note that

$$\mathbf{E}_\pi[-z|z < 0] = 1/\zeta_-.$$

Next consider the test function

$$f_+(z) \triangleq \begin{cases} e^{-\alpha z} & \text{if } z \geq 0, \\ 1 - \alpha z & \text{if } z < 0. \end{cases}$$

Then,

$$\mathcal{A}f_+(z) = \frac{1}{2}\sigma^2\alpha^2 e^{-\alpha z}\mathbb{I}_{\{z \geq 0\}} + \delta\alpha (e^{-\alpha z}\mathbb{I}_{\{z \geq 0\}} + \mathbb{I}_{\{z < 0\}}) + \Delta t^{-1}\alpha z\mathbb{I}_{\{z < 0\}},$$

so that

$$\begin{aligned} 0 &= \mathbf{E}_\pi[\mathcal{A}f_+(z)] \\ &= \frac{1}{2}\sigma^2\alpha^2\pi_+\mathbf{E}_\pi[e^{-\alpha z}|z \geq 0] + \delta\alpha(\pi_+\mathbf{E}_\pi[e^{-\alpha z}|z \geq 0] + \pi_-) + \Delta t^{-1}\alpha\pi_-\mathbf{E}_\pi[z|z < 0]. \end{aligned}$$

Then,

$$\begin{aligned} \mathbf{E}_\pi[e^{-\alpha z}|z \geq 0] &= -\frac{\pi_-}{\pi_+} \frac{\delta + \Delta t^{-1}\mathbf{E}_\pi[z|z < 0]}{\frac{1}{2}\sigma^2\alpha + \delta} \\ &= -\frac{\pi_-}{\pi_+} \frac{\delta - \Delta t^{-1}/\zeta_-}{\frac{1}{2}\sigma^2\alpha + \delta} \end{aligned}$$

Then,  $\pi(z|z \geq 0)$  must be exponential with parameter  $\zeta_+ \triangleq 2\delta/\sigma^2$ . Substituting  $\alpha = 0$ , we have

$$1 = -\frac{\pi_-}{\pi_+} \frac{\delta - \Delta t^{-1}/\zeta_-}{\delta} = -\frac{\pi_-}{1 - \pi_-} \left(1 - \frac{1}{\delta\Delta t\zeta_-}\right).$$

Solving for  $\pi_-$ ,

$$\pi_- = \delta\Delta t\zeta_-, \quad \pi_+ = 1 - \delta\Delta t\zeta_-.$$

**Proof of Theorem 3.** We consider  $\text{LVF}(z_0)$  and  $\text{FT}(z_0)$  separately.

**Loss-versus-fair.** We begin with the LVF calculation. First, consider the case where  $z_0 \geq 0$ . Define the  $\tau_F$  to be the fill time of the order, i.e., the first Poisson block generation time  $\tau_F$  with  $z_{\tau_F} \leq 0$ . Also define  $\tau_0 \triangleq \min\{t \geq 0: z_t = 0\}$  to be first passage time for the boundary  $z_t = 0$ . Since the the process the mispricing process is continuous, we must have that  $\tau_F \geq \tau_0$ . Then,

$$\begin{aligned} \text{LVF}(z_0) &\triangleq \mathbf{E}[1 - e^{z_{\tau_F}} | z_0] \\ &\stackrel{(a)}{=} \mathbf{E}[\mathbf{E}[1 - e^{z_{\tau_F}} | \tau_0, z_{\tau_0}] | z_0] \\ &\stackrel{(b)}{=} \mathbf{E}[\text{LVF}(z_{\tau_0}) | z_0] \\ &\stackrel{(c)}{=} \text{LVF}(0) \triangleq \text{LVF}_+. \end{aligned} \tag{14}$$

where (a) follows from the tower property of expectation, (b) follows from the fact that Poisson arrivals are memoryless and  $\{z_t\}$  is a Markov process, and (c) follows from the fact that  $z_{\tau_0} = 0$ .

Now, consider arbitrary  $z_0 \in \mathbb{R}$ . Let  $\tau_B > 0$  be the first Poisson block generation time. Since  $\tau_F \geq \tau_B$ ,

$$\begin{aligned}
\text{LVF}(z_0) &\triangleq \mathbb{E}[1 - e^{z\tau_F} | z_0] \\
&\stackrel{(a)}{=} \mathbb{E}[\mathbb{E}[1 - e^{z\tau_F} | \tau_B, z_{\tau_B}] | z_0] \\
&\stackrel{(b)}{=} \mathbb{E}[\text{LVF}(z_{\tau_B}) | z_0] \\
&\stackrel{(c)}{=} \mathbb{E}\left[\text{LVF}_+ \mathbb{I}_{\{z_{\tau_B} \geq 0\}} + (1 - e^{z\tau_B}) \mathbb{I}_{\{z_{\tau_B} < 0\}} \mid z_0\right] \\
&\stackrel{(d)}{=} \int_0^\infty \frac{e^{-\tau/\Delta t}}{\Delta t} \int_0^{+\infty} \frac{1}{\sigma\sqrt{2\pi\tau}} e^{-\frac{1}{2}\left(\frac{z+\delta\tau-z_0}{\sigma\sqrt{\tau}}\right)^2} \text{LVF}_+ dz d\tau \\
&\quad + \int_0^\infty \frac{e^{-\tau/\Delta t}}{\Delta t} \int_{-\infty}^0 \frac{1}{\sigma\sqrt{2\pi\tau}} e^{-\frac{1}{2}\left(\frac{z+\delta\tau-z_0}{\sigma\sqrt{\tau}}\right)^2} (1 - e^z) dz d\tau.
\end{aligned} \tag{15}$$

where (a) follows from the tower property of expectation, (b) follows from the fact that Poisson arrivals are memoryless and  $\{z_t\}$  is a Markov process, (c) follows from the fact that  $\text{LVF}(z_{\tau_B}) = \text{LVF}_+$  for  $z_{\tau_B} \geq 0$  while  $\text{LVF}(z_{\tau_B}) = 1 - e^{z\tau_B}$  if  $z_{\tau_B} < 0$ , (d) follows from the fact that  $\tau_B$  is exponentially distributed while, conditional on  $\tau_B$ ,  $z_{\tau_B}$  is normally distributed, and  $\Phi(\cdot)$  is the cumulative normal distribution. Substituting in  $z_0 = 0$  and solving for  $\text{LVF}(z_0) = \text{LVF}_+$ , after integration, we obtain (3). For  $z_0 \leq 0$ , we can substitute (3) into (15) and integrate to obtain (5).

**Time-to-fill.** Suppose we start out at  $z_0 \geq 0$ , and define  $\text{FT}(z_0)$  to be the expected fill time of the next trade, i.e., the first Poisson arrival time  $\tau$  with  $z_{\tau_F} \leq 0$ . Also define  $\tau_0 = \min\{t \geq 0 : z_t = 0\}$  to be the first passage time for the boundary  $z_t = 0$ . Then, since the mispricing process  $\{z_t\}$  is continuous and Markov, and Poisson arrivals are memoryless, we have that  $\tau_F \geq \tau_0$  and

$$\text{FT}(z_0) = \mathbb{E}[\tau_F | z_0] = \mathbb{E}[\tau_0 | z_0] + \mathbb{E}[\mathbb{E}[\tau_F - \tau_0 | \tau_0, z_{\tau_0}] | z_0] = \frac{z_0}{\delta} + \text{FT}(0),$$

where we have used the standard formula for expected first passage time of a Brownian motion with drift.

Thus, we have reduced to the case where  $z_0 = 0$ . Define  $\tau_B > 0$  to be first Poisson block generation time. If  $z_{\tau_B} \leq 0$ , then  $\tau_B$  is also the fill time. On the other hand, if  $z_{\tau_B} > 0$ , we will have to wait an additional amount after  $\tau_B$  given in expectation by  $\text{FT}(z_{\tau_B}) = \text{FT}(0)$ . Thus, integrating over  $\tau_B$ , and using the fact that, given  $\tau_B$ ,  $z_{\tau_B}$  is normally distributed,

$$\begin{aligned}
\text{FT}(0) &= \int_0^\infty \frac{e^{-\tau/\Delta t}}{\Delta t} \left( \tau + \int_0^\infty \frac{1}{\sigma\sqrt{2\pi\tau}} e^{-\frac{1}{2}\left(\frac{z+\delta\tau}{\sigma\sqrt{\tau}}\right)^2} \text{FT}(z) dz \right) d\tau \\
&= \Delta t + \int_0^\infty \frac{e^{-\tau/\Delta t}}{\Delta t} \int_0^\infty \frac{1}{\sigma\sqrt{2\pi\tau}} e^{-\frac{1}{2}\left(\frac{z+\delta\tau}{\sigma\sqrt{\tau}}\right)^2} \left( \frac{z}{\delta} + \text{FT}(0) \right) dz d\tau \\
&= \Delta t + \int_0^\infty \frac{e^{-\tau/\Delta t}}{\Delta t} \left\{ \int_0^\infty \frac{1}{\sigma\sqrt{2\pi\tau}} e^{-\frac{1}{2}\left(\frac{z+\delta\tau}{\sigma\sqrt{\tau}}\right)^2} \frac{z}{\delta} dz + \text{FT}(0) \left( 1 - \Phi\left(\frac{\delta\sqrt{\tau}}{\sigma}\right) \right) \right\} d\tau \\
&= \Delta t + \int_0^\infty \frac{e^{-\tau/\Delta t}}{\Delta t} \left\{ \int_0^\infty \frac{1}{\sigma\sqrt{2\pi\tau}} e^{-\frac{1}{2}\left(\frac{z+\delta\tau}{\sigma\sqrt{\tau}}\right)^2} \frac{z}{\delta} dz + \text{FT}(0) \Phi\left(-\frac{\delta\sqrt{\tau}}{\sigma}\right) \right\} d\tau.
\end{aligned}$$

## 18:16 Loss-Versus-Fair: Efficiency of Dutch Auctions on Blockchains

We can solve this for  $\text{FT}(0)$ , i.e.,

$$\text{FT}(0) = \frac{\Delta t + \int_0^\infty \frac{e^{-\tau/\Delta t}}{\Delta t} \int_0^\infty \frac{1}{\sigma\sqrt{2\pi\tau}} e^{-\frac{1}{2}\left(\frac{z+\delta\tau}{\sigma\sqrt{\tau}}\right)^2} \frac{z}{\delta} dz d\tau}{1 - \int_0^\infty \frac{e^{-\tau/\Delta t}}{\Delta t} \Phi\left(-\frac{\delta\sqrt{\tau}}{\sigma}\right) d\tau} = \frac{\Delta t}{2} \left(1 + \sqrt{1 + \frac{2\sigma^2}{\delta^2\Delta t}}\right),$$

where the final equality is obtained via integration. This establishes (4).

Finally, consider the case where  $z_0 < 0$ . Define  $\tau_B > 0$  to be first Poisson block generation time. If  $z_{\tau_B} \leq 0$ , then  $\tau_B$  is also the fill time. On the other hand, if  $z_{\tau_B} > 0$ , we will have to wait an additional amount after  $\tau_B$  given in expectation by  $\text{FT}(z_{\tau_B})$ . Thus,

$$\begin{aligned} \text{FT}(z_0) &= \int_0^\infty \frac{e^{-\tau/\Delta t}}{\Delta t} \left( \tau + \int_0^\infty \frac{1}{\sigma\sqrt{2\pi\tau}} e^{-\frac{1}{2}\left(\frac{z+\delta\tau-z_0}{\sigma\sqrt{\tau}}\right)^2} \text{FT}(z) dz \right) d\tau \\ &= \Delta t + \int_0^\infty \frac{e^{-\tau/\Delta t}}{\Delta t} \int_0^\infty \frac{1}{\sigma\sqrt{2\pi\tau}} e^{-\frac{1}{2}\left(\frac{z+\delta\tau-z_0}{\sigma\sqrt{\tau}}\right)^2} \left( \frac{z}{\delta} + \text{FT}(0) \right) dz d\tau \\ &= \Delta t + \int_0^\infty \frac{e^{-\tau/\Delta t}}{\Delta t} \left\{ \int_0^\infty \frac{1}{\sigma\sqrt{2\pi\tau}} e^{-\frac{1}{2}\left(\frac{z+\delta\tau-z_0}{\sigma\sqrt{\tau}}\right)^2} \frac{z}{\delta} dz + \text{FT}(0) \Phi\left(-\frac{\delta\tau-z_0}{\sigma\sqrt{\tau}}\right) \right\} d\tau. \end{aligned}$$

After integration, this yields (6). ◀

**Proof of Theorem 5.** We follow the method of [13]. Specifically, using the smoothing formula, e.g., Theorem 13.5.7 of [3],

$$\mathbb{E}[\text{ARB}_T] = \mathbb{E}\left[\sum_{i=1}^{N_T} A^*(P_{\tau_i}, z_{\tau_i-})\right] = \mathbb{E}\left[\int_0^T A^*(P_t, z_{t-}) dN_t\right] = \mathbb{E}\left[\int_0^T A^*(P_t, z_{t-}) \cdot \Delta t^{-1} dt\right].$$

Applying Tonelli's theorem and the fundamental theorem of calculus,

$$\overline{\text{ARB}} \triangleq \lim_{T \rightarrow 0} \frac{\mathbb{E}[\text{ARB}_T]}{T} = \lim_{T \rightarrow 0} \frac{1}{T} \int_0^T \mathbb{E}\left[\frac{A^*(P_t, z_{t-})}{\Delta t}\right] dt = \frac{\mathbb{E}[A^*(P_0, z_{0-})]}{\Delta t} = \frac{\mathbb{E}_\pi[A^*(P, z)]}{\Delta t},$$

where in the final expression,  $P_0 = P$  and  $z$  is distributed according to the stationary distribution  $\pi(\cdot)$ .

Then, using the definition of  $\pi(\cdot)$  from Lemma 2 and  $A^*(\cdot, \cdot)$  from Lemma 4,

$$\begin{aligned} \overline{\text{ARB}} &= \frac{1}{\Delta t} \pi(z|z \leq 0) \frac{Pr}{\lambda} \mathbb{E}_\pi[e^z - 1 - z | z \leq 0] \\ &= \frac{Pr}{\lambda} \delta \zeta_- \left\{ \frac{\zeta_-}{1 + \zeta_-} - 1 + \frac{1}{\zeta_-} \right\} \\ &= \frac{Pr}{\lambda} \times \frac{\delta}{1 + \zeta_-}, \end{aligned}$$

as desired.

The same argument establishes that

$$\overline{\text{VOL}} \triangleq \lim_{T \rightarrow 0} \frac{\mathbb{E}[\text{VOL}_T]}{T} = \frac{\mathbb{E}_\pi[Pq^*(z)]}{\Delta t}.$$

Then, using Lemma 2 and  $q^*(\cdot)$  from Lemma 4,

$$\begin{aligned} \overline{\text{VOL}} &= \frac{1}{\Delta t} \pi(z|z \leq 0) \frac{Pr}{\lambda} \mathbb{E}_\pi[-z | z \leq 0] \\ &= \frac{Pr}{\lambda} \times \delta, \end{aligned}$$

as desired. ◀



## B Formulas Under Fundamental Value Uncertainty

Assume that the prior belief on the initial mispricing is normally distributed, i.e.,  $z_0 \sim N(\mu_0, \sigma_0^2)$ . Then, via direct integration,


$$\begin{aligned} E[\text{LVF}(z_0)] &= \text{LVF}_+ + (1 - \text{LVF}_+) \Phi\left(-\frac{\mu_0}{\sigma_0}\right) + \frac{\Delta t^{-1} e^{\frac{\sigma_0^2}{2}}}{\frac{\sigma^2}{2} - \delta - \Delta t^{-1}} e^{\mu_0} \Phi\left(-\sigma_0 - \frac{\mu_0}{\sigma_0}\right) \\ &+ \left\{ \left( \text{LVF}_+ - \frac{\frac{\sigma^2}{2} - \delta}{\frac{\sigma^2}{2} - \delta - \Delta t^{-1}} \right) e^{\frac{\Delta t^{-1} \sigma_0^2}{\sigma^2} + \frac{\delta}{\sigma^2} (\frac{\delta}{\sigma^2} \sigma_0^2 + \mu_0)} \left( \sqrt{1 + \frac{2\Delta t^{-1} \sigma^2}{\delta^2} + 1} \right) \right. \\ &\quad \left. \times \Phi\left(-\frac{\delta \sigma_0}{\sigma^2} \left( \sqrt{1 + \frac{2\Delta t^{-1} \sigma^2}{\delta^2} + 1} \right) - \frac{\mu_0}{\sigma_0} \right) \right\}. \end{aligned}$$

Similarly,




$$\begin{aligned} E[\text{FT}(z_0)] &= \Delta t + \frac{\sigma_0}{\delta \sqrt{2\pi}} e^{-\frac{\mu_0^2}{2\sigma_0^2}} \\ &+ \left( \text{FT}(0) - \Delta t + \frac{\mu_0}{\delta} \right) \Phi\left(\frac{\mu_0}{\sigma_0}\right) \\ &+ (\text{FT}(0) - \Delta t) e^{\frac{2\Delta t^{-1} \text{FT}(0)}{\sigma^2} \left( \frac{\sigma_0^2 \delta^2}{\sigma^4} + \frac{\mu_0 \delta}{\sigma^2} \right) + \frac{\Delta t^{-1} \sigma_0^2}{\sigma^2}} \Phi\left(-\frac{2\Delta t^{-1} \text{FT}(0) \sigma_0 \delta}{\sigma^2} - \frac{\mu_0}{\sigma_0}\right). \end{aligned}$$



# Credible, Optimal Auctions via Public Broadcast

Tarun Chitra  

Gauntlet, New York, NY, USA

Matheus V. X. Ferreira   

University of Virginia, Charlottesville, VA, USA

Kshitij Kulkarni  

UC Berkeley, CA, USA

---

## Abstract

We study auction design in a setting where agents can communicate over a censorship-resistant broadcast channel like the ones we can implement over a public blockchain. We seek to design credible, strategyproof auctions in a model that differs from the traditional mechanism design framework because communication is not centralized via the auctioneer. We prove this allows us to design a larger class of credible auctions where the auctioneer has no incentive to be strategic. Intuitively, a decentralized communication model weakens the auctioneer’s adversarial capabilities because they can only inject messages into the communication channel but not delete, delay, or modify the messages from legitimate buyers. Our main result is a separation in the following sense: we give the first instance of an auction that is credible only if communication is decentralized. Moreover, we construct the first two-round auction that is credible, strategyproof, and optimal when bidder valuations are  $\alpha$ -strongly regular, for  $\alpha > 0$ . Our result relies on mild assumptions – namely, the existence of a broadcast channel and cryptographic commitments.

**2012 ACM Subject Classification** Applied computing → Online auctions; Security and privacy → Formal security models; Theory of computation → Algorithmic mechanism design

**Keywords and phrases** credible auctions, blockchains, cryptographic auctions, optimal auction design, mechanism design with imperfect commitment

**Digital Object Identifier** 10.4230/LIPIcs.AFT.2024.19

**Related Version** *Full Version:* <https://arxiv.org/pdf/2301.12532> [3]

**Funding** *Matheus V. X. Ferreira:* Research was performed while author was at Harvard University.

## 1 Introduction

Incentive compatibility for buyers is desirable in auctions due to improvements in user experience. For example, in a second-price auction, if the highest bidder bids \$10 and the second highest bidder bids \$5, the highest bidder wins and pays \$5. Thus, for any buyer, bidding the maximum they are willing to pay is an optimal strategy, independently of the strategy of others. This differs from non-incentive compatible auctions, such as first-price auctions, where optimal strategies are a complex balance between demand and the strategy of competing buyers.

Extending incentive compatibility to auctioneers is increasingly becoming a topic of interest in designing auctions within digital marketplaces. In online settings, it is challenging to audit auctions and verify the identity of participants. Thus, a strategic auctioneer can act simultaneously as the seller and a buyer to deviate from the promised auction. For example, in the second-price auction above, buyers must trust the auctioneer can commit to implementing the promised auction. Otherwise, a strategic auctioneer impersonating a buyer can easily leverage their privileged position to submit a bid of \$9, increasing revenue and reducing the buyer’s welfare.



© Tarun Chitra, Matheus V. X. Ferreira, and Kshitij Kulkarni;  
licensed under Creative Commons License CC-BY 4.0  
6th Conference on Advances in Financial Technologies (AFT 2024).

Editors: Rainer Böhme and Lucianna Kiffer; Article No. 19; pp. 19:1–19:16

Leibniz International Proceedings in Informatics



LIPIC Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

*Credibility* is a form of incentive compatibility for auctioneers that formalizes the incentives for an auctioneer to follow their promised specifications. It is desirable for auctioning objects ranging from Non-Fungible Tokens (NFTs) to online advertising because it ensures auction outcomes are auditable. The US Department of Justice’s 2023 antitrust suit against Google [17] effectively argues that Google’s manipulation of ad auctions from the privileged position of auctioneer caused both buyers and users harm. Allegedly, the lack of market transparency afforded Google “power to manipulate the quantity of ad inventory and auction dynamics in ways that allowed it to charge advertisers more than it could in a competitive market”. Thus, credibility is not only a compelling objective for regulators, but also for sellers that wish to prove their auctions are fair.

Unfortunately, recent work has highlighted challenges in designing mechanisms that are simultaneously incentive-compatible for both sellers and buyers. The pioneering work of [1] considered a model where the auctioneer can modulate their private communication with buyers to increase their revenue and potentially reduce buyer welfare via a safe deviation from a promised auction. Informally, a *safe deviation* is any auctioneer deviation that passes undetectable by any buyer alone. An auction is *credible* if safe deviations cannot increase the auctioneer’s expected revenue. For example, an auctioneer waiting for the highest bidder to bid \$10 and impersonating a false buyer that bids \$9 is a profitable, safe deviation from a second-price auction. Unfortunately, [1] demonstrated that auctioneer credibility could not coexist with buyer incentive compatibility unless the communication complexity is unbounded: they showed that an ascending price auction is the only credible, strategyproof optimal auction.

On the other hand, if one is willing to assume that the auctioneer and buyers are computationally bounded – and thus cannot break known cryptographic assumptions – one can get around the theoretical barriers of [1]. Concretely, [6] demonstrated that there are cryptographic auctions that are credible, incentive compatible, and have bounded communication complexity if buyer valuations satisfy a regularity condition. They proposed the (centralized) *Deferred Revelation Auctions (DRA)*, a two-round auction that is optimal, strategyproof, and credible under the assumption buyer valuations are  $\alpha$ -strongly regular for any  $\alpha \geq 1$ . They also show their auction is not always credible if  $\alpha < 1$  and valuations have unbounded support. This challenges adopting these auctions because they are only credible if the buyer valuations have tails not heavier than the exponential distribution, i.e.,  $\alpha$ -strongly regular for  $\alpha \geq 1$  does not contain the Pareto distribution, for example.

In the same line as [6], [4] proposed the Ascending Deferred Revelation Auction (ADRA), which is strategyproof and credible without requiring any assumption on the distributions. However, ADRA communication complexity is constant on expectation and unbounded in the worst case. In contrast, we study auctions with bounded communication in the worst case.

All results above consider a centralized communication model where buyers can only exchange messages with the auctioneer. This assumption is motivated by the scenario where one buyer does not have prior knowledge of the identity of a second buyer. Unfortunately, if the communication is centralized, the auctioneer can launch a man-in-the-middle-like attack by censoring, injecting, and modifying messages they were supposed to forward to other participants. In contrast, our work explores the design of credible auctions when agents can access a broadcast channel where any buyer can broadcast messages to all participants. This assumption is well-motivated in an auction in a physical environment like a traditional auction house. Further, this assumption has also become realistic for auctions implemented over a communication network like the Internet due to the proliferation of censorship-resistant public blockchains. Our main contribution shows a simple change in the communication model (centralized vs. distributed) affects the design of credible auctions.

We summarize our findings as follows:

- **Theorem 21.** Assume buyer values are drawn independently from an  $\alpha$ -strongly regular distribution for  $\alpha > 0$ . Then, the deferred revelation auction with public broadcast is credible, strategyproof, and revenue optimal. Moreover, modifying the auction so buyers can only communicate privately with the auctioneer makes the resulting auction not credible.
- **Theorem 25.** There is a 0-strongly regular buyer valuation that witnesses that deferred revelation auction with public broadcast is not credible.

## 1.1 Related Work

We have already reviewed the most relevant prior work in our earlier discussion. Our model is similar to [1] under the additional assumption of cryptographic primitives plus a public broadcast channel.

The security of auctions using cryptography has been extensively studied in the literature [15, 2]. Notably, Yao’s seminal work on multi-party computation [19] was initially motivated by economic applications. Recent research has revisited the problem of secure auction design, incorporating novel cryptographic tools such as homomorphic encryption and timed encryption techniques [16] [9].

However, these approaches come with stronger trust assumptions. For instance, multi-party computation assumes that a majority of participants are honest. In contrast, our setting allows the auctioneer to create multiple identities. Furthermore, timed encryption, although an intriguing concept, has seen limited practical applications due to its reliance on stronger cryptographic assumptions. Importantly, our goal of reducing the number of auction rounds aims to enhance auction speed, whereas timed encryption would counter this objective by increasing the auction duration.

Credible mechanism design has applications beyond auctions such as in the design of manipulation-resistant decentralized exchanges [18], blockchain transaction fee mechanisms [8, 5], and in Bayesian persuasion [12].

## 1.2 Technical overview

[1] does not consider the existence of a broadcast channel in their framework because they envision auctions executing over the Internet (or over the telephone) and assume buyers do not know the identity of each other beforehand. Implementing a broadcast channel in this scenario is challenging and draws from years of research in consensus and cryptography, starting from the Byzantine general’s problem of [11]. This line of research culminated with the Bitcoin blockchain, which provides censorship-resistant consensus at large scale [14]. In the framework of [1], the auctioneer promises to implement an auction and is the nexus of communication with buyers. A buyer privately sends messages to the auctioneer and trusts that the auctioneer will forward those messages to other buyers.

We propose a simple modification to this framework that, surprisingly, increases the incentive for the auctioneer to commit to a promised auction. Concretely, rather than sending messages privately to the auctioneer, we assume any agent can broadcast messages. Once an agent broadcasts a message  $m$ , all other participants simultaneously learn about message  $m$ .

Under the new framework, our main contribution is the *deferred revelation auction with public broadcast*. It is similar to the centralized deferred revelation auction of [6] with the main difference that buyers can now broadcast messages. Our main result shows *DRA* with public broadcast is a credible auction, assuming buyer valuations are  $\alpha$ -strongly regular for

## 19:4 Credible, Optimal Auctions via Public Broadcast

all  $\alpha > 0$ . Recall [6] showed centralized *DRA* is not credible for these buyer valuations. This has significant practical implications because it provides the first design of a communication-efficient, credible, strategyproof, optimal auction when buyer value distributions have tails as heavy as Pareto distributions.

Informally, the deferred revelation auction with public broadcast is a two-phase auction (see §3) as follows:

1. In the *bidding phase*, each buyer broadcasts a cryptographic commitment of their bid and deposits a collateral.
2. The auctioneer broadcasts the end of the commitment phase and the start of the revelation phase.
3. In the *revelation phase*, each buyer broadcasts the opening of their commitment (e.g., their bid and the random seed used to generate the commitment).
4. The auctioneer marks a bid as revealed if the second phase message opens the cryptographic commitment received in the first phase. Then, the auctioneer implements the second-price auction with reserves using the revealed bids.
5. The auctioneer refunds the collateral if, and only if, a buyer reveals their bid. The confiscated collateral is given to the winner of the auction.

As in [6], we consider a threat model where the auctioneer can shill bid (i.e., impersonate false buyers that submit false bids). To argue the credibility of our auction, we must show that under certain conditions, sufficiently large collateral incentivizes the auctioneer not to impersonate false buyers. Central to our argument is observing that the security of our cryptographic commitment scheme (see Definition 5) together with a broadcast channel ensures the auctioneer cannot commit to a bid that depends on the bids of other buyers. This is not the case for the centralized deferred revelation auction. To see, consider modifying the auction above so that whenever a buyer broadcasts a message, the buyer sends that message to the auctioneer, who “promises” to forward it to all other buyers. The following is a safe deviation to centralized *DRA* where shill bids depends on bids from genuine buyers.

► **Example 1.** Suppose there are genuine buyers  $A$  and  $B$  as well as a false buyer  $C$ . Any message the auctioneer receives from  $B$ , the auctioneer forwards to  $A$ . The auctioneer does not forward any message from  $A$  to  $B$  which makes buyer  $B$  unaware that  $A$  exists. The auctioneer asks buyer  $A$  to open their bid and after learning the bid  $b_A$  of  $A$ , the auctioneer impersonates a false buyer  $C$  that commits to bid  $b_A + \Delta$  to buyer  $B$ . This deviation is undetectable because buyer  $A$  believes only  $A$  and  $B$  participate in the auction. Moreover,  $A$  cannot detect their messages were censored. On the other hand,  $B$  believes only  $B$  and  $C$  participate in the auction. Moreover,  $B$  cannot detect that  $A$ ’s messages were censored (in fact,  $B$  is unaware of  $A$ ). Finally, observe that  $B$  receives a bid from a false buyer correlated with the bid of  $A$ .

This might seem like an innocent deviation, but Section 5 shows centralized *DRA* is not credible for  $\alpha$ -strongly regular valuations for any  $\alpha \in [0, 1)$  if we adapt this strategy and allow the auctioneer to submit shill bids that depend on genuine bids. Clearly this deviation is not possible if a broadcast channel is available since the auctioneer cannot choose who gets to observe  $A$ ’s messages and the auctioneer cannot commit to shill bids after starting the revelation phase. Our main technical contribution shows that safe deviations that leverage shill bids correlated to genuine bids were the only strategies that prevented centralized *DRA* from being a credible auction when buyer valuations are  $\alpha$ -strongly regular for  $\alpha > 0$ .

### 1.3 Paper organization

We provide the necessary background in optimal auction theory in §2. We define the implementation of the deferred revelation auction with public broadcasts in §3. We prove our main result, Theorem 21, in §4 and our negative result, Theorem 25, in §6. §5 shows that a broadcast channel is necessary for Theorem 21. We conclude in §7 and include future directions.

## 2 Preliminaries

We consider a single item,  $n$  buyer auction. Buyer  $i \in [n] = \{1, \dots, n\}$  has private value  $v_i \in \mathbb{R}^+$  and has quasilinear utility: if they receive the item and pay  $p_i$ , then their utility is  $v_i - p_i$ ; if they do not receive the item, then their utility is 0. We assume  $v_i$  is drawn independently from a distribution  $D$  with CDF  $F$  and PDF  $f$ . The auctioneer knows the distribution  $D$ , but not the values  $\{v_i\}_{i=1}^n$ . We refer to  $\vec{v} = (v_1, \dots, v_n)$  as a *value profile*. We write the value profile of all buyers except buyer  $i$  as  $\vec{v}_{-i} = (v_1, \dots, v_{i-1}, v_{i+1}, \dots, v_n)$ .

**Communication model.** Agents can communicate on a *private channel* or a *broadcast channel*. If agent  $i$  sends a message  $m$  in a broadcast channel, then the message is immediately received by all other agents. If agent  $i$  sends a message  $m$  to agent  $j \neq i$  in a private channel, only agent  $j$  observes  $m$ .

**Extensive-form game.** An extensive-form game  $G$  consists of a tree  $(H, E)$  where the nodes  $H$  are the set of histories and edges  $E \subseteq H \times H$  are state transitions. The game starts at the root of  $(H, E)$ , has a set of players  $\{0, 1, \dots, n\}$ , and a collection of actions  $A(h)$  available at each history  $h \in H$ . We refer to player 0 as the auctioneer and player  $i \in [n]$  as buyer  $i$ . Each history  $h \in H$  has one owner  $P(h) \in \{0, \dots, n\}$  responsible for taking the next action when the game is at state  $h$ . After taking action  $a \in A(h)$ , the game moves to another history  $h'$  where  $(h, h') \in E$ . We consider games of incomplete information where only agent  $P(h)$  observes the action  $A(h)$  taken at  $h$ .

A strategy  $s_i$  for buyer  $i \in [n]$  on game  $G$  is a function that takes buyer  $i$ 's private type  $v_i$  and any history  $h \in H$  where  $i \in P(h)$  and outputs the agent's action  $s_i(v_i, h) \in A(h)$  at  $h$ . Consider a strategy profile  $\vec{s} = (s_1, \dots, s_n)$ . An *auction game*  $(G, \vec{s})$  is a communication game on  $G$  when buyers follow strategy  $\vec{s}$  that allocates the item and charges payments.

The outcome of auction game  $(G, \vec{s})$  is a tuple  $(\vec{x}^{(G, \vec{s})}(\vec{v}), \vec{p}^{(G, \vec{s})}(\vec{v}))$  where  $x_i^{(G, \vec{s})}(v)$  and  $p_i^{(G, \vec{s})}(\vec{v})$  denotes the probability that agent  $i$  receives the item and their payment respectively. A strategy  $s_i$  is a best response to  $\vec{s}_{-i}$  if for any strategy  $s'_i$  for buyer  $i$ , for any  $\vec{v}$ ,

$$v_i \cdot x_i^{(G, \vec{s})}(\vec{v}) - p_i^{(G, \vec{s})}(\vec{v}) \geq v_i \cdot x_i^{(G, s'_i, \vec{s}_{-i})}(\vec{v}) - p_i^{(G, s'_i, \vec{s}_{-i})}(\vec{v}).$$

► **Definition 2** (Ex-post Nash/Strategyproof/Individually Rational). *Consider an auction  $(G, \vec{s})$ . A strategy profile  $\vec{s}$  forms an ex-post Nash equilibrium, if for any buyer  $i$ , strategy  $s_i$  is the best response to  $\vec{s}_{-i}$ . An auction is strategyproof if some strategy profile  $\vec{s}$  forms an ex-post Nash equilibrium. An auction is individually rational (IR) if there is a strategy for any buyer that ensures such buyer receives non-negative utility.*

The auctioneer's *expected revenue* on auction game  $(G, \vec{s})$  is  $\text{REV}(G, \vec{s}) := \mathbb{E}_{\vec{v}} \left[ \sum_{i=1}^n p_i^{(G, \vec{s})}(\vec{v}) \right]$ .

## 19:6 Credible, Optimal Auctions via Public Broadcast

We assume the auctioneer can deviate from implementing  $(G, \vec{s})$  as long as any buyer cannot detect a deviation. These are a *safe deviation* from the promised auction  $(G, \vec{s})$ . Formally,  $(G', s)$  is a *safe deviation* from  $(G, \vec{s})$  if for any buyer  $i \in [n]$ , there is a strategy profile  $s_{-i}^i = (s_1^i, \dots, s_{i-1}^i, s_{i+1}^i, \dots, s_{n_i}^i)$  for  $n_i$  buyers where buyer  $i$  observes the same messages in communication games  $(G', \vec{s})$  and  $(G, s_i, s_{-i}^i)$ .

► **Definition 3** (Credible Auction). *An auction game  $(G, \vec{s})$  is credible if for any safe deviation  $(G', \vec{s})$  of  $(G, \vec{s})$ ,  $REV(G, \vec{s}) \geq REV(G', \vec{s})$ .*

### Virtual values

Virtual values functions allow us to formalize optimal auctions. The *virtual value function* associated with continuous CDF  $F$  and PDF  $f$  is  $\varphi^F(x) = x - \frac{1-F(x)}{f(x)}$ . We write  $\varphi(\cdot)$ , omitting the superscript  $F$ , when the distribution is clear from the context. We write  $\varphi^+(x) = \max\{0, \varphi(x)\}$ . A distribution  $F$  is  $\alpha$ -strongly regular for  $\alpha \geq 0$  if for all  $x' \geq x$ ,

$$\varphi(x') - \varphi(x) \geq \alpha(x' - x).$$

A distribution  $F$  has *Monotone Hazard Rate (MHR)*, if  $F$  is 1-strongly regular. A distribution is *regular* if  $F$  is 0-strongly regular. Note that MHR distributions have exponentially decaying tails, whereas distributions with  $\alpha \in (0, 1)$  have polynomially decaying tails.

► **Theorem 4** (Myerson's Theorem [13]). *Consider a strategyproof auction that awards the item to buyer  $i$  with probability  $x_i(\vec{v})$  and charges  $p_i(\vec{v})$  on bids  $\vec{v}$ . Then, the expected revenue is*

$$\mathbb{E}_{\vec{v} \leftarrow D} \left[ \sum_{i=1}^n p_i(\vec{v}) \right] = \mathbb{E}_{\vec{v} \leftarrow D} \left[ \sum_{i=1}^n \varphi(v_i) \cdot x_i(\vec{v}) \right].$$

We refer to the right-hand side as the *expected virtual welfare*. For cases where  $D$  is regular,  $\varphi$  is non-decreasing, and the optimal auction maximizes expected virtual welfare.

We define the inverse of a monotone function  $g(\cdot)$  to be  $g^{-1}(y) = \inf_x \{x \mid g(x) \geq y\}$ . We define to  $r(D) := (\varphi^D)^{-1}(0)$  as *Myerson's reserve price*. From Myerson's theorem, the optimal auction only sells the item to buyers with the value  $v_i \geq r(D)$ . We define  $REV(D^n)$  as the expected revenue of the optimal auction with  $n$  buyers with valuations drawn i.i.d. from  $D$ . We provide facts about  $\alpha$ -strongly regular distributions in Appendix A.

## 3 Deferred Revelation Auction (DRA) with Public Broadcast

This section defines the deferred revelation auction with public broadcast. The central assumption is the existence of a perfectly hiding, computationally binding, and non-malleable cryptographic commitment scheme as follows.

► **Definition 5** (Commitment Scheme). *A commitment scheme is a function  $COMMIT(\cdot, \cdot)$  that takes a message  $m \in \{0, 1\}^*$ , a random string  $r \in \{0, 1\}^\lambda$  where  $\lambda \in \mathbb{N}$  is the security parameter and outputs a commitment  $c \in \{0, 1\}^{POLY(\lambda)}$  where  $POLY(\lambda)$  is a polynomial with variable  $\lambda$ .*

**Perfectly hiding.** *A commitment scheme is perfectly hiding if, for all  $m \neq m'$ ,  $COMMIT(m, r)$  is identically distributed to  $COMMIT(m', r')$  provided that  $r$  and  $r'$  are uniformly random.*



**Computationally binding.** A commitment scheme is computationally binding if for any probabilistic polynomial time algorithm that takes the security parameter  $\lambda$  and terminates in expected time  $\text{POLY}(\lambda)$ , then the probability the algorithm outputs  $(m, r) \neq (m', r')$  such that  $\text{COMMIT}(m, r) = \text{COMMIT}(m', r')$  is at most  $2^{-\lambda}$ .

**Non-malleable.** Consider any communication game where a probabilistic polynomial time adversary receives  $c = \text{COMMIT}(m, r)$  where  $m$  is drawn from a known distribution and  $r$  is uniformly random. In the first round, the adversary must output some commitment  $c' \neq c$ . In the second round, the attacker learns  $(m, r)$  and outputs  $(m', r')$  such that  $\text{COMMIT}(m', r') = c$ . We say the commitment scheme is non-malleable if, for any such game, the random variable  $(m', r')$  is independent of  $(m, r)$ .

Some commitment schemes are malleable; for example, they allow a receiver that observes  $\text{COMMIT}(b, r)$  to compute  $\text{COMMIT}(b - 1, r)$ . This does not violate secrecy since the receiver does not learn  $b$  or can open  $\text{COMMIT}(b - 1, r)$  before the sender opens  $(b, r)$ . Yet, this malleability would pose serious security vulnerabilities in an auction. If a bidder commits to bid  $b$  with  $\text{COMMIT}(b, r)$ , the auctioneer can shill bid and commit to bidding  $b - 1$  by computing  $\text{COMMIT}(b - 1, r)$ . Constructions of non-malleable commitment schemes are involved and outside the scope of this work (see [10, 7] for a more general definition and practical constructions).

► **Definition 6** (Deferred Revelation Auction with Public Broadcast). Let  $\text{COMMIT}(\cdot, \cdot)$  be a perfectly hiding, perfectly binding, and non-malleable commitment scheme satisfying Definition 5. A collateral function  $f(\cdot, \cdot)$  takes the number of buyers  $n$  and a distribution  $D$  and outputs a collateral required from each buyer to bid in the auction. For a collateral function  $f$ ,  $\text{DRA}(f)$  with public broadcast is the following auction:

**Commitment phase (1<sup>st</sup> round):**

- Each buyer  $i \in [n]$  picks a bid  $b_i = v_i$ , draws  $r_i$  uniformly at random, and broadcast  $(i, \text{COMMIT}(b_i, r_i))$ . Moreover, buyer  $i$  sends collateral  $f(n, D)$  to the auctioneer.
- The auctioneer broadcasts “End of Commitment Phase”.

**Revelation phase (2<sup>nd</sup> round):**

- Each buyer  $i$  broadcasts  $(i, b'_i, r'_i)$  where  $b'_i = b_i$  and  $r'_i = r_i$ .
- The auctioneer broadcasts “End of Revelation Phase”.

**Resolution phase:**

- Let  $S$  denote the set of buyers for which  $\text{COMMIT}(b_i, r_i) = \text{COMMIT}(b'_i, r'_i)$ . Let  $b'_i := b_i \cdot \mathbf{1}(i \in S)$ . Let  $i^* := \arg \max_{i \in S} b_i$ .
- If  $b_{i^*} > r(D)$ , award buyer  $i^*$  the item. Charge them

$$\max\{r(D), \max_{i \in S \setminus \{i^*\}} b_i\}.$$

- The auctioneer refunds the collateral of buyer  $i \in S$ .
- The auctioneer transfers the collateral of each buyer  $i \notin S$  to buyer  $i^*$ .

**Tie-breaking:**

- All ties are broken lexicographically, with the auctioneer treated as “buyer zero”.

Before discussing how our auction differs from centralized  $\text{DRA}(f)$ , we quickly observe that  $\text{DRA}(f)$  with public broadcast is indeed strategyproof and revenue optimal.

► **Theorem 7.** For all  $f$ ,  $\text{DRA}(f)$  with public broadcast is a strategyproof optimal auction.

**Proof.** The definition for  $DRA(f)$  instructs each buyer  $i \in [n]$  to follow the strategy where buyer  $i$  sets  $b_i = v_i$ ; in the commitment phase, buyer  $i$  picks a uniformly random  $r_i$  and broadcasts a commitment  $\text{COMMIT}(v_i, r_i)$ ; and in the revelation phase, buyer  $i$  reveals  $(v_i, r_i)$ . Since  $DRA(f)$  implements the same outcome as a second-price auction, it follows this strategy profile and is an ex-post Nash equilibrium, which proves the auction is strategyproof. Moreover, because the auction maximizes expected virtual welfare, Theorem 4 (Myerson's Theorem) implies the auction is revenue optimal.  $\blacktriangleleft$

Definition 8 provides a definition for centralized  $DRA(f)$  [6]. Lemma 9 shows that centralized  $DRA(f)$  has strategy space for the auctioneer at least as ample as  $DRA(f)$  with public broadcast. To be concrete, the lemma shows that any safe deviation to  $DRA(f)$  with public broadcast maps to a safe deviation to centralized  $DRA(f)$ .

► **Definition 8** (Centralized Deferred Revelation Auction). *The centralized  $DRA(f)$  is identical to  $DRA(f)$  with public broadcast except under the following cases:*

- In  $DRA(f)$  with public broadcast, consider a history  $h$  where buyer  $i$  broadcasts a message  $m$ . In centralized  $DRA(f)$ , instead of broadcasting  $m$ , buyer  $i$  sends  $m$  to the auctioneer in a private channel. Then, the auctioneer sends  $m$  to each buyer  $j \neq i$  in a private channel.
- In  $DRA(f)$  with public broadcast, consider a history  $h$  where the auctioneer broadcasts a message  $m$ . In centralized  $DRA(f)$ , instead of broadcasting  $m$ , the auctioneer sends  $m$  to each buyer  $i \in [n]$  in a private channel.

► **Lemma 9.** *Let  $(G, \vec{s})$  be a safe deviation to  $DRA(f)$  with public broadcast, then there is a safe deviation  $(G', \vec{s}')$  to centralized  $DRA(f)$  where  $\text{REV}(G', \vec{s}') = \text{REV}(G, \vec{s})$*

**Proof.** Let  $(G', \vec{s}')$  be a communication game identical to  $(G, \vec{s})$  except on the following scenario:

- Whenever buyer  $i \in [n]$  broadcasts message  $m$  in  $(G, \vec{s})$ , in  $(G', \vec{s}')$ , buyer  $i$  sends  $m$  to the auctioneer. After receiving  $m$ , the auctioneer sends  $m$  to each buyer  $j \neq i$ .
- Whenever the auctioneer broadcasts message  $m$  in  $(G, \vec{s})$ , in  $(G', \vec{s}')$ , the auctioneer sends  $m$  to each buyer  $i \in [n]$ .

The deviation  $(G', \vec{s}')$  is safe assuming  $(G, \vec{s})$  is safe. Moreover, it induces the same allocation/payment rules, meaning it obtains the same revenue as  $(G, \vec{s})$ . This concludes the proof.  $\blacktriangleleft$

Unfortunately, the converse of Lemma 9 is untrue. There are safe deviations to centralized  $DRA(f)$  that do not map to any safe deviation in  $DRA(f)$  with public broadcast. We give the following examples to illustrate this fact.

► **Example 10.** In  $DRA(f)$  with public broadcast, buyer  $i$  sends  $(i, c_i)$  to all buyers. On the other hand, in centralized  $DRA(f)$ , buyer  $i$  must send  $(i, c_i)$  to the auctioneer, and the auctioneer “promises” to forward  $(i, c_i)$  to all buyers  $j \neq i$ . Unfortunately, buyer  $i$  cannot verify whether the auctioneer forwards their message to any buyer  $j \neq i$ . This allows the auctioneer to share  $(i, c_i)$  with a strict subset of buyers.

► **Example 11.** In  $DRA(f)$  with public broadcast, the auctioneer broadcasts the end of the commitment phase to all buyers. On the other hand, on centralized  $DRA(f)$ , the auctioneer “promises” to simultaneously announce the end of the commitment for each buyer. Suppose the auctioneer announces the end of the commitment phase to buyer  $i$  at 10:00 p.m. but only sends this announcement to buyer  $j$  at 11:00 p.m. This deviation is safe because buyer

$i$  does not know which messages buyer  $j$  received and vice-versa. Thus, at 10:10 p.m., the auctioneer requests buyer  $i$  to reveal  $(b_i, r_i)$ . Then, the auctioneer impersonates a false buyer  $z$  that bids  $b_z(b_i)$  that might depend on  $b_i$ . Buyer  $z$  sends  $\text{COMMIT}(b_z(b_i), r_z)$  only to buyer  $j$  at 10:20 p.m.

These examples do not prove there are safe deviations to centralized  $DRA(f)$  that are more profitable than any safe deviation to  $DRA(f)$  with public broadcast. They aim to showcase additional manipulations the auctioneer can perform that they cannot perform when a broadcast channel is present. Our main result will formally prove that these manipulations strictly improve the auctioneer's revenue relative to deviations that do not manipulate the order and time of messages.

Note some deviations are still possible even when buyers communicate in a broadcast channel, which makes arguing about the credibility of  $DRA(f)$  with public broadcast non-trivial – namely, the addition of false bids and the refusal to reveal false bids.

**Broadcasting false bids.** In the commitment phase, the auctioneer can impersonate a *false buyer* – agents that submit bids not coming from any *real buyer*  $i \in [n]$  – which broadcast a *false bid*  $\text{COMMIT}(\hat{b}, \hat{r})$  where  $\hat{r}$  is uniformly random. We refer to  $\tilde{b}(n, D)$  as the highest bid among all false buyers. Set  $\tilde{b}(n, D) = 0$  if the auctioneer does not impersonate any false buyer.

► **Lemma 12.** *Assume the auctioneer follows a safe deviation to  $DRA(f)$  with public broadcast. If, during the commitment phase, a false buyer broadcasts  $\text{COMMIT}(b, r)$ , and, in the revelation phase, the false buyer reveals  $(b, r)$ , then  $b$  is a random variable independent of  $\vec{v}$ .*

**Proof.** Suppose for contradiction the false buyer broadcasts  $\text{COMMIT}(b, r)$  and later reveals  $(b, r)$  where  $b$  is not independent of  $\vec{v}$ . Use this auction to construct an adversary that outputs  $\text{COMMIT}(b, r)$  whenever the false buyer does. Once the false buyer reveals  $(b, r)$ , the adversary reveals  $(b, r)$ . Because  $b$  is correlated to  $\vec{v}$ , this implies the commitment scheme is malleable, a contradiction. ◀

**Withhold false bids.** In the revelation phase, the auctioneer can refuse to reveal any bid  $\hat{b}$  submitted from a false buyer. The decision to reveal or withhold a bid from a false buyer might depend on the real bids  $\vec{b}$ .

Next, we highlight a few relevant facts about our protocol. In the commitment phase, buyer  $i$  observes commitments  $\{\text{COMMIT}(d_j^i, r_j^i)\}_j$  from both real buyers and false buyers (excluding their bid  $b_i$ ). That is,  $d_j^i$  is the  $j$ -th bid buyer  $i$  observes excluding their own bid. Let  $\beta_i(\vec{b}) = \max\{r(D), \max_j\{d_j^i\}\}$  be the highest bid buyer  $i$  observed in the commitment phase (including the reserve price  $r(D)$  and excluding their bid  $b_i$ ) when real buyers bid  $\vec{b}$ . It is possible  $\max_{i \in [n]} \beta_i(\vec{b}) > \max\{r(D), \max_{i \in [n]} \{b_i\}\}$  if the highest bid is from a false buyer.

► **Observation 13.** *Assume the auctioneer follows a safe deviation to  $DRA(f)$  with public broadcast. Then for all value profiles  $\vec{b}$ ,  $b_i > \beta_i(\vec{b})$  for at most one buyer  $i \in [n]$ .*

**Proof.** Suppose for contradiction there are distinct buyers  $i$  and  $j$  such that  $b_i > \beta_i(\vec{b})$  and  $b_j > \beta_j(\vec{b})$ . Observe that buyer  $i$  receives the bid of buyer  $j$  and buyer  $j$  receives the bid of buyer  $i$  which implies  $\beta_i(\vec{b}) \geq b_j$  and  $\beta_j(\vec{b}) \geq b_i$ . The inequalities implies  $b_i > b_j$  and  $b_j > b_i$ , a contradiction. This proves there is at most one buyer  $i$  such that  $b_i > \beta_i(\vec{b})$ . ◀

## 19:10 Credible, Optimal Auctions via Public Broadcast

► **Observation 14.** *Suppose the auctioneer follows a safe deviation to  $DRA(f)$  with public broadcast. If  $b_i > \beta_i(\vec{b})$ , then buyer  $i$  receives the item and pays  $\beta_i(\vec{b})$ .*

**Proof.** Buyer  $i$  can observe that their bid is above the reserve price and they are the highest bidder in the auction. If the auctioneer's deviation is safe, it must allocate the item to  $b_i$  and charge  $\beta_i(\vec{b})$ . ◀

The following Lemma 15 shows that under certain conditions, it is optimal for the auctioneer to reveal any bids from false buyers.

► **Lemma 15.** *Consider any safe deviation  $(G, \vec{s})$  of  $DRA(f)$  where, in the commitment phase, the auctioneer impersonates a false buyer that bids  $0 < b \leq f(n, D)$ , and, in the revelation phase, the auctioneer withholds  $b$ . Let  $h$  be the history where the auctioneer reveals or withholds  $b$ . Let  $(G', \vec{s}')$  be a new deviation identical to  $(G, \vec{s})$  except at history  $h$  the auctioneer reveals  $b$ . Then  $G'$  is a safe deviation and  $REV(G', \vec{s}') \geq REV(G, \vec{s})$ .*

**Proof.** The fact  $(G', \vec{s}')$  is a safe deviation follows directly from the fact  $(G, \vec{s})$  is a safe deviation. Next, we argue  $REV(G', \vec{s}') \geq REV(G, \vec{s})$ .

First, consider the case where no real buyer receives the item at  $(G, \vec{s})$ . Then, no real buyer will receive the item at  $(G', \vec{s}')$ . Moreover,  $(G', \vec{s}')$  improves the auctioneer's revenue relative to  $(G, \vec{s})$  because the auctioneer receives no payments but pays fewer penalties for revealing  $b$ .

Next, consider the case where some buyer  $i \in [n]$  receives the item and pays  $p$  at  $(G, \vec{s})$ . For the case where  $b \leq p$ , buyer  $i$  remains the highest bidder and pays  $p$  while the auctioneer pays fewer penalties for revealing  $b$  at  $(G', \vec{s}')$ . For the case where  $b > p$ , by assumption  $f(n, D) \geq b$ . Then, the auctioneer receives negative profits at  $(G, \vec{s})$  since the penalty for withholding  $b$  is higher than the payment they receive from buyer  $i$ . On the other hand, at  $(G', \vec{s}')$ , the revenue loss for revealing  $b$  is lower than the penalties for withholding  $b$ . ◀

### 4 DRA with Public Broadcast is Credible for $\alpha$ -Strongly Regular Distributions

In this section, we show that for any  $\alpha$ -strongly regular distributions for  $\alpha > 0$ , there is a  $f(\cdot, \cdot)$  that makes  $DRA(f)$  with public broadcast credible. Recall that  $\tilde{b}(n, D)$  is the most significant bid from a false buyer. From Lemma 12  $\tilde{b}(n, D)$  is independent of  $\vec{v}$ .

For the case where  $\alpha \geq 1$ , [6] proved Theorem 16 stating centralized  $DRA(f)$  is a credible auction if we set the collateral to be at least the optimal reserve price. Extending their result for our auction is a simple observation that any safe deviation for  $DRA(f)$  with public broadcast is also a safe deviation for centralized  $DRA(f)$ .

► **Theorem 16** (Theorem 4.1 in [6]). *Assume buyer valuations are  $\alpha$ -strongly regular for any  $\alpha \geq 1$ . If  $f(n, D) \geq r(D)$ , then centralized  $DRA(f)$  is a credible auction.*

► **Theorem 17.** *Assume buyer valuations are  $\alpha$ -strongly regular for any  $\alpha \geq 1$ . If  $f(n, D) \geq r(D)$ , then  $DRA(f)$  with public broadcast is a credible auction.*

**Proof.** Suppose for contradiction  $DRA(f)$  with public broadcast is not a credible auction when  $f(n, D) \geq r(D)$ . There is a safe deviation  $(G, \vec{s})$  to  $DRA(f)$  with public broadcast where  $REV(G, \vec{s}) > REV(D^n)$ . From Lemma 9, there is a safe deviation  $(G', \vec{s}')$  to centralized  $DRA(f)$  where  $REV(G', \vec{s}') = REV(G, \vec{s}) > REV(D^n)$ . Thus, centralized  $DRA(f)$  is not a credible auction, a contradiction to Theorem 16. ◀

The challenging case is to argue  $DRA(f)$  with public broadcast is credible for some  $f(n, D)$  for the case where  $\alpha \in (0, 1)$ . We first show that any safe deviation where false buyers only broadcast bids smaller than the collateral cannot improve the auctioneer's revenue.

► **Lemma 18.** *Assume the auctioneer follows a safe deviation to  $DRA(f)$  with public broadcast. Let  $\tilde{b}(n, D)$  be the highest bid from a false buyer (or zero if there are no false buyers). If  $f(n, D) \geq \tilde{b}(n, D)$ , the auctioneer's revenue is at most  $REV(D^n)$ .*

**Proof.** From Lemma 15 and the fact the highest false buyer bids  $k$ , it is without loss of generality to assume the auctioneer always will reveal  $\tilde{b}(n, D)$ .

Suppose the reserve price is  $r$  and let  $\hat{r} = \max\{r(D), \tilde{b}(n, D)\}$ . Note that  $\hat{r}$  is independent of  $\vec{v}$  because  $r(D)$  depends only on  $D$  and  $\tilde{b}(n, D)$  depends only on  $n$  and  $D$ . Thus, the allocation/payment rule is equivalent to a second-price auction with reserve  $\hat{r}$ . Since the second-price auction with reserve  $\hat{r}$  is a strategyproof auction, Myerson's theorem implies the revenue is at most:

$$\mathbb{E}_{\vec{v} \leftarrow D} \left[ \sum_{i=1}^n p_i(\vec{v}) \right] = \mathbb{E}_{\vec{v} \leftarrow D} \left[ \sum_{i=1}^n \varphi(v_i) \cdot x_i(\vec{v}) \right] \leq \mathbb{E} \left[ \max_i \varphi^+(v_i) \right].$$

The first equality is Theorem 4. The second inequality observes  $\sum_{i=1}^n x_i(\vec{v}) \leq 1$ . From Myerson's theorem, the optimal auction maximizes virtual surplus or equivalently,  $REV(D^n) = \mathbb{E} [\max_i \varphi(v_i)]$ . This concludes the proof. ◀

Next, we consider the case where false buyers might broadcast bids higher than the collateral. Our first Lemma will bound the revenue for events where  $v_j > \beta_j(\vec{v})$  for some buyer  $j$ . The second Lemma bounds the revenue for events where  $v_j < \beta_j(\vec{v})$  for all buyers.

► **Lemma 19.** *Assume the auctioneer follows a safe deviation to  $DRA(f)$  with public broadcast. Let  $R(\vec{v})$  be the auctioneer's revenue when buyers have value profile  $\vec{v}$ . Then*

$$\mathbb{E}_{\vec{v} \leftarrow D} [R(\vec{v}) \cdot \mathbf{1}(\exists j, v_j > \beta_j(\vec{v}))] \leq \mathbb{E}_{\vec{v} \leftarrow D} \left[ \sum_{i=1}^n \varphi(v_i) \cdot \mathbf{1}(v_i > \beta_i(\vec{v})) \right]$$

**Proof.** From Observation 13, there is at most one buyer  $i$  such that  $v_i > \beta_i(\vec{v})$  for any  $\vec{v}$ . Moreover, when  $v_i > \beta_i(\vec{v})$ , buyer  $i$  wins the item and pay  $\beta_i(\vec{v})$ . Since  $\beta_i(\vec{v})$  is independent of  $v_i$ , this payment/allocation rule is strategyproof. From Myerson's theorem, the revenue is the expected virtual surplus  $\mathbb{E}_{\vec{v} \leftarrow D} [\varphi(v_i) \cdot \mathbf{1}(v_i > \beta_i(\vec{v}))]$ . We obtain

$$\begin{aligned} \mathbb{E}_{\vec{v} \leftarrow D} [R(\vec{v}) \cdot \mathbf{1}(\exists j, v_j > \beta_j(\vec{v}))] &= \mathbb{E}_{\vec{v} \leftarrow D} \left[ \sum_{i=1}^n \beta_i(\vec{v}) \cdot \mathbf{1}(v_i > \beta_i(\vec{v})) \right] \\ &= \mathbb{E}_{\vec{v} \leftarrow D} \left[ \sum_{i=1}^n \varphi(v_i) \cdot \mathbf{1}(v_i > \beta_i(\vec{v})) \right] \quad \{\text{By Theorem 4}\} \end{aligned}$$

as desired. ◀

► **Lemma 20.** *Assume the auctioneer follows a safe deviation to  $DRA(f)$  with public broadcast. Assume  $D$  is  $\alpha$ -strongly regular for  $\alpha \in (0, 1)$ . Let  $\tilde{b}(n, D)$  be the highest bid from a false buyer (or zero if there are no false buyers). Assume  $f(n, D) \geq r(D) \left(\frac{n}{\alpha}\right)^{\frac{1-\alpha}{\alpha}} \left(\frac{1}{1-\alpha}\right)^{\frac{1}{\alpha}}$  and  $\tilde{b}(n, D) > f(n, D)$ . Let  $R(\vec{v})$  be the auctioneer's revenue when buyers have value profile  $\vec{v}$ . Then, the auctioneer's expected revenue is at most*

$$\mathbb{E}_{\vec{v} \leftarrow D} [R(\vec{v}) \cdot \mathbf{1}(\forall j, v_j < \beta_j)] \leq REV(D^n) - \mathbb{E}_{\vec{v} \leftarrow D} \left[ \sum_{i=1}^n \varphi(v_i) \cdot \mathbf{1}(v_i > \beta_i(\vec{v})) \right].$$

## 19:12 Credible, Optimal Auctions via Public Broadcast

**Proof.** When  $v_j < \beta_j(\vec{v})$  for all buyers, a false buyer is the highest bidder. Therefore,  $\max_j \beta_j(\vec{v}) = \tilde{b}(n, D) > f(n, D)$  where the inequality is a statement assumption. In this case, any buyer  $j$  can receive the item as long as the auctioneer withholds at least one bid. Because buyer  $j$  pays at most  $v_j$ , the auctioneer receives negative revenue if  $v_j < f(n, D)$ . Recall  $x_i(\vec{v})$  is an indicator variable taking value 1 if and only if buyer  $i$  receives the item. This gives

$$\begin{aligned}
& \mathbb{E}_{\vec{v} \leftarrow D} [R(\vec{v}) \cdot \mathbf{1}(\forall j, v_j < \beta_j(\vec{v}))] \\
& \leq \mathbb{E}_{\vec{v} \leftarrow D} \left[ \sum_{i=1}^n (v_i - f(n, D)) \cdot x_i(\vec{v}) \cdot \mathbf{1}(\forall j, v_j < \beta_j(\vec{v})) \cdot \mathbf{1}(v_i \geq f(n, D)) \right] \\
& \leq \mathbb{E}_{\vec{v} \leftarrow D} \left[ \sum_{i=1}^n \left( \frac{1}{\alpha} \varphi(v_i) + r(D) - f(n, D) \right) \cdot x_i(\vec{v}) \cdot \mathbf{1}(\forall j, v_j < \beta_j(\vec{v})) \cdot \mathbf{1}(v_i \geq f(n, D)) \right] \\
& \leq \mathbb{E}_{\vec{v} \leftarrow D} \left[ \sum_{i=1}^n \frac{1}{\alpha} \varphi(v_i) \cdot x_i(\vec{v}) \cdot \mathbf{1}(\forall j, v_j < \beta_j(\vec{v})) \cdot \mathbf{1}(v_i \geq f(n, D)) \right] \\
& = \mathbb{E}_{\vec{v} \leftarrow D} \left[ \sum_{i=1}^n \frac{1}{\alpha} \varphi(v_i) \cdot x_i(\vec{v}) \cdot \mathbf{1}(\forall j \neq i, v_j < \beta_j(\vec{v})) \cdot \mathbf{1}(f(n, D) \leq v_i < \beta_i(\vec{v})) \right] \\
& = \mathbb{E}_{\vec{v} \leftarrow D} \left[ \sum_{i=1}^n \frac{\varphi(v_i)}{\alpha} \cdot x_i(\vec{v}) \cdot \mathbf{1}(\forall j \neq i, v_j < \beta_j(\vec{v})) \cdot (\mathbf{1}(v_i \geq f(n, D)) - \mathbf{1}(v_i > \beta_i(\vec{v}))) \right] \\
& < \mathbb{E}_{\vec{v} \leftarrow D} \left[ \sum_{i=1}^n \frac{\varphi(v_i)}{\alpha} \cdot \mathbf{1}(v_i \geq f(n, D)) \right] - \mathbb{E}_{\vec{v} \leftarrow D} \left[ \sum_{i=1}^n \varphi(v_i) \cdot \mathbf{1}(v_i > \beta_i(\vec{v})) \right]
\end{aligned}$$

The second line observes that if buyer  $i$  receives the item, they pay at most  $v_i$ , and the auctioneer loses a collateral of  $f(n, D)$  by withholding at least one bid. The third line invokes Lemma 26. To see that the assumptions for the Lemma are satisfied, let  $E$  be the event where  $v_i \geq f(n, D)$  and observe that  $f(n, D) \geq r(D)$  for all  $n \geq 1$  and  $\alpha \in (0, 1)$ . The fourth line observes  $f(n, D) \geq r(D)$ . The fifth line observes the event  $\{\forall j, v_j < \beta_j(\vec{v})\}$  implies  $\{v_i < \beta_i(\vec{v})\}$  and uses the fact  $\beta_i(\vec{v}) > f(n, D)$ . The sixth line uses the fact  $\mathbf{1}(a \leq X < b) = \mathbf{1}(X \geq a) - \mathbf{1}(X \geq b)$  for any random variable  $X$  and constants  $a > b$ . The seventh line uses the fact  $\alpha > 1$  and Observation 13 which states the event  $\{v_i > \beta_i\}$  implies  $x_i(\vec{v})$  and  $v_j < \beta_j(\vec{v})$  for all  $j \neq i$  since  $v_i$  expects to win the item. Moreover, we use the fact

$$x_i(\vec{v}) \cdot \mathbf{1}(\forall j \neq i, v_j < \beta_j(\vec{v})) \cdot \mathbf{1}(v_i \geq f(n, D)) \leq \mathbf{1}(v_i \geq f(n, D)).$$

To conclude, we must show that

$$\mathbb{E}_{\vec{v} \leftarrow D} \left[ \sum_{i=1}^n \frac{\varphi(v_i)}{\alpha} \cdot \mathbf{1}(v_i \geq f(n, D)) \right] \leq \text{REV}(D^n).$$

From Lemma 28,

$$\begin{aligned}
& \mathbb{E}_{\vec{v} \leftarrow D} \left[ \sum_{i=1}^n \frac{\varphi(v_i)}{\alpha} \cdot \mathbf{1}(v_i \geq f(n, D)) \right] \\
& = \frac{1}{\alpha} \left( \frac{1}{1-\alpha} \right)^{\frac{1}{1-\alpha}} \left( \frac{r(D)}{f(n, D)} \right)^{\frac{\alpha}{1-\alpha}} \mathbb{E}_{\vec{v} \leftarrow D} \left[ \sum_{i=1}^n \varphi(v_i) \cdot \mathbf{1}(v_i \geq r(D)) \right] \\
& \leq \frac{\alpha}{\alpha n} \mathbb{E}_{\vec{v} \leftarrow D} \left[ \sum_{i=1}^n \varphi(v_i) \cdot \mathbf{1}(v_i \geq r(D)) \right]
\end{aligned}$$

$$\begin{aligned}
&= \frac{\alpha n}{\alpha n} \mathbb{E}_{v_1 \leftarrow D} [\varphi(v_1) \cdot \mathbf{1}(v_1 \geq r(D))] \\
&= \text{REV}(D) \\
&\leq \text{REV}(D^n)
\end{aligned}$$

The second line observes  $f(n, D) \geq r(D)$  and applies Lemma 28. The third line uses the assumption  $f(n, D) \geq r(D) \left(\frac{n}{\alpha}\right)^{\frac{1-\alpha}{\alpha}} \left(\frac{1}{1-\alpha}\right)^{\frac{1}{\alpha}}$ . The fourth line observes  $\varphi(v_1), \dots, \varphi(v_n)$  are i.i.d.. The fifth line observes  $r(D)$  is the optimal reserve price, and so  $\mathbb{E}_{v_1 \leftarrow D} [\varphi(v_1) \cdot \mathbf{1}(v_1 \geq r(D))]$  is the optimal revenue for the single buyer auction (Theorem 4). The last line observes the revenue is non-decreasing in the number of buyers. ◀

Next, we prove our main result.

► **Theorem 21.** *Assume the auctioneer follows a safe deviation to  $\text{DRA}(f)$  with public broadcast and assume all buyer valuations are  $\alpha$ -strongly regular for  $\alpha > 0$ . Then, there is an  $f$  such that  $\text{DRA}(f)$  with public broadcast is a credible auction.*

**Proof.** Set  $f(n, D) = r(D) \left(\frac{n}{\alpha}\right)^{\frac{1-\alpha}{\alpha}} \left(\frac{1}{1-\alpha}\right)^{\frac{1}{\alpha}}$ . Observe for all  $n \geq 1$  and  $\alpha > 0$ ,  $f(n, D) \geq r(D)$ . For the case where  $\alpha \geq 1$ , the proof follows directly from Theorem 17 because  $f(n, D) \geq r(D)$ . Next, consider the case where  $\alpha \in (0, 1)$ . Recall  $\tilde{b}(n, D)$  refers to the highest bid among false buyers (or zero if no false buyer exists).  $R(\vec{v})$  refers to the auctioneer's revenue when buyers have value  $\vec{v}$ . For the case where  $f(n, D) \geq \tilde{b}(n, D)$ , Lemma 18 states the auctioneer's revenue is at most  $\text{REV}(D^n)$ . Next, consider the case where  $f(n, D) < \tilde{b}(n, D)$ . We can write the revenue as

$$\begin{aligned}
\mathbb{E}_{\vec{v} \leftarrow D} [R(\vec{v})] &= \mathbb{E}_{\vec{v} \leftarrow D} [R(\vec{v}) \cdot \mathbf{1}(\exists j, v_j > \beta_j(\vec{v}))] + \mathbb{E}_{\vec{v} \leftarrow D} [R(\vec{v}) \cdot \mathbf{1}(\forall j, v_j < \beta_j(\vec{v}))] \\
&\leq \mathbb{E}_{\vec{v} \leftarrow D} \left[ \sum_{i=1}^n \varphi(v_i) \cdot \mathbf{1}(v_i > \beta_i(\vec{v})) \right] + \text{REV}(D^n) - \mathbb{E}_{\vec{v} \leftarrow D} \left[ \sum_{i=1}^n \varphi(v_i) \cdot \mathbf{1}(v_i > \beta_i(\vec{v})) \right] \\
&= \text{REV}(D^n)
\end{aligned}$$

The second line is due to Lemma 20 and Lemma 19. This shows the auctioneer's revenue is at most  $\text{REV}(D^n)$  and proves there is a  $f$  such that  $\text{DRA}(f)$  is a credible auction. ◀

## 5 Public Broadcast is Necessary

This section revisits the fact centralized  $\text{DRA}(f)$  is not a credible auction for certain  $\alpha$ -strongly regular valuations when  $\alpha \in (0, 1)$ .

► **Theorem 22** (Theorem 4.4 in [6]). *For all  $f$ ,  $\alpha \in (0, 1)$ , there exists a  $D^n$  that is  $\alpha$ -strongly regular such that centralized  $\text{DRA}(f)$  is not credible for instance  $D^n$ .*

The following is a special case for the instance given in the proof of Theorem 22. By inspection, this strategy is a safe deviation for centralized  $\text{DRA}(f)$  since, in the view of each buyer, the strategy is indistinguishable from the promised auction. In this strategy the auctioneer only sends a shill bids to buyer  $B$  that depend on the bid of buyer  $A$ . This would not be possible if, rather than relying on the auctioneer to forward messages, messages were sent in a broadcast channel because any message one buyer receives is also received by other buyers.

► **Definition 23** (Adaptive Reserve Price). *Consider an auctioneer who promises to implement centralized  $\text{DRA}(f)$  on an instance with two buyers  $A$  and  $B$ . The adaptive reserve price deviation is the following deviation:*

## 19:14 Credible, Optimal Auctions via Public Broadcast

- $A$  sends  $(A, c_A)$  to the auctioneer.
- $B$  sends  $(B, c_B)$  to the auctioneer.
- The auctioneer sends  $(B, c_B)$  to  $A$  and  $(A, c_A)$  to  $B$ .
- The auctioneer sends “End of the Commitment Phase” to buyer  $A$ , then requests  $A$  to reveal their bid.  $A$  complies and reveals  $(b_A, r_A)$  such that  $c_A = \text{COMMIT}(b_A, r_A)$ .
- The auctioneer picks a large threshold  $T$ :
  - If  $b_A < T$ , the auctioneer sends “End of Commitment Phase” to buyer  $B$ , then requests  $B$  to reveal their bid (who complies by revealing  $(b_B, r_B)$  such that  $c_B = \text{COMMIT}(b_B, r_B)$ ). The auctioneer implements the allocation/payment rule for the second-price auction with reserve  $r(D)$  on bids  $\{b_A, b_B\}$ .
  - If  $b_A \geq T$ , the auctioneer impersonates a false buyer  $C$ . Let  $r_C$  be uniformly random and  $b_C = b_A + f(2, D)$ . Then, the auctioneer sends  $(C, \text{COMMIT}(b_C, r_C))$  to  $B$ . The auctioneer sends “End of Commitment Phase” to buyer  $B$ , then request  $B$  to reveal their bid.  $B$  complies and reveal  $(b_B, r_B)$  such that  $c_B = \text{COMMIT}(b_B, r_B)$ . Next, the auctioneer proceeds as follows:
    - \* If  $r(D) \geq \max\{b_A, b_B\}$ , the auctioneer reveals all bids. No one receives the item.
    - \* If  $b_B < b_A$  and  $b_A > r(D)$ , the auctioneer reveals all bids and allocates the item to  $A$  and charges  $\max\{r(D), b_B\}$ .
    - \* If  $b_B \in [b_A, b_C]$  and  $b_B > r(D)$ , the auctioneer reveals  $b_A$  and hides  $b_C$  from  $B$ . Then, the auctioneer allocates the item to  $B$  and charges  $\max\{b_A, r(D)\}$ .
    - \* If  $b_B > b_C$ , the auctioneer reveals all bids and allocates the item to  $B$  who pays  $b_C$ .

## 6 DRA over Public Broadcast for Regular Distributions

Although *DRA* with public broadcast extends the class of distributions where it is credible, it is not a magic bullet. Indeed, Theorem 25 states there is an instance with a single buyer drawn from a regular distribution that witnesses *DRA*( $f$ ) with public broadcast is not credible. The proof relies on a similar negative result in [6].

► **Theorem 24** (Theorem 4.4 in [6]). *There is a regular distribution  $D$  such that for all  $f(\cdot, \cdot)$ , centralized *DRA*( $f$ ) is not credible even when there is a single buyer with valuation drawn from  $D$ .*

► **Theorem 25.** *There is a regular distribution  $D$  such that for all  $f(\cdot, \cdot)$ , *DRA*( $f$ ) over public broadcast is not credible even when there is a single buyer with a valuation drawn from  $D$ .*

**Proof.** We will argue for any instance with a single buyer, any safe deviation to centralized *DRA*( $f$ ) maps to a safe deviation to *DRA*( $f$ ) over public broadcast. To see, let  $(G, \vec{s})$  be a safe deviation to centralized *DRA*( $f$ ). Let  $(G', \vec{s}')$  be a deviation to *DRA*( $f$ ) with public broadcast identical to  $(G, \vec{s})$  except on the following cases:

- Whenever the buyer sends  $m$  to the auctioneer in  $(G, \vec{s})$ , the buyer broadcast  $m$  in  $(G', \vec{s}')$ .
- Whenever the auctioneer sends  $m$  to the buyer in  $(G, \vec{s})$ , the auctioneer broadcast  $m$  in  $(G', \vec{s}')$ .

$(G', \vec{s}')$  is a safe deviation because  $(G, \vec{s})$  is a safe deviation. Moreover,  $(G', \vec{s}')$  induces the same allocation/payment rule as  $(G, \vec{s})$ ; therefore,  $\text{REV}(G', \vec{s}') = \text{REV}(G, \vec{s})$ .

From Theorem 24, there is a  $D$  such that for all  $f(\cdot, \cdot)$ , there is a safe deviation  $(G, \vec{s})$  to centralized *DRA*( $f$ ) where  $\text{REV}(G, \vec{s}) > \text{REV}(D)$ . The mapping above proves there is a safe deviation  $(G', \vec{s}')$  to *DRA*( $f$ ) with public broadcast where  $\text{REV}(G', \vec{s}') = \text{REV}(G, \vec{s}) > \text{REV}(D)$ . This proves *DRA*( $f$ ) with public broadcast is not a credible auction on instance  $D$  as desired. ◀



## 7 Conclusion

Improving the transparency and fairness in Internet platforms is becoming an essential concern for regulators, as observed by the US Department of Justice lawsuit against Google [17]. It is unlikely that customers could unilaterally detect and, more importantly, prove the sophisticated market manipulations alleged in the complaint. Credible auctions formalize the notion that an auction is “auditable” by its participants: the auctioneer has no incentive to deviate from running the promised mechanism in earnest. However, existing credible auctions suffer from restrictive assumptions on valuation distributions and exclude valuations with tails thicker than the exponential distribution.

This work shows that censorship-resistant broadcast channels like blockchains are helpful to circumvent this problem. We propose the deferred revelation auction with public broadcast, a natural modification of the centralized deferred revelation auction of [6]. Although our auction represents a simple modification of a known auction, the resulting auction is credible in instances where no known communication-efficient auctions were known to be credible. This work builds on the emerging line of research that attempts to improve the performance of economic mechanisms by appending cryptographic primitives to them. The need for large collateral is a limitation of our work. Minimizing collateral is an important objective to make these auctions practical which we leave as future direction.

---

### References

- 1 Mohammad Akbarpour and Shengwu Li. Credible auctions: A trilemma. *Econometrica*, 88(2):425–467, 2020.
- 2 Felix Brandt. Cryptographic protocols for secure second-price auctions. In *International Workshop on Cooperative Information Agents*, pages 154–165. Springer, 2001.
- 3 Tarun Chitra, Matheus VX Ferreira, and Kshitij Kulkarni. Credible, optimal auctions via blockchains. *arXiv preprint*, 2023. [arXiv:2301.12532](https://arxiv.org/abs/2301.12532).
- 4 Meryem Essaidi, Matheus VX Ferreira, and S Matthew Weinberg. Credible, strategyproof, optimal, and bounded expected-round single-item auctions for all distributions. In *13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.
- 5 Matheus VX Ferreira, Daniel J Moroz, David C Parkes, and Mitchell Stern. Dynamic posted-price mechanisms for the blockchain transaction-fee market. In *Proceedings of the 3rd ACM Conference on Advances in Financial Technologies*, pages 86–99, 2021.
- 6 Matheus VX Ferreira and S Matthew Weinberg. Credible, truthful, and two-round (optimal) auctions via cryptographic commitments. In *Proceedings of the 21st ACM Conference on Economics and Computation*, pages 683–712, 2020.
- 7 Marc Fischlin and Roger Fischlin. Efficient non-malleable commitment schemes. In *Advances in Cryptology—CRYPTO 2000: 20th Annual International Cryptology Conference Santa Barbara, California, USA, August 20–24, 2000 Proceedings 20*, pages 413–431. Springer, 2000.
- 8 Aadityan Ganesh, Clayton Thomas, and S Matthew Weinberg. Revisiting the primitives of transaction fee mechanism design, 2024.
- 9 Noemi Glaeser, István András Seres, Michael Zhu, and Joseph Bonneau. Cicada: A framework for private non-interactive on-chain auctions and voting. *Cryptology ePrint Archive*, 2023.
- 10 Dakshita Khurana and Amit Sahai. How to achieve non-malleability in one or two rounds. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 564–575. IEEE, 2017.
- 11 Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. In *Concurrency: the works of leslie lamport*, pages 203–226. ACM, 2019.

## 19:16 Credible, Optimal Auctions via Public Broadcast

- 12 Xiao Lin and Ce Liu. Credible persuasion. *Journal of Political Economy*, 132(7):000–000, 2024.
- 13 Roger B Myerson. Optimal auction design. *Mathematics of operations research*, 6(1):58–73, 1981.
- 14 Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2009. URL: <http://www.bitcoin.org/bitcoin.pdf>.
- 15 Michael O Rabin and Christopher Thorpe. Time-lapse cryptography. *Harvard University*, 2006.
- 16 Nirvan Tyagi, Arasu Arun, Cody Freitag, Riad Wahby, Joseph Bonneau, and David Mazières. Riggs: Decentralized sealed-bid auctions. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 1227–1241, 2023.
- 17 US Department of Justice. Justice department sues google for monopolizing digital advertising technologies, 2023. URL: <https://www.justice.gov/opa/pr/justice-department-sues-google-monopolizing-digital-advertising-technologies>.
- 18 Matheus Venturyne Xavier Ferreira and David C Parkes. Credible decentralized exchange design via verifiable sequencing rules. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 723–736, 2023.
- 19 Andrew Chi-Chih Yao. Protocols for secure computations. *Mathematics of operations research*, 82(1):160–164, 1982.

### A Mathematical Background

► **Lemma 26** (Lemma 7.1 in [6]). *Let  $D$  be  $\alpha$ -strongly regular for  $\alpha > 0$ . Let  $E$  be an event such that  $v \geq r(D)$  with probability 1 conditioned on  $E$ . Then*

$$\mathbb{E}[v|E] \leq \frac{1}{\alpha} \mathbb{E}[\varphi^D(v)|E] + r(D).$$

**Proof.** Because  $D$  is  $\alpha$ -strongly regular, for all  $x' > x$ ,

$$\varphi^D(x') - \varphi^D(x) \geq \alpha(x' - x)$$

Then for any  $x' \geq r(D)$ ,  $x' \leq \frac{1}{\alpha}(\varphi^D(v) - \varphi^D(r(D))) + r(D)$ . By definition  $\varphi^D(r(D)) = 0$ . Conditioned on event  $E$ , we have that  $v \geq r(D)$  for all  $v$ . We conclude  $\mathbb{E}_{\bar{v} \leftarrow D}[v|E] \leq \frac{1}{\alpha} \mathbb{E}[\varphi^D(v)|E] + r(D)$  as desired. ◀

► **Lemma 27** (Lemma 7.2 in [6]). *Let  $D$  be a  $\alpha$ -strongly regular distribution. Then for all  $p \geq r(D)$ ,*

$$p \cdot \Pr_{\bar{v} \leftarrow D}[v \geq p] \leq r(D) \cdot \Pr_{\bar{v} \leftarrow D_0}[v \geq r(D)] \left( \frac{1}{1-\alpha} \right)^{\frac{1}{1-\alpha}} \left( \frac{r}{p} \right)^{\frac{\alpha}{1-\alpha}}.$$

► **Lemma 28.** *Let  $D$  be a  $\alpha$ -strongly regular for  $\alpha > 0$ . Then for all  $p \geq r(D)$ ,*


$$\mathbb{E}_{\bar{v} \leftarrow D}[\varphi(v) \cdot \mathbf{1}(v \geq p)] \leq \mathbb{E}_{\bar{v} \leftarrow D}[\varphi(v) \cdot \mathbf{1}(v \geq r(D))] \left( \frac{1}{1-\alpha} \right)^{\frac{1}{1-\alpha}} \left( \frac{r}{p} \right)^{\frac{\alpha}{1-\alpha}}.$$

**Proof.** Consider a single item, single bidder posted-price mechanism that offers the item at a price  $p$ . The bidder value is drawn from  $D$ . The revenue is  $p \Pr_{\bar{v} \leftarrow D}[v \geq p]$  because the buyer purchases whenever their value exceeds  $p$ . From Myerson’s theorem,  $p \Pr_{\bar{v} \leftarrow D}[v \geq p] = \mathbb{E}_{\bar{v} \leftarrow D}[\varphi(v) \cdot \mathbf{1}(v \geq p)]$ . The result follows directly by applying Lemma 27 to the left-hand side of the inequality. ◀

# Optimizing Exit Queues for Proof-Of-Stake Blockchains: A Mechanism Design Approach

Michael Neuder ✉

Ethereum Foundation, New York, NY, USA

Malleh Pai ✉ 

Special Mechanisms Group, Consensys Inc, Dallas, TX, USA

Rice University, Houston, TX, USA

Max Resnick ✉

Special Mechanisms Group, Consensys Inc, Dallas, TX, USA

---

## Abstract

Byzantine fault-tolerant consensus protocols have provable safety and liveness properties for static validator sets. In practice, however, the validator set changes over time, potentially eroding the protocol's security guarantees. For example, systems with accountable safety may lose some of that accountability over time as adversarial validators exit. As a result, protocols must rate limit entry and exit so that the set changes slowly enough to ensure security. Here, the system designer faces a fundamental trade-off. The harder it is to exit the system, the less attractive staking becomes; alternatively, the easier it is to exit the system, the less secure the protocol will be.

This paper provides the first systematic study of exit queues for Proof-of-Stake blockchains. Given a collection of validator-set consistency constraints imposed by the protocol, the social planner's goal is to provide a constrained-optimal mechanism that minimizes disutility for the participants. We introduce the MINSLACK mechanism, a dynamic capacity first-come-first-served queue in which the amount of stake that can exit in a period depends on the number of previous exits and the consistency constraints. We show that MINSLACK is optimal when stakers equally value the processing of their withdrawal. When stakers values are heterogeneous, the optimal mechanism resembles a priority queue with dynamic capacity. However, this mechanism must reserve exit capacity for the future in case a staker with a much higher need for liquidity arrives. We conclude with a survey of known consistency constraints and highlight the diversity of existing exit mechanisms.

**2012 ACM Subject Classification** Information systems

**Keywords and phrases** Mechanism Design, Market Design, Accountable Safety, Proof-of-Stake, Blockchain

**Digital Object Identifier** 10.4230/LIPIcs.AFT.2024.20

**Related Version** *Full Version*: <https://arxiv.org/pdf/2406.05124>

### Supplementary Material

*Software (Source Code for simulations)*: <https://github.com/michaelneuder/withdrawals>

**Acknowledgements** The authors thank Aditya Asgaonkar, Vitalik Buterin, Francesco D'Amato, Barnabé Monnot, and Tim Roughgarden for helpful discussions.

## 1 Introduction

In Proof-of-Stake networks, validators use tokens to participate in the consensus protocol. These staked tokens serve two purposes. First, they solve the problem of Sybil resistance: agents who operate two validators must procure twice as much stake as those who only manage one. Second, they allow the protocol to hold validators accountable for violating the predefined rules. A validator's stake can be *slashed* if adversarial behavior is detected,



© Michael Neuder, Malleh Pai, and Max Resnick;  
licensed under Creative Commons License CC-BY 4.0  
6th Conference on Advances in Financial Technologies (AFT 2024).

Editors: Rainer Böhme and Lucianna Kiffer; Article No. 20; pp. 20:1–20:22

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

providing *crypto-economic security* to the system.<sup>1</sup> Most modern blockchains (e.g., Ethereum and Solana) implement a version of Proof-of-Stake and the principles of staking have been extended beyond base-layer chains and into the smart contract layer (e.g., re-staking as popularized by EigenLayer).

The literature typically treats the set of stakers as static to establish positive results; however, in practice, staking protocols have a validator set that changes over time. New agents may arrive and wish to stake, while existing stakers may want to withdraw their tokens for use elsewhere (see, e.g., [18]). How should a stake-based protocol design this egress procedure?<sup>2</sup> There are two competing desiderata. The first is the security of the underlying protocol. For example, if a malicious validator can corrupt the service for personal gain but then withdraw their stake before the corruption is detected, the validator is immune to punishment, and the protocol is not secure. We describe these concerns in more detail in Section 6. The second desideratum is ensuring that validators can quickly enter and exit the system since delays decrease the utility of participation. Offering fast withdrawals also indirectly benefits the protocol since, *ceteris paribus*, a more rigid protocol must offer higher rewards in the form of emissions to compensate users for the decrease in their optionality. While these general principles are well understood, the design of the exit procedures in the context of blockchains has yet to attract much formal attention.<sup>3</sup> As a consequence, the optimal queue designs we suggest in Sections 4 and 5 perform much better than those currently used in practice, which we survey in Section 6.

The first contribution of this paper is to formally define the designer’s dilemma as a constrained optimization problem: minimizing the adverse effects of withdrawal delays while satisfying the protocol’s safety constraints. In the setting where all validators have the same time sensitivity, we show that a stateful, first-come-first-served queue where the amount of stake withdrawn in each period depends on the history of previous periods is constrained optimal.

However, even among honest validators, the desire to exit can be heterogeneous – for example, a capital-constrained validator might need to withdraw urgently to meet a margin call elsewhere. In this setting, a first-come-first-served queue may no longer be optimal, as the time-sensitive validator may have a sizeable negative utility if their withdrawal is not processed promptly. Instead, the mechanism must allow more time-sensitive stakers to cut the line to achieve efficiency. Further, in some cases, the optimal mechanism reserves capacity for the future in case more time-sensitive agents arrive. We formally define this mechanism as the solution to a Markov Decision Process (MDP) and show that an appropriately defined dynamic Vickrey-Clark-Groves (VCG) mechanism can implement the efficient outcome.

We complement these results with a survey to connect our theoretical model to practice. First, we discuss the exit mechanisms in use today by popular blockchains. Our results suggest that some of these mechanisms are either (highly) sub-optimal or the designers believed the mechanism should satisfy additional constraints external to our model. Further, we should note that no protocol that we are aware of uses a payment-for-priority mechanism.<sup>4</sup> Combined with our theoretical results, this collection may be helpful for blockchains and staking protocols more generally to design or improve their exit procedures.

---

<sup>1</sup> See [20] for an extended definition of crypto-economic security.

<sup>2</sup> Similar considerations apply to the design of deposit (ingress) procedures; we focus on withdrawals (egress) in the present paper.

<sup>3</sup> We defer a discussion of the related literature to Section 2.

<sup>4</sup> Priority payment is standard in other congested parts of blockchains, notably in the context of transaction fees, and a substantial literature explores the design of such fees, see, e.g., [37, 24].

**Organization.** Section 2 presents the related literature. Section 3 defines the model and outlines the form of the security constraints of a staking system. Section 4 studies the common-value setting by defining *MINSLACK* and proving its optimality. Section 5 introduces heterogeneous values, formalizes the extended problem as an MDP, and presents numerical results quantifying the performance of various algorithms. Section 6 justifies the form of the constraints, presents the intricacies of the Ethereum design, and surveys other staking withdrawal procedures. Section 7 concludes.

## 2 Literature Review

Early in the development of Ethereum, Vitalik outlined concerns about a long-range attack on a Proof-of-Stake blockchain [9]. In particular, he described how a malicious staker could withdraw his ETH while building a competing fork starting from a historical epoch before he withdrew. This way, the staker cannot be punished for creating the fork because he has exited from the consensus mechanism. The solution, he argued, was *weak subjectivity*, where nodes locally store a subjectivity “checkpoint” block and ignore any messages from before that epoch [1]. Weak-subjectivity checkpoints prevent long-range attacks but require the validator set to change slowly enough to reach a subjectivity checkpoint without a long-range attack. The naïve approach simply delays *all* withdrawals for the weak-subjectivity period, guaranteeing the chain’s safety. Buterin argued in [10] that this imposed too strict a penalty on validators who wanted to withdraw under normal circumstances when there was no evidence of an attack, arguing for an exit queue model instead. [12] gave a formal case for why the consistency imposed by the exit queue was enough to safely last until the next weak-subjectivity checkpoint through an inductive argument. As detailed in Section 6, the FCFS exit queue has been used in Ethereum since April 2023, when the Shanghai/Capella hard fork enabled beacon chain validators to withdraw. More generally, Buterin was concerned with preserving the formal property known as *accountable safety* [14]. Accountable safety guarantees that in the case of a safety violation, the fault is attributable to a subset of validators (because they must have signed conflicting attestations).

[28, 8] formalized the economic limits of consensus mechanisms and showed that no partially synchronous protocol can fully implement slashing rules without bounding the resolution time of communication between honest nodes. Given some bound on this overhead, protocols could implement slashing against an attacker with  $< 2/3$  of the total stake, a positive result that justifies using the weak-subjectivity period as a heuristic for preventing long-range attacks. [30] formalized the relationship between accountable safety and finality, while [2] introduced a new confirmation rule for potentially improving the pre-finality guarantees for transactions in Ethereum. [25] proposed allowing some withdrawals to be processed ahead of others by the nature of originating from a different source that required payment.

Systematic attention to the design of exit procedures in blockchains has been sparse; however, mechanism design has proved useful for blockchain designers in other contexts, particularly in designing transaction-fee mechanisms. The question here is similar: if there is a finite supply of block space and demand may exceed supply, how should the block space be allocated? Bitcoin used a simple “pay-as-bid” mechanism, which was fruitfully studied using tools from queueing theory in [24]. Pay-as-bid mechanisms result in strategic bidding, contributing to poor user experience. In 2021, Ethereum adopted a dynamic reserve price mechanism, EIP-1559, which was comprehensively studied in the seminal [37] (see also [38]). There has been recent interest in studying dynamic mechanism design in this setting – see e.g., [31, 32]. In a different context, a few market design papers have studied the design of queues, mainly in organ transplantation—see, e.g., [27] and [40].

### 3 Model

Time is discrete, and each period corresponds to a point during which a validator may request a withdrawal and be removed from the active set – for example, Ethereum processes withdrawals at epoch boundaries. Denote the set of possible validators,  $V$ , and at each time  $t$ , let  $S(v, t) \in \{0, 1\}$  denote whether validator  $v$  is currently staked or not.<sup>5</sup> Thus the total amount staked at period  $t$  is  $\bar{S}(t) = \sum_v S(v, t)$ .

At the end of each period, any validator may signal their desire to withdraw their stake by joining a waiting list  $W(t)$ . For now, we model every element of the waiting list as a tuple  $(v, t')$ , where  $v$  is the validator identity and  $t'$  is the period at which they initiated their withdraw. Note that we must have that  $t' \leq t$ , as any element in the waiting list must have joined in the past. Let  $R(t)$  be the set of exit requests arriving in period  $t$ .<sup>6</sup>

An exit mechanism  $\mathcal{M}$  in each period  $t$ , given a waiting list  $W(t)$ , selects a subset  $P(t) \subseteq W(t)$  of withdrawal requests to *process*. We allow the exit mechanism's choices to depend on past choices. Formally, let us define a history of previous withdrawal requests as:

$$H(t) = (P(1), P(2) \dots, P(t-1)),$$

and the set of all possible histories as  $\mathcal{H}$ . A mechanism then formally is:

$$\mathcal{M} : \mathcal{H} \times W(t) \mapsto \{0, 1\}^{|W(t)|},$$

where the binary string is an indicator function for the withdrawals processed during each period. The system follows the rules of motion:

$$\begin{aligned} W(t) &= W(t-1) \setminus P(t-1) \cup R(t), \\ P(t) &= \mathcal{M}(H(t), W(t)), \\ H(t+1) &= H(t) \cup P(t). \end{aligned}$$

The stake distribution  $S(\cdot, t)$  is then updated based on the exits and fresh entries. In words,  $W(t)$  is the *waiting list* of withdraw requests at the *beginning* of period  $t$ ,  $P(t)$  is the subset of waiting to withdraw requests that are *processed* in period  $t$ .  $H(t)$  is the *history* of processed requests up to and including period  $t$ .

The number of withdrawals allowed over various time horizons constrains the protocol designer. We model this as a finite set of constraints, each described by a tuple  $(\delta, T) \in [0, 1] \times \mathbb{N}$ . A constraint of  $(\delta, T)$  means that if, in any period  $t$ , the total stake is  $\bar{S}(t)$ , then the maximum number of withdrawals processed over the following  $T$  periods (from  $t+1$  thru  $t+T$ ) is bounded above by  $\delta \times \bar{S}(t)$ . We take the constraints as given, motivating this construction in Section 6.1. Formally, the designer faces some  $k$  constraints given by  $\mathcal{C} = \{(\delta_1, T_1), \dots, (\delta_k, T_k)\}$  and aims to maximize the utility of the validators withdrawing from the staking system. Calculating this utility depends on validators having differing values for exiting the system; we begin by examining the simplest case, where each validator has a common value.

<sup>5</sup> For simplicity, we assume that each validator has the same quantity of tokens staked, normalized to 1. This can easily be relaxed.

<sup>6</sup> Most blockchains also limit entry to have a stable validator set for consensus. In this paper, we focus on the design of exit queues and consider entry unrestricted.

## 4 Homogeneous Values

To begin our analysis, we consider the case where all agents have the same value for withdrawing or equivalently face the same economic penalty for each period between when they make a withdrawal request and when that request is fulfilled. In this case, the social planner cannot increase efficiency by reordering withdrawal requests, so efficiency demands that exit requests be processed as quickly as possible without violating the established constraints.

Given the constraint set  $\mathcal{C}$ , the following algorithm, which we call MINSLACK, greedily processes the maximum amount of withdrawals allowed within the bounds of the constraints. In other words, for every constraint  $(\delta_i, T_i)$ , calculate the difference between  $\delta_i \bar{S}(t - T_i)$  and the amount withdrawn in periods  $t - T_i$  thru  $t - 1$ .<sup>7</sup> This difference is the maximum number of withdrawals constraint  $i$  allows in period  $t$  given the previous history. It follows that the lowest of these slacks is the maximum amount that can be withdrawn in this period without violating constraints. Since the protocol is indifferent about the withdrawal order, if there is more demand for withdrawing than the allowed quantity, a natural solution is to use the FCFS rule to tie-break. We present this algorithm as Algorithm 1.

---

### Algorithm 1 MINSLACK.

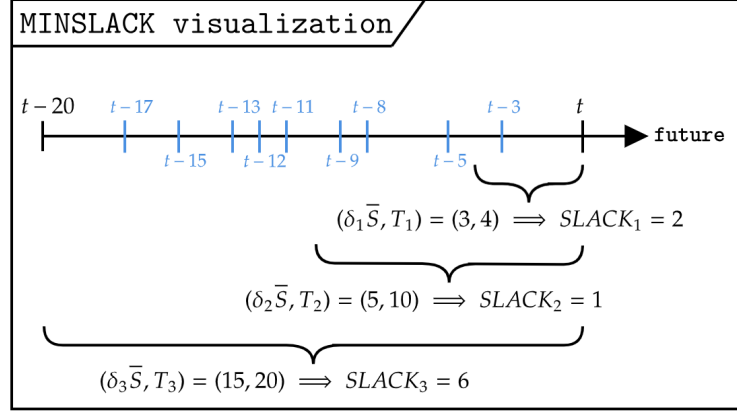
---

- 1: **Input:** Constraints  $\mathcal{C} = \{(\delta_1, T_1), \dots, (\delta_k, T_k)\}$ .
  - 2: **Input:** Initial staking  $S(\cdot, 0)$ .
  - 3:  $\bar{S}(0) \leftarrow \sum_v S(v, 0)$ .
  - 4: **Initialize:**  $H(0), W(0), P(0) \leftarrow \text{NULL}$ .
  - 5: **Initialize:**  $\bar{P}(0) = 0$ .
  - 6: **for** each period  $t \geq 1$  **do**
  - 7:    $W(t) \leftarrow W(t-1) \setminus P(t-1) \cup R(t)$ .
  - 8:   **for** each constraint  $i \leq k$  **do**
  - 9:      $\text{SLACK}_i \leftarrow \delta_i \bar{S}(t - T_i) - \sum_{\tau=t-T_i+1}^{t-1} \bar{P}(\tau)$ .
  - 10:   **end for**
  - 11:    $\text{MINSLACK} \leftarrow \min\{\text{SLACK}_i : 1 \leq i \leq k\}$ .
  - 12:    $P(t) \leftarrow$  Largest prefix of  $W(t)$  such that total withdrawn  $\leq \text{MINSLACK}$
  - 13:    $\bar{P}(t) \leftarrow$  Total withdrawn in  $P(t)$
  - 14:    $H(t+1) \leftarrow H(t) \cup P(t)$
  - 15:   **Update:**  $S(v, t)$  based on  $P(t)$ .
  - 16: **end for**
- 

Proving that this algorithm is feasible and optimal is straightforward: as designed, it processes the maximum amount allowed by the protocol constraints, but never more. Before presenting this result, we explain why such a queue design may be helpful. As we describe in Section 6.2, the relevant constraints are that a given fraction of stake cannot withdraw over an extended period (e.g.,  $\mathcal{O}(\text{weeks})$ ). Nevertheless, the actual queue implemented on Ethereum allows the withdrawal of at most eight validators per epoch (a value set in EIP-7514, [29]). In practice, validators must wait longer than required during periods with higher-than-expected withdrawals. For example, in January 2024, the withdrawal queue on Ethereum rose to

---

<sup>7</sup> For expositional simplicity, we elide over the difficulties caused by the fact that  $\delta_i \bar{S}(\cdot)$  may not be a whole number. In what follows we implicitly assume that this is a whole number, alternately, we could allow for fractional withdrawals at the cost of significantly messier notation.



■ **Figure 1** A visual example of the calculation of  $\text{SLACK}_i$  used in Algorithm 1 (MINSLACK). The example constraints  $\mathcal{C} \implies \{(3, 4), (5, 10), (15, 20)\}$  are read as, e.g.,  $(3, 4) \implies$  “at most three withdrawals over the next four consecutive time steps.” In the diagrammed example, the blue vertical lines represent the timestamps of processed withdrawals. With  $\text{SLACK}_2 = 1$ , the MINSLACK algorithm can process at most one withdrawal during the current period while still conforming to the constraints.

about 16,000 validators or about 5.5 days at peak due to Celsius bankruptcy proceedings.<sup>8</sup> However, there were about 900k total validators during this period, so processing these withdrawal requests immediately would not have violated the consistency constraints defined by the “weak-subjectivity period” [1]. With this motivation, we present a formal treatment of MINSLACK.

▶ **Theorem 1.** *Given any sequence of withdrawal requests  $R(\cdot)$ , let  $P(\cdot)$  be the processed withdrawal requests and  $\bar{P}(\cdot)$  be the resulting total amount withdrawn in each period by Algorithm 1. Then:*

1. **Feasibility:**  $P(\cdot)$  is feasible with respect to the protocol constraints.
2. **Optimality:** For any other feasible withdrawal decisions with total withdrawn in each period given by  $\bar{P}'(\cdot)$ , it must be the case that:

$$\forall t \geq 1 : \sum_{\tau=1}^t \bar{P}'(\tau) \leq \sum_{\tau=1}^t \bar{P}(\tau). \quad (1)$$

**Proof.** To show that the withdrawal resulting from MINSLACK is feasible, observe that in each period, the withdrawal amount is less than  $\min\{\text{SLACK}_i : 1 \leq i \leq k\}$  so it necessarily satisfies all of the constraints. Since each withdrawal satisfies the constraints given the history, applying the algorithm always results in a history that is feasible by construction.

For optimality, consider for the sake of contradiction that there exists a feasible  $\bar{P}'(\cdot)$  that violates (1). Let  $t$  be the earliest time such that:

$$\sum_{\tau=1}^t \bar{P}'(\tau) > \sum_{\tau=1}^t \bar{P}(\tau).$$

<sup>8</sup> See <https://www.validatorqueue.com/> for historical data about the withdrawal queue.



Since  $t$  is the earliest time to violate condition (1), we must have that for all  $t' < t$ ,

$$\forall t' < t: \sum_{\tau=1}^{t'} \bar{P}'(\tau) \leq \sum_{\tau=1}^{t'} \bar{P}(\tau). \quad (2)$$

Analogous to the algorithm, let us term  $\text{SLACK}'_i(\cdot)$  as the maximum withdrawable amount in a given period given process  $P'(\cdot)$ , with  $\text{MINSLACK}'_i(\cdot)$  defined as the smallest constraint  $i \in 1, \dots, k$ . Note that by feasibility, we must have the following:

$$\bar{P}'(t) \leq \text{MINSLACK}'_i(\cdot).$$

Conversely, we know that by construction (see Algorithm 1),

$$\bar{P}(t) = \text{MINSLACK}(t).$$

For the contradiction, it is sufficient to show that

$$\text{MINSLACK}'_i(t) - \text{MINSLACK}(t) \leq \sum_{\tau=1}^{t-1} \bar{P}(\tau) - \sum_{\tau=1}^{t-1} \bar{P}'(\tau). \quad (3)$$

In other words, we must show that the additional slack available at time  $t$  under  $P'$  relative to  $P$  is, at most, the difference between the amount withdrawn up to time  $t-1$  under  $P$  than  $P'$ . Feasibility of the withdrawals under  $P'$  then contradicts the claim that  $\sum_{\tau=1}^{t'} \bar{P}'(\tau) \leq \sum_{\tau=1}^{t'} \bar{P}(\tau)$ . To see (3) it is sufficient to show that for each  $1 \leq k$ :

$$\text{SLACK}'_i(t) - \text{SLACK}(t) \leq \sum_{\tau=1}^{t-1} \bar{P}(\tau) - \sum_{\tau=1}^{t-1} \bar{P}'(\tau). \quad (4)$$

The left-hand side of (4) can be rewritten as:

$$\begin{aligned} \text{SLACK}'_i(t) - \text{SLACK}(t) &= \delta_i \bar{S}'(t - T_i) - \sum_{\tau=t-T_i+1}^{t-1} \bar{P}'(\tau) - \left( \delta_i \bar{S}(t - T_i) - \sum_{\tau=t-T_i+1}^{t-1} \bar{P}(\tau) \right) \\ &\leq \left( \sum_{\tau=1}^{t-T_i} \bar{P}(\tau) + \sum_{\tau=t-T_i+1}^{t-1} \bar{P}(\tau) \right) - \left( \sum_{\tau=1}^{t-T_i} \bar{P}'(\tau) - \sum_{\tau=t-T_i+1}^{t-1} \bar{P}'(\tau) \right) \\ &= \sum_{\tau=1}^{t-1} \bar{P}(\tau) - \sum_{\tau=1}^{t-1} \bar{P}'(\tau). \end{aligned}$$

where the penultimate inequality follows since  $\delta_i \in [0, 1]$ , and we have (2).  $\blacktriangleleft$

Thus,  $\text{MINSLACK}$  is optimal for the common value setting. Still, in reality, stakers may have disparate values for accessing their stake, motivating the need to explore how a withdrawal mechanism could account for heterogeneous values.

## 5 Heterogeneous Values and Paying for Priority

While Theorem 1 shows that Algorithm 1 provides an optimal solution for the case when all stakers have a homogeneous value for withdrawing, in reality, they may have different values for getting access to their staked assets. A staking pool, for example, might be withdrawing some validators gradually to rotate the cryptographic keys used for participating in consensus.

In this case, the pool has a relatively low value for their withdrawal because the underlying reason to withdraw is not highly time-sensitive. On the other hand, a hedge fund trying to withdraw staked capital in time to meet a margin call to avoid a forced liquidation may have an extremely high value for the liquidity from the withdrawal processing.

Each validator looking to exit has a delay cost per unit time  $c$ . The net payoff of a validator of type  $c$  whose withdrawal occurs after a delay  $\Delta$  for a price (bid) of  $b$  is:<sup>9</sup>

$$U(\Delta, b, c) = -c\Delta - b.$$

In other words, their utility is linear in time according to their per-period disutility of waiting, less the amount they pay. We consider this canonical linear form for simplicity. More generally, one can consider other forms for the utility, including the time-varying disutility of waiting, see, e.g., [5].

As described below, the efficient mechanism will be more complicated: efficiency requires agents to express their disutility of waiting in the mechanism and managing agents' incentives involve payments. As in the previous section, we will consider the planner's objective to be efficiency, which is defined formally below.

► **Observation 2.** *When values are heterogeneous, Algorithm 1, MINSLACK, may not be efficient.*

Recall that in every period, MINSLACK greedily processes as many withdrawal requests as possible, given the constraints. However, there are unknown future withdrawal requests at the time of processing. With heterogeneous values, it is possible that highly time-sensitive stakers with a high disutility of waiting may arrive in future periods. Suppose the current withdrawal requests have very low time sensitivity (i.e., very low  $c$ ). In that case, the optimal behavior could be to withhold processing withdrawals in this period and reserve this capacity for the future. Intuitively, an efficient withdrawal mechanism must balance between processing withdrawals now while reserving some slack for hypothetical future withdrawals.

## 5.1 Efficient withdrawals under heterogeneity

This section describes a withdrawal algorithm based on the Vickrey-Clarke-Groves (VCG) mechanism. VCG in such dynamic settings is not novel – see [33] or [26]; it generalizes the second-price sealed-bid auction in static settings and has two desirable properties, namely, (i) it is incentive compatible for each agent to report their cost,  $c$ , and (ii) the mechanism is constrained-efficient.

As is standard in mechanism design, we first describe the efficient allocation rule, i.e., the optimal rule for a planner, in a setting where *the planner observes the delay costs of stakers as they arrive*. Then, we describe payment rules that make it *incentive compatible* for stakers to report their values truthfully.

Since this is a dynamic setting, as alluded to above, the mechanism must have a forecast of future arrivals to decide whether to process withdrawals or to reserve withdrawal slots for future arrivals. In this section, therefore, we assume that there is a known stochastic process behind the withdrawals. The number of withdrawal requests in each period is randomly distributed (for example, this may be the Poisson distribution with known parameter  $\lambda$ ). Each withdrawal request has a type that is an i.i.d. draw according to a known probability distribution on  $\mathbb{R}_+$ .

---

<sup>9</sup> This is a standard model in the context of transaction fees, where users face a similar trade-off between paying for inclusion and suffering a delay – see, e.g., [24].

### 5.1.1 The Efficient Allocation Rule

For now, suppose each agent truthfully states, at the time of joining the waiting list, their private cost,  $c$ . Each element of the waiting list is now a 4-tuple  $(v, s, t', c)$ . Modulo this change, however, the system can be described just as in Section 3.

Given that some set of withdrawal requests  $P(t)$  is processed in period  $t$  from a waiting list of  $W(t)$ , the system collects a penalty (net disutility) of:

$$\text{Penalty} = \sum_{(v,s,t',c) \in W(t) \setminus P(t)} sc.$$

In other words, the planner in period  $t$  collects a penalty equal to the disutility cost of every staker in the waiting list whose withdrawal is not processed. As before, the planner faces some constraints  $\mathcal{C}$  on exits. The system aims to minimize expected discounted penalties over feasible exit plans, where  $\rho \in [0, 1]$  is the planner's discount rate.

This is a dynamic program where the state of the problem at the beginning of period  $t$  is  $(S(t), W(t), H(t-1))$ . We can recursively define the value function of the planner as follows:

$$\begin{aligned} V(S(t), W(t), H(t-1)) \equiv & \quad (5) \\ & \min_{P(t)} \left( \sum_{(v,s,t',c) \in W(t) \setminus P(t)} sc + \delta \mathbb{E}[V(S(t+1), W(t) \setminus P(t) \right. \\ & \quad \left. \cup R(t+1), H(t) \cup P(t))] \right), \\ & \text{s.t. } P(t) \subseteq W(t), \\ & \quad P(t) \text{ feasible wrt } \mathcal{C}. \end{aligned}$$

Here, expectations are taken over the next period withdrawal requests  $R(t+1)$ : both the number of withdrawal requests and the corresponding waiting disutility is unknown at period  $t$ .

This framing is an infinite horizon Markov Decision Problem (MDP). Given the previous history, there is a maximum number of feasible withdrawals in every period. For any withdrawal processed from the waiting list, it is intuitive that the planner will remove the ones with the highest disutility of waiting first. However, as described above, the marginal value of holding onto a withdrawal slot can exceed the penalty of making a current staker on the list wait an extra period. Of course, the precise details depend on the arrival process and the system's current state. The algorithm is described in Program 5.

#### ■ Algorithm 2 OPTIMAL.

- 
- 1: ... {same as MINSLACK}
  - 2: **for** each period  $t \geq 1$  **do**
  - 3:    $W(t) \leftarrow W(t-1) \setminus P(t-1) \cup R(t)$ .
  - 4:    $P(t) \leftarrow$  Solution of Program 5
  - 5:    $\bar{P}(t) \leftarrow$  Total withdrawn in  $P(t)$
  - 6:    $H(t+1) \leftarrow H(t) \cup P(t)$
  - 7:   **Update:**  $S(v,t)$  based on  $P(t)$ ,  $E(t)$ .
  - 8: **end for**
-

## 20:10 Optimizing Exit Queues for Proof-Of-Stake Blockchains

OPTIMAL is nearly identical to MINSLACK; it only replaces the process for calculating the set of withdrawals to process in a current period,  $P(t)$  (shown in brown text in Algorithm 2). The optimization problem in Program 5 must be solved to determine the policy of how many withdrawals to process at each period.

### ■ Algorithm 3 PRIO-MINSLACK.

---

```
1: ... {same as MINSLACK}
2: Sort  $W(t)$  in decreasing order of waiting disutility.
3: for each constraint  $i \leq k$  do
4:    $SLACK_i \leftarrow \delta_i \bar{S}(t - T_i) - \sum_{\tau=t-T_i+1}^{t-1} \bar{P}(\tau)$ .
5: end for
6:  $MINSLACK \leftarrow \min\{SLACK_i : 1 \leq i \leq k\}$ .
7: ... {same as MINSLACK}
```

---

Another candidate withdrawal algorithm, which we refer to as PRIO-MINSLACK, modifies MINSLACK to process withdrawals in order of priority fees. Algorithm 3 represents this one-line change in blue text.

### 5.1.2 Pricing Rule

So far, we have described the problem as an optimization problem where the planner *knows* the disutility from waiting suffered by the stakers in the queue. These are private, and there must be an incentive for stakers to report truthfully. Achieving this is straightforward (albeit computationally inefficient): every staker withdrawn in a period  $t$  should pay the expected delay costs imposed on the system by their presence. Existing theorems (see [33], [5]) show that such a pricing rule results in a Bayes-Nash equilibrium, where each buyer reports their values truthfully.

### 5.1.3 Optimal policy

If new withdrawal arrival and value distributions are known, we can calculate the optimal withdrawal policy by solving the resulting MDP associated with Program 5.

**A tractable instantiation.** Consider the withdrawal problem with a single constraint of  $(t_0, \bar{S}\delta_0) = (5, 5)$  (no more than five withdrawals are allowed over five time periods).<sup>10</sup> Let the number of new withdrawals per period be distributed as  $Y \sim \{0, 1, 5\}$  *w.p.*  $\{0.5, 0.4, 0.1\}$  and the value of these distributions be distributed as  $X \sim \{1, 10\}$

*w.p.*  $\{0.9, 0.1\}$ . We need only two values to represent the state of pending withdrawals,  $W(t)$ . Let  $w_\ell$  and  $w_h$  denote the number of pending “low” ( $c = 1$ ) and “high” ( $c = 10$ ) withdrawals, respectively. Further, let  $h_{-1}, h_{-2}, h_{-3}, h_{-4}$  denote the history of withdrawals processed (called  $H(t-1)$  above) in each of the last four periods (with a  $(5, 5)$  constraint, this is the extent of the history that we must consider when deciding what withdrawals to process in this period). This leads the definition of each state

$$s = [w_\ell, w_h, h_{-1}, h_{-2}, h_{-3}, h_{-4}] \in S.$$

---

<sup>10</sup>For our numerical exercises, for simplicity, we model the constraints as corresponding to an absolute number of validators that can withdraw over some window of periods.

■ **Table 1** Performance over 10,000 simulations for OPTIMAL and PRIO-MINSLACK under different configurations of arrival distributions ( $Y$ ), value distributions ( $X$ ), and discount factors. The performance metric is the discounted value of the rewards starting in the initial state  $[0, 0, 0, 0, 0, 0]$ ; higher values (smaller disutility) are better. **Top three pairs:** *varying discount factors*. We use  $n = 225, 350, 700$  for simulation steps for the discount factors of  $\gamma = 0.85, 0.90, 0.95$  respectively (each selected so that the end of the trial has a weighting of  $\approx 10^{-16}$ ). **Middle three pairs:** *varying the value distribution*. **Bottom three pairs:** *varying the arrival distribution*.

Algorithm	Arrival dist.	Value dist.	Discount	Performance
OPTIMAL	$Y \sim [0, 1, 5]$ <i>w.p.</i> $[0.5, 0.4, 0.1]$	$X \sim [1, 10]$ <i>w.p.</i> $[0.9, 0.1]$	0.85	-2.374
PRIO-MINSLACK				-2.413
OPTIMAL			0.9	-2.933
PRIO-MINSLACK				-2.982
OPTIMAL				0.95
PRIO-MINSLACK			-3.999	
OPTIMAL	$Y \sim [0, 1, 5]$ <i>w.p.</i> $[0.5, 0.4, 0.1]$	$X \sim [1, 5]$ <i>w.p.</i> $[0.9, 0.1]$	0.9	-2.428
PRIO-MINSLACK		-2.422		
OPTIMAL		$X \sim [1, 10]$ <i>w.p.</i> $[0.9, 0.1]$		-2.959
PRIO-MINSLACK		-3.005		
OPTIMAL		$X \sim [1, 20]$ <i>w.p.</i> $[0.9, 0.1]$		-3.902
PRIO-MINSLACK		-4.151		
OPTIMAL	$Y \sim [0, 1, 2]$ <i>w.p.</i> $[0.4, 0.4, 0.2]$	$X \sim [1, 10]$ <i>w.p.</i> $[0.9, 0.1]$	0.9	-1.637
PRIO-MINSLACK	-1.638			
OPTIMAL	$Y \sim [0, 1, 5]$ <i>w.p.</i> $[0.5, 0.4, 0.1]$			-2.925
PRIO-MINSLACK	-2.969			
OPTIMAL	$Y \sim [0, 1, 10]$ <i>w.p.</i> $[0.6, 0.35, 0.05]$			-3.610
PRIO-MINSLACK	-3.620			

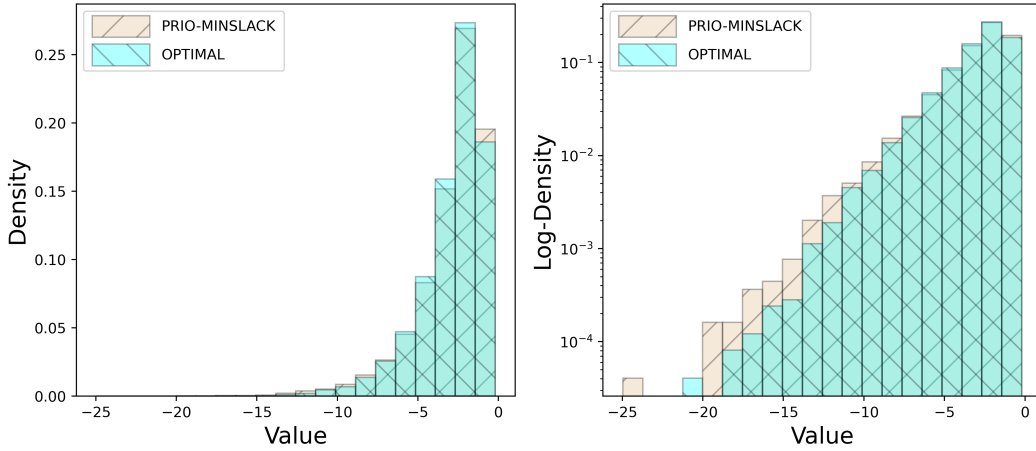
The action space in this MDP is  $A = \{0, 1, 2, 3, 4, 5\}$ , where the action  $a_i$  is legal if  $\sum h_j + a_i \leq 5$ . To limit the size of the action space, we only consider states where  $w_\ell, w_h < 10$ . Even with this extremely reduced setup, there are still  $|A| \times |S| \times |S| = 6 \cdot 15246^2 = 1394643096$  probabilities and rewards to encode. Nevertheless, this is feasible since the transition and reward matrices are sparse.

Using value iteration, we numerically solve for the optimal policy, which determines, “given a state, how many withdrawals should we process during this period.” We now compare the performance of OPTIMAL (Algorithm 2 (under an assumed discount factor of 0.9)) and PRIO-MINSLACK (Algorithm 3).<sup>11</sup> Recall that PRIO-MINSLACK is a much simpler heuristic, where it looks at the history and takes action  $a_i = 5 - \sum h_j$ . While this works well generally, there are situations where it is “overly aggressive” and can result in large disutilities. For example, consider the state.

$$[10, 0, 0, 0, 0, 0] \implies 10 \text{ pending lows, } 0 \text{ pending highs, empty history.}$$

<sup>11</sup>Table 1 considers other discount factors, arrival processes, and distributions of value.

## Overlaid Histograms of Policy Values



■ **Figure 2** Performance comparison of PRIO-MINSLACK and OPTIMAL over 10,000 samples calculating the discounted reward following each policy from the initial state  $s_0 = [0, 0, 0, 0, 0, 0]$  for 350 steps with a discount factor of 0.9. The density of each histogram shows the probability a given trial ends in that range of values. When examining the raw density, the performance seems comparable, but the Log-Density plot demonstrates that the long tail performance of PRIO-MINSLACK is significantly worse than OPTIMAL. Intuitively, PRIO-MINSLACK is more of a “gambler” – the algorithm takes big risks by greedily processing as fast as possible. These risks are rewarded in the median case but occasionally have large disutilities by burning the capacity on low-value withdrawals. See Table 1 for more numerical comparisons between the two algorithms under different parameterizations.

In this situation, PRIO-MINSLACK observes that it can process five low withdrawals immediately and does so ( $a_i = 5$ ). The optimal policy, however, chooses  $a_i = 3$  instead. By processing five withdrawals in a single period, PRIO-MINSLACK forces a state where no more withdrawals are possible for the following four periods. Using the available capacity, the mechanism runs the risk of a high withdrawal arriving and needing to wait, resulting in a large disutility. The optimal algorithm is “more cautious” by reserving two withdrawal slots for the future, protecting for the possibility that a high-value withdrawal comes in the following few periods. Figure 2 shows a performance comparison of PRIO-MINSLACK and OPTIMAL. There are ten states (i.e., configurations of the current queue and history of withdrawals) in which the action dictated by the optimal policy differs from PRIO-MINSLACK by two (e.g., optimal processes two fewer withdrawals than PRIO-MINSLACK) and 338 states in which the optimal action differs from PRIO-MINSLACK by one. Table 1 compares the performance of OPTIMAL and PRIO-MINSLACK under a few variations of (i) arrival distributions, (ii) value distributions, and (iii) discount factors from simulating the two policies.

## 5.2 Practical considerations for the heterogeneous value setting

The previous section outlines dynamic VCG, the optimal withdrawal mechanism given known stationary arrival and value distributions. In practice, the social planner may not know these distributions, and further, the expected number of withdrawals or urgency of the demand for liquidity could change over time. Beyond this, implementing dynamic VCG would require solving the dynamic program outlined in Program 5 and holding funds in escrow to execute the VCG payment rule – both of which seem possible on paper but present significant engineering challenges.

---

**Algorithm 4**  $\alpha$ -MINSLACK.
 

---

- 1: ... {same as MINSLACK}
  - 2: Sort  $W(t)$  in decreasing order of waiting disutility.
  - 3: **for** each constraint  $i \leq k$  **do**
  - 4:    $SLACK_i \leftarrow \delta_i \bar{S}(t - T_i) - \sum_{\tau=t-T_i+1}^{t-1} \bar{P}(\tau)$ .
  - 5: **end for**
  - 6:  $MINSLACK \leftarrow \min\{SLACK_i : 1 \leq i \leq k\}$ .
  - 7:  $P(t) \leftarrow$  Largest prefix of  $W(t)$  such that total withdrawn  $\leq \alpha \cdot MINSLACK$
  - 8: ... {same as MINSLACK}
- 

PRIO-MINSLACK is much simpler to implement, but may suffer under value heterogeneity because it is too eager to process withdrawals. The problem arises when PRIO-MINSLACK receives a burst of low-value withdrawals, in which case it consumes all the available capacity on the low-priority withdrawals and leaves potentially higher-value incoming withdrawals pending longer. These bursts induce a natural question: can we modify PRIO-MINSLACK to be slightly more conservative with its remaining capacity while preserving its simplicity? One solution is to modify PRIO-MINSLACK to consume only an  $\alpha \in (0, 1]$  proportion<sup>12</sup> of the SLACK available at each period.

Algorithm 4, which we call  $\alpha$ -MINSLACK, makes the one-line modification (shown in red) to PRIO-MINSLACK by scaling the amount of processed withdrawals by  $\alpha$ . By tuning  $\alpha$ , we can make  $\alpha$ -MINSLACK more or less aggressive in how much withdrawal capacity it uses now versus saving. At  $\alpha = 1$ , we reduce to the “maximally aggressive” version (PRIO-MINSLACK). In contrast, as  $\alpha \rightarrow 0$ ,  $\alpha$ -MINSLACK becomes increasingly conservative. The outcome here is that the slack continues to build up across the constraints, and you end up processing at the rate  $\alpha \cdot \delta_i/T_i$  per-unit time, where  $(\delta_i, T_i) \in \mathcal{C}$  s.t.,  $\delta_i/T_i = \min_j \delta_j/T_j$ . In other words, process at a constant rate proportional to the “most restrictive” constraint in  $\mathcal{C}$ . More moderate values, e.g.,  $\alpha = 0.5$ , present a more balanced version of  $\alpha$ -MINSLACK where the algorithm functions on the heuristic of “using half of the remaining slack at each period.”

While  $\alpha$ -MINSLACK is not necessarily optimal, it does eliminate the need for arrival distribution knowledge, which the optimal mechanisms rely on. Further, its simplicity makes it much more feasible for an actual production system. We justify this statement by numerically comparing the performance of various mechanisms under different withdrawal distributions. Note that we can expand the set of value distributions compared to the optimal analysis because we are no longer constructing the entire state space of the MDP.

Table 2 compares the performance of four different algorithms across a constant arrival distribution and under three different value distributions. These results demonstrate that the PRIO- and  $\alpha$ - versions of MINSLACK far outperform either the CONSTANT mechanism or regular MINSLACK (which serves as an FCFS-queue rather than a priority queue based on the value of the withdrawal). These results motivate that, under some distributions,  $\alpha$ -MINSLACK may be preferable to PRIO-MINSLACK. Further work could be done to study adaptive algorithms that aim to learn the optimal value of  $\alpha$  in an on-line fashion. Again, these heuristic rules for determining the withdrawal policy of the staking system are far more straightforward to construct and implement than the optimal versions described in Section 5.1.3.

---

<sup>12</sup>The interval is left-open because  $\alpha = 0$  implies no withdrawals are ever processed.

■ **Table 2** Numerical results for algorithm performance under a fixed withdrawal arrival distribution and three different value distributions. The performance metric measures the average disutility over the withdrawals and thus should be minimized (to maximize the utility). We calculate the mean disutility over ten independent samples of 10,000 steps each, with the first 1,000 steps of each sample discarded to allow the system to settle into a steady state. The single constraint was set as  $(\delta, T) = (5, 5)$ : “a maximum of five withdrawals may be processed over five periods.” **CONSTANT** processes one withdrawal per time step. **MINSLACK** and **PRIO-MINSLACK** follow the descriptions in Algorithms 1 and 3 respectively. For  $\alpha$ -**MINSLACK** (Algorithm 4), we use  $\alpha = 0.9$ , which produces the following mapping for calculating how much slack to consume in a given time slot  $[0, 1, 2, 3, 4, 5] \mapsto [0, 1, 2, 3, 4, 4]$ . We can describe this simply as: “If the slack is exactly 5, use only four (reserving one for a potentially high-value arrival). If the slack is less than 5, use it entirely.” The arrival distribution mimics occasional bursts of withdrawal requests while maintaining an expected value  $E[Y] = 0.9$ , less than the average capacity of one derived by the  $(5, 5)$  constraint. The withdrawal values were sampled from Uniform, Exponential, and Pareto distributions to demonstrate that under some conditions,  $\alpha$ -**MINSLACK** can outperform **PRIO-MINSLACK**. In all cases, **CONSTANT** and **MINSLACK** perform far worse than the **PRIO-** and  $\alpha$ - variants.

Algorithm	Arrival dist.	Value dist.	Performance
CONSTANT (1)	$Y \sim [0, 1, 5]$ <i>w.p.</i> $[0.5, 0.4, 0.1]$	$X \sim \text{Uniform}(0, 1)$	-5.768
MINSLACK			-5.464
PRIO-MINSLACK			-2.019
$\alpha$ -MINSLACK ( $\alpha = 0.9$ )			-2.002
CONSTANT (1)		$X \sim \text{Exp}(0.1)$	-12.249
MINSLACK			-11.648
PRIO-MINSLACK			-2.951
$\alpha$ -MINSLACK ( $\alpha = 0.9$ )			-2.986
CONSTANT (1)		$X \sim \text{Pareto}(2, 5)$	-114.913
MINSLACK			-109.354
PRIO-MINSLACK			-67.687
$\alpha$ -MINSLACK ( $\alpha = 0.9$ )			-63.070

## 6 Theory and Practice

**Why limit withdrawals in the first place? A thought experiment.** Assume that withdrawals are not limited. An attacker, Eve, accumulates 1/3 of the total stake in the PoS mechanism and invests heavily in networking infrastructure. Eve contacts Alice to inquire about buying a Tesla Cybertruck<sup>®</sup>. Alice, who is feeling both cyber- and cypherpunk enough to accept ETH for the transaction, sees `txn 0xcb` on Etherscan as finalized, giving her confidence to hand the (car) keys to Eve. From Alice’s perspective, the settlement assurance of 1/3 of all staked ETH (> 33 billion USD as of May 2024) is more than sufficient economic security for her transaction. However, using her networking prowess, Eve had tricked the honest validators into finalizing two conflicting blocks, one which included `txn 0xcb` and another that didn’t by partitioning the honest validators into two separate p2p groups and sharing conflicting attestations with each group. If withdrawals are not limited, she can fully withdraw her stake from both chains by the time honest validators reconnect (once Eve’s network-level attack ends) and try to slash her. Alice has no Tesla Cybertruck<sup>®</sup> nor the ETH originally sent in `txn 0xcb`.



In light of this, blockchains place limits on withdrawals. However, as described below, there is substantial variation in the limits placed and the withdrawal procedure, with little systematic study.

## 6.1 Accountable Safety and Limiting Withdrawals

We begin with the following simple observation.

► **Observation 3.** *The accountable safety of a finalized block **decreases** as time passes because the stake participating in the finalization of the block can withdraw from the system.*

This (rather counter-intuitive) fact means protocol designers must decide: “How quickly should validators be able to withdraw their stake from the system?” Let  $\mathcal{D}$  denote the “maximum-tolerable decay” in the accountable safety of a finalized block. For example, if  $\mathcal{D} = 1/6$ , then a finalized block may have accountable safety (in terms of proportion of the total stake that is slashable in case the transaction history changes) of  $1/3 - \mathcal{D} = 1/6$ . The security decay modifies the statement to, “any transaction in a finalized block will have accountable safety of at least  $1/6$  of all stake.” This remains incomplete because over a sufficiently long time horizon, with withdrawals enabled, more than  $\mathcal{D}$  stake may be removed from the system. Thus, we define an amount of time, denoted  $\delta$ , over which the stake withdrawn must not exceed  $\mathcal{D}$ . This period can serve multiple purposes. One such usage is the weak-subjectivity period [9], where the delay is an upper bound on the communication delay between all honest parties in the partially-synchronous protocol; this value is  $\mathcal{O}(\textit{weeks})$  to account for the natural overhead incurred when social coordination is required to come to consensus.<sup>13</sup> Other constraints might be over much shorter time horizons, e.g.,  $\mathcal{O}(\textit{minutes})$ , to ensure a bound on the rate at which the economic security of a block changes in short windows. Thus, the accountable safety of a Proof-of-Stake mechanism parameterized by  $\mathcal{D}$  and  $\delta$  is “any block finalized more than  $\delta$  time ago is immutable (only social consensus could reverse it), and any block finalized within the past  $\delta$  time has accountable safety of at least  $1/3 - \mathcal{D}$ .”

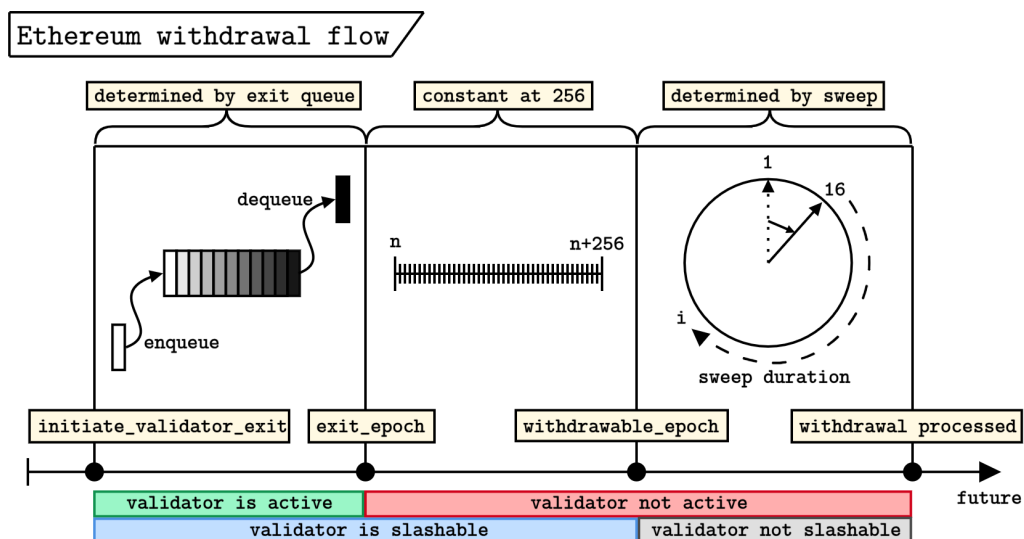
## 6.2 Ethereum

Withdrawals in Ethereum Proof-of-Stake were fully activated in the Shanghai/Capella Hardfork<sup>14</sup> on April 12, 2023. While the full withdrawal process is quite involved, we dig into the details to demonstrate how much engineering can shape the withdrawal mechanisms in use today. Figure 3 demonstrates the full flow of an Ethereum withdrawal, which is split into three distinct phases.

**Phase 1: Exit queue.** When an Ethereum validator wants to withdraw their 32 ETH from the consensus mechanism, they trigger a “Voluntary Exit” [4]. This process sets the validator’s `exit_epoch` based on the rate-limited first-come-first-served exit queue; during each epoch, at most  $\min(4, \lfloor \# \text{ validators} / 2^{16} \rfloor)$  are processed [4] (though this was changed in EIP-7514 to cap the churn limit at 8 validators per-epoch [29], making the new function  $\max(8, \min(4, \lfloor \# \text{ validators} / 2^{16} \rfloor))$ ). The `CHURN_LIMIT_QUOTIENT = 216` was

<sup>13</sup>By ignoring any blocks published prior to the weak-subjectivity checkpoint, validators can also eliminate the risk of long-range attacks (in practice, validators treat their latest finalized block as a ‘genesis’ or irreversible block by simply rejecting any block that conflicts with it).

<sup>14</sup><https://ethereum.org/en/history/#shapella>



■ **Figure 3** The withdrawal flow for Ethereum validators. Each phase has differing lengths and validator properties. The **top row** of tan labels demonstrate what determines the length of each phase. The **middle row** of tan labels annotate the timeline of events as described in the [15]. The **bottom row** of colored labels indicate the activity and slashability of the validator over time.

selected<sup>15</sup> according to the rough heuristic that it should take approximately one month for 10% of the stake to exit (or equivalently, about 100 days for 33% of the stake to exit) [13]. For the entire time a validator is in the exit queue (this phase), they are both “active” (meaning they must continue performing their consensus duties) and “slashable” (meaning their stake is still accountable for their behavior). Keeping the validator active while in the exit queue minimizes the economic cost of a very long exit queue because they continue earning rewards [11].

**Phase 2: Withdrawability delay.** Once the validator’s `exit_epoch` has passed, they incur a constant delay of 256 epochs (27 hours) before their `withdrawable_epoch` [4]. This fixed delay is a significant safety buffer to provide ample time for the protocol to include any slashing proof on chain. During this time, the validator is no longer active (and thus not earning any rewards), but they remain slashable (to avoid committing a slashing violation immediately before the withdrawal). The enforcement of this delay ensures that, even if the exit queue is empty, there is a period where the validator’s stake is still accountable for their actions.

**Phase 3: Validator sweep.** Once past the validator’s `withdrawable_epoch`, the function `is_fully_withdrawable_validator` returns `true` indicating that the withdrawal delay has passed and the validator is no longer slashable [15]. The last delay comes from the amount of time it takes for the actual withdrawal requests to send the ETH to the corresponding withdrawal address. All withdrawals are processed by looping through the validator set in order of validator index. This validator “sweep” can only process 16 withdrawals per block,

<sup>15</sup> Powers of two common for specification constants due to their compact binary representation.

corresponding to 8.8 days to iterate through the entire validator set (thus a 4.4-day additional delay on average).<sup>16</sup> This 8.8-day delay is present regardless of the length of the exit queues because “full” withdrawals (where a validator wants to leave the consensus layer altogether) are inter-mixed with “partial” withdrawals (where a small amount of validator rewards are involuntarily swept from each validator). While the original specification implemented the queues directly into the protocol, [36] changes this only to store the validator index and perform the sweep by mixing partial and full withdrawals.

This withdrawal mechanism is quite complex; the minimum time to exit the system is 27 hours. Due to the validator sweep, if the validator doesn’t strategically time their exit request, the withdrawal will take over 5.5 days on average to fully clear, even if the exit queue is empty. This complexity highlights how engineering decisions can inform the exit queue mechanism design. Beyond Ethereum, there are many additional staking systems, though their withdrawal mechanisms are much simpler and thus presented in Table 3 & Table 4.

### 6.3 Other Proof-of-Stake Blockchains

Table 3 compares several other blockchain protocols and how they handle withdrawals. Ethereum is the only protocol that implements a dynamic queue, and in this regard, Ethereum takes on additional complexity to improve the efficiency of the withdrawal mechanism. Cosmos, Polygon, and Polkadot each implement the simple, fixed-duration withdrawal mechanism with delays of 21, 2, and 28 days, respectively. This mechanism is simple and easy to reason about. Still, it is much less efficient because each withdrawal takes the maximal amount of time regardless of the history of the mechanism [6, 35, 34]. Solana, Cardano, and Avalanche do not have in-protocol slashing, so staking serves only as an anti-Sybil mechanism in their systems; the stake can exit the system without a rate-limiting step and not change their security model [39, 16, 3].

### 6.4 Other applications of staking

Beyond other blockchains, some applications have implemented staking and slashing mechanisms at the application layer of Ethereum. Table 4 performs the same high-level analysis of two such mechanisms.

EigenLayer and Chainlink use stake for slightly different purposes than the chains outlined in Table 3. EigenLayer creates a platform for buying and selling “economic security”; services built on EigenLayer (called “Actively Validated Services” or “AVSs”) purchase this security by incentivizing capital to delegate to an operator running their service. Because EigenLayer encumbers capital with additional slashing conditions, it also enforces a protocol-wide escrow period for stake removal. It is worth noting that the Ethereum withdrawal period can occur concurrently with the EigenLayer escrow period [22]. Further, services buying security from EigenLayer can impose further constraints on the capital allocated to their system. Chainlink, on the other hand, uses stake to provide security for the data feeds supplied by their oracle network [7]. This stake may be slashed for “less objective” faults (e.g., slashing for being offline and not providing a price feed), which was recently dubbed “inter-subjective slashing” in [21] and may grow to play a significant role in the future designs of slashing protocols.

---

<sup>16</sup><https://www.validatorqueue.com/>

## 20:18 Optimizing Exit Queues for Proof-Of-Stake Blockchains

■ **Table 3** Comparing staking and withdrawal mechanisms across L1 protocols and sidechains.

Protocol	Staking purpose	Withdrawal mechanism	One-line analysis
<i>Ethereum</i> [4]	Consensus safety	Rate-limited FCFS queue with minimum duration.	Aims to be fast in the average case, but partial withdrawals induce high-variance delay.
<i>Cosmos</i> [19]	Consensus safety	Fixed 21-day unbonding period.	Simple but inefficient.
<i>Solana</i> [39]	Sybil resistance	All deactivations happen at epoch boundaries. A maximum of 25% of stake can deactivate at any given epoch boundary.	With no slashing, stake does not provide accountable safety to the protocol. Limiting withdrawals ensures the entire stake cannot exit in a single epoch.
<i>Cardano</i> [16]	Sybil resistance	Immediate withdrawals.	With no slashing, stake does not provide accountable safety to the protocol. Withdrawals are immediately processed.
<i>Polygon</i> [35]	Consensus safety	Fixed $\approx 40$ hour unbonding period.	Simple but inefficient. It benefits from the fact that, as a sidechain, state updates are posted to Ethereum and are thus immutable – allowing for a relatively shorter fixed duration.
<i>Polkadot</i> [34]	Consensus safety	Fixed 28-day unbonding period.	Simple but inefficient.
<i>Avalanche</i> [3]	Sybil resistance	Validators dictate the duration of their staking before becoming active. The minimum duration is two weeks. After time has elapsed, the stake is immediately withdrawn.	With no slashing, stake does not provide accountable safety to the protocol. Withdrawals are immediately processed.

■ **Table 4** Comparing staking and withdrawal mechanisms between EigenLayer and Chainlink, two app-layer protocols with slashing.

Protocol	Staking purpose	Withdrawal mechanism	One-line analysis
<i>EigenLayer</i> [22]	Economic security guarantees	Fixed 7-day escrow period for all ETH-denominated withdrawals. Staked EIGEN has a fixed 24-day escrow period.	Withdrawals need to be limited because EigenLayer introduces new slashing conditions. Native restaked ETH may be withdrawn from the beacon chain during the EigenLayer escrow period. Each AVS could add its rate limiting in addition to the system-wide minimums.
<i>Chainlink</i> [17]	Oracle safety	Fixed 28-day cool-down period before LINK is claimable.	Staking provides safety and availability conditions for data feeds. Withdrawals are rate-limited to ensure slashing has time to take place.

## 6.5 Liquid staking & restaking tokens

Liquid staking tokens (LSTs) make a design trade-off when choosing how much of the capital in their system to deploy into consensus mechanisms. If they deploy too much of it, the withdrawals will be rate-limited by the underlying protocol, leading to a more capital-efficient protocol at the cost of a worse UX (slower withdrawals). Keeping some liquidity available for immediate redemption improves the UX, but any capital in that state is not cash-flowing. LSTs are fully collateralized and thus do not face insolvency risk, but holders face the duration risk of holding the LST for however long the withdrawal takes. Liquid restaking tokens (LRTs) have a more complex design space, where they must balance withdrawals against various underlying protocols and services. Their withdrawal mechanisms are plagued by the nature of various protocol rewards denominated in different tokens and emissions rates. [23] explores some design trade-offs, including a market for withdrawals. Overall, this design space is extensive and out-of-scope for the modeling of this paper, but it presents an exciting avenue for future research.

## 7 Conclusion

System designers of staking and restaking protocols face a fundamental trade-off between the security and utility. Based on the mechanisms we surveyed in Section 6, the mechanisms currently in production maximally flexible given the rigidity they claim to require. In other

words, nobody seems to be on the production-possibilities frontier of egress mechanisms in practice; we acknowledge that the practical engineering constraints, e.g., as described in the design of Ethereum’s withdrawal mechanism in Section 6.2, may play a significant role in the decision making of existing protocols.

By formalizing this trade-off as a constrained optimization problem over mechanisms, we aim to improve the state of withdrawal systems more broadly. For blockchain designers, we distill our results into three pieces of advice. First, suppose your consistency constraints are over a longer time horizon than a single epoch. In that case, a queue with dynamic capacity can significantly reduce average wait times without sacrificing security – MINSLACK (Algorithm 1) is a simple example of maximally processing the rate of withdrawals given a set of constraints. Second, if you believe that participants in the system may have heterogeneous disutility from waiting in the exit queue, their welfare would be improved by implementing a priority queue – PRIO-MINSLACK (Algorithm 3) can quickly decrease the overall disutility. Third, if you think that the time-sensitivity or arrival process of future withdrawal requests is particularly fat-tailed, be sure to reserve some capacity in the system to allow the processing of highly time-sensitive withdrawals during periods of congestion –  $\alpha$ -MINSLACK (Algorithm 4) is an example of this reservation.

We point to a few intriguing directions in terms of future work. Firstly, several empirical questions have been raised by this study. Assessing the actual staker surplus lost from sub-optimal queue designs would be helpful. The protocol may care about this staker surplus because reducing the staker disutility may lessen the emissions needed to incentivize token holders to stake in the first place. Further study on the heterogeneity in time preferences among stakers would help determine whether pay-for-priority systems are worth considering. Lastly, validator utility functions that are non-linear (e.g., a validator who needs their withdrawal within the next week but doesn’t care when) may lead to different design considerations and optimal withdrawal mechanisms.

On the theoretical side, note that some of the pay-for-priority systems we have proposed serve as benchmarks and are unlikely to be implementable in practice (e.g., the dynamic programming-based efficient allocation in Algorithm 2, which is both computationally difficult and requires knowledge of the distribution of withdrawal requests). Other mechanisms (e.g., PRIO-MINSLACK, Algorithm 3) are feasible as a pay-your-bid mechanism – reminiscent of the Bitcoin (and Ethereum before EIP-1559) transaction-fee mechanisms. Similar concerns faced in those contexts, users having to choose an appropriate bid, may apply in withdrawal mechanisms too. The natural question is whether designs with better user experience, analogous to EIP-1559, exist in this setting.

---

## References

- 1 Aditya Ansgaonkar. Weak subjectivity in ethereum 2.0, 2020. URL: <https://notes.ethereum.org/@adiasg/weak-subjectvity-eth2>.
- 2 Aditya Asgaonkar, Francesco D’Amato, Roberto Saltini, Luca Zanolini, and Chenyi Zhang. A confirmation rule for the ethereum consensus protocol. *arXiv preprint*, 2024. arXiv: 2405.00549.
- 3 Avalanche-Documentation. How to stake on avalanche, 2024. URL: <https://docs.avax.network/nodes/validate/how-to-stake>.
- 4 Beacon-Chain-Specifications. Phase 0 – the beacon chain, 2020. URL: <https://github.com/ethereum/consensus-specs/blob/dev/specs/phase0/beacon-chain.md>.
- 5 Dirk Bergemann and Juuso Välimäki. The dynamic pivot mechanism. *Econometrica*, 78(2):771–789, 2010.

- 6 Gavin Birch. The staking module, 2020. URL: <https://github.com/gavinly/CosmosParametersWiki/blob/master/Staking.md#1-unbondingtime>.
- 7 Lorenz Breidenbach, Christian Cachin, Benedict Chan, Alex Coventry, Steve Ellis, Ari Juels, Farinaz Koushanfar, Andrew Miller, Brendan Magauran, Daniel Moroz, et al. Chainlink 2.0: Next steps in the evolution of decentralized oracle networks, 2021.
- 8 Eric Budish, Andrew Lewis-Pye, and Tim Roughgarden. The economic limits of permissionless consensus. *arXiv e-prints*, 2024. arXiv:2405.09173.
- 9 Vitalik Buterin. Proof of stake: How i learned to love weak subjectivity. 2014, 2014. URL: <https://blog.ethereum.org/2014/11/25/proof-stake-learned-love-weak-subjectivity>.
- 10 Vitalik Buterin. Suggested average-case improvements to reduce capital costs of being a casper validator, 2018. URL: <https://ethresear.ch/t/suggested-average-case-improvements-to-reduce-capital-costs-of-being-a-casper-validator/3844>.
- 11 Vitalik Buterin. Rate-limiting entry/exits, not withdrawals, 2019. URL: <https://ethresear.ch/t/rate-limiting-entry-exits-not-withdrawals/4942>.
- 12 Vitalik Buterin. Weak subjectivity under the exit queue model, 2019. URL: <https://ethresear.ch/t/weak-subjectivity-under-the-exit-queue-model/5187>.
- 13 Vitalik Buterin. Vitalik’s annotated ethereum 2.0 spec, 2020. URL: <https://github.com/ethereum/annotated-spec/blob/master/phase0/beacon-chain.md>.
- 14 Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget. *arXiv preprint*, 2017. arXiv:1710.09437.
- 15 Capella-Specifications. Capella – the beacon chain, 2023. URL: <https://github.com/ethereum/consensus-specs/blob/dev/specs/capella/beacon-chain.md>.
- 16 Cardano-Undelegation. Cardano (ada) undelegation period, 2023. URL: <https://p2p.org/faq/en/articles/5253938-cardano-ada-undelegation-period>.
- 17 Chainlink-Docs. Introducing the chainlink staking platform: v0.2 upgrade and launch details, 2024. URL: [https://blog.chain.link/chainlink-staking-v0-2-overview/#unbonding\\_mechanism](https://blog.chain.link/chainlink-staking-v0-2-overview/#unbonding_mechanism).
- 18 Tarun Chitra. Competitive equilibria between staking and on-chain lending. *CryptoEconomic Systems (CES)*, 2021.
- 19 Cosmos-Staking-Module. x/staking, 2023. URL: <https://docs.cosmos.network/v0.47/build/modules/staking>.
- 20 Soubhik Deb, Robert Raynor, and Sreeram Kannan. Stakesure: Proof of stake mechanisms with strong cryptoeconomic safety, 2024. arXiv:2401.05797.
- 21 EigenLabs. Eigen: The universal intersubjective work token, 2024. URL: [https://github.com/Layr-Labs/whitepaper/blob/master/EIGEN\\_Token\\_Whitepaper.pdf](https://github.com/Layr-Labs/whitepaper/blob/master/EIGEN_Token_Whitepaper.pdf).
- 22 EigenLayer-Documentation. Escrow period (withdrawal delay), 2023. URL: <https://docs.eigenlayer.xyz/eigenlayer/restaking-guides/restaking-user-guide/#escrow-period-withdrawal-delay>.
- 23 Sam Hart and Max Einhorn. Building a liquid restaking token from first principles, 2024. URL: <https://timewave.computer/liquid-restaking-token>.
- 24 Gur Huberman, Jacob D Leshno, and Ciamac Moallemi. Monopoly without a monopolist: An economic analysis of the bitcoin payment system. *The Review of Economic Studies*, 88(6):3011–3040, 2021.
- 25 Oisín Kyne. Eip-7002: Execution layer triggerable exits, 2023. URL: <https://ethereum-magicians.org/t/eip-7002-execution-layer-triggerable-exits/14195/6>.
- 26 Ron Lavi and Noam Nisan. Competitive analysis of incentive compatible on-line auctions. In *Proceedings of the 2nd ACM Conference on Electronic Commerce*, pages 233–241, 2000.
- 27 Jacob D Leshno. Dynamic matching in overloaded waiting lists. *American Economic Review*, 112(12):3876–3910, 2022.
- 28 Andrew Lewis-Pye and Tim Roughgarden. Permissionless consensus, 2024. arXiv:2304.14701.
- 29 dApp Lion and Tim Bieko. Eip-7514: Add max epoch churn limit, 2023. URL: <https://eips.ethereum.org/EIPS/eip-7514>.

## 20:22 Optimizing Exit Queues for Proof-Of-Stake Blockchains

- 30 Joachim Neu, Ertem Nusret Tas, and David Tse. Short paper: Accountable safety implies finality. *Cryptology ePrint Archive*, 2023.
- 31 Noam Nisan. Serial monopoly on blockchains, 2023. [arXiv:2311.12731](https://arxiv.org/abs/2311.12731).
- 32 Mallesh Pai and Max Resnick. Dynamic transaction fee mechanism design. *arXiv preprint*, 2023.
- 33 David C Parkes and Satinder Singh. An mdp-based approach to online mechanism design. *Advances in neural information processing systems*, 16, 2003.
- 34 Polkadot-Validator-Guide. Run a validator (polkadot), 2024. URL: <https://wiki.polkadot.network/docs/maintain-guides-how-to-validate-polkadot>.
- 35 Polygon-Knowledge-Layer. How to delegate, 2023. URL: <https://docs.polygon.technology/pos/how-to/delegate/#unbond-from-a-validator>.
- 36 Potuz. Withdrawals without queues, 2022. URL: <https://github.com/ethereum/consensus-specs/pull/3068>.
- 37 Tim Roughgarden. Transaction fee mechanism design for the ethereum blockchain: An economic analysis of eip-1559. *arXiv preprint*, 2020. [arXiv:2012.00854](https://arxiv.org/abs/2012.00854).
- 38 Tim Roughgarden. Transaction fee mechanism design. *ACM SIGecom Exchanges*, 19(1):52–55, 2021.
- 39 Solana-Documentation. Delegation timing considerations, 2023. URL: <https://solana.com/staking#delegation-timing-considerations>.
- 40 Xuanming Su and Stefanos A Zenios. Recipient choice can address the efficiency-equity trade-off in kidney transplantation: A mechanism design model. *Management science*, 52(11):1647–1660, 2006.





# Searcher Competition in Block Building

Akaki Mamageishvili<sup>1</sup>  

Offchain Labs, Zurich, Switzerland

Christoph Schlegel 

Flashbots, George Town, Cayman Islands

Benny Sudakov  

ETH Zürich, Switzerland

---

## Abstract

---

We study the amount of maximal extractable value (MEV) captured by validators, as a function of searcher (or order flow provider) competition in blockchains with competitive block building markets such as Ethereum. We argue that the core is a suitable solution concept in this context that makes robust predictions that are independent of implementation details or specific mechanisms chosen. We characterize how much value validators extract in the core and quantify the surplus share of validators as a function of searcher competition. Searchers can obtain at most the marginal value increase of the winning block relative to the best block that can be built without their bundles. Dually this gives a lower bound on the value extracted by the validator. If arbitrages are easy to find and many searchers find similar bundles, the validator gets paid all value almost surely, while searchers can capture most value if there is little searcher competition per arbitrage. For the case of passive block-proposers we study, moreover, mechanisms that implement core allocations in dominant strategies and find that for submodular value, there is a unique dominant-strategy incentive compatible core-selecting mechanism that gives each searcher exactly their marginal value contribution to the winning block.

**2012 ACM Subject Classification** Theory of computation → Algorithmic game theory and mechanism design

**Keywords and phrases** MEV, Block Building, Searchers, Proposer Builder Separation, Core

**Digital Object Identifier** 10.4230/LIPIcs.AFT.2024.21

**Related Version** *Full Version*: <https://arxiv.org/abs/2407.07474>

## 1 Introduction

Blockchains that support smart contracts frequently run decentralized finance (DeFi) applications. This in turn gives rise to the phenomenon of miner/maximal extractable value, [5]: Blockchain protocols give *validators*, sometimes called proposers, the right to order transactions for a particular block. This effectively means that validators have a local monopoly to include or exclude transactions or to order transactions in a particular way, in order to generate value for themselves from this privileged position. However, since extracting value from transaction ordering in an optimal way is a difficult task, more specialized actors such as *block builders* and (arbitrage) *searchers* participate in the value extraction process in smart contract blockchains such as Ethereum. Block builders aggregate different arbitrage opportunities, liquidations or “sandwiches” in one block, using transactions from the public mempool, different private mempools, order flow auctions or other transaction sources together with their own transactions. Then, they bid against other builders, to get their block published in a canonical chain of blocks. The bid is paid to the current block proposer. Arbitrage opportunities are typically found by more specialized players, *searchers* and passed

---

<sup>1</sup> Corresponding author



to the builders. The resulting strategic interaction between searchers, builders and the validator lead to a value distribution of the arbitrage gains between these players. Empirical evidence shows that the majority of the total observable MEV is captured by validators.<sup>2</sup>

In this paper, we analyze the competition between searchers (more generally order-flow providers) in order to explain the value distribution between the proposer (or block builder) and the searchers. Different searchers may capture different value from different opportunities and may or may not find unique opportunities that are not found by competing searchers. This competition and specialization should in turn explain the value capture by different searchers in the MEV supply chain. Our model abstracts away from the intermediate layer of block building where block builders aggregate searcher bundles to blocks and compete in a bidding procedure. However, the ability for different combination of searchers or order flow providers to jointly generate different blocks is captured implicitly by the solution concept we focus on: We consider core allocation where different combinations of players can construct blocks of different values and distribute the value among themselves. The core constraints require that no coalition of players should be able to generate more value for themselves (produce a more valuable block) than they get paid in the realized allocation. Thus, while we generically talk about searchers, our model can for example also capture the case where some searchers act as block builders. The core is a natural solution concept in the context of block building and MEV capture as it captures the competition and possibility of collaboration of different players while abstracting away from any particular mechanisms that would intermediate between searchers, builders and proposers. Thus, it provides robust predictions on the set of plausible value distributions that can arise in any sufficiently competitive block building market. As a next step, one can then look into mechanism that implement particular core allocations through eliciting privately held information of searchers about the value they can generate from different blocks.

Our first result, shows that searchers can at most obtain their marginal contribution to the winning block (the difference in value between the best block that can be built with their transactions and without their transactions) and if value is submodular, then giving each searcher their marginal contribution is in the core.<sup>3</sup> A straightforward calculation shows that in a world with passive block producers, this particular core allocation coincides with the Vickrey-Clark-Growes (VCG) outcome ([11]) and hence can be implemented in dominant strategies (Corollary 5). On the other hand (Proposition 6), no other core-selecting mechanism is dominant-strategy incentive compatible. These result gives additional justification to further study the searcher-optimal point in the core: the value obtained by the validator in this point can be interpreted as the maximal extractable value he can obtain taking into account information rents captured by the searchers.

As a next step, we consider a stochastic model of searcher competition to study the value distribution between the searchers and the validator. Different (arbitrage or MEV) opportunities have a fixed probability of being found by a searcher. The more searchers find an opportunity, the more value can be extracted by the validator and the less value can be extracted by the searchers. We show that if the probability of finding an opportunity is bounded from below by  $p > \log(n)/n$ , where  $n$  is the number of searchers actively searching in the strategy, with high probability the validator captures all value in all core allocations.

---

<sup>2</sup> See <https://www.galaxy.com/insights/perspectives/distribution-of-mev-surplus/>.

<sup>3</sup> While transactions submitted to a builder might exhibit significant complementarities, submodularity can be argued to be a reasonable assumption in a sufficiently consolidated market. Order flow providers with complementary flow have an incentive to integrate to capture more value together. Thus, the realizable value might be submodular in the contributions of the different (consolidated) players.

On the other hand, if the probability of successfully searching is very low,  $p \in \Theta(1/n)$ , then with positive constant probability, searchers will be able to capture all value in the searcher-optimal core allocation.

## Related Literature

The phenomenon of miner extractable value has first been documented in [5]. Another early contribution on front-running in decentralized finance is [6]. MEV has been documented by a variety of public dashboards and data sets, see for example <https://mevboost.pics/>, <https://libmev.com/>. Recently, also the magnitude of non-atomic arbitrage has been empirically investigated, where searchers realize one lag of a trade on chain and one on a different domain, see e.g. [8] and <https://dune.com/flashbots/1o1>.

The block building market structure that has evolved in Ethereum since the change to proof of stake, has been the topic of several recent contributions: The dashboard <https://orderflow.art/> documents empirically the supply chain through which transaction requests land on chain. [1] studies whether MEV and Proof-of-Stake rewards capture leads to centralization, discussing both validator and builder roles in the market formation. [12] argue that in the current market structure searchers and builders have an incentive to vertically integrate. [4] argue that the builder is prone to centralization. [15] provide descriptive statistics on the level of decentralization on the builder landscape. [3] proposes a dynamic MEV sharing mechanism that the authors argue results into better decentralization and fair allocation.

[2] studies questions of implementation with active block producers. Our model is similar to theirs, but we focus on the question of implied value distribution. Our positive results for dominant-strategy incentive compatible core-selecting mechanisms for passive block builders complement their negative result for active block builders. [2] also study the role of searchers as intermediaries.

The notion of the core was first formally defined in [7]. For an overview of results around the core and submodularity see e.g. [9].

## 2 Model

There is a finite set  $\mathcal{S}$  of searchers that submit transactions for inclusion in the block. Similarly as in [2] we will usually identify searchers with (bundles of) transactions they have sent for inclusion. However, our model also allows for the interpretation that the same searcher (address) sends multiple bundles for inclusion. There is one validator (proposer), denoted by  $V$ . For each set of searchers  $A \subseteq \mathcal{S}$  there is a finite set of feasible blocks  $\mathcal{B}(A)$  that can be built from bundles of transactions submitted by searchers in  $A$ . A searcher  $i$  generates value  $v_i(B)$  from a block  $B$  and the validator generates value  $v_V(B)$  from block  $B$ . Our model can capture externalities (searcher  $i$ 's realized value may not only depend on her included transactions but also other transactions in the realized block) and active validators/block producers (we may have  $v_V(B) \neq 0$ ). We assume that utility is transferable and the final utility realized by searcher  $i$  if block  $B$  is realized and she makes a payment of  $p_i$  (e.g. to the validator) is  $v_i(B) - p_i$ .

Since utility is transferable, we can define a coalitional value function  $v : 2^{\mathcal{S} \cup \{V\}} \rightarrow \mathbb{R}_+$  by

$$v(S \cup V) := \max_{B \in \mathcal{B}(S)} \left( \sum_{i \in S} v_i(B) + v_V(B) \right),$$

## 21:4 Searcher Competition in Block Building

and

$$v(S) = 0 \text{ if } V \notin S,$$

i.e. in case the validator is part of the coalition, the coalitional value is the value of the total welfare maximizing block consisting of transactions from searchers in the coalition, and in case the validator is not part of the coalition, no value can be generated, as the validator is necessary to realize a block. It will be useful subsequently to introduce the short hand notation

$$\bar{v}(S) := v(S \cup V)$$

for  $S \subseteq \mathcal{S}$  to denote the value that searchers  $S$  can generate together with the validator. Observe that by construction, the (collective) value function is monotonic,

$$\bar{v}(A) \leq \bar{v}(B) \quad \text{for all } A \subseteq B \subseteq \mathcal{S};$$

if we receive more bundles to build a block that will increase welfare weakly, since we can always discard submitted bundles when building a block. Moreover, we make the following assumption on the (collective) value function:

**Submodularity.** Let  $A, B \subseteq \mathcal{S}$ . Then

$$\bar{v}(A) + \bar{v}(B) \geq \bar{v}(A \cup B) + \bar{v}(A \cap B),$$

Submodularity states that the value of a block we can build from transactions from searchers in  $A$  and  $B$ , is bounded by subtracting the value of a block we can build from transactions in both  $A$  and  $B$  from the sum of values we can achieve from building a block with transactions in  $A$  and a block with transactions in  $B$ .

Submodularity requires that there are not-too-strong complementarities between different submitted bundles. We can justify this assumption in two ways: first, it may be that complementarities are not strong and different MEV opportunities provide value that is mostly independent from other opportunities. Second, it may be that the complementarities are already absorbed by searchers, e.g. in the sense that searchers who provide complementary flow have an incentive to integrate their operations and send their flow together to extract more value. It is worth noting that our upper bound on searcher values holds also for non-submodular value functions, but may be loose in that case.

It is easy to show that for monotone value functions, submodularity is equivalent to requiring decreasing marginal value:

**Decreasing Marginal Value.** Let  $A \subseteq B \subseteq \mathcal{S}$  and  $a \in A$ . Then

$$\bar{v}(B) - \bar{v}(B \setminus \{a\}) \leq \bar{v}(A) - \bar{v}(A \setminus \{a\}).$$

A direct consequence of submodularity is the following lemma which will be useful subsequently:

► **Lemma 1.** Let  $A \subseteq B \subseteq \mathcal{S}$ . Then

$$\bar{v}(B) - \bar{v}(A) \geq \sum_{i \in B \setminus A} (\bar{v}(B) - \bar{v}(B \setminus \{i\})).$$

**Proof.** We prove the result by induction on  $N := |B \setminus A|$ . For  $N = 0$  the result holds trivially. Now suppose the result hold for  $N \geq 0$  and consider the case  $N + 1$ . Let  $j \in B \setminus A$ . By induction assumption

$$\bar{v}(B \setminus \{j\}) - \bar{v}(A) \geq \sum_{i \in B \setminus (A \cup \{j\})} (\bar{v}(B \setminus \{j\}) - \bar{v}(B \setminus \{i, j\})).$$

Adding  $\bar{v}(B) - \bar{v}(B \setminus \{j\})$  on both sides and using submodularity (which implies decreasing marginal value), we obtain

$$\begin{aligned} \bar{v}(B) - \bar{v}(A) &\geq \sum_{i \in B \setminus (A \cup \{j\})} (\bar{v}(B \setminus \{j\}) - \bar{v}(B \setminus \{i, j\})) + \bar{v}(B) - \bar{v}(B \setminus \{j\}) \\ &\geq \sum_{i \in B \setminus (A \cup \{j\})} (\bar{v}(B) - \bar{v}(B \setminus \{i\})). \end{aligned} \quad \blacktriangleleft$$

An **allocation** is a value distribution  $x : \mathcal{S} \cup V \rightarrow \mathbb{R}_+$  such that

$$\sum_{j \in \mathcal{S} \cup V} x_j \leq v(\mathcal{S} \cup V).$$

An allocation is in the **core** if the following inequality

$$\sum_{j \in C} x_j \geq v(C) \tag{1}$$

holds for any subset  $C \subseteq \mathcal{S} \cup V$  and all value is distributed:

$$\sum_{j \in V \cup \mathcal{S}} x_j = v(\mathcal{S} \cup V).$$

► **Remark 2.** As usual in the formulation of the core, the solution proposes a value allocation without specifying an explicit implementation through a block and payments between the searchers and the validator. Let  $B^* \in \mathcal{B}(\mathcal{S})$  be a welfare maximizing block, i.e.  $\sum_{i \in \mathcal{S} \cup V} v_i(B^*) = v(\mathcal{S} \cup V)$ . Then a core value allocation  $x$  can be implemented by realizing the block  $B^*$  and requiring that each individual searcher  $i$  makes a payment of  $p_i = v_i(B^*) - x_i$  to the validator.

Immediately from the requirement that without the validator no value can be realized, it follows that the validator getting all gains is in the core.

► **Observation 3.** *The core is always non-empty. The allocation where  $x_V = v(\mathcal{S} \cup V)$  and  $x_i = 0$  for each  $i \in \mathcal{S}$  is in the core.*

### 3 Analysis

Our first main result states each searcher can at most capture their marginal contribution to the realized block, and any allocation that gives each searcher at most their marginal contribution is in the core. In particular, this implies that there is a searcher optimal core allocation (giving each searcher exactly their marginal contribution to the realized block) and a validator optimal allocation (giving the validator all realized value).

► **Proposition 4.** *An allocation is in the core if and only*

$$0 \leq x_i \leq \bar{v}(\mathcal{S}) - \bar{v}(\mathcal{S} \setminus \{i\})$$

for each  $i \in \mathcal{S}$  and  $x_V = \bar{v}(\mathcal{S}) - \sum_{i \in \mathcal{S}} x_i$ .

## 21:6 Searcher Competition in Block Building

**Proof.** First we show that  $x_i \leq \bar{v}(\mathcal{S}) - \bar{v}(\mathcal{S} \setminus \{i\})$  for each  $i$ . Suppose not. In that case

$$x_V + \sum_{j \in \mathcal{S} \setminus \{i\}} x_j = \bar{v}(\mathcal{S}) - x_i < \bar{v}(\mathcal{S} \setminus \{i\}).$$

The previous inequality contradicts the core-stability of  $x$  which requires inequality (1) to hold for  $C = \mathcal{S} \setminus \{i\} \cup V$ . The lower bound on  $x_i$  is trivial. It remains to show that all allocations with  $0 \leq x_i \leq \bar{v}(\mathcal{S}) - \bar{v}(\mathcal{S} \setminus \{i\})$  are in the core. Let  $x$  be a vector satisfying these inequalities. Let  $A \subseteq \mathcal{S}$  and observe that

$$x_V + \sum_{j \in A} x_j = \bar{v}(\mathcal{S}) - \sum_{i \in \mathcal{S} \setminus A} x_i \geq \bar{v}(A),$$

where the last inequality follows from Lemma 1. Thus, the core inequalities (1) are satisfied for all coalitions  $A \cup V$  with  $A \subseteq \mathcal{S}$ . For coalitions without the validator the core inequalities are trivially satisfied, as all agents get non-negative value in  $x$ . ◀

### Implementation

In reality, the values that searchers obtain from different blocks is private information to them. However, in a world with passive block producers<sup>4</sup>, we can implement the extreme point in the core that gives maximal value to searchers in dominant strategies: Consider the case where the block producer is passive i.e.  $v_V(B) = 0$  for each block  $B$ . Observe that VCG-payments in this problem are defined by

$$p_i := \max_{B \in \mathcal{B}(\mathcal{S} \setminus \{i\})} \sum_{j \neq i} v_j(B) - \sum_{j \neq i} v_j(B^*) = \bar{v}(\mathcal{S} \setminus \{i\}) - \sum_{j \neq i} v_j(B^*),$$

for each  $i \in \mathcal{S}$  where  $B^*$  is the welfare-optimal block that can be produced. A straightforward calculation shows that,

$$v_i(B^*) - p_i = \sum_{j \in \mathcal{S}} v_j(B^*) - \bar{v}(\mathcal{S} \setminus \{i\}) = \bar{v}(\mathcal{S}) - \bar{v}(\mathcal{S} \setminus \{i\}),$$

i.e. the searcher-optimal core outcome coincides with the VCG outcome. We obtain the following corollary of Proposition 4:

► **Corollary 5.** *Under submodular value and with passive block-producers, the searcher-optimal core-outcome can be implemented in dominant strategies.*

On the other hand, it is straightforward to see that other selections from the core that not always select the VCG outcome are not dominant-strategy incentive compatible.

► **Proposition 6.** *Under submodular value and with passive block-producers, any core-selecting mechanism that is not always choosing the searcher-optimal outcome in the core is not dominant-strategy incentive compatible.*

---

<sup>4</sup> We know from [2] that with active block producers implementation of non-trivial solutions is not possible. This in turn resembles previous results from other context where negative results prevail if incomplete information in a two-sided markets is on both sides of the market [10, ]. In particular, for assignment games [14], which can be re-interpreted as the special case of our model where the validator has additively separable value, there is no mechanism that implements a core-allocation in dominant strategies, for any domain of valuations when there is at least one profile of valuations for which a core allocation that gives positive value to some searcher exists. For a proof of this “folk theorem” see e.g. [13].

**Proof.** Suppose for reported value functions  $(v_i)_{i \in \mathcal{S}}$ , a block  $B^*$  and payments  $(p_i)_{i \in \mathcal{S}}$  are chosen by the mechanism. By core-stability and Proposition 4, we have  $0 \leq v_i(B^*) - p_i \leq v(\mathcal{S}) - v(\mathcal{S} \setminus \{i\})$  for searcher  $i$ , or equivalently

$$v_i(B^*) \geq p_i \geq \bar{v}(\mathcal{S} \setminus \{i\}) - \sum_{j \neq i} v_j(B^*)$$

Suppose for the sake of contradiction that the last inequality is strict and  $i$  reports different values  $\tilde{v}_i$  with  $\tilde{v}_i(B) = v_i(B)$  for  $B$  blocks not including transactions by  $i$ , and  $\tilde{v}_i(B) \leq \tilde{v}_i(B^*)$  for blocks including transactions by  $i$  (so that  $B^*$  is still optimal) and

$$p_i > \tilde{v}_i(B^*) > \bar{v}(\mathcal{S} \setminus i) - \sum_{j \neq i} v_j(B^*).$$

Note that this change in value preserves the submodularity of the coalitional value function that the same block  $B^*$  is optimal and that the payment  $\tilde{p}_i$  for searcher  $i$  now satisfies

$$p_i > \tilde{v}_i(B^*) \geq \tilde{p}_i \geq \bar{v}(\mathcal{S} \setminus i) - \sum_{j \neq i} v_j(B^*),$$

which is strictly less. Therefore  $i$  gains from misreporting.  $\blacktriangleleft$

The corollary and previous proposition motivate to further study the searcher-optimal point in the core. The value that a (passive) builder/validator obtains in this point is the maximal extractable value taken into account information rents that searchers can capture. In the next sections, we study the particular case where the value of a block is derived from independent (MEV) opportunities for which different searchers compete. For that case, we derive results on when competition lets the core collapses to one point (in which the validator captures all value) and when lack of competition allows searchers to generate positive value in the searcher-optimal core allocation.

## 4 Independent Bundles and Competing Searchers

In this section, we consider the special case of additively separable value where the value of a block is the sum of values derived from the individual (bundles) of transactions. Moreover, we look at a scenario where multiple searchers may compete for the same (arbitrage) opportunities so that their submitted bundles possibly “clash” with bundles submitted by other searchers.<sup>5</sup> We denote opportunities by  $\mathcal{A}$  and now can identify a block by a matrix  $B = (B_{ij})_{i \in \mathcal{A}, j \in \mathcal{S}}$  where  $B_{ij} = 1$  if searcher  $j$ 's bundle competing for opportunity  $i$  is included,  $B_{ij} = 0$  if it is not included. We require that  $\sum_{j \in \mathcal{S}} B_{ij} \leq 1$  so that multiple clashing bundles cannot be included in the same block. We can then write searcher  $j$ 's value from block  $B$  as

$$v_j(B) = \sum_{i \in \mathcal{A}} v_{ij} B_{ij},$$

where  $v_{ij}$  is the value extracted by searcher  $j$  from opportunity  $i$ . We can add additional constraints such as a capacity constraint on the total number of bundles. In the unconstrained case, where all blocks are feasible, we have

$$v(\mathcal{S} \cup V) = \max_{B \in \mathcal{B}(\mathcal{S})} \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{S}} v_{ij} B_{ij} = \sum_{i \in \mathcal{A}} \max_{j \in \mathcal{S}} v_{ij}. \quad (2)$$

<sup>5</sup> This matches the reality of searching where often different searchers compete in the same strategy and find conflicting bundles among which the block builder chooses the most profitable one and includes it in the block while discarding the less profitable competing bundles. Around one third of submitted bundles to Ethereum block builders “clash”.

## 21:8 Searcher Competition in Block Building

To get an intuition how the core looks like in this case, we start this section with few examples and observations. Previously, in Observation 3 we had observed that giving all value to the validator is always in the core. In the opposite direction, if there is no searcher competition, i.e. no two searchers find the same opportunity, the validator can receive no value in the core.

► **Observation 7.** *There are examples of core allocations where the validator receives 0.*

**Proof.** Suppose that for each opportunity  $i \in \mathcal{A}$  there is at most one searcher submitting a bundle of positive value,  $v_{ij} > 0$  for at most one  $j$ . Then, the value allocation with  $x_j = \sum_{i:v_{ij}>0} v_{ij}$  for each searcher  $j$  and  $x_V = 0$  is in the core. ◀

Next, we give an example where the maximum payment to the validator is enforced in the core.

► **Observation 8.** *There are examples where the validator gets the full value of the winning block in any core allocation.*

**Proof.** Suppose for each opportunity  $i \in \mathcal{A}$  at least two searchers submit the same highest value bundle (or no searcher finds the opportunity). Then for each searcher  $j$  we have  $x_j = v(\mathcal{S}) - v(\mathcal{S} \setminus \{j\}) = 0$ . The claim follows from Proposition 4. ◀

For a vector of non-negative numbers  $X$ , let  $SH(X)$  denote the second highest coordinate of it. Then, let  $M$  denote the following sum:

$$M := \sum_{i \in \mathcal{A}} M_i := \sum_{i \in \mathcal{A}} SH((v_{ij})_{j \in \mathcal{S}}). \quad (3)$$

Additive value function as defined in (2) are submodular. Thus, we obtain the following special case of Proposition 4.

► **Corollary 9.** *The allocation in which the validator receives  $x_V = M$  and searcher  $j$  receives  $x_j = \sum_{i:j \in \arg \max_{j \in \mathcal{S}} v_{ij}} (v_{ij} - M_i)$  is in the core.*

This particular core allocation is the worst for the validator and the best for searchers and can be implemented, see Corollary 5, in dominant strategies by a generalized second price auction for bundles.

## Stochastic Model

In this section, we analyze the core when searchers find bundles with some probability and success is independent across searchers and opportunities. Let  $v_{ij}$  be a binary random variable, which is 1 with probability  $p$  and 0 with probability  $1 - p$ .<sup>6</sup> Thus,  $p$  measures how easy it is to find an arbitrage (bundle).

► **Proposition 10.** *Let  $n := |\mathcal{S}|$ . If  $p > \frac{2 \log n}{n}$  and  $m := |A| < n$ , then the validator receives the entire block value with high probability in any core allocation.*

---

<sup>6</sup> Generalizations of the subsequent results to non unit value that can be different for different opportunities are straightforward. The only assumption needed is that searchers conditional on finding the same opportunity generate the same value from it. We could also accommodate heterogeneous value from the same opportunity as long as the noise is sufficiently bounded.



**Proof.** We show that there are at least 2 searchers who have positive value each arbitrage. This can be done using direct computation of probabilities and an application of the union bound inequality. Consider the following sum  $Y_i := \sum_{j \in \mathcal{S}} v_{ij}$ . Thus,  $Y_i$  is the number of searchers that find opportunity  $i$ . Note that  $Y_i$  is a Binomial random variable with parameters  $n$  and  $p$ , that is  $Y_i \sim \text{Bin}(n, p)$ .

$$\begin{aligned} P[Y_i < 2] &= P[Y_i = 0] + P[Y_i = 1] = (1-p)^n + \binom{n}{1}(1-p)^{n-1}p \\ &\leq e^{-pn} + npe^{-p(n-1)} \leq \frac{1}{n^2} + 2 \log n \frac{1}{n^2} \leq \frac{2 \log n}{n^2}, \end{aligned} \quad (4)$$

where the first inequality is obtained from the well known inequality:  $1 - x \leq e^{-x}$  for any  $x > 0$ . By the union bound, we have:

$$P[\text{at least one } Y_i < 2] \leq mP[Y_1 < 2] \leq n \cdot \frac{\log n}{n^2} = \frac{\log n}{n}.$$

The last inequality is by (4). For any  $\varepsilon > 0$  there is a  $n$  large enough so that  $\frac{\log n}{n} < \varepsilon$  and therefore

$$P[Y_i \geq 2 \text{ for any } i] \geq 1 - \varepsilon.$$

Applying Proposition 4 shows the claim of the proposition.  $\blacktriangleleft$

On the other hand, if the probability of discovering opportunities shrinks sufficiently fast in the number of searchers, then searchers can capture value with positive non-vanishing probability:

► **Proposition 11.** *If  $p \in \Theta(\frac{1}{n})$  then with positive probability that is constant in  $n$  the validator receives 0 in the searcher-optimal core allocation.*

**Proof.** For each opportunity  $i \in \mathcal{A}$ , with constant probability, there is exactly one searcher that has a positive value, i.e.,  $Y_i = 1$ . Namely,

$$P[Y_i = 1] = \binom{n}{1}(1-p)^{n-1}p \rightarrow \frac{1}{e}\Theta(1),$$

as  $n \rightarrow \infty$ . Then,  $P[Y_i = 1 \text{ for any } i] \rightarrow \Theta(\frac{1}{e^m})$  as  $n \rightarrow \infty$ . That is, searchers complement each other in finding different arbitrages. From Proposition 4, with probability  $\Theta(\frac{1}{e^m})$ , we have  $M = 0$ .  $\blacktriangleleft$

► **Remark 12.** The previous results discuss the value distribution for scenarios where the block value is expected to be positive. If probability shrinks faster than  $1/n$  as  $n$  grows, then with high probability the produced block has value 0. However, conditional on the block having positive value, all value is captured by searchers in the searcher-optimal core allocation.

As remarked in footnote 5, block builders observe around  $\sim 1/3$  of submitted bundles clashing. We can use this number to get approximate values of the parameters in our model: we use  $n = 125$  which is the number of addresses that have placed at least 2 bundles in Ethereum blocks within the last 30 days prior to writing this paper (excluding addresses that have landed only one bundle gives us a crude way to identify addresses that have a high chance of being the main address used by a searcher) according to the website <https://libmev.com/>. Then

$$2/3 \approx P[Y_i < 2] = (1-p)^{125} + 125(1-p)^{124}p \Rightarrow p \approx 1\%.$$

### Capacity Constraints

The previous model can be easily adapted to the case of capacity constraints on blocks where a block can only contain up to a fixed number of transactions, denoted by  $K$ . Note that submodularity is maintained when adding a capacity constraint. Thus, Proposition 4 naturally extends to the case of capacity constraints. For the model with independent bundles, we now can consider the value function

$$\bar{v}^K(S) = \max_{B \in \mathcal{B}(S)} \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{S}} v_{ij} B_{ij} = \max_{A' \subseteq \mathcal{A}, |A'| \leq K} \sum_{i \in A'} \max_{j \in \mathcal{S}} v_{ij}.$$

It follows immediately that Corollary 9 holds with

$$M^K := \max_{A' \subseteq \mathcal{A}, |A'| \leq K} \sum_{i \in A'} M_i$$

and  $x_j^K := \sum_{i: j \in A' \cap \arg \max_j v_{ij}} (v_{ij} - M_i)$ .

Next, we consider how the bounds on searcher and validator value capture for the stochastic model change with capacity constraints on the block. We assume that only a constant fraction of possible opportunities can be accommodated. With a bound on the block size, the probability threshold above which the validator captures all value in all core allocation becomes lower, and now matches the corresponding threshold from Proposition 11 where the probability is positive (Proposition 11 still holds with a bound on the block size).

► **Proposition 13.** *Let  $n := |\mathcal{S}|$ . assume the block has capacity to include  $(1-\alpha)m$  transactions where  $m := |\mathcal{A}|$  and  $1/m < \alpha < 1$  is a constant. Then, there is a decreasing function  $\phi$  such that if  $p \in \omega(\frac{\phi(\alpha)}{n})$ , the validator gets the entire block value with high probability in any core allocation.*

**Proof.** We show that there are at least 2 searchers who have positive value each arbitrage. As in the proof of Proposition 10 defining  $Y_i := \sum_{j \in \mathcal{S}} v_{ij}$ , we obtain

$$P[Y_i < 2] \leq e^{-pn} + npe^{-p(n-1)}. \quad (5)$$

Now consider the probability that for more than  $\alpha m$  indices we have  $Y_i < 2$ . This is bounded by

$$P[Y_i < 2 \text{ for at least } \alpha m \text{ indices } i] \leq \binom{m}{\alpha m} P[Y_1 < 2]^{\alpha m} \leq \left( \frac{1}{\alpha} (e^{1-pn} + npe^{1-p(n-1)}) \right)^{\alpha m},$$

where the last inequality uses the well-known inequality  $\binom{a}{b} \leq \left(\frac{a}{b}\right)^b$  and the previously obtained inequality (5). Choosing  $\phi(\alpha)$  to satisfy

$$(1 + \phi(\alpha))e^{-\phi(\alpha)} = \frac{\alpha}{e},$$

we have for  $p \in \omega(\phi(\alpha)/n)$  that for any  $\epsilon > 0$  there is a  $n$  such that

$$\frac{e^{1-pn} + npe^{1-p(n-1)}}{\alpha} < \epsilon$$

and therefore

$$P[Y_i \geq 2 \text{ for at least } (1-\alpha)m \text{ indices } i] \geq 1 - \epsilon^{\alpha m} > 1 - \epsilon.$$

Applying Proposition 4 shows the claim of the proposition. ◀

► **Remark 14.** The previous result is tight in the sense that the matching result Proposition 11 which holds for  $p \in \Theta(1/n)$ , still holds for the case of capacity constraints on the block.

## 5 Conclusion

We have studied MEV extraction in block building as a function of searcher competition and have argued that the core, and in particular the rule that selects the searcher-optimal point within the core are suitable solution concepts that allow us to make robust prediction about value distribution in MEV extraction. We have further identified a dominant-strategy incentive compatible mechanism, giving searchers their marginal value contribution to the realized block, which would be a theoretically appealing payment mechanism for searchers in the case of passive proposers/builders.

The model and solution concept allowed us to make sense of stylized facts about the Ethereum block building market: validators capture most of the value most of the time, and searchers with unique edge that are less exposed to competition are able to capture significant value. A natural extension of our model for further research would add correlation between different MEV opportunities to our stochastic model. Such an enhanced model would be particularly suitable to study the competition between different block builders and would possibly make theoretical predictions about concentration and value capture in the builder market.

---


### References

- 1 Maryam Bahrani, Pranav Garimidi, and Tim Roughgarden. Centralization in block building and proposer-builder separation. *CoRR*, abs/2401.12120, 2024. doi:10.48550/arXiv.2401.12120.
- 2 Maryam Bahrani, Pranav Garimidi, and Tim Roughgarden. Transaction fee mechanism design in a post-mev world. *Cryptology ePrint Archive*, 2024.
- 3 Pedro Braga, Georgios Chionas, Piotr Krysta, Stefanos Leonardos, Georgios Piliouras, and Carmine Ventre. Who gets the maximal extractable value? A dynamic sharing blockchain mechanism. In Mehdi Dastani, Jaime Simão Sichman, Natasha Alechina, and Virginia Dignum, editors, *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2024, Auckland, New Zealand, May 6-10, 2024*, pages 2171–2173. ACM, 2024. doi:10.5555/3635637.3663097.
- 4 Agostino Capponi, Ruizhe Jia, and Sveinn Olafsson. Proposer-builder separation, payment for order flows, and centralization in blockchain. *Payment for Order Flows, and Centralization in Blockchain (February 12, 2024)*, 2024.
- 5 Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In *2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020*, pages 910–927. IEEE, 2020. doi:10.1109/SP40000.2020.00040.
- 6 Shayan Eskandari, Seyedehmahsa Moosavi, and Jeremy Clark. Sok: Transparent dishonesty: front-running attacks on blockchain. *CoRR*, abs/1902.05164, 2019. arXiv:1902.05164.
- 7 Donald B. Gillies. Solutions to general non-zero-sum games. *Contributions to the Theory of Games IV. Annals of Mathematics Studies*, 40:47–85, 1959.
- 8 Lioba Heimbach, Vabuk Pahari, and Eric Schertenleib. Non-atomic arbitrage in decentralized finance. *arXiv preprint*, 2024. arXiv:2401.01622.
- 9 Hervé Moulin. *Axioms of cooperative decision making*. Number 15 in Econometric Society Monographs. Cambridge university press, 1991.
- 10 Roger B Myerson and Mark A Satterthwaite. Efficient mechanisms for bilateral trading. *Journal of Economic Theory*, 29(2):265–281, 1983.
- 11 Noam Nisan, Tim Roughgarden, Éva Tardos, and Vijay V. Vazirani, editors. *Algorithmic Game Theory*. Cambridge University Press, 2007. doi:10.1017/CB09780511800481.

## 21:12 Searcher Competition in Block Building

- 12 Malleh Pai and Max Resnick. Structural advantages for integrated builders in mev-boost. *arXiv preprint*, 2023. [arXiv:2311.09083](#).
- 13 J David Pérez-Castrillo and Marilda Sotomayor. *Two Folk Manipulability Theorems in the General One-to-one Two-sided Matching Markets with Money*. FEA/USP, 2013.
- 14 Lloyd S. Shapley and Martin Shubik. The assignment game i: The core. *International Journal of Game Theory*, 1971.
- 15 Sen Yang, Kartik Nayak, and Fan Zhang. Decentralization of ethereum’s builder market. *CoRR*, abs/2405.01329, 2024. [arXiv:2405.01329](#).


# Who Wins Ethereum Block Building Auctions and Why?

**Burak Öz** ✉ 🏠 

Technical University of Munich, Garching, Germany

**Danning Sui** ✉ 🏠 

Flashbots, San Francisco, CA, USA

**Thomas Thiery** ✉ 🏠 

Ethereum Foundation, Lisbon, Portugal

**Florian Matthes** ✉ 🏠 

Technical University of Munich, Garching, Germany

---

## Abstract

The MEV-Boost block auction contributes approximately 90% of all Ethereum blocks. Between October 2023 and March 2024, only three builders produced 80% of them, highlighting the concentration of power within the block builder market. To foster competition and preserve Ethereum’s decentralized ethos and censorship resistance properties, understanding the dominant players’ competitive edges is essential.

In this paper, we identify features that play a significant role in builders’ ability to win blocks and earn profits by conducting a comprehensive empirical analysis of MEV-Boost auctions over a six-month period. We reveal that block market share positively correlates with order flow diversity, while profitability correlates with access to order flow from Exclusive Providers, such as integrated searchers and external providers with exclusivity deals. Additionally, we show a positive correlation between market share and profit margin among the top ten builders, with features such as exclusive signal, non-atomic arbitrages, and Telegram bot flow strongly correlating with both metrics. This highlights a “chicken-and-egg” problem where builders need differentiated order flow to profit, but only receive such flow if they have a significant market share. Overall, this work provides an in-depth analysis of the key features driving the builder market towards centralization and offers valuable insights for designing further iterations of Ethereum block auctions, preserving Ethereum’s censorship resistance properties.

**2012 ACM Subject Classification** Security and privacy → Economics of security and privacy

**Keywords and phrases** Block Building Auction, Proposer-Builder Separation, Maximal Extractable Value

**Digital Object Identifier** 10.4230/LIPIcs.AFT.2024.22

**Related Version** *Full Version:* <https://arxiv.org/abs/2407.13931> [77]

**Acknowledgements** We thank The Latest in DeFi Research (TLDR) program, funded by the Uniswap Foundation, for supporting this work. Special thanks to Flashbots and Eden for generously providing data. We also express our gratitude for the valuable comments of our reviewers and feedback by Julian Ma, Xin Wan, Christof Ferreira Torres, Kevin Pang, Žan Knafelc, and Tesa Ho.

## 1 Introduction

Ethereum blockchain grows by one block every 12 seconds if the block proposal slot is not missed. The consensus specifications [22] expect the selected consensus participant, the *validator*, to build a block and propose it to the rest of the network. However, nearly 90% [70] of all new blocks are produced via the MEV-Boost [23] block building auction, which is a Proposer-Builder Separation (PBS) [9] implementation. In this mechanism, the proposer



© Burak Öz, Danning Sui, Thomas Thiery, and Florian Matthes;  
licensed under Creative Commons License CC-BY 4.0

6th Conference on Advances in Financial Technologies (AFT 2024).

Editors: Rainer Böhme and Lucianna Kiffer; Article No. 22; pp. 22:1–22:25

Leibniz International Proceedings in Informatics



LIPIC Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

acts as an auctioneer, selling their block building rights, while a set of out-of-protocol entities, referred to as *builders*, compete by sending bids to intermediaries known as *relays*. The proposed MEV-Boost block is signed by the respective proposer, while the payload is constructed by the winning builder. By separating block proposing and building tasks, MEV-Boost aims to avoid validator centralization caused by the sophistication of running complex strategies to extract Maximal Extractable Value (MEV) [15].

As an out-of-protocol instantiation of PBS, MEV-Boost preserves decentralization on the consensus layer as validators have uniform access to MEV rewards [5]. However, it concentrates power within the builder market due to economies of scale. Between October 2023 and March 2024, only three builders, **beaverbuild**, **rsync**, and **Titan**, produced approximately 80% of all MEV-Boost blocks. With a small set of builders creating Ethereum blocks, the network becomes vulnerable to censorship [30, 69, 71].

To preserve Ethereum’s decentralized ethos and censorship resistance properties, fostering competition in the MEV-Boost block building auction is essential. As a first step, we must understand the competitive edges of the dominant players. MEV-Boost block builders currently have varying profitability levels, which do not strictly increase with the block market share they own, as shown in Figure 1. Thus, we need to study the critical factors for both obtaining market share and earning profits. Recent work [76] highlighted the pivotal role certain order flow providers play in winning blocks. We extend their insights by analyzing all active builders, order flow over time, and specific strategies to provide a comprehensive understanding of what drives builders’ success in the MEV-Boost auction.

Our contributions can be summarized as follows:

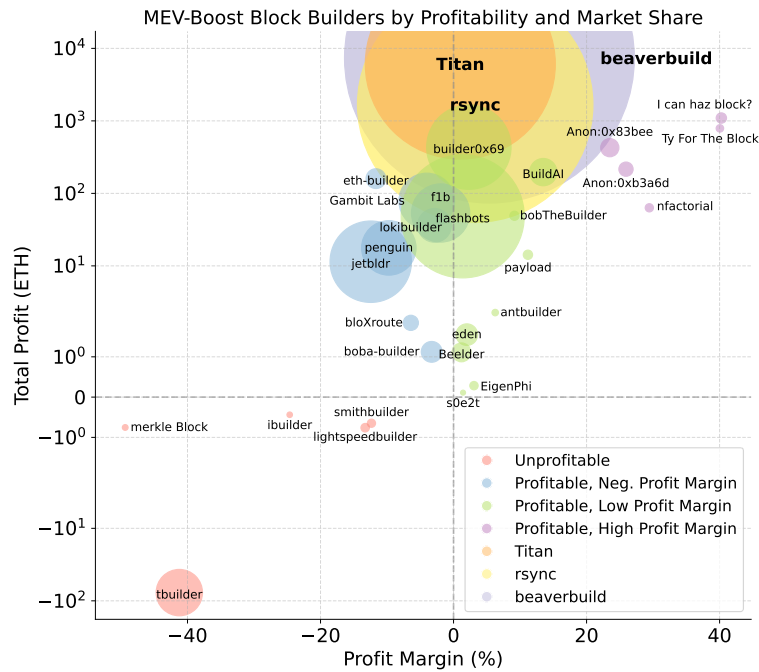
- We develop a novel methodology to identify key features in MEV-Boost builders’ success in winning blocks and earning profits, and provide metrics to monitor the builder market.
- We reveal that builders’ block market share positively correlates with order flow diversity, while profitability correlates with access to order flow from Exclusive Providers (EPs), such as integrated searchers and external providers with exclusivity deals.
- We show a positive correlation between market share and profit margin among the top ten builders, with features such as exclusive signal, non-atomic arbitrages, and Telegram bot flow strongly correlating with both metrics.
- We highlight a “chicken-and-egg” problem where builders need differentiated order flow to profit, but only receive such flow if they have a significant market share.
- We discuss the implications of our findings for Ethereum block auctions and explore existing solutions for addressing the censorship threat posed by the currently centralized block builder market.

## 2 Background and Related Work

In this section, we present the necessary background and discuss the related work.

### 2.1 Proposer-Builder Separation and the MEV-Boost Block Auction

The PBS framework [9] introduces the decoupling of block building and proposal tasks for staked Ethereum validators. While the validator remains responsible for signing and proposing the block, the new builder role handles the block content (i.e., execution payload) preparation. By delegating the MEV extraction task to builders, PBS aims to lower the barrier for entry for validators, who no longer need to become proficient in block building.



■ **Figure 1** Bubble plot illustrating MEV-Boost builders’ profit margin (average block value kept by the builder) on the x-axis and total profit in ETH on the y-axis. Bubble size represents the builder’s market share, measured by the total number of blocks built. Color indicates profitability, with a 15% profit margin threshold distinguishing high-profit margin builders. Builders with less than 0.01% market share or a profit margin below -50% are omitted for brevity.

Achieving the PBS goal is essential for Ethereum’s consensus security [5]. However, it is challenging due to the difficulty of establishing a *fair exchange* of value between the validator and the builder [63]. Validators need assurance of payment and execution payload delivery by the builder, while builders must ensure their block content is protected against unbundling and is included on-chain. Although solutions to fair exchange problem have been discussed [47, 48], there is no enshrined PBS implementation in the Ethereum protocol.

Flashbots proposed an out-of-protocol PBS implementation, MEV-Boost [23], which introduces relay intermediaries to address the fair exchange issue. This implementation became active with the Merge [20] in September 2022. In MEV-Boost, relays receive blocks from builders and validate<sup>1</sup> them to ensure that the promised bid to the validator is paid and that the block is valid to become canonical. To protect builders against unbundling attacks, relays follow a commit-and-reveal scheme, disclosing only the block header when retrieving the proposer’s signature. Although MEV-Boost requires trust in third parties, relays are presumed to act honestly due to their reputations. However, there have been instances where proposers exploited relay vulnerabilities to unpack and steal valuable MEV bundles from searchers [16, 44], indicating that the fair exchange problem remains unresolved.

An MEV-Boost block auction round [27] follows the 12-second slot structure of the Ethereum consensus protocol [22]. To win the right to produce the block proposed at slot  $n$ , builders start competing at slot  $n - 1$  by sending bids to the relays. Over time, builders

<sup>1</sup> *Optimistic* relays [46] such as Ultra Sound [68] can make builder bids available to the proposer without validation since they have additional security mechanisms in place (e.g., collateralized builder funds).

increase the value of their bids, derived from the fees offered by users sending transactions to the public mempool and Exclusive Order Flow (EOF) providers (e.g., MEV searchers, Telegram bots [31, 43]) submitting valuable bundles and transactions to the builders' private Remote Procedure Call (RPC) endpoints. Relays validate the builder bids and make them available to the proposer, who can continuously poll the *getHeader* endpoint on the *mev-boost middleware* to receive the highest bid from every registered relay. Eventually, the validator blindly signs the accepted bid's block header and submits it to the winner relay, who publishes the full block to the network. The described MEV-Boost process is summarized in Figure 2.

## 2.2 Maximal Extractable Value

MEV refers to the sum of value extractable from a blockchain in any given state, through transaction ordering, insertion, and exclusion [15]. Using these techniques and analyzing the blockchain and network state, *MEV searchers*, running bots and sophisticated smart contracts,<sup>2</sup> execute strategies such as Decentralized Exchange (DEX)-DEX or Centralized Exchange (CEX)-DEX arbitrages, sandwiches, and liquidations to earn profits [35, 54].

Due to the competitive, time-sensitive,<sup>3</sup> and state-dependent nature of MEV extraction, searchers must pay fees like bribes to builders for prioritized inclusion and execution in blocks. As a result, builders earn a proportion of the MEV that the searchers extract. However, they must also share their earnings with the proposers to win the MEV-Boost block auction, as described previously (see Section 2.1). Thus, MEV, originating from benign user transactions altering the blockchain state, flows through various entities in the block production pipeline and provides economic incentives for participants throughout the supply chain.

## 2.3 Order Flow Auctions

Order Flow Auctions (OFAs) are auctions where users share trade orders as unsigned transactions with searchers who compete in a sealed-bid format to backrun them. Users are incentivized by refunds up to 90% of the value bidders extract from backrunning opportunities [45], along with frontrun protection. OFAs, such as MEVBlocker [45] and MEV-Share [29], provide an RPC endpoint for users to privately submit transactions instead of sending them to the public mempool. These transactions are broadcasted, either in full detail or selectively, to bidders who compete to submit the highest value bundle to block builders. Builders then refund OFA users using the fee from the winning searcher's bundle.

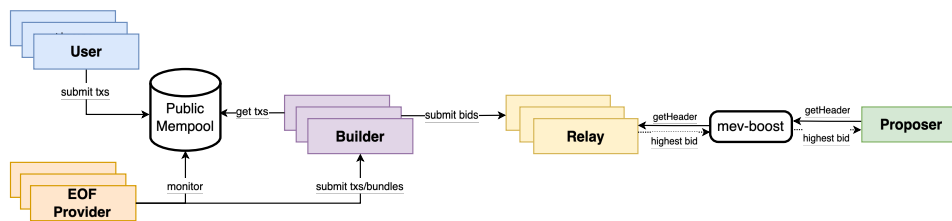
## 2.4 Related Work

Previous work has focused on Ethereum block building auctions from theoretical, game-theoretic, and empirical standpoints. [67] reveals that searchers prefer submitting bundles to builders with high-market share, and new entrants need to subsidize to obtain market share unless they operate their own searchers. [65] conducts an empirical study examining the strategic behavior of MEV-Boost block builders, focusing on their bidding behavior and order flow strategies, and establishes core metrics for analyzing builder profiles and identifying their competitive edges. [75] provides a game-theoretic model of the MEV-Boost auction and adopts an agent-based model to simulate builder strategies, showing the importance

<sup>2</sup> Example MEV bot contract: `0x6980a47bee930a4584b09ee79ebe46484fdbdd0`

<sup>3</sup> Arbitrages between an off-chain CEX and an on-chain DEX or two DEXes on different blockchains can be considered time sensitive as execution is non-atomic.





■ **Figure 2** MEV-Boost block production process. Users (blue) submit transactions to the public mempool, accessible to every builder. EOF providers (orange), including MEV searchers and valuable order flow bots, monitor user transactions and bundle them with their own transactions or directly submit individual transactions to builders. Builders (purple) submit blocks with bids to the relays (yellow), which make them available to the proposer (green) through the mev-boost middleware.

of latency and access to order flow. [76] identifies pivotal order flow providers for block builders and measures the competition and efficiency of the MEV-Boost auctions. [32] shows that integrated High-Frequency Trading (HFT) builders who extract top block position opportunities, such as CEX-DEX arbitrages, are favored to win the block auction when price volatility increases. [35] introduces heuristics tailored to detect CEX-DEX arbitrages, confirming that price volatility and the arbitrage volume are correlated. The work provides empirical evidence that certain builders, such as `beaverbuild` and `rsync`, run their own searchers to extract this value. [5] proves the necessity of a competitive PBS auction to enable homogeneous proposer rewards and avoid concentration of stake among validators proficient in extracting MEV rewards. Block proposal timing games have been studied in [53, 56], showing the marginal value of time for validators to earn more rewards from the builder bids. [72] measures the MEV-Boost market concerning the entities involved in the block production pipeline, and [34] analyzes the Ethereum landscape with a comparison between PBS and non-PBS blocks.

### 3 Methodology

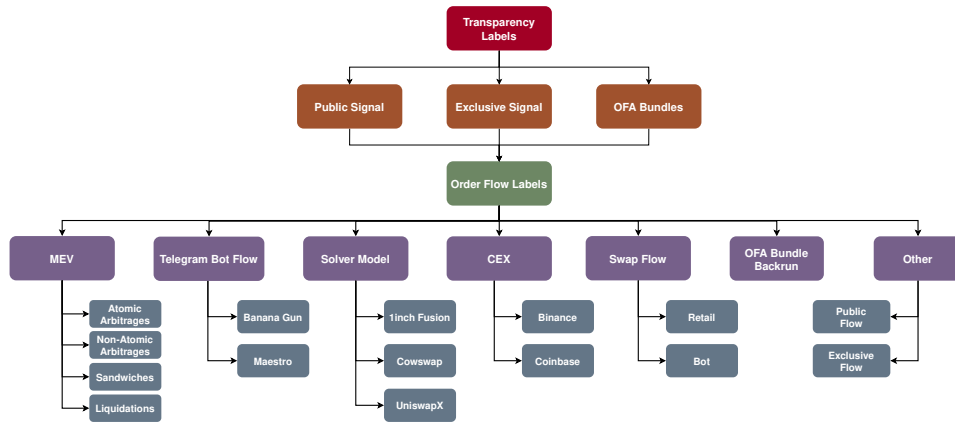
This section outlines the methodology for identifying transaction labels, collecting empirical data, and calculating MEV-Boost block metrics.

#### 3.1 Transaction Labels Taxonomy

We classify Ethereum transactions to understand the competitive edge different types provide to block builders based on the payments they offer. To this end, we determine a *transparency* and an *order flow* label for every transaction, following the taxonomy presented in Figure 3.

##### 3.1.1 Transparency Labels

Ethereum transactions submitted to public RPC endpoints propagate through a peer-to-peer network and enter nodes' public mempool. These transactions are available to every builder running a node in the Ethereum network. However, some transactions are submitted exclusively to a builder's private RPC endpoint, bypassing the public mempool. Additionally, certain transactions submitted to private endpoints are revealed to a selected group of entities, as occurs in OFAs. As part of our methodology, we apply the following conditions to determine the transparency label of a transaction:



■ **Figure 3** Taxonomy of transaction transparency and order flow labels. Every transaction has a transparency label (orange) based on its visibility on the network level and an order flow label (purple) determined by its objective. The gray labels stemming from the order flow labels represent the detailed categories we consider or popular providers of such order flow.

- *Public Signal*: Recorded in the mempool of at least one monitored node.
- *Exclusive Signal*: Not recorded in the mempool of any monitored nodes, indicating exclusive submission to the block builder.
- *OFA Bundle*: Part of an OFA bundle, either as the original user transaction or the searcher backrun. The heuristics for identifying OFA bundles are detailed in Appendix A.1 in [77].

### 3.1.2 Order Flow Labels

Ethereum transactions have various objectives, ranging from simple ETH transfers to complex MEV strategies. Consequently, they have different valuations for their originators, reflected through the payments offered to block builders. To understand the impact of builders’ access to order flow on their success in the MEV-Boost auction, it is critical to measure the value of different order flow types. Therefore, we assign an order flow label for every transaction included in our examined MEV-Boost blocks. While we consider detailed categories, the taxonomy is not exhaustive and omits certain known MEV strategies, such as Just-In-Time (JIT) liquidity [11] and cross-chain arbitrages [1, 51], and misses unknown long-tail strategies employed by the searchers.

We first determine if the transaction can be identified as a known *MEV* type, such as an atomic DEX-DEX or a non-atomic CEX-DEX arbitrage, a sandwich frontrun or backrun,<sup>4</sup> or a liquidation. If no MEV labels are matched, we check if the transaction was submitted by a *Telegram bot* such as Banana Gun [31] or Maestro [43]. These bots create wallets, store private keys for users, and execute highly time-sensitive strategies on their behalf, such as token mint sniping or long-tail token trading, in exchange for fees. Next, we consider *solver model* transactions submitted by solvers, fillers, and resolvers of protocols such as Cowswap [14], UniswapX [40], and 1inch Fusion [2], fulfilling user limit orders. Additionally, as blockchain users require *CEXes* like Binance [6] or Coinbase [12] to deposit and withdraw cryptocurrencies, we also detect and label such transactions individually. Further, if a

<sup>4</sup> Sandwiched user transactions are labeled according to their original objective.

transaction includes an ERC-20 token transfer or a swap but does not belong to prior categories, we label it as *retail swap* or *bot swap* depending on its initially interacted smart contract. If this contract is a known, non-MEV contract such as the Uniswap Universal Router,<sup>5</sup> we label the transaction as retail swap, likely originated through the frontend of the respective protocol. Otherwise, we consider the transaction a bot swap, interacting with an unlabeled contract, potentially executing long-tail MEV strategies or known MEV strategies undetected by our datasets and heuristics. We treat *OFA bundle backrun* as a distinct category, although user transactions within OFA bundles are not labeled individually. Finally, any remaining transaction is labeled as *other public flow* or *other exclusive flow* depending on its transparency type. The former category includes transactions like simple ETH transfers or batch submissions by rollup sequencers, while the latter covers transactions such as MEV bot contract deployments by searchers. The detailed identification methodology for each label is presented in Appendix A.1 in [77].

## 3.2 Data Collection

We curate an empirical dataset covering a six-month period of MEV-Boost blocks produced between October 1, 2023, and March 31, 2024, totaling 1,190,617 blocks. We primarily develop our data collection methods using external platforms such as Dune Analytics [18] and open source them for reproduceability by the community. We first build a compound query on Dune [59], utilizing various datasets available on the platform [33,36–39,61,73]. Through this query, we identify transaction properties beyond the standard payload data. These properties include transactions’ direct payments to builders’ coinbase address, ERC-20 transfers, certain MEV activities, trade volumes and fees in USD, MEV searcher and solver labels, Telegram bot labels, router contract labels, and mempool visibility. We export the resulting dataset, involving 174,240,225 labeled transactions. Additionally, we identify builder-controlled public keys using the extra data field of the Ethereum blocks [60]. Finally, we obtain the remaining necessary data for our methodology, such as all MEV-Boost payloads, bids on the UltraSound relay [68],<sup>6</sup> further builder public keys and searcher addresses, and other smart contract and transaction labels from external resources summarized in Table 3 in Appendix A.2 in [77].

## 3.3 MEV-Boost Block Metrics

The value of an MEV-Boost block is derived from the transactions in it. Each transaction offers a payment to the block builder, depending on its valuation for the originator. There are two ways a transaction can make an on-chain payment: through transaction fees and direct value transfers to the builder. Based on the total value builders receive from these transaction payments and their private valuation for the block, they offer a bid to the validator and make a payment if they win the auction. In this section, we present the methodology for calculating the true value, validator payment, and builder profit of MEV-Boost blocks.

### 3.3.1 True Block Value

We define the set of MEV-Boost blocks we analyze in our study as  $B$ . For each transaction  $t \in T_b$ , where  $b \in B$  and  $T_b$  represents all transactions included in  $b$ , we denote its gas priority fee (i.e., tip) and direct transfer to builder’s coinbase address as  $t_{tip}$  and  $t_{coinbase}$ , respectively.

<sup>5</sup> Uniswap Universal Router contract address: `0x3fc91a3afd70395cd496c647d5a6cc9d4b2b7fad`

<sup>6</sup> UltraSound relay was one of the most dominant relays during the course of this study [70].

## 22:8 Who Wins Ethereum Block Building Auctions and Why?

We refer to the refund amount of builder transactions in OFA bundles as  $t_{refund}$ . For failed transactions included in  $b$ , we set their coinbase payment to 0 and only consider their tip payment to the builder. We define the true value ( $TV$ ) of a block as:

$$b_{TV} = \sum_{t \in T_b} (t_{tip} + t_{coinbase} - t_{refund}).$$

We note that  $TV$  is only a best-effort estimate based on available on-chain data. For a more robust measure,  $TV$  calculation must account for private valuations of the searchers and builders and off-chain payment deals.

### 3.3.2 Validator Payment

For a given MEV-Boost block  $b$ , we denote the validator payment ( $VP$ ) as  $b_{VP}$ , referring to the amount tipped to the designated validator's proposer fee recipient address by  $b$ . Although most MEV-Boost blocks involve a builder transaction at the block's last index to make this payment, there are edge cases where the payment is made by an address related to the builder, or there is no payment from the builder, and the validator's proposer address is set as the block's fee recipient [52]. We assume the first case makes no difference when calculating  $b_{VP}$ , but we omit the blocks from our analysis where the proposer is set as the fee recipient. While builders can still profit from such blocks if they have own searcher transactions or through off-chain deals, since there is no on-chain value transfer to the builder, we ignore them. This strategy might be used when the builder aims to gain market share while avoiding the basefee for the validator payment transaction, especially when the expected profit from the block is low. In Table 7 in Appendix B.6 in [77], we summarize builders' excluded blocks.

In this paper, we calculate  $b_{VP}$  by identifying the validator payment transaction. We avoid using the payload value reported by relays as  $b_{VP}$  since this value can be manipulated if the relay calculates a builder's bid value by the balance difference of the proposer [64], leading to a miscalculation of validator earnings. One reason for such discrepancies is stake withdrawals [21]. When a builder includes a withdrawal transaction to the validator of the block, such relays over-report the bid by considering the withdrawal amount as part of the validator payment done by the builder, potentially causing profit loss for the validator [3]. We share empirical results of builders' validator payment patterns in Table 7 in Appendix B.6 in [77].

### 3.3.3 Builder Profit

We denote the builder profit ( $BP$ ) of an MEV-Boost block  $b$  as  $b_{BP}$ . When calculating  $b_{BP}$ , we handle the following edge cases:

- Besides the validator payment and OFA refund transactions, we do not deduct other builder value transfers from  $b_{TV}$  when calculating  $b_{BP}$ . Since such transfers can be issued to builder-controlled Externally Owned Accounts (EOAs) or smart contracts, deducting their value or using builder coinbase address balance change may underestimate  $b_{BP}$  [52],
- When the builder does not issue a validator payment transaction, we check if the last transaction of the block still pays the validator. If that is the case and the validator's proposer address is not the fee recipient of the block, we assume this payment is made on behalf of the builder, either by an address associated with the builder or one controlled by the UltraSound relay. The latter occurs when the builder opts to use UltraSound's recently deployed bid-adjustment feature [7], which alters the builder's winning bid to be 1 wei above the second-best bid available on any relay and refunds a proportion of the

bid surplus. Before March 5, 2024, the entire difference was refunded, but since then, UltraSound has been taking half of it, which is eventually transferred by the builder. We refer to this relay payment ( $RP$ ) value of block  $b$  as  $b_{RP}$ . In Table 7 in Appendix B.6 in [77], we present the total amount paid by each builder to UltraSound.

In this paper, we define the on-chain builder profits as  $b_{BP} = b_{TV} - b_{VP} - b_{RP}$ , and calculate the profit margin ( $PM$ ) of a builder from  $b$ , referring to the share of true block value kept by the builder, with  $b_{PM} = \frac{b_{BP}}{b_{TV}}$ .

### 3.4 Limitations

Our methodology has the following limitations:

1. **Transparency Labels:** Accuracy is limited by the mempool dataset’s coverage across node providers [33]. Determining transaction exclusivity across builders requires access to losing MEV-Boost bids’ execution payload data, which the relays do not publicly disclose.
2. **Order Flow Labels:** Potentially includes false labels due to our strict heuristics (see Appendix A.1 in [77]) and use of external data (see Table 3 in Appendix A.2 in [77]).
3. **Builder Profits:**
  - Overestimated if the builder has off-chain deals with providers such as MEV searchers or Telegram bots, and pays them for accessing their order flow.
  - Underestimated if the builder is vertically integrated with any MEV searcher, as we only consider on-chain profits made through priority fees and coinbase transfers.
4. **MEV Searcher Labels:** Not exhaustive as we cannot detect all the bots one searcher operates, obscuring the significance of a single searcher across multiple addresses.
5. **Bidding Data:** Can be biased as we only consider bids on the UltraSound relay.

## 4 MEV-Boost Decomposition

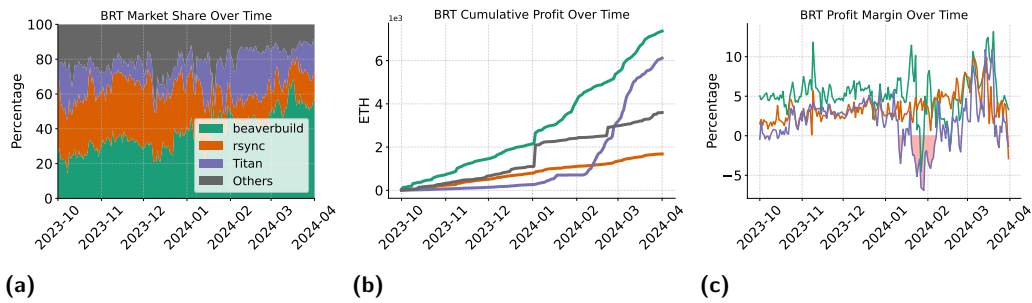
We present our measurements, decomposing the MEV-Boost auction to identify potential features driving builder success. First, we examine the builder market to observe the dominant players in market share and profits. Next, we analyze the order flow in MEV-Boost blocks to determine the most valuable flows and which builders receive them. Finally, we explore various strategies builders adopt to gain a competitive advantage in the auction.

### 4.1 Builder Market Structure

Between October 2023 and March 2024, 39 different block builder entities won the MEV-Boost auction, producing 1,190,617 blocks. `beaverbuild`, `rysnc`, and `Titan` (BRT) contributed the most blocks, accounting for 34.86%, 22.98%, 22.74% of the total, respectively. In approximately 8% of MEV-Boost blocks, the builder set the fee recipient address as the proposer, with `Titan` responsible for 92.86% of these cases (see Table 7 in Appendix B.6 in [77]). We suspect `Titan` adopts this strategy when the block’s profits are insufficient to cover the basefee of the validator payment transaction. In the following analyses, we exclude these blocks for reasons discussed in Section 3.3.2.

In Figure 4, we show the market share (Figure 4a), cumulative profit (Figure 4b), and daily profit margin (Figure 4c) of BRT over time. On average, these three builders construct 80% of all blocks. We observe a surge in `Titan`’s market share and profits since February 2024, surpassing `rsync` in cumulative profits, approaching `beaverbuild`. Interestingly, when measured in USD, `Titan` has the highest total profit, reaching roughly 19.7M USD, despite building fewer blocks than `beaverbuild` and `rsync`, who earned 19.4M USD and 4.27M

## 22:10 Who Wins Ethereum Block Building Auctions and Why?



■ **Figure 4** Trends in the market share, profits, and profit margin of the top three builders with highest market share, BRT, over time. Figure 4a is an area plot highlighting the changes in market share owned by BRT and the aggregated, remaining builders, denoted as “Others” in the legend. Figure 4b is a line plot showing the cumulative ETH profits of the same entities. Figure 4c displays the daily profit margin changes of BRT. The area below 0 on the y-axis is filled with red to indicate days when the builder was, on average, unprofitable.

USD, respectively. Although the profits we calculate gives an insight about builders’ earnings, the exact values can be different since we only consider the value provided through transaction priority fees and coinbase payments. Currently, we miss the upstream value builders make through their integrated searchers [32, 35]. Furthermore, we cannot track profits accurately if builders have any off-chain settlement with order flow providers.

Examining the builders besides BRT, we discover that only seven others have a market share nearly 1% or more (see Table 1). Together, these top ten builders produced 97.67% of the blocks but only earned 83.85% of the 52M USD total builder profit. This discrepancy highlights the diverse builder profiles, with different specializations in gaining market share and profits.

Table 1 summarizes the measurements for the top ten builders with the largest market share.<sup>7</sup> The various profiles of the builders are reflected in their profitability metrics. `flashbots` and `builder0x69` behave neutrally, not winning blocks by paying more than they earn from the included transactions. In contrast, `jetbldr`, `penguin`, and `tbuilder` heavily subsidize and maintain a negative profit margin on average. However, except for `tbuilder`, remaining builders are overall profiting.<sup>8</sup> `beaverbuild` has the highest profit margin at 5.4% within the top ten, whereas across all builders, `I can haz block?` and `Ty For The Block` have approximately 40% profit margins (see Table 5 in Appendix B.2 in [77]).

We note that `Titan` has a lower profit margin than `rsync`, despite earning significantly higher profits. While our profit calculation may not reflect precise values due to discussed issues, the discrepancy between profits and profit margin could be because `Titan` has more days with a negative profit margin compared to the other two top builders, who maintain a more consistent profit margin over time (see Figure 4c). As `beaverbuild` and `rsync` run their own searchers [32, 35], they receive a more consistent order flow stream. Conversely, `Titan` relies on external order flow providers who may not be as consistent, resulting in some blocks with significant profits and others that are entirely unprofitable. This pattern also supports our observation about the high percentage of `Titan` blocks where the proposer is set as the fee recipient (see Table 7 in Appendix B.6 in [77]).

<sup>7</sup> We only consider the MEV-Boost blocks where the builder is the fee recipient address.

<sup>8</sup> This can be due that subsidizing builders seldom earn significant profits, keeping them profitable.

■ **Table 1** Market Share and Profitability Metrics of Top Ten Builders.

Builder	Total Blocks [#]	Market Share [%]	Total Validator Payment [ETH]	Total Subsidy [ETH]	Total Profit [ETH]	Profit Margin [%]
beaverbuild	413,868	37.91	49,871.82	-70.46	7,341.62	5.4
rsync	273,126	25.02	33,691.74	-80.71	1,679.73	3.25
Titan	177,915	16.3	22,025.19	-61.49	6,083.95	1.02
flashbots	74,636	6.84	6,918.79	0.0	48.22	1.38
builder0x69	35,083	3.21	3,878.92	0.0	434.64	2.3
jetblidr	32,670	2.99	1,192.15	-73.21	11.41	-12.45
flb	16,716	1.53	1,251.63	-12.39	53.59	-1.94
Gambit Labs	15,281	1.4	889.45	-19.75	78.42	-4.03
penguin	14,622	1.34	1,028.28	-41.77	17.7	-9.73
tbuilder	10,584	0.97	338.38	-79.1	-77.01	-41.23

## 4.2 Order Flow Breakdown

In this section, we analyze the order flow in the MEV-Boost blocks, focusing on their transparency and significance. We use the labels defined in Section 3.1.

### 4.2.1 Transparency

We first examine the transparency of the Ethereum order flow. Recent work [76] showed an increasing share of value coming from EOF. We confirm their findings and examine the transparency of individual order flow labels and builders' blocks.

In Figure 5a, we show the share of true block value of each transparency label over time. Exclusive transactions and bundles, referred to as exclusive signal, provide 66.69% of all value while consuming only 19.6% of blockspace in terms of gas (see Figure 5b). Similar to the results in [76], we find that this flow constitutes 71% of all block value in more than 50% of the blocks, with nearly five times more value per unit of gas consumed than public signal. While almost all blocks have at least one exclusive and one public transaction, OFA bundles occur in only 4% of them, highlighting the scarce adoption of such protocols.<sup>9</sup> We summarize our measurements of transparency labels in Table 4 in Appendix B.1 in [77].

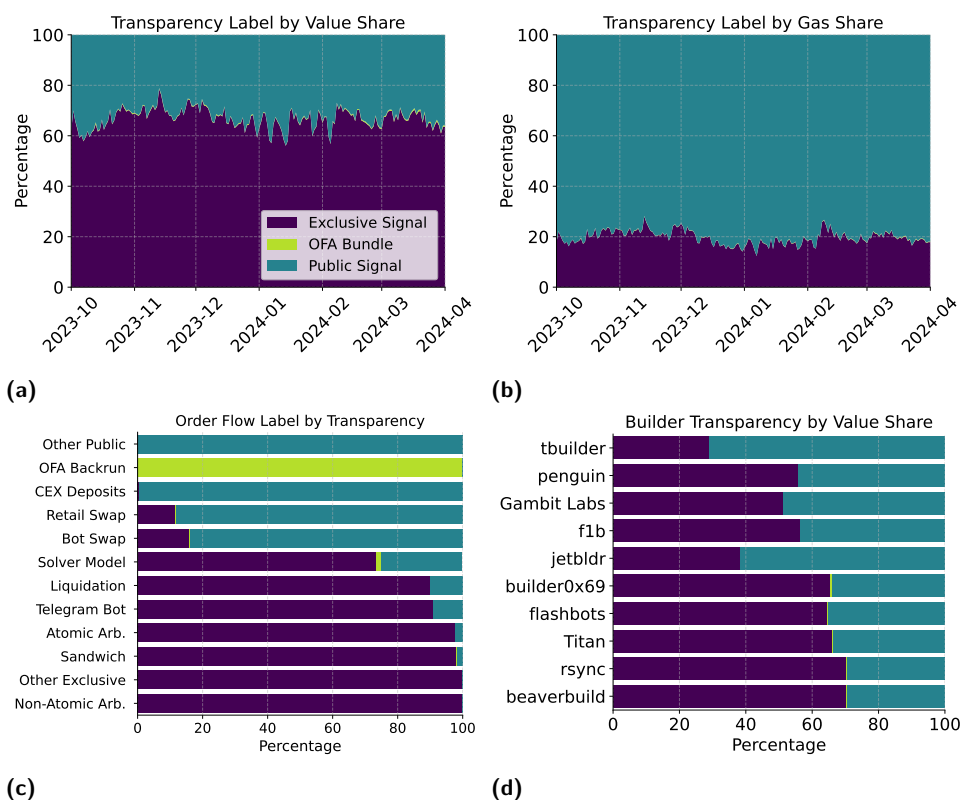
Next, we investigate the transparency of individual order flow labels, expecting certain types to avoid the public mempool to protect their valuable strategies. As shown in Figure 5c, every order flow label has a dominant way its involved in Ethereum blocks, corresponding to more than half of its total volume. Telegram bot flow and MEV strategies, including atomic and non-atomic arbitrages, sandwiches,<sup>10</sup> and liquidations, mostly bypass the public mempool. On average, 99% of all MEV order flow is exclusively submitted to the builders. Conversely, retail and bot swap flows are primarily submitted to the public mempool, exceeding 80% of their total volume. Interestingly, a relatively significant share of solver model flow is involved in OFA bundles (1.28%), receiving refunds while fulfilling user trade orders. Notably, Cowswap solver model is known to be forwarding its flow to the MEVBlocker OFA protocol.

Finally, we assess the transparency of builders' blocks. In Figure 5d, we present the share of total value the top ten builders with the largest market share receive from public and exclusive signal and OFA bundles. BRT, with the largest market share and most profits,

<sup>9</sup> MEVBlocker and MEV-Share (through Flashbots Protect [28]) have a larger volume if we consider the exclusive user transactions submitted to them which were not part of an OFA bundle.

<sup>10</sup> Sandwiches are mostly exclusive as we do not account the sandwiched user transactions in this label. Such transactions are labeled according to their original category, such as retail swap.

## 22:12 Who Wins Ethereum Block Building Auctions and Why?



**Figure 5** Figure 5a and Figure 5b are area plots highlighting the share of value provided and blockspace consumed (in gas) by each transparency label over time. Figure 5c is a horizontal bar plot showing the transparency of each order flow label, measured in total transaction volume. Figure 5d is a horizontal bar plot indicating the share of value that the top ten builders with the highest market share receive from each transparency label. Builders on the y-axis are ordered in ascending order based on their market share, with the builder with the highest share listed at the bottom. The legend in Figure 5a applies to all figures.

receive the highest relative value from EOF, exceeding 65% of their total. The builders with the lowest profit margins among the top ten, `jetbldr` and `tbuilder`, receive the least value from EOF, with shares below 40%. Among builders outside the top ten who have greater than 0.01% market share, more extreme concentrations are observed, with exclusive value shares ranging between 1% and 91% (see Figure 11 in Appendix B.3 in [77]).

### 4.2.2 Significance

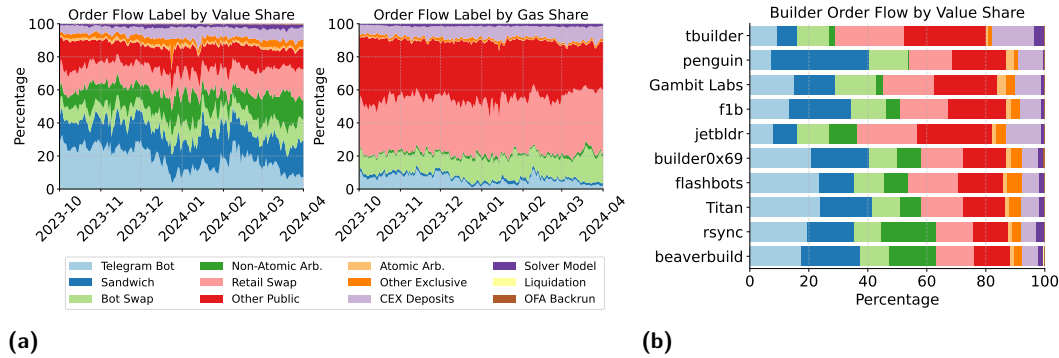
The order flow included in a block is a good estimator of its true value. To understand valuable flows, we analyze MEV-Boost blocks for the significance of the order flow labels. We find that Telegram bot and MEV order flows (see Section 3.1.2) cumulatively contribute around 51% of all block value while consuming merely 10% of all gas spent in MEV-Boost blocks (see Figure 6a). These flows provide around 53% of all block value in more than half of the blocks. Notably, they are primarily exclusively submitted to block builders, as shown in Figure 5c. In contrast, order flow labels such as retail swap and other public flow, most frequently observed in the public mempool, cumulatively provide a little over a quarter of the total block value while consuming more than 67% of the total blockspace. These



measurements of order flow labels, also summarized in Table 4 in Appendix B.1 in [77], indicate that valuable order flow is predominantly exclusively submitted, and blocks are mostly filled with less valuable, public transactions.

One of the drivers of competition in the MEV-Boost auction is the order flow builders receive [75]. As order flow labels have varying levels of public and exclusive volume (see Figure 5c) and value (see Table 4 in Appendix B.1 in [77]), we must analyze builders’ order flow compositions concerning these labels to understand who possesses the valuable flow.

In Figure 6b, we show the order flow composition of the top ten builders with the highest market shares. Builders have varying compositions, with certain builders receiving a more significant share from specific order flows than others. Some of the most valuable flows, such as Telegram bot flow and non-atomic arbitrages, are more significant in the compositions of the top three builders, BRT, while almost all builders receive a considerable share of sandwich flow, which is a common strategy among MEV searchers. Furthermore, we examine the order flow composition of builders outside the top ten and discover more remarkable concentrations, with a single order flow providing more than 80% of the total value a builder receives, such as the sandwich flow of `s0e2t`, as shown in Figure 12 in Appendix B.3 in [77].



**Figure 6** Figure 6a shows the significance of order flow labels over time. The left and right panels depict the share of total block value provided and blockspace consumed (in gas) by each order flow label. Figure 6b displays horizontal bars representing the share of value the top ten builders with highest market share receive from each order flow label. Builders on the y-axis are ordered in ascending order based on their market share, with the builder with the highest share listed at the bottom. Order flow labels in Figure 6b use the same coloring as indicated in the legend in Figure 6a.

### 4.3 Builder Strategies

MEV-Boost block builders adopt various strategies to gain a competitive edge. While order flow makes up their value, bidding and latency strategies also play an important role during the auction process [75]. In this section, we measure the significance of four builder strategies.

#### 4.3.1 Block Packing

Authors in [32] discuss the importance of accessing value from top block positions to win the block auction. They show through a theoretical model that builders who earn more from Top-of-Block (ToB) are likely to dominate the PBS auction. We analyze the block packing strategies of builders to support this outcome with empirical data. To this extent, we measure the share of the value builders earn from ToB, Body-of-Block (BoB), and End-of-Block (EoB) positions. We define ToB as the first 10% of transactions based on their normalized index in the block, following [4]. Similarly, EoB, denotes the bottom 10%. Lastly, BoB represents the middle 80% of the block. Builder transactions are excluded to prevent skewing the data.

In Figure 7a, we show the share of value the top ten builders by market share earn from transactions in each position group. ToB contributes the most significant value to all top builders except `penguin`, and BRT receive more than 74 % of their block value from order flow in ToB. The significance of ToB can be attributed to the fact that more than half of the total EOF volume resides in this position, providing, on average, 88.5 % of the total EOF value. Conversely, BoB and EoB only contribute approximately 10.5 % and 1 % of the total EOF value. Interestingly, `Titan`, `Gambit Labs`, and `penguin` receive a relatively considerable value from EoB. These builders could be allowing MEV searching at the end of their blocks, on the state transitioned to by the included transactions, and receiving significant payment from the granted searchers. The valuable block positions of the remaining builders are summarized in Table 5 in Appendix B.2 in [77].

### 4.3.2 Subsidization

Builders who have just entered the market, not receiving significant volumes of valuable order flow, are expected to subsidize their blocks unless they run their own MEV searchers [67]. Since we previously found that certain builders have negative profit margins (see Table 1), we are interested in how significantly they subsidize to win the auction. Figure 7b shows the shares of blocks the top ten builders with highest market shares profited, subsidized, or were neutral, making dust amount or zero profit.<sup>11</sup> Notably, these measurements are solely based on on-chain profits and may not reflect the complete picture, as builders can profit from their integrated searchers or return a percentage of the value to order flow providers.

The top three builders in market share, BRT, also have the highest shares of profitable blocks, with `beaverbuild` exceeding 40 %. While `Titan` is neutral in approximately 50 % of their blocks, this share goes up to 68 % when considering the blocks where they set the proposer as the fee recipient as zero profit as well (see Table 7 in Appendix B.6 in [77]).<sup>12</sup> `flashbots` and `builder0x69` predominantly break even, supporting the measurements summarized in Table 1. Conversely, `jetbldr`, `penguin`, and `tbuilder` win more than 75 % of their blocks by subsidizing, paying extra value on top of the on-chain value they receive from their order flow. Among builders outside the top ten, some adopt a profit-only strategy. Examples include `Anon:0x83bee`, `Anon:0xb3a6d`, `I can haz block?`, and `Ty For The Block`, who were also labeled as profitable, high-profit margin builders in Figure 1. The subsidization and profitability metrics of the rest of the builders are available in Table 5 in Appendix B.2 in [77].

### 4.3.3 Exclusive Order Flow Access

EOF contributes the majority of value to MEV-Boost blocks, as shown by our study (see Section 4.2.1) and previous works [65, 76]. We also discovered that builders have varying level of EOF in their blocks (see Figure 5d), indicating a non-uniform access. Extending studies on integrated builders and prominent order flow providers [32, 35, 67, 74, 76], we examine the significance of EOF providers for each builder. We consider providers including, public entities such as Flashbots Protect<sup>13</sup> and MEVBlocker, MEV searchers, and solvers.<sup>14</sup>

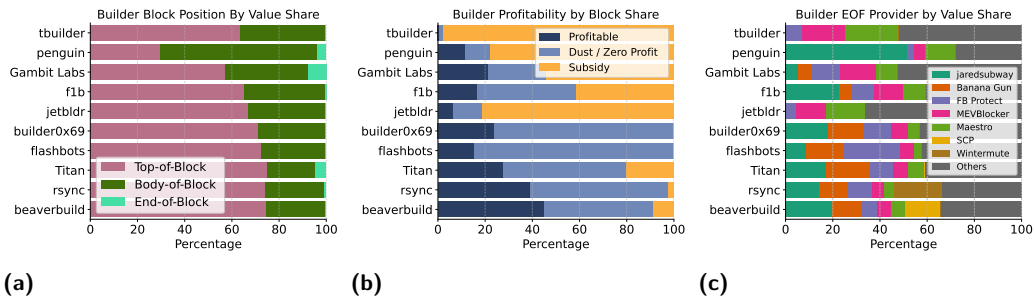
<sup>11</sup>We set 0.001 ETH as the dust amount threshold.

<sup>12</sup>We assume zero on-chain builder profit from such blocks although the builder can still earn profits from their integrated searchers or off-chain deals, as discussed in Section 3.3.2.

<sup>13</sup>Flashbots Protect is considered as the provider of the flow received by Flashbots's private RPC endpoints, including MEV-Share OFA bundles.

<sup>14</sup>Solvers refer to the liquidity routing entity in the solver model introduced in Section 3.1.

Figure 7c presents the shares of EOF value that the top ten builders with the highest market share receive from the seven most significant providers based on total value. These providers contribute roughly 70 % of all EOF value in more than half of the MEV-Boost blocks, and supply over 48 % of EOF value to each builder in the top ten, except `jetblidr`, who primarily sources EOF from other providers. We find that Flashbots Protect and MEVBlocker appear consistently, likely due to their public EOF access requirements, without needing private deals [76]. Maestro Telegram bot provides considerable flow to each top builder, unlike Banana Gun, which sends almost no flow to `jetblidr`, `penguin`, and `tbuilder` but provides significant value to `Titan`, especially starting from February 2024 (see Figure 13b in Appendix B.4 in [77]). Additionally, `jetblidr` and `tbuilder` receive negligible EOF from the well-known sandwich searcher bot, `jaredfromsubway.eth`, while `penguin` gets more than 50 % of its EOF value from this provider, potentially explaining the prominence of sandwich flow in their order flow composition (see Figure 6b). Finally, SCP and Wintermute are highly significant only for `beaverbuild` and `rsync`, respectively, as these providers are operated by them [32,35].<sup>15</sup> Further integrations and private deals could exist among the remaining builders and EOF providers, as shown in Figure 14 in Appendix B.4 in [77].



**Figure 7** Figure 7a is a horizontal bar plot showing the share of value the top ten builders by market share earn from ToB, BoB, and EoB. Figure 7b is a horizontal bar plot presenting the share of blocks the top ten builders profited, subsidized, or made dust amount or zero profit. Figure 7c is a horizontal bar plot displaying the share of EOF value the top ten builders receive from the seven most significant EOF providers based on total value. In the plot legend, “jaredsubway” and “FB Protect” stand for “jaredfromsubway.eth” and “Flashbots Protect” providers, respectively. The total EOF value of the remaining providers is aggregated and denoted as “Others”. In all figures, builders on the y-axis are ordered from top to bottom in ascending order of market share.

### 4.3.4 Latency and Bidding

MEV-Boost block builders compete in latency when placing bids on the relays. Although the proposer is expected to release the block at slot start [22], the exact time is unknown. Due to the stochastic nature of this process, low latency is beneficial for builders to update bids [65], reflecting changing valuations for the block over time and reacting to other bids.

Using data from the UltraSound relay, we examine MEV-Boost builders’ latency and bidding behavior. We measure their bidding frequency by the average number of bids they place in each slot. Furthermore, we calculate the lag between their bid updates to understand

<sup>15</sup> PLM and Rizzolver solvers, identified to be belonging to `beaverbuild` and `rsync` [61], are aggregated with SCP and Wintermute providers, respectively.

## 22:16 Who Wins Ethereum Block Building Auctions and Why?

how quickly they can reflect their new valuation and react to other bids. Lastly, we identify cases where builders strategically lowered their bid values through cancellations [50], a strategy discussed to be especially useful for builders running CEX-DEX arbitrage bots [65].

In Table 2, we summarize the bidding behavior of the top ten builders in market share for the blocks they won on UltraSound. We observe that the market share gap between **rsync** and **Titan** became more significant compared to their market shares when considering blocks from all relays (see Table 1), with a jump from approximately 8.7% to 12.2%. This could be because the UltraSound relay allows optimistic submissions [46], potentially favoring integrated builders such as **beaverbuild** and **rsync** [32, 35], as they can more frequently update their bids based on the value they derive from their searchers.

We measure that the top ten builders have diverse bidding strategies, with the average number of placed bids ranging from 6.1 by **tbuilder** to 49.64 by **penguin**, who also has the lowest average update lag between all of their bids in a slot (see Table 2). Builders can update bids quickly and frequently by having low latency relative to the relay’s geographical position or by submitting new bids simultaneously from multiple public keys they control.

BRT, who won the most auctions on the UltraSound relay, issue the highest number of cancellations, with, on average, more than one cancellation per block.<sup>16</sup> We note that, two of these builders, **beaverbuild** and **rsync**, have relatively high non-atomic arbitrage shares (see Figure 6b) and were identified to be running their own CEX-DEX arbitrage searchers [32, 35]. Therefore, we confirm findings in [65] regarding the use of cancellations by integrated builders. We also observe that some builders outside the top ten, such as **antbuilder**, who also has a relatively large non-atomic arbitrage flow (see Figure 12 in Appendix B.3 in [77]), issued around ten cancellations per block they built (see Table 6 in Appendix B.5 in [77]).

We note the distribution of average winning times for builders, referring to the average slot time a builder’s winning blocks were selected by the proposer. While the winning time for each builder is beyond the slot start (0ms), possibly due to timing games [53, 56], there is approximately a 580ms difference between the earliest and the latest winner times. This difference can be related to the slot times builders start and stop bidding in the auction [65].

■ **Table 2** Bidding and Latency Metrics of Top Ten Builders (UltraSound Relay).

Builder	Total Blocks [#]	Avg. Bids [#]	Avg. Update Lag [ms]	Avg. Winner Time [ms]	Total Cancels [#]	Avg. Cancels [#]
beaverbuild	137,721	31.54	130.41	586.59	248,195	1.74
rsync	95,310	25.27	166.21	531.38	212,504	2.29
Titan	47,539	31.11	186.25	605.83	91,337	1.78
flashbots	32,612	16.25	411.36	348.45	3,014	0.1
builder0x69	23,294	20.64	183.08	575.09	17,445	0.65
jetbldr	12,960	35.2	103.08	605.75	904	0.09
Gambit Labs	9,530	15.85	349.05	496.3	492	0.1
flb	9,242	25.42	214.18	680.45	10,494	0.98
penguin	7,783	49.64	69.92	652.91	966	0.13
tbuilder	4,786	6.1	476.68	104.71	81	0.01

<sup>16</sup>The average number of cancellations BRT issue per block increases from 1.85 to 2.09 after the introduction of UltraSound’s bid adjustment feature [7].

## 5 Results

In this section, we identify the features associated with gaining market share and earning profits in the MEV-Boost auction. We apply a Spearman correlation with a p-value set to 0.05 to assess the strength and direction of the associations based on rank orders. We exclude builders with less than 0.01 % market share (see Table 5 in Appendix B.2 in [77]).

### 5.1 Order Flow Diversity

The diversity of order flow a builder receives can indicate access to multiple sources, making the builder more competitive in auctions compared to those who depend on a specific order flow type, which may not always be available. To capture this, we measure the Shannon entropy [41] of builders' order flow. With twelve unique order flow labels (see Section 3.1.2), the Shannon entropy is defined as  $H(X) = -\sum_{i=1}^{12} p_i \log_2(p_i)$ , where  $p_i$  represents the value share of each label in a builder's order flow composition. Higher entropy values indicate a more diverse order flow, while smaller values signify a concentrated composition.

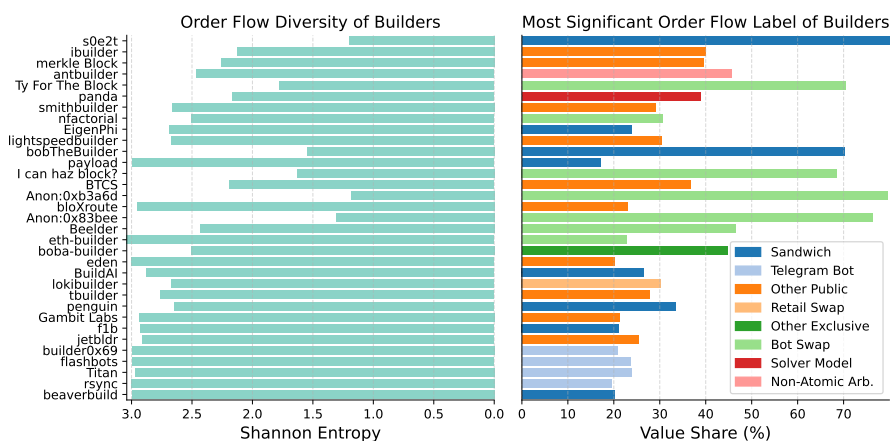
In the left panel of Figure 8, we show the entropy value of each builder's order flow composition. The top ten builders with the most market share have entropy values closer to the maximum ( $H_{\max} = \log_2(12) \approx 3.585$ ). However, among builders with smaller market shares, lower values are observed, suggesting a concentration around a few order flow labels. To better understand the order flow concentration, we measure builders' highest value share label, denoted as Most Significant Order Flow (MSOF). The outcome in the right panel of Figure 8 shows that the top ten builders have various MSOFs labels, although all have less than 32 % value share. In contrast, across the remaining builders, MSOF value share can exceed 70 %, indicating a highly concentrated order flow composition.

To examine the link between builders' success in winning the auction and the diversity of their order flow, we measure the correlation between builders' market share and order flow entropy. We find a significant positive correlation, with a coefficient of 0.66 and a p-value of  $2.22e - 05$ . This suggests that builders receiving different order flow types have an advantage in the auction and can win more frequently, as they are not dependent on a single type of flow. Conversely, builders with a concentrated order flow composition, where MSOF is more prominent, can only be competitive when that specific order flow type is available.

### 5.2 Exclusive Providers

MEV-Boost block builders receive EOF from various providers, as discussed in Section 4.3.3. Builders with own providers, such as `beaverbuild` and `rsync` [32,35], can produce profitable blocks as they access order flow not shared with others. To identify such exclusive relationships, we employ Linear Discriminant Analysis (LDA), classifying providers that can successfully separate a builder's blocks from others. We consider the model's Decoding Accuracy (DA) to be significant if it exceeds a threshold calculated using a binomial cumulative distribution [13].

In Figure 9, we present a heatmap showing the DA of EOF providers for each builder with at least one statistically significant provider. Unidentified providers are enumerated according to the mapping in [62]. A high DA indicates that the builder is distinguished by the significance of EOF they receive from the provider. If an EOF provider is prominent for many builders, it can be considered a neutral provider, accessible by all. Conversely, an EOF provider will be significant for only a single builder if they are vertically integrated or have an exclusivity deal. We define these providers as EPs, identifiable by columns with a single colored cell in the heatmap.



**Figure 8** The left panel is a horizontal bar plot presenting each builder’s order flow diversity through a Shannon entropy value. Builders are ordered from top to bottom in ascending order of market share. The right panel is a horizontal bar plot depicting each builder’s MSOF, referring to their highest value order flow label (in percentage). Bars are colored based on the order flow label.

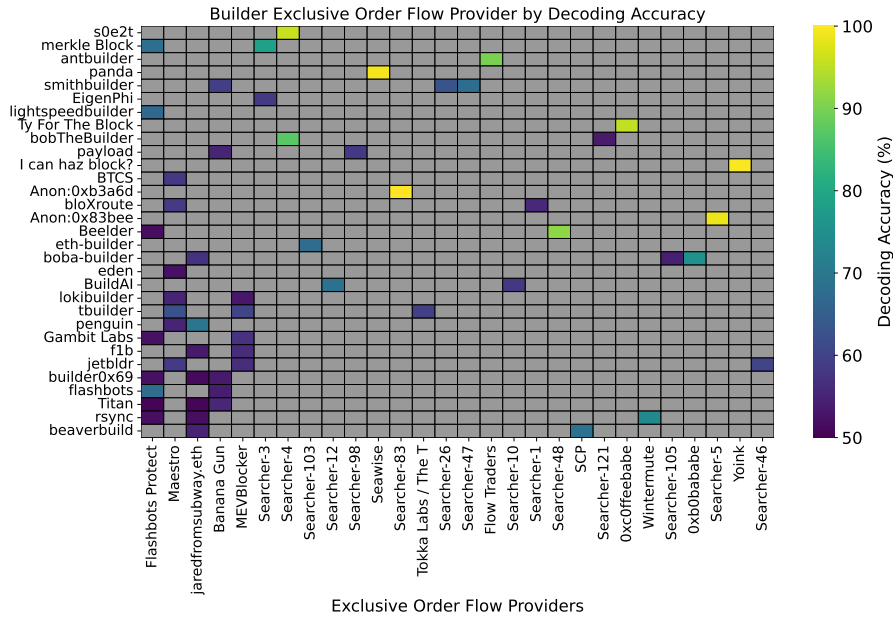
We find that approximately 55% of the builders have at least one EP. These builders have a higher rate of profitable blocks (46%) compared to those without any EP (20%). This result is supported by a Chi-square test on the number of profitable blocks built by the two sets of builders (Chi-square = 33,077.29,  $p < 0.05$ ). Additionally, we find a positive correlation between the rate of profitable blocks and whether builders have an EP, with a coefficient of 0.4 and a p-value of 0.02.

Our results suggests that to produce profitable blocks, builders need exclusive relationships with providers who only supply significant EOF to them. We note that the correlation we measured could have been stronger if EP profits were accounted towards builders, as certain EPs, such as integrated searchers, do not necessarily reflect all the value they extract in their builder payments (see Section 3.4). Interestingly, around 46% of the builders with a negative profit margin (see Table 5 in Appendix B.2 in [77]) have an EP, indicating that the value they subsidize on-chain could already be covered by the profits of their provider.

### 5.3 Key Features for Market Share and Profitability

The MEV-Boost decomposition revealed various features potentially associated with builders’ success in winning blocks and earning profits (see Section 4). We find that, for the top ten builders who produced the most blocks, there is a significant positive correlation between their market share and profit margin (coef = 0.87,  $p < 0.05$ ). Additionally, we identify overlapping features correlated with both metrics (see Figure 10), suggesting that the skills needed to succeed in both areas of the auction are similar.

Exclusive signal, non-atomic arbitrages, and Telegram bot flow strongly positively correlate with both market share (see Figure 10a) and profit margin (see Figure 10b), whereas public signal, including retail swaps, negatively correlates with them. This suggests that high-market share builders receive the more valuable exclusive flows, enabling them to keep significant profits, while those with smaller shares must fill their blocks with less valuable public transactions. For similar reasons, we observe a positive correlation of top ten builders’



■ **Figure 9** Heatmap displaying the DA of EOF providers (x-axis) for each builder (y-axis). Colored cells indicate a statistically significant DA, whereas gray cells show negligible values. Columns with a single colored cell indicate EPs.

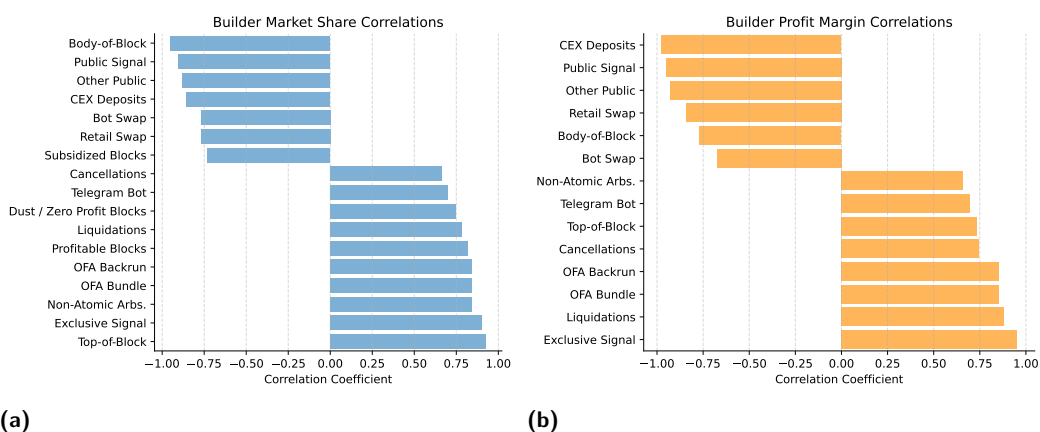
market share with their profitable block rate,<sup>17</sup> and a negative correlation with their subsidy rate. Thus, we encounter the “chicken-and-egg” problem described in [67]: *builders cannot profit from their blocks if they do not have order flow differentiating them from others, and they do not receive such flow unless they have a significant market share.*

Furthermore, top ten builders’ success in gaining market share is strongly positively correlated with the value they derive from order flow in ToB, supporting the findings in [32] (see Figure 10a). Conversely, BoB has the strongest negative correlation. Similar correlations are observed with builders’ profit margin (see Figure 10b). We expect that valuable EOF received by successful builders is prioritized for execution at ToB, allowing them to operate on the latest transitioned state and avoid unexpected changes caused by prior transactions.

Finally, we find that builders’ market share and profit margin are positively correlated with the number of bid cancellations they place per block they win on the UltraSound relay, as shown in Figure 10. This highlights the importance of such bidding strategies during the auction process, allowing builders to lower their active bid, either to avoid making a loss due changing private valuation for the block or to keep higher profits if there is a significant gap with the second highest bid. This is also facilitated by the bid adjustment feature of UltraSound [7]. Interestingly, we do not find a significant correlation with bid update lag, suggesting that latency is not necessarily linked with succeeding in the auction. However, this result may be biased as our latency analysis only considers the blocks relayed through UltraSound. For definitive results, bidding data on other relays must be examined as well.

<sup>17</sup>Correlation between profit margin and profitable block rate is omitted as they are related by definition.

## 22:20 Who Wins Ethereum Block Building Auctions and Why?



**Figure 10** Spearman rank correlations of order flow and builder strategy features for the top ten builders with the highest market share. Figure 10a is a horizontal bar plot showing the significant correlations between features and builders’ market share. Figure 10b is a horizontal bar plot presenting the significant correlations between features and builders’ profit margin.

## 6 Discussion

The current PBS implementation, MEV-Boost, has successfully enabled uniform access to MEV rewards for Ethereum validators, thereby avoiding centralization of the consensus participants. However, the current design of the block building auction promotes competition in EOF access and latency, raising the barrier to entry for new builders and leading to a centralized market (see Section 5). While this centralization is not as detrimental to the protocol’s security as a centralized validator set, it still undermines Ethereum’s censorship resistance [30, 71] and *neutrality* [42] properties.

MEV-Boost block builders are incentivized to access valuable order flow from diverse sources and, notably, from distinguished EPs (see Section 5). The top two builders with the highest market shares, *beaverbuild* and *rsync*, are operated by HFT firms running integrated searchers specializing in non-atomic arbitrages [32, 35]. The third best builder, *Titan*, appears to have an exclusive deal with the Banana Gun Telegram bot starting from February 2024 (see Section 4.3.3). The current market dominance by three players, who build approximately 80% of MEV-Boost blocks (see Section 4.1), raises the barrier of entry for builders, requiring them to secure order flow deals and operate their own searchers to become competitive [67]. Builders unable to do so are forced to adopt subsidy strategies, which are unsustainable in the long run.

Furthermore, builders with a latency advantage due to advanced infrastructure, such as HFT firms, can scale up by running multiple instances and adjusting bid values until the last milliseconds of the block auction (see Section 4.3.4). This gives them an edge in reacting to others’ bids and placing cancellations, at the cost of further raising barriers to entry, misaligning validators’ incentives, and increasing the auction’s gameability [50].

To uphold Ethereum’s censorship resistance and neutrality, it is essential to foster competition and maintain a sufficiently decentralized builder market. There are multiple solution directions for achieving this:

- **Changing Auction Format:** To diminish the impact of latency and bidding strategies, the block auction format can be changed to a sealed-bid auction, eliminating adaptive bidding strategies [75]. However, such an opaque design requires further consideration to



reduce dependency on relay intermediaries for not disclosing bid values while maintaining an efficient auction, secure against collusion between the builders and the relays via side-channels.

- **Decentralized Builders:** Solutions to decentralize the block builder role, such as SUAVE [26],<sup>18</sup> aim to remove social trust in builder by running block building logic in privacy-preserving execution environments like Trusted Execution Environments (TEEs). They can contribute to a less monopolized builder market if they become competitive by attracting a diverse set of order flow providers. Features like gas fee refunds [25], making searchers pay the second-best price of the bundles competing for access to the same state, can also help achieve this.
- **Redistributing and Reducing MEV:** Capturing the MEV extracted by searchers and redistributing it could decrease the value leaked to builders and, eventually, to validators. Solutions for this include MEV-aware decentralized applications (e.g., DEXes [2,14,40,58]), intermediary protocols like OFAs [29,45], and further mechanisms such as MEV taxes [55]. Additionally, encrypted mempools (e.g., Shutter Network [57], SUAVE [26]) where users send encrypted transactions that are decrypted before execution, can prevent censoring and potentially reduce the MEV exposed.
- **In-Protocol Mechanisms:** In-protocol mechanisms such as inclusion lists [10,66] and concurrent proposers [30,49] enforce the involvement of a set of transactions/bundles in a produced block. This could strengthen Ethereum’s censorship resistance properties by reducing the monopoly of a single builder over the block payload, making block production a cooperative activity. Moreover, further unbundling consensus from execution by clearly separating block proposal and attesting duties, referred to as Attestor-Proposer Separation (APS), would allow shielding attestors from centralizing forces. Both execution tickets [17] and execution auctions [8] are proposals in that direction, and are discussed along with inclusion lists to prevent timing games [53,56] without introducing new MEV vectors [19].

Overall, the discussed solutions propose various ways to address the censorship threat posed by the currently centralized block builder market. We hope our study provides valuable insights and considerations for designing further iterations of such mechanisms and Ethereum block auctions, preserving Ethereum’s censorship resistance properties.

## 7 Conclusion

In this paper, we examined the MEV-Boost auction to identify features significant for winning blocks and earning profits. We found that builders’ order flow diversity and access to order flow by EPs are correlated with their block market share and profitability, respectively. Additionally, we showed a positive correlation between market share and profit margin among the top ten builders, suggesting a “chicken-and-egg” problem where builders need order flow distinguishing them from others to profit but only receive such flow if they have a significant market share. Finally, we discussed how the key features we identified for success in MEV-Boost raise the barrier of entry for new builders, driving the builder market towards centralization, and explored existing solutions for preserving Ethereum’s censorship resistance properties, hoping the insights we provide help move the solution space forward.

---

<sup>18</sup>SUAVE is currently under development [24].

## References

- 1 Oxbrainjar. Cross-Chain MEV – Can it Surpass CEX-DEX Arbitrage MEV? - Cross-domain MEV, November 2023. Section: Cross-domain MEV. URL: <https://research.composable.finance/t/cross-chain-mev-can-it-surpass-cex-dex-arbitrage-mev/50>.
- 2 linch. linch Fusion - a new DEX standard. built on top of linch’s Aggregation and Limit Order protocols. URL: <https://linch.io/fusion/>.
- 3 Data Always. Distortion of MEV Auctions by Withdrawals, 2024. URL: <https://hackmd.io/@dataalways/HkUH7hZ26>.
- 4 Ankit Chiplunkar (@ankitchiplunkar). Why should you split the block auction between the top of the block (TOB) and the rest of the block (ROB)? URL: <https://x.com/ankitchiplunkar/status/1687806136747966464>.
- 5 Maryam Bahrani, Pranav Garimidi, and Tim Roughgarden. Centralization in block building and proposer-builder separation. doi:10.48550/arXiv.2401.12120.
- 6 Binance. Binance - Cryptocurrency Exchange for Bitcoin, Ethereum & Altcoins. URL: <https://www.binance.com/en>.
- 7 Niclas Blomberg. Bid adjustment, 2023. URL: <https://gist.github.com/blombern/c2550a5245d8c2996b688d2db5fd160b>.
- 8 John Burian and David Crapis. Execution auctions as an alternative to execution tickets - proof-of-stake / block proposer, 2024. Section: Proof-of-Stake. URL: <https://ethresear.ch/t/execution-auctions-as-an-alternative-to-execution-tickets/19894>.
- 9 Vitalik Buterin. Proposer/block builder separation-friendly fee market designs - economics, 2021. Section: Economics. URL: <https://ethresear.ch/t/proposer-block-builder-separation-friendly-fee-market-designs/9725>.
- 10 Vitalik Buterin and Mike Neuder. No free lunch – a new inclusion list design - proof-of-stake, 2023. Section: Proof-of-Stake. URL: <https://ethresear.ch/t/no-free-lunch-a-new-inclusion-list-design/16389>.
- 11 Agostino Capponi, Ruizhe Jia, and Brian Zhu. The Paradox Of Just-in-Time Liquidity in Decentralized Exchanges: More Providers Can Sometimes Mean Less Liquidity, February 2024. arXiv:2311.18164 [cs, q-fin]. doi:10.48550/arXiv.2311.18164.
- 12 Coinbase. Coinbase - Buy & Sell Bitcoin, Ethereum, and more with trust. URL: <https://www.coinbase.com/en-de/>.
- 13 Etienne Combrisson and Karim Jerbi. Exceeding chance level by chance: The caveat of theoretical chance levels in brain signal classification and statistical assessment of decoding accuracy. *Journal of Neuroscience Methods*, 250:126–136, 2015. doi:10.1016/j.jneumeth.2015.01.010.
- 14 CoW Protocol. URL: <https://cow.fi/>.
- 15 Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 910–927, 2020. doi:10.1109/SP40000.2020.00040.
- 16 Francesco D’amato and Mike Neuder. Equivocation attacks in mev-boost and ePBS - proof-of-stake, 2023. Section: Proof-of-Stake. URL: <https://ethresear.ch/t/equivocation-attacks-in-mev-boost-and-epbs/15338>.
- 17 Justin Drake and Mike Neuder. Execution tickets - proof-of-stake / economics, 2023. Section: Proof-of-Stake. URL: <https://ethresear.ch/t/execution-tickets/17944>.
- 18 Dune. Dune — Crypto Analytics Powered by Community. URL: <https://dune.com/home>.
- 19 Anders Elowsson. MEV resistant dynamic pricing auction of execution proposal rights - proof-of-stake / block proposer, 2024. Section: Proof-of-Stake. URL: <https://ethresear.ch/t/mev-resistant-dynamic-pricing-auction-of-execution-proposal-rights/20024>.
- 20 Ethereum. The merge. URL: <https://ethereum.org/en/roadmap/merge/>.
- 21 Ethereum. Staking withdrawals. URL: <https://ethereum.org/en/staking/withdrawals/>.

- 22 Ethereum. ethereum/consensus-specs, April 2024. original-date: 2018-09-20T05:12:54Z. URL: <https://github.com/ethereum/consensus-specs>.
- 23 Flashbots. flashbots/mev-boost. original-date: 2021-11-17T18:55:22Z. URL: <https://github.com/flashbots/mev-boost>.
- 24 Flashbots. flashbots/suave-specs. original-date: 2023-09-29T11:20:43Z. URL: <https://github.com/flashbots/suave-specs>.
- 25 Flashbots. Gas fee refunds | flashbots docs. URL: <https://docs.flashbots.net/flashbots-auction/advanced/gas-fee-refunds>.
- 26 Flashbots. The future of MEV is SUAVE | flashbots writings, 2022. URL: <https://writings.flashbots.net/the-future-of-mev-is-suave>.
- 27 Flashbots. MEV-Boost Block Proposal | Flashbots Docs, March 2024. URL: <https://docs.flashbots.net/flashbots-mev-boost/architecture-overview/block-proposal>.
- 28 Flashbots. MEV Protection Overview | Flashbots Docs, March 2024. URL: <https://docs.flashbots.net/flashbots-protect/overview>.
- 29 Flashbots. MEV-Share | Flashbots Docs, March 2024. URL: <https://docs.flashbots.net/flashbots-protect/mev-share>.
- 30 Elijah Fox, Malleh Pai, and Max Resnick. Censorship Resistance in On-Chain Auctions, June 2023. arXiv:2301.13321 [cs, econ]. doi:10.48550/arXiv.2301.13321.
- 31 Banana Gun. Trade Crypto the Banana Way | Banana Gun. URL: <https://bananagun.io/>.
- 32 Tivas Gupta, Malleh M. Pai, and Max Resnick. The Centralizing Effects of Private Order Flow on Proposer-Builder Separation. In Joseph Bonneau and S. Matthew Weinberg, editors, *5th Conference on Advances in Financial Technologies (AFT 2023)*, volume 282 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 20:1–20:15, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.AFT.2023.20.
- 33 Chris Hager. Mempool-dumpster. URL: <https://github.com/flashbots/mempool-dumpster>.
- 34 Lioba Heimbach, Lucianna Kiffer, Christof Ferreira Torres, and Roger Wattenhofer. Ethereum’s proposer-builder separation: Promises and realities. In *Proceedings of the 2023 ACM on Internet Measurement Conference, IMC ’23*, pages 406–420, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3618257.3624824.
- 35 Lioba Heimbach, Vabuk Pahari, and Eric Schertenleib. Non-atomic arbitrage in decentralized finance. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 224–224, Los Alamitos, CA, USA, May 2024. IEEE Computer Society. doi:10.1109/SP54263.2024.00256.
- 36 hildobby. Atomic arbitrage transactions query. URL: [https://github.com/duneanalytics/spellbook/blob/main/macros/models/\\_sector/dex/dex\\_atomic\\_arbitrages.sql](https://github.com/duneanalytics/spellbook/blob/main/macros/models/_sector/dex/dex_atomic_arbitrages.sql).
- 37 hildobby. Cex deposit wallets query. URL: <https://dune.com/queries/3237025>.
- 38 hildobby. Frontruns and backruns in sandwiching bundles query. URL: [https://github.com/duneanalytics/spellbook/blob/main/macros/models/\\_sector/dex/dex\\_sandwiches.sql](https://github.com/duneanalytics/spellbook/blob/main/macros/models/_sector/dex/dex_sandwiches.sql).
- 39 hildobby. Sandwiched victim transactions query. URL: [https://github.com/duneanalytics/spellbook/blob/main/macros/models/\\_sector/dex/dex\\_sandwiched.sql](https://github.com/duneanalytics/spellbook/blob/main/macros/models/_sector/dex/dex_sandwiched.sql).
- 40 Uniswap Labs. Uniswap x: A decentralized trading protocol, 2024. Accessed: 2024-05-23. URL: <https://uniswap.org/whitepaper-uniswapx.pdf>.
- 41 Jianhua Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, 1991. doi:10.1109/18.61115.
- 42 Julian Ma, Barnabé Monnot, and Thomas Thiery. Uncrowdable inclusion lists: The tension between chain neutrality, preconfirmations and proposer commitments - economics, 2024. Section: Economics. URL: <https://ethresear.ch/t/uncrowdable-inclusion-lists-the-tension-between-chain-neutrality-preconfirmations-and-proposer-commitments/19372/5>.
- 43 Maestro. Maestro - The Leading Telegram Sniping and trading Bot for DeFi. URL: <https://www.maestrobots.com/>.
- 44 Dan Marzec and Dan Thibault. Subverting the total eclipse (of the heart), 2023. URL: <https://hackmd.io/@dmarz/total-eclipse-of-the-relay>.

## 22:24 Who Wins Ethereum Block Building Auctions and Why?

- 45 MEV Blocker. URL: <https://mevblocker.io>.
- 46 Mike Neuder. optimistic-relay-documentation/proposal.md at main · michaelneuder/optimistic-relay-documentation. <https://github.com/michaelneuder/optimistic-relay-documentation/blob/main/proposal.md>.
- 47 Mike Neuder. ePBS – the infinite buffet, 2023. URL: <https://notes.ethereum.org/@mikeneuder/infinite-buffet>.
- 48 Mike Neuder and Justin Drake. Why enshrine proposer-builder separation? a viable path to ePBS – proof-of-stake, 2023. URL: <https://ethresear.ch/t/why-enshrine-proposer-builder-separation-a-viable-path-to-epbs/15710/7>.
- 49 Mike Neuder and Max Resnick. Concurrent block proposers in ethereum – proof-of-stake / block proposer, 2024. Section: Proof-of-Stake. URL: <https://ethresear.ch/t/concurrent-block-proposers-in-ethereum/18777>.
- 50 Mike Neuder, Thomas Thiery, and Chris Hager. Bid cancellations considered harmful – Proof-of-Stake, May 2023. Section: Proof-of-Stake. URL: <https://ethresear.ch/t/bid-cancellations-considered-harmful/15500>.
- 51 Alexandre Obadia, Alejo Salles, Lakshman Sankar, Tarun Chitra, Vaibhav Chellani, and Philip Daian. Unity is strength: A formalization of cross-domain maximal extractable value, 2021. [arXiv:2112.01472](https://arxiv.org/abs/2112.01472).
- 52 OreoMev. Block builder profitability – research – data, 2023. Section: Data. URL: <https://collective.flashbots.net/t/block-builder-profitability-research/2803>.
- 53 Burak Öz, Benjamin Kraner, Nicolò Vallarano, Bingle Stegmann Kruger, Florian Matthes, and Claudio Juan Tessone. Time moves faster when there is nothing you anticipate: The role of time in mev rewards. In *Proceedings of the 2023 Workshop on Decentralized Finance and Security, DeFi '23*, pages 1–8, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3605768.3623563.
- 54 Kaihua Qin, Liyi Zhou, and Arthur Gervais. Quantifying blockchain extractable value: How dark is the forest? In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 198–214. IEEE, 2022.
- 55 Dan Robinson and David White. Priority is all you need, 2024. URL: <https://www.paradigm.xyz/2024/06/priority-is-all-you-need>.
- 56 Caspar Schwarz-Schilling, Fahad Saleh, Thomas Thiery, Jennifer Pan, Nihar Shah, and Barnabé Monnot. Time Is Money: Strategic Timing Games in Proof-Of-Stake Protocols. In Joseph Bonneau and S. Matthew Weinberg, editors, *5th Conference on Advances in Financial Technologies (AFT 2023)*, volume 282 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 30:1–30:17, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.AFT.2023.30.
- 57 Shutter. URL: <https://shutter.network/>.
- 58 Sorella. URL: <https://sorella-website.vercel.app/>.
- 59 Danning Sui. Labeled Transactions Query. URL: <https://dune.com/queries/3317765>.
- 60 Danning Sui. Builder Public Keys, 2024. URL: [https://docs.google.com/spreadsheets/d/1rPR4ZcPA0fLzBN4Lk\\_Ja2X8h6KkdINJBZy6245D1joo](https://docs.google.com/spreadsheets/d/1rPR4ZcPA0fLzBN4Lk_Ja2X8h6KkdINJBZy6245D1joo).
- 61 Danning Sui and Angela Lu. Orderflow.art | Illuminating Ethereum’s order flow landscape. URL: <https://orderflow.art/methodology>.
- 62 Danning Sui and Burak Öz. Exclusive Transaction Addresses, 2024. URL: <https://gist.github.com/boz1/a86ef8853c2ad03b86ba679e9f4cfee9>.
- 63 Espresso Systems. Fair exchange in proposer-builder separation - HackMD, 2013. URL: <https://hackmd.io/@EspressoSystems/fair-exchange-in-proposer-builder-separation>.
- 64 thegostep. Block scoring for mev-boost relays - relays, 2022. Section: Relays. URL: <https://collective.flashbots.net/t/block-scoring-for-mev-boost-relays/202>.
- 65 Thomas Thiery. Empirical analysis of builders’ behavioral profiles (BBPs) - economics, 2023. URL: <https://ethresear.ch/t/empirical-analysis-of-builders-behavioral-profiles-bbps/16327>.

- 66 Thomas Thiery, Francesco D'amato, and Julian Ma. Fork-choice enforced inclusion lists (FO-CIL): A simple committee-based inclusion list proposal - proof-of-stake / block proposer, 2024. Section: Proof-of-Stake. URL: <https://ethresear.ch/t/fork-choice-enforced-inclusion-lists-focil-a-simple-committee-based-inclusion-list-proposal/19870>.
- 67 Titan. Builder Dominance and Searcher Dependence, 2023. URL: <https://frontier.tech/builder-dominance-and-searcher-dependence>.
- 68 Ultra Sound Relay. URL: <https://relay.ultrasound.money/>.
- 69 Anton Wahrstätter. Censorship.pics. URL: <https://censorship.pics/censorship.pics>.
- 70 Anton Wahrstätter. MEV-boost. URL: <https://mevboost.pics/mevboost.pics>.
- 71 Anton Wahrstätter, Jens Ernstberger, Aviv Yaish, Liyi Zhou, Kaihua Qin, Taro Tsuchiya, Sebastian Steinhorst, Davor Svetinovic, Nicolas Christin, Mikolaj Barczentewicz, and Arthur Gervais. Blockchain Censorship, June 2023. arXiv:2305.18545 [cs]. doi:10.48550/arXiv.2305.18545.
- 72 Anton Wahrstätter, Liyi Zhou, Kaihua Qin, Davor Svetinovic, and Arthur Gervais. Time to bribe: Measuring block construction market. doi:10.48550/arXiv.2305.16468.
- 73 whalehunter. Ethereum telegram bot trades query. URL: <https://dune.com/queries/3375587>.
- 74 Winnie. Searcher Builder Dashboard. URL: <https://www.searcherbuilder.pics/>.
- 75 Fei Wu, Thomas Thiery, Stefanos Leonardos, and Carmine Ventre. Strategic bidding wars in on-chain auctions, 2023. doi:10.48550/arXiv.2312.14510.
- 76 Sen Yang, Kartik Nayak, and Fan Zhang. Decentralization of Ethereum's Builder Market, May 2024. arXiv:2405.01329 [cs]. arXiv:2405.01329.
- 77 Burak Öz, Danning Sui, Thomas Thiery, and Florian Matthes. Who Wins Ethereum Block Building Auctions and Why?, July 2024. arXiv:2407.13931 [cs]. arXiv:2407.13931.



# A Shortfall in Investor Expectations of Leveraged Tokens

Reza Rahimian  

Concordia University, Montreal, Canada

Jeremy Clark   

Concordia, University, Montreal, Canada

---

## Abstract

Leveraged tokens (LVTs) are emerging crypto-assets primarily issued by centralized exchanges. The concept is borrowed from leveraged ETFs (LETFs) in traditional financial markets, which offer higher gains (and higher losses) relative to price movements in the underlying asset. Leverage is commonly used by short-term traders to amplify returns from daily market shifts. However, LVTs have been implemented differently from LETFs by exchanges in the crypto market, with variations across platforms. We examine the mechanics and constituent components of LVTs, demonstrating that the lack of a standard has resulted in deficiencies and unexpected technical and economic outcomes. To identify existing problems, we analyze more than 1,600 leveraged tokens from 10 issuers. Our analysis reveals that 99.9% of LVTs are centralized, with 80% lacking blockchain interaction, leading to transparency issues. Total supply information is difficult to access for 53% of them, and 41% appear inadequately backed at launch. Additionally, 97% of LVTs are vulnerable to front-running during well-known events, and they deviate from their stated leverage ratios more than LETFs, partly due to inconsistent re-leveraging processes and higher management fees. This work provides a framework for crypto investors, blockchain developers, and data analysts to gain a deep understanding of leveraged tokens and their impact on market dynamics, liquidity, and price movements. It also offers insights for crypto exchanges and auditors into the internal functionalities and financial performance of LVTs under varying market conditions.

**2012 ACM Subject Classification** Security and privacy

**Keywords and phrases** crypto-assets, ethereum, leverage, derivatives

**Digital Object Identifier** 10.4230/LIPIcs.AFT.2024.23

**Related Version** *Full Paper*: <https://ssrn.com/abstract=4899657> [40]

**Funding** *Jeremy Clark*: Acknowledges support for this research project from (i) the National Sciences and Engineering Research Council (NSERC), Raymond Chabot Grant Thornton, and Catalaxy Industrial Research Chair in Blockchain Technologies (IRCPJ/545498-2018), (ii) an NSERC Discovery Grant (RGPIN/04019-2021), and (iii) an Autorité des Marchés Financiers (AMF) Research Grant.

**Acknowledgements** Reviewers' comments were very helpful and addressed in the final version.

## 1 Introduction

A typical Exchange-Traded Fund (ETF) is a weighted basket of stocks from firms with a common characteristic (e.g., they all operate in a specific sector or have a high market capitalization). The issuer splits the basket into shares, which are bought and sold on exchanges just like individual stocks [32].

► **Example 1.** One of the most widely traded ETFs is the *SPDR S&P500 ETF*, with the



© Reza Rahimian and Jeremy Clark;  
licensed under Creative Commons License CC-BY 4.0  
6th Conference on Advances in Financial Technologies (AFT 2024).

Editors: Rainer Böhme and Lucianna Kiffer; Article No. 23; pp. 23:1–23:24

Leibniz International Proceedings in Informatics



LIPIC Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 23:2 A Shortfall in Investor Expectations of Leveraged Tokens

ticker symbol *SPY*. It is issued by SSGA<sup>1</sup> and holds a basket of stocks from nearly 500 publicly traded companies that are included in the S&P500<sup>2</sup> index. The S&P500 index has globally served as a gauge for the performance of the U.S. stock market as a whole, due to its depth and diversity. Since *SPY* tracks the S&P500 index, investors can gain broad exposure and diversify their investment risk across the stock performance of 500 companies in 11 sectors without the logistics or starting capital required to buy shares in all these companies.

Leveraged ETFs (LETFs) were introduced in 2006 and are ETFs designed to amplify the daily performance of the underlying basket (more on leverage in Section 3.1).<sup>3</sup> Inverse LETFs aim to achieve a return that is a multiple of the inverse of the underlying asset's daily performance [25, 11, 42]. Many investors alternatively refer to LETFs and inverse LETFs as “Bullish” and “Bearish” LETFs, respectively, reflecting their short-term sentiment on future price movements.

► **Example 2.** *Direxion Daily S&P500 Bull 3x ETF (SPXL)* is a 3x (three times) LETF that seeks to deliver triple the daily performance of the S&P500. It magnifies each 1% gain in the S&P500 index into a 3% gain and loses 3% for every 1% drop in the index. *Direxion Daily S&P500 Bear 3x ETF (SPXS)* delivers triple the opposite daily performance of the S&P500 index. If the S&P500 index depreciates by 1%, *SPXS* gains 3%, and vice versa [52, 30].

A Leveraged Token (LVT) in the cryptocurrency and crypto-asset (“crypto”) market can be compared to a Leveraged ETF (LETF) in the traditional financial market. Similar to LETFs, LVTs use leveraged products available in the crypto market to outperform the underlying asset's return on a daily basis. While the majority of LETFs are actively managed funds<sup>4</sup>, LVTs employ one of three management models: (i) centralized, (ii) decentralized, and (iii) hybrid. Centralized LVTs are primarily managed by crypto exchanges and can be purchased on the spot market or directly from the exchange (*cf.* Appendix A.5 of the full version [40] on investing in LVTs). Decentralized LVTs operate on-chain and can be traded by interacting directly with the smart contract. Hybrid LVTs are essentially decentralized LVTs that are traded on centralized crypto exchanges. Users trade on centralized exchanges for their user-friendly interfaces, continuous-time order books (rather than automated market makers, which are the only trading mechanism efficient enough to run on-chain), and increased liquidity due to aggregated buy and sell orders.<sup>5</sup> However, this model introduces certain disadvantages resulting from the combination of centralized and decentralized systems (*e.g.* functional complexities, security concerns, custodial risks, *etc.*).

► **Example 3.** An issuer may offer *BTC3L/BTC3S* as a pair of LVTs tracking Bitcoin (BTC) as the underlying asset. A Bitcoin futures contract (BTC-Perp<sup>6</sup>) can be used as the leveraged product to outperform Bitcoin in the short term. The number three in the LVT name

---

<sup>1</sup> State Street Bank and Trust Company (SSGA) is one of the three dominant companies in the ETF market, with a 14.01% market share, following BlackRock and Vanguard, which have 33.64% and 29.16%, respectively [46].

<sup>2</sup> The S&P 500 index comprises 500 of the top publicly traded companies in the U.S. It was launched in 1957 by the credit rating agency Standard and Poor's [26].

<sup>3</sup> The underlying asset can be stocks, market indexes (e.g., S&P 500, NASDAQ-100, etc.), commodities (e.g., gold, oil, corn, etc.), or any asset with a price.

<sup>4</sup> In actively managed funds, investment managers actively buy and sell assets with the goal of outperforming a specified benchmark index, resulting in higher management fees.

<sup>5</sup> The more liquid an asset is, the easier and more efficient it is to convert back into cash. Less liquid assets take more time and may incur higher costs [24].

<sup>6</sup> A type of Bitcoin futures contract without a defined expiration date (known as a “perpetual” contract).



■ **Table 1** *Left table*: Number of issued leveraged tokens per year, average per year, and number of unique underlying assets, which we collected manually from different sources. An underlying asset might be used to create multiple tokens with different leverage levels. *Right table*: Characteristics of issued LVTs by different issuers. Only 20% of tokens have been created on the blockchain. 99.9% of LVTs use derivatives as the leveraged product, which is offered by the same issuer (Internal for Pionex as of Jan 2023). Except for *Index Coop*, the rest of the issuers use off-chain fund management systems. Rebalancing triggers for *Index Coop* are still off-chain.

Issuer	Number of LVTs per year							Underlying Assets	Characteristics of LVTs			
	2019	2020	2021	2022	2023	Total	Average		Fund Source	Blockchain Rep.	Fund Management Algorithm	Leveraged Product
MEXC		162	116	102	76	456	114	217				
AscendEX				228	112	340	170	94				
Gate.io		116	96	54	8	274	69	123				
Pionex		60	102	36	2	200	50	76				
FTX	102	27				129	65	43				
KuCoin			50	6	38	94	31	45				
Binance		38	2			40	20	20				
ByDFi				16	24	40	20	20				
ByBit				34		34	34	17				
Index Coop			2			2	2	2				
<b>Total</b>	<b>102</b>	<b>403</b>	<b>368</b>	<b>476</b>	<b>260</b>	<b>1609</b>	<b>322</b>	<b>654</b>				

Fund Source	Blockchain Rep.	Fund Management Algorithm	Leveraged Product
Internal	No	Off-Chain	Futures
Internal / External	Yes		
Internal	No		
External	Yes	On-Chain / Off-Chain	Debt

represents the multiplier (triple-leveraged), while L/S stands for going long/short on the market.<sup>7</sup> BTC3L gains 3% when the price of Bitcoin rises by 1%, and loses 3% for every 1% price drop. Conversely, when Bitcoin drops by 1%, BTC3S gains 3%, and loses 3% for every 1% price rise.

Since 2019, more than 1,600 LVTs have been issued by various crypto exchanges. The FTX exchange introduced the original concept by issuing 102 tokens on the blockchain.<sup>8</sup> Trading volumes exceeded \$1 million per day [15]. This upward trend has continued, with other exchanges issuing approximately 32 new LVTs per month on average from January 2020 to November 2023 (see Table 1).

### Motivation for studying LVTs

- *LVT attractiveness for investors*: Investment in LETFs nearly doubled in 2022 compared to 2021 [48], demonstrating an appetite for low-risk leverage, which is satisfied in the crypto market by LVTs. LVTs reduce liquidation risks compared to derivatives and margin trading. However, other characteristics (e.g., volatility drag) must be understood to avoid unexpected value destruction. These risks are not unique to LVTs; they also exist in LETFs.<sup>9</sup> (See the full version [40], Appendix A.5, and comparative Table 6 for more details on why investors are attracted to this type of token.)
- *LVT distinctive dynamics*: Section 2 offers a cohesive framework for understanding key aspects of LVTs, such as their underlying dynamics, peculiarities in product design, effects on crypto markets, and investor suitability. Leveraged products can impact market dynamics, especially in highly volatile markets [44]. More technical details are provided in Section 3, which can be useful for those involved in the design and implementation of LVTs to understand how these tokens affect liquidity and price movements, potentially influencing the robustness and reliability of trading algorithms.

<sup>7</sup> Going long refers to buying an asset with the expectation that its value will increase, allowing it to be sold for a profit later. Conversely, going short refers to profiting from a decline in the asset's value [27].

<sup>8</sup> The ERC-20 standard is the most prominent standard for fungible tokens on the Ethereum blockchain. These tokens can represent financial assets such as LVTs and can be exchanged between users.

<sup>9</sup> In 2018, Credit Suisse had to close an ETF ETN after its price plunged 90% in one day. In another example, WisdomTree had to close its 3x oil products in March 2020 after their value was wiped out [48].

## 23:4 A Shortfall in Investor Expectations of Leveraged Tokens

- *Regulatory implications:* LVTs introduce new risks for market regulation, investor protection, and financial stability. Our work contributes to broader discussions on how to effectively regulate emerging financial technologies like LVTs. Additionally, since LVTs are often held by commercial firms requiring audited financial statements [17], auditors should understand how LVTs function, their risks, and how they perform under different market conditions.

### Contributions of this paper

In this work, we study more than 1,600 leveraged tokens from 10 issuers, examining various aspects such as underlying assets, interaction with the blockchain, types of leveraged products, and fund management algorithms. We dedicate part of the paper to carefully explaining LVT mechanics and constituent components to help the reader understand the functionality of leveraged funds, rebalancing mechanisms, and smart contracts. We then address six research questions about LVTs:

- RQ 1: What information is visible to traders of an LVT?
- RQ 2: To what extent are LVTs locked to the offering exchange?
- RQ 3: Are the LVTs offered today adequately backed?
- RQ 4: What are the possibilities of front-running in LVTs?
- RQ 5: How well do LVTs track their asserted leverage ratios?
- RQ 6: Are LVT fees in-line with traditional LETFs?

### Methodology of measurements and dataset

To extract the list of issuers, we identified the top 100 crypto exchanges based on 24-hour trading volume, as reported by crypto comparison websites.<sup>10</sup> Then, by visiting each provider’s website, we manually checked whether LVTs were offered. We supplemented this with searches on Google<sup>11</sup>, online forums and blogs<sup>12</sup>, and crypto news sites<sup>13</sup>. The combined list should be comprehensive as of 2023 and includes LVTs from various types of exchanges (both large and small, centralized and decentralized), covering different asset classes.

The majority of LVTs exhibited common elements that allowed for formal representation in Section 2. We reviewed each LVT’s documentation to understand its functionality [22, 3, 16, 38, 39, 29, 5, 7, 6, 14], enabling us to identify their components as discussed in Section 3. Outliers that did not fit into the typical LVT model due to unique structural features were not excluded (e.g., Hybrid LVTs by FTX). Instead, the model was generalized to include these 6% of outliers. We have described the parts of the model that were extended for this category (e.g., the smart contract component for decentralized LVTs).

In Section 4, the functionality of tokens is evaluated by answering six research questions. To address these, we collected data from both the token issuers and historical data available through various exchanges and financial databases. Issuer documentation provided crucial details on the structure, mechanics, and intended use of LVTs. Historical data were gathered from reputable financial data providers<sup>14</sup>, including price histories, trading volumes, issuance dates, and other relevant metrics. Additionally, we cross-referenced data from multiple sources

---

<sup>10</sup> Websites: [coinecko.com](https://coinecko.com), [coinmarketcap.com](https://coinmarketcap.com), [cryptocompare.com](https://cryptocompare.com), and [coinranking.com](https://coinranking.com).

<sup>11</sup> Search terms: leveraged tokens, leveraged ETFs cryptocurrency, leveraged crypto assets, crypto leverage trading platforms, crypto leverage trading token issuers, etc.

<sup>12</sup> Crypto blogs and forums such as [reddit.com](https://reddit.com) (e.g., /cryptocurrency or /binance subreddits).

<sup>13</sup> News sources: [coindesk.com](https://coindesk.com) and [cointelegraph.com](https://cointelegraph.com).

<sup>14</sup> Data sources: [tradingview.com](https://tradingview.com), [cryptodatadownload.com](https://cryptodatadownload.com), [etherscan.io](https://etherscan.io), and [finance.yahoo.com](https://finance.yahoo.com).

to validate the accuracy and consistency of our dataset. It is worth mentioning that the analyzed data from sources (e.g., [tradingview.com](https://tradingview.com)) are aggregated directly from the source exchanges, ensuring the information is accurate and up-to-date. The direct connection to exchanges means that the data reflect real-time market conditions and historical performance accurately. Moreover, these data sources are widely used and trusted within the financial and cryptocurrency communities, providing tools and data to a large number of traders, analysts, and researchers, underscoring their credibility.

### Related work

To our knowledge, this is the first academic paper on LVTs; however, our work overlaps with studies on LETFs, which have established the following research findings:

- The effect of compounded returns intensifies with longer holding periods, causing LETFs to struggle to maintain their stated leverage over time. As a result, long-term performance is not linearly related to the return of the underlying asset [50, 34, 23, 12, 9].
- LETFs can underperform over longer periods without efficient rebalancing. Researchers state that frequent rebalancing during periods of high volatility is necessary to maintain leveraged exposure to the tracking index. They conclude that reducing rebalancing frequency can significantly decrease tracking errors [9, 8, 18].
- The impact of LETFs on market volatility and liquidity shows that their daily rebalancing can increase volatility and trading volume near the market close, potentially distorting the market price of LETFs and creating additional inefficiencies [12, 23, 9, 49, 41, 44].
- Investors often do not fully understand the mechanisms, risks, and proper uses of LETFs, which require tolerating increased risk. Consequently, LETFs are suitable only for experienced and skilled investors who comprehend the complexities and hazards of trading with them [9, 18, 31].

## 2 Price and Return Dynamics

As daily returns is embedded in the design of LVTs, a  $k$ -leveraged LVT should generally earn  $k$  times of the daily return of the underlying. The amplification ratio, known as leverage ( $k$ ), can be fixed or dynamic. A proportional change in the underlying price is  $k$ -times the proportional change in LVT price. In the short-term, LVT return is consistent with  $k$ , but beyond a single day, the return is highly path dependent, making LVTs unsuitable for buy-and-hold strategies. This is an issue that is ignored by most retail investors, leading to unexpected loss of capital. In the following, the price and return dynamics of LVTs have been discussed, aiding analysis and simulation of such issues.

### 2.1 LVT Price Dynamics

LVTs are intentionally designed with leverage as a core component of their architecture. They are aimed at outperforming the return of the underlying benchmark on a daily basis.<sup>15</sup> Let  $P_{t_n}$  represent the LVT price at calendar time  $t_n$ , expressed as:

$$P_{t_n} = P_{t_{n-1}} \left( 1 + k \frac{\Delta S_{t_n}}{S_{t_{n-1}}} \right) \quad n \in [1, 365], t \geq 0, k \in [-5, -0.5] \cup [0.5, 5] \quad (1)$$

<sup>15</sup>Returns will be slightly lower after deducting fund management fees, accounting for market volatility, interest paid on borrowing, and other associated expenses.

$S_{t_n}$  is the underlying price at time  $t_n$ , indexed by  $n$ , where  $n$  denotes the days of the year. The frequency of  $n$  does not have to be daily; it can be redefined in hours or minutes without any loss of generality. However, since daily returns are embedded in the LVT product design,  $n$  is effectively daily.  $P_{t_n}$  represents the price of the LVT at the close of trading day  $n$ .  $S_{t_{n-1}}$  and  $P_{t_{n-1}}$  are the initial prices of the underlying asset and LVT, respectively, at the beginning of trading day  $n$  (or at the end of trading day  $n - 1$ ).  $\Delta S_{t_n}$  is the amount of change in the underlying price relative to the initial price. The constant variable  $k$  is the LVT multiplier (leverage), which can be defined as either a fixed or dynamic value, depending on the issuer. LVTs with fixed leverage can take values from the set  $\{-5, -3, -2, -1, -0.5, 0.5, 2, 3, 5\}$ , while dynamic leverage fluctuates within the range  $[-4.0, -1.25] \cup [1.25, 4.0]$ .

The multiplier  $k$  further divides LVTs into three main functional groups: (i) Long LVTs, where  $k \in \{2, 3, 5\} \cup [1.25, 4.0]$ . The value of a long LVT rises  $k$  times faster than the underlying asset and is profitable in rising markets; (ii) Short LVTs, where  $k \in \{-5, -3, -2, -1\} \cup [-4.0, -1.25]$ . The value of a short LVT rises  $|k|$  times faster than the underlying asset and is profitable in falling markets. This type of LVT is also used to hedge<sup>16</sup> positions or as a substitute for short-selling the underlying asset; (iii) Low-risk LVTs, where  $k \in \{-0.5, 0.5\}$ . Low-risk LVTs can be used to reduce the impact of adverse market movements without the full risks associated with higher leverage factors. Every 1% change in the underlying leads to a 0.5% change in the price of a low-risk LVT (*i.e.* less profit with less risk). Equation (1) indicates a linear relationship between the LVT price and the underlying price from time  $t_{n-1}$  to  $t_n$ . It can be shown algebraically that:

$$P_{t_n} = P_{t_{n-1}} + k P_{t_{n-1}} \left( \frac{\Delta S_{t_n}}{S_{t_{n-1}}} \right) \Rightarrow \frac{\Delta P_{t_n}}{P_{t_{n-1}}} = k \left( \frac{\Delta S_{t_n}}{S_{t_{n-1}}} \right) \quad (2)$$

When  $n$  and  $n - 1$  are close enough to each other in equation (2), the proportional change in the LVT price relative to its initial price equates to the proportional change in the underlying price relative to its initial value. Equation (2) then becomes:

$$\frac{dP_t}{P_t} = k \frac{dS_t}{S_t} \quad (3)$$

If there is not enough initial capital for the fund, the issuer may borrow  $(k - 1)P_t$  from external financial sources at an interest rate of  $r \geq 0$ . Considering  $f$  as the expense ratio of the fund, equation (3) can be completed as:

$$\frac{dP_t}{P_t} = k \frac{dS_t}{S_t} - ((k - 1)r + f)dt \quad (4)$$

As will become clear later in Section 2.2, equation (4) does not represent the return of an LVT; rather, it only shows the LVT price change relative to the underlying for discrete time intervals. Discrete time is commonly used in financial modeling, where prices are calculated at specific intervals, such as daily, hourly, or minutely. Therefore, equation (1) can be used, for example, to calculate the LVT price change on trading day 2 compared to day 1. The price of an LVT at any arbitrary interval can also be modeled in continuous time, as detailed in Appendix A.12 of the full version [40]. However, for the modeling of LVTs in this work, discrete time is more appropriate, reducing unnecessary complications.

<sup>16</sup> Hedging is a common practice to reduce the risk of adverse price movements in another position. For example, opening a \$100K long Bitcoin position and hedging it by buying a \$20K 2x short Bitcoin LVT. If the long position experiences a 5% price drop, the net loss is partially offset by gains from the hedge position, resulting in a net loss of  $(2 \times 5\% \times \$20K) - (5\% \times \$100K) = \$2K - \$5K = -\$3K$ , which is less than the  $-\$5K$  loss without hedging.

## 2.2 LVT Return Dynamics

The return of LVTs cannot simply be considered as  $k$  times the return of the underlying asset. Let  $R_{t_{(n-1)} \rightarrow n}$  represent the return of the underlying asset (in percentage) at price  $S$  from time  $t_{n-1}$  to  $t_n$ :

$$R_{t_{(n-1)} \rightarrow n} = \frac{\Delta S_{t_n}}{S_{t_{n-1}}} \quad n \geq 1, t \geq 0, t_0 = 0 \quad (5)$$

Considering the relationship between the price of LVT and the underlying in (3), the return of the LVT from time  $t_{n-1}$  to  $t_n$  can be expressed as  $G_{t_{(n-1)} \rightarrow n} = 1 + kR_{t_{(n-1)} \rightarrow n}$ . Since daily return is embedded in the design of LVTs, the frequency of  $n$  here is daily. In the short run (a trading day), where the underlying volatility is almost constant, the change in the LVT price relative to the underlying can be assumed to be equivalent to  $k$ -times the return. However, in the long run (several trading days or weeks), LVT return may be significantly lower or higher than the underlying due to the compounding effect, as given by:

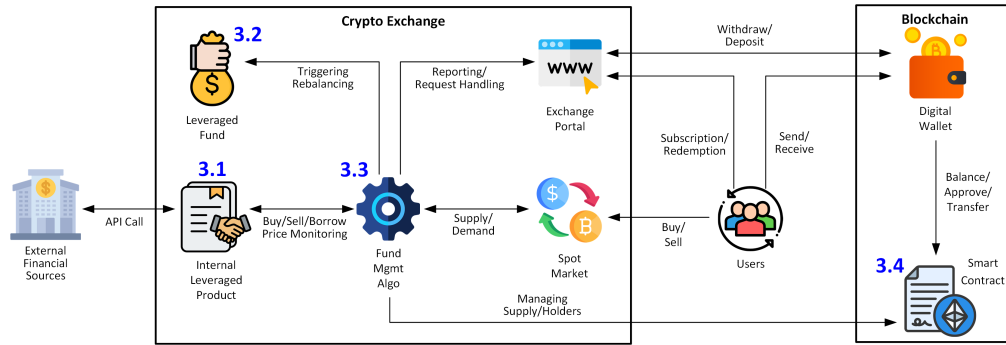
$$G_{t_0 \rightarrow n} = (1 + kR_{t_0 \rightarrow 1})(1 + kR_{t_1 \rightarrow 2}) \cdots (1 + kR_{t_{(n-1)} \rightarrow n}) = \prod_{n=1}^N (1 + kR_{t_{(n-1)} \rightarrow n}) \quad (6)$$

The longer-term return of LVTs is impacted by the carried-over  $k$  multiplier on each trading day, which magnifies both profit and loss due to the compounding effect. Even though the intention behind long and short LVTs is to move in opposing directions on a daily basis, it is common for both types to generate negative cumulative returns when held over an extended period (see example 30 in Appendix A.5 of the full version [40] on the compounding effect).

**Takeaways:** Daily returns are embedded in the design of LVTs. Generally, a  $k$ -leveraged LVT should earn  $k$  times the daily return of the underlying asset. The amplification ratio, known as leverage ( $k$ ), can be fixed or dynamic. Leverage divides LVTs into three functional groups: Long, Short, and Low-risk. A proportional change in the underlying price results in a  $k$ -times proportional change in the LVT price. In the short term, LVT returns align with  $k$ , but beyond a single day, the return becomes highly path-dependent, making LVTs unsuitable for buy-and-hold strategies. This is an issue ignored by most retail investors, leading to unexpected capital losses.

## 3 Leveraged Token Mechanics

LVTs are tokenized representations of a leveraged fund whose value is derived from the value of a leveraged product. Leveraged products are essential components of LVTs, allowing issuers to form a leveraged fund and offer it as centralized or decentralized tokens. 99.9% of LVT issuers use crypto futures as the leveraged product. To properly reflect the value of the fund through circulating LVTs, the notional value of all tokens must match the fund's notional value. As the price of the leveraged product changes over the trading day, the leverage of the fund gradually diverges from the promised ratio. The *fund management algorithm* resets this deviation by buying or selling the leveraged product on a daily basis. It also implements the logic of the LVT and defines how it should function in different market conditions. If an LVT is designed to be fully on-chain and decentralized, the smart contract provides the functionality of the *fund management algorithm* as well, by extending common features of the ERC-20 standard. The constituent components of LVTs vary depending on the issuer. According to issuer documentation, the general components of a typical LVT are illustrated in Figure 1, followed by a brief explanation of the functionality of each component.



■ **Figure 1** The constituent components of LVTs, according to the issuer’s documentation, have been implemented internally by some issuers, resulting in missing blockchain components.

### 3.1 Leveraged product

LETFs use leverage to open positions worth more than the required capital. In LVTs, leveraged cannot appear out of nowhere. In the absence of Total Return Swaps (TRS) in the crypto market (more on TRS and financial leverage in Appendix A.1 of the full version [40]), other mechanisms for achieving leverage are (i) opening positions in the crypto derivatives market, which provides up to 200x leverage, and (ii) borrowing capital from external sources, generating up to 10x leverage. LVT issuers typically do not use high factors, offering tokens with up to 5x leverage. This allows to choose either derivatives or debt as the leveraged product. 99.9% of issued LVTs use futures (a type of derivative), and only *Index Coop* uses the debt market to finance investments.<sup>17</sup> The desired outcome for them is to generate future returns that outweigh the cost of borrowing. Other issuers that use derivatives aim to minimize dependency on other exchanges for buying and selling futures. They often offer the corresponding futures trading in their own portfolio to facilitate LVT management and reduce the cost of transactions between exchanges (compare the *Leveraged Product* and *Fund Source* columns in Table 1). For example, every issuer that launched BTC Long/Short tokens offers BTC-Perp futures as the underlying. Internal leveraged products facilitate LVT operations, such as adjusting fund positions, monitoring underlying price fluctuations, and triggering fund rebalancing.

### 3.2 Leveraged fund

It is a fund that derives its *notional value* from a basket of leveraged products.<sup>18</sup> The leveraged products provide leveraged exposure, upon which the value of the issued tokens is based. Let  $V_{t_n}$  represent the price of the  $k$ -times leveraged product  $V$  on trading day  $n$ , tracking the underlying asset  $S$ . The price of  $V_{t_n}$  differs from  $S$  as it carries  $k$ -times exposure. A leveraged fund  $L$  with a notional value of  $L_{t_n}$  can be formed by purchasing a basket of  $V$ , given by:

$$L_{t_n} = kV_{t_n} B_{t_n} (1 + (\rho_{t_n} + \phi_{t_n})) \quad t \geq 0, 0\% \leq \rho_{t_n} + \phi_{t_n} \leq 0.5\%, \rho_{t_0} = 0, \phi_{t_0} = 0 \quad (7)$$

<sup>17</sup> *Index Coop* uses money market protocols on Ethereum (e.g., Compound protocol) that offer permissionless borrowing and lending capabilities [13].

<sup>18</sup> The notional value represents the total value of a financial instrument or contract at its full face value (i.e., controlled money by the financial instrument). The notional value is not typically exchanged between counterparties; instead, it serves as a reference point for calculating payments or obligations [21].

Where  $B_{t_n}$  is the number of  $V$  units forming the fund at the rebalancing time  $t_n$ .  $\rho_{t_n}$  and  $\phi_{t_n}$  represent management and futures funding fees, respectively.  $\rho_{t_n}$  is always negative, as the issuer deducts associated expenses from the fund's value.  $\phi_{t_n}$  can be positive or negative, depending on the *futures funding fee* payments (more in Section 4.4.3). The sum of  $\rho_{t_n}$  and  $\phi_{t_n}$  (i.e., the total daily fee) varies per LVT issuer and ranges from 0.01 to 0.5 percent daily. Note that the change in the price of the leveraged product ( $V_{t_n}$ ) is proportional to the underlying price ( $S_{t_n}$ ), but does not vary based on the  $k$  multiplier. More precisely,  $V_{t_n} = V_{t_{n-1}}(1 + R_{t_{(n-1) \rightarrow n}})$ . In equation (7),  $L_{t_n}$  represents the financial value controlled by the leveraged fund  $L$ , which originates from the leveraged product  $V$ . The change in the price of  $V$  is proportional to  $S$ , but the value of  $L$  changes with respect to the  $k$  factor. In simple terms,  $V$  represents the price of the leveraged product, while  $L$  represents the amount of money that can be controlled using  $V$ .

► **Example 4.** An issuer may arrange an *Ether long double-leveraged fund* by purchasing 4 *Ether-Perp long 2x futures* at \$1.5K ( $V_{t_0}$ ). The 2x leverage of Ether-Perp allows the issuer to pay half of the Ether price, which is assumed to be \$3K ( $S_{t_0}$ ). With zero fees in (7) at  $t_0$  ( $\rho_{t_0} + \phi_{t_0} = 0$ ), a leveraged fund  $L$  worth  $2 \times \$1.5K \times 4 = \$12K$  can initially be formed ( $L_{t_0}$ ). A 10% change in the price of Ether affects the price of futures by the same 10%, bringing it to \$1.65K ( $V_{t_n}$ ). However, the notional value of the fund ( $L_{t_n}$ ) changes according to  $k = 2$ , reaching  $2 \times \$1.65K \times 4 = \$13.2K$ . This demonstrates the effect of  $k$  on  $L$  compared to  $V$ .

LVTs are issued with a certain initial supply that can be adjusted through the *Subscription* and *Redemption* process (more on this process in Appendix A.5 of the full version [40]). For added or removed tokens, the issuer offsets the notional value of the fund with the notional value of the tokens by buying or selling the corresponding amount of the leveraged product. Let  $N_{t_n}$  represent the total supply of a  $k$ -leveraged LVT at time  $t_n$ . The notional value of the issued LVTs ( $A_{t_n}$ ) can be expressed as:

$$A_{t_n} = kN_{t_n}P_{t_n} \quad (8)$$

Equating (7) and (8) gives the total number of tokens ( $N_{t_n}$ ) that should exist at the price of  $P_{t_n}$  at time  $t_n$ . Mathematically, this can be expressed as:  $kN_{t_n}P_{t_n} = kV_{t_n}B_{t_n} \Rightarrow N_{t_n} = \frac{V_{t_n}B_{t_n}}{P_{t_n}}$ ,  $t \geq 0, P_{t_0} \geq 1$ .

► **Example 5.** Assume  $P_{t_0} = \$10$  as the initial offering price of ETH2L in the previous example 4.  $P_{t_0}$  is typically set by the issuer at either \$1 or \$10 per token.<sup>19</sup> The initial supply of ETH2L at \$10 per token is:

$$N_{t_0} = \frac{V_{t_n}B_{t_n}}{P_{t_n}} = \frac{\$1.5K \times 4}{\$10} = 600$$

The notional value of all 600 tokens ( $A_{t_0} = 2 \times 600 \times \$10 = \$12K$ ) is consistent with the notional value of the leveraged fund ( $L_{t_0} = 2 \times \$1.5K \times 4 = \$12K$ ). Investors purchase a portion of this fund in the form of LVT, allowing them to generate twice the profit compared to the underlying Ether-Perp. Essentially, the value of the ETH2L token is derived from the *Ether long 2x leveraged fund*, which in turn is derived from the 4 positions in the *Ether-Perp long 2x futures*.

<sup>19</sup> Taken from the initial public offering (IPO) price of a SPAC (Special Purpose Acquisition Company), which is typically set at a nominal \$10 per unit. Unlike a traditional IPO, the SPAC IPO price is not based on the valuation of an existing business but rather on future income expectations.

### 3.3 Fund management algorithm

As the price of the leveraged product ( $V_{t_n}$ ) fluctuates over time, the notional value of the leveraged fund ( $L_{t_n}$ ) changes, causing the leverage ratio of the LVT to deviate from the stated leverage. Let  $k_{t_n}^{\sim}$  represent the realized leverage ratio that the notional value of the tokens ( $A_{t_n}$ ) represents at time  $t_n$ , expressed as  $k_{t_n}^{\sim} = \frac{L_{t_n}}{N_{t_n} P_{t_n}}$ .

► **Example 6.** Referring to the first trading day of ETH2L in examples 4 and 5, in which the price of Ether ( $S_{t_0}$ ) increases by 10%, the notional value of the fund ( $L_{t_n}$ ) changes to  $2 \times \$1.65K \times 4 = \$13.2K$ . The 2x leverage of the LVT increases its price by 20%, rising to \$12 ( $P_{t_n}$ ) from the initial \$10 ( $P_{t_0}$ ). Since the LVT supply remains constant at 600 tokens, the leverage ratio of the fund drops from 2x to 1.8x. ( $k_{t_n}^{\sim} = \frac{L_{t_n}}{N_{t_n} P_{t_n}} = \frac{\$13.2K}{600 \times \$12} = 1.8\bar{3}$ ).

The analysis above suggests that with the change in the price of the underlying ( $S_{t_n}$ ) and subsequently the price of the leveraged product ( $V_{t_n}$ ), the notional value of the leveraged fund ( $L_{t_n}$ ) changes, and the realized leverage ratio of LVTs ( $k_{t_n}^{\sim}$ ) becomes higher or lower than the stated leverage  $k$ . Mathematically, if  $\mathbb{E}[k_{t_n}^{\sim}]$  represents the expected change in  $k_{t_n}^{\sim}$  in relation to the underlying price change ( $S_{t_n}$ ), applying equations (1) gives us:

$$\mathbb{E}[k_{t_n}^{\sim}] = \frac{kV_{t_n} B_{t_n} (1 + R_{t_{(n-1)} \rightarrow t_n})}{kN_{t_n} P_{t_n}} = \frac{V_{t_n} B_{t_n} (1 + R_{t_{(n-1)} \rightarrow t_n})}{N_{t_n} P_{t_{n-1}} (1 + kR_{t_{(n-1)} \rightarrow t_n})} \quad (9)$$

As the number of tokens ( $N_{t_n}$ ) remains constant while the price of the underlying changes at a rate of  $(1 + R_{t_{(n-1)} \rightarrow t_n})$ , the denominator of (9) changes  $k$ -times faster (or slower) than the numerator, resulting in positive or negative leverage skewness. This highlights the need to re-leverage the fund on a daily basis, a process managed by the *fund management*.<sup>20</sup>

Fund management is an off-chain algorithm (or on-chain for decentralized tokens) that dynamically adjusts the fund to maintain the leverage at the expected ratio. When the token's leverage increases, it sells some of the fund's positions to reduce the leverage and return it to the expected level (*cf.* Full version [40] appendix A.7 for rebalancing details). The majority of algorithms are off-chain with no interaction with the blockchain. The only on-chain instance is implemented by *Index Coop* [13]. In addition to correcting the leverage, the algorithm interacts with other components to adjust supply, update balances, monitor the price of the underlying, and deduct daily fees.

### 3.4 On-chain contracts

For decentralized LVTs, a smart contract represents the leveraged fund on the blockchain. It is typically implemented as an ERC-20 token [20, 2, 43], allowing users to exchange LVTs on the blockchain without issuer intervention. As indicated in the *Blockchain Representation* column of Table 1, 80% of issuers have not created LVTs on the blockchain. As a result, this component is missing from Figure 1. The absence of a smart contract leads to several deficiencies, which are discussed in the next section.

**Takeaways:** An LVT is a tokenized representation of a leveraged fund, whose value is derived from the value of a leveraged product. 99.9% of LVT issuers use crypto futures as the leveraged product. As the price of the leveraged product fluctuates over a trading day, the leverage of the fund gradually diverges from the promised ratio. The *fund management algorithm* resets this deviation by buying or selling the leveraged product on a daily basis.

<sup>20</sup> Also referred to as *fund management agent*, *fund management party*, or *certified fund manager*.



## 4 Research Questions

Due to the lack of a common standard in LVTs for defining the rebalancing process, data transparency, implementation standards, etc., these tokens are issued with varying features at the discretion of the issuer. We examine the characteristics of issued tokens per issuer and discuss the respective deficiencies as research questions RQ1 to RQ6.

### 4.1 RQ1: What information is visible to traders of an LVT?

Among the 10 LVT issuers, only *FTX* and *Index Coop* have created tokens on the Ethereum blockchain. *FTX*'s management model was hybrid (*i.e.* tradable decentralized tokens on a centralized exchange), while only *Index Coop*'s tokens are fully decentralized. The remaining 8 exchanges prefer to implement LVTs centrally and entirely internal.

► **Example 7.** Binance leveraged tokens (BLVTs) are one of the centralized LVTs that are entirely accessible within Binance's ecosystem. They can be exclusively traded on Binance's spot market with no possibility of withdrawal. BLVTs are not even published on Binance's own blockchain (BNB Smart Chain) and are created more like a pseudo-crypto.<sup>21</sup>

#### 4.1.1 Transparency in total supply

Total supply is used to calculate the Net Asset Value (NAV) of LVTs as a representation of the market's fair value. Due to imbalances in supply and demand, the market price of LVTs may deviate from the NAV, trading at a premium or discount. In the long run, LVT prices converge to the NAV due to a mechanism similar to arbitrage in traditional markets. Orders placed far from the NAV price lose or gain value over time (*cf.* Full version [40] appendix A.13 on the general arbitrage mechanism). In the short run, however, investors use the NAV as a reference price when buying or selling, especially in bulk. The NAV of LVTs can be calculated by equating (7) and (8), with the current token supply  $N_{t_n}$ :

$$kN_{t_n}P_{t_n} = kV_{t_n}B_{t_n} \Rightarrow P_{t_n} = \frac{V_{t_n}B_{t_n}}{N_{t_n}} \quad t \geq 0, N_{t_0} \geq 1 \quad (10)$$

► **Example 8.** In the previous examples (4) to (6), when the Ether price increases by 10% on day 2, the market price of ETH2L trades at \$12 (after a  $2 \times 10\% = 20\%$  increase), while its NAV price is  $(\$1.65K \times 4)/600 = \$11$ . ETH2L is, in fact, overvalued, and traders should wait for either (i) the arbitrage mechanism to play out and bring the LVT price down, or (ii) the next rebalancing schedule, which will match the fund's value with the notional value of the tokens.

For LVTs hosted on the blockchain, total supply is public and can be retrieved for NAV calculations. However, for centralized LVTs, investors must refer to the exchange's website. The total supply of tokens on some exchanges, such as AscendEX, Pionex, Gate.io, and ByDFi, does not appear to be public, making it difficult to verify the real value of LVTs (*i.e.* 53% of all tokens). LVTs are *open-end funds* with a theoretically unlimited token supply.<sup>22</sup> Issuers can increase the supply based on market liquidity and demand for the

<sup>21</sup> A cryptocurrency that is not sufficiently decentralized [1].

<sup>22</sup> Open-end funds can issue an unlimited number of shares. The fund sponsor sells shares directly to investors and redeems them as well. The NAV per share of an open-end fund is calculated daily by dividing the total value of the fund (minus liabilities) by the total number of shares outstanding [10].

## 23:12 A Shortfall in Investor Expectations of Leveraged Tokens

token. Transparency in the number of issued tokens builds trust and reduces the risk of investment. Moreover, it addresses audit questions such as, Has the fund's value changed proportionately after increasing or decreasing the supply of LVTs? How much were the fund's value deviations in the previous audit period, and were they within the acceptable range?

### 4.1.2 Transparency in transactions

Transactions on the blockchain show the flow of tokens and the movement of the fund. This enables investors to analyze transactions and ensure the expected functionality of LVTs.

► **Example 9.** We reviewed all *Mint* and *Burn* transactions of ETCBULL (FTX 3x Long Ethereum Classic) on the blockchain.<sup>23</sup> The analysis suggests that a total of 51,640,895 tokens were issued, and 24,207 were destroyed (*i.e.* 51,616,688 circulating tokens). The trend of issuing tokens has taken on exponential velocity since April 2022. A total of 783,022 tokens were issued during the 960-day period between October 2019 and May 2022, while 50,857,873 tokens were issued over just 184 days from April to October 2022. In other words, 98.5% of all tokens were issued in just 6 months. Checking the recipient address indicates FTX's possible sub-wallet as the receiver. This sudden change in token supply warrants further investigation, especially given FTX's collapse shortly afterward. A possible explanation for this anomaly is presented in Appendix A.11 of the full version [40].

This is just an example indicating the importance of transparency in LVT transactions. Transactions of centralized tokens are not public and only available to the issuer. Statistically, transactions of 80% of LVTs cannot be analyzed as we did in the above case.

### 4.1.3 Transparency in token holders

Holders of tokens created on the blockchain are public, allowing investors to check them as a measure of the token's liquidity. A small number of market participants reduces the token's liquidity and can make it more challenging to execute large orders. It may also lead to a wider bid-ask spread, increasing the cost of executing trades. Investors generally prefer assets with higher liquidity, narrower bid-ask spreads, and more market participants.

► **Example 10.** 90% of XRPBULL (FTX 3x Long Ripple) tokens are distributed among four holders.<sup>24</sup> In another example, three accounts own 94% of all issued FTX 3x Long Cardano (ADABULL) tokens.<sup>25</sup>

Holding a large number of tokens by a limited number of accounts can noticeably elevate investment risk. One holder may decide to sell a significant number of tokens at any moment, potentially resulting in a notable price drop within a short period of time, leading to significant losses for smaller holders. Since most issuers do not publish the list and respective ownership percentages of centralized LVTs, the participants of 80% of LVTs remain uncertain.

### 4.1.4 Inability to audit

Conducting audits ensures the security, functionality, and compliance of LVTs as claimed by the issuer. Unlike centralized LVTs, the code of tokens created on the blockchain is public,

---

<sup>23</sup> ETCBULL transactions on the Ethereum blockchain filtered for issued and deposited tokens to FTX's own address: <https://bit.ly/3MwHVqv>.

<sup>24</sup> List of XRPBULL holders: <https://bit.ly/3MQurrc>.

<sup>25</sup> List of ADABULL holders: <https://bit.ly/3MUxPRZ>.

allowing auditors to identify vulnerabilities and associated risks. The security of these 20% decentralized LVTs can be evaluated by reviewing the code against industry best practices such as SWC [45]. Moreover, external audits are essential for LVTs to ensure they function as intended, such as verifying the output of methods when transferring tokens or updating balances. Auditors may also provide recommendations to improve the security, functionality, and compliance of LVTs. For centralized LVTs, the code is not public, requiring cooperation and willingness of the issuer to conduct a thorough review and quality assessment. Sharing the code and the results of an independent audit would improve transparency and help build trust between token holders and issuers.

**Takeaways:** In some centralized LVTs, investors do not have access to crucial information such as total supply, transactions, and holders. Total supply is a necessary parameter for calculating the fair value of tokens and is also used by auditors to evaluate the consistency and efficiency of LVTs. Transparency in transactions enables investors, auditors, and anyone involved in the LVT ecosystem to analyze token flow, detect suspicious activities, enhance security, ensure compliance, and verify token functionality. The number of holders for centrally issued tokens is unknown, leading to a much higher investment risk compared to decentralized counterparts.

## 4.2 RQ2: To what extent are LVTs locked to the offering exchange?

### 4.2.1 Interoperability with dapps and DeFi

In 2019, the total value locked in Decentralized Finance (DeFi) was approximately 700 million USD. As of April 2022, it stands at around 150 billion USD, representing more than 200% growth in less than three years [51]. Hosted LVTs on the blockchain (which usually comply with one of the fungible<sup>26</sup> token standards) facilitate interaction with DeFi systems, unlocking potential interoperability opportunities.

► **Example 11.** FTX was able to employ blockchain interoperability to share its ETHBULL (3x Long Ethereum) with other exchanges such as Poloniex, Indodax, Bittrex, and Gate.io.<sup>27</sup> These exchanges owned 20%, 4%, 3%, and 2% of ETHBULL, respectively, and offered it on their platforms due to the possibility of interaction with DeFi.

In contrast, centrally issued LVTs cannot interact with other platforms and operate in isolation, preventing LVTs from moving across different platforms and systems. Decentralized LVTs, on the other hand, foster connectivity and enable users to access a wide range of services and functionalities without being confined to a single exchange.

### 4.2.2 Inability to custody

At first glance, the custody issue seems common to all assets on centralized exchanges. However, BTC buyers can transfer it to their personal wallets, while centralized LVTs remain locked within the exchange. Holders do not own the actual tokens but are simply betting on price movements. Some explain this custodial issue by viewing LVTs as “token contracts”, though this term is not widely recognized nor aligns with the functionality of crypto derivatives. LVTs are essentially tokenized forms of derivative exposures.

<sup>26</sup> Fungible (interchangeable) token standards are widely used by decentralized applications (dApps) to interact with other applications. ERC-20 is the dominant standard, followed by ERC-777 and ERC-1155.

<sup>27</sup> List of ETHBULL holders: <https://bit.ly/3MSZX7P>.

► **Example 12.** BTCUP and BTCDOWN are issued by Binance and track Bitcoin as the underlying asset. Unlike Bitcoin holders, owners of these tokens cannot withdraw or transfer them to their own digital wallets. In contrast, similar Bitcoin leveraged tokens were created by FTX on the Ethereum blockchain (known as BULL and BEAR tokens). Holders of these tokens still had the opportunity to exchange them on decentralized exchanges, such as Uniswap<sup>28</sup>, shortly after FTX’s bankruptcy. Holders could recover 80% of the token value on the first day of the bankruptcy, 50% on the second day, and up to 20% on the third day.

**Takeaways:** The inability to self-custody centralized LVTs raises greater concerns compared to decentralized counterparts, potentially increasing the investment risk in these crypto-assets. Another important advantage of tokens created on the blockchain is their interoperability with other dApps, crypto exchanges, and the DeFi ecosystem in general. LVTs that interact with DeFi offer the opportunity to participate in a more open and transparent financial system that operates without the need for intermediaries.

### 4.3 RQ3: Are the LVTs offered today adequately backed?

The simplest definition of an LVT is a tokenized leveraged fund. According to the documentation of LVT issuers, 99.9% of leveraged funds derive their value from a basket of positions in the futures market [3, 22, 5, 6, 7, 39, 16, 29, 38]. The issuer must either (i) offer futures trading in their portfolio, or (ii) open futures positions on other crypto exchanges and manage them systematically through APIs as the underlying asset fluctuates.<sup>29</sup> The question we raise, due to the lack of external audits, is to what extent LVT issuers have properly prepared futures contracts before launching LVTs. Have users invested in tokens that are properly backed, or are they simply trusting the issuer and potentially investing in tokens with no real value?

#### 4.3.1 Missing futures product

Some issuers have launched LVTs without offering the corresponding futures products. While we cannot rule out the possibility that they hold the necessary futures positions on other exchanges, this raises concerns that these LVTs might not be adequately backed.

► **Example 13.** AscendEX uses its own futures products and does not rely on futures products from other exchanges. However, they issued 3x/5x Long/Short Monero (XMR3L/S and XMR5L/S) without offering XMR perpetual contracts initially. To our knowledge, no XMR futures products were available on the market from any exchange to be used as leveraged products at the time of the launch of these XMR tokens.

The above example is one of 390 issued tokens lacking a corresponding futures product (see column B of Table 2). Based on available historical data and information from the issuer’s website, 24% of LVTs did not have the necessary futures product offered by the same issuer and instead relied on futures from other exchanges.

<sup>28</sup> DEX transactions of FTX 3X Long Bitcoin Token (BULL) and FTX 3X Short Bitcoin Token (BEAR) on the Ethereum blockchain: <https://bit.ly/3p0h3uz>, <https://bit.ly/41Da181>.

<sup>29</sup> Among 10 LVT issuers, Pionex used the *Binance Broker API* for its *Futures Arbitrage Bot*, but it has been terminated since June 2021 [37]. After reviewing Pionex’s documentation [38], it remains unclear whether *Binance Futures* is still used as the leveraged product for LVTs. However, Pionex launched its own futures product in January 2023. If they no longer use *Binance Futures* and rely solely on their own futures product, it appears that 148 LVTs did not have corresponding futures contracts at the time of launch (e.g., ETC3L/S, ZRX2L/S, XLM3L/S).

■ **Table 2** *Left table*: Number of issued LVTs with delayed or missing futures products, analyzed using historical data and undisclosed information. To our knowledge, 41% of the issued LVTs did not have sufficient financial backing at the time of launch. *Right table*: Rebalancing and fee deduction schedules, which we collected manually from the issuers' websites. Regardless of leverage type, rebalancing is performed daily at different times. Additionally, Threshold-Based (TBR) or Out of Range (OOR) rebalancing methods are used to trigger interim rebalancing. Fund expenses are also deducted at various times with variable percentages.

Issuer	Delayed Futures Launch	Missing Futures Product	Total Delayed or Missing	Total Launched LVTs	% of unbacked LVTs	Leverage	Rebalancing Schedule		Fee deduction	
	(A)	(B)	(C)	(D)	(E)		Regular Daily	Interim	Daily Schedule	Expense Ration
	(a)	(b)	(c)	(d)	(e)					
AscendEX	36	214	250	340	74%	Fixed	02:30 UTC	10% TBR / OOR	00:00 UTC	0.500%
Pionex	0	148	148	200	74%	Fixed / Variable	00:00 UTC+8	10% TBR		0.030%
MEXC	176	12	188	456	71%	Fixed	00:00 UTC	15% TBR	00:00 UTC+8	0.100%
ByDFi	16	0	16	40	40%		08:00 UTC+8			0.030%
FTX	18	0	18	129	14%		02:00 UTC	10% TBR	00:00 UTC	0.030%
Gate.io	18	16	34	274	12%		00:00 UTC-4		00:00 UTC+8	0.300%
Binance	0	0	0	40	0%	Variable	N/A	10% TBR / OOR		
ByBit	0	0	0	34	0%		00:00 UTC		00:00 UTC	0.005%
KuCoin	0	0	0	94	0%	Fixed / Variable	08:00 UTC+8	14% TBR	23:45 UTC+8	0.045%
Index Coop	0	0	0	2	0%	Fixed	00:00 UTC	20% TBR	00:00 UTC	0.023%
<b>Total</b>	<b>264</b>	<b>390</b>	<b>654</b>	<b>1609</b>	<b>41%</b>					

### 4.3.2 Delayed futures product

Missing futures products are not the only issue with centralized LVTs. For 264 tokens, the corresponding futures product was only offered after the issuance of the tokens (see column A of Table 2). In other words, at the time of the launch of 17% of LVTs, the required futures may not have existed. According to the LVT documentation on issuers' websites, these issuers did not disclose using futures from other crypto exchanges. Internal futures trading was introduced later, after the token was launched.

► **Example 14.** MEXC issued 3x Long/Short Cardano (ADA3L/S) in February 2020, while ADA-Perp was only launched in July 2020, resulting in a 154-day delay. They did not disclose using futures from other exchanges, indicating the fund might have been operating without financial backing during this period.

According to available information, on average, 41% of LVTs have missing or delayed futures products (see column E of Table 2). The main financial issue with LVTs is the lack of transparency in the fund management system. Centralized LVTs function like a black box to investors and are fully managed by the issuer. Even for tokens with proper futures backing (e.g., Binance, ByBit, and KuCoin), investors can rely solely on numeric assertions made on the issuer's website.

**Takeaways:** The value of LVTs is derived from a leveraged fund, which itself is based on the value of futures. In the absence of futures at the time of offering, investors may have purchased LVTs that lacked proper financial backing. An analysis of available historical data on the issuer's website shows that, on average, 2 out of every 5 issued LVTs did not appear to have proper financial backing at the time of launch. Although the exchange may have addressed this issue over time, the required futures contracts were missing at the time of issuance. Without external audits, investors can rely solely on the exchange's claims and assume that LVTs are financially backed as promised.

## 4.4 RQ4: What are the possibilities of front-running in LVTs?

Front-running is an illegal practice in the equity market where non-public information is used to purchase shares of a company before the price moves [19, 53, 4]. For instance,

FINRA<sup>30</sup> announced a \$700K fine against *Citadel Securities*<sup>31</sup> in 2020 for front-running activities between 2012 and 2014 [35]. In the design of LVTs, certain well-known events can be exploited by traders to benefit from anticipated price movements. They can engage in similar front-running practices that may impact the price of the underlying asset and the token itself. We review these events and explore possible front-running scenarios as follows.

#### 4.4.1 Event I: Impending fund rebalancing

Rebalancing is the process of maintaining the desired leverage ratio of funds over time. To keep the leverage at the stated ratio, issuers perform periodic rebalancing. This can be triggered at predefined intervals (e.g., every day, every 8 hours, or every  $n$  blocks), or upon meeting certain conditions (e.g., after exceeding a specific threshold). All LVT issuers perform regular daily rebalancing and trigger interim rebalancing in volatile markets (see columns B and C of Table 2). They may trigger rebalancing when the underlying asset's price fluctuates by more than  $X\%$ , or when the leverage passes a threshold. The *fund management algorithm* governs the rebalancing process, adjusting futures positions and restoring the leverage ratio to the target level (*cf.* Full version [40] appendix A.7 for details on the rebalancing process).

The number of contracts that must be bought or sold to restore the leverage is predictable, making front-running possible. Let  $\Delta B_{t_n}$  represent the number of required futures contracts to rebalance the fund at time  $t_n$ .  $\Delta B_{t_n}$  can be easily calculated by considering the return of the underlying asset from time  $t_{n-1}$  to  $t_n$  ( $R_{t_{(n-1) \rightarrow n}}$ ). The number of required futures contracts to restore the fund leverage can be calculated by subtracting the notional value of the tokens (equation 8) from the notional value of the fund (equation 7):

$$\Delta B_{t_n} = (1 + kR_{t_{(n-1) \rightarrow n}})kN_{t_{n-1}}P_{t_{n-1}} - (1 + R_{t_{(n-1) \rightarrow n}})kV_{t_{n-1}}B_{t_{n-1}} - (\rho_{t_n} + \phi_{t_n})L_{t_n} + \epsilon_{t_n}$$

This equation is quadratic and can be simplified as  $ax^2 - bx - c$  for long tokens, and  $-ax^2 + bx - c$  for short tokens. When the underlying return is positive ( $R_{t_{(n-1) \rightarrow n}} > 0$ ),  $\Delta B_{t_n}$  is always positive ( $a > 0$ ), and when the return of the underlying is negative,  $\Delta B_{t_n}$  is negative as well ( $a < 0$ ). In simpler terms, for long LVTs, futures exposure must be increased when the underlying price is rising, and decreased when the underlying price is falling (see *Fund Basket Delta* in Tables 12 and 13 of appendix A.7 in the full version [40]).

There is also a fund expenses term  $(\rho_{t_n} + \phi_{t_n})L_{t_n}$ , which is usually deducted from the fund's value to cover operating expenses. However, this term can turn positive when the received funding fees ( $\phi_{t_n}$ ) exceed the fund expenses ( $\rho_{t_n}$ ).  $\epsilon_{t_n}$  represents a disturbance term that captures the effects of news or shocks in the underlying. Since rebalancing is a predictable event, by buying or selling  $\Delta B_{t_n}$  of the leveraged product, other traders can front-run the trade, potentially impacting the price of the token or even the underlying asset.

► **Example 15.** Consider the following sequence in which Alice calculates  $\Delta B_{t_n}$  to potentially front-run the rebalancing trade:

1. Alice checks the issuer's website for the upcoming rebalancing of BTC5L (5x Long Bitcoin). She notices the next daily rebalancing is scheduled for 00:00 UTC.
2. Alice calculates the number of contracts that will be bought or sold by the issuer to maintain the 5x target leverage of BTC5L.

<sup>30</sup>The Financial Industry Regulatory Authority (FINRA) is a government-authorized organization that oversees U.S. equity markets by regulating member brokerage firms and exchange markets.

<sup>31</sup>Citadel Securities is the largest designated market maker on the New York Stock Exchange (NYSE).

3. Alice front-runs the rebalancing trade by placing an order just before 00:00 UTC (ahead of the rebalancing trade). If she anticipates that the algorithm will buy Bitcoin futures, she may buy Bitcoin futures expecting increased demand, driving up the price, and giving her the opportunity to sell futures at higher prices. Conversely, if BTC5L will be selling the fund's positions, leading to increased supply, she may sell Bitcoin futures.

Front-running in the above example may not only manipulate the price of Bitcoin futures but also inflate the price of BTC5L. Consider the following scenarios (A) and (B), with corresponding calculations in Table 8 of appendix A.5 in the full version [40].

- Alice calculates the *Basket Delta* of BTC5L prior to the rebalancing schedule and realizes that the algorithm will purchase 594.21 new contracts at \$33,000 (Scenario A in Table 8 of appendix A.5 in the full version [40]). Assuming this purchase increases the price of Bitcoin futures by 1%, she can buy contracts just before the rebalancing trade at \$33,000 and sell them afterward at \$33,330. A 1% increase in the underlying price inflates the token price to \$15.05. This provides Alice an additional opportunity to buy the token for \$15 before the rebalancing and sell it at a higher price afterward.
- The effect of Alice's strategy in the previous scenario may be amplified if many traders engage in front-running. The increased demand may raise the price of Bitcoin futures even before the rebalancing trade. If the influx of other traders pushes Bitcoin futures up by 1%, and the rebalancing trade further increases the price by another 1%, this secondary effect could also inflate the token price further and create more price distortion (Scenario B in Table 8 of appendix A.5 in the full version [40]).

#### 4.4.2 Event II: Management fee deduction

Similar to LETFs, daily fees and expenses are deducted from the leveraged fund to cover associated costs (*cf.* Full version [40] appendix A.8 on incurred costs). The fee rate and daily schedule vary by issuer (see columns D and E of Table 2). Management fees are deducted at specific times, allowing adversaries to exploit this known event, potentially coinciding with rebalancing. The simultaneous occurrence of events I and II can intensify the front-running effect during the rebalancing process.

► **Example 16.** Consider the same sequence as the previous example, where Alice calculates the *Basket Delta* of BTC5L at the same time as the management fee deduction. The issuer's withdrawal of \$99,000 (Management fee row in Table 8 of appendix A.5 in the full version [40]) reduces the fund's value. To compensate,  $\$99,000/\$33,000 = 3$  additional contracts need to be purchased. The coincidence of these two events causes the rebalancing algorithm to slightly increase demand by purchasing 597.21 contracts instead of 594.21.

#### 4.4.3 Event III: Futures funding fee exchanges

Funding fee is a mechanism in Perps to converge the price of contracts with the price of the underlying crypto. It is calculated based on the notional value of the futures position and is exchanged between short and long traders who keep their positions open. Shorts pay longs when the funding rate is negative, and longs pay shorts when the rate is positive (*cf.* Full version [40] appendix A.9 for details on funding fee dynamics). The intervals for *Funding fee exchange* are public and displayed on the issuer's website, typically occurring every 8 hours at 00:00 UTC, 08:00 UTC, and 16:00 UTC. Since the fund is composed of futures, it either pays or receives funding fees at these times. This predictably increases or decreases the value of the leveraged fund, which can be exploited to amplify the effects of front-running.

## 23:18 A Shortfall in Investor Expectations of Leveraged Tokens

The impact of front-running can be exacerbated when events I, II, and III occur simultaneously. Such concurrency may force the algorithm to buy or sell more contracts than would be required for fund rebalancing alone (*i.e.*  $\Delta \tilde{B}_{t_n} = \sum_{n=1}^3 \Delta B_{t_n}$ ).

► **Example 17.** As calculated in the *Basket Delta* row of Table 8 in appendix A.5 of the full version [40], in the first rebalancing cycle, an additional 2.70 futures contracts are required to cover the 0.3% daily management fee deduction and 0.03% daily funding fee exchange. This means that 2.7 more contracts will be added if events II and III coincide with event I. The impact on basket delta can be even more significant as the value of the fund increases in a volatile market (see *Fee Basket Size* row in Table 12 of appendix A.5 in the full version [40] on rebalancing in volatile markets).

To mitigate front-running in LVTs, issuers should avoid rebalancing the fund on predetermined schedules. Techniques such as intraday or randomized rebalancing, or algorithmic trading, can help reduce the visibility of rebalancing trades.<sup>32</sup> Only Binance avoids regular daily rebalancing, instead triggering it when the underlying price fluctuates more than 10% or when leverage falls outside the range of  $[-4.0, -1.25] \cup [1.25, 4.0]$ . This means 97% of current LVTs perform fund rebalancing at specific intervals, increasing the likelihood of daily front-running (see column B of Table 2).

**Takeaways:** The possibility of front-running in LVTs arises from the predictability of impending fund rebalancing, management fee deductions, and futures funding fee exchanges. Adversaries can exploit the temporary changes in supply and demand initiated by the fund. This effect may be intensified if all three events occur simultaneously or if there is large-scale participation by multiple traders. Front-running has also been a concern in LETFs, though it is mitigated by the continuous oversight of regulatory bodies such as the SEC and FINRA.

### 4.5 RQ5: How well do LVTs track their asserted leverage ratios?

The leverage ratio of LVTs is determined by the issuer and can be either variable (dynamic) or fixed. If an LVT uses the *Underlying+Leverage+Long/Short* naming convention, the leverage is most likely fixed. The *Underlying+Up/Down* format is used for LVTs with variable leverage.

► **Example 18.** KuCoin has issued ETH3L as a 3x long token tracking Ether as the underlying. Binance similarly offers ETHUP and ETHDOWN tokens with a target leverage in the range of  $[1.25, 4]$  and  $[-4, -1.25]$ , respectively.

Very high leverage factors such as  $\pm 10x$  or  $\pm 15x$  are not common in currently issued LVTs, as the majority of them provide  $\pm 3x$  leverage (see Figure 3a in Appendix A.4 of the full version [40]). Low leverage is aimed at minimizing losses and extending the liquidation point during periods of high volatility. Highly leveraged LVTs lose value in the same proportion as the underlying asset and may not be attractive to investors. Crypto exchanges advertise LVTs as an investment vehicle providing leveraged exposure to crypto-assets with minimal liquidation risk. However, LVTs with high leverage factors defeat this promise.

---

<sup>32</sup> Iceberg orders, which are large orders broken into smaller lots, are a sophisticated trading algorithm used to execute rebalancing trades in smaller, more discrete chunks over time.



### 4.5.1 Inconsistency of fixed leverage

Approximately 16% of LVTs are issued with variable leverage, fluctuating in the range of  $[-4.0, -1.25] \cup [1.25, 4.0]$ . Additionally, 9%, 59%, and 11% of LVTs have fixed 2x, 3x, and 5x leverage ratios, respectively. As shown in Table 10 of Appendix A.6 in the full version [40], LVTs with fixed leverage may not always provide exactly the promoted leverage. One aspect of risk involves leverage deviation (also called tracking error). Furthermore, some issuers do not rebalance the fund in the same way.

► **Example 19.** MEXC and KuCoin adjust the leverage of only the tokens that have lost value. Consider a volatile market where Bitcoin loses 10% in a day. These issuers adjust only the leverage of BTC3S, while the leverage of BTC3L remains unchanged, as it gained value [22, 28]. For example, if the price of Bitcoin is \$30K and the fund holds 600 contracts, the fund's initial value is \$18M. Assuming 600K issued tokens at an initial offering price of \$10, the target leverage is 3x (*i.e.*  $k = (600 \times \$30K) / (600K \times \$10) = 3$ ). A 10% increase in Bitcoin's price changes the leverage of BTC3L and BTC3S to 2.53x and 4.71x, respectively. However, these exchanges correct the leverage of BTC3S to prevent further capital loss in case of more price decline. As a result, this rebalancing process undervalues the BTC3L fund (*i.e.* inflates the value of BTC3L). Instead of only rebalancing the losing side, both BTC3L and BTC3S positions should be adjusted simultaneously to bring the leverage back to 3x as advertised by the issuer.

We compared the leverage deviation of Bitcoin LVTs with LETFs over the course of a year. Analysis details are provided in Appendix A.6 and Table 10 of the full version [40]. As can be seen, LVTs exhibit higher leverage deviations than similar products in the equity market. This issue becomes more apparent when comparing the standard deviation of returns in the equity and crypto markets. Leverage deviation leads to underperformance or overperformance of tokens, causing investors to experience returns that deviate from the intended amplification effect of LVTs. This is particularly important in light of previous research on LETF returns, which shows that LETFs, on average, do not negatively impact investor short-term returns [33]. Results indicate that the daily return distribution using real-world historical data is significantly more leptokurtic than the normal distribution. However, in LVTs, returns tend to have a wider or flatter shape (platykurtic) due to higher leverage deviation.

### 4.5.2 Disadvantages of variable leverage

LVTs with variable leverage aim to: (i) minimize the impact of volatility drag (*cf.* Full version [40] Appendix A.10), and (ii) reduce the possibility of front-running (as discussed in Section 4.4). LVTs are advertised as an investment vehicle that amplifies returns relative to a certain multiplier, although this factor changes constantly in tokens with variable leverage. This introduces an additional risk dimension, requiring regular monitoring and adjustment of positions as the leverage fluctuates. Additionally, these types of tokens rebalance on an as-needed basis with no predetermined schedules. Rebalancing can therefore be triggered by (i) a sudden fluctuation in the underlying price (such as more than 15%), (ii) exceeding the expected leverage range (such as above 4x or below 1.25x for long LVTs), and (iii) handling subscription or redemption requests, which change the total supply. One disadvantage of this type of rebalancing is that funds can remain undervalued or overvalued for extended periods.

► **Example 20.** The rebalancing events of BTCUP (a Long Bitcoin LVT by Binance) over the past 3 years are listed in Table 14 of Appendix A.6 in the full version [40]. A rebalancing

## 23:20 A Shortfall in Investor Expectations of Leveraged Tokens

event occurred on 03-Jan-2023, 204 days after the previous one on 13-Jun-2022. During those 204 days, no rebalancing event was triggered because the changes in Bitcoin's price did not exceed the 10% threshold limit, and the fund's leverage fluctuated within the expected range of 1.25x to 4x. During this period, the fund's value was much lower (or higher) than the amount required to support the value of issued tokens, but no rebalancing occurred. Undervalued funds may benefit the issuer, while investors hold inflated tokens.

Another disadvantage of dynamic leverage is the imbalance in rebalancing triggers. Some issuers initiate the rebalancing process at different ranges for long and short tokens.

► **Example 21.** Pionex triggers rebalancing when the leverage of long tokens exceeds the range of [2.2, 4.0], while this range is [1.8, 4.8] for short tokens [36]. This inconsistency increases the complexity of position management, leading to unfavorable outcomes for investors.

LVTs with variable leverage may reduce the probability of front-running but also reduce token transparency and increase the complexity of managing positions. In the absence of a specific standard for the rebalancing of dynamic leverage, each issuer implements its own algorithm, which is not necessarily consistent with others. This causes confusion for investors when switching from one exchange to another, as they may expect similar performance.

**Takeaways:** As the price of the underlying asset fluctuates, the value of the fund and the presented leverage change at different rates. The price of the token might be at a premium or discount, not reflecting the actual value of the fund. All LVT issuers reconcile the total value of the tokens and the fund through a daily rebalancing schedule. However, the way this process is implemented differs by issuer, leading to deviations from the target leverage. Rebalancing in both types of tokens—those with fixed and dynamic leverage—has its shortcomings. Referring to LETFs, where almost 100% of traditional LETFs adhere to a fixed leverage strategy [47], suggests that fixed leverage may be more appropriate for LVTs as well. However, there is still a need to reduce the current high leverage deviation compared to LETFs, which can be achieved by modifying the relevant algorithms.

### 4.6 RQ6: Are LVT fees in-line with traditional LETFs?

Issuers of LETFs/LVTs charge daily fees to cover the associated costs of operating the fund (*cf.* Full version [40] Appendix A.8 on associated expenses). In recent years, fees have generally come down, with the average annual Management Expense Ratio (MER) for traditional ETFs and LETFs being 0.45% and 0.95%, respectively. We reviewed the daily fees of all LVTs and summarized the results in Table 3. Depending on the issuer, the annual MER for LVTs varies from 1.83% to 36.5%. Comparatively, the standard deviation of MER in LETFs and LVTs is 0.38% and 34.21%, respectively. This signifies that fees are less predictable and more volatile in LVTs than in LETFs (up to 90 times), impacting the overall expense ratio and net returns. Additionally, high fees often indicate a developing market with a lack of a broad base of issuers. This could be a risk indicator for investors, who typically expect lower fees due to increased competition and improved market maturity.

**Takeaways:** High daily fees in LVTs act as a continuous drag on performance, eroding returns, leading to underperformance, and making them less attractive. Daily costs in LVTs should be lower than in LETFs, given that futures transactions are internal and there is much less regulatory overhead. Furthermore, higher fees serve as a risk indicator for developing markets with limited competition and inefficient cost management.

**Table 3** Comparison of the daily expense ratio in the equity and crypto markets. Each day, the daily fee is deducted from the price of ETFs/LETfs/LVTs, which negatively impacts the ROI. Therefore, investing in assets with lower daily rates (green rows) is less risky with higher return.

Market	Symbol	Issuer	Underlying Index/Asset	Leverage	Annual Expense Ratio	Daily Expense Ratio	Market	Symbol	Issuer	Underlying Index/Asset	Leverage	Annual Expense Ratio	Daily Expense Ratio
Equity	IVV	BlackRock	S&P500	+1x	0.0300%	0.000119%	Crypto	BTC3L	ByBit	Bitcoin	+2x to +4x	1.8250%	0.005000%
	VOO	Vanguard	S&P500	+1x	0.0300%	0.000119%		BTC3S	ByBit	Bitcoin	-2x to -4x	1.8250%	0.005000%
	SPY	SSGA	S&P500	+1x	0.0945%	0.000375%		BTCUP	Binance	Bitcoin	+1.25 to +4x	3.6500%	0.010000%
	QQQ	Invesco	NASDAQ-100	+1x	0.2000%	0.000794%		BTCDOWN	Binance	Bitcoin	-1.25 to -4x	3.6500%	0.010000%
	TQQQ	ProShares	NASDAQ-100	+3x	0.8600%	0.003413%		BTC3L	Pionex	Bitcoin	+2.2x to +4x	10.9500%	0.030000%
	SH	ProShares	S&P500	-1x	0.8900%	0.003492%		BTC3S	Pionex	Bitcoin	-2.2x to -4x	10.9500%	0.030000%
	SSO	ProShares	S&P500	+2x	0.8900%	0.003532%		BTC3L	ByDFi	Bitcoin	+3x	10.9500%	0.030000%
	SDS	ProShares	S&P500	-2x	0.9000%	0.003571%		BTC3S	ByDFi	Bitcoin	-3x	10.9500%	0.030000%
	SPXU	ProShares	S&P500	-3x	0.9000%	0.003571%		BTC3L	KuCoin	Bitcoin	+3x	16.4250%	0.045000%
	UPRO	ProShares	S&P500	+3x	0.9100%	0.003611%		BTC3S	KuCoin	Bitcoin	-3x	16.4250%	0.045000%
	PSQ	ProShares	NASDAQ-100	-1x	0.9500%	0.003770%		BTC3L	MEXC	Bitcoin	+3x	36.5000%	0.100000%
	QLD	ProShares	NASDAQ-100	+2x	0.9500%	0.003770%		BTC3S	MEXC	Bitcoin	-3x	36.5000%	0.100000%
	QID	ProShares	NASDAQ-100	-2x	0.9500%	0.003770%		BTC3L	Gate.io	Bitcoin	+3x	36.5000%	0.100000%
	SQQQ	ProShares	NASDAQ-100	-3x	0.9500%	0.003770%		BTC3S	Gate.io	Bitcoin	-3x	36.5000%	0.100000%
	SPXL	Direxion	S&P500	+2x	1.0000%	0.003968%		BTC3L	AscendEX	Bitcoin	+3x	109.5000%	0.300000%
SPXS	Direxion	S&P500	-2x	1.0800%	0.004286%	BTC3S	AscendEX	Bitcoin	-3x	109.5000%	0.300000%		

### 5 Concluding remarks

Like leveraged ETFs, the primary goal of LVTs is to simplify investing in leveraged positions by reducing the complexities of managing such positions and limiting the risk of liquidation. During our study period, we identified numerous issues with LVTs, including a lack of transparency, custody by the issuing exchange, and possible inadequate backing. 99.9% of LVTs are implemented as centralized products, accessible only within the exchange’s ecosystem. 80% of them have no interaction with the blockchain, leading to a lack of transparency in total supply, transactions, and holders.

We examined these issues along with several financial and security concerns. The total supply of 53% of LVTs is not published by the issuers, making it difficult for investors to calculate the NAV and trade LVTs at a fair market price. Additionally, 41% of LVTs may be issued with inadequate financial backing at launch, as the required futures contracts were either issued late or may not have existed at the time of the initial offering. 97% of LVTs carry the risk of front-running during well-known events, where adversaries can potentially exploit rebalancing trades. LVTs exhibit greater leverage deviation from the stated ratio compared to LETfs, due to inconsistent fund management in tokens with fixed leverage or inefficiencies in the rebalancing algorithm in LVTs with dynamic leverage.

LVTs generally have higher management fees compared to LETfs, which impacts the fund’s ability to achieve its expected return. LVTs tend to underperform over extended periods (monthly or weekly) due to the compounding effect, making them unsuitable as a long-term investment vehicle.

All our findings point to the same conclusion: investors expecting simple leveraged positions that “just work” are likely to be disappointed by leveraged tokens. LVTs require careful consideration of their unique characteristics, making them more suitable for experienced traders.

#### Future works

Increased scrutiny from regulators and mandatory audits are potential avenues to ensure that LVTs are adequately backed. Moving LVTs on-chain could improve transparency regarding supply, transactions, and holders, while enabling self-custody. Front-running mitigation should be explored through randomized rebalancing or stealth trading (e.g., iceberg orders). LVT algorithms should be adjusted to reduce deviations from stated leverage. These measures aim to better align LVTs with investor expectations.

## References

- 1 Cloud Security Alliance. What is a pseudo cryptocurrency? <https://cloudsecurityalliance.org/blog/2019/11/25/what-s-a-pseudo-cryptocurrency/>, November 2019. [Online; accessed 16-May-2023].
- 2 Khalid Husain Ansari and Umesh Kulkarni. Implementation of ethereum request for comment (erc20) token. In *Proceedings of the 3rd International Conference on Advances in Science & Technology (ICAST)*, 2020.
- 3 AscendEX. About ascendex’s leveraged token trading mechanism. <https://ascendex.com/en/support/articles/68525-about-ascendexs-leveraged-token-trading-mechanism>, May 2023. [Online; accessed 07-May-2023].
- 4 Carsten Baum, James Hsin-yu Chiang, Bernardo David, Tore Kasper Frederiksen, and Lorenzo Gentile. Sok: Mitigation of front-running in decentralized finance. In *International Conference on Financial Cryptography and Data Security*, pages 250–271. Springer, 2022.
- 5 Binance. Binance leveraged tokens agreement. <https://www.binance.com/en/leveraged-tokens/agreement>, May 2023. [Online; accessed 07-May-2023].
- 6 ByBit. Bybit leveraged tokens: A complete get started guide. <https://learn.bybit.com/trading/bybit-leveraged-tokens-explained/>, May 2023. [Online; accessed 07-May-2023].
- 7 ByDFi. What are leveraged tokens? <https://support.bydfi.com/hc/en-us/articles/5692456018319-What-are-Leveraged-Tokens->, May 2023. [Online; accessed 07-May-2023].
- 8 Andrew Carver. Do leveraged and inverse etfs converge to zero? *Institutional Investor Journals Guide to Exchange Traded Funds*, 2009, January 2009.
- 9 Narat Charupat and Peter Miu. The pricing and performance of leveraged exchange-traded funds. *Journal of Banking & Finance*, 35:966–977, April 2011. doi:10.1016/j.jbankfin.2010.09.012.
- 10 James Chen. Open-ended fund: Definition, example, pros and cons. <https://www.investopedia.com/terms/o/open-endfund.asp>, December 2021. [Online; accessed 02-Jun-2023].
- 11 Minder Cheng and Ananth Madhavan. The dynamics of leveraged and inverse exchange-traded funds. *Journal of investment management*, 16(4):43, 2009.
- 12 Minder Cheng and Ananth Madhavan. The dynamics of leveraged and inverse exchange-traded funds. *Journal of Investment Management*, 7, January 2010.
- 13 Index Coop. How composability powers fli tokens. <https://indexcoop.com/blog/how-composability-powers-fli-tokens>, September 2021. [Online; accessed 17-May-2023].
- 14 The Index Coop. Flexible leverage index. <https://indexcoop.com/products/eth-flexible-leverage-index>, May 2023. [Online; accessed 07-May-2023].
- 15 Sunflower Corporation. What are leveraged tokens? <https://medium.com/coinmonks/what-are-leveraged-tokens-639fb186744a>, August 2022. [Online; accessed 09-Jun-2023].
- 16 Gateway To Crypto. Etf leveraged tokens. <https://www.gate.io/etf>, May 2023. [Online; accessed 07-May-2023].
- 17 Luke DeVault, Harry J Turtle, and Kainan Wang. Blessing or curse? institutional investment in leveraged etfs. *Journal of Banking & Finance*, 129:106169, 2021.
- 18 Tim Dulaney, Tim Husson, and Craig Mccann. Leveraged, inverse, and futures-based etfs. *SSRN Electronic Journal*, July 2012.
- 19 Shayan Eskandari, Seyedehmahsa Moosavi, and Jeremy Clark. Sok: Transparent dishonesty: front-running attacks on blockchain. In *Financial Cryptography and Data Security: FC 2019 International Workshops, VOTING and WTSC, St. Kitts, St. Kitts and Nevis, February 18–22, 2019, Revised Selected Papers 23*, pages 170–189. Springer, 2020.
- 20 Vitalik Buterin Fabian Vogelsteller. Erc-20 token standard. <https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md>, November 2015. [Online; accessed 17-May-2023].
- 21 Akhilesh Ganti. Understanding notional value and how it works. <https://www.investopedia.com/terms/n/notionalvalue.asp>, February 2024. [Online; accessed 17-Mar-2024].

- 22 MEXC Global. Guide to leveraged etf. <https://support.mexc.com/hc/en-001/articles/360038986011-Guide-to-Leveraged-ETF>, May 2023. [Online; accessed 07-May-2023].
- 23 Ilan Guedj, Guohua Li, and Craig Mccann. Leveraged and inverse etfs, holding periods, and investment shortfalls. *The Journal of Index Investing*, 1:45–57, November 2010. doi: 10.3905/jii.2010.1.3.045.
- 24 Adam Hayes. Understanding liquidity and how to measure it. <https://www.investopedia.com/terms/l/liquidity.asp>, September 2023. [Online; accessed 18-Dec-2023].
- 25 Joanne Hill and George Foster. Understanding returns of leveraged and inverse funds. *Journal of Indexes*, 12(5):40–58, 2009.
- 26 Will Kenton. S&p 500 index: What it's for and why it's important in investing. <https://www.investopedia.com/terms/s/sp500.asp>, September 2023. [Online; accessed 24-Mar-2024].
- 27 Leslie Kramer. Long position vs. short position: What's the difference? URL: <https://www.investopedia.com/ask/answers/100314/whats-difference-between-long-and-short-position-market.asp>, February 2024. [Online; accessed 24-Mar-2024].
- 28 KuCoin. Kucoin etf dynamic multiplier leveraged tokens. <https://www.kucoin.com/support/18437931016857>, November 2022. [Online; accessed 05-Jun-2023].
- 29 KuCoin. Leveraged token trading guide. <https://www.kucoin.com/support/sections/6024431447321>, May 2023. [Online; accessed 07-May-2023].
- 30 Martin Lettau and Ananth Madhavan. Exchange-traded funds 101 for economists. *Journal of Economic Perspectives*, 32(1):135–154, 2018.
- 31 Tim Leung and Marco Santoli. Leveraged etfs: Admissible leverage and risk horizon. *SSRN Electronic Journal*, November 2012. doi:10.2139/ssrn.2172096.
- 32 Luca J Liebi. The effect of etfs on financial markets: a literature review. *Financial Markets and Portfolio Management*, 34(2):165–178, 2020.
- 33 Anthony Loviscek, Hongfei Tang, and Xiaoqing Eleanor Xu. Do leveraged exchange-traded products deliver their stated multiples? *Journal of Banking & Finance*, 43:29–47, 2014.
- 34 P. Mackintosh. “Double trouble”. etf and indexing. *CFA Magazine*, pages 25–31, January 2008.
- 35 Cory Mitchell. Front-running definition, example, and legality. <https://www.investopedia.com/terms/f/frontrunning.asp>, May 2022. [Online; accessed 28-May-2023].
- 36 Pionex. Leveraged tokens. <https://www.pionex.com/blog/what-arepionex-leveraged-tokens/>, May 2023. [Online; accessed 03-May-2023].
- 37 Pionex. Pionex will terminate futures manual trading since 2021/6/30. <https://pionex.zendesk.com/hc/en-us/articles/900007639683-Pionex-will-terminate-Futures-manual-trading-since-2021-6-30>, May 2023. [Online; accessed 15-May-2023].
- 38 Pionex. What are pionex leveraged tokens. <https://www.pionex.com/blog/what-arepionex-leveraged-tokens/>, May 2023. [Online; accessed 07-May-2023].
- 39 Poloniex. Ftx leveraged tokens faq. <https://support.poloniex.com/hc/en-us/articles/360045644653-FTX-Leveraged-Tokens-FAQ>, May 2023. [Online; accessed 07-May-2023].
- 40 Reza Rahimian and Jeremy Clark. A shortfall in investor expectations of leveraged tokens. <https://ssrn.com/abstract=4899657>, July 2024. [Online; accessed 23-Jul-2024].
- 41 Gerasimos Rompotis. Return and volatility of emerging markets leveraged etfs. *Journal of Asset Management*, 17:165–194, May 2016. doi:10.1057/jam.2016.2.
- 42 U.S. Securities and Exchange Commission. Leveraged and inverse etfs: Specialized products with extra risks for buy-and-hold investors. <https://www.sec.gov/investor/pubs/leveragedetfs-alert>, February 2023. [Online; accessed 02-May-2023].
- 43 Mahesh Shirole, Maneesh Darisi, and Sunil Bhirud. Cryptocurrency token: An overview. In *IC-BCT 2019: Proceedings of the International Conference on Blockchain Technology*, pages 133–140. Springer, 2020.
- 44 Pauline Shum, Walid Hejazi, Edgar Haryanto, and Arthur Rodier. Intraday share price volatility and leveraged etf rebalancing. *Review of Finance*, 20(6):2379–2409, 2016.

## 23:24 A Shortfall in Investor Expectations of Leveraged Tokens

- 45 SmartContractSecurity. Smart contract weakness classification and test cases. <https://swcregistry.io/>, January 2020. [Online; accessed 25-May-2023].
- 46 Statista. Market share of largest providers of exchange traded funds (etfs) in the united states as of september 2022. <https://www.statista.com/statistics/294411/market-share-etf-providers-in-the-us/>, May 2023. [Online; accessed 13-May-2023].
- 47 FactSet Research Systems. Leveraged etf list. <https://etfdb.com/etfs/leveraged/>, April 2024. [Online; accessed 20-April-2024].
- 48 Financial Times. Investors pump record sums into leveraged etfs. <https://www.usa.gov/agencies/securities-and-exchange-commission>, November 2022. [Online; accessed 19-Feb-2024].
- 49 William Trainor. Do leveraged etfs increase volatility. *Technology and Investment*, 1:215–220, January 2010. doi:10.4236/ti.2010.13026.
- 50 William Trainor and E. Baryla. Leveraged etfs: A risky double that doesn’t multiply by two. *Journal of Financial Planning*, pages 48–55, January 2008.
- 51 Sam Werner, Daniel Perez, Lewis Gudgeon, Aria Klages-Mundt, Dominik Harz, and William Knottenbelt. Sok: Decentralized finance (defi). In *Proceedings of the 4th ACM Conference on Advances in Financial Technologies*, pages 30–46, 2022.
- 52 KOUT Wided. On the properties of leveraged etfs. *idea*, 2:2, 2019.
- 53 Liyi Zhou, Kaihua Qin, Christof Ferreira Torres, Duc V Le, and Arthur Gervais. High-frequency trading on decentralized on-chain exchanges. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 428–445. IEEE, 2021.

# Investigating Wrench Attacks: Physical Attacks Targeting Cryptocurrency Users

Marilyne Ordekian ✉ 

Department of Computer Science, University College London, UK

Gilberto Atondo-Siu ✉ 

Department of Computer Science, University of Cambridge, UK

Alice Hutchings ✉ 

Department of Computer Science, University of Cambridge, UK

Marie Vasek ✉ 

Department of Computer Science, University College London, UK

---

## Abstract

Cryptocurrency wrench attacks are physical attacks targeting cryptocurrency users in the real world to illegally obtain cryptocurrencies. These attacks significantly undermine the efficacy of existing digital security norms when confronted with real-world threats. We present the first comprehensive study on wrench attacks. We propose a theoretical approach to defining wrench attacks per criminal law norms, and an interdisciplinary empirical approach to measure their incidence. Leveraging three data sources, we perform crime script analysis, detecting incidents globally across 10 interviews with victims and experts, 146 news articles, and 37 online forums. Our findings reveal diverse groups of attackers ranging from organized crime groups to friends and family, various *modi operandi*, and different forms of attacks varying from blackmail to murder. Despite existing since Bitcoin's early days, these attacks are underreported due to revictimization fears. Additionally, unlike other cryptocurrency crimes, users with advanced security experience were not immune to them. We identify potential vulnerabilities in users' behavior and encourage cryptocurrency holders to lean into digital as well as physical safety measures to protect themselves and their cryptocurrency. We offer actionable recommendations for the security community and regulators, highlighting the double-edged sword of Know Your Customer policies.

**2012 ACM Subject Classification** Applied computing → Law; Applied computing → Digital cash; Security and privacy → Social aspects of security and privacy; Social and professional topics → Financial crime

**Keywords and phrases** cryptocurrency, Bitcoin, crime, wrench attack, physical attack

**Digital Object Identifier** 10.4230/LIPICs.AFT.2024.24

**Related Version** *Extended Version*: <https://discovery.ucl.ac.uk/id/eprint/10195033>

**Funding** *Marilyne Ordekian*: UK Engineering and Physical Sciences Research Council (EPSRC), grant No EP/S022503/1.

*Gilberto Atondo-Siu*: European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme, grant No 949127.

*Alice Hutchings*: European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme, grant No 949127.

## 1 Introduction

Since the launch of Bitcoin in 2009, cryptocurrency owners have faced a constant threat of cyberattacks, financial crimes, and emerging risks threatening the safety and security of their funds [26, 10, 44, 4]. In 2022 alone, \$3.8B was reportedly stolen from cryptocurrency users and service providers [14].



© Marilyne Ordekian, Gilberto Atondo-Siu, Alice Hutchings, and Marie Vasek; licensed under Creative Commons License CC-BY 4.0

6th Conference on Advances in Financial Technologies (AFT 2024).

Editors: Rainer Böhme and Lucianna Kiffer; Article No. 24; pp. 24:1–24:24

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

While cryptocurrencies may open users up to cyberattacks, the threat of physical attacks has not diminished. Hal Finney, a highly influential cypherpunk and computer scientist, and the first user to download and receive Bitcoin in 2009, was a victim of such attack [41]. Unlike other forms of cryptocurrency-specific/facilitated crime [10, 52], this threat targets users physically outside the cyber world. These attacks, also known as “wrench attacks,” target users in the real world to illicitly acquire their cryptocurrencies or the means of access.<sup>12</sup>

The term **\$5 wrench attack** first appears in the webcomic, XKCD [59]. The comic describes two characters discussing a physical attack using a \$5 wrench to force the victim to provide information rather than orchestrating a cyberattack. This term has been adopted in the cryptocurrency space [35], hence the terminology we use throughout this paper.

Five aspects distinguish cryptocurrency wrench attacks from their digital counterparts and make them a serious threat requiring attention. First, the crime scene is in the physical world rather than the digital, thereby endangering the physical security and safety of users. Second, the conventional *modi operandi* distinguish these, as attackers forgo the technical skills required to bypass cybersecurity measures and revert to primitive tools and methods reminiscent of conventional crimes, such as violence, robberies, extortion, etc. Third, wrench attacks are crimes against persons and property; targets are not just property and ownership, but also people (users). Fourth, wrench attacks challenge existing cybersecurity measures, as no existing security measure can ensure that the funds of a victim with a gun pointed at them are secure. Fifth, everyone is a potential victim, as attackers do not distinguish between old and new users, professional traders and amateurs, or levels of security awareness.

To deeply understand this emerging threat, we investigate the following research questions:

**RQ1:** What are wrench attacks? What distinguishes them from other crimes?

**RQ2:** How do wrench attacks work? Considering the different types, stages, *modi operandi*, attackers, and repercussions.

**RQ3:** How do users perceive this threat? How can they and the cryptocurrency industry best defend against wrench attacks?

We take an interdisciplinary approach to answer these research questions. We collect three separate datasets and implement data triangulation to overcome biases that may be present in a single dataset. First, we collect *forum posts* from 37 online forums and programmatically parse out wrench attack-related content. We also conduct in-depth semi-structured *interviews* with 10 victims and experts. Finally, we analyze 147 incidents reported in 146 *news articles*.

**Contributions.** To our knowledge, this is the first investigation of cryptocurrency wrench attacks. Our contributions are the following:

- We collect three novel datasets: interviews, news articles, and forum posts. We combine common analysis methods from computer science along with legal and crime science methods in a way new to the computer science audience.
- In the absence of legal and scholarly definitions, we craft the first definition of a wrench attack. Each form of a wrench attack involves at least one form of traditionally recognized crime, e.g. robbery; we systematically contextualize these crimes within a wrench attack. Our definition allows wrench attacks to be separately measured and studied.

---

<sup>1</sup> Acquiring cryptocurrencies often happens when a victim is forced to transfer their cryptocurrencies to the attacker; whereas acquiring the means of access is where an attacker gains direct access to a user’s wallet. We discuss this distinction further in §3.

<sup>2</sup> “Means of Access” incorporates digital means (e.g. private key, wallet password) and physical means (e.g. hardware devices like cold wallets or computers) allowing access and/or control of cryptocurrencies.



- We perform a crime script analysis and identify seven forms of wrench attacks dating back to 2014, including violent crimes, aggravated thefts, and a new form of domestic abuse we pin as cryptocurrency-facilitated domestic economic abuse.
- We identify new physical and cyber security vulnerabilities in cryptocurrency users' behaviors. Accordingly, we devise recommendations for users, policymakers, software designers, and other stakeholders.

## 2 Background

In Section 2.1 we overview prior work which touches upon cryptocurrency physical attacks. Then in Section 2.2 we explain our methodology, crime script analysis.

### 2.1 Cryptocurrencies and Physical Attacks

Cryptocurrency users encounter a diverse range of threats, with prior work categorizing these threats based on varied levels of risk [26, 1]. These user-centered threats span cybersecurity and technical risks, financial and economic risks, and social and legal risks [26, 1, 10, 48].

Physical attacks have been briefly acknowledged in prior work as a source of threat to users, however, none comprehensively and specifically investigate wrench attacks or physical attacks targeting cryptocurrency users. Froehlich et al. identify physical attacks as one of six threats faced by cryptocurrency users; they focus on the devices or physical objects, without considering the harms or attacks directed towards users [26]. Voskobochnikov et al. explore the concerns of cryptocurrency users, including physical safety and the fear of a gun being held to their heads [57]. Other works explore the reasons for the non-adoption of cryptocurrencies, highlighting the fear of physical safety as a factor for avoidance [56]. Empirical work examining mobile wallets identifies physical safety concerns as well like the fear of phones being snatched whilst making mobile payments [58]. There has been some work into making Bitcoin wallets more secure, including against physical attacks [29, 6], though the threat models for these improved techniques are often not robust against a coercive physical attacker.

### 2.2 Crime Script Analysis

Crime script analysis is a methodology from the crime science field used to systematically identify the stages carried out when committing a specific crime. These stages include actions preceding, during, and following the commission of a crime [17, 18] where a criminal event encompasses specific actors, tools, actions, locations, and motivations. By unraveling the necessary processes to commit a crime, this approach provides a deeper understanding of how crimes are committed, situational factors, and other influences. Crime scripting is an emerging method for identifying intervention approaches derived from different fields. Crime scripts can be developed with a diverse range of data, including police reports and interviews, and are developed by explicitly recording the steps and stages involved in the process.

Researchers can use crime scripts to understand various types and classes of crimes [20]. These include complex crimes like organized crimes or financial crimes which incorporate a longer process, more actors, more preparation, and often a mixture of a few different classes of crimes [34, 28, 16].

### 3 Definition and Crime Steps of Wrench Attacks

There is currently no definition of a wrench attack in legislation or academic work, making it difficult to measure the scope of such attacks. Other work investigates threats with measurable, technical definitions (e.g. malware is determined by analyzing network traffic, files changed, etc., or some signature found in the code itself [5]), however, physical crime does not yield itself to technical definitions. Instead, we use legal methods derived from criminal law to formally define these attacks committed in the physical world. This assists in the subsequent measurement of the incidents.

Criminal courts and law enforcement agencies utilize national criminal codes or laws to break down an act into steps; this process determines whether an act is punishable by law, and if so, what type of punishment it entails. According to criminal law principles, an act is considered “criminal” only if it is defined in the law and its steps are outlined [60]. This is the universal concept of “no punishment without law”.<sup>3</sup> These defined steps are referred to as crime elements; they constitute a checklist used to determine whether an act follows predetermined steps and requires penalizing the perpetrator.

Crime elements consist of two main components: the *Mens Rea* element, also known as the “guilty mind”, represents the criminal intent of a perpetrator; and the *Actus Reus* element, or the “guilty act”, refers to the physical element of a crime, i.e. physical conduct(s) that constitutes a crime [31]. The *Actus Reus* requires a 1) act, 2) result, and 3) causation [31].

We propose a definition outlining the steps (crime elements). To craft this definition, a criminal law expert on our team examined the English common law and French civil law, both key references for legal systems worldwide. Analyzing the French “code pénal” and English criminal law, provides insights into crime elements and how they can be adapted and distilled into steps; hence a checklist [36, 50]. Using this method, we propose our definition of a wrench attack, create its specific crime elements and aid in understanding how it unfolds from planning to execution.

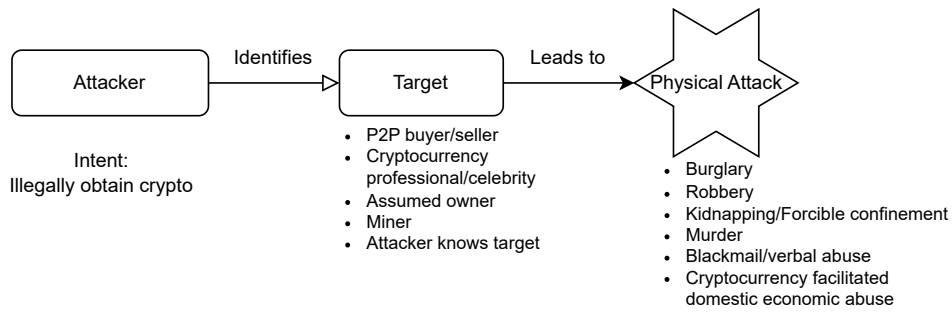
**Definition.** We define wrench attacks as the physical targeting of cryptocurrency owners with the intention to gain unlawful possession and ownership of their cryptocurrencies by means of physical force or threat of force or harm. The act combines offences against property, and offences against natural persons.

**Elements.** Our proposed elements for wrench attacks are detailed in Table 1; we define these elements per legal norms and provide a loose understanding for a general audience.

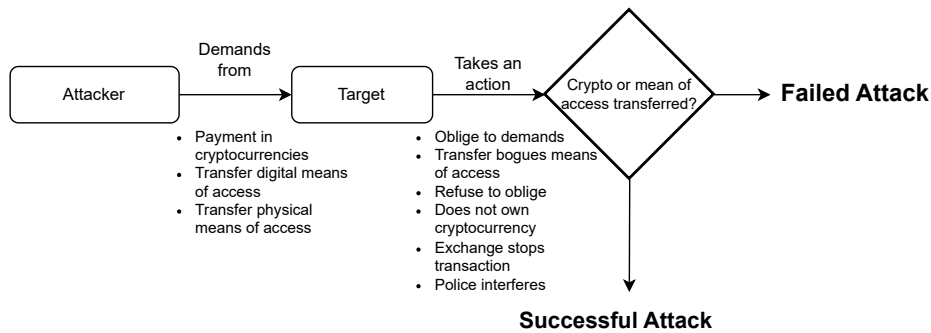
Wrench attacks are intentional crimes and cannot occur accidentally. Furthermore, similar to many crimes, they have additional unique requirements, such as “property” and the property “belonging to another” (here the victim’s cryptocurrency or means of access). The targeted “property” is owned or possessed by someone other than the attacker, as the attack itself will shift that possession from the victim to the attacker. Finally, wrench attacks can take seven different forms (Table 3), yet not all can result in success; some are failed attempts i.e. for reasons not intended by the perpetrator, the desired outcome does not occur. Though, as demonstrated in Table 5, most of the attacks were successful.

---

<sup>3</sup> This is also known as the Principle of Legality in criminal law. It was developed in the 18th century by Cesare Beccaria [7].



■ **Figure 1** Anatomy of a Wrench Attack: Preparation.



■ **Figure 2** Anatomy of a Wrench Attack: During and After.

**Anatomy of a Wrench Attack.** We translate this definition and elements into a step-by-step systematic guide on how wrench attacks are committed. This anatomy is presented in Figures 1 and 2, which break up the attack into events precursing the physical attack (Fig. 1) and events occurring during the attack resulting in the outcome (Fig. 2).

**Exclusion Criteria.** Our proposed definition acts as an inclusion criterion as it outlines what qualifies as a wrench attack. By following this definition, we exclude scenarios like insider threats (not targeting an individual’s cryptocurrency) and attacks on physical infrastructure (not targeting a person). This is detailed further in the extended paper.

## 4 Methodology

Informed by the definition outlined in §3, we use data triangulation, a research method that uses multiple datasets, methods, and approaches to answer a research question [13, 53, 21]. The goal of data triangulation is to enhance validity and credibility. Therefore, we implement three different research designs and data sources to present a comprehensive understanding of wrench attacks; these three datasets are later used to perform the crime script analysis in §5. We present a mixture of qualitative and quantitative research designs, collecting data via interviews §4.1, forum posts §4.2, and news articles §4.3. Table 2 summarizes our datasets.

■ **Table 1** Crime elements of wrench attacks per our proposed definition and scope.

Element	Definition	Loose Understanding
Property	Funds in possession (i.e. cryptocurrencies) or the means of access that provide the right to access and transfer funds, such as keys, passwords, and seed phrases.	What the attackers desire to get through the attack (cryptocurrency).
Belonging to Another	The perpetrators are aware that the property subject of the attack, at the time of the attack, is under the possession or control of another “person” (natural or legal person).	Perpetrators are aware that the funds belong to someone else.
Act	Willed and controlled bodily movements [42]. Acts are detailed in Table 3.	What the attacker did.
Result	Appropriation; in this case, transfer of possession (of the funds) i.e. the victim is permanently or for a prolonged period deprived of their ownership, as offenders assume the legal rights over the victim’s property (i.e. cryptocurrencies or the means of access).	Attacker must take (or forcibly lend from) the victim’s cryptocurrency.
Causation	The conduct of using force or threat of force or harm caused the acquisition of the means of access and/or the transfer of funds.	The attacker’s conduct itself caused the harm or damage to the victim and led to their loss of funds.
<i>Mens Rea</i>	Wrench attacks are intentional acts. We consider: 1) general intention where the offender is aware of the nature of the conduct and has a desire to perform it, 2) specific intention where the offender intends to permanently deprive the victim of their funds or means of access.	Attacker must intend to steal cryptocurrency.
Attempt	1) Acquiring the means of access, but failing to transfer the coins, 2) acquiring means of access, but the wallet contains no funds, 3) failure to acquire genuine means of access from the victim; i.e., faulty means of access, 4) the victim does not give in to the threats or assault, 5) the victim does not or no longer has a wallet(s)/funds/or access to the means of access.	The attacker’s conduct failed to deliver the desired outcome i.e. acquiring the cryptocurrencies.

## 4.1 Interviews

We conducted semi-structured interviews to gain a deeper understanding of wrench attacks, victimization process, user susceptibility and security behaviors that either ignite or prevent wrench attacks. We interviewed three groups of users: 1) victims, 2) people who personally know a victim, and/or 3) academics or industry personnel actively involved in the cryptocurrency ecosystem.

### 4.1.1 Recruitment

Cryptocurrency owners in general are difficult to survey [3, 2]. Identifying participants for wrench attacks is even more challenging due to the sensitive nature of these incidents. We took measures to ensure potential victims felt safe coming forward and speaking with us while maintaining their privacy during initial contact. When advertising the interviews, we initially advertised to people who knew a victim and cryptocurrency experts. This was crucial. All victims we interviewed initially signed up to participate as experts, but during the interviews, they disclosed that they were victims.

■ **Table 2** Summary of wrench attack data sources and incidents. Reported incidents are filtered via our criteria to yield our wrench attack dataset.

Source	Dataset Size	Reported Incidents	Wrench Attacks
Interviews	10	11	11
News articles	146	147	105
Forum posts	672	54	3

■ **Table 3** The main acts involved in the wrench attacks from our dataset of news articles. The majority involved more than one act, but incidents are sorted here based on the dominating act.

Act	Africa	Asia	Europe	N America	Oceania	S America	Unspecified	Total
Burglary	0	9	20	7	1	1	0	38
Kidnapping	1	12	8	1	0	2	0	24
Robbery	0	9	4	9	0	1	0	23
Forcible Confinement	1	2	2	2	0	0	0	7
Murder	1	3	2	0	0	0	0	6
Blackmail	0	2	0	1	0	0	0	3
Cryptocurrency Facilitated	0	1	0	1	0	0	1	3
Domestic Economic Abuse								
Fraud	0	0	1	0	0	0	0	1

We followed a multi-step recruitment process. We reached out to academics and cryptocurrency experts, securing five interviews. We contacted 98 attendees of an academic information security conference, obtaining a further 5 interviews. Despite efforts to engage public figures, we received no responses here. Finally, we posted interview invitations on Bitcointalk [8]; this yielded eight comments but no participants.

We outlined our rigorous security measures and spent weeks building trust with participants to gain their consent to participate. Our recruitment focused on gathering personal experiences, excluding participants informed of wrench attacks solely by news reports. In total, we conducted 10 interviews both online and in person.

### 4.1.2 Interview Schedule

We employ a semi-structured interview schedule. The interview schedule comprises 7 sections and 2 main categories: establishing and identifying the occurrence and characteristics of a wrench attack, and a series of questions about the security behavior and risk assessment of participants, general and cryptocurrency-specific demographics, and recommendations for mitigating wrench attacks. Overall, the final schedule includes 59 questions with a duration ranging from 35 to 60 minutes. The interview schedule is included in the extended paper.

### 4.1.3 Profile of Participants

Our sample of 10 interviews includes industry/academic experts, out of which 6 were victims or directly associated with victims, reporting 11 wrench attacks. We report general demographics in the extended paper. As for **cryptocurrency-specific demographics**,

most participants have over four years of experience with cryptocurrencies, with about half being early adopters. Three report using peer-to-peer (P2P) in-person transactions, which we outline as a major risk factor in §5.1.1, while a minority (two) use ATMs. Notably, all use centralized exchanges such as Binance, hence all underwent Know Your Customer (KYC) verification. Half the participants knew of specific breaches on exchanges they used; the rest either assumed their exchange had been breached or were entirely unaware.

Half the sample, especially those residing in financially unstable countries, rely on cryptocurrencies as an alternative payment method. Three use cryptocurrencies for research or as a store of value. Nearly all participants own multiple cryptocurrencies, with Bitcoin being the most common.

## **4.2 Forum Posts**

In order to ensure comprehensiveness, we search for additional reports on social media.

Our first data source is the CrimeBB dataset [46], created in 2018, which amalgamates underground forum data. This dataset is available for academic research use under a data-sharing agreement with the Cambridge Cybercrime Centre. We search through ~110 million posts made by 6 million members from 36 underground forums (some of which have been active since 2007) including HackForums and Dread. This yielded no wrench attack reports.

We additionally use the online forum Bitcointalk. Satoshi created this in February 2009 and it is the largest cryptocurrency-focused forum with more than 3.5M members as of January 2024. We crawl through over 45M posts from July 2010 until August 2023. We use machine learning to classify our data, as we detail in the extended paper. Our classification yielded 672 posts about wrench attacks including 3 victim narratives. We also parsed out links to news articles yielding two additional news articles not already included in Section 4.3. One of these articles referred to two different wrench attacks, therefore three incidents were added to our news article dataset (§4.3).

## **4.3 News Articles**

We use an up-to-date list of news articles curated by cryptocurrency expert Jameson Lopp [37]. The list includes publicly reported physical attack cases involving cryptocurrencies. We collect 144 news articles available from December 2014 through October 2023, reporting 144 unique incidents. As outlined in §4.2, our analysis of Bitcointalk yields 2 additional news articles reporting 3 incidents. This yields a total of 146 news articles reporting 147 incidents.

We apply our definition (§3) as a selection criterion. This excludes 42 articles, leaving 104 news articles reporting 105 wrench attacks, which we use in our analysis.

## **4.4 Coding and Analysis**

We analyze the three datasets qualitatively. Qualitative analysis provides deep insights into a subject matter beyond mere quantification. The coding of the data was inductive and data-driven, with codes and themes derived directly from the data [27]. Coding of wrench attack-related sections of data followed Cornish’s universal crime script scenes [17]. There is no single universal script, as it can be adapted and used diversely, depending on the complexity of the crime and its composition. In conducting this crime script, we borrow from Hutchings et al. [33], where the script is adapted and divided into three acts tacitly reflecting the original nine tracks as proposed by Cornish [18, 17].

## 4.5 Ethics

This work uses data obtained through interviews, online forums, and news articles. The ethics committee at the Department of Computer Science & Technology, University of Cambridge, approved this research. Our recruitment process was covered by this remit. Interview participants were provided with an overview of the research before providing informed consent. All interview data was stored locally until transcription. Transcripts exclude any information identifying the participant or third parties, and the recordings were deleted along with emails and any other records that contained participants' personal data. Participants were advised that they were free to withdraw from the study at any time and could opt to not answer any of the questions asked.

Our forum data and news articles were extracted from publicly accessible sources. In our analysis, we paraphrased any quoted text to limit searchability. Furthermore, this work focuses on analyzing aggregate information and collective behavior of online communities using publicly available data and under the British Society of Criminology's Statement of Ethics, it falls outside the requirement of informed consent [12].

## 4.6 Limitations

Crime research tends to have limitations due to the hidden nature of offenses, with victims often being unwilling to report, and incidents that are reported are not necessarily similar to those that are not. We aim to reduce these limitations by triangulating three data sources, using data relating to public disclosures of attacks (media reports), anonymous disclosures (forum posts), and victim accounts (interviews).

Additional limitations include privacy and personal safety concerns led some potential participants (victims) to opt against participating, this limited the variety of perspectives included in the study. Furthermore, while the captured experiences of the victims vastly enriched the dataset, and the recruitment process proved to be immensely challenging, the generalizability of the sample is constrained.

There are additional limitations related to the forum analysis. Our Bitcointalk dataset represents approximately 75% of the forum (as of August 2023). We crawl historic forums, so removed posts are excluded. Our use of specific keywords to create our training sample may add an inherent bias. Thus, we might not include all posts that are wrench attack-related.

# 5 Crime Script Analysis

Wrench attacks involve a combination of crimes, with the main aim being financial gain. The key element that facilitates this goal is targeting individuals. Thus, wrench attacks are possible by a combination of actions targeting both individuals and their personal property. We analyze these attacks using three datasets, dividing each incident into 3 parts: Preparation (Act 1), attack (Act 2), and the aftermath (Act 3). This allows us to encompass all crimes documented in our datasets.

## 5.1 Act 1: Preparation

When preparing a physical attack against a victim, the physical location and the primitive tools and methods utilized in perpetrating the offense play a pivotal role.

### 5.1.1 Actors

There are two main actors identified in wrench attacks, the victim(s) and the offender(s). Actor roles differ depending on circumstances. We find no notable distinction or a pivot on a specific type of users. Our three datasets reveal a variety of offending actors, indicating the absence of a singular or specific type of dominant perpetrators for wrench attacks. However, we do note the prevalence of co-offending compared to solo offending (Table 4b).

**Over the Counter (OTC) brokers or peer-to-peer (P2P) transactors.** In-person P2P operations are a prevalent method of exchanging cryptocurrencies with fiat or other cryptocurrencies. P2P transactions usually take place in person and do not require service providers or KYC verification, nor does it necessarily engage the banking system. It is also a prevalent approach embraced by those who are unbanked or underbanked, allowing them an alternative to transfer funds locally and globally.

Based on our interview sample, in three instances the offender(s) were either OTC brokers or P2P transactors. Of the 104 inspected news articles, 25 reported incidents involving P2P transactions, while we found two victims with similar encounters on Bitcointalk.

However, OTC brokers can also be targeted by attackers. One of the authors informally spoke to an OTC broker whose shop was targeted on multiple occasions. The victim preferred not to be interviewed for security reasons.

**Accepting payments in cryptocurrencies.** The offender here is a person accepting cryptocurrencies in exchange for goods. In our interview sample, the victim was in a bar, reimbursed a person in Bitcoin for buying them a round of beer, only for this person to attack the victim and snatch their phone after learning about their Bitcoin ownership.

**Family, friends, and business partners.** Offenders may also be acquaintances, business partners, family members, and romantic partners; i.e. persons who know the victims and are aware of their involvement with cryptocurrencies. The involvement of these individuals might either be as a principal perpetrator, or by being a secondary party (accessory) that aids, abets, procures or counsels the principal(s) offenders. This applies to five incidents in our interview dataset but only eleven in the news articles study (Table 4b).

**Organized crime groups.** There are indications that crime groups are involved in wrench attacks. We note that the role of organized crime groups in technology-related crime can often be overstated [32, 38], so we refrain from quantifying this to avoid inaccurate assumptions about group offenders.

**Victims as offenders.** We record one incident in our interviews and three in the news articles where the offenders were former victims seeking revenge through their attack.

**Corrupt law enforcement agents as offenders.** Corrupt law enforcement agents could either abuse their badges or misuse confidential information gained through police records. Our news articles dataset includes five such incidents.

### 5.1.2 Crime Location

**Real physical world.** A factor setting wrench attacks apart from other cryptocurrency-related crimes is their occurrence in the physical realm. This entails direct physical contact between the offender and the victim, involving face-to-face or direct contact like calling the victim on their private number.



■ **Table 4** Factors in different wrench attacks (news articles).

(a) Tools used per each crime type.

Tool	Burglary	Kidnapping	Robbery	Forcible Confinement	Murder	Blackmail	Domestic Violence	Fraud
Physical Violence	19	15	7	6	2	0	0	0
Firearm	13	5	6	0	1	0	0	0
Offensive Weapon	2	0	5	0	1	0	0	0
Spiking	1	0	0	0	0	0	3	0
Legal Extortion	0	0	0	1	0	1	0	0
Swatting	0	0	0	0	0	1	0	0
Unspecified	3	4	5	0	2	1	0	1

(b) Type of offender carrying out wrench attack and their relationship. Each victim outlined independently so numbers add to more than 105.

	Solo	Group	Total
Strangers	13	91	104
Non strangers	2	9	11
Total	15	100	

**No favorable environment.** Wrench attacks manifest across a diverse spectrum of locations and environments. Crime scenes span populated public streets, commercial establishments like shops, private residences, and secluded locales. This was unexpected, particularly the number of instances of violent crimes on busy streets in broad daylight.

**Geographically.** The attacks in our interview series span South America, Europe, Asia, and the Middle East. In the news article dataset, we find attacks occurring in all continents, with the predominant ones being Europe and Asia (Table 3).

### 5.1.3 Target Selection

We differentiate between random and non-random selection, whether victims are chosen specifically because of an identified association with cryptocurrencies or entirely at random. In our interview dataset, all targets were selected non-randomly. Offenders had varying degrees of knowledge or familiarity with victims, choosing them based on a presumed holding of cryptocurrencies. This prior knowledge could stem from acquaintanceship, transactional meetups, investigation of assumed ownership, and publicly available information e.g. the victim is a known cryptocurrency professional/figure.

In the news articles dataset, detailed information on the victim selection process or prior relationship was inconsistent. Hence, we omit implied information on the randomness of the selection, and only record cases where either a prior relationship existed between the victim and the offender (11) or the victim is a professional/public figure in the space (27).

### 5.1.4 Attacks over Time

Interviewees refrained from disclosing precise dates of attacks to avoid identification, but indicated timeframe; spanning from the early days (2011-2012) to the 2017/18 ICO boom and beyond. Despite a broad distribution of attacks over the years, the rate of attacks increased notably at the end of 2017; this coincides with Bitcoin reaching (at the time) an all-time high. This trend is evident in both the interview and articles datasets, with the second-highest recorded articles (20) reported in 2018. The highest number of attacks (25) is noted in 2021, following the return from Covid-19 lockdowns and the all-time high price of Bitcoin nearing \$65,000.

### 5.1.5 Tools or Attack Methods

Wrench attacks rely on conventional methods of committing crime. Many wrench attack offenders resort to physical assault (crimes against persons). The majority of incidents involved weapons, tools, or objects that could inflict harm. Other methods involved imposing physical restrictions, spiking, etc. Table 4a outlines tools used per each crime type. Physical violence and firearms are mostly used in burglaries and kidnapping; robberies use both as well as offensive weapons (usually knives). Spiking is only used in domestic violence cases.

### 5.1.6 Motivation

The overarching aim of wrench attacks is to secure substantial funds. The resort to physical attacks originates from two primary motivations. First, some find it easier to illegally acquire cryptocurrencies through physical means rather than resorting to sophisticated cyberattacks.

Second, targeting affluent individuals outside the cryptocurrency space is challenging as forcing victims to make large bank payments is difficult. Unlike bank payments, there is no threshold for transferred funds in a single transaction. Additionally, offenders benefit from the absence of comprehensive and global regulatory requirements, simplifying the unrestricted transfer and cash-out process of cryptocurrencies.

## 5.2 Act 2: Methods

Wrench attacks are mostly perpetrated in line with other crimes. The current act explores the various methods (tracks) by which wrench attacks are committed. As a reminder, the primary goal of the attackers is financial gain, particularly to illicitly gain cryptocurrencies. Section 5.2.2 details the demands made by attackers to achieve their goal.

### 5.2.1 Tracks

These tracks outline variations in the wrench attack crime script found in our three datasets. We summarize the findings from the news articles in Table 3.

**Track: Attacks on personal liberty.** Kidnapping and forcible confinement violate the personal liberty of the victim. Kidnapping requires abducting and relocating someone by force or deception [54]; in forcible confinement, the victim's freedom of movement is confined, i.e. they are not relocated nor abducted [39]. In both cases, the aim is financial gain, either directly through the victim or by demanding a ransom from family members. Offenders primarily use physical violence, among other methods to commit this (Table 4a).

One of our interview participants was kidnapped and cuffed by acquaintances, and was forced to hand over a hardware wallet under verbal threats. Notably, five incidents in the news data involved corrupt law enforcement agents, with victims being forcibly taken to police stations and extorted by fake police reports and accusations in return for cryptocurrencies.<sup>4</sup> Another notable method involved offenders impersonating law enforcement agents or posing as fake investors and kidnapping victims during a business meeting in a foreign country.

Bitcointalk users express fears of kidnapping, especially fears of loved ones being kidnapped for a Bitcoin ransom or corrupt government officials leaking information to criminals.

---

<sup>4</sup> We examine the Corruption Perceptions Index (CPI) rank for the countries involving these corrupt law enforcement agents [51]. These incidents occurred in India (rank 85), Ukraine (rank 116), and Nigeria (rank 150).

**Track: Violent crimes.** Some wrench attacks have resulted in **murder**. In our interviews, an interviewee describes a wrench attack involving a murder, where the victim was kidnapped into a jungle by a contract killer hired by the victim’s business partner. The news articles dataset includes six murder cases, all occurring after the 2017 ICO boom. Notably, two cases involved victims of investment scams turning into wrench attack offenders, murdering scammers who had deceived them into investing in cryptocurrencies.

**Track: Crimes against property.** **Burglary**, which entails trespassing a private premise to commit theft [30], is the most common form of a wrench attack reported in the media. As seen in Table 4a, burglaries can be hostile as they are the crime type mostly associated with physical violence and possessing firearms. In three distinct cases, the wrench attack took the form of a heist, where offenders broke into cryptocurrency firms or service providers (e.g. exchange), and assaulted employees. In the remaining incidents, the victims in most cases were either cryptocurrency experts, consultants, miners, or bloggers who publicly discussed cryptocurrencies.

In our interview dataset, a burglary incident involves breaking into a cryptocurrency user’s home to take over their funds. Bitcointalk users have also been concerned about burglary as early as 2014, even though a user refers to the idea as “absurd”, stating: *“How would a potential attacker with a gun even identify which house to break in? This scenario seems more like fiction and spreads unnecessary fear.”* **Robberies** are also committed with the use of firearms or physical force (Table 4a), but unlike burglaries, they can occur anywhere. We see a direct relation between these incidents and P2P transactions. Our interviews reveal two cases of armed robberies, in Europe and the Middle East. Both were involved in public P2P transactions between buyers/sellers during which the victims were held at gunpoint. In one case, the armed robbery escalated further into a car chase. Our interviews also include a victim who was mugged in a pub while making a Bitcoin payment with their phone. The offender upon seeing the displayed amounts of Bitcoin on the screen, stabbed the victim and fled with the phone. The news media includes 23 incidents of robberies, 17 occurring during P2P transactions in North America and Europe. One Bitcointalk post recounts an armed robbery by a gang during a P2P transaction in Europe. Another Bitcointalk user reports an attempted mugging during a P2P transaction, where the offender failed to successfully snatch their phone whilst transferring Bitcoin.

**Track: Blackmail or verbal abuse.** Many of the tracks also involve the use of blackmail/extortion and verbal abuse. Here, we only report instances occurring independently of any other crimes. **Blackmail** here ranges from threatening to reveal private, damaging, or embarrassing information about the victim, or threatening to harm them or a relative or a friend, unless they comply with their demands [25]. There also exists “legal” tools used in extortion, such as threatening to report someone to the police or sue them in court.

Our interview participants reported several instances of blackmail and threats. Victims report being extorted with old and/or intimate pictures of them that could damage their reputation. The offenders were previous friends, previous romantic partners, and random strangers claiming to possess such images. In one incident, the offender used the victim’s family to exercise pressure. In another case, the offender extorted and threatened the interviewee with legal actions promising to get them into legal trouble.

**Verbal abuse** takes many forms, ranging from harassment, threats, hate speech, to insulting or abusive language. The intent is to cause the victim distress and intimidation, harass them, and/or create an unpleasant and unsafe environment. In wrench attacks, the

■ **Table 5** The distinct demands made by wrench attackers in our news articles dataset, and the outcome of the attack divided between failed attempts and successful ones.

Attacker Demand(s)	Count	Crime Outcome	Count
Cryptocurrencies (no specification)	40	Successful	70
Means of access (private keys, storing device, etc.)	30	Failed	29
Specifically requested only Bitcoin	26	Unspecified	6
Unspecified	9		

offender has ulterior motives, i.e. obtaining the victim’s cryptocurrencies. The victims we interviewed disclosed instances of verbal abuse, mostly by friends, distant family members, or acquaintances who knew the victim owned cryptocurrencies. One victim was stalked by their harasser; another incident involved the harassment of a woman during P2P transactions.

Both blackmail and verbal abuse were reported more frequently in our interview dataset (six incidents) compared to the news articles dataset (three incidents). One reason for this difference may be that news articles recount crimes that have been reported to law enforcement, and blackmail and verbal abuse might be under-reported or not taken seriously.

**Track: Cryptocurrency-facilitated domestic economic abuse.** When an intimate partner or family member exercises economic abuse to take over their victim’s cryptocurrencies, we are faced with a combination of acts: a wrench attack and a new term we pin as cryptocurrency-facilitated domestic economic abuse. In our interview dataset, an intimate partner coerces and/or harms their partner to take unlawful possession of their cryptocurrencies. This form of economic abuse cases also occurs outside long-term intimate partner relations, such as in family settings or new romances. The news articles dataset records three cases of such abuse. In two cases, the offender and victim had a short romantic relationship after meeting on an online dating app. The other case involves a son stealing his father’s funds. Notably, this track primarily occurs through spiking (Table 4a).

### 5.2.2 Offender Demands

The primary goal of wrench attackers is to illegally acquire cryptocurrencies through physical means. However, not all attackers coerce their victims to transfer cryptocurrencies, instead, we find in our datasets a variety of requests made by offenders, as shown in Table 5.

**Demanding the transfer of cryptocurrencies.** In person, the offender(s) coerces the victim to personally transfer cryptocurrencies. In a successful attempt, the victim under duress, transfers cryptocurrencies to a designated address. Many offenders specifically ask for Bitcoin, however, other cryptocurrencies are demanded as well.

**Demanding means of access: Storage device.** The victim is coerced in person to transfer the storage device, e.g. a hardware wallet, a mobile phone, or a computer. Often the offender(s) has prior knowledge that a device exists. Consequently, the device holding cryptocurrencies is transferred.

**Demanding means of access: Access information.** The victim is coerced in person to reveal the private key and/or any other digital security layer that grants full access and control of the funds. Access demands are not limited to a specific type of wallet, e.g. if the victim uses a mobile wallet, the offender(s) ask for the phone PIN and the wallet app credentials. Here, there is a reveal of access/control information.

**Fraud during P2P transactions.** Unlike the previous scenarios, the perpetrator resorts to deception here. The perpetrator and victim meet in person to exchange cryptocurrencies/ fiat. Once the victim makes a transfer, the perpetrator refuses to transfer the equivalent funds they had initially agreed on. The offender often verbally abuses or threatens the victim if they refuse to oblige.

### 5.3 Act 3: Attempt or Completion

The third Act includes the actions that take place following the commission of the crime, as detailed in Act 2.

#### 5.3.1 Crime Outcome

**Successful appropriation – successful wrench attack.** A successful wrench attack involves the successful transfer of funds to offenders, or their acquisition of either a storage device(s) or means of access.

**Failed attempt – failed wrench attack.** Failed attempts occur when for any reason, the offenders do not end up with the victim’s cryptocurrencies or the means of access. While not all media articles provide information on the outcome of the crime, of those that did, 28 incidents resulted in failed attempts. Attempts are typically thwarted through no funds being available in the targeted wallet, fictitious means of access, or the victim not submitting to the offender’s demands.

#### 5.3.2 Role of Law Enforcement

**Under reporting.** The media reports include just 105 incidents reported between 2014 and October 2023. Of the 11 incidents discussed in the interviews, only two incidents were reported to the police. Our interview participants had decided not to report due to a number of concerns. These included privacy and security considerations, as they were concerned that exposing themselves as cryptocurrency owners could create further risks. Others wanted to avoid future complications with the same offenders, as they lacked confidence in law enforcement agencies. Some victims highlighted that they thought their case might not be taken seriously, or they were hesitant about the outcome. This under-reporting is consistent with other research on online property crime [49].

**Shortcoming in involvement.** Law enforcement involvement varies, which can be ascribed to several factors. During the early days of Bitcoin, cryptocurrencies were often trivialized as “magic Internet money” which led to minimal law enforcement interest. One interviewee held at gunpoint in public, reported the incident to authorities. As they state: *“From the start, it was ignored.”* Another early-day victim, posting their experience with attempted street robbery on Bitcointalk, questioned the usefulness of law enforcement: *“I’ll report the incident to the police, but I’m doubtful anything good will come out of it.”*

In recent years, the involvement of law enforcement seems to increase due to cryptocurrencies gaining more popularity and value. We can conclude this from the reporting in media (§5.1.4). Yet, not all law enforcement agencies have the capabilities or access to tools that assist in dealing with cryptocurrency crime. This can be extended to wrench attacks.

The limited role of law enforcement in usefully addressing wrench attacks helps motivate our effort in thoroughly defining wrench attacks. While all of the attacks we study were crimes and therefore under the purview of law enforcement, few were reported and even fewer still were investigated. One role of definitions is to highlight attention in understudied areas.

### 5.3.3 Post Attack Alert

Among the victims we interviewed, a minority chose to alert the community, the rest were hesitant. This hesitancy is observed in our online forum posts dataset, as a minority chose to share their experience. The methods of alerting others varied. Some opted to post on online cryptocurrency communities such as Bitcointalk, or other public platforms such as podcasts. Others notified local groups through Telegram or WhatsApp. Nevertheless, most were inclined to preserve their status as cryptocurrency users and decided to remain silent.

## 6 Security Behaviors and Risk Perception

The cryptocurrency userbase has become more diverse over time [9, 47, 3]. Abramova et al. [3] suggest a new typology that groups users into three clusters (cypherpunks, hodlers, and rookies) based on their risk perceptions and security behavior. Contrary to this, we find no relationship between user experience or security awareness and wrench attack victimization.

During the interviews, we were interested in understanding participants' security behaviors, threat assessment, and perceptions of past/future wrench attacks. This could assist in recognizing behaviors or knowledge gaps among users that increase risk or make them more favorable targets for attack. Our objective is not to engage in victim blaming, but rather discern proactive measures to counteract potential attackers.

### 6.1 Threat Assessment

We explore users' threat assessment relating to their cryptocurrency ownership. Participants communicated concerns about the potential exploitation of personal data as a precursor to a wrench attack. Here, they expressed distrust towards cryptocurrency service providers (e.g. exchanges) collecting excessive personal data including government IDs, biometrics, etc., necessary for KYC verification. Ordekian et al. highlight that existing AML/CFT policies applied within the cryptocurrency space have inadequacies that could cause more harm than good, especially relating to the security of personal information gathered for KYC verification [45]. An interviewee expressed these concerns, stating: *"... I have to provide a driver's license to buy a \$10 NFT... But if my identity gets compromised as a result of making a transaction, it's a much higher risk, and that's purely created by the government."*

### 6.2 Wrench Attack Risk Perception

Existing literature identifies vulnerable groups and behaviors that predispose users to vulnerabilities in the cryptocurrency ecosystem: security breaches, poor security behavior, and self-inflicting errors. Understanding one's vulnerability to potential security threats, coupled with precautionary security behavior, influences informed security decision-making [55]. Hence, we investigate two key aspects: 1) the risk perceptions of both users and victims, and 2) their confidence in their existing security measures in thwarting future wrench attacks.

**Risk perception.** We asked participants about the likelihood they would experience a wrench attack in the future. For victims, we inquired if they anticipate experiencing a wrench attack again. Half anticipate the possibility, with the remaining feeling secure for diverse reasons. One participant felt secure as they resided in a jurisdiction with a low crime rate. Others believe they are unlikely targets as they own insignificant amounts of cryptocurrencies, primarily for research and curiosity purposes. However, we note many

wrench attack victims are targeted because of their affiliation with the cryptocurrency sector, as attackers presume ownership. Hence, we challenge the assumption that limited funds ownership reduces susceptibility when affiliation exists.

**Confidence level in security practices.** Participants varied in their confidence that their security practices were effective against wrench attacks. Three expressed confidence, while others emphasized situational nuances, like the type of attack or the attacker's knowledge and skill level. A security expert was also concerned that attackers might target family members as an easier route to reach them.

Geographical location was identified by two participants as a key factor affecting their confidence level; one avoided certain countries due to security concerns. Moreover, confidence levels varied based on the wallet type. Online or mobile wallets were considered less secure and easier to steal.

**Perpetrators possessing key information.** In a scenario where attackers possessed information enabling fund access, 7 out of 10 participants doubted their security measures. Concerns were voiced again about the security of user information held by service providers, with participants noting that if an exchange is breached, a successful wrench attack would be possible. These concerns of exchange data breaches [3] align with prior work investigating the adverse consequences, like social engineering attacks users face due to leaked data [2].

### 6.3 Repeat and Multiple Victimization

Victims with a history of victimization may be at a higher risk of future victimization [19]. Understanding this and identifying patterns in victimization, such as crime types, specific environments, and the dynamics of victimization, assists in informing preventative measures.

**Repeat victimization.** We find being a wrench attack victim does not grant immunity against future incidents. Though a sensitive topic, two participants reported multiple wrench attack incidents, suggesting their public figure status and being early adopters as contributing factors to this.

**Multiple victimization.** Our interviewees report being the victims of non-cryptocurrency cybercrime. Three wrench attack victims recounted constant phishing attacks via email or SMS attempts to gain unauthorized access. Another victim reports a smart contract exploit having their NFT wallet drained. One victim thwarted a romance scam attempt. Two of the wrench attack victims attributed their multiple targeting to their fame, with one reporting online stalking and the other being impersonated with fake cryptocurrency projects and scams being promoted in their name.

### 6.4 Post Wrench Attack Changes

Following an attack, two participants spoke openly about behavioral changes. The first emphasized the significance of alertness, awareness, and openly discussing incidents to alert the community. The second participant mentioned avoiding carrying significant cryptocurrency amounts, especially during P2P transactions.

## 7 Recommendations and Intervention Areas

In this section, we outline several recommendations for interventions to help prevent wrench attacks. These recommendations are informed by suggestions made by security experts we interviewed as well as our expertise. Cryptocurrency holders may have different risk appetites and exposure, so they may choose to implement what makes sense to their individual situation. We also address intermediaries who can help prevent or mitigate wrench attacks.

### 7.1 Precautionary Measures for Users

In this section, we outline recommendations for users that could aid them in protecting against wrench attacks.

#### 7.1.1 Keeping a Low Profile

Eight out of ten interviewees emphasize keeping a low profile to avoid targeting. This includes refraining from bragging, flashing wealth, and disclosing financial details. Some advise not disclosing holding funds entirely, others suggest not specifying the held amount. An interviewee explained: *“We disclose we hold, we disclose we deal, but we never disclose the amount so that we don’t become more of a target.”*

Besides maintaining secrecy, users should be careful when discussing cryptocurrencies, since eavesdroppers and discussants have turned into adversaries. Users are recommended to discuss cryptocurrencies only with trusted persons and refrain from public advertisement of their ownership, even on online forums with pseudonyms which can still be identifiable.

#### 7.1.2 Fund Management

To prudently manage funds, strategic approaches encompass wealth distribution and storage. Geographical distribution of funds or means of access was recommended. This practice involves spreading wealth across regions to mitigate localized threats and reduce losses. Storage diversification adds an extra layer of protection, minimizing exposure to a single point of failure and enhancing overall resilience. Using multifaceted approaches by mixing hot and cold wallets helps users avoid losing everything at once. Three interviewees describe this as *“not keeping your eggs in one basket.”*

#### 7.1.3 Digital and Physical Safety

Considering the nature of wrench attacks, a combination of digital and physical security measures can best protect against them.

**Digital safety.** Multisignature wallets are recommended for securely storing cryptocurrency. This method mandates  $m$  signatures out of a possible  $n$  to access funds. Regarding wrench attacks, these wallets could give victims plausible deniability that the victim would be unable to transfer the funds. The tradeoff here is that while requiring more signatures could make it harder for attackers to steal funds, it can be harder for users to set up and potentially easier for a user to lose their funds. Other digital safety measures include using 2FA on cryptocurrency online platforms or creating read-only wallets. Both of these measures would allow victims to be unable to transfer funds or otherwise add time/friction.



**Physical safety.** Physical security is crucial in addressing wrench attacks. Situational awareness is key, considering that different geographical locations pose varying risk levels. By staying attuned to these risks, users can adjust their behaviors to reduce potential exposure to threats. Some interviewees feel safer discussing cryptocurrencies in a country with generally low crime rates; emphasizing that the risks associated with wrench attacks are similar to other crimes, as it all depends on location. Others consider being in countries with wider and massive cryptocurrency adoption increases risk exposure, requiring extra caution.

Safety measures are necessary. In addition to keeping a low profile more generally (§7.1.1), users are recommended to avoid revealing their location in advance of travel and limit sharing personal information. Additionally, it is important to ensure personal safety during in-person cryptocurrency gatherings, particularly around due diligence on the identities and intentions of individuals attending these gatherings to minimize the risk of malicious encounters.

### 7.1.4 Peer-2-Peer Specific Measures

In-person cryptocurrency transactions are quite common, especially in countries with limited access to banking, financial crises, or under international sanctions [15]. Yet, this method carries risks due to direct physical contact between transactors. In the incidents reported in the news articles, 25% of cases occurred during P2P transactions.

There are two primary precautions for P2P transactions. First, exercise diligence with the seller/buyer by assessing trustworthiness before the meet-up. Users should avoid meeting random or potentially risky individuals, especially alone, have an escape plan, and choose crowded public areas with access to police. Second, exercise diligence with transactions, starting with smaller transactions to build mutual trust. Users are advised to avoid carrying large sums of funds, and only bring what is necessary. An additional recommended layer of diligence is validating large transactions and considering time-delaying transfers.

## 7.2 Collaborative Initiatives and Interventions

Stakeholders including governments, the cryptocurrency industry, and the community, can help protect users against wrench attacks. This section details intervention strategies.

### 7.2.1 Know Your Customer Policies

KYC processes are increasingly imposed by governments on cryptocurrency service providers (e.g. exchanges) to combat money laundering and terrorist financing. KYC verification involves collecting/storing/sharing personal information including physical addresses, government IDs, financial data, etc. [23, 24]. Yet, the porous security of these businesses made them highly susceptible to data breaches [43, 44, 40]. This increases the risk for users, making them potential targets for both cybercrime and wrench attacks [2].

One participant expresses how KYC verification could ignite wrench attacks: “[...] *government requirements for KYC, AML [with centralized exchanges], I would say your criminal organization that’s operating in some country that has essentially ability to act in an area, they would get a list of customers of exchanges that are in that area and then they have to know which of these people [exchange customers] are approachable and everything else [...]. So the government requirements that you provide identity [KYC process] actually creates like a shopping list for criminals for those kinds of stuff [wrench attacks].*”

Cryptocurrency users have voiced privacy fears over KYC verification and the substantial collection of personal information, as a minority have already been targets for physical threats following data leaks [58, 2]. Legal academics also argue that the extensive information

## 24:20 Investigating Wrench Attacks: Physical Attacks Targeting Cryptocurrency Users

collected by cryptocurrency service providers for KYC compliance poses a security risk to users, highlighting the unsuitability of already existing anti-money laundering regulations for the cryptocurrency industry [45]. Hence, governments should either reconsider some of these policies that are criticized in the banking system for not ideally achieving required aims [11], or impose higher security standards on these service providers.

### 7.2.2 Cryptocurrency Exchanges

Cryptocurrency exchanges play the role of an intermediary. They can delay or stop certain transactions going through their services. In two incidents from the news articles dataset, the wrench attackers, who successfully coerced the victims to initiate a transfer, failed to fully receive the cryptocurrencies as the transactions went through exchanges. The latter exchanges had a 24-hour delay/verification feature which enabled victims to flag the transactions and stop them. While some exchanges implement this process for large transactions in compliance with AML/CFT policies, these processes are not standard.

### 7.2.3 Educational Efforts

Educational resources and awareness could help non-tech-savvy users understand basic concepts like fund/key management, safe storage, and protective security measures. Participants stressed the importance for the public to be aware of emerging risks, such as wrench attacks.

## 7.3 System Design Change

This section proposes areas for system design changes.

### 7.3.1 Cryptocurrency Protocols

Cryptocurrencies themselves can be designed to keep their users safe against wrench attacks. Better protocol properties like zero knowledge protocols can assist in hiding how much a user holds. If implemented and used broadly, these can also increase privacy on a protocol level where it is impossible to tell which users are a part of which transactions. This limits information attackers can glean on potential victims.

### 7.3.2 Wallet Software Underpinnings

Wallet software could, for instance, allow the user to create wallets with false proofs of no funds. This could thwart potential attacks where a victim could show the false proof which could be validated by the attacker. Mechanisms for easy recovery of wallets could allow users to take back their money before the transaction is on the network. Making the software of hardware wallets seamless and changing how seed phrases are handled would make the use of backup wallets more straightforward. While this might not fully thwart known attackers, it could help mitigate the impact, particularly with users who currently rely on online or mobile wallets to store all their funds.

### 7.3.3 Wallet Interface Design

The user interface of cryptocurrency wallets could be changed to allow more security for the users against physical attacks. For instance, not showing transaction history/details would allow users to hide their behavior. Similarly, displaying on the main screen of the app/service the amount that a user has in their wallet is a known threat (we have a victim in our interviews

who got stabbed because the offender saw their Bitcoin holdings on their phone screen). Early research demonstrates that users are rightly concerned here [58]. Not all victims are necessarily tech-savvy – a user-friendly interface while broadly useful, could help thwart attacks, since many users struggle with cryptocurrency wallet user interfaces [58, 57, 22].

## 8 Conclusion

There have been substantial recent efforts towards securing cryptocurrency infrastructure against digital threats. This has caused some offenders to pivot towards more antiquated methods of stealing, namely by physical force or threat.

Wrench attacks are a novel, yet unsophisticated, type of crime that is increasing in frequency. While compared to other forms of cryptocurrency crimes, wrench attacks are less prevalent, yet, their outcome is more hazardous. This not only imperils users but also impacts the trust in the space. This is particularly worrying for users residing in countries experiencing financial unrest, who have sought refuge in cryptocurrencies as an alternative [15].

The media primarily reports cryptomillionaires or dramatic incidents, but we find many attacks go unreported. There is no adequate regulatory landscape here, and existing technical defenses seem obsolete. Hence, this paper is an urgent plea to tackle this issue. Our contributions extend beyond identifying this issue; they serve as the foundation for regulators, researchers, and stakeholders to collaborate in developing strategies to mitigate the adverse risks posed by these attacks.

Wrench attacks are an example of criminals eschewing sophisticated methods of committing crime, and reverting to old-school tactics to exploit new technologies. By acknowledging these methods, we can better protect users and alleviate the spread of these attacks. Future work should investigate how regular users are being identified and whether there is a relation with data breaches.

---

## References

- 1 Svetlana Abramova and Rainer Böhme. Perceived benefit and risk as multidimensional determinants of bitcoin use: A quantitative exploratory study. In *Proceedings of the International Conference on Information Systems - Digital Innovation at the Crossroads*, 2016.
- 2 Svetlana Abramova and Rainer Böhme. Anatomy of a High-Profile data breach: Dissecting the aftermath of a Crypto-Wallet case. In *32nd USENIX Security Symposium (USENIX Security 23)*. USENIX Association, 2023.
- 3 Svetlana Abramova, Artemij Voskobojnikov, Konstantin Beznosov, and Rainer Böhme. Bits under the mattress: Understanding different risk perceptions and security behaviors of crypto-asset users. In *Conference on Human Factors in Computing Systems*, pages 1–19, 2021.
- 4 Ross Anderson, Chris Barton, Rainer Böhme, Richard Clayton, Carlos Gañán, Tom Grasso, Michael Levi, Tyler Moore, Stefan Savage, and Marie Vasek. Measuring the changing cost of cybercrime. In *18th Workshop on the Economics of Information Security (WEIS)*, 2019.
- 5 Michael Bailey, Jon Oberheide, Jon Andersen, Z Morley Mao, Farnam Jahanian, and Jose Nazario. Automated classification and analysis of internet malware. In *Recent Advances in Intrusion Detection*, pages 178–197. Springer, 2007.
- 6 Tobias Bamert, Christian Decker, Roger Wattenhofer, and Samuel Welten. Bluewallet: The secure bitcoin wallet. In *Security and Trust Management: 10th International Workshop*, pages 65–80, 2014.
- 7 Cesare Beccaria. *On crimes and punishments*. Transaction Publishers, 2016.
- 8 BitcoinTalk. URL: <https://bitcointalk.org/>.

- 9 Apolline Blandin, Gina C Pieters, Yue Wu, Anton Dek, Thomas Eisermann, Damaris Njoki, and Sean Taylor. 3rd global cryptoasset benchmarking study. *Cambridge Centre for Alternative Finance Reports*, 2021. URL: <https://www.jbs.cam.ac.uk/wp-content/uploads/2021/01/2021-ccaf-3rd-global-cryptoasset-benchmarking-study.pdf>.
- 10 Rainer Böhme, Nicolas Christin, Benjamin Edelman, and Tyler Moore. Bitcoin: Economics, technology, and governance. *Journal of Economic Perspectives*, 29(2):213–238, 2015.
- 11 Rainer Böhme, Johanna Grzywotz, Pesch Paulina, Christian Ruckert, and Christoph Safferling. Bitcoin and alt-coin crime prevention, 2017. URL: <https://www.bitcrime.de/presse-publikationen/pdf/BITCRIME-RegulRep.pdf>.
- 12 British Society of Criminology. Statement of ethics, 2015. URL: <https://www.britsocrim.org/ethics/>.
- 13 Nancy Carter, Denise Bryant-Lukosius, Alba Dicenso, Jennifer Blythe, and Alan Neville. The use of triangulation in qualitative research. *Oncology Nursing Forum*, 41:545–547, September 2014.
- 14 Chainalysis. 2023 crypto crime trends: Illicit cryptocurrency volumes reach all-time highs amid surge in sanctions designations and hacking, 2023. URL: <https://blog.chainalysis.com/reports/2023-crypto-crime-report-introduction/>.
- 15 Chainalysis. The 2023 global crypto adoption index, 2023. URL: <https://www.chainalysis.com/blog/2023-global-crypto-adoption-index/>.
- 16 Yi-Ning Chiu, Benoit Leclerc, and Michael Townsley. Crime script analysis of drug manufacturing in clandestine laboratories: Implications for prevention. *The British Journal of Criminology*, 51(2):355–374, March 2011.
- 17 Derek B Cornish. Crimes as scripts. In *International Seminar on Environmental Criminology and Crime Analysis*, volume 1, pages 30–45. Florida Criminal Justice Executive Institute, 1994.
- 18 Derek B Cornish. The procedural analysis of offending and its relevance for situational prevention. *Crime prevention studies*, 3(1):151–196, 1994.
- 19 Sara Giro Correia. Patterns of online repeat victimisation and implications for crime prevention. In *2020 APWG Symposium on Electronic Crime Research (eCrime)*, pages 1–11. IEEE, 2020.
- 20 Hashem Dehghanniri and Hervé Borrión. Crime scripting: A systematic review. *European Journal of Criminology*, 18(4):504–525, 2021.
- 21 Norman K Denzin. Triangulation. *The Blackwell encyclopedia of sociology*, 2007.
- 22 Shayan Eskandari, David Barrera, Elizabeth Stobert, and Jeremy Clark. A first look at the usability of bitcoin key management. *Workshop on Usable Security (USEC)*, February 2015.
- 23 European Union. Regulation (EU) 2023/1113 of the european parliament and of the council of 31 may 2023 on information accompanying transfers of funds and certain crypto-assets and amending directive (EU) 2015/849, 2023.
- 24 FATF. Guidance for a risk-based approach to virtual assets and virtual asset service providers, 2019. URL: <https://www.fatf-gafi.org/publications/fatfrecommendations/documents/guidance-rba-virtual-assets.html>.
- 25 George P Fletcher. Blackmail: The paradigmatic crime. *University of Pennsylvania Law Review*, 141(5):1617–1638, 1993.
- 26 Michael Froehlich, Philipp Hulm, and Florian Alt. Under pressure. a user-centered threat model for cryptocurrency owners. In *4th International Conference on Blockchain Technology and Applications*, pages 39–50, 2021.
- 27 Graham R Gibbs. *Analyzing qualitative data*. SAGE Publications Ltd, 2018.
- 28 Nicholas Gilmour. Understanding money laundering. a crime script approach. *The European Review of Organised Crime*, 1(2):35–56, 2014.
- 29 Andriana Gkaniatsou, Myrto Arapinis, and Aggelos Kiayias. Low-level attacks in bitcoin wallets. In *International Conference on Information Security*, pages 233–253, 2017.

- 30 Wim Hardyns and Anneleen Rummens. Predictive policing as a new tool for law enforcement? recent developments and challenges. *European journal on criminal policy and research*, 24:201–218, 2018.
- 31 Jonathan Herring. *Criminal law: Text, cases, and materials*. Oxford University Press, USA, 2014.
- 32 Alice Hutchings. Crime from the keyboard: organised cybercrime, co-offending, initiation and knowledge transmission. *Crime, Law and Social Change*, 62:1–20, 2014.
- 33 Alice Hutchings. Leaving on a jet plane: the trade in fraudulently obtained airline tickets. *Crime, Law and Social Change*, 70(4):461–487, 2018.
- 34 Alice Hutchings and Thomas J Holt. A crime script analysis of the online stolen data market. *British Journal of Criminology*, 55(3):596–614, 2015.
- 35 James Jones. Protect yourself against \$5 wrench attacks, 2017. URL: <https://steemit.com/bitcoin/@jamesjones/protect-yourself-against-usd5-wrench-attacks>.
- 36 Légifrance. Code pénal, 1992. URL: [https://www.legifrance.gouv.fr/codes/texte\\_lc/LEGITEXT000006070719/2024-05-23](https://www.legifrance.gouv.fr/codes/texte_lc/LEGITEXT000006070719/2024-05-23).
- 37 Jameson Lopp. Known physical bitcoin attacks. URL: <https://github.com/jlopp/physical-bitcoin-attacks/>.
- 38 Jonathan Lusthaus. How organised is organised cybercrime? *Global Crime*, 14(1):52–60, 2013.
- 39 Donna L Mailloux and Ralph C Serin. Sexual assaults during hostage takings and forcible confinements: Implications for practice. *Sexual abuse: a journal of research and treatment*, 15:161–170, 2003.
- 40 Patrick McCorry, Malte Möser, and Syed Taha Ali. Why preventing a cryptocurrency exchange heist isn’t good enough. In *Security Protocols XXVI: 26th International Workshop*, pages 225–233. Springer, March 2018.
- 41 Robert McMillan. An extortionist has been making life hell for bitcoin’s earliest adopters, 2014. URL: <https://www.wired.com/2014/12/finney-swat/>.
- 42 Michael S Moore. *Act and crime: The philosophy of action and its implications for criminal law*. Oxford University Press, 2010.
- 43 Tyler Moore and Nicolas Christin. Beware the middleman: Empirical analysis of bitcoin-exchange risk. In *Financial Cryptography and Data Security*, pages 25–33. Springer, April 2013.
- 44 Tyler Moore, Nicolas Christin, and Janos Szurdi. Revisiting the risks of bitcoin currency exchange closure. *ACM Transactions on Internet Technology (TOIT)*, 18(4):1–18, 2018.
- 45 Marilyne Ordekian, Ingolf Becker, and Marie Vasek. Shaping cryptocurrency gatekeepers with a regulatory “trial and error”. In *Workshop on the Coordination of Decentralized Finance*, pages 113–133. Springer, May 2023.
- 46 Sergio Pastrana, Daniel R Thomas, Alice Hutchings, and Richard Clayton. Crimebb: Enabling cybercrime research on underground forums at scale. In *Proceedings of the 2018 World Wide Web Conference*, pages 1845–1854, 2018.
- 47 Michel Rauchs, Apolline Blandin, Kristina Klein, Gina C Pieters, Martino Recanatini, and Bryan Zheng Zhang. 2nd global cryptoasset benchmarking study. *Cambridge Centre for Alternative Finance Reports*, December 2018. URL: <https://ideas.repec.org/p/jbs/altfin/201812-sgcbs.html>.
- 48 Corina Sas and Irni Eliana Khairuddin. Design for trust: An exploration of the challenges and opportunities of bitcoin users. In *Conference on Human Factors in Computing Systems*, pages 6499–6510, May 2017.
- 49 Maria Tcherni, Andrew Davies, Giza Lopes, and Alan Lizotte. The dark figure of online property crime: Is cyberspace hiding a crime wave? *Justice Quarterly*, 33(5):890–911, 2016.
- 50 Theft act, 1968. URL: <https://www.legislation.gov.uk/ukpga/1968/60>.
- 51 Transparency International. Corruption perception index, 2022. URL: <https://www.transparency.org/en/cpi/2022>.

- 52 Arianna Trozze, Josh Kamps, Eray Arda Akartuna, Florian J Hetzel, Bennett Kleinberg, Toby Davies, and Shane D Johnson. Cryptocurrencies and future financial crime. *Crime Science*, 11:1–35, 2022.
- 53 Scott F Turner, Laura B Cardinal, and Richard M Burton. Research design for mixed methods: A triangulation-based framework and roadmap. *Organizational Research Methods*, 20(2):243–267, 2017.
- 54 Rodanthi Tzanelli. Capitalizing on value: Towards a sociological understanding of kidnapping. *Sociology*, 40(5):929–947, 2006.
- 55 Paul Van Schaik, Debora Jeske, Joseph Onibokun, Lynne Coventry, Jurjen Jansen, and Petko Kusev. Risk perceptions of cyber-security and precautionary behaviour. *Computers in Human Behavior*, 75:547–559, 2017.
- 56 Artemij Voskoboynikov, Svetlana Abramova, Konstantin Beznosov, and Rainer Böhme. Non-adoption of crypto-assets: Exploring the role of trust, self-efficacy, and risk. In *European Conference on Information Systems*, 2021.
- 57 Artemij Voskoboynikov, Borke Obada-Obieh, Yue Huang, and Konstantin Beznosov. Surviving the cryptojungle: Perception and management of risk among north american cryptocurrency (non) users. In *Financial Cryptography and Data Security*, pages 595–614. Springer, 2020.
- 58 Artemij Voskoboynikov, Oliver Wiese, Masoud Mehrabi Koushki, Volker Roth, and Konstantin Beznosov. The u in crypto stands for usable: An empirical study of user experience with mobile cryptocurrency wallets. In *Conference on Human Factors in Computing Systems*, pages 1–14, 2021.
- 59 XKCD. Security, 2009. URL: <https://xkcd.com/538/>.
- 60 Lucia Zedner. *Criminal justice*. Oxford University Press, 2004.

# Adaptive Curves for Optimally Efficient Market Making

Viraj Nadkarni   

Princeton University, NJ, USA

Sanjeev Kulkarni   

Princeton University, NJ, USA

Pramod Viswanath   

Princeton University, NJ, USA

---

## Abstract

Automated Market Makers (AMMs) are essential in Decentralized Finance (DeFi) as they match liquidity supply with demand. They function through liquidity providers (LPs) who deposit assets into liquidity pools. However, the asset trading prices in these pools often trail behind those in more dynamic, centralized exchanges, leading to potential arbitrage losses for LPs. This issue is tackled by adapting market maker bonding curves to trader behavior, based on the classical market microstructure model of Glosten and Milgrom. Our approach ensures a zero-profit condition for the market maker's prices. We derive the differential equation that an optimal adaptive curve should follow to minimize arbitrage losses while remaining competitive. Solutions to this optimality equation are obtained for standard Gaussian and Lognormal price models using Kalman filtering. A key feature of our method is its ability to estimate the external market price without relying on price or loss oracles. We also provide an equivalent differential equation for the implied dynamics of canonical static bonding curves and establish conditions for their optimality. Our algorithms demonstrate robustness to changing market conditions and adversarial perturbations, and we offer an on-chain implementation using Uniswap v4 alongside off-chain AI co-processors.

**2012 ACM Subject Classification** Theory of computation → Market equilibria

**Keywords and phrases** Automated market makers, Adaptive, Glosten-Milgrom, Decentralized Finance

**Digital Object Identifier** 10.4230/LIPIcs.AFT.2024.25

**Related Version** *Full Version:* <https://arxiv.org/pdf/2406.13794> [47]

**Funding** This work was supported by the National Science Foundation via grants CNS-2325477, the Army Research Office via grant W911NF2310147, a grant from C3.AI, a SEAS Innovation grant from Princeton University and a gift from XinFin Private Limited.

## 1 Introduction

Market making plays a crucial role in enhancing liquidity in financial systems. In traditional finance, effective market making strategies involve setting buy and sell prices (bid and ask quotes) as narrowly separated as possible, ensuring these quotes closely mirror the asset's true price on a limit order book. This strategy enables market makers to earn a marginal profit. Such efficacy in market making is driven by sophisticated models that analyze trader behavior [25, 34, 27]. These models have become fundamental in understanding the principles of microeconomics and the microstructure of markets.

**Market Makers in DeFi.** In the field of Decentralized Finance (DeFi), the concept of *automated* market making has gained prominence. DeFi employs Automated Market Makers (AMMs), particularly Constant Function Market Makers (CFMMs) [45][2], offering an



© Viraj Nadkarni, Sanjeev Kulkarni, and Pramod Viswanath;  
licensed under Creative Commons License CC-BY 4.0

6th Conference on Advances in Financial Technologies (AFT 2024).

Editors: Rainer Böhme and Lucianna Kiffer; Article No. 25; pp. 25:1–25:22

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 25:2 Adaptive Curves for Optimally Efficient Market Making

alternative to traditional limit order books. This approach reduces the computational effort needed to facilitate trades and ensures liquidity is available for tokens that are less frequently traded. Unlike conventional markets that primarily utilize limit order books for Peer-to-Peer transactions, decentralized markets implement a Peer-to-Pool-to-Peer structure. In this model, Liquidity Providers (LPs) aggregate their resources in a contract, which traders then utilize to meet their liquidity needs. Thus, any DeFi market needs to incentivize both the LPs and the traders to ensure a fair and efficient market is created.

**Market depth and volatility.** DeFi markets exhibit a range of distinct characteristics, primarily differentiated by market depth (or liquidity) and price volatility. Notably, the trading volume of stablecoins (approximately \$11.1 trillion) has recently exceeded the transaction volumes of centralized entities like MasterCard and PayPal [31]. Markets with significant liquidity, especially those trading stablecoins, are highlighted in this context [35]. Such markets typically experience minimal volatility, and their substantial depth minimizes the price impact of retail trades. Conversely, DeFi features hundreds of infrequently traded tokens, which suffer from a lack of liquidity, leading to high volatility and price sensitivity to even small-scale retail trades. These markets are also susceptible to swings in price caused by flash loan transactions [59]. This paper addresses the optimization of market making strategies for the latter (less liquid) kind of markets.

**Incentives of LPs.** A key challenge for Constant Function Market Makers (CFMMs) is motivating Liquidity Providers (LPs) to contribute their tokens to the pool. For this incentive to work, it is crucial for CFMMs to minimize the average losses on pooled assets. Yet, it is widely acknowledged that LPs often incur losses due to fluctuations in reserves [36] and a lack of market insights [43]. This paper concentrates on reducing the losses that stem from such informational deficiencies. Specifically, static curves in CFMMs frequently lead to LP losses as a result of arbitrage activities. These losses are intended to be offset by transaction fees, contrasting with centralized exchanges which benefit from higher liquidity and trading volumes but impose lower fees. For example, Binance, a centralized exchange, records a daily trading volume of approximately \$15 billion, significantly higher than Uniswap's \$1.1 billion [9], the largest decentralized exchange. The lower liquidity on platforms like Uniswap results in less current prices, making them more susceptible to arbitrage losses.

**Arbitrage Loss.** The specific type of arbitrage loss known as loss-versus-rebalancing (LVR) can be quantified in certain scenarios [43], and these losses continue to occur despite the implementation of trading fees [41]. In the case of a generic market maker who sets bid and ask prices for a volatile asset, arbitrage losses are defined relative to the asset's true market price. An arbitrageur engages in a buy transaction when the market price surpasses the ask price, and in a sell transaction when it drops below the bid price. The resultant loss for the market maker is calculated as the product of the price difference and the volume of the asset traded.

**Trader behavior.** In traditional financial systems, arbitrage-related losses are conceptualized as *adverse selection* costs, which arise from interactions with *informed traders* – those who are privy to the external market price, akin to arbitrageurs. A market maker achieves optimal operation by balancing these costs against the profits gained from *uninformed traders*, also known as *noise traders*. This balancing principle was first delineated by Glosten and Milgrom [25]. Within the Decentralized Finance (DeFi) ecosystem, trading parties



are differentiated into *toxic* and *non-toxic* order flows, which correspond to informed and uninformed traders, respectively [48, 15]. We extend this model to a more nuanced framework where traders are categorized along a continuous spectrum of information awareness, ranging from highly informed to completely uninformed, rather than being strictly classified as either *toxic* or *non-toxic* (Section 3).

**CFMMs as prediction markets.** For CFMMs, a significant portion of their losses also arises from the need to encourage traders to disclose their genuine price perceptions during transactions. In essence, CFMMs provide compensation to informed traders in exchange for their crucial market insights, which mirrors the principles of market scoring rules utilized in prediction markets to extract valuable information [24].

**Conditions for optimality.** A straightforward approach to reduce losses to arbitrage would be aligning the marginal price exactly with the external market price, which would require real-time data from a *price oracle* [16]. Yet, integrating oracles with market making strategies can lead to potential frontrunning risks [37] and necessitates reliance on centralized, potentially manipulable external entities [21, 67]. To circumvent these issues, our framework explicitly excludes the use of oracles. The objective is to deduce the hidden market price by analyzing trade history data, aiming for maximum efficiency in terms of data utilization. Further, the market maker uses this to adaptively set its bonding curve so that the loss to arbitrageurs is as close to zero as possible, ensuring an optimally efficient market. The market maker turning a profit would be undesirable since this would allow a competitor to undercut its prices and take away their order flow. In other words, it should quote an efficient and competitive market price, given only the information it has in form of the trading history. Keeping this objective in mind, we outline the key contributions of this work.

## Our contributions

**Optimal algorithms for adapting curves.** (Section 4) We provide the differential equation that the demand curve of an optimally efficient market should follow (Theorem 1). When the statistics governing trader and price behavior are known and Gaussian/Lognormal, we show that this differential equation can be solved exactly using a dynamic bonding curve that changes its operating point using the Kalman Filter (Theorem 2 and Theorem 3).

**Adapting to unknown market conditions.** When the statistics governing trader and price behavior are unknown, we extend the previous approach by using an Adaptive Kalman Filter. We empirically show that both these approaches suffer significantly lower arbitrage losses compared to a static CFMM. (Section 5.2)

**Robustness to adversarial manipulation.** In presence of irrational traders that seek to make the price of the market maker deviate from the external market, we present a robust version of the adaptive curve algorithm that tolerates upto 50% of trader population being adversarial. (Section 5.3)

**Comparisons with static curves.** (Section 6) We provide theoretical comparisons of static and the proposed adaptive curve models by showing that the error in the adaptive AMM price, when viewed as an oracle, decays with more trades, while that of a static curve remains unchanged (Theorem 4).

**Implied dynamics of static curves.** (Section 6) We derive a differential equation (30) that governs the *implied* dynamical model given the static CFMM curves that are used in practice. We show that these CFMMs can only be optimally efficient in a model where inter-block time for the underlying blockchain vanishes, the CFMM has more liquidity than the external market, and that the trader and price behaviour is severely constrained. (Theorem 5)

**On-chain Implementation.** We specify the end-to-end system design for our proposed market maker. Furthermore, we provide an implementation of our algorithm using the recently released Uniswap v4 [57] platform and an off-chain machine learning co-processor Axiom [51]. This co-processor provides guarantees that the algorithms derived in this work are executed, and the result is put securely on-chain. (Section 8)

## 2 Related work

In this section, we reprise relevant literature surrounding the problem formulated in this paper. Although the motivation of the problem stems from literature studying AMMs in DeFi, our formulation derives heavily from classical works in market microstructure. The algorithms that we present in our work derive heavily from literature on control and robust filtering.

**Automated Market Makers.** Automated Market Makers (AMMs), particularly in the form of Constant Function Market Makers (CFMMs) [66, 45], are designed to incentivize trades that align prices with a more liquid external market [3]. However, this mechanism imposes costs on CFMM liquidity providers while generating profits for arbitrageurs [23, 29, 61]. This arbitrage profit, often quantified as “loss-versus-rebalancing” in scenarios where only arbitrageurs (informed traders) interact with the market maker, is proportional to external price volatility [43]. Various methods have been proposed to capture this loss, including on-chain auctions [39] and the application of auction theory to dynamically recommend ask and bid prices for an AMM [42]. Another recent study [26] suggests an optimal curve for a CFMM based on liquidity providers’ price beliefs. However, this study does not consider a dynamic model where traders react to market maker price settings. In [42], a dynamic trading model is examined, deriving optimal ask and bid prices. Yet, this approach necessitates the market maker’s knowledge of underlying model parameters, limiting adaptability to market conditions. Additionally, while [42] focuses on a monopolistic market maker, our work examines a *competitive* market maker. Our research aligns closely with [46], which addresses competitive market conditions but only within a liquid, non-volatile market with restricted trade sizes. Reinforcement learning algorithms to adapt CFMM bonding curves have been explored in [14], though the primary objectives there are fee revenue control and minimization of failed trades.

**Optimal market making.** The trader behavior model we employ is derived from the Glosten-Milgrom model [25], which is widely used in market microstructure literature. However, we modify it to incorporate a continuously changing external price. Several subsequent studies [17, 18] develop optimal market-making rules within a modified Glosten-Milgrom framework, but they assume that the underlying model parameters are known and that external price changes are communicated to the market maker. A more data-driven reinforcement learning approach is taken in [12], but their reward function presupposes direct information about the external hidden price, whereas we assume no access to a price oracle. Another aspect of

optimal market making in traditional market microstructure literature focuses on inventory management [30, 7], rather than the information asymmetry between traders and market makers. Our goal is to design a market maker that mitigates losses due to information asymmetry, similar to the Glosten-Milgrom model, without imposing inventory constraints. The Glosten-Milgrom model has been considered for AMMs in DeFi, though only for individual trades [5, 4].

**Optimal filtering and control.** The classical filtering and control literature underpins many contemporary automated systems, offering theoretical foundations and practical applications for dynamic system regulation. Kalman filtering, a robust statistical method, is widely used for estimating the state of a linear dynamic system from incomplete and noisy measurements [32]. This technique combines real-time measurements with prior estimates to produce updated predictions, proving crucial for systems requiring high accuracy and responsiveness, such as navigation systems, aerospace engineering, and automated trading. In control theory, concepts like optimal control and feedback mechanisms are fundamental for designing systems that maintain desired output levels despite external condition changes. These principles have been extensively explored in works such as [65], and further developed through modern control theories addressing non-linearities and uncertainties in system dynamics [68]. The integration of filtering and control methodologies has led to the development of Adaptive Kalman Filtering [40], which adjusts its parameters based on observed errors, enhancing performance in varying conditions. Adaptive Kalman Filtering has seen applications in diverse areas, including robotics [63], automotive systems [28], and finance [20], illustrating its versatility and robustness in handling dynamic, uncertain environments.

### 3 Preliminaries and model

We now describe the framework used for modeling trader behavior in response to the evolution of an external price process, that is hidden from the market maker, and the prices set by the market maker. We also state the objective that the market maker seeks to optimize, and provide the motivation behind it. The model and the objective are based on the canonical Glosten-Milgrom model [25] studied extensively in market microstructure literature. We assume that the market maker has access to an inventory of the asset and numeraire. The price of the asset is expressed in terms of the numeraire. We assume that time is discrete and indexed by  $t$ .

**External price process.** The external price process  $p_{ext}^t$  of a risky asset is assumed to follow a discrete time random walk, where the distribution of a price jump at any  $t$  is parametrized by  $\sigma$ . That is, we have

$$p_{ext}^{t+1} = p_{ext}^t + \Delta p_{ext}^t \tag{1}$$

where  $\Delta p_{ext}^t$  is i.i.d.  $\sim \mathcal{D}_{\sigma, p_{ext}^t}$ . Intuitively, the parameter  $\sigma$  can be thought of as a measure of volatility of the external price. Further, the random process  $p_{ext}^t$  can either represent the price of the asset in a larger and much more liquid exchange, or some underlying “true” value of the asset. In both cases, we assume that it is hidden from the market maker. To put it in DeFi terms, the market maker does not have an access to any “price oracle” that can tell it information about this external price.

**Market Maker.** The market maker consists of a pool containing an asset and a numeraire. It then publishes a demand curve  $g_t(p)$  [42] before each trade. All traders have full access to the demand curve at any time. The demand curve specifies how the price of the asset changes based on the inventory available. In particular,  $g(p_0^t)$  specifies the amount of asset when the initial operating point of the demand curve is  $p_0^t$ . That is, the price of an infinitesimal amount  $dx$  of asset is  $p_0^t dx$ . The amount of numeraire in the inventory at the same operating point is  $-\int_0^{p_0^t} pdg(p)$ . For an incentive compatible market maker, we constrain  $g(\cdot)$  to be non-increasing (Proposition 2.1 in [42]). Equivalently, the market maker may be represented by the canonical *bonding curve*  $\phi_t(x, y)$  [66] which is a function (of reserves  $x, y$  of the asset, numeraire respectively) that stays constant over any single trade.

**Trader behavior.** We assume that a trader appears at every time step  $t$ , and sees a noisy version of the external price in each time step, denoted by  $p_{traded}^t$ , with the noise distribution being parametrized by  $\eta$ .

$$p_{traded}^t = p_{ext}^t + \Delta p_{traded}^t \quad (2)$$

where  $\Delta p_{traded}^t$  is i.i.d. noise  $\sim \mathcal{D}_{\eta, p_{ext}^t}$  sampled at each time step. Intuitively, the parameter  $\eta$  can be construed as measuring the level of “toxicity” or informed nature of the traders.

**Trade actions.** As mentioned before, the market maker publishes a demand curve  $g_t(p)$  at each time step. The trader performs a trade that brings the operating point of the demand curve from  $p_0^t$  to  $p_{traded}^t$ . Thus, the trader behaves as if it is performing arbitrage between a market with price  $p_{traded}^t$  and the demand curve published. For instance, if the operating point  $p_0^t$  of the market maker is less than the trader price  $p_{traded}^t$ , then a trader would buy asset from the market maker to sell on the external market until the operating point shifts to  $p_{traded}^t$ .

**Objective.** Our objective is to design an algorithm to set ask and bid prices for the market maker, such that the expected loss with respect to the external market is minimized and the market maker stays competitive. Taking inspiration from [25], but extending to the modified trader behaviour, we get the following condition

$$p^t = E[p_{ext}^t | \mathcal{H}_{t-1}, p_{traded}^t] \quad \forall p_{traded}^t \quad (3)$$

where  $\mathcal{H}_{t-1} = \langle (p_{traded}^\tau, g_\tau(\cdot)) \rangle_{\tau=0}^{t-1}$  is the history of trades and demand curves until time  $t-1$ , and  $p^t$  is the net price of the trade (the ratio of change in the reserves of the numeraire with the change in the reserves of the asset).

**Interpreting the objective.** Setting the prices as per the above objective makes the expected loss of the market makers to traders vanish, since

$$E[(p_{ext}^t - p^t) | \mathcal{H}_{t-1}] = 0 \quad (4)$$

Note that the expression above quantifies the expected loss for per unit trade of the asset.

The market maker can obtain a strictly positive profit by choosing a curve with slightly higher prices (than the ones obtained in (3)) on the ask side or slightly lower prices on the bid side (in other words, by choosing a CFMM with a higher curvature [4]). However, doing this would make it less competitive, since any other market maker with slightly greater bid or a slightly lesser ask would offer a better price and take away the trade volume. Although

we do not explicitly model other market makers, their presence is implicit in setting prices according to (3). This equation represents the ideal conditions for capital efficiency, where both the trader gets the best price possible while the market maker avoids a loss.

Also, note that the market maker incurs a loss in *every trade* made by an perfectly informed trader. Thus, to make the expected loss vanish, it should learn to set prices so that the loss to more informed traders is balanced by the profit obtained from less informed traders. The equation (3) can also be interpreted as striking this balance.

#### 4 Differential Equation for the optimal curve

As mentioned in Section 3, we make use of the general *demand curve* formulation for market makers. Assume that the amount of asset in reserves at any price  $p_0^t$  is  $g_t(p_0^t)$ , while the amount of numeraire is  $-\int_0^{p_0^t} pdg_t(p)$ , where the function  $g_t$  needs to be non-increasing for the AMM to be incentive compatible [42]. Here,  $p_0^t$  is the initial operating point of the AMM at the beginning of the time slot  $t$ . Let  $f_t(p)$  be the belief (probability distribution function) of the AMM over the price  $p$  of the asset at time  $t$ . Let  $f_\eta(p)$  be the distribution of noise through which the price is observed by the traders. Let  $p_{traded}^t$  be the price that the trader observes. Then, the trader would make a trade with the AMM such that the marginal price of the asset just after the trade is  $p_{traded}^t$ . We now solve the optimal market conditions given by (3).

In this case, the effective price of a trade at time  $t$  is given by

$$p^t = \frac{-\int_{p_0^t}^{p_{traded}^t} pdg_t(p)}{g_t(p_0^t) - g_t(p_{traded}^t)}, \quad (5)$$

where  $p_0^t$  is the operating point of the AMM just before the trade. Further, by definition of conditional expectation of the external market price, we also have

$$E[p_{ext}^t | \mathcal{H}_{t-1}, p_{traded}^t] = \frac{\int_0^\infty pf_\eta(p_{traded}^t - p)f_t(p)dp}{\int_0^\infty f_\eta(p_{traded}^t - p)f_t(p)dp}. \quad (6)$$

since the noise at time  $t$  is independent of the external price at time  $t$ . Substituting (6) and (5) in (3) gives us

$$\frac{-\int_{p_0^t}^{p_{traded}^t} pdg_t(p)}{g_t(p_0^t) - g_t(p_{traded}^t)} = \frac{\int_0^\infty pf_\eta(p_{traded}^t - p)f_t(p)dp}{\int_0^\infty f_\eta(p_{traded}^t - p)f_t(p)dp} \quad \forall p_{traded}^t \quad (7)$$

Rearranging gives us

$$g_t(p_{traded}^t) = g_t(p_0^t) + \frac{\int_{p_0^t}^{p_{traded}^t} pg_t'(p)dp}{\beta_t(p_{traded}^t)} \quad \forall p_{traded}^t \quad (8)$$

where  $\beta_t(p_{traded}^t) = \frac{\int_0^\infty pf_\eta(p_{traded}^t - p)f_t(p)dp}{\int_0^\infty f_\eta(p_{traded}^t - p)f_t(p)dp}$ . On the other hand, if we let  $p_{traded}^t \rightarrow p_0^t$  in (7), we get

$$p_0^t = \beta_t(p_0^t) \quad (9)$$

Thus, using (8) and (9), the demand curve of the optimal market maker can be given by the following theorem.

► **Theorem 1.** *Let the amounts of asset and numeraire in the reserves of a market maker be  $x_0^t, y_0^t$ . Then, the optimal demand curve  $g_t(p)$  for a market maker obeys the following differential equation*

$$(\beta_t(p) - p)g_t'(p) + \beta_t'(p)g_t(p) - \beta_t'(p)x_0^t = 0 \quad (10)$$

with the constraints  $\lim_{\delta \rightarrow 0^-} g(p_0^t + \delta) \geq x_0^t$  and  $\lim_{\delta \rightarrow 0^-} -\int_{p_0^t}^{p_0^t + \delta} pdg(p) \geq y_0^t$ . This equation can be solved separately for  $p > p_0^t$  and  $p < p_0^t$ . Furthermore, the initial operating point  $p_0^t$  is a solution to the fixed point equation

$$p = \beta_t(p). \quad (11)$$

The differential equation (10) is obtained by simply applying the Leibniz rule of the derivative of a definite integral on (8). Note that we can rewrite (8) as

$$\int_{p_0^t}^{p_{trad}^t} \left(1 - \frac{p}{\beta_t(p_{trad}^t)}\right) dg_t(p) = 0 \quad (12)$$

We see that a discontinuity at  $p_0^t$  implies that  $dg_t(p)$  is not a differential but is a negative real number. Since  $p = \beta_t(p)$  at  $p_0^t$ , we have the term inside the integral  $\rightarrow 0$  as  $p_{trad}^t \rightarrow p_0^t$ . This implies that a discontinuity in  $g_t(p)$  is allowed at  $p_0^t$ . Further, since the (12) holds for  $p_{trad}^t > p_0^t$  and for  $p_{trad}^t < p_0^t$ , the differential equation can be solved separately for  $p > p_0^t$  and  $p < p_0^t$ .

Finally, assuming that the AMM finds solutions to (10) and (11), the trader is free to move the market to  $p_{trad}^t$ . The market maker then simply uses Bayes' rule to update its beliefs over the external prices as

$$f_{t+1}(p) = \frac{f_\eta(p_{trad}^t - p)f_t(p)}{\int_0^\infty f_\eta(p_{trad}^t - p)f_t(p)dp} \quad (13)$$

The updated belief  $f_{t+1}(p)$  is now used to compute  $\beta_{t+1}(p)$ , thus completing the market making algorithm. Equations (10), (11) and (13) describe the complete dynamics of an optimally efficient market maker.

## 5 Optimal Solutions for special cases

In this section, we present solutions to the differential equation derived in Theorem 1 for some special cases: Gaussian and Lognormal price jumps and trader noises. That is, we assume that both  $\Delta p_{ext}^t$  and  $\Delta p_{trad}^t$  follow a Gaussian or Lognormal distribution. Under these assumptions, we get the following key results.

### 5.1 Kalman Filter algorithm for known market parameters

► **Theorem 2.** *If  $\Delta p_{ext}^t \sim \mathcal{N}(0, \sigma^2)$  and  $\Delta p_{trad}^t \sim \mathcal{N}(0, \eta^2)$  where  $\sigma, \eta$  are known to the market maker, then the fixed point equation  $p_0^t = \beta_t(p_0^t)$  (11) has a unique solution given by the Kalman filter [32] estimate of  $p_{ext}^t$ , that is, we have*

$$p_0^t = E[p_{ext}^t | \mathcal{H}_{t-1}]. \quad (14)$$

Further, the differential equation (10) has a family of solutions, given by

$$g_t(p) = \begin{cases} x_0^t + \frac{y_0^t}{p_0^t} & \text{if } p \leq p_0^t \\ \max(0, \tilde{x}_0^t - C_t(p - p_0^t)^{\frac{K_t}{1-K_t}}) & \text{if } p > p_0^t \end{cases} \quad (15)$$

where  $K_t$  is the Kalman gain, and  $C_t, \tilde{x}_0^t$  are non-negative constants, such that  $\tilde{x}_0^t \leq x_0^t$ .

■ **Algorithm 1** A Kalman filter based algorithm to adapt the AMM curve.

**Require:** Known  $\eta, \sigma$ , Reserves  $x_0, y_0$

---

```

1:  $t \leftarrow 0$ 
2:  $T \leftarrow$  Number of total time slots
3: Initial price estimate  $p_{ext}^{0|0} \leftarrow p_{ext}^0$ 
4:  $P_{0|0} = 0$ 
5: while  $t \leq T$  do
6:    $\theta_t \leftarrow \frac{p_{ext}^{t-1|t-1} x_{t-1}}{p_{ext}^{t-1|t-1} x_{t-1} + y_{t-1}}$ 
7:   Publish bonding curve  $x^{\theta_t} y^{1-\theta_t} = x_{t-1}^{\theta_t} y_{t-1}^{1-\theta_t}$ 
8:   Observe trader action  $p_{trad}^t$ 
9:    $K_t \leftarrow \frac{P_{t-1|t-1} + \sigma^2}{P_{t-1|t-1} + \sigma^2 + \eta^2}$  ▷ Update Kalman gain
10:   $p_{ext}^{t|t} \leftarrow (1 - K_t)p_{ext}^{t-1|t-1} + K_t p_{trad}^t$  ▷ Update Kalman estimate of the external price
11:   $P_{t|t} = (1 - K_t)(P_{t-1|t-1} + \sigma^2)$  ▷ Update Kalman uncertainty
12: end while

```

---

The above solution corresponds to a CFMM with the initial slope of the bonding curve given by the Kalman estimate of  $p_{ext}^t$  by treating  $p_{trad}^t$  as noisy observations. Thus, we only need to calculate a single quantity  $E[p_{ext}^t | \mathcal{H}_t]$  (see Algorithm 1), and set the other parameters of the demand curve according to (15). We get a similar solution if we assume that the external price follows geometric Brownian motion.

► **Theorem 3.** *If  $\log \frac{p_{ext}^t}{p_{ext}^{t-1}} \sim \mathcal{N}(0, \sigma^2)$  and  $\log \frac{p_{trad}^t}{p_{ext}^t} \sim \mathcal{N}(0, \eta^2)$  where  $\sigma, \eta$  are known to the market maker, then the fixed point equation  $p_0^t = \beta_t(p_0^t)$  (11) has a unique solution which is a function of the Kalman filter [32] estimate of  $\log p_{ext}^t$ , that is, we have*

$$p_0^t = \exp \left( E[\log p_{ext}^t | \mathcal{H}_{t-1}] + \frac{P_{t|t}}{2(1 - K_t)} \right), \quad (16)$$

where  $K_t$  is the Kalman gain, and  $P_{t|t}$  is the variance of the Kalman estimate of  $\log p_{ext}^t$ . Further, the differential equation (10) has a solution given by

$$g_t(p) = \begin{cases} x_0^t + \frac{y_0^t}{p_0^t} & \text{if } p \leq p_0^t \\ \max(0, \tilde{x}_0^t - C_t(p^{1-K_t} - \kappa_t)^{\frac{K_t}{1-K_t}} p^{K_t}) & \text{if } p > p_0^t \end{cases} \quad (17)$$

where  $C_t, \kappa_t, \tilde{x}_0^t$  are non-negative constants such that  $\tilde{x}_0^t \leq x_0^t$ .

**Family of optimal demand curves.** We note that both Theorem 2 and Theorem 3 recommend a *family* of demand curves that satisfy the Glosten-Milgrom condition (3). The simplest curve in this family is the one where  $\tilde{x}_0^t = C_t = 0$ . This gives the simple demand curve which is constant except for a discontinuity at  $p_0^t$ . This corresponds to a constant sum market maker with bonding curve  $y + p_0^t x = k$  with an adaptive slope given by the Kalman estimate of the external price conditioned on trader behavior.

**An apparent contradiction, and a resolution.** A feature of the optimal demand curves derived is the fact that they do not make it possible for a trader to express any price  $p_{trad}^t$  between 0 and  $\infty$ . For instance, if we take the constant sum instance of the family of curves, it only allows the trader to express if the price of the trade is greater than or

## 25:10 Adaptive Curves for Optimally Efficient Market Making

less than  $p_0^t$ , but not the exact value of  $p_{trad}^t$ , which contradicts our assumption that the market maker can observe all of the history of trader prices  $\langle p_{trad}^\tau \rangle_{\tau=1}^t$ . We can resolve this contradiction by approximating the optimal demand curve as a sum of two demand curves  $g_t(p) = g_t^{opt}(p) + g_t^{exp}(p)$ , where the former curve is the *optimal* solution with most of the liquidity, and the latter curve with low liquidity to improve its price *expressiveness*. For instance, we can have  $g_t(p) = (1 - \epsilon)g_t^{CSMM}(p) + \epsilon g_t^{CPMM}(p)$  where we have

$$g_t^{CSMM}(p) = \begin{cases} x_0^t + \frac{y_0^t}{p_0^t} & \text{if } p \leq p_0^t \\ 0 & \text{if } p > p_0^t \end{cases} \quad (18)$$

$$g_t^{CPMM}(p) = 1/\sqrt{p} \quad (19)$$

with  $\epsilon \ll 1$ . This combines the demand curves of the constant sum market maker (the optimal solution) and a constant product market maker (the expressive solution) as an approximation.

**Other practical approximations.** Another way we can parametrize the curve of the market maker is to choose it from a family of curves such that the initial marginal price matches  $p_0^t$  as prescribed by Theorem 2 and Theorem 3, and processing a trade on any curve in that family is computationally simple. To that end, we can use the Constant Mean Market Makers [22, 38] with its weighting factor as our variable parameter. Note that the CMMMs performs trades along the curve  $x^\theta y^{1-\theta} = k$  where  $k$  is a constant, and  $x, y$  are the quantities of the asset and the numeraire in the market maker [22]. This ensures that no trade can exhaust either the asset or the numeraire from the AMM reserves. We know that the marginal price of the asset at any state of the reserves is given by  $p = \frac{\theta y}{(1-\theta)x}$ . Therefore, in our case, we set the value of parameter as  $\theta_t = \frac{p_0^t x}{p_0^t x + y}$ . This ensures that the starting price of any trade is  $p_0^t$ , and the market maker can only get a better price than that for a large trade.

## 5.2 Adaptive Kalman Filter algorithm for unknown market parameters

**Need for more adaptivity.** A major assumption while solving for the optimal demand curve in the Gaussian/Lognormal model was that the market parameters  $\sigma, \eta$  that control the variances of the price jump and noise were known to the AMM. These can indeed be obtained by analysing historical trading data in any market, and can be assumed to change slowly on the timescale that prices undergo changes. However, this assumption might not always hold for assets or tokens that are less well known or have no historical data. To deal with this case, we propose a modification to Algorithm 1. This ensures that we simultaneously estimate the parameters  $\eta, \sigma$  and hence help estimate the hidden external market price. To that end, we observe that we can write the likelihood function of all random variables and parameters of our model at time  $t$  as follows

$$L_t = L(\langle p_{trad}^\tau, p_{ext}^\tau \rangle_{\tau=1}^t, \eta, \sigma) = \prod_{\tau=1}^t \frac{1}{2\pi\sqrt{\eta\sigma}} \exp\left(-\frac{(p_{ext}^\tau - p_{ext}^{\tau-1})^2}{2\sigma^2} - \frac{(p_{trad}^\tau - p_{ext}^\tau)^2}{2\eta^2}\right), \quad (20)$$

which gives us the conditional log-likelihood given the trader actions as

$$E[\log L_t | \langle p_{trad}^\tau \rangle_{\tau=1}^t] = -\frac{t \log \sigma}{2} - \frac{t \log \eta}{2} - \frac{\sum_{\tau=1}^t A_\tau}{2\sigma^2} - \frac{\sum_{\tau=1}^t B_\tau}{2\eta^2}, \quad (21)$$



■ **Algorithm 2** Adaptive Kalman filter : EM algorithm estimating unknown parameters at time step  $t$ .

---

**Require:** Trader data  $\langle p_{trad}^\tau \rangle_{\tau=1}^t$ , Error Tolerance  $\epsilon$

- 1:  $\sigma \leftarrow$  Initial guess  $\sigma_0$
- 2:  $\eta \leftarrow$  Initial guess  $\eta_0$
- 3:  $i \leftarrow 0$
- 4: LogLikelihood =  $-\infty$
- 5: **while** LogLikelihood  $< -\epsilon$  **do**
- 6:   E step: Set  $A_\tau, B_\tau$  as per Equations (22) and (23), assuming guesses  $\sigma, \eta$
- 7:   M step: Set  $\sigma = \sqrt{2 \sum_{\tau=1}^t A_\tau / t}, \eta = \sqrt{2 \sum_{\tau=1}^t B_\tau / t}$
- 8:   LogLikelihood  $\leftarrow$  Equation (21)
- 9:    $i \leftarrow i + 1$
- 10: **end while**

---

where  $A_\tau, B_\tau$  are given by

$$A_\tau = E[(p_{ext}^\tau)^2 | \langle p_{trad}^\tau \rangle_{\tau=1}^t] + E[(p_{ext}^{\tau-1})^2 | \langle p_{trad}^\tau \rangle_{\tau=1}^t] + 2E[p_{ext}^\tau p_{ext}^{\tau-1} | \langle p_{trad}^\tau \rangle_{\tau=1}^t], \quad (22)$$

$$B_\tau = (p_{trad}^\tau)^2 + E[(p_{ext}^\tau)^2 | \langle p_{trad}^\tau \rangle_{\tau=1}^t] + 2p_{trad}^\tau E[p_{ext}^\tau | \langle p_{trad}^\tau \rangle_{\tau=1}^t]. \quad (23)$$

**Estimating the unknowns.** We now estimate the market parameters  $\sigma, \eta$  using the EM algorithm [19]. This can be done by first setting the terms  $A_\tau, B_\tau$  using (22) and (23) with the expectations on the RHS calculated via forward and backward runs of the Kalman filter assuming an initial guess estimate of  $\sigma, \eta$ . The forward runs of the algorithm involve computing  $E[p_{ext}^\tau | \langle p_{trad}^\tau \rangle_{i=1}^\tau]$  for all  $\tau = 1, \dots, t$ . These are estimates of the external price given only the data in the past. This can be done using the Kalman filter updates given in Algorithm 1. Next, we use the Rauch-Tung-Striebel smoother [60], an essentially backward run of the Kalman filter algorithm given all the statistics obtained from the forward run. This computes statistics such as  $E[p_{ext}^\tau | \langle p_{trad}^\tau \rangle_{i=1}^t]$ , that is, the estimate of the external price in the past given *all* of the observations till the present time slot. This evaluates all of the terms in  $A_\tau$  and  $B_\tau$ , and completes the E-step of the EM algorithm.

After that, we use (21) to find values of  $\sigma, \eta$  that maximize the conditional log-likelihood function. This involves setting the gradient of the expected log likelihood function to zero

$$\nabla_\sigma E[\log L_t | \langle p_{trad}^\tau \rangle_{\tau=1}^t] = 0, \quad \nabla_\eta E[\log L_t | \langle p_{trad}^\tau \rangle_{\tau=1}^t] = 0. \quad (24)$$

While doing this, we ignore the dependence of  $A_\tau, B_\tau$  on  $\eta, \sigma$  to obtain

$$\sigma^* = \sqrt{2 \frac{\sum_{\tau=1}^t A_\tau}{t}}, \quad \eta^* = \sqrt{2 \frac{\sum_{\tau=1}^t B_\tau}{t}} \quad (25)$$

which completely specifies the M-step of the EM algorithm. This has been summarized in Algorithm 2.

**Managing computation, and adapting to a non-stationary market.** While Algorithm 2 added on top of our AMM helps us estimate the unknown market parameters, its computational complexity keeps growing linearly with each additional trade. This is because every trade adds another term in the series that computes the log-likelihood function, hence increasing the number of iterations in both the forward and backward runs of the Kalman

filter. Additionally, this algorithm also assumes there is no change in the market parameters  $\eta, \sigma$  with time, that is, the market conditions are stationary. We can make the algorithm less computationally heavy and adapt to non-stationarities by truncating the trading data to a recent history of  $p_{traded}^\tau$ . This approach keeps track of all price estimates of the past, since they get refined with every backward run of the Adaptive Kalman algorithm, but only runs the algorithm on the recent history of price estimates and trading data.

### 5.3 Adversarial robustness

A prominent danger to the proper functioning of market making protocols is the presence of adversarial trader behavior. The model described so far and the optimal solutions presented remain valid only when traders interacting with the market are rational with respect to the external market. However, there is always the possibility that an AMM is being manipulated for profits extracted from other protocols (e.g. lenders[6], derivative markets, etc.) relying on the AMM as a price oracle [2]. Although any protocol using an AMM as a price oracle usually takes necessary precautions, such as ensuring a diverse portfolio of price signals, using outlier-robust statistics such as medians rather than means, etc. we show that our market making algorithms can be made robust to such market adversaries, when the proportion of such adversarial is less than half of all trading interactions.

**Adversary model.** The adversarial behavior we seek to guard against is the manipulation of  $p_{traded}^t$  that the AMM observes and not the external price  $p_{ext}^t$ . We assume that the external price is inferred from a deep market that is not easily manipulated. We further assume that a proportion  $\alpha$  of the trader population is adversarial and the rest behave as per the rational model in Section 3. However, since the AMM does not know which trades are being manipulated by the adversary, we assign a sequence of learnable weights  $w_\tau$  for all trade observations in the past  $\langle p_{traded}^\tau \rangle_{\tau=1}^t$ . To successfully manipulate the price, the adversary needs to push  $p_{traded}^t$  in a specific direction so as to induce a large discrepancy in the marginal prices of the AMM curve and  $p_{ext}^t$ .

**Robust adaptive curve algorithm.** A simple modification of the EM algorithm enables us to distinguish adversarial trades from honest trades [13, 64]. We first rewrite the log-likelihood function of the AMM as

$$E[\log L_t | \langle p_{traded}^\tau \rangle_{\tau=1}^t] = -\frac{t \log \sigma}{2} - \frac{t \log \eta}{2} + \sum_{\tau=1}^t \left( \frac{\log w_\tau}{4} - \frac{A_\tau}{2\sigma^2} - \frac{w_\tau B_\tau}{2\eta^2} \right), \quad (26)$$

which basically assumes that each datapoint has a different variance in noise  $\eta^2/w_\tau$ . This implies that data with low weights have a higher variance, and are hence the adversarial outliers. We start our algorithm with an equal weight given to all datapoints, and then estimate  $A_\tau, B_\tau$  assuming those weights and running the forward and backward runs of the Kalman Filtering algorithm. After that, we set new weights by getting the critical points for the log-likelihood maximization using

$$\nabla_{w_\tau} E[\log L_t | \langle p_{traded}^\tau \rangle_{\tau=1}^t] = 0 \implies w_\tau^* = \frac{\eta^2}{2B_\tau}. \quad (27)$$

This completes the adversarially robust version of the Kalman filtering algorithm. We empirically demonstrate the effectiveness of the approach for  $\alpha < 0.5$  compared to static curves, and a naive Kalman filtering approach (Section 7).

## 6 Implications for AMMs with static curves

If we view the market maker as a price oracle, then one can compare the performance of different algorithms based on how the mean squared error of the AMM changes with the incoming trades. In particular, one can compare how quickly the error goes down in the adaptive protocol proposed in Section 5.2 and a static curve such as Uniswap. This comparison has already been done for liquid markets [46], where we observe an exponential decay of the error for an adaptive protocol compared to a linear decay for a static one.

**Error performance with trades.** For our case, let us assume that we are comparing how Algorithm 1 performs in contrast to a static curve, if we assume that it is deployed on a blockchain with  $T$  trades in a block. Then, we can prove that the error decreases with number of trades for Algorithm 1 and stays constant for a static curve.

► **Theorem 4.** *Let there be  $T$  trades in a single block of transactions, with the external price at the creation of the block being  $p_{ext}$ . We denote by  $p_{KF}^T$  and  $p_{SC}^T$  the marginal prices of the Algorithm 1 and a static curve at the end of the block. Then, we have*

$$E[(p_{ext} - p_{KF}^T)^2] = \frac{\eta^2 \sigma^2}{T\sigma^2 + \eta^2} \quad (28)$$

$$E[(p_{ext} - p_{SC}^T)^2] = \eta^2 \quad (29)$$

**Implied dynamics of static curves.** We see that static curves are worse oracles because they do not use the realistic dynamical model to get the best estimate of the external price. The question then arises if there is any dynamical model that a particular static curve *is* optimal for. Note that the differential equation (10) can be viewed as an equation in  $\beta_t(p)$  if the curve  $g_t(p)$  is given. We now use this observation to work out the *implied* dynamical model underlying commonly used static curves. More formally, given the demand curve of a static AMM  $g(p)$ , we can find the corresponding  $\beta_t(p)$  function by solving the following differential equation

$$\beta_t'(p)(g_t(p) - x_0^t) + \beta_t(p)g_t'(p) - pg_t'(p) = 0 \quad (30)$$

with the initial operating point  $p_0^t$  satisfying the constraint  $p = \beta_t(p)$ . We now solve this equation for some common CFMM curves to get the underlying implied price/trader dynamics.

**Constant Sum Market Makers.** The demand curve for a constant sum market maker is given by

$$g_t(p) = \begin{cases} x_0^t + y_0^t/p_0^t & \text{if } p \leq p_0^t \\ 0 & \text{if } p > p_0^t \end{cases} \quad (31)$$

where  $p_0^t = p_0$  stays constant with time. Note that substituting the demand curve in (30) gives a trivial equation  $\beta_t'(p) = 0$ . This, coupled with the condition on the initial operating point implies that  $\beta_t(p) = p_0$  for all  $t$ . Clearly, this implies a dynamical model where  $\Delta p_{ext}^t = \Delta p_{trad}^t = 0$ . In other words, a CSMM assumes that the external price stays constant with time and that the traders have a noiseless view of the price at all times.

## 25:14 Adaptive Curves for Optimally Efficient Market Making

**Constant Mean Market Makers.** The demand curve for Constant Mean Market Makers is given by

$$g_t(p) = \frac{c}{p^{1-\theta}} \left( \frac{\theta}{1-\theta} \right)^{1-\theta} \quad (32)$$

where the bonding curve for the constant mean market maker is of the form  $x^\theta y^{1-\theta} = \text{constant}$ , where  $x, y$  denote the reserves for the asset and numeraire respectively.

Substituting this in (30) and solving for  $\beta_t(p)$  gives us the following function

$$\beta_t(p) = \frac{1-\theta}{\theta} p^\theta (p_0^t)^{1-\theta} \frac{1 - (p_0^t/p)^\theta}{1 - (p_0^t/p)^{1-\theta}} \quad (33)$$

which also satisfies  $\beta_t(p_0^t) = p_0^t$ . In particular, if we look at constant product market makers, we have  $\theta = 1/2$ , giving us  $\beta_t(p) = \sqrt{p_0^t p}$ . For this  $\beta_t(\cdot)$  function, we have the following result.

► **Theorem 5.** *The following price and trader behavior dynamics yields  $\beta_t(p)$  which obeys (33). Equivalently, the following model has a constant mean market maker with weight parameter  $\theta$  as its optimally efficient solution,*

$$\log p_{ext}^t = \log p_{trad}^{t-1} + \epsilon_\sigma^t \quad (34)$$

$$\log p_{trad}^t = \log p_{ext}^t + \epsilon_\eta^t, \quad (35)$$

where  $\epsilon_\sigma, \epsilon_\eta$  are independent Gaussian random variables with zero mean and variances  $\sigma, \eta$  respectively, where the variances satisfy the following conditions.

$$\sigma \ll 1 \quad (36)$$

$$\eta = \sigma \left( \sqrt{1/\theta - 1} \right) \quad (37)$$

The above theorem sheds light on why static curves fail to prevent arbitrage loss as effectively – the implied dynamic model that they assume is mismatched with more realistic trader behavior. This mismatch manifests itself in the following ways, as indicated by (36) and (37).

**Low latency blockchain.** Firstly, static curves assume that the price jumps between consecutive trades have a very small variance. This is equivalent to assuming that the inter-block times on the underlying blockchain go to 0. This is because the standard deviation of price jumps  $\sigma$  between blocks depends on the price volatility  $\sigma'$  as  $\sigma = \sigma' \sqrt{\Delta t}$ , where  $\Delta t$  is the inter-block arrival time. This observation confirms the conclusion reached in [41] and the broader DeFi community [62], where the arbitrage loss (or LVR) is calculated for the special case of all of the trader population being arbitrageurs and concludes that this loss indeed approaches 0 as the inter-block time goes to 0.

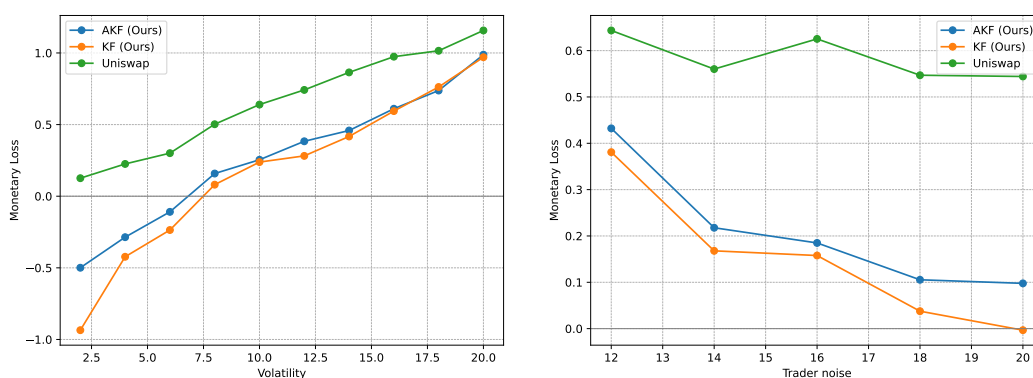
**Constraints on noise traders.** Secondly, the noise in the price that the traders see obeys a specific structure –  $\eta^2 = \sigma^2(1/\theta - 1)$ . For a constant product market maker, this specifically assumes that the variance of price jumps is exactly the same as the variance of the noise in trader beliefs about the price. This implies additional restrictions on price and trader behaviour that might not always be true in real markets. However, these constraints can potentially help decide how toxic and non-toxic trade flow is guided in “DEX-aggregator”

such as UniswapX [58], 1inch [1], CoWSwap [52], etc. to ensure that passive LPs (that invest in pools) get a fair price as defined by (3). The main threat that such aggregators pose to passive LPs is that much of the non-toxic (or uninformed, or noise) trades get satisfied internally without any trades flowing through the pools, while the surplus (usually toxic or informed) gets routed through the passive pools [8]. The equation (36) prescribes how much non-toxic flow should be “added in the mix” to ensure fairness to the passive pool LPs.

**Dominance of DEXes.** Thirdly, (34) highlights another key assumption – *the external market also reacts to the price on the AMM*. This is only true when the AMM has an amount of liquidity that is more than the external market, which is not true for most AMM pools or decentralized exchanges today [11]. This means that static curves are guaranteed to make a loss to arbitrageurs unless decentralized exchanges become the main sources of liquidity, and the inter-block arrival time on the underlying blockchain become negligible.

## 7 Empirical Results

In this section, we present the empirical performance of the algorithms discussed so far in this work<sup>1</sup>.



(a) Varying price volatility  $\sigma$ .

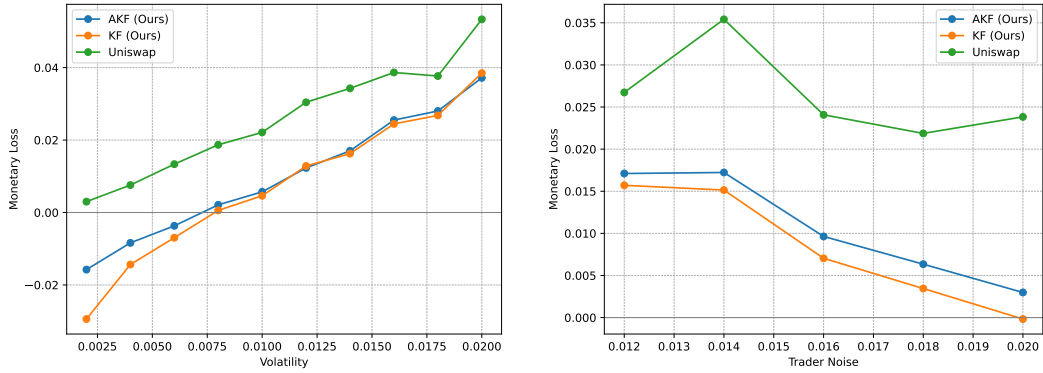
(b) Varying trader noise  $\eta$ .

**Figure 1** Percentage monetary loss per trade of our market making algorithms (Kalman Filtering and Adaptive Kalman Filtering) is much less than a static Uniswap curve for a Gaussian price jump and trader noise models.

**Comparison with static curves.** We simulate the model described in Section 3 and compare the performance of the algorithms we proposed (Figure [1]). We see that adapting the AMM curve according to algorithms 1 and 2 give a much lower monetary loss per trade than a static constant product curve that is used in Uniswap. Furthermore, the Adaptive Kalman Filter algorithm estimates the unknown market parameters correctly, leading to it achieving close performance with the optimal Kalman Filtering algorithm.

The same observation holds for prices that follow a geometric Brownian motion, as seen in Figure [2].

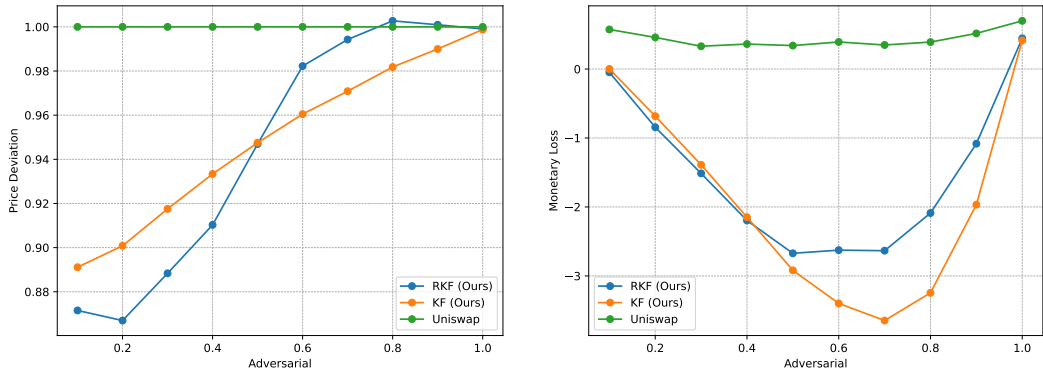
<sup>1</sup> Code for running all experiments has been shared [here](#)



(a) Varying price volatility  $\sigma$ .

(b) Varying trader noise  $\eta$ .

■ **Figure 2** Percentage monetary loss per trade of our making algorithms (Kalman Filtering and Adaptive Kalman Filtering) is much less than a static Uniswap curve for a Lognormal price jump and trader noise models.



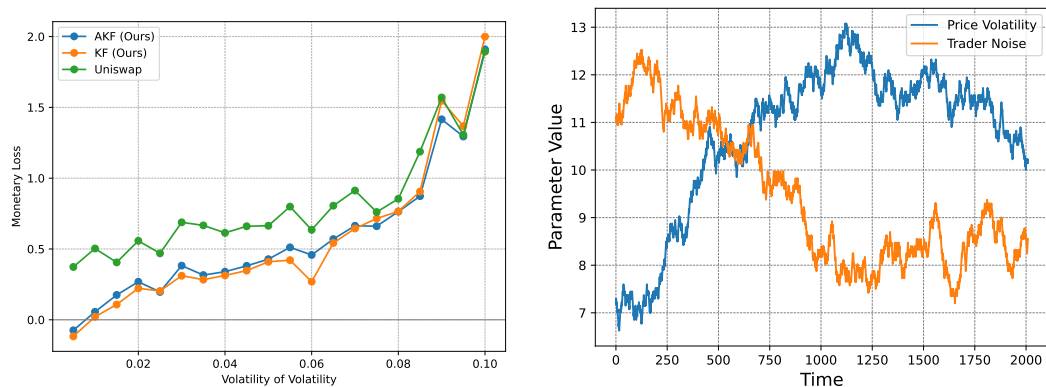
(a) Root Mean Squared Deviation of external price from the price recommended by the AMM.

(b) Percentage monetary loss compared across AMMs.

■ **Figure 3** If the AMM is treated as an oracle, then we can use the Robust Kalman Filtering to get a more accurate reading of the hidden external price.

**Robustness to adversarial traders.** In presence of adversarial traders, the Robust Kalman Filtering algorithm is used to change the curve of the AMM gives us a more accurate reading of the hidden external price in the presence of less than 50% of the population of traders being adversarial (Figure [3]). The adversarial traders, in this case, are assumed to perform large buy trades (with price belief  $p_{trad}^t$  about 5-7 standard deviations beyond normal trading size) to keep the AMM price above the external market. We also note that the monetary loss of the AMMs against the adversary stays in the profitable region for the adaptive curves, while the static curve suffers a loss to arbitrageurs even if the adversary is making trades that are irrational.

**Robustness to non-stationary markets.** Figure [4] tests the truncated version of the adaptive Kalman algorithm for changing market conditions. This algorithm is compared with the Kalman Filter algorithm, that knows the underlying market parameters exactly at every time step, and the static Uniswap curve. The monetary loss incurred by the AMMs is measured against changing variability of market conditions, measured by the volatility of



(a) Varying Volatility of Volatility.

(b) Changing market conditions  $\eta, \sigma$  for a single sample path.

■ **Figure 4** Percentage monetary loss per trade of our market making algorithms (Kalman Filtering and Adaptive Kalman Filtering) is much less than a static Uniswap curve for continuously changing market conditions.

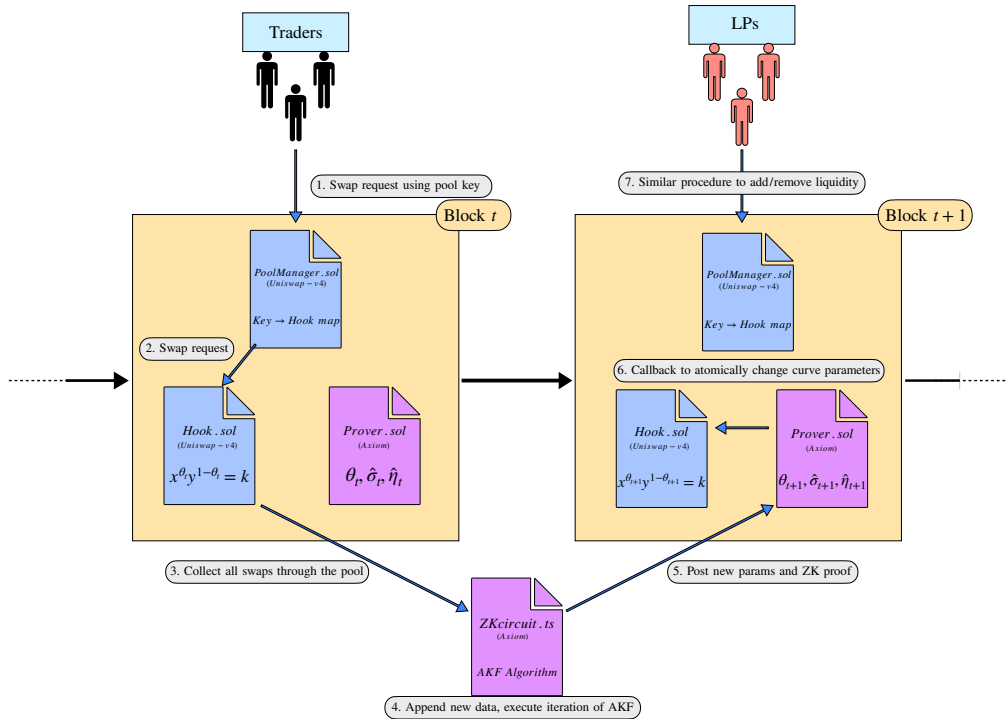
price jump variance and trader noise (termed as volatility of volatility  $\sigma_{\eta, \sigma}$ ). This metric measures the standard deviation of changing  $\eta, \sigma$  after every trade, with a sample path of these changing variables shown in Figure [4b] for  $\sigma_{\eta, \sigma} = 0.04$ . Recall that  $\eta, \sigma$  themselves govern how the external price is perceived by the traders and how it changes after every trade respectively. We see that the Adaptive Kalman Filter is able to perform almost as well as the Kalman Filter when the market changes are slow enough. However, the performance advantage over static curves vanishes as the changing market conditions become more erratic. This happens because the timescale over which market parameters suffer large changes becomes comparable to the timescale of the recent history considered by the truncated adaptive Kalman filter. This observation offers guidance to AMM designers on choosing the timescale over which adaptivity can offer an advantage over static markets.

## 8 On-chain System Implementation

Many prior works [14, 46] seek to implement adaptive market makers on a blockchain, where the adapting is done using machine learning algorithms that must be necessarily performed off-chain because of their computational load [53]. To that end, a group of protocol validators (separate from the validators of the underlying blockchain) are assumed, who run the bulk of the computation off-chain and post their results (such as satisfied orders, their prices, etc.) on-chain. However, recent developments in Layer 2 or rollup [33, 49] infrastructure, machine learning co-processors with zero-knowledge guarantees [51, 10], has given rise to several platforms that can be utilized directly to implement the adaptive market makers (or machine learning algorithms in general) we derived in the previous sections, without any need of additional validators. We draw upon these innovations for the blockchain implementation, and divide the approach into two parts. The overall design has been shown in Figure [5].

**Hook Contract.** The first part of such an implementation is the liquidity pool contract, which allows the canonical interactions with LPs and traders given a specific demand/bonding curve. In the Ethereum DeFi ecosystem, a recent proposal [57] by the Uniswap protocol (called Uniswap-v4) presents a highly customizable platform for adaptive market making. The main innovation is the introduction of a “hook” smart contract [54]. While prior versions

[55, 56] of the protocol presented market making in a canonical manner where the exact bonding curve followed during trade execution was static and only gave freedoms to LPs in terms of the distribution of liquidity along that fixed curve, the new platform allows the LP to specify changes to the bonding curve just before/after every single trade via the hook contract. We use this functionality to execute trades and add/remove liquidity according to the curve  $x^{\theta_t}y^{1-\theta_t} = constant$ . When traders/LPs put in canonical swap/liquidity addition transactions, they are first routed through the PoolManager contract. This maps a pool to a specific Hook contract that specifies all modifications to the curve before/after trade execution. All interactions with this contract are collected as input data by the second part of the system.



■ **Figure 5** System design for an on-chain implementation of our algorithms.

**Off-Chain Co-processor.** The second part runs the algorithm used to change the demand/-bonding curve given the history of trader interactions. This is done off-chain due to the high computational load of running the EM algorithm as part of Adaptive Kalman Filtering. For our implementation, we chose Axiom [51] as a platform to run this off-chain computation. The main part of this implementation is a typescript file `ZKcircuit.ts` containing the details of Algorithm 2 implemented as an algebraic circuit, so that a zero-knowledge proof can be generated corresponding to the computation [50]. This file also verifiably collects data from the previous block, and runs the algorithm to come up with new estimates for the market parameters  $\sigma, \eta$  and hence the curve parameter  $\theta$ . The off-chain Axiom client [50] posts this in the next block and is verified by the on-chain `Prover` contract of Axiom. This also invokes a callback to the `Hook` where the changes to the curve parameters are finally implemented. We provide open access to all files used in our implementation <sup>2</sup>.

<sup>2</sup> Code for the proof-of-concept implementation has been shared [here](#)



## 9 Conclusion and Future Work

**Generalising to non-Gaussian and non-stationary behavior.** Traders usually do not perfectly conform to the distributional assumptions we use to derive optimal solutions in Section 5.1. Our approach can be potentially be extended to such situations by the use of Neural Kalman Filters [44], which claim to work for non-Gaussian/non-stationary state space models.

**Balancing toxic and non-toxic orderflow.** The blockchain-level conditions for the optimality of static curves, as outlined in Section 6, provide guidance on how toxic/non-toxic orderflow, if discriminated correctly [62], should be allowed to use passive liquidity and still give LPs a fair price. Developing DEX aggregators that aware of these conditions would help limit the dangers to passive LPs in DeFi.

**Extensions to other adaptive protocols.** In this work, we have derived a correspondence between a dynamical model for prices and its optimal market making curve. This principle can be extended for stable control of other DeFi protocols, such as lending, that currently use static curves.

---

### References

- 1 linch protocol. linch dex aggregator. <https://linch.io>. Accessed: 2023-05.
- 2 Guillermo Angeris, Akshay Agrawal, Alex Evans, Tarun Chitra, and Stephen Boyd. Constant function market makers: Multi-asset trades via convex optimization, 2021. [arXiv:2107.12484](https://arxiv.org/abs/2107.12484).
- 3 Guillermo Angeris and Tarun Chitra. Improved price oracles. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*. ACM, October 2020. doi:10.1145/3419614.3423251.
- 4 Guillermo Angeris, Alex Evans, and Tarun Chitra. When does the tail wag the dog? curvature and market making, 2020. [arXiv:2012.08040](https://arxiv.org/abs/2012.08040).
- 5 Jun Aoyagi. Liquidity provision by automated market makers, 2020. URL: <https://ssrn.com/abstract=3674178>.
- 6 Ayana T. Aspembitova and Michael A. Bentley. Oracles in decentralized finance: Attack costs, profits and mitigation measures. *Entropy*, 25(1), 2023. doi:10.3390/e25010060.
- 7 Marco Avellaneda and Sasha Stoikov. High frequency trading in a limit order book. *Quantitative Finance*, 8:217–224, April 2008. doi:10.1080/14697680701381228.
- 8 Ashwath Balakrishnan. Understanding uniswap’s new liquidity aggregator (tl;dr at the end). <https://members.delphidigital.io/feed/understanding-uniswaps-new-liquidity-aggregator-tldr-at-the-end>, 2023. [arXiv:2305.14604](https://arxiv.org/abs/2305.14604).
- 9 The Block. Block.co trade volume comparisons. <https://www.theblock.co/data/decentralized-finance/dex-non-custodial/uniswap-vs-coinbase-and-binance-trade-volume-7dma>. Accessed: 2023-09.
- 10 Brevis. Brevis website. <https://docs.brevis.network/>. Accessed: 2023-05.
- 11 centicio. Centralized exchange (cex) vs decentralized exchange (dex). which is the best crypto exchange? <https://medium.com/@centicio/centralized-exchange-cex-vs-decentralized-exchange-dex-which-is-the-best-crypto-exchange-148f48ea51c1>. Accessed: 2023-05.
- 12 Nicholas Chan and Christian Shelton. An electronic market-maker, January 2001.
- 13 Sitan Chen, Frederic Koehler, Ankur Moitra, and Morris Yau. Kalman filtering with adversarial corruptions. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 832–845, 2022.
- 14 Dev Churiwala and Bhaskar Krishnamachari. Qlamm: A q-learning agent for optimizing fees on automated market making protocols, 2022. [arXiv:2211.14977](https://arxiv.org/abs/2211.14977).

## 25:20 Adaptive Curves for Optimally Efficient Market Making

- 15 CrocSwap. Discrimination of toxic flow in uniswap v3. <https://crocswap.medium.com/discrimination-of-toxic-flow-in-uniswap-v3-part-1-fb5b6e01398b>. Accessed: 2023-09.
- 16 Diane Dai. Dodo integrates chainlink live on mainnet, kickstarts the on-chain liquidity revolution. <https://blog.dodoex.io/dodo-integrates-chainlink-live-on-mainnet-kickstarts-the-on-chain-liquidity-revolution-ee27e136e122>. Accessed: 2023-09.
- 17 Sanmay Das\*. A learning market-maker in the glisten–milgrom model. *Quantitative Finance*, 5(2):169–180, 2005.
- 18 Sanmay Das and Malik Magdon-Ismail. Adapting to a market shock: Optimal sequential market-making. *Advances in Neural Information Processing Systems*, 21, 2008.
- 19 A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977. URL: <http://www.jstor.org/stable/2984875>.
- 20 Graham Elliott. Forecasting with trending data. *Handbook of economic forecasting*, 1:555–604, 2006.
- 21 Shayan Eskandari, Mehdi Salehi, Wanyun Catherine Gu, and Jeremy Clark. SoK. In *Proceedings of the 3rd ACM Conference on Advances in Financial Technologies*. ACM, September 2021. doi:10.1145/3479722.3480994.
- 22 Alex Evans. Liquidity provider returns in geometric mean markets, 2020. arXiv:2006.08806.
- 23 Alex Evans, Guillermo Angeris, and Tarun Chitra. Optimal fees for geometric mean market makers, 2021. arXiv:2104.00446.
- 24 Rafael Frongillo, Maneesha Papireddygar, and Bo Waggoner. An axiomatic characterization of cfms and equivalence to prediction markets. *arXiv preprint*, 2023. arXiv:2302.00196.
- 25 Lawrence R. Glosten and Paul R. Milgrom. Bid, ask and transaction prices in a specialist market with heterogeneously informed traders. *Journal of Financial Economics*, 14(1):71–100, 1985. doi:10.1016/0304-405X(85)90044-3.
- 26 Mohak Goyal, Geoffrey Ramseyer, Ashish Goel, and David Mazières. Finding the right curve: Optimal design of constant function market makers, 2023. arXiv:2212.03340.
- 27 SANFORD J. GROSSMAN and MERTON H. MILLER. Liquidity and market structure. *The Journal of Finance*, 43(3):617–633, 1988. doi:10.1111/j.1540-6261.1988.tb04594.x.
- 28 F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund. Particle filters for positioning, navigation, and tracking. *IEEE Transactions on Signal Processing*, 50(2):425–437, 2002. doi:10.1109/78.978396.
- 29 Lioba Heimbach, Eric Schertenleib, and Roger Wattenhofer. Risks and returns of uniswap v3 liquidity providers. In *Proceedings of the 4th ACM Conference on Advances in Financial Technologies*. ACM, September 2022. doi:10.1145/3558535.3559772.
- 30 Thomas S. Y. Ho and Hans R. Stoll. The dynamics of dealer markets under competition. *The Journal of Finance*, 38(4):1053–1074, 1983. URL: <http://www.jstor.org/stable/2328011>.
- 31 Peter Johnson and Sai Nimmagadda. The relentless rise of stablecoins. <https://digify.com/a/#/f/p/ef09be008ee64ab68bda4f0a558302a2>. Accessed: 2023-09.
- 32 Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME—Journal of Basic Engineering*, 82(Series D):35–45, 1960.
- 33 Harry Kalodner, Steven Goldfeder, Xiaoqi Chen, S. Matthew Weinberg, and Edward W. Felten. Arbitrum: Scalable, private smart contracts. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1353–1370, Baltimore, MD, August 2018. USENIX Association. URL: <https://www.usenix.org/conference/usenixsecurity18/presentation/kalodner>.
- 34 Albert S. Kyle. Continuous auctions and insider trading. *Econometrica*, 53(6):1315–1335, 1985. URL: <http://www.jstor.org/stable/1913210>.
- 35 DeFi Llama. Uniswap-v3 tvl comparison for stable coins vs non-stablecoins. <https://defillama.com/protocol/uniswap-v3>. Accessed: 2023-09.
- 36 Stefan Loesch, Nate Hindman, Mark B Richardson, and Nicholas Welch. Impermanent loss in uniswap v3, 2021. arXiv:2111.09192.

- 37 Christos Makidris. Front running, bots, slippage, oracle pricing errors: Amms are great, but there are problems. <https://cointelegraph.com/magazine/trouble-with-crypto-automated-market-makers/>. Accessed: 2023-09.
- 38 F. Martinelli. Balancer whitepaper. <https://balancer.fi/whitepaper.pdf>. Accessed: 2023-05.
- 39 Conor McMenamin, Vanesa Daza, and Bruno Mazorra. Diamonds are forever, loss-versus-rebalancing is not, 2022. [arXiv:2210.10601](https://arxiv.org/abs/2210.10601).
- 40 Raman K. Mehra. On the identification of variances and adaptive kalman filtering. *IEEE Transactions on Automatic Control*, 15:175–184, 1970. URL: <https://api.semanticscholar.org/CorpusID:238574860>.
- 41 Jason Milionis, Ciamac C. Moallemi, and Tim Roughgarden. Automated market making and arbitrage profits in the presence of fees, 2023. [arXiv:2305.14604](https://arxiv.org/abs/2305.14604).
- 42 Jason Milionis, Ciamac C. Moallemi, and Tim Roughgarden. A myersonian framework for optimal liquidity provision in automated market makers, 2023. [arXiv:2303.00208](https://arxiv.org/abs/2303.00208).
- 43 Jason Milionis, Ciamac C. Moallemi, Tim Roughgarden, and Anthony Lee Zhang. Automated market making and loss-versus-rebalancing, 2022. [arXiv:2208.06046](https://arxiv.org/abs/2208.06046).
- 44 Beren Millidge, Alexander Tschantz, Anil Seth, and Christopher Buckley. Neural kalman filtering, 2021. [arXiv:2102.10021](https://arxiv.org/abs/2102.10021).
- 45 Vijay Mohan. Automated market makers and decentralized exchanges: a defi primer, December 2020. [doi:10.2139/ssrn.3722714](https://doi.org/10.2139/ssrn.3722714).
- 46 Viraj Nadkarni, Jiachen Hu, Ranvir Rana, Chi Jin, Sanjeev Kulkarni, and Pramod Viswanath. Zeroswap: Data-driven optimal market making in defi. *arXiv preprint*, 2023. [arXiv:2310.09413](https://arxiv.org/abs/2310.09413).
- 47 Viraj Nadkarni, Sanjeev Kulkarni, and Pramod Viswanath. Adaptive curves for optimally efficient market making. *arXiv preprint*, 2024. [arXiv:2406.13794](https://arxiv.org/abs/2406.13794).
- 48 Alex Nezhobin. Order flow toxicity on dexes. <https://ethresear.ch/t/order-flow-toxicity-on-dexes/13177>. Accessed: 2023-09.
- 49 Optimism. Optimism docs. <https://community.optimism.io/>. Accessed: 2023-09.
- 50 Axiom Protocol. Axiom client circuit. <https://docs.axiom.xyz/docs/axiom-developer-flow/axiom-client-circuit>. Accessed: 2023-05.
- 51 Axiom Protocol. Axiom website. <https://docs.axiom.xyz/>. Accessed: 2023-05.
- 52 CoW protocol. Cowswap docs. <https://docs.cow.fi/overview/coincidence-of-wants>. Accessed: 2023-09.
- 53 Gate protocol]. Offchain compute is all you need. <https://www.gate.io/learn/articles/off-chain-compute-is-all-you-need/1225>. Accessed: 2023-05.
- 54 Uniswap Protocol. Hooks on uniswap v4. <https://docs.uniswap.org/contracts/v4/concepts/hook-deployment>. Accessed: 2023-05.
- 55 Uniswap Protocol. Uniswap v2 core. <https://uniswap.org/whitepaper.pdf>. Accessed: 2023-09.
- 56 Uniswap Protocol. Uniswap v3 core. <https://uniswap.org/whitepaper-v3.pdf>. Accessed: 2023-09.
- 57 Uniswap Protocol. Uniswap v4 docs. <https://docs.uniswap.org/contracts/v4/concepts/intro-to-v4>. Accessed: 2023-09.
- 58 Uniswap Protocol. Uniswapx docs. <https://blog.uniswap.org/uniswapx-protocol>. Accessed: 2023-09.
- 59 Palamarchuk Roman. Flash loan attacks: Risks and prevention. <https://hacken.io/discover/flash-loan-attacks/>. Accessed: 2023-05.
- 60 Robert H Shumway and David S Stoffer. An approach to time series smoothing and forecasting using the em algorithm. *Journal of time series analysis*, 3(4):253–264, 1982.
- 61 Rohan Tangri, Peter Yatsyshin, Elisabeth A. Duijnste, and Danilo Mandic. Generalizing impermanent loss on decentralized exchanges with constant function market makers, 2023. [arXiv:2301.06831](https://arxiv.org/abs/2301.06831).

## 25:22 Adaptive Curves for Optimally Efficient Market Making

- 62 TheiaResearch. Better blockchains lead to more profitable liquidity providers. <https://twitter.com/TheiaResearch/status/1790717593440952757>. Accessed: 2024.
- 63 Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT Press, Cambridge, Mass., 2005. URL: <http://www.amazon.de/gp/product/0262201623/102-8479661-9831324?v=glance&n=283155&n=507846&s=books&v=glance>.
- 64 Jo-Anne Ting, Evangelos Theodorou, and Stefan Schaal. Learning an outlier-robust kalman filter. In Joost N. Kok, Jacek Koronacki, Raomon Lopez de Mantaras, Stan Matwin, Dunja Mladenič, and Andrzej Skowron, editors, *Machine Learning: ECML 2007*, pages 748–756, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.
- 65 Norbert Wiener. *Cybernetics: or Control and Communication in the Animal and the Machine*. MIT Press, Cambridge, MA, 2 edition, 1948.
- 66 Jiahua Xu, Krzysztof Paruch, Simon Cousaert, and Yebo Feng. Sok: Decentralized exchanges (dex) with automated market maker (amm) protocols. *ACM Comput. Surv.*, 55(11), February 2023. doi:10.1145/3570639.
- 67 Adelyn Zhou. Flash loans aren't the problem, centralized price oracles are. <https://www.coindesk.com/tech/2020/11/11/flash-loans-arent-the-problem-centralized-price-oracles-are/>. Accessed: 2023-09.
- 68 K. Zhou, J.C. Doyle, and K. Glover. *Robust and Optimal Control*. Feher/Prentice Hall Digital and. Prentice Hall, 1996. URL: <https://books.google.com/books?id=RPSOQgAACAAJ>.

# Competitive Policies for Online Collateral Maintenance

Ghada Almashaqbeh ✉

University of Connecticut, Storrs, CT, USA

Sixia Chen ✉

Adelphi University, Garden City, NY, USA

Alexander Russell ✉

University of Connecticut, Storrs, CT, USA

IOG, Singapore

---

## Abstract

Layer-two blockchain protocols emerged to address scalability issues related to fees, storage cost, and confirmation delay of on-chain transactions. They aggregate off-chain transactions into fewer on-chain ones, thus offering immediate settlement and reduced transaction fees. To preserve security of the underlying ledger, layer-two protocols often work in a collateralized model; resources are committed on-chain to backup off-chain activities. A fundamental challenge that arises in this setup is determining a policy for establishing, committing, and replenishing the collateral in a way that maximizes the value of settled transactions.

In this paper, we study this problem under two settings that model collateralized layer-two protocols. The first is a general model in which a party has an on-chain collateral  $C$  with a policy to decide on whether to settle or discard each incoming transaction. The policy also specifies when to replenish  $C$  based on the remaining collateral value. The second model considers a discrete setup in which  $C$  is divided among  $k$  wallets, each of which is of size  $C/k$ , such that when a wallet is full, and so cannot settle any incoming transactions, it will be replenished. We devise several online policies for these models, and show how competitive they are compared to optimal (offline) policies that have full knowledge of the incoming transaction stream. To the best of our knowledge, we are the first to study and formulate online competitive policies for collateral and wallet management in the blockchain setting.

**2012 ACM Subject Classification** Theory of computation → Online algorithms; Applied computing → Digital cash

**Keywords and phrases** Blockchain layer-two solutions, Wallets, Collateral management, Online algorithms, Competitive analysis

**Digital Object Identifier** 10.4230/LIPIcs.AFT.2024.26

**Related Version** *Previous Version*: <https://arxiv.org/abs/2406.17121>

**Funding** Ghada Almashaqbeh: supported by NSF under grant No. CNS-2226932.

**Acknowledgements** We thank Mathias Fitzi for conversations that led to the original formulation of these questions.

## 1 Introduction

Distributed ledger technology has provided a financial and computational platform realizing an unprecedented combination of trust assumptions, transparency, and flexibility. Operationally, these platforms introduce two natural sources of “friction”: settlement delays and settlement costs. The Bitcoin protocol, for example, provides rather lackluster performance in both dimensions, with nominal settlement delays of approximately one hour and average fees of approximately 1 USD per transaction. Layer-two protocols have been the ready response to



© Ghada Almashaqbeh, Sixia Chen, and Alexander Russell;  
licensed under Creative Commons License CC-BY 4.0

6th Conference on Advances in Financial Technologies (AFT 2024).

Editors: Rainer Böhme and Lucianna Kiffer; Article No. 26; pp. 26:1–26:16

Leibniz International Proceedings in Informatics



LIPIC Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

these complaints as they can provide instant settlement and, furthermore, can significantly reduce transaction costs by aggregating related off-chain transactions so that they ultimately correspond to fewer underlying ledger, or on-chain, transactions. Examples of such protocols include payment channels and networks [9, 16], probabilistic micropayments [5, 7, 14], state channels and networks [6, 10, 13], and rollups [12, 15].

However, in order for layer-two protocols to provide these remarkable advantages without sacrificing the security guarantees of the underlying ledger, they must *collateralize* their activities. In particular, there must be resources committed on-chain that provide explicit recourse to layer-two clients in the event of a malicious or faulty layer-two peer or server. Moreover, the total value of the on-chain collateral must scale with the value of “in flight” transactions supported by the layer-two protocol.

These considerations point to a fundamental challenge faced by layer-two protocols: determining a policy for establishing, committing, and replenishing the collateral. Such a policy must ensure sufficient available collateral to settle anticipated transaction patterns while minimizing the total collateral and controlling the resulting number of on-chain transactions. Of course, any fixed collateralization policy can be frustrated by the appearance of an individual transaction – or a sudden burst of transactions – that exceeds the total current collateral. More generally, it would appear that designing a satisfactory policy must rely on detailed information about future transaction size and frequency, i.e., transaction distribution. From a practical perspective, this poses a serious obstacle because real-world transaction patterns are noteworthy for their unpredictability and mercurial failure to adhere to a steady state. Analytically, this immediately calls in to question the value of distribution-specific solutions. These considerations motivate us to elevate *distribution independence* as a principal design consideration for collateral policies.

We formulate a distribution-independent approach by adapting to our setting the classical framework of competitive analysis. In particular, we study two natural models: the *k-wallet* model in which the total collateral  $C$  is divided among  $k$  wallets of fixed size, and a general model in which  $C$  is viewed as one wallet that allows replenishment of any portion of  $C$ . After fixing only two parameters of the underlying system – the total collateral  $C$  and the size  $T$  of the largest transaction that we wish to support – we measure the performance of a given collateral policy against the performance of an *optimal, omniscient* policy. This optimal policy utilizes the same total collateral, but has full knowledge of the future sequence of transactions as it commits and replenishes collateral. Naively, this would appear to be an overly ambitious benchmark against which to measure an algorithm that must make choices on the fly based only on the past sequence of transactions. Our principal contribution is to show that the natural policies for these two models perform well, even when compared against this high bar.

## 1.1 Contributions

Our formal modeling is intended to reflect the challenges faced by standard layer-two protocols. The most immediate of the models we consider arises as follows: Consider a layer-two protocol with a total of  $C$  collateral that must serve an unknown transaction sequence  $\text{Tx} = (\text{tx}_1, \text{tx}_2, \dots)$ . As each transaction arrives, the policy may either commit a corresponding portion of its available collateral to *settle* this transaction or simply *discard* it; in particular, in any circumstances where there isn’t sufficient uncommitted collateral to cover a given transaction, the transaction must be discarded. The policy may also – whenever it chooses – replenish its currently committed collateral. This “flush” procedure returns the committed collateral to the available pool of collateral after a fixed time delay  $F$  and involves

a fixed cost  $\tau$  (so transactions arriving during  $F$  will be discarded if no other sufficient collateral is available). Thus, the challenge is to schedule the flush events so as to minimize the total cost while simultaneously maximizing the total value of settled transactions.

We remark that transactions “discarded” in the model above would typically be handled by some other fallback measure in a practical setting. The flush operation, in practice, corresponds to on-chain settlement of a family of transactions that releases the associated collateral so that it can be reused as surety for additional transactions. While we assume that the flush procedure is associated with a fixed, constant cost for simplicity, in practice this cost may scale with the complexity of the aggregated transactions. We remark that a fixed cost directly models Lightning-like payment channels and networks, or escrow-based probabilistic micropayments, where the total number of participants is bounded.<sup>1</sup>

In this general setting, we study the natural family of policies determined by a parameter  $\eta \in (0,1)$  that settle transactions as they arrive until an  $\eta$ -fraction of all collateral is consumed; at this point the committed collateral is flushed and the process is continued with the remaining collateral. Our analytic development first focuses on a simpler variation – of interest in its own right – that we call the *k-wallet* problem. As above, the policy is challenged to serve a sequence Tx of transactions with a total of  $C$  collateral; however, the collateral is now organized into  $k$  wallets, each holding  $C/k$  collateral, with the understanding that an entire wallet must be flushed at once. When a wallet is flushed it becomes entirely unavailable for settlement – regardless of how much of the wallet was actually committed to settled transactions – until the end of the flush period  $F$ , when the collateral in the wallet is again fully available for future settlement. As above, the policy may settle a transaction by committing a portion of collateral in one of the wallets corresponding to the size of the transaction. This version of the problem has the advantage that performance is captured by a single quantity: the total value of settled transactions.

## 1.2 A Survey of the Results

Continuing to discuss the *k-wallet* model, we consider a sequence Tx of transactions, each of value no more than  $T$ . We focus on the natural FLUSHWHENFULL policy, which maintains a *single active wallet* (unless all wallets are currently unavailable) that is used to settle all arriving transactions; if settling a transaction would leave negative residual committed collateral in the active wallet, the wallet is flushed and a new wallet is activated as soon as one becomes available. We prove that this simple, attractive policy settles at least a fraction

$$\frac{1 - kT/C}{1 + 1/k}$$

of the total value settled by an optimal, offline strategy with  $C$  collateral, even one that is not restricted to a *k-wallet* policy but can flush any portion of its collateral at will. We remark that this tends to optimality for large  $k$  and small  $T < C/k$ . This result also answers a related question: that of how many wallets one should choose for a given total collateral  $C$  and maximum transaction size  $T$ . We find that optimal  $k$  in this case is  $\approx \sqrt{1 + C/T} - 1$ .

As for the more flexible setting – under the general  $C$  collateral model – where the policy may flush any portion of its collateral at will by paying a transaction fee  $\tau$ , recall that this poses a bicriteria challenge: maximizing settled transactions while reducing settlement fees.

<sup>1</sup> On-chain transaction cost also varies based on network conditions; during periods of high activity or congestion, transaction issuers may resort to increasing transaction fees to incentivize miners to prioritize their transactions. As such,  $\tau$  above is viewed as the average transaction cost.

We study this by establishing the natural figure of merit that arises by assuming that each settled transaction yields positive utility to the policy that scales with its value (e.g., a “profit margin”). Thus, the policy seeks to maximize  $pV - \tau f$ , where  $V$  is the total value of settled transactions,  $f$  is the total number of flushes, and  $p$  is the profit margin. Here we study the family of policies that flush when currently committed (but unflushed) collateral climbs to an  $\eta$ -fraction of  $C$  ( $\eta$  is a policy parameter). We find that this policy achieves total utility of at least  $1/\alpha$  fraction of that achieved by the optimal omniscient policy, where

$$\alpha = \frac{1}{1 - \eta - T/C} \cdot \frac{p/\tau - 1/C}{p/\tau - 1/(\eta C)}.$$

In this case, we are also able to determine the optimal constant  $\eta^*$  (as a function of  $C$ ,  $p$ , and  $\tau$ ) that maximizes the policy utility:

$$\eta^* = \sqrt{(1 - T/C) \cdot \tau / (pC)}.$$

We remark that our results in the  $k$ -wallet setting can also be applied to directly yield results with this accounting that assigns a flush cost and a profit margin.

We study some additional questions that arise naturally. For example, we show that no deterministic, single wallet policy can be competitive if the maximum transaction size can be as large as the wallet size and show that, on the other hand, a natural randomized algorithm is  $O(1)$ -competitive.

### 1.3 Applications

Online collateral management arises in various layer-two protocols, as well as in Web 3.0 and decentralized finance (DeFi) applications. For layer-two protocols, payment networks are an emblematic example: A relay party creates payment channels with several parties, allowing her to relay payments over multi-hop routes. Each payment channel is tied to a collateral  $C$  such that the relay cannot accept a transaction to be relayed if the remaining collateral cannot cover it. This applies as well to state channels, where transactions created off-chain – while the channel is active – are accepted only if their accumulated value does not exceed the initial fund committed when the channel was created. These configurations adhere to the general collateral model discussed above.

Probabilistic micropayments follow a slightly different setting. Micropayments are usually used to permit service-payment exchange without a trusted party to reduce financial risks in case of misbehaving entities. A client creates an escrow fund containing the collateral backing all anticipated payments to a set of servers. A server provides a service to the client (e.g., file storage or content distribution) in small chunks, so that the client pays a micropayment for each chunk. For any incoming service exchange, the client cannot take it unless her collateral can pay for it. The client can decide to replenish the escrow fund to avoid service interruption, thus this also follows the general collateral model. The client may also choose to divide her collateral among several escrows, each of which has a different or similar setting with respect to, e.g., the set of servers who can be paid using an escrow and the total service payment amount. This configuration follows the  $k$ -wallet model.

Apart from layer-two scalability solutions, online collateral management captures scenarios related to Web 3.0 and DeFi applications. The framework of decentralized resource markets build systems that provide digital services, e.g., file storage, content distribution, computation outsourcing or video transcoding, in a fully decentralized way [1–3]. Due to their open-access nature, where anyone can join the system and serve others, these systems usually involve



some form of collateral. In this case, a collateral represents the amount of service a party wants to pledge in the system. For example, in Filecoin [1] – a distributed file storage network – a storage server commits collateral proportional to the amount of storage she claims to own. This server cannot accept more file storage contracts, and subsequently more storage payments, than what can be covered by the pledged storage (or alternatively collateral).

In the DeFi setting, online collateral management is encountered in a variety of applications. Loan management is a potential example [11,17]; incoming loan requests cannot be accepted unless the loan funding pool can support them. The loan DeFi application then has to decide a policy for loan request accept criteria (to favor some requests over others under the limited funding constraint) and when to replenish the loan pool balance.

Another potential application, of perhaps an extended version of our models and policies, that we believe to be of interest is the case of automated market makers (AMMs) [19]. Here, a liquidity pool trades a pair of tokens against each other, say token  $A$  and token  $B$ , such that a trade buying an amount of token  $A$  pays for that using an amount of token  $B$ , and vice versa. Incoming trades are accepted only if the liquidity pool can satisfy them, so in a sense having tokens that can serve the requested trades is the collateral. Replenishing the pool fund, or liquidity, can be done organically based on the trades. That is, a particular trade, say to buy  $A$  tokens, reduces the backing fund of token  $A$  while increasing it for token  $B$ . Another approach for pool replenishment is via liquidity providers; particular parties provide their tokens to the pool to serve incoming trades (or token swaps) in return for some commission fees. These providers can configure when their offered liquidity can be used, i.e., at what trading price range, under what is called concentrated liquidity as in Uniswap [4]. An interesting open question is to develop competitive collateral policies that capture this setting where settling a transaction does not only depend on whether the remaining collateral  $C$  (i.e., pool liquidity) can cover it, but also on transaction-specific parameters to meet certain collateral-related conditions. Even the replenishment itself, i.e., providing liquidity, could be subject by other factors such as the resulting price slippage, so an incoming mint transaction (in the language of AMMs) that provides liquidity may not be accepted immediately. We leave these questions as part of our future work directions.

In general, our work lays down foundations for wallet management to address issues related to robustness, availability, and profitability of the wallet(s) holding the collateral. Maintaining one wallet may lead to periods of interruption; a party must wait for a while before a new wallet is created to replace an older expired one. Maintaining several wallets may help, but given the cost of locking currency in a wallet or renewing it, the number of active wallets and their individual balances must be carefully selected. Moreover, under this multi-wallet setting, it is important to consider how incoming transactions are matched to the wallets, and whether factors such as payment amount or frequency may impact this decision. A potential extension to our model is considering adaptive policy management, where the size of the collateral and the number of wallets can be adjusted after each flush decision to account for these varying factors.

## 2 The Model; Measuring Policy Quality

As discussed above, we consider the problem of designing an online collateral management policy in which a collateral fund of initial value  $C$  is used to settle transactions – each with a positive real value no more than  $T$  – chosen from a sequence  $\text{Tx} = (\text{tx}_1, \text{tx}_2, \dots)$ . Operationally, the policy is presented with the transactions one-by-one and, as each transaction arrives, it

must immediately choose whether to *settle* the transaction or *discard* it. Settling a transaction requires committing a portion of the collateral equal to the value of the transaction; such committed collateral cannot be used to settle future transactions. Of course, if there isn't sufficient uncommitted collateral remaining to settle a given transaction when it arrives, the transaction must be discarded. Committed collateral may be returned to service by an operation we call a *flush*; we focus on two different conventions for the flush operation, described below, but in either case the collateral only becomes available for use after a fixed time delay  $F$ . We assess the performance of a particular online policy  $A$  against that of an *optimal offline policy*  $OPT$  that knows the full sequence  $\text{Tx}$  and can make decisions based on this knowledge.

Below, we describe two models for the collateral: the discrete  $k$ -wallet model and the general collateral model.

## 2.1 The Discrete $k$ -Wallet Model

The  $k$ -wallet model calls for the collateral to be divided into  $k$  wallets, each with  $C/k$  collateral value. Wallets support two operations: (i) a wallet with uncommitted collateral  $R$  may immediately settle any transaction  $\text{tx}$  of value  $v \leq R$ ; this reduces the available collateral of the wallet to  $R - v$ , and (ii) a wallet may be flushed, which takes the wallet entirely offline for a flush period  $F$  after which the available collateral  $R$  is reset to  $C/k$ . As a matter of bookkeeping, we mentally organize time into short discrete slots indexed with natural numbers: we then treat the transaction  $\text{tx}_t$  as arriving at time(slot)  $t$ , and set  $\text{tx}_t = 0$  for times  $t$  when no transactions arrive. We treat the flush period as a half-open and half-closed interval: if a wallet flushes at time  $t$ , then it is offline during the time interval  $(t, t + F]$ . In this model, the figure of merit is the *total value* of settled transactions. We let  $\text{Disc}_T^{C,k}$  denote this discrete  $k$ -wallet model with maximum transaction size  $T$ .

**Settlement algorithms, settled value, and the competitive ratio.** A  *$k$ -wallet settlement algorithm*  $A$  is an algorithm that determines, for any transaction sequence  $\text{Tx}$ , whether to settle each transaction, which wallet to use, and when to flush each wallet. For such an algorithm  $A$  and a sequence  $\text{Tx} = \text{tx}_1, \text{tx}_2, \dots, \text{tx}_n$  we let  $A[\text{Disc}_T^{C,k}; \text{Tx}]$  denote the total value of all transactions settled by the algorithm. In general, we use the notation  $A[\mathcal{M}; \text{Tx}]$  to denote the value achieved by algorithm  $A$  in model  $\mathcal{M}$  with input sequence  $\text{Tx}$ . When the model is clear from context, we simply write  $A[\text{Tx}]$ .

We say that an algorithm  $A$  is *online* if, for every  $N$ , any decisions made by the algorithm at time  $N$  depend only on  $\text{tx}_1, \text{tx}_2, \dots, \text{tx}_N$ , i.e., transactions seen so far. We let  $OPT$  denote the optimal (offline) policy; thus  $OPT[\text{Disc}_T^{C,k}; \text{Tx}]$  denotes the maximum possible value that can be achieved by any policy, even one with a full view of all (past and future) transactions.

► **Definition 2.1.** We say that an algorithm  $A$  is  $\alpha$ -competitive in the  $k$ -wallet model if, for any sequence  $\text{Tx} = \text{tx}_1, \dots, \text{tx}_n$  with maximum value no more than  $T$ ,

$$OPT[\text{Disc}_T^{C,k}; \text{Tx}] \leq \alpha \cdot A[\text{Disc}_T^{C,k}; \text{Tx}] + O(1),$$

where the constant in the asymptotic notation may depend on the model parameters ( $C$ ,  $k$ , and  $T$ ), but not the sequence  $\text{Tx}$  or its length  $n$ .

► **Remark 2.2 (Relation to the bin packing problem).** We remark on the relationship between our problem and the well-studied online bin packing problem [8, 18], where an algorithm must pack arriving objects into bins of constant size, while *opening* a new bin any time a

newly arriving object does not fit into any of the current bins. In this context, the  $k$ -wallet model calls for a bounded number of *bins* (a.k.a., wallets) that can only be reset with the flush operation. Also, we measure the total settled value rather than the number of utilized bins. In any case, we adopt the standard classical paradigm of competitive analysis to study our algorithms, as described previously.

## 2.2 The General Collateral Model

In contrast to the discrete  $k$ -wallet model, where each wallet must be flushed as a whole, the general setting permits any portion of the collateral to be flushed at any time. The basic framework is identical: the policy is presented with a sequence of transactions  $\mathbf{tx}_1, \mathbf{tx}_2, \dots$  and must decide whether each transaction will be settled or discarded; the total collateral  $C$  and the maximum transaction size  $T$  are parameters of the problem. Settling a transaction requires committing collateral of value equal to the transaction; however, any portion of the committed collateral can be flushed at any time. As before, each flush period is  $F$  and is defined to be a half-open and half-closed time interval. We denote this model as  $\text{Gen}_T^C$ .

Since there is no penalty for flushing collateral in this model, it is clear that any algorithm may as well immediately flush any committed collateral. Despite the simple appearance of the model, it is still useful to consider this setting as a comparison reference point for  $k$ -wallet policies, and we define  $A[\text{Gen}_T^C; \mathbf{Tx}]$  to be the total value of transactions settled by algorithm  $A$  in this general model for a transaction sequence  $\mathbf{Tx}$  (with total collateral  $C$  and maximum transaction size  $T$ ).

► **Definition 2.3.** *We say that an algorithm  $A$  is  $\alpha$ -competitive in the general collateral model if, for any sequence  $\mathbf{Tx} = \mathbf{tx}_1, \dots, \mathbf{tx}_n$  with maximum value  $T$ ,*

$$\text{OPT}[\text{Gen}_T^C; \mathbf{Tx}] \leq \alpha A[\text{Gen}_T^C; \mathbf{Tx}] + O(1).$$

where the  $O(1)$  term may depend on model parameters but not on  $\mathbf{Tx}$  or  $n$ .

Note that for any algorithm  $A$  defined in the  $k$ -wallet model the following is always true:

$$A[\text{Disc}_T^{C,k}; \mathbf{Tx}] \leq \text{OPT}[\text{Disc}_T^{C,k}; \mathbf{Tx}] \leq \text{OPT}[\text{Gen}_T^C; \mathbf{Tx}].$$

A more natural model arises by introducing a cost for flushes. In order to reflect the relative cost of flushes in the context of settled transactions, we introduce two additional parameters:

1. *Profit margin  $p$* : a profit  $p \cdot v$  is gained when a transaction with value  $v$  is settled.
2. *Flush cost  $\tau$* : each flush operation costs  $\tau$ .

We assume throughout that  $pC > \tau$ ; otherwise there is no value to settling transactions because the cost of even single flush exceeds the total profit that can be accrued from the flushed collateral. We let  $\text{Gen}_{T;p}^{C;\tau}$  denote this model, observing that  $\text{Gen}_T^C$  and  $\text{Gen}_{T;1}^{C;0}$  coincide. In keeping with the notation above, we let  $A[\text{Gen}_{T;p}^{C;\tau}; \mathbf{Tx}]$  denote the total profit minus flush cost by applying algorithm  $A$  in the general model with total collateral  $C$ , maximum transaction size  $T$ , profit margin  $p$ , flush cost  $\tau$ , and transaction sequence  $\mathbf{Tx}$ . Then, we have the following.

► **Definition 2.4.** *We say that an algorithm  $A$  is  $\alpha$ -competitive in the general collateral model with flush costs if, for any sequence  $\mathbf{Tx} = \mathbf{tx}_1, \dots, \mathbf{tx}_n$  with maximum value  $T$ ,*

$$\text{OPT}[\text{Gen}_{T;p}^{C;\tau}; \mathbf{Tx}] \leq \alpha \cdot A[\text{Gen}_{T;p}^{C;\tau}; \mathbf{Tx}] + O(1),$$

where the  $O(1)$  term may depend on the model parameters but not  $\mathbf{Tx}$  or  $n$ .

■ **Table 1** Summary of our results. Here  $r = kT/C$ ,  $\tau$  is the flush cost,  $p$  is the profit margin, and  $f$  is the number of flushes.

<b>Discrete <math>k</math>-wallet model</b>	
$r < 1$	Theorem 3.1: FLUSHALL is $(2 - r)/(1 - r)$ -competitive
	Theorem 3.2: FLUSHWHENFULL is $(k + 1)/(k(1 - r))$ -competitive
$r = 1$	$k = 1$ Theorem 3.3: No competitive deterministic settlement algorithm
	$k > 1$ Theorem 3.5: FLUSHALL is 3-competitive
	$k > 1$ Theorem 3.6: FLUSHTWOWHENFULL is $2(k + 1)/k$ -competitive
<b>General collateral model</b>	
maximize $V$	Corollary 4.1: FLUSHWHENFULL is $(k + 1)/(k(1 - r))$ -competitive
maximize $pV - \tau f$	Theorem 4.4: $A_\eta$ is $(1 - \beta)/(\sqrt{1 - T/C} - \sqrt{\beta})^2$ -competitive, where $\eta = \sqrt{\beta(1 - T/C)}$ and $\beta = \tau/pC$

**Transaction size.** Our analysis identifies two regimes of interest regarding transaction costs (for both of the previous models): the “micro-transaction” setting, where  $T \ll C$  (arising in micropayment applications) and “arbitrary” transaction size when  $T \approx C$  (arising in more general settings).

In the next two sections, we analyze policy competitiveness under each model; the discrete  $k$ -wallet model can be found in Section 3 and the general collateral model can be found in Section 4. Table 1 summarizes our results.

### 3 The Discrete $k$ -Wallet Setting

We now formally consider the  $k$ -wallet setting. Our focal points are two natural policies described next: FLUSHALL and FLUSHWHENFULL.

#### 3.1 The FlushAll Algorithm

We begin with the simple FLUSHALL algorithm, which uses  $k$  wallets placed in (arbitrary, but fixed) order  $W_1, \dots, W_k$ . The algorithm packs transactions into its wallets using the *first fit* algorithm: each transaction is settled by the first wallet (in the established order) that can fit the transaction until a transaction arrives that cannot fit into any wallet. At that time, all  $k$  wallets are simultaneously flushed (and so during the flush period  $F$  all incoming transactions will be discarded).

In the following theorems, we use  $r$  to denote  $kT/C$ , which is the ratio between the maximum transaction size and the wallet size. Note that  $r \leq 1$ .

► **Theorem 3.1.** *FLUSHALL is  $(2 - r)/(1 - r)$ -competitive in the  $\text{Disc}_T^{C,k}$  model, where  $r = kT/C$ .*

**Proof.** For a sequence Tx of transactions, subdivide time into *epochs* according to the behavior of the FLUSHALL algorithm. The first epoch begins at time 0 and continues through the first flush of the  $k$  wallets; the epoch ends in the last timeslot of this flush period. Each subsequent epoch begins in the timeslot when the wallets come back online (that is, in the timeslot just after the previous epoch ends) and continues through the next flush to the end of the flush period. In general, there may be a final *partial epoch* at the end of the transaction sequence; other epochs are referred to as *full*. Any full epoch can be further broken into two phases: the *accumulation phase* when all transactions are settled by FLUSHALL, and the *flush phase*, during which no transactions can be settled (as all wallets are offline).

For any particular full epoch, let  $V$  be the total value packed by FLUSHALL into its wallets in the accumulation phase. We note that  $V \geq k(C/k - T) = C - kT$ , since every wallet will clearly be filled to at least  $C/k - T$ . As for OPT, during the accumulation phase it can settle at most  $V$  (as this is the value of all transactions appearing in that phase) and during the flush phase it can settle at most  $C$  (as a unit of collateral can settle at most one transaction unit in any  $F$  period). Therefore, the ratio between the value settled by OPT and FLUSHALL in a full epoch is no more than

$$\max_{C-kT \leq V \leq C} \frac{V+C}{V} \leq \frac{C-kT+C}{C-kT} = \frac{(2-kT/C)}{(1-kT/C)} = \frac{2-r}{1-r}.$$

Moreover, the same formula above can be said for any partial epoch, since the accumulation phase comes first.

Thus, the competitive ratio is  $\alpha = (2-r)/(1-r)$ . Observe that when  $r$  decreases, the competitive ratio approaches 2.  $\blacktriangleleft$

Aside from the simplicity of the analysis, FLUSHALL may have an advantage for certain sequences of transactions in practice: keeping all  $k$  wallets open during the epoch (rather than optimistically flushing some earlier so as to bring new collateral online earlier) may permit higher density packing of transactions into the wallets. Indeed, one could consider leveraging an approximation algorithm for bin packing for the purposes of optimizing this. On the other hand, in situations where some of the wallets may become nearly full early in an epoch it seems wasteful to wait to flush these wallets until all others are full. This motivates the FLUSHWHENFULL algorithm, which attempts to more eagerly flush wallets so as to bring them online sooner.

### 3.2 The FlushWhenFull Algorithm

We now consider the FLUSHWHENFULL algorithm, which fills wallets in a round-robin order. Specifically, transactions are settled by a particular wallet until a new transaction arrives that cannot fit; at that point the wallet is immediately flushed, and the algorithm moves on to the next wallet in cyclic order. (In cases where the next wallet is offline, the algorithm waits for the wallet to finish its flush before processing further transactions, so all transactions arriving during this wait period will be discarded.)

► **Theorem 3.2.** *For  $k > 1$ , FLUSHWHENFULL is  $(k+1)/(k(1-r))$ -competitive in the  $\text{Disc}_T^{C,k}$  model, where  $r = kT/C$ .*

**Proof.** Assume, for the purpose of contradiction, that there is a time  $t$  for which the interval  $I = (0, t]$  satisfies

$$V_{\text{OPT}}(I) > (k+1)/(k(1-r)) \cdot V_{\text{FWF}}(I),$$

where  $V_{\text{OPT}}(I)$  and  $V_{\text{FWF}}(I)$  are the total values of transactions OPT and FLUSHWHENFULL settle during  $I$ , respectively; let  $t_e$  be the earliest such  $t$ .

Since  $t_e$  is the earliest such time, there must be a transaction tx at  $t_e$  that is not settled by FLUSHWHENFULL. As FLUSHWHENFULL does not take tx, it must be the case that either all wallets are offline at  $t_e$  or  $k-1$  wallets are already offline at  $t_e$  and the remaining wallet goes offline at  $t_e$  after failing to fit tx. Therefore, every wallet flushes during  $I_f = (t_e - F, t_e]$ . Suppose, without loss of generality, that they do so in order  $W_1, W_2, \dots, W_k$ .

## 26:10 Competitive Policies for Online Collateral Maintenance

If  $t_e \leq F$ , then OPT settles transaction value at most  $C$  in the interval  $(0, t_e]$  since each wallet settles at most  $C/k$ . In the same interval, FLUSHWHENFULL settles at least  $k(C/k - T)$  since each wallet settles at least  $C/k - T$ . Therefore,

$$\frac{V_{\text{OPT}}(I)}{V_{\text{FWF}}(I)} \leq \frac{C}{k(C/k - T)} = \frac{C}{C - kT} < \frac{kC}{kC - k^2T} + \frac{C}{kC - k^2T} = \frac{k+1}{k(1-r)},$$

which would contradict our assumption.

Otherwise  $t_e - F > t_0$ . Observe that of the  $k$  wallets, at least  $W_2, W_3, \dots, W_k$  began taking transactions during  $I_f$  since, if a wallet  $W_i$ 's transaction activity before its last flush starts at a time before  $I_f$  for any  $i = 2, \dots, n$ , then  $W_{i-1}$ 's last flush time must also be before  $I_f$  which contradicts the earlier conclusion that all the  $k$  wallets' last flush times are during  $I_f$ . Therefore, those  $k-1$  wallets together contribute  $(k-1)(C/k - T)$  to  $V_{\text{FWF}}(I_f)$ . The only wallet that may have started taking transactions before  $I_f$  is  $W_1$ . Let  $t_s$  denote the last time before  $t_e$  that  $W_1$  came back online and  $t_{s'}$  denote the time  $W_1$  flushes. Note that  $t_{s'} \in I_f$ , while  $t_s$  may or may not be in the interval. Let  $I_s = (t_s, t_{s'}]$  and  $I_{s'} = (t_{s'}, t_e]$ ; then we have  $V_{\text{FWF}}(I_s \cup I_{s'}) \geq k(C/k - T)$  since each wallet starts to take transactions and then flushes within the interval  $I_s \cup I_{s'}$ . We also have  $V_{\text{OPT}}(I_s) \leq V_{\text{FWF}}(I_s) < C/k$  since wallet  $W_1$  is active during  $I_s$ .

Additionally, we have  $V_{\text{OPT}}(I_{s'}) \leq C$  since the length of  $I_{s'}$  is no more than  $F$ , leading to  $V_{\text{OPT}}(I_s \cup I_{s'}) \leq C/k + C$ . Therefore,

$$\frac{V_{\text{OPT}}(I_s \cup I_{s'})}{V_{\text{FWF}}(I_s \cup I_{s'})} \leq \frac{C/k + C}{k(C/k - T)} = \frac{k+1}{k} \cdot \frac{C}{C - kT} = \frac{k+1}{k(1-r)}.$$

But this contradicts our initial assumption; we conclude that there is no such  $t$ . ◀

### 3.3 Optimal Wallet Number

When  $k$  is large and  $r$  is small, FLUSHWHENFULL approaches optimality. For a given total collateral  $C$  and maximum transaction size  $T$ , it is natural to ask how many wallets one should choose so as to optimize the competitive ratio of FLUSHWHENFULL. This amounts to determining a  $k$  that minimizes  $(k+1)/(k(1-kT/C))$ . By computing

$$\frac{\partial}{\partial k} \left( \frac{k+1}{k(1-kT/C)} \right) = 0,$$

we find that the optimal value  $k^*$  for  $k$  is  $\sqrt{1 + C/T} - 1$ . Of course, the actual number of wallets must be an integer. We remark that if  $k \approx \sqrt{C/T}$ , then each wallet has size  $\approx \sqrt{CT}$  and the competitive ratio is approximately

$$\frac{\sqrt{C} + \sqrt{T}}{\sqrt{C} - \sqrt{T}}.$$

### 3.4 Remarks on the profit margin–transaction cost setting

We remark that the competitive analyses above focusing on total settled value immediately give rise to a bound for the setting that introduces a profit margin  $p$  and a flush cost  $\tau$ . Observe that, for any algorithm constrained to the  $k$ -wallet framework that settles total value  $V$ , the maximum profit is  $V(p - \tau k/C)$ , as only  $C/k$  value can be settled in any single flush. Thus the profit of OPT is no more than  $V_{\text{OPT}}(p - \tau k/C)$ . On the other hand, the profit of

FLUSHWHENFULL is at least  $V_{\text{FWF}}(p - \tau k / (C - kT)) - O(1)$ , as each wallet is flushed with at least  $C/k - T$  value (except for the last wallet, which may introduce a  $O(1)$  additional penalty). It follows that the competitive ratio in the profit model is inflated by a factor

$$\frac{p/\tau - k/C}{p/\tau - k/(C - kT)}$$

over that of the “value-only”  $k$ -wallet setting.

Note that the same argument can be applied to FLUSHALL because each wallet is likewise flushed with at least  $C/k - T$  value (except perhaps for the last flush event).

### 3.5 Remarks on the Case $r = 1$

If the maximum transaction size can be as large as the wallet size, we make a few additional observations:

1. No deterministic algorithm can be competitive if there is only one wallet.
2. FLUSHALL is 3-competitive.
3. FLUSHWHENFULL is not competitive, but a variation on the scheme that groups wallets into pairs can solve the problem.

We prove these in the following.

► **Theorem 3.3.** *There is no competitive, deterministic 1-wallet settlement algorithm if  $r = 1$ .*

**Proof.** For the sake of simplicity, we assume the wallet size and maximum transaction size are both 1. Fixing an online algorithm A, consider the following schedule of transactions:

- Begin with a rapid succession of one or more *microtransactions* each having size  $\epsilon$ , terminating with the first microtransaction that the algorithm chooses to settle.
  1. If the algorithm does not choose to settle any of the microtransactions, end the succession after  $1/\epsilon$  transactions.
  2. If the algorithm *does* choose to settle one, follow it immediately with a transaction of size 1.
- Allow an interval of length  $F$  to pass without any transactions.
- Repeat indefinitely.

In any iteration of the above, either case 1 or case 2 applies. In case 1, the online algorithm settles no transactions, while the optimal offline algorithm settles a total value of 1. In case 2, the online algorithm settles a single transaction worth  $\epsilon$  while the optimal offline algorithm settles a single transaction of size 1. Therefore, the competitive ratio is no better than  $1/\epsilon$ . As  $\epsilon$  can be chosen arbitrarily, it follows that the algorithm cannot achieve any fixed ratio. ◀

► **Remark 3.4.** A simple randomized algorithm can achieve constant competitive ratio when both  $k$  and  $r$  are 1. We first show that FLUSHALL with 2 wallets is 2-competitive against OPT with one wallet. During each epoch, which extends from the time the two wallets come back online after the previous flush until the end of the next flush period, FLUSHALL settles total value  $V \geq 1$ . On the other hand, OPT can settle at most  $V + 1$ , that is, during the time FLUSHALL settles transactions, OPT settles  $V$ , and during the flush time period of FLUSHALL, OPT packs 1. Therefore, the competitive ratio is  $(V + 1)/V \leq 2$ . Now we will let our randomized algorithm that uses one wallet to simulate one of the wallets in the FLUSHALL algorithm with 2 wallets. At each time when the wallet comes back online, we flip

## 26:12 Competitive Policies for Online Collateral Maintenance

a coin, if it is heads, it simulates the first wallet in FLUSHALL, and if it is tails, it simulates the second wallet in FLUSHALL. That is, the wallet in the randomized algorithm only settles the transactions that are taken by the chosen wallet and ignores the other transactions. The expected value the randomized algorithm can pack in each epoch is half of what FLUSHALL can pack. Hence the competitive ratio against one-wallet OPT is 4.

► **Theorem 3.5.** *For any number  $k > 1$  of wallets FLUSHALL is 3-competitive if  $r = 1$ .*

**Proof.** We use a similar analysis as the proof in Theorem 3.1. Time is divided into *epochs*, each of which contains the *accumulation phase* and the *flush phase*. For any particular full epoch, let  $V$  be the total value packed by FLUSHALL into its wallets in the accumulation phase. We note that  $V \geq C/2$ . To see this, observe that for any pair of wallets  $W_i$  and  $W_j$  with  $i < j$  the final transaction values  $v_i$  and  $v_j$  of the wallets must satisfy  $v_i + v_j > C/k$  – otherwise the transactions in the later wallet  $j$  would have been placed in the earlier wallet  $i$  by first fit. Summing these constraints

$$\sum_{i < j} (v_i + v_j) \geq \sum_{i < j} \frac{C}{k} \quad \Rightarrow \quad (k-1) \sum_i v_i \geq \frac{k(k-1)}{2} \frac{C}{k} \quad \Rightarrow \quad \sum_i v_i \geq \frac{C}{2}.$$

OPT can settle at most  $V + C$  in this epoch. Considering that  $V \geq C/2$ , the quantity  $V + C \leq 3V$ , as desired. It follows that the competitive ratio is  $\alpha \leq 3$  as desired. ◀

Unfortunately, when  $r = 1$ , the competitive ratio for FLUSHWHENFULL is unbounded. To see that, again, assume the maximum transaction size and wallet size are both 1. The adversary can produce a series of suitably spaced transactions alternating in value between  $\epsilon$  and 1. FLUSHWHENFULL will be forced to take all the  $\epsilon$ -valued transactions and forgo the high-value transactions, while OPT can decline to process the low-value transactions in order to process all the high-value ones. Therefore, the competitive ratio would be  $1/\epsilon$ . This problem can be solved if we pair consecutive wallets and flush each pair when a transaction can not be settled by either of the two wallets. Within each pair, the second wallet takes a transaction when it is too large for the first wallet. We denote this algorithm as FLUSHTWOWHENFULL, for which we have the following result.

► **Theorem 3.6.** *When  $k > 1$ , FLUSHTWOWHENFULL is  $2(k+1)/k$ -competitive if  $r = 1$ .*

**Proof.** The proof is similar to the proof of Theorem 3.2. We use the same notations as before. Between time interval  $(t_0, t]$ , FLUSHTWOWHENFULL can settle transaction value at least  $C/2$  since each pair settles at least  $C/k$  before they flush, while OPT settles at most  $C + C/k$ . Therefore, the competitive ratio is  $2(k+1)/k$ . ◀

### 4 The General Collateral Setting

In this section, we study the general model where the entire collateral  $C$  is held in a single pool. A collateral maintenance policy can replenish any portion of committed collateral (used to settle a transaction) at any time. Even with this additional flexibility, a unit of collateral can only be used for settlement once in a time period of length  $F$ ; it follows that the total settled value of transactions in any time period of length  $F$  is no more than  $C$ . Thus, using the same proof as in Theorem 3.2, we conclude the following, which shows that FLUSHWHENFULL is competitive even when compared against an adversary who may use the full power of the general model (while FLUSHWHENFULL continues to be constrained operate in the  $k$ -wallet discrete model).



► **Corollary 4.1.** *Setting  $r = kT/C$ ,*

$$\text{OPT}[\text{Gen}_T^C; \text{Tx}] \leq \frac{k+1}{k(1-r)} \cdot \text{FLUSHWHENFULL}[\text{Disc}_T^{C,k}; \text{Tx}].$$

The above result concerns the total transaction value  $V$  settled by an algorithm. As mentioned in the introduction, without further constraints on the adversary it's clear that the optimal approach (in the general model) is to immediately flush any collateral used to settle a transaction. In practice, this is unattractive as there is, in fact, a cost associated with the (typically on-chain) transaction used to refresh collateral. To study this, we introduce two new parameters: (i.)  $p$ , the profit margin: the algorithm is provided a reward of  $p \cdot v$  for settling a transaction of value  $v$ , (ii.)  $\tau$ , the cost of any flush (regardless of the amount of collateral involved in the flush operation).

We seek to maximize the total profit with flush cost deducted. Formally, we would like to find an algorithm that selects transactions to settle so that  $p \cdot V - \tau f$  is maximized, where  $V$  is the total value of settled transactions and  $f$  is the total number of flushes. (Note that by scaling the figure of merit by  $1/\tau$ , this is equivalent to maximizing  $(p/\tau)V - f$  and it follows that the single parameter  $p/\tau$  suffices; we separate these merely for the purpose of intuition.) Recall that we use  $A[\text{Gen}_{T;p}^{C;\tau}; \text{Tx}]$  to denote  $pV - \tau f$  for an algorithm  $A$ .

Inspired by the algorithm `FLUSHWHENFULL`, we consider a family of policies that flush when the currently committed collateral has reached a specified fraction of  $C$ .

#### 4.1 The Threshold Algorithm $A_\eta$

This algorithm is parameterized by a threshold  $\eta$  for which  $T/C \leq \eta \leq 1$ . The behavior of the algorithm is determined by the running quantity  $R$ , the current total collateral that has been committed to settle transactions, but not (yet) flushed. The algorithm proceeds as follows: When a new transaction  $\text{tx}$  arrives, it is settled if and only if there is sufficient remaining collateral. Immediately after settling a transaction, if  $R \geq \eta C$  (so that there is at least  $\eta C$  committed but unflushed collateral), then it flushes exactly  $\eta C$  collateral.

The following analysis derives the competitive ratio of  $A_\eta$  and then computes the optimal value of  $\eta$ , denoted by  $\eta^*$ , that minimizes this competitive ratio.

► **Lemma 4.2.**  $\text{OPT}[\text{Gen}_T^C; \text{Tx}] \leq \frac{C}{C - \eta C - T} A_\eta[\text{Gen}_T^C; \text{Tx}].$

**Proof.** The proof is similar to the proof of Theorem 3.2, so we are somewhat more brief. For contradiction, assume there is a (first) time  $t_e$  for which the interval  $I = (0, t_e]$  satisfies

$$V_{\text{OPT}}(I) > C/(C - \eta C - T) \cdot V_{A_\eta}(I),$$

where  $V_{\text{OPT}}(I)$  and  $V_{A_\eta}(I)$  are the total values of transactions `OPT` and  $A_\eta$  settle during  $I$ , respectively.

Since  $t_e$  is the earliest such time, there must be a transaction  $\text{tx}$  at  $t_e$  that is not settled by  $A_\eta$ . As  $A_\eta$  does not take  $\text{tx}$ , there are two possibilities: 1) all collateral is offline at  $t_e$ ; 2) the remaining uncommitted collateral is insufficient to settle  $\text{tx}$ . Let  $I_f = (t_e - F, t_e]$ . Recall that collateral is flushed sequentially in portions of size  $\eta C$ , and that any such portion will only start to take transactions after (or at the same time that) the previous portion has been flushed. Let  $W_k$ , refer to the remaining portion of unflushed collateral at time  $t_e$ , if any, and to the last-flushed portion of collateral otherwise. Let  $W_1, W_2, \dots, W_{k-1}$  refer to the portions of collateral flushed during all prior flush events throughout  $I_f$ . We have  $\sum_{i=1}^k W_i = C$ .

## 26:14 Competitive Policies for Online Collateral Maintenance

If  $t_e \leq F$ , then OPT settles transaction value at most  $C$  in the interval  $(0, t_e]$ . In the same interval,  $A_\eta$  settles at least  $C - T$  since the uncommitted collateral is no more than  $T$ . Therefore,

$$\frac{V_{\text{OPT}}(I)}{V_{A_\eta}(I)} \leq \frac{C}{C - T} < \frac{C}{C - \eta C - T},$$

which would contradict our assumption.

Otherwise  $t_e - F > t_0$ . Observe that of the  $k$  portions,  $W_2, W_3, \dots$ , and  $W_k$  began settling transactions during  $I_f$  since if a portion  $W_i$ 's transaction activity before its last flush starts at a time before  $I_f$  for any  $i = 2, \dots, n$ , then  $W_{i-1}$ 's last flush time must also be before  $I_f$ . The only portion that may have started settling transactions before  $I_f$  is  $W_1$ . Since  $W_1$  has size equal  $\eta C$  and the uncommitted collateral in  $W_k$  is at most  $T$ ,  $V_{A_\eta}(I_f) \geq C - \eta C - T$ .

Again, we have  $V_{\text{OPT}}(I_f) \leq C$  since the length of  $I_f$  is  $F$ . Therefore,

$$\frac{V_{\text{OPT}}(I_f)}{V_{A_\eta}(I_f)} \leq \frac{C}{C - \eta C - T}.$$

This contradicts our initial assumption so we conclude that there is no such  $t_e$ .  $\blacktriangleleft$

**► Theorem 4.3.** *Let  $p \in (0, 1)$  and  $\tau > 0$  be a profit margin and flush cost. For a threshold  $\eta \in (0, 1]$  the algorithm  $A_\eta$  is  $\alpha$ -competitive in the  $\text{Gen}_{T;p}^{C;\tau}$  model for*

$$\alpha = \frac{1}{1 - \eta - T/C} \cdot \frac{p/\tau - 1/C}{p/\tau - 1/(\eta C)}.$$

**Proof.** For simplicity, assume that at the end of the sequence  $\text{Tx}$  any committed but unflushed collateral is flushed in both algorithms. Note then that the algorithm  $A_\eta$  flushed total collateral equal to the total settled value and, furthermore, that each flush processes exactly  $\eta C$  collateral with the exception of the last which may be smaller. It follows that the total number of flushes is exactly  $\lceil A_\eta[\text{Gen}_T^C; \text{Tx}] / (\eta C) \rceil$ . We conclude that

$$\begin{aligned} A_\eta[\text{Gen}_{T;p}^{C;\tau}; \text{Tx}] &= p \cdot A_\eta[\text{Gen}_T^C; \text{Tx}] - \tau \cdot \left\lceil \frac{A_\eta[\text{Gen}_T^C; \text{Tx}]}{\eta C} \right\rceil \\ &\geq p \cdot A_\eta[\text{Gen}_T^C; \text{Tx}] - \tau \cdot \left( \frac{A_\eta[\text{Gen}_T^C; \text{Tx}]}{\eta C} + 1 \right) \\ &= A_\eta[\text{Gen}_T^C; \text{Tx}] \left( p - \frac{\tau}{\eta C} \right) - O(1). \end{aligned} \quad (1)$$

OPT flushes at least once when it commits  $C$  collateral, therefore

$$\text{OPT}[\text{Gen}_{T;p}^{C;\tau}; \text{Tx}] \leq p \cdot \text{OPT}[\text{Gen}_T^C; \text{Tx}] - \tau \cdot \frac{\text{OPT}[\text{Gen}_T^C; \text{Tx}]}{C} = \text{OPT}[\text{Gen}_T^C; \text{Tx}] (p - \tau/C). \quad (2)$$

We combine these to conclude that

$$\begin{aligned} \text{OPT}[\text{Gen}_{T;p}^{C;\tau}; \text{Tx}] &\leq \text{OPT}[\text{Gen}_T^C; \text{Tx}] (p - \tau/C) \leq A_\eta[\text{Gen}_T^C; \text{Tx}] \frac{C}{C - \eta C - T} (p - \tau/C) \\ &\leq A_\eta[\text{Gen}_{T;p}^{C;\tau}; \text{Tx}] \frac{C}{C - \eta C - T} \cdot \frac{p - \tau/C}{p - \tau/(\eta C)} + O(1), \end{aligned}$$

as desired. The second inequality holds because of the inequality in Lemma 4.2.  $\blacktriangleleft$

**Optimal value of  $\eta$ .** The optimal value of  $\eta$  (which we denote  $\eta^*$ ) satisfies:

$$\frac{\partial}{\partial \eta} \left( \frac{1}{1 - \eta - T/C} \cdot \frac{p/\tau - 1/C}{p/\tau - 1/(\eta C)} \right) = 0,$$

which leads to the optimal value  $\eta^*$ , where  $\beta = \tau/(pC)$ :

$$\eta^* = \sqrt{(1 - T/C) \cdot \beta}.$$

Intuitively, as  $\beta$  approaches 0, the flush fee becomes negligible, and the algorithm should flush as often as possible. Using this optimal  $\eta^*$ , the competitive ratio is  $(1 - \beta)/(\sqrt{1 - T/C} - \sqrt{\beta})^2$ , which approaches 1 as  $\beta$  approaches 0. As a final result, we have the following theorem.

► **Theorem 4.4.** *Choosing  $\eta = \sqrt{\beta(1 - T/C)}$ , the competitive ratio for  $A_\eta$  is*

$$\frac{1 - \beta}{(\sqrt{1 - T/C} - \sqrt{\beta})^2}.$$

## 5 Conclusion

We constructed a modeling framework for collateral management policies of layer-two protocols in the blockchain setting. This framework targets two natural models encountered in practice: the  $k$ -wallet model in which the collateral  $C$  is divided among  $k$  wallets, and the general model in which  $C$  is viewed as one wallet (or collateral pool). We adopt the standard classical paradigm of competitive analysis in which an online algorithm  $A$ , that only knows the transactions encountered so far, is compared against an optimal algorithm  $OPT$  that has full knowledge of the transaction stream including future transactions. Our analysis is agnostic to transaction distribution and only requires knowing the maximum transaction size (i.e., value). Given the dynamic nature of blockchain applications and the unpredictable behavior of their transactions and workload, developing transaction distribution-independent techniques is highly desirable.

Using our framework, we study natural collateral management policies for the  $k$ -wallet and the general models, and we show how competitive they are compared to  $OPT$ . This is measured in terms of the total transaction value that can be settled and when to replenish the collateral to allow settling future transactions. The general model also studies the replenishment cost and how this affects the utility of the policy. We also derive the optimal configuration for the policy parameters, in terms of the number of wallets and the fraction of the committed collateral to be replenished.

To the best of our knowledge, this work is the first to study the collateral management problem for layer-two protocols. Our future work include extending this model to account for more factors, e.g., transaction specific conditions rather than just a transaction value, and develop dynamic policies in which the number of wallets, and even the collateral value itself, can change over time based on the experienced transaction stream.

---

## References

- 1 Filecoin. URL: <https://filecoin.io/>.
- 2 Golem. URL: <https://golem.network/>.
- 3 Livepeer. URL: <https://livepeer.com/>.
- 4 Uniswap protocol. <https://uniswap.org/>.

## 26:16 Competitive Policies for Online Collateral Maintenance

- 5 Ghada Almashaqbeh, Allison Bishop, and Justin Cappos. Microcash: Practical concurrent processing of micropayments. In *International Conference on Financial Cryptography and Data Security*, pages 227–244. Springer, 2020.
- 6 Manuel MT Chakravarty, Sandro Coretti, Matthias Fitz, Peter Gazi, Philipp Kant, Aggelos Kiayias, and Alexander Russell. Hydra: Fast isomorphic state channels. *Cryptology ePrint Archive*, 2020.
- 7 Alessandro Chiesa, Matthew Green, Jingcheng Liu, Peihan Miao, Ian Miers, and Pratyush Mishra. Decentralized anonymous micropayments. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 609–642. Springer, 2017.
- 8 Yoga Jaideep Darapuneni. A survey of classical and recent results in bin packing problem. *UNLV Theses, Dissertations, Professional Papers, and Capstones*, 2012.
- 9 Christian Decker and Roger Wattenhofer. A fast and scalable payment network with bitcoin duplex micropayment channels. In *Symposium on Self-Stabilizing Systems*, pages 3–18. Springer, 2015.
- 10 Stefan Dziembowski, Sebastian Faust, and Kristina Hostáková. General state channel networks. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 949–966, 2018.
- 11 Rundong Gan, Le Wang, Xiangyu Ruan, and Xiaodong Lin. Understanding flash-loan-based wash trading. In *Proceedings of the 4th ACM Conference on Advances in Financial Technologies*, pages 74–88, 2022.
- 12 Alex Gluchowski. Zk rollup: scaling with zero-knowledge proofs. *Matter Labs*, 2019.
- 13 Andrew Miller, Iddo Bentov, Surya Bakshi, Ranjit Kumaresan, and Patrick McCorry. Sprites and state channels: Payment networks that go faster than lightning. In *International conference on financial cryptography and data security*, pages 508–526. Springer, 2019.
- 14 Rafael Pass and Abhi Shelat. Micropayments for decentralized currencies. In *CCS*, pages 207–218. ACM, 2015.
- 15 Joseph Poon and Vitalik Buterin. Plasma: Scalable autonomous smart contracts. *White paper*, pages 1–47, 2017.
- 16 Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments. *Technical Report (draft)*, 2015.
- 17 Kanis Saengchote. Decentralized lending and its users: Insights from compound. *Journal of International Financial Markets, Institutions and Money*, 87:101807, 2023.
- 18 Steven S Seiden. On the online bin packing problem. *Journal of the ACM (JACM)*, 49(5):640–671, 2002.
- 19 Jiahua Xu, Krzysztof Paruch, Simon Cousaert, and Yebo Feng. Sok: Decentralized exchanges (dex) with automated market maker (amm) protocols. *ACM Computing Surveys*, 55(11):1–50, 2023.

# Thinking Fast and Slow: Data-Driven Adaptive DeFi Borrow-Lending Protocol

**Mahsa Bastankhah** ✉ 🏠   
Princeton University, NJ, USA

**Viraj Nadkarni** ✉ 🏠   
Princeton University, NJ, USA

**Chi Jin** ✉ 🏠   
Princeton University, NJ, USA

**Sanjeev Kulkarni** ✉ 🏠   
Princeton University, NJ, USA

**Pramod Viswanath** ✉ 🏠   
Princeton University, NJ, USA

---

## Abstract

Decentralized finance (DeFi) borrowing and lending platforms are crucial to the decentralized economy, involving two main participants: lenders who provide assets for interest and borrowers who offer collateral exceeding their debt and pay interest. Collateral volatility necessitates over-collateralization to protect lenders and ensure competitive returns. Traditional DeFi platforms use a fixed interest rate curve based on the utilization rate (the fraction of available assets borrowed) and determine over-collateralization offline through simulations to manage risk. This method doesn't adapt well to dynamic market changes, such as price fluctuations and evolving user needs, often resulting in losses for lenders or borrowers. In this paper, we introduce an adaptive, data-driven protocol for DeFi borrowing and lending. Our approach includes a high-frequency controller that dynamically adjusts interest rates to maintain market stability and competitiveness with external markets. Unlike traditional protocols, which rely on user reactions and often adjust slowly, our controller uses a learning-based algorithm to quickly find optimal interest rates, reducing the opportunity cost for users during periods of misalignment with external rates. Additionally, we use a low-frequency planner that analyzes user behavior to set an optimal over-collateralization ratio, balancing risk reduction with profit maximization over the long term. This dual approach is essential for adaptive markets: the short-term component maintains market stability, preventing exploitation, while the long-term planner optimizes market parameters to enhance profitability and reduce risks. We provide theoretical guarantees on the convergence rates and adversarial robustness of the short-term component and the long-term effectiveness of our protocol. Empirical validation confirms our protocol's theoretical benefits.

**2012 ACM Subject Classification** Theory of computation → Market equilibria

**Keywords and phrases** Defi borrow-lending, adaptive market design, decentralized finance

**Digital Object Identifier** 10.4230/LIPIcs.AFT.2024.27

**Related Version** *Full Version*: <https://arxiv.org/pdf/2407.10890> [6]

**Funding** This work was supported by the National Science Foundation via grants CNS-2325477, the Army Research Office via grant W911NF2310147, a grant from C3.AI, a SEAS Innovation grant from Princeton University and a gift from XinFin Private Limited.



© Mahsa Bastankhah, Viraj Nadkarni, Chi Jin, Sanjeev Kulkarni, and Pramod Viswanath;  
licensed under Creative Commons License CC-BY 4.0

6th Conference on Advances in Financial Technologies (AFT 2024).

Editors: Rainer Böhme and Lucianna Kiffer; Article No. 27; pp. 27:1–27:23

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1 Introduction

Decentralized Finance (DeFi) has revolutionized lending and borrowing by eliminating centralized intermediaries. The main paradigm shift has been around moving away from opaque financial entities such as banks, that use proprietary models and data to match deposits with borrowers [31], to transparent pools with published algorithms to change interest rates, and borrowing conditions. Major DeFi lending platforms like Aave [2] and Compound [14] function through these liquidity pools, where lenders provide capital that borrowers can access. These protocols ensure that borrowers pledge enough collateral to cover their debt, along with an additional safety buffer.

The simplest variable of interest that any lending protocol seeks to control is the supply-demand ratio of the pool, referred to as “utilization.” The objective is to maintain a stable utilization around a designated “optimal utilization” threshold. A coarse rule of thumb is that when the utilization is low, interest rates remain low to encourage borrowing [5]. As utilization increases, interest rates rise to balance demand with supply and to prevent excessively high utilization, which could restrict lenders’ ability to withdraw their funds, thus rendering the market less attractive.

Besides the interest rate, other parameters like the over-collateralization ratio, also known as the “collateral factor,” govern the long-term risks and profits of the market [30]. In particular, the cash flow that any lender gets from the protocol is at risk of liquidation, and in the more severe cases, default. This risk can be minimized by demanding a large amount of collateral from borrowers, which makes the risk vanishingly small while making the lending market incredibly inefficient and unattractive for borrowers, especially if the asset used as the collateral does not suffer frequent price fluctuations. Thus, the collateral factor needs to be determined based on a careful analysis of recent historic behavior of the collateral asset price, and the risk appetite of the lender.

Present DeFi platforms fix the interest rate as a static function of the utilization [3, 15]. Choosing utilization as the primary indicator of both supply/demand dynamics and market risk/attractiveness and employing a fixed interest rate curve to manage these aspects is very arbitrary and manually determined. Furthermore, traditional DeFi borrowing and lending markets set the collateral factor through a comprehensive process involving community proposals and review phases [1, 13]. However, this method is notably slow and struggles to adapt quickly to rapid market changes, potentially leading to losses and excessive risks due to the delayed adjustment of parameters in response to market fluctuations and experiencing long periods of extreme low liquidity or market inefficiencies. For instance, the authors of [21] have found that the markets for DAI and USDC frequently exhibit periods of extreme low liquidity with utilization exceeding 80% and 90%, respectively, which further highlights the inadequacy of current interest rate models.

In this study, we propose an adaptive, automated, and data-driven approach for designing a borrowing/lending protocol. We begin by modeling the behaviors of borrowers and lenders based on their incentives in a principled manner and examine how external factors alter these behaviors over time. We then define market equilibrium (Definition 2), where the market remains stable within a broader external market, and rates offered by our protocol do not allow borrowers or lenders to gain an advantage over external market rates. Achieving equilibrium is crucial in a market with conflicting interests, as instability tends to disproportionately benefit one group over another, reducing overall fairness and attractiveness of the market [23]. If a protocol cannot dynamically adjust to achieve equilibrium, it risks losing liquidity and users. Market stability must be promptly restored after disruptions, which may be caused by changes in external market conditions or shifts in price distributions that affect the market’s risk and profit structure.

Traditional stationary-curve borrowing/lending markets depend on user interactions to push towards equilibrium – for instance, high interest rates prompt borrowers to repay loans, reducing utilization and interest rates. However, this process is slow and often results in impermanent loss [36], especially for users with less flexibility in managing their assets. In order to address this problem, our protocol includes an “interest rate controller” submodule that learns the equilibrium interest rate from user behaviors, providing a faster convergence rate, even in the presence of uninformed users who lack precise information on competitive rates. Unlike traditional methods, our approach does not solely rely on user actions but actively learns from them to accurately assess and adapt to market conditions. Moreover, we provide an adversarial robust version of our interest rate controller as well which learns the equilibrium interest rate as long as the adversary controls less than 50% of the borrowing demand.

In addition to promptly restoring stability following market disruptions, it is essential for a protocol to adaptively optimize hyperparameters that enhance long-term system efficiency and manage risk. Our protocol features a long-term planner that uses the collateralization ratio as a control variable to adapt to market changes and stabilize the market at a desired level (Section 3.2). The collateralization ratio is critical for managing long-term risks and rewards in the system. This ratio has a complex relationship with user behavior and the overall risk and profitability of the market, which we explore in detail in our paper (Section 4). The objectives of the protocol within this long-term planner can be defined in many different ways; In this paper our focus is on maintaining long-term utilization at a target level and controlling default, however more complex objective functions could be implemented to address specific market needs or objectives. Our approach provides a general framework for designing adaptive markets with heterogeneous users who may have varying incentives.

Additionally, we implemented our protocol and tested it with simulated borrowers and lenders, empirically comparing its performance against fixed-curve baselines. We evaluated the correctness of our theoretical guarantees in practice and demonstrated that our interest rate controller can quickly learn the equilibrium interest rate after each market disruption, regardless of borrowers’ and lenders’ elasticity. In contrast, the baseline protocol fails to find the equilibrium interest rate when user elasticity is low due to its reliance on user reactions to push the market toward equilibrium. Moreover, we showed that in the presence of major market changes, our protocol’s collateral factor planner adaptively activates. By learning new price and market parameters, it sets the collateral factor to maintain utilization near a predefined optimal level in the long term.

## Related work

Various models on lender and borrower behavior and their equilibria have been explored. [12] assume parametrized supply and demand curves based on interest rates, approximating the curve around equilibrium to recommend rates, but they ignore external markets and default risk minimization. [33] consider external markets, measuring protocol efficiency by interest rate differences, but their models lack long-term decision-making and liquidation considerations. [10] examines Nash equilibrium in a model with independent quality shocks, showing that exogenous asset prices yield one equilibrium, while protocol-influenced prices cause oscillation and propose ad-hoc contract adjustments. Empirical studies on lender/borrower behavior [19, 20, 34] inform our parameter values. [35] discusses borrower trading strategies with market makers. Adversarial attacks on lending protocols have also been highlighted [9, 11, 8].

Several recent works in the mechanism design of financial systems have been advocating for the use of automated adaptivity [40]. The presence of impermanent loss and arbitrage loss in the design of market makers has also spawned multiple works in adaptive market making [18, 27, 29]. We seek to bring similar automated methods to lending in DeFi. Platforms such as Morpho [28] and Ajna [4] provide lenders and borrowers more flexibility when it comes to equilibrium interest rate discovery via an order-book like structure. However, such protocols require constant monitoring on the part of the participants for fairness and optimality. Our objective is to bring these notions of fairness/optimality to more passive pool-based lending protocols.

The methods used in this work are based on optimal control and filtering literature using the least squares method [22]. This method has been used in the estimation of underlying dynamics, given a noisy access to measurements [25, 39]. Several recent works have provided extensions of this algorithm to ensure adversarial robustness [7, 37, 26], which use hard thresholding and concentration inequalities to weed out adversarial data.

## 2 Problem formulation

### 2.1 Market actors

The DeFi borrowing and lending market includes four key participants: lenders, borrowers, liquidators, and the protocol, here called  $\mathcal{P}$ . These actors interact within a shared pool. This subsection briefly clarifies each participant’s role and the mechanisms protocols use to regulate their interactions.

To prevent defaults during price declines, the protocol employs *liquidation*. This occurs when a borrower’s *loan-to-value* ratio (debt-to-collateral value) exceeds a threshold liquidation threshold ( $LT$ ), set between 0 and 1 and higher than the initial loan-to-value ratio  $c$ . When this threshold is surpassed, liquidators can claim a portion of the borrower’s collateral to repay the debt, reducing the loan-to-value ratio. Liquidators receive a fee  $LI$  from the borrower’s collateral. Liquidations enhance system safety but are unfavorable for borrowers due to the incentive fee. This prompts borrowers to increase their collateral preemptively. Despite liquidation mechanisms, defaults can occur if collateral prices drop abruptly or if liquidators lack sufficient incentives to act.

The protocol must adjust parameters  $\{r_t, c_t, LT_t, LI_t\}$  over time to stabilize the pool. Objectives include stabilizing loan supply and demand by setting an interest rate  $r_t$  and optimizing parameters to minimize defaults and liquidations while maintaining an ideal utilization rate. Our paper focuses on creating a competitive DeFi protocol with efficient rates, not on revenue maximization. The openness of DeFi protocols and minimal fees should ensure that the most competitive protocol eventually dominates the market.

The lending pool consists of two assets: a stable asset,  $\mathcal{A}_l$ , provided by lenders for interest, and a volatile asset,  $\mathcal{A}_c$ , used by borrowers as collateral. Borrowers can only borrow a fraction collateral factor ( $c$ ) of their collateral, set by the protocol. At timeslot  $t$ , the overall pool’s assets of type  $\mathcal{A}_l$ , considering both lent-out funds and available liquidity, are denoted by  $L_t$ . Note that  $L_t$  increases over time as lenders accrue interest on the lent-out portion. The asset of a particular lender  $i$  is represented by  $L_t(i)$ .

The interest rate  $r_t$  is set by the protocol at each block. Borrowers pay this rate, but lenders earn interest only on the utilized fraction  $U_t$  of their deposit, defined as  $U_t = \frac{B_t}{L_t}$ , where  $B_t$  represents the overall debt across all borrowers, and  $B_t(i)$  represents the debt of borrower  $i$ . The debt amount also increases over time due to accrued interest. The quantity of the overall collateral posted by all borrowers is denoted by  $C_t$ , and  $C_t(i)$  denotes the



collateral of borrower  $i$ . Hence,  $p_t \cdot C_t$  determines the value of the collateral in terms of the lent-out asset,  $\mathcal{A}_l$  (for a thorough list of the notations and their description refer to Appendix D of the full paper [6]). When borrowers repay, they return the loan plus interest and retrieve their collateral. Lenders receive interest based on the protocol rate and fund utilization. Defaults can affect the final interest rate for lenders. If collateral value drops below the debt, the protocol cannot compel repayment of the insolvent debt, resulting in a loss that impacts the lenders' interest rate.

## 2.2 Environment model

**Asset price model.** We operate within discrete time intervals, denoted as  $\Delta$ , each corresponding to one blocktime. We use a discrete price model to monitor the collateral asset's price from one block to the next. For simplicity, we assume the lent-out asset is a stablecoin with a relatively stable price, while the collateral asset's price follows an exogenous geometric Brownian motion with volatility  $\sigma$ . We assume constant volatility over short periods of time, with occasional sporadic jumps, but no fluctuations from one timeslot to the next. In particular, we assume that the price volatility is constant within timescales denoted by  $T_m > 1$  ( $m$  for market, denoting the timeframe within which the market is stable) which consists of multiple timeslots and can change arbitrarily every  $T_m$  timeslots.

The price at time  $t$ , denoted as  $p_t$ , follows a Geometric Brownian motion with drift  $\mu_{\text{price}}$  and volatility  $\sigma$ . The initial price  $p_0$  is the starting point, for notation simplicity we normalize and consider  $\Delta = 1$  and hence formally, the model is:

$$p_t = p_{t-1} \exp(\mu + \sigma \varepsilon_t), \quad \varepsilon_t \sim \mathcal{N}(0, 1) \quad (1)$$

where  $\varepsilon_t$  is the innovation term for the volatility.

**External market competition.** We assume the existence of an external competitor market which offers risk-free borrow rate  $r_o^b$  and risk-free lend rate  $r_o^l$ . These rates are constant during each market period  $T_m$  and can change arbitrarily between periods. This assumption accounts for the competition and the broader market within which our protocol operates.  $r_o^l$  and  $r_o^b$  might represent the existence of complex alternatives rather than simple risk-free rates. In Appendix B of the full paper [6], we discuss interpreting these parameters based on real-world strategies and competitors in the Defi ecosystem. Throughout the paper, we abstract these concepts into  $r_o^l$  and  $r_o^b$ .

## 2.3 Protocol behaviour and pool logic

In this section, we establish the structure of a decentralized borrowing-lending protocol, denoted by  $\mathcal{P}$ . The protocol fulfills two primary roles: 1)  $\mathcal{P}$  sets the pool's parameters for each block, denoted as  $\{r_t, c_t, LT_t, LI_t\}$ , by transmitting a transaction to the underlying blockchain. These parameters govern the pool's logic. 2)  $\mathcal{P}$  updates the state variables  $L_t$ ,  $B_t$ , and  $C_t$  every block to apply interest rate accumulation and liquidation or default due to price fluctuations.

**Handling default.** At the beginning of each timeslot  $t$ ,  $\mathcal{P}$  receives the latest price of  $\mathcal{A}_c$ ,  $p_t$ , from an oracle. The protocol calculates potential defaults accrued in the last timeslot for each user. The default for borrower  $i$  at timeslot  $[t-1, t]$  is:

$$\pi_{t-1}^i(p_t) := \max\{0, B_{t-1}(i) - C_{t-1}(i) \cdot p_t\} \quad (2)$$

## 27:6 Data-Driven Adaptive DeFi Borrow-Lending Protocol

The overall default, normalized by  $L_{t-1}$ , is:

$$\pi_{t-1}(p_t) := \frac{1}{L_{t-1}} \sum_{i \in \text{borrowers}} \max\{0, B_{t-1}(i) - C_{t-1}(i) \cdot p_t\} \quad (3)$$

The protocol seizes the remaining collateral of defaulted positions, exchanges it for  $\mathcal{A}_l$ , and sets the debt of defaulted borrowers to zero. The gained  $\mathcal{A}_l$  assets are added back to the pool. Moreover the underwater debt is deduced from the lender's deposit accordingly:

$$L_t(i) = L_{t-1}(i) - \pi_{t-1}(p_t) \cdot L_{t-1}(i), \quad \forall i \in \text{Lenders} \quad (4)$$

For a more thorough explanation of why we handle defaults in this way rather than using a safety reserve similar to most of the current working Defi borrow-lending platforms refer to Appendix A of the full paper [6].

**Interest update.** The debt of non-defaulted borrowers is updated by:

$$B_t(i) = B_{t-1}(i) \cdot (1 + r_{t-1}), \quad \forall i \in \text{Borrowers}$$

The interest rate on the utilized portion of the pool applies to the lenders as well:

$$L_t(i) = L_{t-1}(i) \cdot \left(1 + r_{t-1} \frac{B_{t-1}}{L_{t-1}}\right), \quad \forall i \in \text{Lenders}$$

**Liquidation.**  $\mathcal{P}$  tracks borrow positions exceeding the liquidation threshold. A position  $i$  is eligible for liquidation if  $LT_{t-1} < \frac{B_t(i)}{C_t(i) \cdot p_t} < 1$ . Liquidators reduce the user's debt by purchasing collateral, restoring the loan-to-value ratio below  $LT_{t-1}$ . The debt and collateral after liquidation are updated accordingly. The minimum liquidation amount that reduces the user's loan-to-value below  $LT_{t-1}$  is

$$\lambda_{t-1}^i(p_t) := \max\left\{0, \frac{B_t(i) - LT_{t-1} \cdot C_t(i) \cdot p_t}{1 - LT_{t-1}(1 + LI_{t-1})}\right\}$$

**Setting new parameters.**  $\mathcal{P}$  sets new parameters for the next timeslot:  $\{r_t, c_t, LT_t, LI_t\}$ . These parameters determine the interest rate, maximum loan-to-value, and liquidation parameters for the next timeslot.

**Admitting new users.** The protocol accepts all new lend and repay requests. Borrow requests are accepted only if they adhere to the maximum collateralization factor  $c_t$  and there is sufficient  $\mathcal{A}_l$  in the pool to satisfy the request. Additionally, it processes withdrawal requests from lenders as long as the withdrawal amount does not exceed the available  $\mathcal{A}_l$  in the pool.

### 2.4 User behavior model

In this section, we establish the model that a rational user would use to interact with the protocol. We consider a continuum of lenders and borrowers, each controlling a single unit of demand or supply. In each timeslot, lenders can deposit or withdraw their unit, and borrowers can borrow, repay, or adjust their loan-to-value ratio by sending a transaction to the smart contract.

### 2.4.1 Lender

We assume that a continuum lender with one unit of supply, planning for the next timeslot, will calculate the following utility function at time  $t$  to decide whether to deposit into the pool or, if already deposited, to withdraw and invest in another external alternative offering  $r_o^l$ .

$$\text{Utility}_t^l := r_t U_t - \mathbb{E}[\pi_t(p_{t+1})] - r_o^l \quad (5)$$

where the expectation is over the price at timeslot  $t + 1$  price. The first term in 5 represents the interest earned by the lender on the utilized portion of their deposit. Although the interest rate is compounded, we approximate it linearly for simplicity. The second term denotes the normalized defaulted debt deducted from the lender's deposit. Finally, we subtract the external interest rate  $r_o^l$  to account for the lender's opportunity cost.

We now describe the dynamics by which lenders add or withdraw their deposits based on utility. We introduce a new parameter  $\eta_l$ , which reflects the average elasticity of lenders at time  $t$ . This value can change over time (not faster than  $T_m$ ) and is unknown to the protocol. We assume that the relative rate at which lenders deposit or withdraw from the pool is governed by the following model:

$$\begin{aligned} \frac{L_{t+1} - L_t}{L_t} &= \eta_l \cdot \text{Utility}_t^l + \varepsilon_t \\ &= \eta_l \cdot (r_t U_t - \mathbb{E}[\pi_t(p_{t+1})] - r_o^l) + \varepsilon_t, \quad \varepsilon_t \stackrel{\text{i.i.d.}}{\sim} \mathcal{N}(0, \zeta^2) \end{aligned} \quad (6)$$

The noise term accounts for the behavior of uninformed or less informed users.

### 2.4.2 Borrower

From observing the real lending markets, we identify two types of DeFi borrowers, financing and leveraged trading borrowers (see Appendix B.2 of the full paper [6] for more details).

The first type borrows an asset to use elsewhere, gaining value by leveraging it, e.g., for yield farming or real-world purposes. This group's value from the borrowed asset is represented as  $r_o^b$ , measured as an interest rate. The utility function of a borrower of this type controlling one unit of demand, considering the opportunity cost of locked collateral, is

$$\begin{aligned} \text{Utility}_t^{b,1} &:= r_o^b - r_t + \mathbb{E}[\pi_t^i(p_{t+1})] - \mathbb{E}[\lambda_t^i(p_{t+1})] \cdot LI_t \\ &\quad + C_t(i) \cdot \mathbb{E}[\mathbb{1}_{p_{t+1}-p_t < 0}(p_{t+1} - p_t)] \end{aligned} \quad (7)$$

This includes inherent value ( $r_o^b$ ), interest rate ( $-r_t$ ), default value ( $\mathbb{E}[\pi_t^i(p_{t+1})]$ ), liquidation cost ( $-\mathbb{E}[\lambda_t^i(p_{t+1})] \cdot LI_t$ ), and opportunity cost of locked collateral  $\mathbb{E}[\mathbb{1}_{p_{t+1}-p_t < 0}(p_{t+1} - p_t)]$ .

The second type aims to take a long position on  $\mathcal{A}_c$ . They borrow  $\frac{1}{p_t}$  units of  $\mathcal{A}_c$  from an external provider  $\mathcal{Z}$ , add more collateral to meet the over-collateralization requirement  $c_t$ , and borrow one unit of  $\mathcal{A}_l$  from  $\mathcal{P}$ , then exchange it for  $\mathcal{A}_c$  to repay  $\mathcal{Z}$ . This borrowing strategy is studied in details in [35]. Their utility function is:

$$\text{Utility}_t^{b,2} = -r_t + \mathbb{E}[\pi_t^i(p_{t+1})] - \mathbb{E}[\lambda_t^i(p_{t+1})] \cdot LI_t + \mathbb{E}\left[\frac{p_{t+1} - p_t}{p_t}\right] \quad (8)$$

This includes interest rate ( $-r_t$ ), default value ( $\mathbb{E}[\pi_t^i(p_{t+1})]$ ), liquidation cost ( $-\mathbb{E}[\lambda_t^i(p_{t+1})] \cdot LI_t$ ), and gain from a price change of the  $\frac{1}{p_t}$  investment in  $\mathcal{A}_c$  which was possible through interacting with  $\mathcal{P}$  i.e.,  $\mathbb{E}\left[\frac{p_{t+1} - p_t}{p_t}\right]$ . Refer to Appendix B.2 of the full paper [6] for more details.

Assuming  $\alpha$  fraction of borrowers are type 1 and the rest type 2, the rate of borrowing or repaying is a linear function of borrower's elasticity  $\eta_b$  and their average utility plus noise:

$$\begin{aligned} \frac{B_{t+1} - B_t}{B_t} &= \eta_b \cdot (\alpha \cdot \text{Utility}_t^{b,1} + (1 - \alpha) \cdot \text{Utility}_t^{b,2}) + \varepsilon_t \\ &= \eta_b \cdot \left( \alpha \cdot r_o^b - r_t + \mathbb{E}[\pi_t^i(p_{t+1})] - \mathbb{E}[\lambda_t^i(p_{t+1})] \cdot LI_t \right. \\ &\quad \left. + \alpha \cdot C_t(i) \mathbb{E}[\mathbb{1}_{p_{t+1} - p_t < 0}(p_{t+1} - p_t)] + (1 - \alpha) \cdot \mathbb{E}\left[\frac{p_{t+1} - p_t}{p_t}\right] \right) + \varepsilon_t, \end{aligned} \quad (9)$$

where  $\varepsilon_t \stackrel{\text{i.i.d}}{\sim} \mathcal{N}(0, \zeta^2)$ .

Throughout this paper, we assume that  $\mathcal{P}$  efficiently sets the buffer between the collateral factor and the liquidation threshold to ensure that the expected liquidation,  $\mathbb{E}[\lambda_t^i(p_{t+1})]$ , for a borrower  $i$  who maintains the posted collateral factor, is negligible. This assumption differs from the current borrowing and lending platforms, which experience significant liquidations even among borrowers who adhere to the posted collateral factor. However, we believe that a competent borrowing and lending protocol should set risk parameters to minimize this risk. In Section 3.2, we explain how we determine the liquidation threshold and collateral factor to ensure that the expected liquidations remain negligible. Moreover, throughout this paper, we assume that the liquidation incentive,  $LI_t$ , is set sufficiently high by the protocol to encourage the borrowers to maintain the collateral factor,  $c_t$ , and avoid liquidations.

- **Lemma 1** (Maximum loan-to-value adoption). *Consider the following conditions:*
- *The collateral factor,  $c_t$ , and the liquidation threshold,  $LI_t$ , are chosen such that for a given  $LI_t$ ,  $c_t$  is the maximum collateral factor that ensures the expected liquidation,  $\mathbb{E}[\lambda_t^i]$ , is approximately zero for a user  $i$  who maintains  $c_t$ .*
  - *The liquidation incentive,  $LI_t$ , is set high enough to incentivize rational borrowers to avoid liquidation by ensuring that  $\frac{B_t(i)}{C_t(i)p_t} \leq c_t$ .*

*Then rational borrowers will adopt the maximum loan to value allowed by the protocol i.e.,  $c_t$ .*

Hence from now on, we assume that  $\mathbb{E}[\lambda_t^i]$  for a rational continuum borrower is negligible and for ease of notation, we denote it by  $\lambda(c_t, LI_t)$ . The proof of all the lemmas and theorems can be found in Appendix C of the full paper [6].

### 2.4.3 Liquidator

We assume that the liquidation incentive  $LI$  is set at a level that consistently incentivizes liquidators, ensuring their prompt engagement and immediate liquidation up to the limit allowed by  $\mathcal{P}$ . Additionally, since we use an exogenous price model for collateral, phenomena like liquidation spirals studied in previous works [24] are not considered in our analysis.

## 2.5 Equilibrium analysis

In this section, we will formally define the concept of equilibrium in the borrow-lending framework we established. And we will analytically identify the set of equilibria of this market when users follow the behaviour outlined in 2.4.

► **Definition 2** (Market equilibrium). *A lending pool governed by protocol  $\mathcal{P}$  parameterized by  $\{r_t, c_t, LI_t, LI_t\}$ , and lender's and borrower's behavior respectively governed by 6, and 9 is in equilibrium if and only if:*

$$\mathbb{E}\left[\frac{B_{t+1} - B_t}{B_t}\right] = 0 \quad \text{and} \quad \mathbb{E}\left[\frac{L_{t+1} - L_t}{L_t}\right] = 0$$

*where the expectation is over the noise term in the user behavior model.*

► **Lemma 3** (Simplified default and price change terms). *In the presence of rational continuum lenders and rational continuum borrowers who follow collateral factor  $c_t$  (due to Lemma 1) we have:*

$$\begin{aligned}\pi(c_t) &:= \mathbb{E} [\pi_t^i(p_{t+1})] \\ &= \Phi\left(\frac{\log(c_t) - \mu}{\sigma}\right) - \frac{\exp(\frac{\sigma^2}{2} + \mu)}{c_t} \cdot \Phi\left(\frac{-\mu + \log(c_t) - \sigma^2}{\sigma}\right)\end{aligned}\quad (10)$$

$$\mathbb{E}[\pi_t(p_{t+1})] = U_t \pi(c_t) \quad (11)$$

$$C_t(i) \mathbb{E}[\mathbb{1}_{p_{t+T_u} - p_t < 0} (p_{t+T_u} - p_t)] = \frac{1}{c_t} \left( e^{\mu + \frac{\sigma^2}{2}} \frac{\Phi\left(\frac{-\mu - \sigma^2}{\sigma}\right)}{\Phi\left(\frac{-\mu}{\sigma}\right)} - 1 \right) \quad (12)$$

$$\mathbb{E}\left[\frac{p_{t+1} - p_t}{p_t}\right] = e^{\mu + \frac{\sigma^2}{2}} - 1$$

Where  $\phi(\cdot)$  and  $\Phi(\cdot)$  denote the PDF and CDF, respectively, of the standard normal distribution.

Throughout the rest of the paper, we denote the expected default for one unit of debt by  $\pi(c_t)$ . Conceptually, Lemma 3 implies that if the pool's loan-to-value ratio is maintained close to  $c_t$  in each timeslot, the probability of default remains memoryless. This is due to the Brownian motion of price, where the price ratio between consecutive timeslots follows a stationary distribution. Therefore, the probability that the next timeslot's price falls below the default threshold is stationary and memoryless, changing only when the price distribution itself changes.

► **Theorem 4.** *Let the lender behavior be described by Equation 6, the borrower behavior by Equation 9 (for  $\eta_b > 0$ ), and the assumptions of Lemma 1 hold. A protocol with parameters  $\{r_t, c_t, LI_t, LT_t\}$ , achieves a non-trivial equilibrium if and only if  $r_t = r^*$ :*

$$r^* = \alpha r_o^b + \pi(c_t) + \frac{\alpha}{c_t} \left( e^{\mu + \frac{\sigma^2}{2}} \frac{\Phi\left(\frac{-\mu - \sigma^2}{\sigma}\right)}{\Phi\left(\frac{-\mu}{\sigma}\right)} - 1 \right) + (1 - \alpha) \left( e^{\mu + \frac{\sigma^2}{2}} - 1 \right) \quad (13)$$

Furthermore, if  $\eta_l > 0$ , then the unique equilibrium utilization is:

$$U^* = \begin{cases} \frac{r_o^l}{r^* - \pi(c_t)} & \text{if } r^* > \pi(c_t) \\ 1 & \text{otherwise} \end{cases} \quad (14)$$

**Equilibrium dynamics.** In order to achieve the equilibrium point, The protocol first should find the equilibrium interest rate,  $r^*$ , and set  $r_t = r^*$  to prevent borrowers from leaving or joining the system. Once  $r^*$  is set, if the utilization  $U_t$  is above  $U^*$ , lenders' utility is positive, so they keep lending until  $U_t$  reaches  $U^*$ , stabilizing the system. Conversely, if  $U_t$  is below  $U^*$ , lenders' utility is negative, causing them to withdraw until  $U_t = U^*$ . At this point, lenders are indifferent between the pool and external competitors and remain fixed. The equilibrium utilization is  $U^* = 1$  if, regardless of utilization, lenders' utility under  $r_t = r^*$  remains non-positive. Consequently, they withdraw fully, yielding  $U^* = 1$ , and they remain trapped in an unfavorable equilibrium where they prefer external rates but since their fund is being borrowed, they cannot leave the system.

## 2.6 Protocol objectives and evaluation metrics

In this section, we define three key metrics to evaluate a borrow-lending protocol. The protocols considered follow the behavior outlined in 2.3. Each protocol selects a deterministic or randomized interest rate function  $r_t : \mathcal{H}_1^t \rightarrow \mathbb{R}^+$ , a collateral factor function  $c_t : \mathcal{H}_1^t \rightarrow [0, 1]$  and a liquidation threshold function  $LI_t : \mathcal{H}_1^t \rightarrow [0, 1]$  mapping the pool's history to an interest rate, a collateral factor and a liquidation threshold.

### 2.6.1 Rate of equilibrium convergence

Achieving market equilibrium is crucial for any DeFi application, as equilibrium points represent the market's most competitive state. At equilibrium, no participant is overpaid or underpaid, ensuring all users are equally satisfied with  $\mathcal{P}$  as with any external alternative. In two-sided markets like borrow-lending, the time to achieve stability can disproportionately benefit one side. Setting interest rates below what borrowers are willing to pay results in lenders receiving less than in a competitive market. Conversely, if interest rates exceed the equilibrium rate, borrowers pay more than in a stable market. The more elastic user benefits at the expense of the less elastic user, who incurs an impermanent loss.

Adapting to market changes allows the protocol to stabilize the pool when user behavior or price volatility changes. As these parameters change over time, the protocol must respond dynamically to maintain system stability within a reasonable timeframe. In the borrow-lending market framework, each time user behavior parameters (e.g.,  $r_o^b, r_o^l, \alpha$ ) or price volatility ( $\sigma$ ) change, the market stabilizes at a new  $r^*$  (refer to Theorem 4). One objective of  $\mathcal{P}$  is to rapidly identify and set this new equilibrium rate. We formalize this concept as the rate of convergence, using it as a metric to evaluate our protocol against non-learning baselines.

► **Definition 5** (Rate of equilibrium convergence). *Consider a stable borrow-lending pool with the interest rate  $r_t = r^*$ , where  $r^*$  is the equilibrium interest rate determined by Equation 13. At time  $t + 1$ , user behavior models adapt to new parameters  $\{\bar{\eta}_l, \bar{r}_o^l, \bar{\eta}_b, \bar{r}_o^b, \alpha\}$ , and price volatility changes to  $\bar{\sigma}$ . Let  $r_\tau$  represent the interest rate set by  $\mathcal{P}$  at time  $\tau > t$ , and let  $\bar{r}^*$  denote the new equilibrium interest rate for the updated market parameters. We define the rate of equilibrium convergence of  $\mathcal{P}$ , denoted by  $\mathcal{R}_{\mathcal{P}}(\delta, \tau)$ , as the infimum of functions  $f(\delta, \tau)$ , such that there exists some  $T(\delta)$  for which, with probability  $1 - \delta$ ,*

$$|r_\tau - \bar{r}^*| < f(\delta, \tau), \quad \forall \tau > T(\delta),$$

*for any initial and secondary set of parameters in the user model and price model. Probabilities are calculated on the randomness of the protocol and the noise in the user behavior model.*

### 2.6.2 Equilibrium Optimality Index

Any Defi borrow-lending market aims to meet specific long-term objectives. For instance, the pool should maintain utilization at an optimal level; Because low utilization reduces capital efficiency, requiring the protocol to pay higher interest rates to lenders, and high utilization can make it difficult for lenders to withdraw and borrowers to secure loans. Moreover, expected defaults and liquidations are risk metrics that the protocol aims to control. These objectives operate on a different timescale than the interest rate adjustments discussed in the convergence rate. For instance, we care about the average utilization or expected default over a long period rather than their local values in each timeslot. To address this, we define a metric called optimality index, which evaluates the desirability of the system's equilibriums during the timeslots when the pool has reached its equilibrium, denoting the set of these timeslots by  $\mathcal{T}_e$ .

► **Definition 6** (Optimality Index). *The Optimality Index of a protocol, denoted by  $OI_{\mathcal{P}}$ , is defined as follows:*

$$OI_{\mathcal{P}} := \frac{1}{|\mathcal{T}_e|} \sum_{t \in \mathcal{T}_e} \mathbb{E} \left[ -(U_t - U_{opt})^2 - \gamma (U_t \pi(c_t) + \lambda(c_t, LT_t)) \right], \quad (15)$$

*The expectation is taken over the protocol's randomness and user behavior.  $U_{opt}$  and  $\lambda$  are constant parameters.*

While the first term in the  $OI_{\mathcal{P}}$  ensures that utilization is kept near a desired point, the second term acts as a regularization term, controlling the expected default and liquidation. Our definition of Optimality Index evaluates a specific notion of optimality, but it is just one of many possible definitions. Our protocol design methodology can be applied to other objective functions beyond Function 15. In this paper, we showcase our ideas for this specific objective function and discuss how to extend the methodology to other objectives.

### 2.6.3 Adversarial robustness

Protocols in DeFi are always susceptible to adversarial behavior that can be used to manipulate an adaptive algorithm to respond in a suboptimal manner. This behavior is usually observed when some borrower/lender agents interact with protocols outside of  $\mathcal{P}$  in conjunction with  $\mathcal{P}$  to achieve a profit. This can involve oracle manipulations attacks used to run away with valuable assets while providing worthless collateral, or attacks that move interest rates in the opposite direction of the equilibrium rates. We focus on the latter type of adversarial behavior in this work. Moving interest rates away from equilibrium leads to market inefficiencies. Further, undue hikes in these rates can be used to trigger unexpected liquidations for profit.

We thus propose that the susceptibility of  $\mathcal{P}$  to adversarial manipulation should also be measured. To do that, we first assume that a fraction  $\beta$  of the population of lenders/borrowers are adversarial. Thus,  $\mathcal{P}$  will face an adversarial lender/borrower for an approximately  $\beta$  fraction of time slots. In those time slots, the adversary can manipulate the borrow/lend reserves arbitrarily. Let  $T_{\beta}$  denote the set of time slots that the protocol faces an adversary.

We measure how susceptible a protocol is to adversarial manipulation using the following metric.

► **Definition 7** (Adversarial Susceptibility). *The adversarial susceptibility of a protocol, denoted by  $AS_{\mathcal{P}}$ , is defined as follows:*

$$AS_{\mathcal{P}} := \mathbb{E} \left[ \sum_{t \in \mathcal{T}_e} r_{\mathcal{P}}^t - r_{\mathcal{P}|T_{\beta}}^t \right].$$

*The expectation is taken over the protocol's randomness, user/adversarial behavior, where  $r_{\mathcal{P}}^t$  denotes the interest rate recommended by the protocol and  $r_{\mathcal{P}|T_{\beta}}^t$  is the same, when the indices of adversarial actions are known.*

Since an adversary can manipulate the protocol arbitrarily, the above measure signifies how adept the protocol is in weeding out historical data that has been manipulated, thus ensuring a cleaner convergence to the true interest rate equilibrium.

## 2.7 Baseline

For the baseline, we consider protocols akin to Compound, which utilize a piecewise linear interest rate curve to ensure stability. These protocols dynamically adjust interest rates at each block according to the model:

$$R_t = \begin{cases} R_0 + \frac{U_t}{U_{\text{opt}}} R_{\text{slope1}}, & \text{if } U_t \leq U_{\text{opt}} \\ R_0 + R_{\text{slope1}} + \left( \frac{U_t - U_{\text{opt}}}{1 - U_{\text{opt}}} \right) R_{\text{slope2}}, & \text{if } U_t > U_{\text{opt}} \end{cases}$$

In contrast to our proposed approach, these platforms generally set collateral factors and other market parameters through offline simulations that attempt to forecast near-future market conditions. Parameters are selected based on simulation outcomes and are subject to decentralized governance voting. Since this phase happens in an offline and opaque manner by centralized companies, we cannot compare this aspect of their protocol with ours.

Due to the piecewise linear nature of the interest rate as a function of utilization, these protocols achieve market stability only when either borrowers or lenders, but not both, exhibit elasticity [17]. To see why this is the case, note that according to Theorem 4, if borrowers are elastic ( $\eta_b > 0$ ), the equilibrium interest rate can be uniquely determined. And if lenders are elastic too, the equilibrium utilization is determined by Equation 14 which is not a linear function of  $r^*$ , hence a piecewise linear interest rate curve cannot satisfy the equilibrium conditions if both sides are elastic. Traditional DeFi platforms typically monitor the pool to identify the more elastic side of the market (usually borrowers) and design the curve accordingly.

## 3 Fast-slow thinker protocol

In this section, we outline a design for the interest rate and collateral factor function of  $\mathcal{P}$  that aims to achieve both the best possible convergence rate and the optimal optimality index. The protocol has the following two components.

A least squares estimator detects market disruptions that lead to instability and learns the equilibrium interest rate from borrowers' reactions, setting it agilely.

The long-term parameter planner consists of three parts: 1) A user behavior parameter estimator, which estimates the user behaviour model parameters that are required to optimize the optimality index. 2) An optimization module, which selects the optimal collateral factor to maximize optimality index, assuming negligible expected liquidation. 3) A liquidation threshold determination module, which sets the liquidation threshold as a function of the collateral factor to ensure zero expected liquidation.

The canonical scenario we use to evaluate our protocol is the following: at the beginning of the timeslot  $t$ , one or some of the market parameters (e.g.,  $\sigma, r_o^l, r_o^b, \alpha$ ) change to a new level and remain constant for a period  $T_m$ . The protocol must adapt to these new parameters by setting the equilibrium interest rate and the optimal collateral factor that maximizes the optimality index when the pool reaches equilibrium. Refer to Figure 1 for a visual representation of the protocol and its interaction with the pool.

### 3.1 Online interest rate controller

We model the problem of finding  $r^*$  using the linear regression method, where the borrow/re-pay rate depends on the difference  $r_t - r^*$  (motivating factor for borrowers) and  $\eta_b$  (borrower elasticity), with an added noise component; And use this model to estimate  $r^*$  adaptively. Here is the linear regression problem formulation:



$$\begin{aligned} \frac{B_{t+1} - B_t}{B_t} &= \eta_b \cdot \left( \alpha r_o^b - r_t + \pi(c_t) + \frac{\alpha}{c_t} \mathbb{E} \left[ \mathbb{1}_{p_{t+1} - p_t < 0} \left( \frac{p_{t+1} - p_t}{p_t} \right) \right] + \right. \\ &\quad \left. (1 - \alpha) \cdot \mathbb{E} \left[ \frac{p_{t+1} - p_t}{p_t} \right] \right) \\ &\stackrel{13}{=} \eta_b (r^* - r_t) \end{aligned}$$

$$\Delta \mathbf{B} = \mathbf{P}_b^t \cdot \Theta_b + \varepsilon \quad (16)$$

where:

- $\Delta \mathbf{B} = \left[ \frac{B_1 - B_0}{B_0}, \frac{B_2 - B_1}{B_1}, \dots, \frac{B_{t+1} - B_t}{B_t} \right]^T$  is the vector of normalized changes.
- $\mathbf{P}_b^t$  is the matrix of pool variables that affect borrower's behaviour:

$$(\mathbf{P}_b^t)^\top = \begin{bmatrix} 1 & 1 & \dots & 1 \\ r_0 & r_1 & \dots & r_t \end{bmatrix}$$

- $\Theta_b = [\eta_b r^*, -\eta_b]^T$  is the parameter vector
- $\varepsilon = [\varepsilon_0, \varepsilon_1, \dots, \varepsilon_t]^T$  is the noise vector.

The interest rate controller algorithm, outlined in 1, activates when  $\Delta B_t$  exceeds a threshold  $\delta$ , indicating  $r_t \neq r^*$ . The algorithm collects  $\Delta \mathbf{B}$  and  $\mathbf{P}_b^t$  to estimate  $\hat{\Theta}_b$ , setting  $r_t$  as the estimated  $\hat{r}^*$  according to  $\hat{r}^* = -\frac{\hat{\Theta}_b(0)}{\hat{\Theta}_b(1)}$ .

We cannot use vanilla LSE in this setting and simply output  $r_t = \hat{r}^* = -\frac{\hat{\Theta}_b(0)}{\hat{\Theta}_b(1)}$  because feeding back the estimated  $\hat{r}^*$  to the protocol may cause the rows of the  $\mathbf{P}_b$  matrix to become very close to each other. This happens when the estimator outputs an  $r_t$  that has been seen before, leading to redundant data points. Consequently, the theoretical guarantee of LSE to converge to the correct  $\Theta_b^*$  as the number of data points increases is impaired. To mitigate this problem, with a small probability, we sample a random interest rate. Finally, after running for a few iterations and accumulating a sufficiently large number of samples, the estimator stops and outputs the estimated  $r^*$ .

## Theoretical analysis

► **Theorem 8** (LSE convergence rate). *Assuming  $\eta_b > 0$ , the interest rate controller described in Algorithm 1 with stopping time  $\tau$  (taking  $\tau$  samples), satisfies the following:*

$$\mathcal{R}_{\mathcal{P}}(\delta, \tau) \sim \mathcal{O} \left( \frac{\log \frac{1}{\delta}}{\sqrt{\tau}} \right).$$

Moreover, if  $\eta_b$  is known:  $\mathbb{E}[r_\tau] = r^*, \forall \tau$

► **Theorem 9** (Baseline convergence rate). *Under a piece-wise linear interest rate function*

- *In the presence of elastic borrowers and inelastic lenders, we have:*

$$\begin{aligned} \mathbb{E}[r_\tau] &= r^* + D(1 - K \eta_b)^\tau, \quad 0 < 1 - K \eta_b < 1 \\ \mathcal{R}_{\mathcal{P}}(\delta, \tau) &\sim o \left( \frac{1}{\sqrt{\delta}} \right). \end{aligned}$$

where  $K, D$  are constants determined from the specifications of the interest rate curve.

- *If lenders are elastic as well, the protocol never stabilizes with rate  $r^*$ .*

## 27:14 Data-Driven Adaptive DeFi Borrow-Lending Protocol

Our interest rate controller provides an unbiased estimation of the equilibrium interest rate, with the estimation variance decreasing over time. In contrast, the baseline algorithm is a biased estimator of the equilibrium rate, becoming unbiased only as  $\tau \rightarrow \infty$ ; And the rate of convergence of the average error to zero is proportional to  $\eta_b$ , as the baseline protocol relies heavily on user actions to adjust the rate toward equilibrium rather than actively learning from user behavior. Additionally, the baseline algorithm maintains a constant error relative to the equilibrium rate due to the noise in user behavior and its inability to filter out this noise.

■ **Algorithm 1** Interest rate controller, utilizing a least squares optimization approach.

---

```

1: Initialize:  $t \leftarrow 0$ ,  $\delta \leftarrow$  stability threshold,  $t_{\text{sleep}} \leftarrow$  sleep time,  $\nu$  exploration probability
2: while True do
3:   if  $\frac{B_t - B_{t-1}}{B_{t-1}} < \delta$  then
4:     Sleep for  $t_{\text{sleep}}$ 
5:     Reset matrices  $\Delta \mathbf{B}$  and  $\mathbf{P}_b$ 
6:   else
7:     Add the new row  $[1, r_{t-1}]$  to  $\mathbf{P}_b^{t-2}$  to construct  $\mathbf{P}_b^{t-1}$  and
8:     Add the new column  $[\frac{B_t - B_{t-1}}{B_{t-1}}]$  to  $\Delta \mathbf{B}$ 
9:     Perform least squares estimation to find  $\hat{\Theta}_b \leftarrow ((\mathbf{P}_b^{t-1})^T \mathbf{P}_b^{t-1})^{-1} (\mathbf{P}_b^{t-1})^T \Delta \mathbf{B}$ 
10:    Parse  $\hat{\Theta}_b$  as  $[\hat{\eta}_b \hat{r}^*, -\hat{\eta}_b]$  and extract  $\hat{r}^*$  and set  $r_t = \hat{r}^*$ 
11:    With probability  $\nu$ , choose a random  $r_t \in [r_{\min}, r_{\max}]$ 
12:   end if
13:    $t \leftarrow t + 1$ 
14: end while
15: end algorithm

```

---

### 3.2 Risk parameters planner

First, we examine the conditions that ensure zero expected liquidation. These conditions provide the planner with the necessary bounds for setting the liquidation threshold.

► **Lemma 10.** *The expected liquidation incurred at time  $t + 1$ , given that the loan-to-value ratio at time  $t$  is  $c_t$ , can be expressed as*

$$\begin{aligned} \lambda(c_t, LT_t) &:= E[\lambda_t^i(p_{t+1})] \\ &= \frac{1}{1 - LT_t} \left( \Phi \left( \frac{\ln \left( \frac{c_t}{LT_t} \right) - \mu + \sigma^2}{\sigma} \right) - \frac{LT_t}{c_t} e^{\mu + \sigma^2} \Phi \left( \frac{\ln \left( \frac{c_t}{LT_t} \right) - \mu - \sigma^2}{\sigma} \right) \right) \end{aligned}$$

where  $\Phi$  denotes the cumulative distribution function of the standard normal distribution.

To maintain the expected liquidation below a small threshold (nearly zero), this lemma provides bounds on  $LT_t$  and  $\frac{c_t}{LT_t}$ . These bounds are used by the planner to set  $LT_t$  and constrain  $c_t$ . With negligible expected liquidation, the optimality index is simplified to include only the utilization error and the default term.

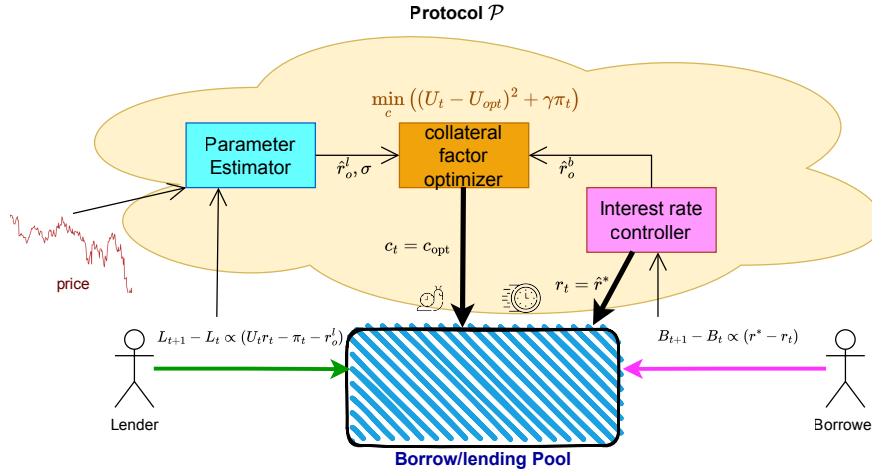
► **Corollary 11.** Given a fixed set of parameters  $r_o^l, r_o^b, \sigma, \mu$  and assuming that  $\lambda(LT_t, c_t) \approx 0$ , the maximization problem of  $OI_{\mathcal{P}}$  with respect to  $c_t$  can be formulated as follows:

$$\max_{c_t} OI_{\mathcal{P}} = \min_{c_t} \left( \frac{r_o^l}{b + \frac{a}{c_t}} - U_{opt} \right)^2 + \gamma \frac{r_o^l}{b + \frac{a}{c_t}} \Phi \left( \frac{\log(c_t) - \mu}{\sigma} \right) \quad (17)$$

$$- \gamma \frac{r_o^l}{b + \frac{a}{c_t}} \frac{\exp\left(\frac{\sigma^2}{2} + \mu\right)}{c_t} \cdot \Phi \left( \frac{-\mu + \log(c_t) - \sigma^2}{\sigma} \right) \quad (18)$$

where  $a := \alpha \left( e^{\mu + \frac{\sigma^2}{2}} \frac{\Phi\left(\frac{-\mu - \sigma^2}{\sigma}\right)}{\Phi\left(\frac{-\mu}{\sigma}\right)} - 1 \right)$  and  $b := \alpha r_o^b + (1 - \alpha) \left( e^{\mu + \frac{\sigma^2}{2}} - 1 \right)$ .

We can derive this corollary by substituting the default, expected price fall, and price change terms from Lemma 3 into the definition of optimality index stated in Definition 6.



■ **Figure 1** Protocol Overview. The interest rate controller observes borrower actions to estimate  $r^*$  and set  $r_t = \hat{r}^*$ . The collateral factor planner includes a parameter estimator and an optimizer: the estimator finds  $r_o^l, r_o^b$ , and  $\sigma$ , while the optimizer uses these estimates to determine the optimal collateral factor for the market.

The estimator module, described in Algorithm 2, uses a least squares estimator to analyze user behavior. It integrates outputs from the interest rate controller to determine the parameters  $r_o^l$  and  $r_o^b$ . Additionally, it learns the empirical price volatility  $\sigma$  and drift  $\mu$  from the recent price history. The optimizer component, detailed in Algorithm 3, utilizes these parameters to tackle the optimization problem presented in Corollary 11 and output the optimal collateral factor. To construct the estimator submodule, we model the lender's behavior using a linear regression approach, analogous to that of the borrower's behavior. This relationship is formulated as follows:

$$\frac{L_{t+1} - L_t}{L_t} = \eta_l \cdot (r_t U_t - U_t \pi(c_t) - r_o^l) + \varepsilon_t \quad (19)$$

## 27:16 Data-Driven Adaptive DeFi Borrow-Lending Protocol

■ **Algorithm 2**  $\hat{r}_o^l, \hat{r}_o^b$  - Estimator, auxiliary to the optimizer procedure.

---

```

1: Initialize:  $t \leftarrow 1$ ,  $\hat{\Theta}_l^0 \leftarrow \mathbf{0}$ , and read  $L_0, U_0, r_0, c$  from the pool,  $\delta_l \leftarrow$  stability threshold
2: Initialize:  $t_{\text{sleep}} \leftarrow$  sleep time,  $\delta_\theta \leftarrow$  least square convergence threshold,  $T_{\text{optimizer}} \leftarrow$  the
   minimum time interval between successive executions of the optimizer.  $\alpha \leftarrow$  fraction of
   first type borrowers
3: while True do
4:   if  $\frac{L_t - L_{t-1}}{L_{t-1}} < \delta_l$  then
5:     Reset  $\Delta\mathbf{L}, \mathbf{P}_l$ 
6:     Sleep for  $t_{\text{sleep}}$ 
7:   else
8:     Calculate  $\sigma, \mu$  empirically from the recent price history and use them to calculate
        $\pi(c_t)$ 
9:     Add the new row  $[1, -U_{t-1}\pi(c_t) + r_{t-1}U_{t-1}]$  to  $\mathbf{P}_l$  and  $[\frac{L_t - L_{t-1}}{L_{t-1}}]$  to  $\Delta\mathbf{L}$ 
10:    Perform least squares estimation to find  $\hat{\Theta}_l^t \leftarrow (\mathbf{P}_l^T \mathbf{P}_l)^{-1} \mathbf{P}_l^T \Delta\mathbf{L}$ 
11:    Parse  $\hat{\Theta}_l^t$  as  $[-\hat{\eta}_l \hat{r}_o^l, \hat{\eta}_l]^T$  and extract  $\hat{r}_o^l$ 
12:    Read the latest  $\hat{r}^*$  from Algorithm 1, and extract  $\hat{r}_o^b$  as follows:
13:    
$$\alpha \hat{r}_o^b \leftarrow \hat{r}^* - \pi(c_t) - \frac{\alpha}{c_t} \left( e^{\mu + \frac{\sigma^2}{2}} \frac{\Phi\left(\frac{-\mu - \sigma^2}{\sigma}\right)}{\Phi\left(\frac{-\mu}{\sigma}\right)} - 1 \right) - (1 - \alpha) \left( e^{\mu + \frac{\sigma^2}{2}} - 1 \right)$$

14:    if  $|\hat{\Theta}_l^t - \hat{\Theta}_l^{t-1}| < \delta_\theta$  then
15:       $c \leftarrow \text{OPTIMIZER}(\hat{r}_o^l, \hat{r}_o^b, \sigma)$ 
16:      Reset  $\Delta\mathbf{L}, \mathbf{P}_l$ 
17:      sleep for  $T_{\text{optimizer}}$ 
18:    end if
19:  end if
20:   $t \leftarrow t + 1$ 
21: end while
22: end algorithm

```

---

where  $\pi(c_t)$  is defined as per the simplifications in Lemma 3. The linear regression model for this behavior is represented by:

$$\Delta\mathbf{L} = \mathbf{P}_l \cdot \Theta_l + \varepsilon, \quad (20)$$

with:

■  $\Delta\mathbf{L} = \left[ \frac{L_1 - L_0}{L_0}, \frac{L_2 - L_1}{L_1}, \dots, \frac{L_{t+1} - L_t}{L_t} \right]^T$  capturing the normalized changes in lenders' supply.

■  $\mathbf{P}_l = \begin{bmatrix} 1 & -U_0 \pi(c_0) + r_0 U_0 \\ 1 & -U_1 \pi(c_1) + r_1 U_1 \\ \vdots & \vdots \\ 1 & -U_t \pi(c_t) + r_t U_t \end{bmatrix}$  as the matrix of pool variables at each time step.

■  $\Theta_l = [-\eta_l r_o^l, \eta_l]^T$  representing the parameter vector.

■  $\varepsilon = [\varepsilon_0, \varepsilon_1, \dots, \varepsilon_t]^T$  as the vector of noise terms.

Refer to Algorithm 2 and 3 to find a detailed description of the planner.

---

**Algorithm 3** Optimizer module.

---

```

1: procedure OPTIMIZER( $\hat{r}_o^l, \hat{r}_o^b, \sigma$ )
2:   Initialize:  $\alpha \leftarrow$  fraction of first type borrowers,  $U_{\text{opt}} \leftarrow$  desired utilization level, initial
   guess for collateral factor  $c(0) \stackrel{\text{U}}{\sim} [0, 1]$ ,  $\gamma \leftarrow$  default regularization factor,  $\kappa \leftarrow$  learning
   rate,  $i \leftarrow 0$  the gradient descent iterator,  $\delta \leftarrow$  gradient descent stop threshold.
3:   Set  $a = \alpha \left( \exp\left(\mu - \frac{\sigma^2}{2}\right) - 1 \right)$  and  $b = \alpha \hat{r}_o^b + (1 - \alpha) \left( \exp\left(\mu + \frac{\sigma^2}{2}\right) - 1 \right)$ 
4:    $\Psi(c) := \left( \left( \frac{\hat{r}_o^l}{b + \frac{a}{c}} - U_{\text{opt}} \right)^2 + \gamma \frac{\hat{r}_o^l}{b + \frac{a}{c}} \Phi\left(\frac{\log(c) - \mu}{\sigma}\right) - \lambda \frac{\hat{r}_o^l}{b + \frac{a}{c}} \frac{\exp\left(\frac{\sigma^2}{2} + \mu\right)}{c} \cdot \Phi\left(\frac{-\mu + \log(c) - \sigma^2}{\sigma}\right) \right)$ 
5:   while  $|\Psi(c(i)) - \Psi(c(i-1))| > \delta$  do
6:      $i \leftarrow i + 1$ 
7:      $c(i) \leftarrow c(i-1) - \kappa \frac{d\Psi(c)}{dc} \Big|_{c=c(i-1)}$ 
8:   end while
9:   return  $c(i)$ 
10: end procedure

```

---

### 3.3 Adversarial Robustness

In this section, we employ a gradient descent-based approach to perform robust linear regression, adapting the Torrent-GD method from the robust regression literature [7]. This method leverages the resilience of gradient descent to sparse corruptions and is highly efficient for large datasets, and helps us get an improved short-term algorithm for optimizing Adversarial Susceptibility.

The Torrent-GD modification to Algorithm 1 proceeds by updating the regression coefficients iteratively, reducing the influence of corrupt data points effectively. The update rule for the regression coefficients  $\hat{\Theta}_b$  at each iteration  $t$  is given by:

$$\hat{\Theta}_b(t+1) = \hat{\Theta}_b(t) - \kappa \nabla L_S(\hat{\Theta}_b(t)), \quad (21)$$

where  $\kappa$  is the learning rate and  $\nabla L_S(\hat{\Theta}_b(t))$  represents the gradient of the loss function computed only over the subset of data points  $S$  believed to be uncorrupted. This subset is dynamically determined in each iteration based on the residual errors.

The gradient of the loss function with respect to the regression coefficients is computed as:

$$\nabla L_S(w) = (\mathbf{P}_b^S)^T (\mathbf{P}_b^S \hat{\Theta}_b - \Delta \mathbf{B}_S), \quad (22)$$

where  $\mathbf{P}_b^S$  and  $\Delta \mathbf{B}_S$  are the features and responses of the uncorrupted subset, respectively.

The active set  $S$ , which includes indices of the data points assumed to be uncorrupted, is updated using a hard thresholding operator that selects the points with the smallest residuals:

$$S^{(t+1)} = \text{HT}\left(r^{(t)}, k\right), \quad (23)$$

where  $r^{(t)} = \Delta \mathbf{B} - \mathbf{P}_b \hat{\Theta}_b(t)$  represents the residuals at iteration  $t$ , and  $k$  is a threshold parameter controlling the number of points included in  $S$ , and the hard thresholding operator HT for a vector  $v \in \mathbb{R}^n$  and a threshold  $\tau$  is defined as follows:

$$\text{HT}(v_i, \tau) = \begin{cases} v_i & \text{if } |v_i| \geq \tau, \\ 0 & \text{otherwise.} \end{cases}$$

This operation retains only the components of  $v$  that are greater than or equal to the threshold  $\tau$  in absolute value, effectively zeroing out smaller coefficients.

The convergence of Torrent-GD is guaranteed under conditions that the noise and corruptions are sparse [7]. The algorithm can tolerate arbitrary adversarial corruptions as long as  $\beta < 50\%$ .

### 3.4 Blockchain implementation

To implement this protocol on a blockchain, we can utilize an optimistic Rollup solution like Arbitrum or Optimism. The core idea behind these Rollups is that the computation is performed off-chain by a Rollup validator. Only the state update data is posted on the blockchain, allowing challengers to validate the state update with the list of user transactions and challenge the validators in case of discrepancies. Therefore, in an optimistic scenario, no computation is performed on-chain.

Our online algorithm and the parameter estimator modules each have a computational complexity of  $\mathcal{O}(W)$ , where  $W$  is the sliding window used to collect and retain data for estimation. In practice, using  $W \approx 50$  was sufficient in our implementation when the user's elasticity is not awfully low (An elasticity of approximately 10 is sufficient when the noise standard deviation does not exceed 1). The optimizer module, in general, can be computationally infeasible since its objective function is not necessarily convex. However, since the range of possible collateral factors is limited, we can discretize the possible values into approximately 100 levels and find the optimal collateral factor through brute force. The calculation of the objective function itself is not computationally intensive.

#### Gas fee estimation

Even the Roll-up solution might necessitate running the computation on-chain in case of a dispute, therefore we need to estimate the on-chain computation cost. To estimate the gas cost for updating the protocol's slow and fast parts, we consider the least squares estimator (LSE) update of Algorithm 1 and 2. This update requires approximately  $12W$  multiplications and  $10W$  additions, where  $W$  is the length of the history window. Given the gas costs (5 gas per multiplication and 3 gas per addition based on [16]), for  $W = 50$ , we require 4500 gas for multiplications and additions. Additionally, storing new rows of  $\mathbf{P}_b$  and  $\Delta\mathbf{B}$  costs 20,000 gas per 32-byte storage, totaling 40,000 gas. Thus, the overall gas needed for the LSE update and storing new rows is 44,500 gas. The slow planner additionally needs to store a table of the optimality index values for different collateral factors and volatilities. Discretizing each into 100 and 10 values respectively, the storage cost for this table is  $20,000 \times 1000 = 20,000,000$  gas. Additionally, storing a table of the CDF of Gaussian variables to calculate expected defaults and liquidations requires  $20,000 \times 100 = 2,000,000$  gas for a reasonable discretization with 100 points. While more opcodes are involved in each interest rate and collateral factor update, they mainly consist of single arithmetic operations or multiple data storage, making the cost manageable.

### 3.5 Limitations

**User behaviour model.** We build upon a specific model of lender and borrower behavior, assuming that no single user controls a significant portion of the supply or demand. However, prior research has shown that this assumption might be unrealistic for many current platforms [32, 38]. In these scenarios, rational user strategies would differ from those assumed in our model. While our model manages to capture the main factors driving both sides of the

market in borrow-lending platforms, it fails to adapt to markets where a few entities control the majority of the funds. Furthermore, we assume that liquidators are always available to liquidate positions when protocols permit. However, real data indicates that this is not always the case, especially when the collateral asset lacks liquidity, resulting in considerable slippage when reselling the collateral [32].

**Price distribution.** Our choice of price model does not necessarily accurately describe the actual price distribution. However, as long as the price distribution belongs to a parameterized family of distributions with parameters that change slowly over time, the protocol can learn risk terms from user behavior, similar to our designed protocol. However, it may be challenging to analytically infer the price distribution parameters from user actions or to analytically relate the price distribution to key risk metrics, such as expected default or price fall.

**Risk neutrality.** In defining our utility functions, we assume users are risk-neutral and perceive their utility as the sum of profit minus expected risk, accurately calculating expectations over the price distribution. However, this may not be realistic. Our protocol learns the projection of user behavior onto our specific linear utility function and identifies parameters that best describe this behavior. For more complex user behavior, neural networks can be used to learn a vector representation of user behavior end-to-end. These user representation vectors are then fed to the interest rate controller and collateral factor planner, which are replaced by Deep Reinforcement Learning agents. These agents learn the optimal strategy based on feedback from their reward function. A key challenge is designing reward functions that best meet the protocol's needs.

**Adversarial robustness.** The adversarial resistance model we use for the LSE algorithm may be inadequate for permissionless blockchains. In such blockchains, any participant, including miners who organize and submit transactions, can act as adversaries. These adversaries can run the regression algorithm off-chain with different sets of transactions and find the set and the order that benefits them the most. Our current notion of adversarial robustness only protects against adversaries who control less than 50% of the funds and occasionally send transactions that deviate from the system's assumed supply and demand dynamics i.e., producing outlier data points.

**Single equilibrium point.** Our borrower behavior model assumes that there is always a single, unique interest rate that all borrowers are willing to pay, and that this equilibrium interest rate is known to all borrowers with some noise. However, in reality, this assumption may not hold true. Different groups of borrowers may have varying perceptions of the interest rate they are willing to pay. As a result, the system could have multiple equilibrium points, each attracting a different subgroup of borrowers.

## 4 Evaluation

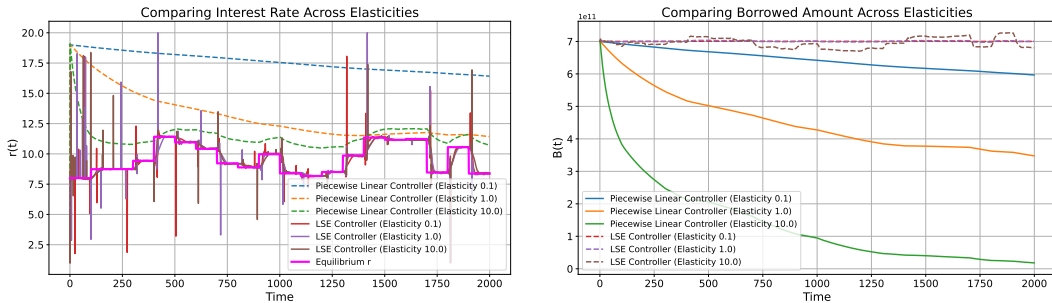
In this section, we test the interest rate controller and collateral factor planner described in section 3 and demonstrate their robustness to the change of market and user behaviours, moreover, we compare our interest rate controller with that of the baseline.

#### 4.1 Interest rate controller

**Experiment set-up.** In this experiment, we start a borrow-lending pool with an initial borrow supply of  $7 \times 10^{11}$  and an initial lend supply of  $10^{12}$ . Both the elasticity of borrowing and lending are set to 50, and the standard deviation of the borrowed and supply dynamic noise is 0.1. We assume very low price volatility, as it does not affect the determination of the equilibrium interest rate in this experiment. Every 100 time slots, we change  $r_o^b$  and allow the interest rate controller to adjust to the new equilibrium interest rate based on borrowers' behavior. During the exploration phases, the interest rate is randomly selected between  $r_{min} = 1$  and  $r_{max} = 20$  (outliers in figure 2a).

As shown in Figure 2a, the LSE-based controller adapts to the equilibrium interest rate consistently across different user elasticities with a nearly identical convergence rate, aligning with our theoretical guarantees. In contrast, the baseline controller, which sets the interest rate as a piecewise linear function of utilization, heavily relies on elasticity. It struggles to adapt to the new equilibrium interest rate in low elasticity scenarios and performs slightly better as elasticity increases.

While the performance of the baseline controller improves as borrower's elasticity increases, the consequences of misadjustment are more severe, causing significant borrower capital to leave the system when the interest rate is too high and to flood the system when it is too low. Figure 2b illustrates how the borrowed value changes with market conditions. The LSE-based controller consistently finds the stable interest rate, resulting in minimal market disruption whenever changes occur. In contrast, the baseline controller's inability to quickly find the correct rate causes substantial borrower exit from the system. This issue worsens with increased borrower elasticity due to higher repayment rates. Thus, even in high elasticity markets, the baseline controller is ineffective in preventing excessive capital inflows or outflows due to interest rate misadjustments.



(a) The LSE-based controller adapts to the new equilibrium interest rate quickly. In contrast, the piecewise linear interest rate curve fails to track the equilibrium interest rate when the borrower's elasticity is low.

(b) The LSE controller prevents the exit of borrower capital by setting a competitive interest rate, unlike the baseline controller.

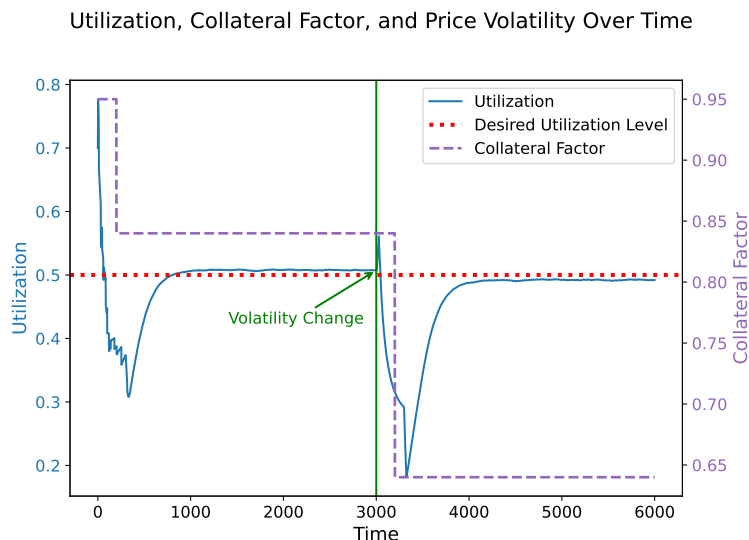
■ **Figure 2** Comparing the LSE-based interest rate controller and the piecewise linear curve.

#### 4.2 Collateral factor planner

We conducted an experiment to evaluate the performance of our collateral factor planner. The optimality index is defined as in Definition 6 with  $\lambda = 0$  and  $U_{opt} = 0.5$ . Initially, the system starts with an unoptimized collateral factor of  $c = 0.95$ . At timeslot  $t = 200$ , the optimizer activates and sets a new collateral factor of  $c = 0.84$ , which adjusts the utilization



to the desired level of 0.5 soon after that (the pace of reaching the equilibrium is a function of lenders' elasticity). At timeslot  $t = 3000$ , a change in price volatility disrupts the system. By timeslot  $t = 3200$ , the estimator accurately detects the new volatility and triggers the optimizer. The optimizer then sets a new collateral factor of  $c = 0.64$ , which stabilizes the utilization around 0.5 once again. This entire process is automated. The resulting utilization curve is shown in Figure 3.



■ **Figure 3** The collateral factor planner adapts to volatility changes and optimizes the collateral factor for achieving optimal utilization.

## 5 Conclusion and discussion

In this paper, we present a first-principles model of the behavior and incentives of borrowers and lenders in a DeFi market. We consider their alternative strategies and analyze how to achieve market equilibrium in the presence of price volatility. We mention empirical evidence on the validity of our model.

We propose a data-driven, borrow-lending protocol that sets the interest rate and over-collateralization ratio adaptively. By monitoring user reactions and learning from their behavior, our protocol determines a competitive interest rates and optimal collateral factors. The protocol consists of two components, 1) Fast interest rate controller: This component reacts online to user behavior, ensuring competitive interest rates and preventing over- or underpaying users. It has theoretical guarantees for fast convergence to the equilibrium interest rate. 2) Slow collateral factor planner: This component uses accurate market condition estimates to adjust the collateral factor, maintaining utilization at a desired level while controlling default risk. Overall, our protocol ensures rapid convergence to the equilibrium interest rate and optimal tuning of the collateral factor to achieve a desired equilibrium.

We implement our protocol and test its performance using simulated users, including uninformed ones. We compare our protocol with a baseline that uses piecewise linear functions to set interest rates based on utilization. Our protocol demonstrates superior performance compared to the baseline in practice.

---

**References**

---

- 1 Aave. Aave finance governance. <https://app.aave.com/governance/>. Accessed: 2023-05.
- 2 Aave. Aave whitepaper. [https://github.com/aave/aave-protocol/blob/master/docs/Aave\\_Protocol\\_Whitepaper\\_v1\\_0.pdf](https://github.com/aave/aave-protocol/blob/master/docs/Aave_Protocol_Whitepaper_v1_0.pdf). Accessed: 2024-02.
- 3 Aave. Interest rate curve in aave. <https://docs.aave.com/risk/liquidity-risk/borrow-interest-rate>. Accessed: 2023-05.
- 4 Ajna. Ajna whitepaper. [https://www.ajna.finance/pdf/Ajna\\_Protocol\\_Whitepaper\\_01-11-2024.pdf](https://www.ajna.finance/pdf/Ajna_Protocol_Whitepaper_01-11-2024.pdf). Accessed: 2023-05.
- 5 Jean Barthélemy, Clément Delaneau, Thomas Martignon, Benoît Nguyen, Marten Riho Pallum, and Salim Talout Zitan. Interest rates in decentralised finance. <https://www.banque-france.fr/en/publications-and-statistics/publications/interest-rates-decentralised-finance#:~:text=The%20setting%20of%20interest%20rates%20in%20DeFi&text=The%20formula%20is%20simple%3A%20the,interest%20rates%20cannot%20be%20negative.>, 2023. Accessed: 2023-05.
- 6 Mahsa Bastankhah, Viraj Nadkarni, Xuechao Wang, Chi Jin, Sanjeev Kulkarni, and Pramod Viswanath. Thinking fast and slow: Data-driven adaptive defi borrow-lending protocol, 2024. [arXiv:2407.10890](https://arxiv.org/abs/2407.10890).
- 7 Kush Bhatia, Prateek Jain, and Purushottam Kar. Robust regression via hard thresholding, 2015. [arXiv:1506.02428](https://arxiv.org/abs/1506.02428).
- 8 Sylvain Carre and Franck Gabriel. Security and efficiency in defi lending. *Université Paris-Dauphine Research Paper*, 2023.
- 9 Tarun Chitra, Peteris Erins, and Kshitij Kulkarni. Attacks on dynamic defi interest rate curves. *arXiv preprint*, 2023. [arXiv:2307.13139](https://arxiv.org/abs/2307.13139).
- 10 Jonathan Chiu, Emre Ozdenoren, Kathy Yuan, and Shengxing Zhang. On the fragility of defi lending. *Available at SSRN 4328481*, 2022.
- 11 Samuel N Cohen, Marc Sabate-Vidales, Lukasz Szpruch, and Mathis Gontier Delaunay. The paradox of adversarial liquidation in decentralised lending. *Available at SSRN 4540333*, 2023.
- 12 Samuel N Cohen, Leandro Sánchez-Betancourt, and Lukasz Szpruch. The economics of interest rate models in decentralised lending protocols. *Available at SSRN*, 2023.
- 13 Compound. Compound finance governance. <https://compound.finance/governance>. Accessed: 2023-05.
- 14 Compound. Compound whitepaper. <https://compound.finance/documents/Compound.Whitepaper.pdf>. Accessed: 2024-02.
- 15 Compound. Impermanent loss in defi lending. <https://docs.compound.finance/interest-rates/>. Accessed: 2023-05.
- 16 ethereum.org. Opcodes for the evm, 2024. URL: <https://ethereum.org/en/developers/docs/evm/opcodes/>.
- 17 Gauntlet. Arfc aave v3 interest rate curve recommendations from gauntlet, 2023. Accessed: 2024-05-20. URL: <https://governance.aave.com/t/arfc-aave-v3-interest-rate-curve-recommendations-from-gauntlet-2023-04-27/12921>.
- 18 Mohak Goyal, Geoffrey Ramseyer, Ashish Goel, and David Mazières. Finding the right curve: Optimal design of constant function market makers, 2023. [arXiv:2212.03340](https://arxiv.org/abs/2212.03340).
- 19 Gauntlet Research Group. Aave market risk analysis. URL: <https://gauntlet.network/reports/aave>.
- 20 Gauntlet Research Group. Compound market risk analysis. URL: [https://assets-global.website-files.com/648bdc0d4b8ce322f27da0af/651f146b6a1c8cc78e6c9cac\\_Gauntlet%20-%20Compound%20Market%20Risk%20Assessment.pdf](https://assets-global.website-files.com/648bdc0d4b8ce322f27da0af/651f146b6a1c8cc78e6c9cac_Gauntlet%20-%20Compound%20Market%20Risk%20Assessment.pdf).
- 21 Lewis Gudgeon, Sam Werner, Daniel Perez, and William J Knottenbelt. Defi protocols for loanable funds: Interest rates, liquidity and market efficiency. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, pages 92–112, 2020.
- 22 Thomas Kailath. *Linear Systems*. Prentice-Hall, 1980.



- 23 Will Kenton. Market distortion. <https://www.investopedia.com/terms/m/market-distortion.asp>. Accessed: 2023-05.
- 24 Ariaeh Klages-Mundt and Andreea Minca. (in) stability for the blockchain: Deleveraging spirals and stablecoin attacks. *PubPub*, 2021.
- 25 Arthur D Kuo. A least-squares estimation approach to improving the precision of inverse dynamics computations. *ASME digital collection*, 1998.
- 26 Brian McWilliams, Gabriel Kruppenacher, Mario Lucic, and Joachim M Buhmann. Fast and robust least squares estimation in corrupted linear models. *Advances in Neural Information Processing Systems*, 27, 2014.
- 27 Jason Milionis, Ciamac C. Moallemi, and Tim Roughgarden. Automated market making and arbitrage profits in the presence of fees, 2023. [arXiv:2305.14604](https://arxiv.org/abs/2305.14604).
- 28 Morpho. Morpho whitepaper. <https://whitepaper.morpho.xyz/>, 2023. Accessed: 2023-05.
- 29 Viraj Nadkarni, Jiachen Hu, Ranvir Rana, Chi Jin, Sanjeev Kulkarni, and Pramod Viswanath. Zeroswap: Data-driven optimal market making in defi. *arXiv preprint*, 2023. [arXiv:2310.09413](https://arxiv.org/abs/2310.09413).
- 30 Pedro M. Negrón. Navigating risks in defi lending. <https://medium.com/intotheblock/navigating-risks-in-defi-lending-a034dbf83b54>. Accessed: 2023-05.
- 31 Board of governors of the federal reserve systems. Capital planning at large bank holding companies: Supervisory expectations and range of current practice. <https://www.federalreserve.gov/bankinforeg/stress-tests/ccar/august-2013-estimation-methodologies-for-losses-revenues-and-expenses.htm>. Accessed: 2023-05.
- 32 Kaihua Qin, Liyi Zhou, Pablo Gamito, Philipp Jovanovic, and Arthur Gervais. An empirical study of defi liquidations: Incentives, risks, and instabilities. In *Proceedings of the 21st ACM Internet Measurement Conference*, pages 336–350, 2021.
- 33 Thomas J Rivera, Fahad Saleh, and Quentin Vandeweyer. Equilibrium in a defi lending market. *Available at SSRN 4389890*, 2023.
- 34 Kanis Saengchote. Decentralized lending and its users: Insights from compound. *Journal of International Financial Markets, Institutions and Money*, 87:101807, 2023. [doi:10.1016/j.intfin.2023.101807](https://doi.org/10.1016/j.intfin.2023.101807).
- 35 Lukasz Szpruch, Jiahua Xu, Marc Sabate-Vidales, and Kamel Aouane. Leveraged trading via lending platforms. *Available at SSRN*, 2024.
- 36 Eli Tan. Impermanent loss in defi lending. <https://www.coindesk.com/learn/defi-lending-3-major-risks-to-know/>. Accessed: 2023-05.
- 37 Yue Xing, Ruizhi Zhang, and Guang Cheng. Adversarially robust estimate and risk analysis in linear regression. In *International Conference on Artificial Intelligence and Statistics*, pages 514–522. PMLR, 2021.
- 38 Aviv Yaish, Maya Dotan, Kaihua Qin, Aviv Zohar, and Arthur Gervais. Suboptimality in defi. *Cryptology ePrint Archive*, 2023.
- 39 Xiao Zhang and Feng Ding. Adaptive parameter estimation for a general dynamical system with unknown states. *International Journal of Robust and Nonlinear Control*, 30(4):1351–1372, 2020.
- 40 Stephan Zheng, Alexander Trott, Sunil Srinivasa, David C. Parkes, and Richard Socher. The AI economist: Optimal economic policy design via two-level deep reinforcement learning. *CoRR*, abs/2108.02755, 2021. [arXiv:2108.02755](https://arxiv.org/abs/2108.02755).



# SoK: Attacks on DAOs

Rainer Feichtinger<sup>1</sup>  

ETH Zürich, Switzerland

Robin Fritsch  

ETH Zürich, Switzerland

Lioba Heimbach  

ETH Zürich, Switzerland

Yann Vonlanthen  

ETH Zürich, Switzerland

Roger Wattenhofer  

ETH Zürich, Switzerland

---

## Abstract

*Decentralized Autonomous Organizations (DAOs)* are blockchain-based organizations that facilitate decentralized governance. Today, DAOs not only hold billions of dollars in their treasury but also govern many of the most popular *Decentralized Finance (DeFi)* protocols. This paper systematically analyses security threats to DAOs, focusing on the types of attacks they face. We study attacks on DAOs that took place in the past, attacks that have been theorized to be possible, and potential attacks that were uncovered and prevented in audits. For each of these (potential) attacks, we describe and categorize the attack vectors utilized into four categories. This reveals that while many attacks on DAOs take advantage of the less tangible and more complex human nature involved in governance, audits tend to focus on code and protocol vulnerabilities. Thus, additionally, the paper examines empirical data on DAO vulnerabilities, outlines risk factors contributing to these attacks, and suggests mitigation strategies to safeguard against such vulnerabilities.

**2012 ACM Subject Classification** Security and privacy → Economics of security and privacy; Human-centered computing → Collaborative and social computing

**Keywords and phrases** blockchain, DAO, governance, security, measurements, voting systems

**Digital Object Identifier** 10.4230/LIPIcs.AFT.2024.28

**Related Version** *Full Version*: <https://arxiv.org/abs/2406.15071> [51]

**Supplementary Material** *Dataset (Website)*: <https://daoattacks.ethz.ch>

**Acknowledgements** We thank Hubert Ritzdorf from ChainSecurity for his precious feedback.

## 1 Introduction

*Decentralized Autonomous Organizations (DAO)* are organizational structures that facilitate the trustless management of projects that run on a blockchain [11]. In DAOs, governance is typically controlled by the holders of a designated governance token. Those who own these tokens can thus determine the course of the DAO. Today, DAOs govern various blockchain projects, such as ecosystem governance of Layer 2s (e.g., Arbitrum and Optimism) and many of the most-used decentralized applications (e.g., Aave, Compound, ENS, Lido, MakerDAO, and Uniswap). Moreover, DAOs are estimated to hold and control in excess of \$30B in their treasuries [41]. Consequently, they hold significant power and a central position in the blockchain ecosystem.

---

<sup>1</sup> The authors of this work are listed alphabetically, correspondence through [daoattacks@ethz.ch](mailto:daoattacks@ethz.ch).



DAOs have been threatened by attacks and hacks ever since their inception. “*The DAO*” on Ethereum was the first attempt at creating a DAO on a blockchain. However, in 2016 an infamous hack stole \$50M worth of ETH from the DAO before it even became operational [106]. The event was so severe that it led to a controversial hard fork of the Ethereum blockchain. The original (unforked) blockchain still operates today and is known as Ethereum Classic. Notably, even before the fatal hack, other possible attacks on The DAO had been discussed [86]. The DAO hack highlights the significant threat attacks on DAOs can present not only to the DAOs themselves but also to the broader ecosystem. Additionally, given that DAOs are still in their early days and the ongoing evolution of their design frameworks, DAOs are particularly vulnerable to various novel attack vectors.

In this work, we study real-world incidents and attacks on DAOs, attacks that have been theorized to be possible, and new potential attack vectors. We summarize our main contributions in the following.

- We present and categorize attack vectors on DAOs. To be precise, we categorize attacks on DAOs into four categories: (i) bribing (BR) attacks, (ii) token control (TC) attacks, (iii) human-computer interaction (HCI) attacks, and (iv) code and protocol vulnerability (CP) attacks.
- We examine 28 real-world incidents and attacks across four blockchains and indicate the attack vectors utilized. Our work finds that these attacks exploited vectors from all four introduced categories fairly evenly. Similarly, we categorize attacks described in academic papers or reports as well as those uncovered and prevented through audits. Notably, less tangible attack vectors that take advantage of human and economic aspects involved in governance represent a majority of real-world incidents but are generally not analyzed in audits, which heavily skew toward code and protocol vulnerability attacks.
- Guided by our categorization of real-world incidents, we introduce seven risk factors for DAOs and empirically analyze the susceptibility of 26 DAOs of all shapes and sizes to them.
- Finally, we collect and discuss various mitigations and safeguards for DAOs.

With our work, we aim to enhance the understanding of DAO security challenges and guide the development of more robust governance frameworks.

## 2 Background

An early definition of DAOs was provided by Vitalik Buterin, who argued that DAOs are entities with internal capital (i.e., *treasuries*), that have automated processing at their core, and human processing at their edges [117].

This definition still holds true today, though DAOs have been continually evolving and are still undergoing substantial efforts to improve. Therefore, countless implementations exist. Nonetheless, some specific designs have reached greater popularity, as new DAOs borrow ideas, pieces of code from the smart contracts implementing the DAOs logic, or entire structures from pre-existing DAOs. Examples thereof are the OpenZeppelin implementations based on the Compound Governor contracts (also used e.g., by Uniswap, ENS, and Gitcoin) and Aragon DAOs (used by e.g., Lido and Curve). Other DAOs might follow a more unique design (e.g., Maker and Optimism) or reuse code only partially. In the following, we aim to distill the main attributes that current DAOs share.

**Token Voting.** Reaching an agreement between individuals in the decentralized setting of a blockchain is not as simple as in the physical world, where identities are known. On blockchains, only pseudonymous addresses are publicly available. In particular, these

addresses could include *sybils*, i.e., multiple addresses created and controlled by a single individual. To address this issue, DAOs typically issue governance tokens. These tokens come with voting rights.<sup>2</sup> Initially, these tokens can be distributed among stakeholders through various methods. The most popular methods include giveaways (*airdrop*) to early users of a protocol, as well as allocating a portion of the tokens to the development team or early investors. After launching, the governance tokens become freely tradable on the open market, enabling individuals to acquire them and thus acquire voting power. In this regard, governance tokens exhibit parallels with shares in companies that grant holders voting rights at shareholder meetings.

**Voting Rights and Delegation.** While token holders directly vote on each proposal themselves in some DAOs (e.g., Lido), other DAOs introduce an intermediary in the voting process: delegates. In these DAOs, token holders can delegate the voting power associated with their tokens to a delegate they believe represents their views well. Inspiration for this *delegative* approach is taken from *liquid* democracy [54, 10]. Proponents of this approach for DAO voting argue that delegation allows token holders to participate in the governance process passively (i.e., they must not actively keep track of each proposal) and can further help reduce the blockchain transaction fees incurred by DAO members. While delegation is optional in some DAOs (e.g., Aave), many DAOs require delegation (e.g., Compound, ENS, and Uniswap). In DAOs requiring delegation, before tokens can be used for voting, the address holding the tokens must delegate them to a *delegate* address (which may be the same address). Finally, some DAOs require the tokens to be locked in a specified contract to gain the associated voting rights (e.g., Curve and Maker).

**Deliberation Phase.** Generally, DAO governance processes involve multiple steps, which could include discussions on public governance forums or off-chain *temperature check* votes before a final vote [87] ahead of a proposal being put forward on-chain and voted on. However, these steps are often mere social conventions. The final on-chain vote is often the only obligatory part of the process.

**Proposals.** Generally, any token holder with a sufficient number of tokens (exceeding a pre-defined proposal threshold) can put forward a proposal. There are some DAOs though that force proposals to be vetted by a committee before being accepted on-chain.

**Voting Phase.** Once a proposal is submitted on-chain, there is generally a *proposal delay*, i.e., a pre-specified number of blocks before a snapshot is taken of the balances of all token holders (or delegates). The snapshot determines their voting power, and cannot be changed a posteriori. Thereafter, voting starts and generally lasts for a pre-determined number of blocks. During voting, any address with voting rights (token holder or delegate) may submit a vote.

**Execution Phase.** If a majority votes in favor of a proposal and a pre-determined Quorum is reached, the proposal is accepted. In some DAOs, on-chain proposal execution is automatic, potentially only after a pre-defined *timelock delay*. The execution is scheduled manually by a trusted party in other DAOs.

---

<sup>2</sup> Generally, each token counts as one vote. Curve is an exception, where the voting power depends not only on the number of tokens held but also the duration the tokens are locked for.

### 3 Categorization of Attack Vectors

In the following, we provide a categorization and description of attack vectors.

#### 3.1 Bribing

► **Definition.** In a *bribing* (*BR*) attack, an attacker pays to change votes or to acquire voting power without acquiring the underlying governance tokens. The controlled votes and voting power are then utilized to pass a malicious proposal in a governance vote.

Bribing attacks can take the form of paying to obtain voting rights of governance tokens to vote for a certain proposal without acquiring the underlying token, which is often referred to as *vote buying*. Another possibility is directly bribing token holders or delegates.

Given that vote buying has been documented in shareholder governance of traditional companies [76], it is a plausible future concern for DAO governance and was already been a topic of discussion since the early days of DAOs [37, 20].

**Bribing Token Holders or Delegates (BR1).** Bribing governance participants, i.e. token holders or delegates, can take many forms: it can be done on-chain or off-chain, programmatically using smart contracts or by personal contact. Furthermore, bribes could be fixed sums or a proportion of the proceeds from a successful attack. Note that using proceeds of the attack to bribe leads to a situation similar to an attack by a majority coalition, where the proceeds are split among participants (see TC5).

When voting power is highly centralized, as is the case for many DAOs at the time of writing [52], bribing only a few of them can suffice to change a vote. On the other hand, voting rights being highly distributed can also make it cheaper for an attacker to bribe: holders of small amounts of voting power, besides having little to lose from a successful attack, also have little influence on the outcome of a vote. Hence, it can be economically rational for them to cheaply sell their vote, as described by Buterin [19].

Bribing delegates to vote a certain way could potentially be particularly attractive for an attacker. For governance systems using delegated token voting, a small number of delegates often controls large amounts of voting rights. On the other hand, these delegates do not actually hold the corresponding amount of governance tokens, meaning they are not exposed to the price risk from a successful governance attack. Hence, bribing them could potentially be significantly cheaper for an attacker than bribing governance token holders.

One deterrent against a delegate bribing attack, that is present in most current DAOs, is the fact that most delegates are often publicly (or at least pseudonymously) known. This means that delegates stand to lose their reputation and future earnings based on it, and may even face a risk of criminal charges for accepting bribes.

**Vote Buying Protocols (BR2).** The act of vote buying or *bribing* can be facilitated by a smart contract protocol. Such protocols allow token holders to deposit their governance tokens into pools, and earn fees from users paying to use the voting rights of the pooled tokens. In particular, this means that vote buyers do not need to deposit collateral, contrary to using traditional lending platforms such as Compound (see Section 3.2). Paladin Lending, as one example of a vote buying protocol, is described in the following.



**Case Study** Paladin Lending

Paladin Lending [98] lets holders deposit their tokens into pools and in return receive a proportional share of the fees collected in the pool. Users can then borrow the voting power of deposited tokens. A loan contract is automatically created if a user wants to borrow voting power. The borrowed token amount is transferred from the pool to the loan contract, and the votes are delegated to the user. Hence, the user has no direct access to the tokens but can use their voting power. The user pays a fee for borrowing the voting power. At the latest when this fee has been consumed, the tokens in the loan contract will be returned to the pool.

Importantly, with a vote buying protocol such as Paladin Lending, one does not borrow the actual token, but only the voting rights. We also perform an empirical analysis of Paladin Lending (see Appendix of the full version of this paper [51]) to show that low liquidity currently does not allow attacks exclusively using this attack vector.

Daian et al. [37] introduce a particular type of vote buying protocol: *Dark DAOs*. In addition to facilitating vote buying using smart contracts, Dark DAOs are implemented in a privacy-preserving manner. Note that activity on vote buying protocols such as Paladin Lending is publicly recorded on the blockchain. Vote buying activities through Dark DAOs, on the other hand, cannot be detected, meaning that other governance participants cannot react to such an attack. While there are no known cases of active Dark DAOs at the time of writing, they have been theoretically studied by Austgen et al. [5], and proof-of-concept prototypes have been published [4].

### 3.2 Token Control

► **Definition.** With *token control (TC)* attacks, an attacker takes possession or is already in possession of a significant amount of governance tokens. The attacker then uses the voting power associated with these tokens to get their malicious proposal accepted in a governance vote.

This family of attack vectors is of the simplest nature. The attacker merely gains control of a sufficient number of governance tokens to take over the DAO by passing a malicious proposal according to DAO's intended voting process.

Depending on the governance model implemented by the DAO, the required proportion of governance tokens for a successful attack varies. For instance, many DAOs require tokens to be delegated to an address for them to be used in voting and take a snapshot of the current state of delegations at the start of the voting period. For such governance systems, an attacker must only hold a token amount exceeding the amount of previously delegated tokens, and delegate these governance tokens to themselves, thereby securing a majority of the delegated votes. By timing the creation of a proposal accordingly, an attacker can leave very little time (depending on the governance system's parameter choices, see RF4 in Section 5 for more details) for others to react and delegate their tokens. This can almost guarantee the attacker the required voting power to pass their desired proposals. For DAOs that do not require the tokens to be delegated, the attacker would need to hold more than 50% of the circulating token supply for a guaranteed victory of their proposal or hope that not sufficiently many votes are cast, i.e., voter turnout does not increase dramatically in face of a malicious proposal. Short voting windows as well as the absence of reliable communication channels further increase the risk of such attacks for these DAOs.

In the following, we discuss the main possibilities for an attacker to gain possession of the required voting power. Note that in Section 5, we provide an additional empirical analysis of the susceptibility of a set of 26 DAOs to this kind of attack.

**Token Purchase (TC1).** The attacker buys governance tokens on the open market. This can be done on-chain through decentralized exchanges, or on off-chain centralized exchanges. After using the tokens for voting, the attacker can sell back the tokens to the open market. Importantly, when buying governance tokens, the attacker takes on price risk while holding the tokens. If the attack leads to a decrease in the governance token's market price, the attacker incurs a financial loss. Additionally, the attacker pays trading fees when buying and selling the tokens. Note that the attacker can potentially hedge the price risk using derivatives. However, the availability of such derivatives may be limited depending on the governance token in question.

Attacks through token acquisition have been attempted and have occurred in several DAOs. They are especially attractive and profitable if the value of the treasury (excluding the governance token itself, which is likely to decrease in value in the event of an attack) exceeds the capital required to buy the necessary voting power. In Section 5, we compare the treasury values of DAOs to the value of delegated tokens for a set of 26 DAOs. It is relatively common for the total value of the DAO's treasury to exceed the value of delegated tokens, though this is only rarely the case when excluding the governance tokens from the treasury.

In the following, we present a case study of two consecutive recent governance attacks through token acquisition on the Indexed Finance DAO, a protocol for portfolio management. While interest in the project declined after it was hacked in October 2021 [1], various tokens remained in the project's timelock contract controlled by the DAO.

#### Case Study Indexed Finance

On 16 November 2023, over ten hours, the attacker bought NDX tokens (i.e., the protocol's governance token) via decentralized exchanges, self-delegated these tokens, initiated a proposal, voted in favor of this proposal, and sold the tokens again [1]. The proposal would allow the attacker to take control of the timelock, mint new NDX tokens, and steal tokens from the timelock (including both NDX and other tokens). A call for action by one of the protocol founders asked users to vote against the proposal. In the end, user votes against the proposal were sufficient to narrowly prevent the attack. Interestingly, the attacker sold his NDX tokens before the end of the proposal and thereby lost his voting power. As a result, the proposer would have been below the proposal threshold and the proposal could have been canceled by anyone. However, this was not done.

Fearing a potential second attack, the community attempted to implement defensive measures. They created a proposal to transfer control of the timelock to a smart contract not be under anyone's control, i.e., the tokens in the timelock would forever be inaccessible if the proposal were executed. Then, on 22 November 2023, another attacker (i.e., a different account than the previous attacker) created a similar proposal that would transfer the admin rights of the timelock to the attacker. This time, the attacker acquired more NDX tokens than the 16 November attacker, and there were not enough votes against this proposal. Thus, the only way to stop the attacker from getting access to the tokens was through passing the proposal that would make the tokens forever inaccessible. Importantly, as this proposal was created a day earlier, it would not only execute first but the attacker also only acquired the tokens after voting had started on the community's proposal and therefore did not have the majority in that vote. What followed, as no one wanted the community's proposal to be executed, was a message exchange between the attacker and the Indexed Finance team using input data of Ethereum transactions. In the end, an agreement was reached, and the attacker received  $\approx$  \$10K via an escrow contract after withdrawing his proposal. In conclusion, the two attacks were only mitigated by luck (i.e., the first attacker bought too few tokens) and by unorthodox proposals (i.e., making the tokens forever inaccessible).

In the aftermath of the attacks, the Indexed Finance DAO accepted a proposal that transferred control of the timelock to a multi-signature wallet controlled by former protocol contributors.

Indexed Finance demonstrates the complexities of protecting against this attack vector in the absence of adequate countermeasures. Nevertheless, there exist potential protections that DAOs can put in place. For example, DAOs may opt to restrict proposals from spending the entire treasury or grant veto power to a multi-signature. We provide more detail in Section 6.

**Token Loan (TC2).** The attacker borrows governance tokens against collateral using lending protocols. Apart from needing to post collateral, the attacker also pays borrowing fees for the period of borrowing the tokens. Importantly, the attacker does not take on price risk when borrowing tokens. After voting for an attacking proposal, the full amount of governance tokens can be returned and the attacker receives back their collateral.

There have been several alleged attempts of DAO attacks through token loans. In early 2022, Justin Sun presumably borrowed large amounts of MKR, the governance token of MakerDAO, to sway a vote. However, he returned the tokens after his actions were detected and did not end up voting [2]. A couple of days later, a similar failed attempt by Justin Sun took place in Compound's governance with borrowed COMP tokens [114].

**Flash Loan (TC3).** With a flash loan, the attacker only borrows the governance tokens for the duration of a transaction. While the attacker pays a fee to borrow the governance token, the attacker does not need to post any collateral, i.e., does not require access to significant funds. Many protocols protect themselves against flash loan attacks by implementing a delay between the proposal creation and the start of the voting period. Nonetheless, flash loan attacks on DAOs have occurred in the past, the most prominent example is a flash loan attack on the Beanstalk governance described in the following case study.

#### Case Study Beanstalk

Beanstalk is a stablecoin protocol. On 17 April 2022, Beanstalk suffered an attack that resulted in damages of approximately \$182M, netting the attacker a profit of around \$76M [44, 50, 13]. The attacker exploited a vulnerability in Beanstalk's governance system, which was not secure against flash loan attacks. The attacker took a flash loan worth approximately \$1B. This loan allowed them to achieve a two-thirds majority in Beanstalk's governance. With this majority, they could execute a malicious proposal immediately using an emergency commit function.

**Whale Activation (TC4).** Inactive token holders with a large number of tokens (often referred to as whales) can suddenly become active in the governance. In DAOs requiring tokens to be delegated, this can be especially problematic. An attacking whale can delegate their tokens and promptly initiate a proposal. Importantly, large entities holding sufficiently many tokens to take over the DAO exist for many DAOs using delegated token voting (see Section 5). Notably, there was one instance in the past where a centralized exchange unexpectedly delegated the UNI governance tokens it held, i.e., the tokens custodied on behalf of its users. They, however, claimed to have accidentally delegated these tokens [85].

**Majority Coalition (TC5).** In governance systems using majority token voting, it is generally possible for a simple majority of voting tokens to accept any proposal, and effectively, take control of the DAO. In particular, the majority could distribute the entire DAO treasury among themselves. Settings of this type have been modeled in game theory as *coalition games with transferable utility* or *majority games with stable sets* describing possible attacking coalitions [17, 70]. Such coalition attacks are specifically attractive when the treasury value of a DAO is high compared to the value of (delegated) governance tokens. We have empirically studied this relation in Section 5 (see RF3) for 26 DAOs.

Of course, a majority of voting tokens can also vote to split the treasury among *all* token holders, or more generally, dissolve the DAO. In this particular case, i.e., if all token holders get a share of the treasury proportional to their voting power, a majority coalition would not pose an attack. An example of this happening in practice is DigixDAO's token holders voting to dissolve the DAO and return all ETH held in the treasury to the token holders (which was worth more than the value of all governance tokens) [120]. However, in all other cases, where a strict subset of token holders come together to take control of a DAO, a majority coalition presents an attack.

### 3.3 Human-Computer Interaction

► **Definition.** *Human-computer interaction (HCI)* attacks aim to manipulate the voting process by exploiting user-facing interfaces and applications or human behaviors involved in the DAO's voting process.

This family of attacks lies at the boundary between the blockchains (computers) and humans. The attack vectors in this family do not exploit vulnerabilities in the underlying governance protocol itself, but rather in the interfaces, applications, or human behaviors surrounding DAO governance.

**User Interface Issues (HCI1).** Many users participate in the voting process through aggregator websites that provide a convenient *user interface (UI)*. Thus, bugs or malicious code in these UIs can lead to users not voting as they intended or not being able to vote at all. For example, users voting through a UI typically sign a vote transaction prepared by the UI. If this transaction is incorrectly prepared, users will potentially vote differently than they intended by signing the transaction. An incident of this type occurred with Tally, a closed-source and widely-used UI for on-chain governance.

#### Case Study Tally

On 19 August 2021, a bug, which had persisted from 30 April to 19 August 2021, was discovered on Tally [110]. The bug inadvertently altered the voting process: transactions of users wishing to vote *against* a proposal were erroneously constructed by Tally. This led to these votes being recorded as votes *in favor* on the blockchain. The issue went unnoticed since the transaction arguments were not presented in an easily understandable format, making it challenging for users to notice the discrepancy between their intended vote and the registered vote.

While there is no evidence to suggest that this bug in Tally significantly influenced the outcomes of any votes, it nonetheless highlights a critical vulnerability in centralized, closed-source front-ends for governance systems. Once a vote is cast on-chain for a proposal, it cannot be retracted or altered. This means that if a user realizes their vote has been incorrectly cast due to a platform error, they are powerless to correct it. Thus, bugs in UIs can heavily influence the voting process in DAOs, and the possibility of inserting malicious code into these UIs poses a serious risk for DAOs.

On a similar note, the unavailability of the aforementioned UIs can pose a threat to a functioning DAO governance vote. The unavailability could be caused by technical issues with the UI as well as by deliberate *Denial of Service (DoS)* attacks. If widely-used UIs became unreachable ahead of a vote, users relying on these platforms to cast their votes might be deterred or prevented from voting. To the best of our knowledge, no such attack has taken place or been attempted. Nonetheless, it presents a risk worth considering for DAOs when designing their governance systems.

**Proposal Obfuscation (HCI2).** Obfuscation of the real intent of a proposal is a further possible attack vector, which presents a risk to DAOs, especially in combination with a weak validation of the proposal – making sure that the proposal description matches its contents. Take as an example a proposal that appears to be a legitimate proposal but, in reality, inserts malicious code that allows the attacker to steal the DAO’s funds. Such an attack was successfully performed on the Tornado Cash governance.

#### Case Study Tornado Cash

On 20 May 2023, an attacker gained control of the governance system of Tornado Cash [44, 9]. The attacker purchased TORN tokens through decentralized exchanges and imitated a previously accepted proposal. Due to the striking resemblance to this earlier proposal, the new, malicious one was also approved by the community. However, there was a critical and deliberate difference in the attacker’s proposal: it included a self-destruction feature. After the proposal was approved, the attacker activated this self-destruction functionality, destroyed the existing proposal contract, and replaced it with malicious code. The newly inserted code allowed the attacker to withdraw TORN tokens, i.e., the DAO’s governance tokens.

The Tornado Cash incident highlights a general vulnerability in governance mechanisms of decentralized platforms: the lack of a guaranteed match between a proposal’s description and its actual code. Proposals might have unintentional errors or, as in the Tornado Cash case, be subject to deliberate manipulation. Notably, the Tornado Cash attack is not the only example of a malicious mismatch between a proposal’s description and implementation. The proposal of the flash loan attack on Beanstalk (see Section 3.2) claimed to be donating funds to Ukraine but in reality, stole the DAO’s assets [13].

**Proposal Spam (HCI3).** A further attack vector that can be utilized to hide a malicious proposal is to spam the protocol’s governance with many proposals, such that the malicious proposal is hidden in a flood of proposals. One notable example was a governance attack on Synthetify – a protocol on the Solana blockchain whose DAO had been inactive since December 2022. The following case study details the attack, which also involved aspects of token control attacks (see Section 3.2).

#### Case Study Synthetify

On 17 October 2023, an attacker gained access to the assets controlled by Synthetify’s DAO [92, 75, 29]. The attacker first bought sufficient amounts of the protocol’s governance token SNY to make a proposal and to hold more tokens than the three biggest holders. Then the attacker used spam to distract from the attack. In particular, the attacker created more than 20 spam proposals over two months and tested whether they would go unnoticed over the seven-day voting period. No one but the attacker voted on any of these proposals, i.e., the attacker was able to pass them without a problem. Knowing that no one was paying attention, the attacker then hid malicious code that allowed them to withdraw the funds controlled by the governance. The proposal passed without any opposition.

Many protocols attempt to protect against such attacks by only allowing one active proposal per account, which must sufficient tokens to exceed the proposal threshold. Nevertheless, workarounds might still pose a threat to DAOs. Consider the following workaround for DAOs utilizing the delegation model. The attacker creates a proposal with one account to which they delegate their tokens. The attacker waits for the votes to come in and cancels the proposal after a significant proportion of votes have been cast. Then, the attacker delegates the tokens to another account. The attacker then creates a new proposal and continues in this fashion in hopes of tiring the DAO’s voters who pay fees for every vote.

**Social Infiltration (HCI4).** Individuals and institutions can take up positions of power in DAOs. For instance, delegates often vote with significantly more tokens than they hold. Moreover, some DAOs grant certain powers in the governance process to *multisignature addresses (multisig)* which are jointly controlled by multiple key holders. The members of the multisig are chosen and voted upon by the DAO. One can imagine that malicious parties can maneuver themselves into these positions of power and then use their position to attack the protocol. The scandal surrounding Wonderland DAO [113] highlights the potential risk that can stem from social infiltration. The treasury manager was found out to be Michael Patryn, a convicted criminal who had hidden his identity.

**Behavioral Manipulations (HCI5).** Contrary to many voting systems, preliminary results of DAO votes are known to everyone. In a system where voting is associated with high costs, access to interim results could be seen as beneficial, as voters can be mobilized only if needed. However, access to preliminary results also opens up attack vectors. Yaish et al. [119] highlight these attack vectors, which are attested by a large body of work on voting systems and online polls [21, 124, 90, 88, 3].

First, voters might be manipulated not to vote because they observe that their preferred outcome appears to have garnered enough support to win. An attacker can then vote at the last moment, not offering others time to react. This behavioral pattern called *vote sniping* has been reported anecdotally before [93]. Rosello [101] draws parallels to corporate governance and empirically shows the negative effects vote sniping has on token value.

Conversely, attackers voting early might sway uninformed voters to follow their direction. This is commonly referred to as *bandwagon voting*, an effect supported by a large amount of empirical evidence [124, 90, 3]. Yaish et al. [119] analyze this setting theoretically, and show that interim results piled with high voting costs can entice informed voters to follow a mixed strategy of voting either early or late.

### 3.4 Code & Protocol Vulnerability

► **Definition.** *Code and protocol vulnerability (CP)* attacks exploit code or logic vulnerabilities, either in the governance smart contracts or the protocols they are connected to.

**Code Vulnerability (CP1).** To attack a DAO, an attacker can take advantage of any existing bugs in the governance smart contracts. The arguably most prominent attack on a DAO did exactly that.

#### Case Study The DAO

The DAO was a crowd-funded investment fund and one of the first DAOs. On 17 June 2016, an attack on The DAO occurred [106]. The attack exploited a loophole in the code, that allowed the attacker to perform a reentrancy attack to repeatedly withdraw ETH from The DAO [99]. Notably, the hack was so severe that it led to a highly controversial hard fork of the Ethereum blockchain. The majority of the Ethereum community decided to fork the chain to undo the hack's damages. The unaltered version of the chain continues to operate as Ethereum Classic.

The DAO hack highlights the complexities of writing secure governance smart contracts. Given these complexities and the ongoing development of DAOs, code vulnerabilities appear infrequently. However, in some cases, these bugs are identified in audits and fixed before they can be exploited. For instance, in two DAOs (MakerDAO and Keep3R Network) vote tallying could be exploited [107, 95]. In the case of Keep3r Network, the contracts permitted users to re-vote on a proposal but failed to properly subtract the user's previous vote.

Based on audits, the most well-known smart contract vulnerabilities apart from reentrancy and re-vote vulnerabilities include insufficient proposal validation and absence of transfer validation [103]. To prevent code vulnerabilities, re-using audited and time-tested code is typically seen as a good practice. However, mixing and matching code from different sources has caused at least two hacks too [63].

**Protocol Vulnerability (CP2).** Vulnerabilities in the protocols associated with a DAO can extend to the DAO itself given the often intertwined nature of the two. One example of how vulnerabilities in a protocol can affect the DAO is the attack on Mango Markets.

#### Case Study Mango Markets

In October 2022, Avi Heisenberg performed an attack on Mango Markets and its governance [68]. Heisenberg manipulated the price oracle for MNGO, the protocol’s governance token, that allowed him to take out massive loans against the protocol’s treasury which the DAO controls. In doing so, Heisenberg effectively drained the treasury. He went on to create a proposal in the DAO promising to return the majority of the funds if the DAO agreed to repay the protocol’s bad debt. Further, the attacker’s proposal sought to ensure that the token holders could not pursue any legal action against the attacker. The attacker’s proposal did not pass, but the DAO later passed an alternative proposal, leading to part of the funds being returned. The attacker, who publicly identified himself [47] and infamously described the hack as a “highly profitable trading strategy”, was later charged by the US government for his attack [102].

The previously outlined incident exemplifies how the interconnectedness of a protocol and its DAO can pose a risk to the DAO. When such an intertwined nature is wished for or required, it is especially challenging to fully protect against such attacks, as complexity increases and attack vectors are likely unique to each protocol.

## 4 Real-World Incidents & Attacks

In the following, we analyze past attacks and incidents, as well as potential attacks described in audits and papers relating to DAOs. The data set in the paper includes all incidents known to us at the time of writing.<sup>3</sup> We further provide an up-to-date data set under the webpage [daoattacks.ethz.ch](http://daoattacks.ethz.ch) and welcome readers to report any additional or new incidents.

Table 1 lists all (theorized) incidents we analyzed. For each incident, we indicate the date and blockchain on which it occurred. Additionally, for real-world incidents, we indicate the purpose of the attack, whether it was successful, and if it was, the financial damage. Finally, we highlight which of the attack vectors introduced in the previous sections are utilized. We provide a summary for all (theorized) attacks in the full version of our paper [51].

Turning to Table 1, we observe a relatively balanced distribution of attack vectors used in real-world incidents across the four previously introduced categories. Specifically, among the 28 attacks analyzed, 4 utilized at least one attack vector from the BR category, 14 employed TC attack vectors, 9 involved HCI attack vectors, and 9 exploit CP attack vectors.

Table 1 further summarizes critical vulnerabilities of DAOs that were uncovered in academic works, reported to the protocols, or discovered as part of audits. While attacks documented in academic papers and reports span multiple categories, those identified through audits almost exclusively belong to the CP category.

<sup>3</sup> We collected the incidents by searching the web for papers, audits, news articles, blog posts, and tweets that discuss them as well as talking to experts in the field.





We only found a relatively small set of critical vulnerabilities identified by audits, limiting its representativeness. On the other hand, a closer examination of audits that did not uncover critical vulnerabilities reveals a similar skew towards CP attack vectors [103, 91]. Although most DeFi protocols are primarily susceptible to CP attack vectors [123], the governance aspect introduces an array of exceedingly complex attack vectors. These additional attack vectors are often less tangible to analyze and are typically not accounted for in audit processes.

Additionally, it is worth mentioning that a notable portion of attacks (specifically, 8 out of 28) combine multiple attack vectors. This heterogeneous nature of attacks targeting DAOs can make it challenging for DAOs to anticipate and protect against all potential attacks while, at the same time, striving to innovate and develop.

## 5 Risk Factors

Guided by our description and analysis of historical precedence cases, we identify seven risk factors that either directly or indirectly correlate with attacks on DAOs. Further, for a set of 26 DAOs on Ethereum and its Layer 2s, we empirically analyze how vulnerable these DAOs are for each of our identified risk factors in Table 2. These DAOs represent both the biggest DAOs in the Ethereum ecosystem in terms of the size of the treasuries or protocols they govern, along with smaller DAOs. This combination allows us to accurately portray the state of DAOs of all shapes and sizes. Note that we provide a brief description of our data collection in the full version of our paper [51] and open-source the data collection code [81].

**Voter Apathy (RF1).** If token holders do not delegate or vote themselves, it becomes much easier for an attacker to pass malicious proposals. In all but four of the DAOs we empirically analyzed in Table 2, tokens must be delegated before voting. Importantly, when voting takes place, no more delegation is possible. We show the percentage of both delegated tokens voting and the percentage of the total token supply voting on average in the last five votes – a measure of voter apathy. Note that any tokens that are not delegated ahead of the voting period are completely excluded from voting. When regarding the first two columns in Table 2, notice the relatively low participation from the delegated tokens at 34% across the 20 DAOs that require delegation in our data set. While some DAOs have a high participation of more than 81.13% (i.e., Ampleforth), in other DAOs the participation of delegated tokens sits around 1% (i.e., Pooh). Additionally, even more startlingly, of the entire token supply, on average only 5% of tokens participate in the governance across the DAOs we analyzed. We highlight that these low participation rates of (delegated) tokens can be seen as a considerable risk factor, as an attacker can attempt a majority attack, even when holding just a fraction of the tokens.

**High Governance Token Liquidity (RF2).** High governance token liquidity entails the possibility and comparatively low cost of buying or lending the governance token – making the attack vectors in the token control category we presented in Section 3.2 feasible. Table 2 shows that available liquidity on Uniswap V2 and V3 (the two biggest decentralized exchanges on Ethereum in terms of *total value locked (TVL)* [42]). We show the available liquidity as a percentage of (1) the proposal threshold, i.e., the minimum number of tokens required to create a proposal in the governance, (2) the delegated votes, i.e., the number of tokens required to almost guarantee success in the analyzed DAOs, and (3) the average number of tokens voting in the last five governance votes. We observe that for 17 DAOs the available liquidity exceeds the proposal threshold, whereas only for zero and two DAOs the available liquidity exceeds the delegated votes and the average number of votes respectively. While

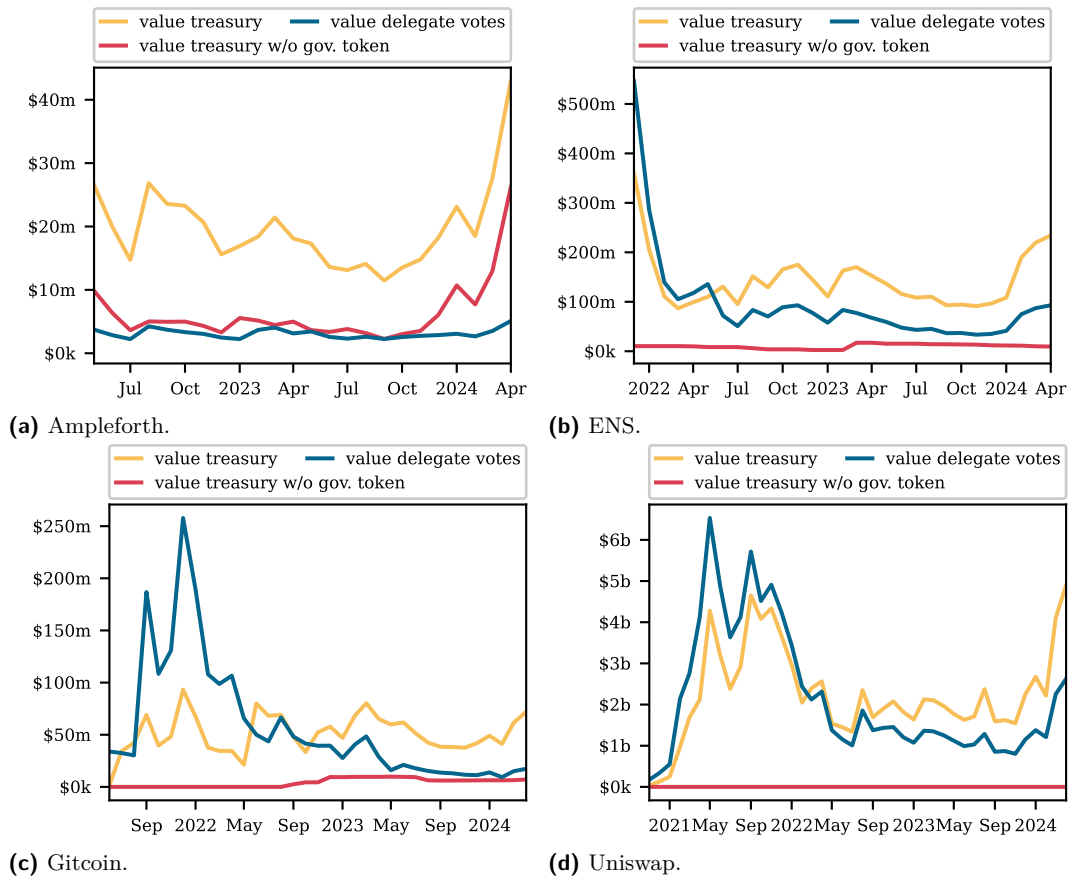
**Table 2** An empirical analysis of the susceptibility of a set of 26 DAOs on the Ethereum blockchain as well as Layer 2s Arbitrum and Optimism to the risk factors presented in Section 5. The data is as of the last block of 31 March 2024 on the respective blockchain and block number delays are indicated according to the underlying chain. Any mention of the average votes refers to an average of the previous five executed votes that were not canceled for each DAO. Additionally, the available liquidity refers to the available liquidity of the respective governance token on Uniswap V2 and Uniswap V3 on Ethereum and Uniswap V3 on Arbitrum and Optimism. Finally, missing entries indicate that the respective risk factor measure does not apply to the DAO, e.g., the risk factor measures related to token delegation are only relevant for DAOs that require delegation.

DAO	voter apathy (RF1)		gov. token liquidity (RF2)		large treasury (RF3)		configuration (RF4)			centralization (RF5)			code (RF6)			
	avg. votes in % of delegated vote	avg. votes in % of token supply	avail. liquidity in % of proposal thresh.	avail. liquidity in % of delegated votes	avail. liquidity in % of avg. votes	treasury value w/ gov. token in % of delegated votes	treasury value w/ gov. token in % of delegated votes	treasury value w/o gov. token in % of delegated votes	proposal delay in blocks	voting period in blocks	timelock delay in blocks	Nakamoto coefficient of delegated votes	Nakamoto coefficient of token supply	number EOAs holding more gov. token than delegated votes	guardian	ownership renounce [25]
Aave	81.13	3.21	8.74	8.64	4.53	837.18	508.76	7,200	72,000	172,800	8	3	4	✓	✓	✓
Ampleforth		4.58	93.55		10.02			13,140	19,710					5	✓	✓
ArbitrumCore		0.85	1,595.42		18.68			21,600	100,800					6	✓	✓
Arbitrum/Treasury		0.72	1,595.42		22.19			21,600	100,800					6	✓	✓
Babylon	32.07	8.71	52.68	1.80	3.02	0.00	0.00	1	45,818	86,400	2	4	0	✓	✓	✓
Braintrust	75.93	0.10	182,186.84	2.92	381.12	29.83	0.00	1	17,280	604,800	1	1	0	✓	✓	✓
Compound	22.19	5.12	300.00	4.26	14.64	1.21	0.94	13,140	19,710	172,800	12	12	0	✓	✓	✓
Cryptex	48.02	5.79	0.01	0.00	0.00	294.88	1.06	1	17,280	259,200	3	3	0	✓	✓	✓
Curve		47.46			2.35				604,800		2	2	0	✓	✓	✓
ENS	33.91	1.47	124.80	3.01	8.49	251.78	10.21	1	45,818	172,800	17	17	0	✓	✓	✓
Fei	18.78	4.83	393.22	9.63	20.67	0.00	0.00	1	13,000	86,400	14	14	0	?	?	?
Gas	54.85	1.08	116.82	18.61	10.78	0.00	0.00	1	45,818	0	2	2	4	✓	✓	✓
Gitcoin	34.31	2.92	217.49	4.02	11.17	416.09	40.89	13,140	40,320	172,800	3	3	1	✓	✓	✓
Hifi	51.80	1.98	216.31	2.11	3.87	0.00	0.00	13,140	36,000	172,800	4	4	3	✓	✓	✓
Hop	24.07	0.42	361.06	22.42	85.17	3,396.69	0.00	1	45,818	172,800	7	7	0	✓	✓	✓
Idle	24.82	7.88	365.90	11.61	46.46	0.00	0.00	100	17,280	172,800	2	6	0	✓	✓	✓
InstaDapp	22.25	4.65	440.24	21.04	94.62	0.00	0.00	7,200	14,400	172,800	3	3	0	✓	✓	✓
Lido		4.11			8.41				21,600		10	10	0	✓	✓	✓
Maker		15.10			20.94						16	16	0	✓	✓	✓
Optimism	35.61	1.11	∞	3.03	5.72	0.00	0.00	1	259,200		10	10	1	✓	✓	✓
Pooh	1.38	0.11	4,545.07	68.55	4,203.75	0.00	0.00	1	50,400	259,200	2	35	0	✓	?	?
Radicle	37.22	3.98	95.61	9.16	24.05	489.50	25.56	1	17,280	172,800	2	2	0	✓	✓	✓
Silo	26.26	2.09	1,183.54	0.97	5.66	161.22	0.00	128	21,000	172,800	3	3	0	✓	✓	✓
Strike	70.15	2.00	39.50	17.00	19.65	0.00	0.00	1	17,280	172,800	1	1	2	✓	✓	✓
Sudoswap	21.50	2.81	425.47	23.51	78.18	274.26	0.00	14,400	21,600		6	6	2	✓	✓	✓
Uniswap	25.14	4.96	325.10	1.60	6.55	186.66	0.00	13,140	40,320	172,800	16	16	0	✓	✓	✓

this appears promising, we underline that the figures presented are a strict lower bound as they for example do not include centralized exchanges where the available liquidity is not easily quantifiable. Even though for the analyzed DAOs liquidity currently appears low, we presented 14 attacks that still attempt to exploit DAOs through token control (see Section 4). Thus, we reiterate that for a DAO's safety, lower liquidity is advantageous.

**Large Treasury (RF3).** The impact and attractiveness of an attack increase, the more value is stored in the treasury. Since in the aftermath of an attack, token prices are expected to plummet, we wager that the treasury value excluding the governance token itself is the most important driving factor. Our empirical analysis in Table 2 presents the treasury value with respect to the value of all delegated tokens, both with and without the governance token. A considerable chunk of DAOs (i.e., 6) hold less than 10% of their treasury value in tokens other than their governance token and are thus likely less at risk for a governance attack that aims to empty the treasury. Startlingly, for the Ampleforth DAO, the value of the treasury without the governance tokens exceeds the value of all delegated tokens – making it an attractive target for attacks. Additionally, we highlight that if the value of the treasury (without the governance token) exceeds 50% of the delegated votes, 51% attacks of token holders that have delegated their tokens could be rational. A few DAOs are close to reaching this threshold (e.g., Gitcoin) or have been in the past. Note that we are not aware of any precedence for such an attack, but protocols have forked before [65]. In addition to the empirical snapshot of 31 January 2024 presented in Table 2, for a smaller subset of DAOs we also visualize the historical value of the treasury in comparison to the delegated token values (see Figure 1). We observe significant fluctuations over time in the relative value of the delegate tokens in comparison to the treasury for the three DAOs: Ampleforth, ENS, Gitcoin, and Uniswap. While initially for three of the DAOs (i.e., ENS, Gitcoin, and Uniswap) the value of the delegate votes (blue line) exceeded the value of the treasury (yellow line) this is no longer the case for all of them. For these DAOs, except for Uniswap (which does not hold tokens other than its governance token), the difference between the value of the delegate votes and the value of the treasury without the governance token is shrinking over time. Finally, for Ampleforth the value of the delegate votes never exceeded the value of the treasury and also currently does not exceed the value of the treasury without the governance token. We conclude that DAOs need to constantly monitor the value of the treasury to ensure that they are not an attractive target for token control attacks.

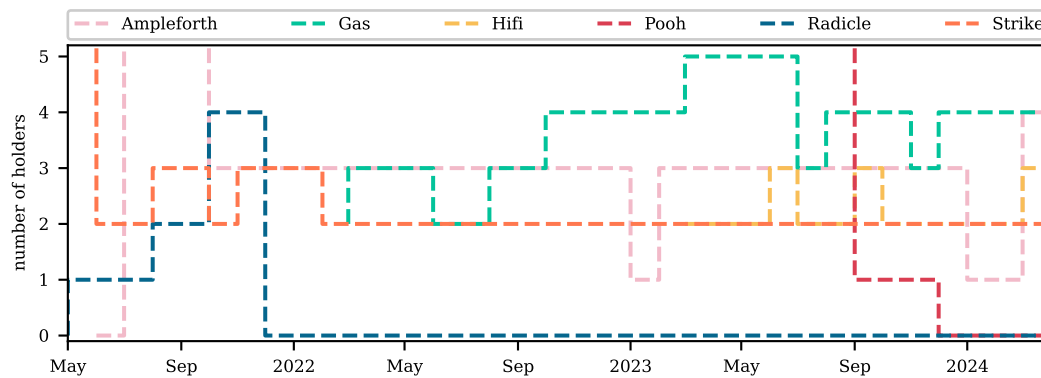
**Inadequate Configuration (RF4).** Inadequate configuration of voting contracts can leave a wide scope of vulnerabilities open. We discuss the most important parameters in the following. First, *proposal delay*, i.e., the delay between proposal creation and the start of the voting period, must be larger than 0 to avoid flash loan attacks. A proposal delay of 1 block, as used by DAOs (see Table 2) is also not without issues though, especially for DAOs that require delegation. Such a small delay does not leave time for non-delegated tokens to be delegated in case of a malicious proposal. For similar reasons, a short *voting window*, might also be dangerous, as delegates might not be reached in time to vote against a malicious proposal. However, all DAOs we analyzed have a voting window that runs for a couple of days (e.g., there are around 7,000 blocks a day on Ethereum). Finally, adjusting the duration a proposal must remain in the timelock can also be beneficial, i.e., *timelock delay*. Extending this period forces an attacker to maintain a number of votes, at least equal to the proposal threshold, for a longer duration. This approach increases the risk for the attacker and makes the potential profits less predictable.



**Figure 1** Comparison of treasury values and the total value of all delegated governance tokens. If the value of the treasury (yellow line) or even the value of the treasury without the governance token (red line) exceeds the value of delegate votes (blue line) this represents an economic risk.

**Centralization (RF5).** If a large (delegated) token supply is held only by a few addresses or entities, many attack vectors become more likely to succeed (e.g., majority coalition, whale activation). In Table 2, we show the Nakamoto coefficient of the delegate votes and the token supply, i.e., the minimum number of addresses collectively holding more than 50% of the delegate votes and the token supply. The lower the Nakamoto coefficient, the higher the centralization. We find that, startlingly, for three DAOs the Nakamoto coefficient of the delegate tokens is one – one delegate has the majority of delegate votes. Finally, we also consider the number of *externally owned addresses (EOAs)* that hold more governance tokens than are currently delegated. Importantly, more than one holder can hold more votes than delegated governance votes, as not all tokens are delegated. These EOAs could delegate their tokens and would have a majority of the delegate tokens. In combination with a small proposal delay (RF4), they could easily acquire the majority of votes. For six DAOs there was at least one EOA that could perform such a 51% attack on 31 March 2024. Additionally, we also analyze how this figure evolves over time for the five DAOs of these DAOs on the Ethereum blockchain in Figure 2. We observe that for these five DAOs, there was generally at least one EOA that held sufficient tokens for a 51% attack and thereby posed a threat.

Table 2 also shows that some DAOs have a guardian in their governance contract or in their timelock contract. This involves special rights that, for example, enable an EOA or a multi-signature wallet to cancel proposals. On the one hand, this functionality can be abused and lead to a situation where only decisions that aren't canceled by the guardian can be



■ **Figure 2** Number of holders, i.e. EAOs, who hold more tokens than delegated governance votes on a monthly basis. These holders would have the majority of the delegated votes after they delegate their tokens.

made, or if not implemented carefully, give the guardian privileged access to the treasury or other critical infrastructure. On the other hand, a trustworthy guardian can mitigate the effect of malicious proposals.

**Code Uncertainties (RF6).** When smart contracts are created, the contract bytecode that is uploaded on-chain can contain arbitrary logic. As smart contracts may contain various unknown mechanisms, any uncertainty can be viewed as risky. Firstly, the smart contract creators should thus publish the source code, allowing anyone to verify its logic. Additionally, some code functionalities are associated with a higher risk. For instance, the presence of a mint functionality might allow an attacker to create more tokens. The mint function can be a particular risk as it allows attackers to empty the liquidity pools with the governance token (see Build Finance and Curio in the Appendix of the full version [51]). We observe in Table 2 that five of the analyzed DAOs implement such functionality in their smart contract. Risks are also associated when external calls are allowed, and when a proxy contract is used (as the proxy contract may be changed to point to a different contract, bypassing the DAO) [31, 16]. Table 2 shows that only for one DAO the ownership of the contract was renounced. This is considered a good practice, as the contract then cannot be called with elevated owner privilege anymore [24].

**Lack of Reliable Communication Channels (RF7).** DAO community members mainly communicate through X (formerly Twitter), Telegram, and Discord. These platforms are crucial parts in defending an attack, as seen in the Indexed Finance case study presented in Section 3.2. Still, it is difficult to reach all delegates and token holders, especially if the projects are no longer active, as was the case for Indexed Finance. Thus, better infrastructure to reliably reach holders and inform them about ongoing governance votes would be beneficial. To the best of our knowledge, none of the DAOs we have analyzed have implemented any more reliable communication channels than those mentioned in the beginning.

The described risk factors are diverse and thus preventing against them all simultaneously is a difficult task. Our empirical evaluation of 26 DAOs and their susceptibility to these risk factors also revealed that smaller DAOs tend to be more at risk. For these DAOs, in the absence of the same resources as their larger counterparts, it is likely especially hard to protect against all possible attack vectors. Thus, especially smaller projects should weigh the benefits and disadvantages of a DAO carefully. For those, that choose a DAO as their governance form we continue by describing and discussing safeguards.

## 6 Mitigation and Safeguards

We present and discuss mitigation strategies to reduce risks. Throughout, we distinguish between mitigations that lower the impact (👊) of an attack and those that lower the likelihood (🎯) of success for an attack. In parentheses, we specify which attack vector categories are targeted.

**Conservative Implementation** 👊 🎯 (BR, TC, HCI, CP). Through conservative implementation, DAOs make sure that exogenous factors cannot be exploited to attack a DAO. Examples include *limiting the number of proposals* that can be made by a single proposer at any given time [116] to prevent spamming attacks and having long enough *proposal delays* (see Section 5). This involves trade-offs, as extending the on-chain proposal process can make an attack less appealing, but it also slows down governance in general. Thus, a balance must be struck between a DAO's agility and safeguarding against potential attacks. We further note that a lack of agility for a DAO can pose additional risks depending on the protocols they govern [66, 67].

**Limiting the Governance Scope** 👊 (BR, TC, HCI, CP). Another approach to lessening the impact of attacks is for a DAO to add restrictions on its action space that can reduce the attack surface. For instance, if the DAO is only granted control over a few parameters, the extent of potential attacks is much narrower. Additionally, one can imagine only allowing a proposal to spend a fixed maximum amount of the treasury.

**Emergency Shutdown** 👊 (BR, TC, HCI, CP). Implementing an emergency shutdown is a very invasive mitigation strategy. Here, a set of holders can halt all transactions. In the case of MakerDAO [84], the emergency shutdown allows token holders to receive a share of the treasury, mitigating potential attacks that were underway.

**Governance Forks** 👊 🎯 (BR, TC, HCI, CP). A similar, but less drastic, safeguard could be achieved through forking, a design primitive where a fraction of token holders can vote to create a fork of the DAO. For instance, The DAO allowed token holders to create *child DAOs* and later withdraw their portion of the DAO deposits from there. Another example of the occurrence of a DAO fork is NounsDAO: A large fraction of holders decided to leave the original DAO for a forked DAO taking with them their proportional share of the treasury [53]. The forked DAO then allowed each token holder to *rage-quit* and retrieve their individual share of the treasury. This process is usually not very fast, and thus can typically only prevent foreseeable attacks. Nonetheless, allowing DAOs to fork is a possibility to prevent a majority (coalition) from taking over a DAO (and its treasury). With a fork, a minority would still have the possibility to take their part of the DAO's assets. However, if a DAO governs more than a fungible treasury, e.g., the parameters of a lending protocol, forking may of course not be a viable option.

**User Authentication** 🎯 (BR, TC). Through user authentication, voting power is to be constrained on a per-person basis. This can enable different voting mechanisms, that might be less vulnerable to token control attacks, such as *quadratic voting* (voting power is proportional to the square root of tokens owned) and *democratic voting* (one person one vote). Examples of user authentication include *know-your-customer (KYC)* or decentralized identifiers, like *Proof-of-Personhood* [15]. The Optimism Governance recently implemented a form of user

authentication. In particular, they implement a bicameral governance design, with a token house [97] (one token one vote) and a citizen house [96] (one person one vote), only those with citizenship can vote in the citizen house.

**Ballot Privacy ? (HCI)** Ensuring ballot/tally privacy during the voting period can help in mitigating behavioral manipulations (HCI5). Cicada [57] is an existing framework for the EVM which achieves this. While it is costly to implement on Ethereum, the costs are more reasonable on L2s.

**Governance Tools ? (HCI).** The development of novel governance tools reduces the hurdles of participation in governance and can help prevent HCI attacks. For instance, through better communication and notifications on current proposals, voter apathy can be combated. Moreover, they may provide the necessary education for voters to be able to make informed decisions more easily, also mitigating behavioral manipulations (HCI5). We believe it is important that these tools are open-source (i.e., such that bugs as in the Tally [110] case are less likely to happen) and that they cannot easily be spammed or taken down. While these tools can do a great part in reducing the load in governance participation, they can become potential attack victims themselves (see Section 3.3).

**Veto Power ? (HCI).** DAOs may also introduce a veto functionality. Through a veto, a small group of holders can prolong a vote, giving the holders more time to counter malicious proposals. Excessive use of veto power itself leads to issues, but we hypothesize that incentives could deter its misuse (e.g., vetoing could be made expensive).

**Objection Phase and Vote Extension ? (HCI).** A more targeted safeguard consists of the addition of a second round of voting (also called *objection phase*), where voters can only vote against the proposal, or change their vote from in favor to against. This has been introduced by Lido [80], with the goal to protect the DAO from vote sniping (see HCI5). Other proposed remedies to vote sniping include the extension of the voting period after high activity (or sway votes) are observed, as well as randomized voting durations. Both have recently been suggested by Decentraland DAO [40].

**Scheduled Voting Windows ? (HCI).** Some protocols are experimenting with votes being scheduled on a regular basis (e.g., once a month). This can prevent proposal spam (HCI3) to reduce voter apathy and dampen the effect of behavioral manipulations (HCI5).

**Escape Hatches ? (CP).** Escape hatches can be added to DAOs to limit the severity of an attack. The *Decentralized Escape Hatch* proposed by Eyal and Sirer [49] for example suggests that outgoing transactions can be buffered (e.g., for 24 hours). Buffered transactions can then be reversed automatically, by specifying programmatic invariants. Such invariants could for example limit the outflow over time, or check whether outflow is consistent with respective inflows. Note that invariants themselves are hard to get right. The authors, thus, also suggest community involvement by crowdsourcing the reversal, for example through a majority involvement.

**Bug Bounties ? ? (CP).** A widespread and important tool to prevent technical attacks is bug bounties. Their extent has been researched in a broader context of cybersecurity and was shown to be a cost-effective instrument [118]. Bug bounties are widespread in the blockchain ecosystem and advertised by several DAOs.

**Audits ? (CP).** Last but not least, audits by external companies can help verify that the DAOs underlying smart contracts are implemented to the state-of-the-art. Audits will make sure that code best practices are respected [32], according to the platform and language used. We observe that audits typically focus on technical vulnerabilities. While we find that they could also consider the more governance-specific attack vectors we present, technical audits also hold immense importance for the security of DAOs.

## 7 Related Work

Possible attacks on DAOs have been discussed in blog posts almost as long as DAOs have existed [86, 37] including by Ethereum’s founder Vitalik Buterin [19, 20]. Among other things, they discuss the risks of low voter participation, centralization, game-theoretic attacks, and vote buying, as well as possible mitigation strategies such as *limited governance*, *non-coin-driven governance*, and *skin in the game*.

An early instance of a DAO governance attack documented in academic literature is a potential attack on the governance of the MakerDAO protocol, the centerpiece of DeFi at the time, by Gudgeon et al. [60]. More recently, Augsten et al. [5] have discussed the potential of hidden vote buying in DAO governance facilitated by smart contracts, i.e., what is referred to as *Dark DAOs*. Related to the attack on DAO governance, the term *Governance Extractable Value (GEV)* has been coined to describe the potential value that can be gained from influencing DAO governance votes [78]. Note that the term is an homage to the widely-studied concept of *Miner/Maximal Extractable Value (MEV)* [36].

Two recent systematizations of knowledge (SoKs) cover topics related to DAO attacks: Zhou et al. [123] survey hacks and incidents in DeFi protocols in general. However, most described attacks are not attacks on the protocol’s governance system, which we focus on in this paper. A general overview and systematization of the concept of governance for blockchains and blockchain-based protocols can be found in the SoK by Kiayias and Lazos [71]. It discusses the governance processes of blockchains such as Bitcoin and Ethereum, along with examples of protocols running on blockchains – which are the focus of our SoK. Additionally, Ethereum’s governance process, including which actors have how much influence on it, has also been studied in detail by Fracassi et al. [55]. To the best of our knowledge, this paper represents the first SoK to study attack vectors, risks, and possible mitigation of attacks on the governance of DAOs.

Recently, the literature surrounding DAOs has rapidly expanded, including two reports of the WEF on DAOs [58, 59]. This encompasses a flurry of empirical studies on a variety of DAOs covering aspects such as token distributions, voting turnout and voting behavior [56, 7, 52, 6, 109, 44, 104, 72, 89]. In particular, many of the studies (see e.g., Feichtinger et al. [52]) make a number of observations relevant to attacks covered in this paper: They reveal that a majority of voting power is often concentrated in the hands of a very small number of holders and delegates. Additionally, they highlight that participation rates in governance votes are frequently low across many DAOs.

The vast majority of DAOs today, including those covered in the aforementioned studies, use simple token voting (one-token-one-vote). An alternative governance model using *vote escrowed* tokens (governance tokens locked for a fixed time period), which is for instance used by Curve and Balancer, is discussed by Lloyd et al. [82].

Finally, Tan et al. [111] describe open research problems surrounding DAOs in fields ranging from computer science and economics to ethics, law, and politics.



## 8 Conclusion

In this paper, we systematically analyzed potential attacks on DAOs along with 28 real-world incidents to illustrate the scope of security vulnerabilities. By describing and categorizing the multitude of attack vectors, we provided a comprehensive overview of the threats faced by DAOs. Additionally, we identified and empirically measured risk factors across a set of 26 DAOs, offering insights into the prevalent risks and their impact.

We believe that it is highly advisable for a DAO to engage early with the possibility of such an attack, to monitor parameters closely, and to ensure that an attack does not become economically attractive. Understanding these challenges is critical when designing and operating a DAO, and poses a significant challenge to DAOs. Ultimately, with our systematization of attacks on DAOs, the vulnerabilities of DAOs, and possible safeguards, we seek to arm future DAO designs with the necessary knowledge to anticipate and mitigate these threats.

---

## References

- 1 Zack Abrams. Indexed dao to distribute remaining treasury after defeating hijack attempts. <https://www.theblock.co/post/264679/indexed-dao-to-distribute-remaining-treasury-after-defeating-hijack-attempts>, 2023.
- 2 AnnabelTUSD. Open letter to the makerdao community from tusd. <https://forum.makerdao.com/t/open-letter-to-the-makerdao-community-from-tusd/12753/1>, 2022.
- 3 Victor Araújo and Malu AC Gatto. Casting ballots when knowing results. *British Journal of Political Science*, 52(4):1709–1727, 2022.
- 4 James Austgen, Andres Fabrega, Sarah Allen, Kushal Babel, Mahimna Kelkar, and Ari Juels. Daos must confront dark daos — or fall under their shadow. <https://initc3org.medium.com/daos-must-confront-dark-daos-or-fall-under-their-shadow-b4c47cb6a1be>, 2024.
- 5 James Austgen, Andrés Fábrega, Sarah Allen, Kushal Babel, Mahimna Kelkar, and Ari Juels. Dao decentralization: Voting-bloc entropy, bribery, and dark daos, 2023. [arXiv:2311.03530](https://arxiv.org/abs/2311.03530).
- 6 Tom Barbereau, Reilly Smethurst, Orestis Papageorgiou, Johannes Sedlmeir, and Gilbert Fridgen. Decentralised finance’s timocratic governance: The distribution and exercise of tokenised voting rights. *Technology in Society*, 73:102251, 2023. [doi:10.1016/j.techsoc.2023.102251](https://doi.org/10.1016/j.techsoc.2023.102251).
- 7 Tom Josua Barbereau, Reilly Smethurst, Orestis Papageorgiou, Alexander Rieger, and Gilbert Fridgen. Defi, not so decentralized: The measured distribution of voting rights. In *Proceedings of the Hawaii International Conference on System Sciences 2022*, page 10, 2022.
- 8 Rob Behnke. Explained: The mochi inu governance hack (november 2021). <https://www.halborn.com/blog/post/explained-the-mochi-inu-governance-hack-november-2021>, 2021.
- 9 Rob Behnke. Explained: The Tornado Cash Hack. <https://www.halborn.com/blog/post/explained-the-tornado-cash-hack-may-2023>, May 2023.
- 10 Jan Behrens. The origins of liquid democracy. *The Liquid Democracy Journal on electronic participation, collective moderation, and voting systems*, 5, May 2017. URL: [https://liquid-democracy-journal.org/issue/5/The\\_Liquid\\_Democracy\\_Journal-Issue005-02-The\\_Origins\\_of\\_Liquid\\_Democracy.html](https://liquid-democracy-journal.org/issue/5/The_Liquid_Democracy_Journal-Issue005-02-The_Origins_of_Liquid_Democracy.html).
- 11 Tom W Bell. Blockchain and authoritarianism: The evolution of decentralized autonomous organizations. In *Blockchain and Public Law*, pages 90–104. Edward Elgar Publishing, 2021.
- 12 BIGCAP. Community alert! this is scam dao proposal. <https://twitter.com/BIGCAPProject/status/1697958233204490494>, 2023. Twitter post.
- 13 Everything Blockchain. Beanstalk Exploit - A Simplified Post-Mortem Analysis. <https://medium.com/coinmonks/beanstalk-exploit-a-simplified-post-mortem-analysis-92e6cdb17ace>, 2022.

## 28:22 SoK: Attacks on DAOs

- 14 BlockSec. Twitter post on temple dao attack. <https://twitter.com/BlockSecTeam/status/1579843881893769222>, 2022.
- 15 Maria Borge, Eleftherios Kokoris-Kogias, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, and Bryan Ford. Proof-of-personhood: Redemocratizing permissionless cryptocurrencies. In *2017 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 23–26. IEEE, 2017.
- 16 Boring Security. All About Proxy Contracts. <https://boringsecurity.com/articles/all-about-proxy-contracts>, 2023.
- 17 James M. Buchanan. Simple majority voting, game theory, and resource use. *Canadian Journal of Economics and Political Science*, 27(3):337–348, 1961. doi:10.2307/139591.
- 18 BuildFinance. Twitter post on governance attack. [https://twitter.com/finance\\_build/status/1493223190071554049](https://twitter.com/finance_build/status/1493223190071554049), 2022.
- 19 Vitalik Buterin. Notes on blockchain governance. <https://vitalik.eth.limo/general/2017/12/17/voting.html>, 2017.
- 20 Vitalik Buterin. Moving beyond coin voting governance. <https://vitalik.eth.limo/general/2021/08/16/voting3.html>, 2021.
- 21 Steven Callander. Bandwagons and momentum in sequential voting. *The Review of Economic Studies*, 74(3):653–684, 2007.
- 22 Certik. Exploiting a smart contract without security vulnerabilities: Analysis of true seigniorage dollar attack event. <https://www.certik.com/resources/blog/exploitingasmartcontractwithoutsecurityvulnerabilitiesanalysisoftrueseignioragedollarattackevent>, 2021.
- 23 Certik. Security Assessment GameDAO. <https://skynet.certik.com/projects/gamedao>, 2021.
- 24 Certik. Securing The Web3 World. <https://www.certik.com/>, 2023.
- 25 Certik. Top DAO Dashboards. <https://skynet.certik.com/boards/dao>, 2024.
- 26 ChainSecurity. Security Audit of POA NETWORK’s Smart Contracts. [https://chainsecurity.com/wp-content/uploads/2019/03/ChainSecurity\\_PoA.pdf](https://chainsecurity.com/wp-content/uploads/2019/03/ChainSecurity_PoA.pdf), 2018.
- 27 ChainSecurity. Code Assessment of the Hoprnnet Token Smart Contracts. [https://cdn.prod.website-files.com/65d35b01a4034b72499019e8/6644c996df51a11845ac7de3\\_210629\\_H0PR-Token\\_Smart-Contract-Audit-Report\\_ChainSecurity\\_compressed.pdf](https://cdn.prod.website-files.com/65d35b01a4034b72499019e8/6644c996df51a11845ac7de3_210629_H0PR-Token_Smart-Contract-Audit-Report_ChainSecurity_compressed.pdf), 2021.
- 28 ChainSecurity. Code Assessment of the Snapshot X Smart Contracts. [https://cdn.prod.website-files.com/65d35b01a4034b72499019e8/6645a5f08d64f89be8ee4856\\_ChainSecurity\\_PoA\\_compressed.pdf](https://cdn.prod.website-files.com/65d35b01a4034b72499019e8/6645a5f08d64f89be8ee4856_ChainSecurity_PoA_compressed.pdf), 2023.
- 29 coinlive. Synthetify suffers \$230,000 loss due to governance failure. <https://www.coinlive.com/news-flash/298994>, 2023.
- 30 Cointelgraph. Hacker drains \$1.08M from Audius following passing of malicious proposal. <https://cointelegraph.com/news/hacker-drains-1-08m-from-audius-following-passing-of-malicious-proposal>, 2022.
- 31 Consensys. Ethereum Smart Contract Best Practices. <https://consensys.github.io/smart-contract-best-practices/development-recommendations/general/external-calls/>, 2023.
- 32 Consensys. Ethereum smart contract best practices. <https://consensys.github.io/smart-contract-best-practices/development-recommendations/>, 2023.
- 33 Tim Copeland. Steem vs tron: The rebellion against a cryptocurrency empire. <https://decrypt.co/38050/steem-steemit-tron-justin-sun-cryptocurrency-war>, 2020.
- 34 Tim Copeland. Build finance dao suffers ‘hostile governance takeover’ loses \$470,000. <https://www.theblock.co/post/134180/build-finance-dao-suffers-hostile-governance-takeover-loses-470000>, 2022.
- 35 Phil Daian. Analysis of the dao exploit. <https://hackingdistributed.com/2016/06/18/analysis-of-the-dao-exploit/>, 2016.

- 36 Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 910–927, 2020. doi:10.1109/SP40000.2020.00040.
- 37 Philip Daian, Tyler Kell, Ian Miers, and Ari Juels. On-chain vote buying and the rise of dark daos. <https://hackingdistributed.com/2018/07/02/on-chain-vote-buying/>, 2018.
- 38 Mike Dalton. Build finance dao suffers governance takeover attack. <https://cryptobriefing.com/build-finance-dao-suffers-governance-takeover-attack/>, 2022.
- 39 Roxana Danila. Responsible vulnerability disclosure. <https://medium.com/nexus-mutual/responsible-vulnerability-disclosure-ece3fe3bcefa>, 2020.
- 40 Decentraland. Change Gov Mechanism to Mitigate Last-Minute Voting in DAO. <https://decentraland.org/governance/proposal/?id=00a79921-2dca-4bde-829e-3a503fc602c2>, 2024.
- 41 DeepDAO. Organizations. <https://deepdao.io/organizations>, 2023.
- 42 Defillama. Dexes tvl rankings. <https://defillama.com/protocols/dexes/Ethereum>, 2023.
- 43 True Seigniorage Dollar. Twitter post on TSD attack. <https://twitter.com/TrueSeigniorage/status/1370956726489415683>, 2021.
- 44 Maya Dotan, Aviv Yaish, Hsin-Chu Yin, Eytan Tsytkin, and Aviv Zohar. The vulnerable nature of decentralized governance in defi. In *Proceedings of the 2023 Workshop on Decentralized Finance and Security, DeFi '23*, pages 25–31, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3605768.3623539.
- 45 Quinn DuPont. Experiments in algorithmic governance: A history and ethnography of “the dao,” a failed decentralized autonomous organization. In *Bitcoin and beyond*, pages 157–177. Routledge, 2017.
- 46 Etherscan. Build Finance. <https://etherscan.io/tx/0xf7709b0587d89b9d9b04ca04ce54fd02a5a30435daf1fb4ba1174486e365c9f>, 2022. Ethereum transaction.
- 47 Avraham Eisenberg. Twitter post on mango markets. [https://twitter.com/avi\\_eisen/status/1581326197241180160](https://twitter.com/avi_eisen/status/1581326197241180160), 2022.
- 48 Etherscan. Yuan Finance. <https://etherscan.io/tx/0x4556acce865abe3304eefc7d055112afdcab0d64f838790b46fa0d6dde189c9b>, 2021. Ethereum transaction.
- 49 Ittay Eyal and Emin Gün Sirer. A Decentralized Escape Hatch for DAOs. <https://hackingdistributed.com/2016/07/11/decentralized-escape-hatches-for-smart-contracts/>, 2016.
- 50 Corin Faife. How to stole an election: BeanStalk DAO \$80million FlashLoan attack study case. <https://blog.verichains.io/p/how-to-stole-an-election-beanstalk>, 2022.
- 51 Rainer Feichtinger, Robin Fritsch, Lioba Heimbach, Yann Vonlanthen, and Roger Wattenhofer. SoK: Attacks on DAOs, 2024. arXiv:2310.19201.
- 52 Rainer Feichtinger, Robin Fritsch, Yann Vonlanthen, and Roger Wattenhofer. The hidden shortcomings of (d)aos – an empirical study of on-chain governance. In *Financial Cryptography and Data Security. FC 2023 International Workshops*, pages 165–185, Cham, 2024. Springer Nature Switzerland.
- 53 Owen Fernau. Nouns NFT Holders Opt To ‘Rage Quit’ Through New Fork. <https://thedefiant.io/nouns-nft-holders-opt-to-rage-quit-through-new-forky>, September 2023.
- 54 Bryan Alexander Ford. Delegative democracy. Technical report, EPFL scientific publications, May 2002. URL: <https://infoscience.epfl.ch/record/265695>.
- 55 Cesare Fracassi, Moazzam Khoja, and Fabian Schär. Decentralized crypto governance? transparency and concentration in ethereum decision-making. *Transparency and Concentration in Ethereum Decision-Making (January 10, 2024)*, 2024.
- 56 Robin Fritsch, Marino Müller, and Roger Wattenhofer. Analyzing voting power in decentralized governance: Who controls daos?, 2022. arXiv:2204.01176.
- 57 Noemi Glaeser, István András Seres, Michael Zhu, and Joseph Bonneau. Cicada: A framework for private non-interactive on-chain auctions and voting. *Cryptology ePrint Archive*, 2023.

- 58 David Gogel, Bianca Kremer, Aiden Slavin, and Kevin Werbach. Decentralized autonomous organizations: Beyond the hype, June 2022. URL: [https://www3.weforum.org/docs/WEF\\_De-centralized\\_Autonomous\\_Organizations\\_Beyond\\_the\\_Hype\\_2022.pdf](https://www3.weforum.org/docs/WEF_De-centralized_Autonomous_Organizations_Beyond_the_Hype_2022.pdf).
- 59 David Gogel, Bianca Kremer, Aiden Slavin, and Kevin Werbach. Decentralized autonomous organization toolkit, January 2023. URL: [https://www3.weforum.org/docs/WEF\\_Decentralized\\_Autonomous\\_Organization\\_Toolkit\\_2023.pdf](https://www3.weforum.org/docs/WEF_Decentralized_Autonomous_Organization_Toolkit_2023.pdf).
- 60 Lewis Gudgeon, Daniel Perez, Dominik Harz, Benjamin Livshits, and Arthur Gervais. The decentralized financial crisis. In *2020 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pages 1–15, 2020. doi:10.1109/CVCBT50464.2020.00005.
- 61 Hacken. DAO Maker Audit Report. <https://hacken.io/audits/dao-maker/>, 2021.
- 62 Hacken. Consitution DAO Smart Contract Code Review and Security Analysis. [https://wp.hacken.io/wp-content/uploads/2022/01/%D0%A1onstitution-DAO\\_11012022Audit\\_Report.pdf](https://wp.hacken.io/wp-content/uploads/2022/01/%D0%A1onstitution-DAO_11012022Audit_Report.pdf), 2022.
- 63 Halborn. Explained: The ForceDAO Hack (April 2021). <https://www.halborn.com/blog/post/explained-the-forcedao-hack-april-2021>, 2021.
- 64 Halborn. Explained: The Curio Hack (March 2024). <https://www.halborn.com/blog/post/explained-the-curio-hack-march-2024>, 2024.
- 65 Andrew Hayward. Nouns Fork: Disgruntled NFT Holders Exit With \$27 Million From Treasury. <https://decrypt.co/197400/nouns-fork-disgruntled-nft-holders-exit-27-million-from-treasury>, 2023.
- 66 Lioba Heimbach, Eric Schertenleib, and Roger Wattenhofer. DeFi Lending During The Merge. In *5th Conference on Advances in Financial Technologies (AFT), Princeton, NJ, USA*, October 2023.
- 67 Lioba Heimbach, Eric Schertenleib, and Roger Wattenhofer. Short Squeeze in DeFi Lending Market: Decentralization in Jeopardy? In *3rd Workshop on Decentralized Finance (DeFi), Bol, Brac, Croatia*, May 2023.
- 68 Louis Husney. Mango markets madness: A case study on the mango markets exploit. <https://infotrend.com/mango-markets-madness-a-case-study-on-the-mango-markets-exploit/>, 2023.
- 69 Jimmy Aki. The curve wars. <https://www.techopedia.com/definition/the-curve-wars>, 2023.
- 70 James S. Jordan. *Majority rule with dollar voting*, pages 211–220. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003. doi:10.1007/978-3-540-24784-5\_13.
- 71 Aggelos Kiyias and Philip Lazos. Sok: Blockchain governance. In *Proceedings of the 4th ACM Conference on Advances in Financial Technologies, AFT '22*, pages 61–73, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3558535.3559794.
- 72 Stefan Kitzler, Stefano Ballelli, Pietro Saggese, Bernhard Haslhofer, and Markus Strohmaier. The governance of decentralized autonomous organizations: A study of contributors' influence, networks, and shifts in voting power, 2023. arXiv:2309.14232.
- 73 Kleros. Kleros Blocks Attack on POH Governor, Saves 46 ETH. [https://typefully.com/Kleros\\_io/5yDM4vb](https://typefully.com/Kleros_io/5yDM4vb), 2023.
- 74 Oliver Knight. Defi protocol temple dao struck by \$2.3m exploit. <https://www.coindesk.com/business/2022/10/11/defi-protocol-temple-dao-struck-by-23m-exploit/>, 2022.
- 75 Jack Kubinec. Dao on solana loses \$230k after 'attack proposal' goes unnoticed. <https://blockworks.co/news/solana-exploit-dao-hacker>, 2023.
- 76 Luh Luh Lan and Loizos Leracleous. Shareholder votes for sale, July 2005. URL: <https://hbr.org/2005/06/shareholder-votes-for-sale>.
- 77 Isabelle Lee. A crypto collective lost \$470,000 after one individual amassed enough tokens to take control of the group's treasury. <https://markets.businessinsider.com/news/currencies/build-finance-dao-treasury-discord-crypto-build-token-metric-2022-2>, 2022.
- 78 Leland Lee and Ariaah Klages-Mundt. Governance extractable value. <https://ournetwork.substack.com/p/our-network-deep-dive-2>, April 2021.

- 79 Adam Levi. A technical analysis of the genesis alpha hack. <https://medium.com/daostack/a-technical-analysis-of-the-genesis-alpha-hack-f8e34433c14b>, 2019.
- 80 Lido. Moving To Two-Phase Voting. <https://blog.lido.fi/moving-to-two-phase-voting/>, 2022.
- 81 Lioba Heimbach. DAO Vulnerability. <https://github.com/liobaheimbach/DAOVulnerability>, 2024.
- 82 Thomas Lloyd, Daire O’Broin, and Martin Harrigan. Emergent outcomes of the vetoken model. In *2023 IEEE International Conference on Omni-layer Intelligent Systems (COINS)*, pages 1–6, 2023. doi:10.1109/COINS57856.2023.10189201.
- 83 LongForWisdom. [Urgent] Flash Loans and securing the Maker Protocol. <https://forum.makerdao.com/t/urgent-flash-loans-and-securing-the-maker-protocol/4901>, 2020.
- 84 Maker. Maker Protocol Emergency Shutdown. <https://docs.makerdao.com/smart-contract-modules/shutdown>, 2023.
- 85 Shaurya Malwa. Binance denies allegations it intends to use users’ uniswap tokens for voting. <https://www.coindesk.com/tech/2022/10/20/binance-denies-allegations-that-it-intends-to-use-users-uniswap-tokens-for-voting/>, 2022.
- 86 Dino Mark, Vlad Zamfir, and Emin Gün Sirer. A call for a temporary moratorium on the dao. <https://hackingdistributed.com/2016/05/27/dao-call-for-moratorium/>, 2016.
- 87 Matt Hussey. What is Snapshot? The Decentralized Voting System. <https://decrypt.co/resources/what-is-snapshot-the-decentralized-voting-system>, 2021.
- 88 Reshef Meir, Kobi Gal, and Maor Tal. Strategic voting in the lab: compromise and leader bias behavior. *Autonomous Agents and Multi-Agent Systems*, 34:1–37, 2020.
- 89 Johnnatan Messias, Vabuk Pahari, Balakrishnan Chandrasekaran, Krishna P. Gummadi, and Patrick Loiseau. Understanding blockchain governance: Analyzing decentralized voting to amend defi smart contracts, 2024. arXiv:2305.17655.
- 90 Rebecca B Morton, Daniel Muller, Lionel Page, and Benno Torgler. Exit polls, turnout, and bandwagon voting: Evidence from a natural experiment. *European Economic Review*, 77:65–81, 2015.
- 91 Konstantin Nekrasov. DAO Voting Vulnerabilities. <https://mixbytes.io/blog/dao-voting-vulnerabilities#rec506108657>, 2023.
- 92 Neodyme. Twitter post on synthetify attack. <https://twitter.com/Neodyme/status/1715149044794655145?s=20>, 2023.
- 93 Evan Van Ness. Aragon vote shows the perils of onchain governance. <https://evanvanness.com/post/184616403861/aragon-vote-shows-the-perils-of-onchain-governance>, 2019.
- 94 Zach Obront. Agora Audit Report. [https://github.com/voteagora/optimism-gov/blob/main/audits/23-05-12\\_zachobront.md](https://github.com/voteagora/optimism-gov/blob/main/audits/23-05-12_zachobront.md), 2023.
- 95 OpenZeppelin Security. Technical Description of Critical Vulnerability in MakerDAO Governance. <https://blog.openzeppelin.com/makerdao-critical-vulnerability>, 2019.
- 96 Optimism. Citizens’ house overview. <https://community.optimism.io/docs/governance/citizens-house/>, 2023.
- 97 Optimism. Token house history. <https://community.optimism.io/docs/governance/token-house-history/>, 2023.
- 98 Paladin. Documentation. <https://doc.paladin.vote/>, 2023.
- 99 Zubin Pratap. Reentrancy Attacks and The DAO Hack. <https://blog.chain.link/reentrancy-attacks-and-the-dao-hack/>, 2022.
- 100 Rikta Mandal. Venus Protocol Prevented Hostile Takeover Attempt. <https://www.cryptotimes.io/2021/09/18/venus-protocol-prevented-hostile-takeover-attempt/>, 2021.
- 101 Romain Rossello. Blockholders and strategic voting in daos’ governance. *Available at SSRN 4706759*, 2024.
- 102 SEC. SEC Charges Avraham Eisenberg with Manipulating Mango Markets’ “Governance Token” to Steal \$116 Million of Crypto Assets. <https://www.sec.gov/news/press-release/2023-13>, 2023.

- 103 Mundus Security. Typical governance vulnerabilities: from DAO building to DAO smart contract audit. <https://mundus.dev/blog/typical-dao-and-governance-smart-contracts-vulnerabilities>, 2023.
- 104 Tanusree Sharma, Yujin Kwon, Kornrapat Pongmala, Henry Wang, Andrew Miller, Dawn Song, and Yang Wang. Unpacking how decentralized autonomous organizations (daos) work in practice, 2023. [arXiv:2304.09822](https://arxiv.org/abs/2304.09822).
- 105 Shashank. Temple dao hack analysis. <https://blog.solidityscan.com/temple-dao-hack-analysis-c96db856322c>, 2022.
- 106 David Siegel. Understanding The DAO Hack. <https://www.coindesk.com/learn/understanding-the-dao-attack/>, 2023.
- 107 Statemind. KP3R Vulnerability Report. <https://statemind.io/blog/kp3r-vulnerability-report>, 2019.
- 108 Sujith Somraaj. Yam Finance Safeguards \$3.1M Treasury From Governance Attack. <https://decrypt.co/104848/yam-finance-safeguards-3-1m-treasury-governance-attack>, 2022.
- 109 Xiaotong Sun, Charalampos Stasinakis, and Georgios Sermpinis. Decentralization illusion in decentralized finance: Evidence from tokenized voting in makerdao polls, 2023. [arXiv:2203.16612](https://arxiv.org/abs/2203.16612).
- 110 Tally. Post mortem and impact summary: Tally voting bug. <https://blog.tally.xyz/post-mortem-and-impact-summary-tally-voting-bug-6a12616ce717?gi=3bda9305d9b9>, 2023.
- 111 Joshua Z. Tan, Tara Merk, Sarah Hubbard, Eliza R. Oak, Joni Pirovich, Ellie Rennie, Rolf Hoefler, Michael Zargham, Jason Potts, Chris Berg, Reuben Youngblom, Primavera De Filippi, Seth Frey, Jeff Strnad, Morshed Mannan, Kelsie Nabben, Silke Noa Elrifai, Jake Hartnell, Benjamin Mako Hill, Alexia Maddox, Woojin Lim, Tobin South, Ari Juels, and Dan Boneh. Open problems in daos, 2023. [arXiv:2310.19201](https://arxiv.org/abs/2310.19201).
- 112 Team Audius. Audius Governance Takeover Post-Mortem 7/23/22. <https://blog.audius.co/article/audius-governance-takeover-post-mortem-7-23-22>, 2022.
- 113 Andrew Thurman. How did a former quadriga exec end up running a defi protocol? wonderland founder explains. <https://www.coindesk.com/tech/2022/01/27/how-did-a-former-quadriga-exec-end-up-running-a-defi-protocol-wonderland-founder-explains/>, 2021.
- 114 Andrew Thurman. Tron’s justin sun accused of ‘governance attack’ on defi lender compound. <https://www.coindesk.com/tech/2022/02/04/trons-justin-sun-accused-of-governance-attack-on-defi-lender-compound/>, 2022.
- 115 TrailOfBits. Curve DAO Security Assessment. <https://github.com/trailofbits/publications/blob/master/reviews/CurveDAO.pdf>, 2020.
- 116 Uniswap. GovernorBravoDelegate. <https://github.com/getty/uniswap-gov/blob/main/contracts/GovernorBravoDelegate.sol>, 2024.
- 117 Vitalik Buterin. DAOs, DACs, DAs and More: An Incomplete Terminology Guide. <https://blog.ethereum.org/2014/05/06/daos-dacs-das-and-more-an-incomplete-terminology-guide>, 2014.
- 118 Thomas Walshe and Andrew Simpson. An empirical study of bug bounty programs. In *2020 IEEE 2nd International Workshop on Intelligent Bug Fixing (IBF)*, pages 35–44. IEEE, 2020.
- 119 Aviv Yaish, Svetlana Abramova, and Rainer Böhme. Strategic vote timing in online elections with public tallies. *arXiv preprint arXiv:2402.09776*, 2024.
- 120 Ryan Youngjoon Yi. Digixdao: A divorce story – a case study for voting systems and cryptonative arbitrage. <https://blog.coinfund.io/digixdao-divorce-story-6ed74b00e2bd>, February 2020.
- 121 Yuan Finance. Yuan Governance Attack Update and Migration Plan. <https://medium.com/yuan-finance/yuan-governance-attack-update-and-migration-plan-3b5d949ab466>, 2021.

- 122 zefram.eth. Twitter post on mochi. <https://twitter.com/boredGenius/status/1458732732540854276>, 2021.
- 123 Liyi Zhou, Xihan Xiong, Jens Ernstberger, Stefanos Chaliasos, Zhipeng Wang, Ye Wang, Kaihua Qin, Roger Wattenhofer, Dawn Song, and Arthur Gervais. Sok: Decentralized finance (defi) attacks. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 2444–2461. IEEE, 2023.
- 124 James Zou, Reshef Meir, and David Parkes. Strategic voting behavior in doodle polls. In *Proceedings of the 18th ACM conference on computer supported cooperative work & social computing*, pages 464–472, 2015.





# Transaction Fee Mechanism Design in a Post-MEV World

**Maryam Bahrani** ✉

Ritual, New York, NY, USA

**Pranav Garimidi** ✉

a16z Crypto, New York, NY, USA

**Tim Roughgarden** ✉

a16z Crypto, New York, NY, USA

Columbia University, New York, NY, USA

---

## Abstract

The incentive-compatibility properties of blockchain transaction fee mechanisms have been investigated with passive block producers that are motivated purely by the net rewards earned at the consensus layer. This paper introduces a model of active block producers that have their own private valuations for blocks (representing, for example, additional value derived from the application layer). The block producer surplus in our model can be interpreted as one of the more common colloquial meanings of the phrase “maximal extractable value (MEV).”

We first prove that transaction fee mechanism design is fundamentally more difficult with active block producers than with passive ones: With active block producers, no non-trivial or approximately welfare-maximizing transaction fee mechanism can be incentive-compatible for both users and block producers. These results can be interpreted as a mathematical justification for augmenting transaction fee mechanisms with additional components such as order flow auctions, block producer competition, trusted hardware, or cryptographic techniques.

We then consider a more fine-grained model of block production that more accurately reflects current practice, in which we distinguish the roles of “searchers” (who actively identify opportunities for value extraction from the application layer and compete for the right to take advantage of them) and “proposers” (who participate directly in the blockchain protocol and make the final choice of the published block). Searchers can effectively act as an “MEV oracle” for a transaction fee mechanism, thereby enlarging the design space. Here, we first consider a TFM that is inspired by how searchers have traditionally been incorporated into the block production process, with each transaction effectively sold off to a searcher through a first-price auction. We then explore the TFM design space with searchers more generally, and design a mechanism that circumvents our impossibility results for TFMs without searchers. Our mechanism (the “SAKA” mechanism) is incentive-compatible (for users, searchers, and the block producer), sybil-proof, and guarantees roughly 50% of the maximum-possible welfare when transaction sizes are small relative to block sizes. We conclude with a matching negative result: even when transaction sizes are small, no DSIC and sybil-proof deterministic TFM can guarantee more than 50% of the maximum-possible welfare.

**2012 ACM Subject Classification** Theory of computation → Computational pricing and auctions; Security and privacy → Distributed systems security

**Keywords and phrases** MEV, Transaction Fee Mechanisms, Auctions

**Digital Object Identifier** 10.4230/LIPIcs.AFT.2024.29

**Related Version** *Full Version:* <https://eprint.iacr.org/2024/331> [7]

**Funding** *Tim Roughgarden:* Research at Columbia University supported in part by NSF awards CCF-2006737 and CNS-2212745, and research awards from the Briger Family Digital Finance Lab and the Center for Digital Finance and Technologies.



© Maryam Bahrani, Pranav Garimidi, and Tim Roughgarden;  
licensed under Creative Commons License CC-BY 4.0

6th Conference on Advances in Financial Technologies (AFT 2024).

Editors: Rainer Böhme and Lucianna Kiffer; Article No. 29; pp. 29:1–29:24

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1 Introduction

### 1.1 Transaction Fee Mechanisms for Allocating Blockspace

Blockchain protocols such as Bitcoin and Ethereum process transactions submitted by users, with each transaction advancing the “state” of the protocol (e.g., the set of Bitcoin UTXOs, or the state of the Ethereum Virtual Machine). Such protocols have finite processing power, so when demand for transaction processing exceeds the available supply, a strict subset of the submitted transactions must be chosen for processing. To encourage the selection of the “most valuable” transactions, the transactions chosen for processing are typically charged a transaction fee. The component of a blockchain protocol responsible for choosing the transactions to process and what to charge for them is called its *transaction fee mechanism (TFM)*.

Previous academic work on TFM design (surveyed in Section 1.5) has focused on the game-theoretic properties of different designs, such as incentive-compatibility from the perspective of users (ideally, with a user motivated to bid its true value for the execution of its transaction), of block producers (ideally, with a block producer motivated to select transactions to process as suggested by the TFM), and of cartels of users and/or block producers. Discussing incentive-compatibility requires defining utility functions for the relevant participants. In most previous works on TFM design (and in this paper), users are modeled as having a private value for transaction inclusion and a quasi-linear utility function (i.e., value enjoyed minus price paid). In previous work – and, crucially, unlike in this work – a block producer was modeled as *passive*, meaning its utility function was the net reward earned (canonically, the unburned portion of the transaction fees paid by users, possibly plus a block reward).

While this model is a natural one for the initial investigation of the basic properties of TFMs, it effectively assumes that block producers are unaware of or unconcerned with the semantics of the transactions that they process – that there is a clean separation between users (who have value only for activity at the application layer) and block producers (who, if passive, care only about payments received at the consensus layer).

### 1.2 MEV and Active Block Producers

It is now commonly accepted that, at least for blockchain protocols that support a decentralized finance (“DeFi”) ecosystem, there are unavoidable interactions between the consensus layer (block producers) and the application layer (users), and specifically with block producers deriving value from the application layer that depends on which transactions they choose to process (and in which order). For a canonical example, consider a transaction that executes a trade on an automated market maker (AMM), exchanging one type of token for another (e.g., USDC for ETH). The spot price of a typical AMM moves with every trade, so by executing such a transaction, a block producer may move the AMM’s spot price out of line with the external market (e.g., on centralized exchanges (CEXs) like Coinbase), thereby opening up an arbitrage opportunity (e.g., buying ETH on a CEX at the going market price and then selling it on an AMM with a larger spot price). The block producer is uniquely positioned to capture this arbitrage opportunity, by executing its own “backrunning” transaction (i.e., a trade in the opposite direction) immediately after the submitted trade transaction.

The first goal of this paper is to generalize the existing models of TFM design in the minimal way that accommodates *active* block producers, meaning block producers with a utility function that depends on both the transactions in a block and the net fees earned.

Specifically, in addition to the standard private valuations for transaction inclusion possessed by users, the block producer will have its own private valuation, which is an abstract function of the block that it publishes. We then assume that a block producer acts to maximize its *block producer surplus (BPS)*, meaning its private value for the published block plus any additional profits (or losses) from fees (or burns). In the interests of a simple but general model, we deliberately avoid microfounding the private valuation function of a block producer or committing to any specifics of the application layer. Our model captures, in particular, canonical on-chain DeFi opportunities such as arbitrage and liquidation opportunities, but a block producer’s valuation can reflect arbitrary preferences, perhaps derived also from off-chain activities (e.g., a bet with a friend that settles on-chain) or subjective considerations.

The extraction of application-layer value by block producers, in DeFi and more generally, was first studied by Daian *et al.* [17] under the name “MEV” (for “maximal extractable value”). At this point, the term has transcended any specific definition – in both the literature and popular discourse, it is used, often informally, to refer to a number of related but different concepts. For a brief survey see Section 1.5.4. We argue that our definition of BPS captures, in a precise way and in a concrete economic model, one of the more common colloquial meanings of the term “MEV.”

### 1.3 The Block Production Supply-Chain

In the first part of this paper, we treat a block producer as a single entity that publishes a block based on the transactions that it is aware of. This would be an accurate model of block production, as carried out by miners in proof-of-work protocols and validators in proof-of-stake protocols, up until a few years ago. More recently, especially in the Ethereum ecosystem, block production has evolved into a more complex process, typically involving “searchers” (who identify opportunities for extraction from the application layer), “builders” (who assemble such opportunities into a valid block), “relays” (who gather blocks from builders and select the most profitable one for the proposer), and “proposers” (who participate directly in the blockchain protocol and make the final choice of the published block), and several others. One interpretation of a block producer in our model is as a vertically integrated party performing the job of all these entities.

In the second part of the paper, we consider a more fine-grained model of the block production process, in which the role of finding MEV extraction opportunities is decoupled from the proposer’s role of participating in consensus and is instead performed by specialized searchers. An interpretation of this model is that the proposer runs an open-source consensus client to collect block rewards, while outsourcing the complicated task of finding MEV opportunities to searchers. This is in the same spirit as *mev-geth*, which was a widely-used Ethereum client written by Flashbots that proposers could run to allow for the submission of both regular transactions by users and wrapped bundles of transactions by searchers.<sup>1</sup> Prior to *mev-geth*, searchers and users were treated equally by proposers and competed with each other for inclusion; among other issues, multiple searchers pursuing the same MEV extraction opportunity would often have their extraction transactions included in a block, with the first such transaction capturing the opportunity and the rest failing (but still paying transaction fees for inclusion and wasting valuable blockspace). *Mev-geth* introduced an explicit auction, upstream from the blockchain’s fee mechanism, in which searchers could compete directly with each other to capture MEV extraction opportunities. Our model can be viewed as formalizing this idea by allowing a TFM to treat searchers and users differently, subject to different rules for inclusion and payment.

---

<sup>1</sup> See <https://github.com/flashbots/mev-geth/blob/master/README.md>.

## 1.4 Overview of Results

Our starting point is the model for transaction fee mechanism design defined in [45]. In this model, each user has a private valuation for the inclusion of a transaction in a block, and submits a bid along with its transaction. As in [45], we consider TFMs that choose the included transactions and payments based solely on the bids of the pending transactions (as opposed to, say, based also on something derived from the semantics of those transactions). A block producer publishes any block that it wants, subject to feasibility (e.g., with the total size of the included transactions respecting some maximum block size). A TFM is said to be *dominant-strategy incentive-compatible (DSIC)* if every user has a dominant (i.e., always-optimal) bidding strategy. The DSIC property is often associated with a good “user experience (UX),” in the sense that each user has an obvious optimal bid. In [45], a TFM was said to be *incentive-compatible for myopic miners (MMIC)* if it expects a block producer to publish a block that maximizes the net fees earned (at the consensus layer). Here, we introduce an analogous definition that accommodates active block producers: We call a TFM *incentive-compatible for block producers (BPIC)* if it expects a block producer to publish a block that maximizes its private valuation plus the net fees earned. An ideal TFM would satisfy, among other properties, both DSIC and BPIC.

### 1.4.1 Vertically Integrated Active Block Producers

We begin with a model in which there are only users and a single (vertically integrated) active block producer, and show that there are fundamental barriers to designing ideal transaction fee mechanisms in this case.

Our first result (Theorem 11) is a proof that with active block producers *no* non-trivial TFM satisfies both DSIC and BPIC, where “non-trivial” means that users must at least in some cases pay a nonzero amount for transaction inclusion<sup>2</sup>. (In contrast, with passive block producers and no MEV, the “tipless mechanism” suggested in [45] is non-trivial and satisfies both DSIC and BPIC; the same is true of the EIP-1559 mechanism of Buterin et al. [12], provided the mechanism’s base fee is not excessively low [45].) In particular, the EIP-1559 and tipless mechanisms fail to satisfy DSIC and BPIC when block producers can be active. Intuitively, for these mechanisms, a user might be motivated to underbid in the hopes of receiving an effective subsidy by the block producer (who may include the transaction anyways, if it derives outside value from it).

Our second result (Theorem 13) formalizes the intuition that TFMs that do not charge non-zero transaction fees – and in particular (by Theorem 11), TFMs that are both DSIC and BPIC – cannot guarantee any approximation of the maximum-possible social welfare. Intuitively, the issue is the lack of alignment between the preferences of users and of the block producer: If a block producer earns no transaction fees from any block, it might choose a block with non-zero private value but only very low-value transactions over one with no private value but very high-value transactions.

### 1.4.2 TFMs with Competitive Searchers

We then consider a more fine-grained model of block production that more accurately reflects current practice, where we distinguish the roles of “searchers” (who actively identify opportunities for value extraction from the application layer and compete for the right to

---

<sup>2</sup> We distinguish this result from surface level connections to previous impossibility theorems in mechanism design in Section 1.5.5

take advantage of them) and “proposers” (who participate directly in the blockchain protocol and make the final choice of the published block). Searchers can effectively act as an “MEV oracle” for a transaction fee mechanism, thereby enlarging the mechanism design space.

In this model, we first consider a TFM that is inspired by how searchers have traditionally been incorporated into the block production process, and specifically by mev-geth (see Section 2.5). Intuitively, this mechanism runs a first-price auction for each transaction among the interested searchers; the winning bid then acts as estimate of the transaction’s MEV, which the TFM can then use to charge prices to users in a way that recovers the DSIC property for users (Theorem 16).

We then explore the TFM design space with searchers more generally, with a focus on good approximate welfare guarantees. Our main contribution here is a mechanism, which we call the SAKA mechanism, which is DSIC for users, DSIC for searchers, BPIC, sybil-proof, and guarantees roughly 50% of the maximum-possible welfare when transaction sizes are small relative to block sizes (as they are in practice); see Theorems 19 and 20. In particular, this combination of guarantees shows that TFMs with searchers can evade impossibility results that apply to TFMs without searchers (such as Theorem 13). We further show in Theorem 21 that, even when transaction sizes are small, no DSIC and sybil-proof deterministic TFM can guarantee more than 50% of the maximum-possible welfare. (By “sybil-proof,” we mean that no user or searcher can ever profit from creating additional user or searcher identities and submitting fake transactions or bundles under those identities.)

## 1.5 Related Work

### 1.5.1 General TFM literature

The model in this paper is closest to the one used by Roughgarden [45] to analyze (with passive block producers) the economic properties of the EIP-1559 mechanism [12], the TFM used currently in the Ethereum blockchain. Precursors to that work (also with passive block producers) include studies of a “monopolistic price” transaction fee mechanism [32, 51] (also considered recently by Nisan [38]), and work of Basu et al. [10] that proposed a more sophisticated variant of that mechanism. There have also been several follow-up works to [45] that use similar models (again, with passive block producers). Chung and Shi [16] proved impossibility results showing that the incentive-compatible guarantees of the EIP-1559 mechanism are in some respects the best possible. There have also been attempts to circumvent this impossibility result by relaxing the notion of incentive compatibility [16, 24], using cryptography [47], considering a Bayesian setting [53], or mixtures of these ideas [49]. Other recent works [20, 33] study the dynamics of the base fee in the EIP-1559 mechanism.

### 1.5.2 MEV-aware mechanism design

There has been much interest among both researchers and practitioners in restructuring the block production supply chain to address MEV [50, 26]. On the academic side, the bulk of these approaches involve cryptographic techniques [29, 34, 52, 11] or changes at the consensus layer [28, 27, 13, 31]. Relatively recently, there have been some initial studies on the impact of economic mechanisms for mitigating MEV such as order-flow auctions [25] and mev-boost [2]; see [40, 43, 6]. In practice, to this point, economic approaches to addressing MEV have been more popular than cryptographic ones; examples include, among others, mev-share [36], UniswapX [3], and MEV Blocker [1]. The model in this paper aims to integrate some of the ideas behind these deployed applications into the existing mathematical frameworks for transaction fee mechanism design.

### 1.5.3 Credible mechanisms

Akbarpour and Li [4] introduce the notion of *credible* mechanisms, where any profitable deviations by the auctioneer can be detected by at least one user. While similar in spirit to the concept of BPIC introduced here (and the special case of MMIC introduced in [45]), there are several important differences. For example, the theory of credible mechanisms assumes fully private communication between bidders and the auctioneer and no communication among bidders, whereas TFM bids are commonly collected from a public mempool. Another difference is that a block producer in our model can manipulate only the allocation rule of a mechanism (as the payment rule is enforced by the blockchain protocol), while in the credible mechanisms framework the auctioneer can also manipulate the payment rule. In a different direction, there is also a line of follow-up work that takes advantage of cryptographic primitives to build credible auctions on the blockchain [22, 19, 15, 21].

### 1.5.4 Defining MEV

Daian et al. [17] introduced the notion of miner/maximal extractable value. They defined MEV as the value that miners or validators could obtain by manipulating the transactions in a block. Since this work, there have been many follow-up works attempting to formalize MEV and analyze its effects in both theory and practice. Attempts to give exact theoretical characterizations of MEV appear in [46, 39, 9, 5]. Broadly, these works define MEV by defining sets of valid transaction sequences and allowing the block producer to maximize their value over these sequences. These definitions are very general, but in exchange have to this point proved analytically intractable. Several empirical papers study the impact and magnitude of MEV using heuristics applied to on-chain data [41, 42, 48]. Another line of work [30, 26, 8] studies MEV in specific contexts, such as for arbitrage in AMMs, in which it is possible to characterize how much MEV can be realized from certain transactions. In particular, Kulkarni et al. [30] give formal statements on how, under different AMM designs, MEV affects the social welfare of the overall system.

### 1.5.5 Impossibility results in mechanism design

The impossibility results in Section 3 may appear superficially related to other such results in mechanism design. For example, the classic Myerson-Satterwaite Theorem [37] states that there is no efficient, individually rational, Bayesian incentive compatible, and budget-balanced mechanism for bilateral trade. Fundamentally, this result is driven by the tension between welfare and budget-balance in the presence of incentive-compatibility constraints on the participants. Our main impossibility result (Theorem 11), meanwhile, is driven by the combination of incentive-compatibility constraints for users (analogous to the usual participants of a mechanism design problem) and also such a constraint for a self-interested party that is tasked with carrying out the allocation rule of the mechanism (the block producer). As such, our setup more closely resembles that of credible mechanisms than more traditional mechanism design settings. In particular, Theorem 11 holds even in the absence of any welfare-maximization or exact budget-balance requirements (a non-zero burning rule in the sense of Section 2.3 is tantamount to relaxing budget-balance).

## 2 Model

This section defines transaction fee mechanisms, the relevant players and their objectives, and the relevant incentive-compatibility notions. Sections 2.1–2.4 describe the basic model (with vertically integrated, active block producers) that is considered in Section 3, while Section 2.5 augments this model with searchers, which play a central role in Sections 4 and 5.

### 2.1 The Players and Their Objectives

#### 2.1.1 Users

Users submit *transactions* to the blockchain protocol. The execution of a transaction updates the state of the protocol (e.g., users' account balances). The rules of the protocol specify whether a given transaction is *valid* (e.g., whether it is accompanied by the required cryptographic signatures). From now on, we assume that all transactions under consideration are valid. Every transaction  $t$  has a publicly known *size*  $s_t$  (e.g., the gas limit of an Ethereum transaction).

We assume that each user submits a single transaction  $t$  and has a nonnegative *valuation*  $v_t$ , denominated in a base currency like USD or ETH, for its execution in the next block. This valuation is *private*, in the sense that it is initially unknown to all other parties. We assume that the utility function of each user – the function that the user acts to maximize – is quasi-linear, meaning that its utility is either 0 (if its transaction is not included in the next block) or  $v_t - p$  (if its transaction is included and it must pay a fee of  $p$ ). We denote the set of transactions submitted to the TFM by  $T$ .

#### 2.1.2 Blocks

A *block* is a finite set of transactions. A *feasible block* is a block that respects any additional constraints imposed by the protocol. For example, if the protocol specifies a maximum block size, then feasible blocks might be defined as those that comprise only valid transactions and also respect the block size limit.

#### 2.1.3 Block producers (BPs)

We consider blockchain protocols for which the contents of each block are selected by a single entity, which we call the *block producer* (*BP*). We focus on the decision-making of the BP that has been chosen at a particular moment in time (perhaps using a proof-of-work or proof-of-stake-based lottery) to produce the next block. We assume that whatever block the BP chooses is in fact published, with all the included transactions finalized and executed.

A BP chooses a block  $B$  from some abstract non-empty set  $\mathcal{B}$  of feasible blocks, called its *blockset*. For example, the set  $\mathcal{B}$  might consist of all the feasible blocks that comprise only transactions that the BP knows about (perhaps from a public mempool, or perhaps from private communications) along with transactions that the BP is in a position to create itself (e.g., a backrunning transaction). As with users, we model the preferences of a BP with a quasi-linear utility function, meaning the difference between its private value for a block (again, denominated in a base currency like USD or ETH) minus the (possibly negative) payment that it must make. Unlike with users, to avoid modeling any details of why a BP might value a block (e.g., due to the extraction of value from the application layer), we allow a BP to have essentially arbitrary preferences over blocks. More formally, we assume that a BP has a private valuation that is an arbitrary (real-valued) function  $v_{BP}$  over blocks, and the BP acts to maximize its *block producer surplus* (*BPS*):

$$\underbrace{v_{BP}(B) + \text{net fees earned.}}_{\text{block producer surplus (BPS)}}$$

### 2.1.4 Holders

The final category of participants, which are non-strategic in our model but relevant for our definition of welfare in Section 2.2, are the holders of the blockchain protocol’s native currency. As we’ll see in Section 2.3, TFMs are in a position to mint or burn this currency, which corresponds to inflation or deflation, respectively. We treat TFM mints and burns as transfers from and to, respectively, the existing holders of this currency. Formally, we define the collective utility function of currency holders to be the net amount of currency burned by a TFM.

## 2.2 Welfare

According to the principle of welfare-maximization, a scarce resource like blockspace should be allocated to maximize the total utility of all the “relevant participants,” which in our case includes the users, the BP, and the currency holders. Because all parties have quasi-linear utility functions and all TFM transfers will be between members of this group (from users to the BP, from the BP to holders, etc.), the welfare of a block is simply the sum of the user and BP valuations for it:

$$\underbrace{W(B) := v_{BP}(B) + \sum_{t \in B} v_t}_{\text{welfare of } B} \tag{1}$$

Holders are assumed to be passive and thus have no valuations to contribute to the sum.<sup>3</sup>

## 2.3 Transaction Fee Mechanisms

The outcome of a transaction fee mechanism is a block to publish and a set of transfers (user payments, burns, etc.) that will be made upon the block’s publication. In line with the preceding literature on TFMs and the currently deployed TFM designs, we assume that each user that creates a transaction  $t$  submits along with it a nonnegative *bid*  $b_t$  (i.e., willingness to pay), and that a TFM bases its transfers on the set of available transactions and the corresponding bids. (The BP submits nothing to the TFM.) A TFM is defined primarily by its *payment* and *burning* rules, which specify the fees paid by users and the burned funds implicitly received by holders (with the BP pocketing the difference).

### 2.3.1 Payment and burning rules

The payment rule specifies the payments made by users in exchange for transaction inclusion.

► **Definition 1** (Payment Rule). *A payment rule is a function  $\mathbf{p}$  that specifies a nonnegative payment  $p_t(B, \mathbf{b})$  for each transaction  $t \in B$  in a block  $B$ , given the bids  $\mathbf{b}$  of all known transactions.*

<sup>3</sup> We stress that the welfare of a block (1) measures the “size of the pie” and says nothing about how this welfare might be split between users, the BP, and holders (i.e., about the size of each slice). Distributional considerations are important, of course, but they are outside the scope of this paper.



The value of  $p_t(B, \mathbf{b})$  indicates the payment from the creator of an included transaction  $t \in B$  to the BP that published that block. (Or, if the rule is randomized, the expected payment.<sup>4</sup>) We consider only *individually rational* payment rules, meaning that  $p_t(B, \mathbf{b}) \leq b_t$  for every included transaction  $t \in B$ . We can interpret  $p_t(B, \mathbf{b})$  as 0 whenever  $t \notin B$ . Finally, we assume that every creator of an included transaction has the funds available to pay its full bid, if necessary (otherwise, the block  $B$  should be considered infeasible).

The burning rule specifies how much money must be burned by a BP along with the publication of a given block.<sup>5</sup>

► **Definition 2** (Burning Rule). *A burning rule is a function  $q$  that specifies a nonnegative burn  $q(B, \mathbf{b})$  for a block  $B$ , given the bids  $\mathbf{b}$  of all known transactions.*

The value of  $q_t(B, \mathbf{b})$  indicates the amount of money burned (i.e., paid to currency holders) by the BP upon publication of the block  $B$ . (Or, if the rule is randomized, the expected amount.)<sup>6</sup> We assume that, after receiving users' payments for the block, the BP has sufficient funds to pay the burn required of the block that it publishes (otherwise, the block  $B$  should be considered infeasible).

We stress that the payment and burning rules of a TFM are hard-wired into a blockchain protocol as part of its code. This is why their arguments – the transactions chosen for execution and their bids, and perhaps (as in [16]) the bids of some additional, not-to-be-executed transactions – must be publicly recorded as part of the blockchain's history. (E.g., late arrivals should be able to reconstruct users' balances, including any payments dictated by a TFM, from this history.) A BP cannot manipulate the payment and burning rules of a TFM, except inasmuch as it can choose which block  $B \in \mathcal{B}$  to publish.

### 2.3.2 Allocation rules

In our model, a BP has unilateral control over the block that it chooses to publish. Thus, a TFM's allocation rule – which specifies the block that should be published, given all of the relevant information – can only be viewed as a recommendation to a BP. Because the (suggested) allocation rule would be carried out by the BP and not by the TFM directly, it can sensibly depend on arguments not known to the TFM (but known to the BP), specifically the BP's valuation  $v_{BP}$  and blockset  $\mathcal{B}$ .

► **Definition 3** (Allocation Rule). *An allocation rule is a function  $\mathbf{x}$  that specifies a block  $\mathbf{x}(\mathbf{b}, v_{BP}, \mathcal{B}) \in \mathcal{B}$ , given the bids  $\mathbf{b}$  of all known transactions, the BP valuation  $v_{BP}$ , and the BP blockset  $\mathcal{B}$ .*

An allocation rule  $\mathbf{x}$  induces per-transaction allocation rules with, for a transaction  $t$ ,  $x_t(\mathbf{b}, v_{BP}, \mathcal{B}) = 1$  if  $t \in \mathbf{x}(\mathbf{b}, v_{BP}, \mathcal{B})$  and 0 otherwise.

► **Definition 4** (Transaction Fee Mechanism (TFM)). *A transaction fee mechanism (TFM) is a triple  $(\mathbf{x}, \mathbf{p}, q)$  in which  $\mathbf{x}$  is a (suggested) allocation rule,  $\mathbf{p}$  is a payment rule, and  $q$  is a burning rule.*

<sup>4</sup> We assume that users and BPs are risk-neutral when interacting with a randomized TFM.

<sup>5</sup> This differs superficially from the formalism in [45], in which a burning rule specifies per-transaction (rather than per-block) transfers from users (rather than the BP) to currency holders. The payment rule here can be interpreted as the sum of the payment and burning rules in [45], and the per-block burning rule here can be interpreted as the sum of the burns of a block's transactions in [45].

<sup>6</sup> An alternative to money-burning that has similar game-theoretic and welfare properties is to transfer  $q(B, \mathbf{b})$  to entities other than the BP, such as a foundation or the producers of future blocks.

A TFM is defined relative to a specific block publishing opportunity. A blockchain protocol is free to use different TFMs for different blocks (e.g., with different base fees), perhaps informed by the blockchain’s past history.

### 2.3.3 Utility functions and BPS revisited

With Definitions 1–4 in place, we can express more precisely the strategy spaces and utility functions introduced in Section 2.1. We begin with an expression for the utility of a user (as a function of its bid) for a TFM’s outcome, under the assumption that the BP always chooses the block suggested by the TFM’s allocation rule.

► **Definition 5** (User Utility Function). *For a TFM  $(\mathbf{x}, \mathbf{p}, q)$ , BP valuation  $v_{BP}$ , BP blockset  $\mathcal{B}$ , and bids  $\mathbf{b}_{-t}$  of other transactions, the utility of the originator of a transaction  $t$  with valuation  $v_t$  and bid  $b_t$  is*

$$u_t(b_t) := v_t \cdot x_t((b_t, \mathbf{b}_{-t}), v_{BP}, \mathcal{B}) - p_t(B, (b_t, \mathbf{b}_{-t})), \quad (2)$$

where  $B := x_t((b_t, \mathbf{b}_{-t}), v_{BP}, \mathcal{B})$ .

In (2), we highlight the dependence of the utility function on the argument that is directly under a user’s control, the bid  $b_t$  submitted with its transaction.

The BP’s utility function, the block producer surplus, is then:

► **Definition 6** (Block Producer Surplus (BPS)). *For a TFM  $(\mathbf{x}, \mathbf{p}, q)$ , BP valuation  $v_{BP}$ , BP blockset  $\mathcal{B}$ , and transaction bids  $\mathbf{b}$ , the block producer surplus of a BP that chooses the block  $B \in \mathcal{B}$  is*

$$u_{BP}(B) := v_{BP}(B) + \sum_{t \in B} p_t(B, \mathbf{b}) - q(B, \mathbf{b}). \quad (3)$$

In (3), we highlight the dependence of the BP’s utility function on the argument that is under its direct control, its choice of a block. The BP’s utility depends on the payment and burning rules of the TFM, but not on its allocation rule (which the BP is free to ignore, if desired).

Finally, the collective utility function of (passive) currency holders for a block  $B$  with transaction bids  $\mathbf{b}$  is  $q(B, \mathbf{b})$ , the amount of currency burned by the BP. (As promised, for a block  $B$ , no matter what the bids and the TFM, the sum of the utilities of users, the BP, and holders is exactly the welfare defined in (1).)

## 2.4 Incentive-Compatible TFMs

In this paper, we focus on two incentive-compatibility notions for TFMs – which, as we’ll see, are already largely incompatible – one for users and one for block producers. We begin with the latter.

### 2.4.1 BPIC TFMs

We assume that a BP will choose a block to maximize its utility function, the BPS (Definition 6). The defining equation (3) shows that, once the payment and burning rules of a TFM are fixed, a BP’s valuation and blockset dictate the unique (up to tie-breaking) BPS-maximizing block for each bid vector. We call an allocation rule *consonant* if, given the payment and burning rules, it instructs a BP to always choose such a block (breaking ties in

an arbitrary but consistent fashion). Because a BP can see all bids after they are submitted, they can also insert their own “fake” transactions along with “shill” bids for them (e.g., to manipulate the payment and/or burning rules of the TFM), we require that a BP is never incentivized to include such shill bids.

► **Definition 7** (Consonant Allocation Rule). *An allocation rule  $\mathbf{x}$  is consonant with the payment and burning rules  $\mathbf{p}$  and  $q$  if:*

(a) *for every BP valuation  $v_{BP}$  and blockset  $\mathcal{B}$ , and for every choice of transaction bids  $\mathbf{b}$ ,*

$$\underbrace{\mathbf{x}(\mathbf{b}, v_{BP}, \mathcal{B})}_{\text{recommended block}} \in \underbrace{\operatorname{argmax}_{B \in \mathcal{B}} \left\{ v_{BP}(B) + \sum_{t \in B} p_t(B, \mathbf{b}) - q(B, \mathbf{b}) \right\}}_{\text{BPS-maximizing block}};$$

(b) *for some fixed total ordering on the blocks of  $\mathcal{B}$ , the rule breaks ties between BPS-maximizing blocks according to this ordering.*

► **Definition 8** (Shill-Proof). *Payment and burning rules  $\mathbf{p}$  and  $q$  are shill-proof if for every BP valuation  $v_{BP}$  and blockset  $\mathcal{B}$ , and for every choice of transaction bids  $\mathbf{b}$ , there is no set  $F$  of fake transactions with shill bids  $\mathbf{b}'$  such that:*

$$\underbrace{\max_{B \in \mathcal{B}} \left\{ v_{BP}(B) + \sum_{t \in B \setminus F} p_t(B, (\mathbf{b}, \mathbf{b}')) - q(B, (\mathbf{b}, \mathbf{b}')) \right\}}_{\text{optimal BPS with shill bids}} > \underbrace{\max_{B \in \mathcal{B}} \left\{ v_{BP}(B) + \sum_{t \in B} p_t(B, \mathbf{b}) - q(B, \mathbf{b}) \right\}}_{\text{optimal BPS without shill bids}}. \quad (4)$$

BPIC TFMs are then precisely those that always instruct a BP to choose a BPS-maximizing block (breaking ties consistently) while also being shill-proof.

► **Definition 9** (Incentive-Compatibility for Block Producers (BPIC)). *A TFM  $(\mathbf{x}, \mathbf{p}, q)$  is incentive-compatible for block producers (BPIC) if:*

(a)  *$\mathbf{x}$  is consonant with  $\mathbf{p}$  and  $q$ ;*

(b)  *$\mathbf{p}$  and  $q$  are shill-proof.*

### 2.4.1.1 DSIC TFMs

Dominant-strategy incentive-compatibility (DSIC) is one way to formalize the idea of a “good user experience (UX)” for TFMs. The condition asserts that every user has an “obviously optimal” bid, meaning a bid that, provided the BP follows the TFM’s allocation rule, is guaranteed to maximize the user’s utility (no matter what other users might be bidding). In the next definition, by a *bidding strategy*, we mean a function  $\sigma$  that maps a valuation to a recommended bid for a user with that valuation.

► **Definition 10** (Dominant-Strategy Incentive-Compatibility (DSIC)). *A TFM  $(\mathbf{x}, \mathbf{p}, q)$  is dominant-strategy incentive-compatible (DSIC) if there is a bidding strategy  $\sigma$  such that, for every BP valuation  $v_{BP}$  and blockset  $\mathcal{B}$ , every user  $i$  with transaction  $t$ , every valuation  $v_t$  for  $i$ , and every choice of other users’ bids  $\mathbf{b}_{-t}$ ,*

$$\underbrace{\sigma(v_t)}_{\text{recommended bid}} \in \underbrace{\operatorname{argmax}_{b_t} \{u_t(b_t)\}}_{\text{utility-maximizing bid}}, \quad (5)$$

where  $u_t$  is defined as in (2).

That is, bidding according to the recommendation of the bidding strategy  $\sigma$  is guaranteed to maximize a user’s utility.<sup>7</sup> This is a strong property: a bidding strategy can depend only on what a user knows (i.e., its private valuation), while the right-hand side of (5) implicitly depends (through (2)) also on the bids of the other users and the BP’s preferences.

Note that the classic EIP-1559 mechanism [12] is no longer BPIC for an active BP, even when the base fee is not excessively low (all the transactions with bid at least the base fee fit into the block). The concern is that an active BP is incentivized to include transactions that bid below the base fee (effectively subsidizing them) if the BP has sufficiently high value for those transactions. The main result of Section 3 (Theorem 11) shows that the difficulty of achieving DSIC and BPIC simultaneously is not particular to the EIP-1559 mechanism: When BPs are active, *no* TFM that charges non-zero user fees can be both DSIC and BPIC. In contrast, for a passive BP the DSIC and BPIC properties can be achieved simultaneously via the *tipless mechanism* [45].

In this work we do not focus on *offchain agreement proofness*, a third incentive-compatibility notion commonly studied in the context of transaction fee mechanisms. We note that our impossibility results (Theorems 11 and 13) apply already to mechanisms that are merely DSIC and BPIC (and not necessarily OCA-proof).

## 2.5 Adding Competitive Searchers

Next we describe the changes to the basic model that are needed in Sections 4 and 5, in which we suppose that block proposers outsource the problem of value extraction to searchers.

### 2.5.1 Searchers and bundles

Searchers submit bundles to the blockchain protocol, where a *bundle* consists of a single user-submitted transaction  $t$  and any additional transactions needed to extract value from the transaction. We interchange between referring to bundles by either  $w$  or  $t^i$ , with  $t^i$  explicitly referencing a bundle that includes transaction  $t$ . We assume that there is a canonical way to extend a transaction with size  $s_t$  into a bundle, and denote by  $s'_t$  the size of the latter (with  $s'_t \geq s_t$ ). For example, if  $t$  represents an AMM trade, the corresponding canonical bundle might include a subsequent backrunning trade. Just as users submit bids with their transactions, searchers submit bids with their bundles. A TFM now takes as input both transactions (with their user bids) and bundles (with their searcher bids), and its allocation, payment, and burning rules can depend on the bids of all users and all searchers. We assume that a TFM can distinguish between transactions and bundles, and can therefore treat them differently (e.g., the payment rule can differ for users and for searchers). Like users, searchers have private nonnegative valuations for bundle inclusion and quasi-linear utility functions. The DSIC condition is defined for searchers exactly as it is for users (Definition 10).

---

<sup>7</sup> The term “DSIC” is often used to refer specifically to mechanisms that satisfy the condition in Definition 10 with the truthful bidding strategy,  $\sigma(v_i) = v_i$ . Any mechanism that is DSIC in the sense of Definition 10 can be transformed into one in which truthful bidding is a dominant strategy, simply by enclosing the mechanism in an outer wrapper that accepts truthful bids, applies the assumed bidding strategy  $\sigma$  to each, and passes on the results to the given DSIC mechanism. (This trick is known as the “Revelation Principle”; see e.g. [44].)

## 2.5.2 Blocks

Blocks can now include both transactions and bundles. Multiple searchers may submit bundles corresponding to the same transaction, but in a feasible block, a given transaction can be included (directly or as part of a bundle) at most once. The inclusion of a bundle that contains a transaction  $t$  necessarily implies the inclusion of  $t$  itself – in this sense, the space of feasible allocations is no longer downward-closed. Equivalently, a block now specifies a set of user-submitted transactions and, for each such transaction  $t$ , the searcher (if any) responsible for the included bundle that contains  $t$ . Users continue to have a private value  $v_t$  for inclusion (whether as part of a bundle or not).

## 2.5.3 Revised incentive-compatibility goals

Thus far, the addition of searchers strictly generalizes the model in Sections 1–4, and so our impossibility results (Theorems 11 and 13) for the basic model apply immediately to it as well.

But the whole point of accommodating a competitive ecosystem of searchers is for proposers (the entities that participate directly in the blockchain protocol) to outsource the specialized task of assembling high-value blocks to searchers. That is, searchers are meant to allow proposers to on the one hand act passively (by simply using the most valuable bundles submitted by searchers) and on the other hand earn almost all of the extractable value (with searchers competing the value of their bundles away to the proposer through the bidding process).<sup>8</sup> Mathematically, with searchers, the idea is that what had been the private valuation  $v_{BP}$  of the (vertically integrated) BP in Section 2.1 is now distributed specifically across the searchers. This interpretation is particularly clear in the additive case – meaning the vertically integrated BP valuation  $v_{BP}(B)$  would have been  $\sum_{t \in B} \mu_t$ , with  $\mu_t$  the value extractable from a transaction  $t$  and no interactions between different transactions – with every searcher that submits a bundle involving transaction  $t$  having a value of  $\mu_t$  for that bundle.<sup>9</sup>

With this interpretation in mind, in the model with searchers, there will be three incentive-compatibility goals: (i) DSIC for users; (ii) DSIC for searchers; and (iii) BPIC for the proposer, assuming that the proposer is passive (i.e., with the all-zero valuation for blocks and with utility equal to the net revenue at the consensus layer, including any payments to it from searchers). In effect, this revised model shatters what had been a vertically integrated BP into a single proposer and a number of searchers, and what had been BPIC (with an active BP) now translates to DSIC for (active) searchers and BPIC for a passive proposer.<sup>10</sup>

<sup>8</sup> See [6] for a rigorous analysis of this idea.

<sup>9</sup> For example, transactions could represent trades on different AMMs, or once-per-block MEV opportunities such as top-of-block CEX-DEX arbitrage or liquidation opportunities (the latter two types modeled via a dummy transaction that has a user bid of zero but non-zero value for searchers).

<sup>10</sup> The combination of (i)–(iii) can technically be achieved by using the tipless mechanism and always ignoring any searchers that might be present. Our interest in Section 4 will be the incentive-compatibility properties of a more interesting TFM that incorporates searchers in a way that resembles current practice; the goal in Section 5 is to design novel TFMs that, in addition to satisfying (i) – (iii) and unlike the searcher-excluding tipless mechanism, guarantee a constant fraction of the maximum-possible welfare.

### 2.5.4 Welfare

With searchers, we redefine the welfare (1) of a block  $B$  to reflect the private valuations of searchers and the fact that the proposer is assumed to have an all-zero valuation:

$$W(B) := \sum_{t \in B_T} v_t + \sum_{w \in B_S} v_w, \quad (6)$$

where  $B_T$  and  $B_S$  denote the transactions and bundles, respectively, in the block  $B$ .

## 3 An Impossibility Result for DSIC and BPIC Mechanisms

### 3.1 Can DSIC and BPIC Be Achieved Simultaneously?

The DSIC property (Definition 10) encodes the idea of a transaction fee mechanism with “good UX,” meaning that bidding is straightforward for users. Given the unilateral power of BPs in typical blockchain protocols, the BPIC property (Definition 9) would seem necessary, absent any additional assumptions, to have any faith that a TFM will be carried out by BPs as intended. One can imagine a long wish list of properties that we’d like a TFM to satisfy; can we at least achieve these two?

The tipless mechanism [45] is an example of a TFM that is DSIC and BPIC in the special case of passive BPs. This TFM is also “non-trivial,” in the sense that users generally pay for the privilege of transaction inclusion. With active BPs, meanwhile, the DSIC and BPIC properties can technically be achieved simultaneously by the following “trivial” TFM: the payment rule  $\mathbf{p}$  and burning rule  $q$  are identically zero, and the allocation rule  $\mathbf{x}$  instructs the BP to choose the feasible block that maximizes its private value (breaking ties in a bid-independent way). This TFM is BPIC by construction, and it is DSIC because a user has no control over whether it is included in the chosen block (it’s either in the BP’s favorite block or it’s not) or its payment (which is always 0).

Thus, the refined version of the key question is:

Does there exist a non-trivial TFM that is DSIC and BPIC with active BPs?

### 3.2 Only Trivial Mechanisms Can Be DSIC and BPIC

The main result of this section is a negative answer to the preceding question. By the *range* of a bidding strategy  $\sigma$ , we mean the set of bid vectors realized by nonnegative valuations:  $\{\sigma(\mathbf{v}) : \mathbf{v} \geq 0\}$ , where  $\sigma(\mathbf{v})$  denotes the componentwise application of  $\sigma$ .

► **Theorem 11** (Impossibility of DSIC, BPIC, Non-Triviality). *If the TFM  $(\mathbf{x}, \mathbf{p}, q)$  is DSIC with bidding strategy  $\sigma$  and BPIC with active block producers, then the payment rule  $\mathbf{p}$  is identically zero on the range of  $\sigma$ .*

The proof of Theorem 11 is quite general and holds even if BPs are restricted to have nonnegative additive valuations and all known transactions have the same size and can be included simultaneously into a single feasible block. We sketch the proof here with details in the full version. Towards a contradiction, let  $(\mathbf{x}, \mathbf{p}, q)$  define a BPIC and DSIC TFM with a non-zero payment rule. Thus assume there is a transaction  $t^*$  and a set of bids  $\mathbf{b} = (b_{t^*}, \mathbf{b}_{-t^*})$  where  $p_{t^*}(B, \mathbf{b}) > 0$  where  $B$  is the BP’s BPS maximizing block for some  $v_{BP}$ . Now define an alternative bid vector  $\mathbf{b}' = (0, \mathbf{b}_{-t^*})$  that is identical to  $\mathbf{b}$  except for  $t^*$  dropping their bid to 0. Since  $0 < p_{t^*}(B, \mathbf{b})$ , for the mechanism to be DSIC, we must have  $\mathbf{x}_t(v_{BP}, \mathbf{b}', \mathcal{B}) = 0$  regardless of  $v_{BP}$ . However, we show that we can define a BP valuation  $\hat{v}_{BP}$  where  $\hat{v}_{BP}(\{t^*\})$  is sufficiently high such that for the mechanism to be BPIC,  $\mathbf{x}_{t^*}(\hat{v}_{BP}, \mathbf{b}', \mathcal{B}) = 1$  even with

$\mathbf{b}'_{t^*} = 0$ . This in turn leads to a contradiction. The technical part of the proof lies in choosing  $\hat{v}_{BP}$  properly to show there is *no* choice of payment and burning rule such that there is a consonant allocation rule where the BP doesn't include  $t^*$  under  $\mathbf{b}'$ .

### 3.2.1 Discussion

The role of an impossibility result like Theorem 11 is to illuminate the most promising paths forward. From it, we learn that our options are (i) constrained; and (ii) already being actively explored by the blockchain research community. Specifically, with active BPs, to design a non-trivial TFM, we must choose from among three options:

1. Give up on “good UX,” at least as it is expressed by the DSIC property.
2. Give up on the BPIC property, presumably compensating with restrictions on block producer behavior (perhaps enforced using, e.g., trusted hardware [23] or cryptographic techniques [14]).
3. Expand the TFM design space, for example by incorporating order flow auctions (e.g., [36]) or block producer competition (e.g., [18]) to expose information about a BP's private valuation to a TFM. We explore this idea further in Sections 4 and 5.

► **Remark 12 (Variations of Theorem 11).** Variations on the proof of Theorem 11 show that the same conclusion holds for:

- (a) BPs that have a non-zero private value for only one block (a very special case of single-minded valuations). This version of the argument does not require the consistent tie-breaking assumption in Definition 7(b).
- (b) Burning rules that need not be nonnegative (i.e., rules that can print money), provided that, for every bid vector  $\mathbf{b}$ , there is a finite lower bound on the minimum-possible burn  $\min_{B \in \mathcal{B}} q(B, \mathbf{b})$ . (This would be the case if, for example, the blockset  $\mathcal{B}$  is finite.)
- (c) Bid spaces and payment rules that need not be nonnegative (i.e., with negative bids and user rebates allowed, subject to individual rationality), provided there is a finite minimum bid  $b_{min} \in (-\infty, 0]$  and that  $p_t(B, \mathbf{b}) = b_{min}$  whenever  $t \in B$  with  $b_t = b_{min}$ . In this case, the argument shows that the payment rule  $\mathbf{p}$  must be identically equal to  $b_{min}$  on the range of  $\sigma$ .

## 3.3 The Welfare Achieved by DSIC and BPIC Mechanisms

Theorem 11 shows that TFMs that are DSIC and BPIC must be “trivial,” in the sense that users are never charged for the privilege of transaction inclusion. The next result formalizes the intuitive consequence that such TFMs may, if both users and the BP follow their recommended actions, produce blocks with welfare arbitrarily worse than the maximum possible. (Recall that the welfare  $W(B)$  of a block  $B$  is defined in expression (1) in Section 2.2.) That is, no approximately welfare-maximizing TFM can be both DSIC and BPIC with active BPs. This result is not entirely trivial because the conclusion of Theorem 11 imposes no restrictions on the burning rule of a TFM.

► **Theorem 13 (Impossibility of DSIC, BPIC, and Non-Trivial Welfare Guarantees).** *Let  $(\mathbf{x}, \mathbf{p}, q)$  denote a TFM that is BPIC and DSIC with bidding strategy  $\sigma$ . For every approximation factor  $\rho > 0$ , there exists a BP valuation  $v_{BP}$ , BP blockset  $\mathcal{B}$ , block  $B^* \in \mathcal{B}$ , and transactions with corresponding user valuations  $\mathbf{v}$  such that*

$$W(B) \leq \rho \cdot W(B^*),$$

where  $B = \mathbf{x}(\sigma(\mathbf{v}), v_{BP}, \mathcal{B})$ .

In the absence of a burning rule, Theorem 13 follows directly from Theorem 11, since any mechanism with  $\mathbf{p} = 0$  effectively ignores user bids when choosing a block. However, it's not immediately obvious that there is no burning rule that can entice the BP to pick a welfare maximizing block even while ignoring users' payments. We show that DSIC rules such mechanisms out since a user being able to affect the burning rule and hence allocation rule while having their payment fixed to 0 would give them an incentive to misreport their value.

► **Remark 14 (Generalizations of Theorem 13).** The proof of Theorem 13 shows that the result holds already with BPs that have additive or single-minded valuations. (As discussed in Remark 12, Theorem 11 holds in both these cases, and the BP valuation  $v_{BP}$  used in the proof of Theorem 13 is both additive and single-minded). A slight variation of the proof shows that the result holds more generally for DSIC and BPIC TFMs that use a not-always-nonnegative burning rule, under the same condition as in Remark 12(b).

## 4 Transaction Fee Mechanisms with Searchers

### 4.1 Incorporating Searchers

The impossibility results in Section 3 are consistent with practice, in the sense that modern attempts to mitigate the negative consequence of MEV through economic mechanisms generally lie outside the basic design space of TFMs introduced in Sections 1–4. The dominant such mechanisms distribute the task of block production across multiple parties; in this section and the next, we adopt the model described in Section 2.5, which captures some of this complexity through the addition of searchers that can submit bundles (of a user-submitted transaction together with the searcher's value-extracting transactions) to a TFM. Recall from Section 2.5 that, in this model, what had been the private valuation  $v_{BP}$  of a vertically integrated BP is effectively distributed across a set of searchers, with the block proposer, having outsourced the task of value extraction, then acting passively to maximize its revenue (including the payments from searchers for included bundles). The winning bid of a searcher can be interpreted as an “MEV oracle” that provides a TFM with an estimate of the value that can be extracted from the bundled transaction. In this sense, the TFM design space with searchers is richer than the basic model with users only, and there is hope that a TFM can take advantage of such estimates to define payments for user-submitted transactions in a DSIC-respecting way (e.g., with searchers' bids leading in some cases to user refunds). Indeed, we'll see that this expanded design space allows for positive results that would be impossible in the basic model.

In this section, we propose an abstraction of how searchers have traditionally been incorporated into the block production process, inspired specifically by *mev-geth* (see Section 2.5), and study the incentive-compatibility properties of the resulting mechanism. Section 5 explores the TFM design space with searchers more generally, with a focus on welfare guarantees.

### 4.2 The *s*-Tipless Mechanism

We next introduce the *Searcher Augmented Tipless Mechanism* (*s-tipless mechanism*). Like the EIP-1559 and tipless mechanisms, it has a fixed base fee  $r$  that is charged per unit size. Intuitively, for each user-submitted transaction  $t$ , the mechanism runs a first-price auction among the interested searchers; such an auction is often referred to as an “order-flow auction.” (Thus, the mechanism does not attempt to be DSIC for searchers.) If the winning bid  $b_w$



in this auction is high enough to pay the base fee charges (i.e.,  $b_w \geq r \cdot s'_t$ , where  $s'_t$  is the size of a bundle that contains  $t$ ), then  $w$ 's bundle is included in the block and  $w$  pays its bid (while the user that submitted  $t$  pays nothing). If the winning searcher bid is less than  $r \cdot s'_t$  then, if the user that submitted  $t$  bids at least the relevant base fee charges (i.e.,  $b_t \geq r \cdot s_t$ ), the transaction  $t$  is included in the block and the submitting user pays  $r \cdot s_t$ . In either case, all base fee revenues ( $r \cdot s_t$  or  $r \cdot s'_t$ ) are burned. (The block proposer may still collect revenue from the first-price auction among searchers if the winning bid exceeds  $r \cdot s'_t$ .) In effect, searchers can cover base fee charges for a user if their transaction is sufficiently valuable for them.

► **Definition 15** (Searcher-Augmented Tipless Mechanism (s-tipless mechanism)). *Fix a base fee  $r \geq 0$ :*

(a) **Allocation rule:** *A transaction should be included if either it clears its base fee, or it has a bundle that clears the bundle's base fee. If multiple bundles for a transaction clear the base fee, the bundle with the highest bid should be included. For each  $t \in T$ , let  $S_t$  denote the submitted bundles that contain  $t$ ,  $w$  a generic such bundle, and  $t^* = \operatorname{argmax}_{w \in S_t} \{b_w\}$ . Define*

$$S^* = \{t^* : t \in T, b_{t^*} \geq r \cdot s'_t\} \text{ and } T^* = \{t \in T : b_t \geq r \cdot s_t \vee S_t \cap S^* \neq \emptyset\},$$

and the allocation rule by

$$\mathbf{x}(\mathbf{b}, \mathcal{B}) = T^* \cup S^*.$$

(b) **Payment rule:**

*For all transactions  $t$  in a block  $B$ :*

$$p_t(B, \mathbf{b}) = \begin{cases} 0 & \text{if } S_t \cap B \neq \emptyset \\ r \cdot s_t & \text{otherwise.} \end{cases}$$

*For all bundles  $w$  in a block  $B$ :*

$$p_w(B, \mathbf{b}) = b_w.$$

(c) **Burning rule:** *For a block  $B$  with transactions  $B_T$  and bundles  $B_S$ ,*<sup>11</sup>

$$q(B, \mathbf{b}) = \sum_{t \in B_T} r \cdot s_t + \sum_{w \in B_S} r \cdot (s'_t - s_t).$$

In Definition 15 and Theorem 16 below, we assume for simplicity that the base fee  $r$  is large enough that there is sufficient room in the block for all of the transactions that the mechanism would like to include (i.e., all transactions for which either the user or some searcher is willing to cover the relevant base fee charges). In practice, ala the EIP-1559 mechanism, the base fee  $r$  would generally be adjusted by local search so that this property typically holds. Definition 15 and Theorem 16 can be extended to the general case (with contention between sufficiently high-bidding transactions and bundles) by redefining the allocation rule to maximize the total revenue (i.e.,  $\sum_{t^i \in B_S} (b_{t^i} - r \cdot s'_t)$ ), breaking ties in a consistent fashion.

<sup>11</sup>We subtract  $s_t$  for every bundle  $w \in B_S$  as to not double count  $s_t$  both as part of a bundle and as a standalone transaction

► **Theorem 16.** *The  $s$ -tipless mechanism is DSIC for users and BPIC.*

We give a sketch of the proof here with the details in the full version. To see that the mechanism is DSIC for users, note that if a transaction has a bundle included for it, then it always pays 0 regardless of what it bids, trivially giving the user a dominant bidding strategy. Otherwise, the user faces a fixed price for inclusion and hence has a dominant strategy to only bid above that price if their value is above it. To see that the mechanism is BPIC, note that the only revenue the BP gets is from searcher bids above the basefee. Standalone transactions have no net effect on the BP's BPS. Thus any allocation rule that picks the highest bid searchers above the base fee and picks transactions clearing the base fee is consonant. Furthermore, since the amount searchers pay is only a function of their own bids, the BP has no way to increase their BPS via inserting shill bids.

## 5 Welfare Guarantees

This section continues to investigate transaction fee mechanism design in the presence of searchers, as in the model in Section 2.5. While the previous section proposed abstractions for some of the economic mechanisms that are currently used in practice, this section zooms out and explores the expanded design space more generally.

### 5.1 What Do We Want from a TFM?

Starting from a blank page, we naturally want to design a mechanism that scores well with respect to all the criteria we have considered thus far:

- (P1) DSIC for users;
- (P2) DSIC for (active) searchers;
- (P3) BPIC (with a passive block proposer);
- (P4) good welfare guarantees.

Without searchers, Theorem 13 shows that the combination of (P1), (P3), and (P4) is unachievable. We also noted in passing (footnote 10) that the tipless mechanism, modified to always ignore searchers, satisfies (P1)–(P3). (Such a mechanism can obviously lead to a highly welfare-suboptimal outcome when the valuations of searchers are significantly bigger than those of the users.)

Given the welfare-maximization goal (P4), one obvious starting point is the Vickrey-Clarke-Groves (VCG) mechanism, which in this context would accept bids from all users and searchers, output a feasible block that maximizes the social welfare (6) (taking users' and searchers' bids at face value), and charge each user or searcher its externality (i.e., what the maximum social welfare would have been had that user or searcher been absent). As always, the VCG mechanism is DSIC (in this case, for both users and searchers) and maximizes the social welfare at its dominant-strategy equilibrium. It does not, however, satisfy property (P3). For example, even with only one user-submitted transaction and a number of corresponding searchers (i.e., a second-price auction), the block proposer is generally incentivized to masquerade as a searcher and insert a shill bid (just below the highest searcher bid) to increase its revenue.<sup>12</sup>

---

<sup>12</sup>A similar problem would arise if the  $s$ -tipless mechanism in Section 4 were defined with second-price rather than first-price searcher auctions.

One easy way to turn the VCG mechanism – or really, any TFM with a passive block proposer – into a BPIC mechanism is to always burn all the payments made by users and searchers. The block proposer would then be indifferent over blocks and willing to carry out an arbitrary allocation rule. An extension of this idea that attempts to trade welfare for a non-zero amount of BP revenue would be to use bidder-specific reserve prices (like  $r \cdot s_t$  and  $r \cdot s'_t$  in the  $s$ -tipless mechanism) that don't get burned.<sup>13</sup>

Summarizing, the VCG mechanism with all payments burned satisfies all of (P1)–(P4), and in particular shows that the addition of searchers allows TFMs to circumvent the impossibility result in Theorem 13. Should we declare victory?

## 5.2 Sybil-Proof Mechanisms

In a permissionless blockchain protocol like Bitcoin or Ethereum, it is easy to generate multiple identities in an undetectable way. For example, a user can easily participate as a “fake searcher” in a TFM if it so chooses. This challenge of “sybils,” especially in tandem with the non-downward-closed nature of the set of feasible blocks (with inclusion of a bundle implying inclusion of the corresponding transaction), renders the VCG mechanism extremely easy to manipulate (despite being DSIC for users and searchers separately).

For example, consider a sample instance with a block size of  $k$  where all the transactions and bundles are unit sized and there is one searcher per transaction, i.e.  $\forall t \in T, s_t = s'_t = 1$  and  $S_t = \{t^*\}$ . In this case, the VCG mechanism will include the transactions and bundles corresponding to the  $k$  highest values of  $b_t + b_{t^*}$ . Let the  $(k + 1)$ th-highest of these values be  $r$ . The included user and searcher for transaction  $t$  would then pay  $\max\{r - b_{t^*}, 0\}$  and  $\max\{r - b_t, 0\}$  respectively. In the case that both  $b_t \geq r$  and  $b_{t^*} \geq r$  it follows that neither the user nor searcher has to pay anything at all. Hence there is a clear incentive for a user to deviate by making their bid arbitrarily high and including an arbitrarily high searcher bid for their transaction to get included without paying anything. Even a user with only  $\epsilon$  value for inclusion has an incentive to do this. It follows that users engaging in such manipulations can cause the mechanism to produce outcomes with arbitrarily bad welfare. Furthermore, in permissionless blockchains, such manipulations are easy to carry out. This motivates seeking out TFMs that are, among other properties, “sybil-proof” in some sense.

Our definition of sybil-proofness (for users and searchers) mirrors our definition of BPIC, in that it asserts that the party in question cannot increase their utility through the submission of fake transactions and shill bids.

► **Definition 17** (Sybil-Proofness). *A mechanism is sybil-proof if for every agent  $t$  and every vector of bids  $\mathbf{b}'$ , there exists some bid  $b_t$  such that  $u_t(b_t) \geq \bar{u}_t(\mathbf{b}')$  where  $\bar{u}_t(\cdot)$  is the agent's net utility across the multiple transactions and/or bundles they submitted.*

Intuitively, this definition asserts that a user or searcher should never earn more utility from submitting multiple bids than they could have through a single bid for their transaction or bundle.

We now augment our previous desiderata with:

**(P5)** sybil-proof.

Next we provide a TFM that satisfies the full set (P1)–(P5) of desired properties.

<sup>13</sup> A mechanism with any non-zero reserve prices cannot offer any worst-case approximate welfare guarantees: for all the mechanism knows, only one participant has a non-zero valuation, which is just below the mechanism's non-zero reserve price for that participant. We leave a Bayesian analysis (e.g., with the choice of reserve prices informed by historical bidding data) of the revenue-welfare trade-offs of such mechanisms to future work.

### 5.3 The Searcher-Augmented Knapsack Auction

We will consider a mechanism that chooses which transactions and bundles to include based on their bid-to-size ratios. For ease of exposition, we assume that these ratios are distinct. (This assumption can be removed through standard lexicographic tie-breaking.) The mechanism finds a threshold ratio such that all transactions and bundles that have bid-to-size ratios above this threshold can fit into the block. This ratio is then used as a per-size price charged to included transactions and bundles. Similarly to the s-tipless mechanism, an included bundle pays all the costs for its corresponding transaction. For included bundles, in the case that the second highest bundle bid for a transaction is greater than the threshold payment, the winning searcher pays the second-highest bid instead. Finally, the burning rule is set to be the sum of users' and searchers' payments so that the block proposer always receives zero BPS.

► **Definition 18** (Searcher-Augmented Knapsack Auction (SAKA)).

(a) **Allocation rule:** Recall that  $t^*$  denotes the bundle with the highest bid for transaction  $t$ . For a given  $\mu$ , let

$$S^\mu = \{t^* : t \in T, b_{t^*}/s'_t \geq \mu\} \text{ and } T^\mu = \{t \in T : b_t/s_t \geq \mu \vee S_t \cap S^\mu \neq \emptyset\}.$$

Then let  $B^\mu = T^\mu \cup S^\mu$  be the block consisting of all transactions and bundles that have a bid-to-size ratio of at least  $\mu$ .<sup>14</sup>

Define  $\mu^* := \inf\{\mu : \sum_{t \in B_T^\mu} s_t + \sum_{t^i \in B_S^\mu} (s'_t - s_t) \leq k\}$ , where  $B_T^\mu$  and  $B_S^\mu$  denote the transactions and bundles, respectively, in the block  $B^\mu$ . Then,

$$x(\mathbf{b}, \mathcal{B}) = B^{\mu^*}.$$

(b) **Payment rule:** Define  $b_{t'} := \max_{t^i \in S^t, t^i \neq t^*} \{b_{t^i}\}$  as the second-highest bundle bid for transaction  $t$ . (If there is no such bid, interpret  $b_{t'}$  as 0.) For  $t \in B_T$ :

$$p_t(B, \mathbf{b}) = \begin{cases} 0 & \text{if } S_t \cap B \neq \emptyset \\ \mu^* \cdot s_t & \text{otherwise.} \end{cases}$$

For  $t^i \in B_S$ :

$$p_{t^i}(B, \mathbf{b}) = \max\{\mu^* \cdot s'_t, b_{t'}\}.$$

(c) **Burning rule:**

$$q(B, \mathbf{b}) = \sum_{t \in B_T} p_t(B, \mathbf{b}) + \sum_{w \in B_S} p_w(B, \mathbf{b}).$$

### 5.4 Analysis

We consider the incentive-compatibility properties of the SAKA mechanism in Theorem 19 and its welfare guarantee in Theorem 20. We conclude with Theorem 21, which shows that the welfare guarantee in Theorem 20 is near-optimal among TFMs that satisfy properties (P1)–(P5).

<sup>14</sup>Subject to the usual constraint that each transaction is included (by itself or as part of a bundle) at most once.

► **Theorem 19.** *The Searcher-Augmented Knapsack Auction (SAKA) mechanism is DSIC for both users and searchers, BPIC, and sybil-proof.*

We give the main ideas of the proof here with details in the full version. SAKA being BPIC follows immediately from the burning rule. To see that the mechanism is DSIC, we can focus on the case where a transaction doesn't have a bundle included for it (otherwise the transaction always pays 0). The allocation rule is monotone since once a user's bid clears  $\mu^* \cdot s_t$  they will always be included. Furthermore,  $\mu^* \cdot s_t$  is the minimal amount  $t$  can bid to be included since otherwise bidding below  $\mu^* \cdot s_t$  and being included would contradict the definition of  $\mu^*$ . The case for searcher DSIC follows identically with the addition of needing to pay at least the second highest searcher bid to still be included. Sybil-proofness follows from the fact that  $\mu^*$  is weakly increasing in the number of bids. So users and searchers have no way to decrease their payment by bidding on fake transactions.<sup>15</sup>

We parameterize the mechanism's welfare guarantee by the maximum fraction  $\gamma$  of a block's capacity that is consumed by a single transaction or bundle. (In many blockchain protocols,  $\gamma$  is typically 2% or less.)

► **Theorem 20.** *Assuming truthful bids by users and searchers, the outcome of the SAKA mechanism has social welfare at least  $(1 - \gamma)/2$  times the maximum possible welfare.*

Note that SAKA implements a greedy knapsack algorithm, except it scores bundles using a scoring rule of  $\frac{v_t^*}{s_t}$  instead of  $\frac{v_t + v_t^*}{s_t}$  as it would optimally. However, we either have  $\frac{v_t^*}{s_t} \geq \frac{v_t + v_t^*}{2s_t}$  or  $\frac{v_t^*}{s_t} < \frac{v_t + v_t^*}{2s_t} \implies \frac{v_t}{s_t} > \frac{v_t + v_t^*}{2s_t}$ . It follows that the density SAKA assigns to a bundle is either at least half of what it should be or that the bundle's corresponding transaction carries half the bundle's true density by itself. Since the mechanism includes transactions and bundles with the highest densities, it follows that a bundle being left out because its density was misjudged would be replaced with transactions and/or bundles with at least half its density. Since the greedy algorithm will fill up at least  $1 - \gamma$  of the block limit, this implies an approximation factor of  $\frac{1 - \gamma}{2}$ . The details can be found in the full version.

Our final result shows that, modulo the factor of  $1 - \gamma$  – which, as discussed above, is typically close to 1 in our context – the welfare approximation guarantee in Theorem 20 is optimal among deterministic mechanisms that are both DSIC (for users and searchers) and sybil-proof in the sense of Definition 17. The key insight is that it's difficult to split the payment between user and searcher when a bundle is included due to the bundle requiring its corresponding transaction's inclusion. In particular we show DSIC + sybil-proofness implies user/searcher pairs can only be included based on the max of their values rather than the sum of their values (as would be optimal). We leave the details to the full version.

► **Theorem 21.** *No deterministic mechanism that is DSIC for users and searchers and sybil-proof can achieve better than a  $1/2$ -approximation to the optimal social welfare, even when transaction sizes are a negligible fraction of the block size.*

---

## References

- 1 Mev blocker. <https://mevblocker.io/>, 2024.
- 2 What is mev-boost. <https://docs.flashbots.net/flashbots-mev-boost/introduction>, 2024.

---

<sup>15</sup>As a bonus, the SAKA auction can be implemented as a deferred acceptance mechanism [35] and is therefore also robust to certain forms of collusion between users and searchers (formally, the mechanism is weakly groupstrategyproof).

- 3 Hayden Adams, Emily Williams, Will Pote, Zhiyuan Yang, Noah Zinsmeister, Xin Wan, Allen Lin, Riley Campbell, Dan Robinson, Mark Toda, Matteo Leibowitz, Eric Zhong, and Alex Karys. Uniswapx protocol, July 2023. Available at Uniswap.org.
- 4 Mohammad Akbarpour and Shengwu Li. Credible auctions: A trilemma. *Econometrica*, 88(2):425–467, 2020.
- 5 Kushal Babel, Philip Daian, Mahimna Kelkar, and Ari Juels. Clockwork finance: Automated analysis of economic security in smart contracts. In *IEEE Symposium on Security and Privacy*, pages 2499–2516, 2023.
- 6 Maryam Bahrani, Pranav Garimidi, and Tim Roughgarden. Centralization in block building and proposer-builder separation. In Jeremy Clark and Elaine Shi, editors, *Financial Cryptography and Data Security - 28th International Conference, FC 2024, Curaçao, March 4-8, 2024*, 2024.
- 7 Maryam Bahrani, Pranav Garimidi, and Tim Roughgarden. Transaction fee mechanism design in a post-mev world. *IACR Cryptol. ePrint Arch.*, page 331, 2024. URL: <https://eprint.iacr.org/2024/331>.
- 8 Massimo Bartoletti, James Hsin-yu Chiang, and Alberto Lluch-Lafuente. Maximizing extractable value from automated market makers. In Ittay Eyal and Juan A. Garay, editors, *Financial Cryptography and Data Security - 26th International Conference, FC 2022, Grenada, May 2-6, 2022, Revised Selected Papers*, volume 13411 of *Lecture Notes in Computer Science*, pages 3–19. Springer, 2022.
- 9 Massimo Bartoletti and Roberto Zunino. A theoretical basis for blockchain extractable value. *CoRR*, abs/2302.02154, 2023. [arXiv:2302.02154](https://arxiv.org/abs/2302.02154).
- 10 Soumya Basu, David Easley, Maureen O’Hara, and Emin Gün Sirer. Towards a functional fee market for cryptocurrencies. *arXiv preprint arXiv:1901.06830*, 2019.
- 11 Iddo Bentov, Yan Ji, Fan Zhang, Lorenz Breidenbach, Philip Daian, and Ari Juels. Tesseract: Real-time cryptocurrency exchange using trusted hardware. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 1521–1538. ACM, 2019.
- 12 Vitalik Buterin, Eric Conner, Rick Dudley, Matthew Slipper, Ian Norden, and Abdelhamid Bakhta. EIP-1559: Fee market change for ETH 1.0 chain. URL: <https://github.com/ethereum/EIPs/blob/master/EIPS/eip-1559.md>, 2019.
- 13 Christian Cachin, Jovana Micic, Nathalie Steinhauer, and Luca Zanolini. Quick order fairness. In Ittay Eyal and Juan A. Garay, editors, *Financial Cryptography and Data Security - 26th International Conference, FC 2022, Grenada, May 2-6, 2022, Revised Selected Papers*, volume 13411 of *Lecture Notes in Computer Science*, pages 316–333. Springer, 2022.
- 14 Jon Charbonneau. Encrypted mempools. URL: <https://joncharbonneau.substack.com/p/encrypted-mempools>, March 2023.
- 15 Tarun Chitra, Matheus V. X. Ferreira, and Kshitij Kulkarni. Credible, optimal auctions via blockchains. *arXiv preprint arXiv:2301.12532*, 2023.
- 16 Hao Chung and Elaine Shi. Foundations of transaction fee mechanism design. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3856–3899. SIAM, 2023.
- 17 Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In *2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020*, pages 910–927. IEEE, 2020.
- 18 Domothy. Burning MEV through block proposer auctions. URL: <https://ethresear.ch/t/burning-mev-through-block-proposer-auctions/14029>, October 2022.
- 19 Meryem Essaidi, Matheus V. X. Ferreira, and S Matthew Weinberg. Credible, strategyproof, optimal, and bounded expected-round single-item auctions for all distributions. In *Proceedings of the 13th Innovations in Theoretical Computer Science Conference (ITCS)*, 2022.

- 20 Matheus V. X. Ferreira, Daniel J. Moroz, David C. Parkes, and Mitchell Stern. Dynamic posted-price mechanisms for the blockchain transaction-fee market. In *Proceedings of the 3rd ACM Conference on Advances in Financial Technologies*, pages 86–99, 2021.
- 21 Matheus V. X. Ferreira and David C. Parkes. Credible decentralized exchange design via verifiable sequencing rules. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 723–736, 2023.
- 22 Matheus V. X. Ferreira and S Matthew Weinberg. Credible, truthful, and two-round (optimal) auctions via cryptographic commitments. In *Proceedings of the 21st ACM Conference on Economics and Computation*, pages 683–712, 2020.
- 23 Flashbots. The Future of MEV is SUAVE. URL: <https://writings.flashbots.net/the-future-of-mev-is-suave/>, November 2022.
- 24 Yotam Gafni and Aviv Yaish. Greedy transaction fee mechanisms for (non-) myopic miners. *arXiv preprint arXiv:2210.07793*, 2022.
- 25 Stephane Gosselin and Ankit Chiplunkar. The orderflow auction design space. <https://frontier.tech/the-orderflow-auction-design-space>, 2023.
- 26 Lioba Heimbach and Roger Wattenhofer. Eliminating sandwich attacks with the help of game theory. In Yuji Suga, Kouichi Sakurai, Xuhua Ding, and Kazue Sako, editors, *ASIA CCS '22: ACM Asia Conference on Computer and Communications Security, Nagasaki, Japan, 30 May 2022 - 3 June 2022*, pages 153–167. ACM, 2022.
- 27 Mahimna Kelkar, Soubhik Deb, Sishan Long, Ari Juels, and Sreeram Kannan. Themis: Fast, strong order-fairness in byzantine consensus. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023*, pages 475–489. ACM, 2023.
- 28 Mahimna Kelkar, Fan Zhang, Steven Goldfeder, and Ari Juels. Order-fairness for byzantine consensus. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part III*, volume 12172 of *Lecture Notes in Computer Science*, pages 451–480. Springer, 2020.
- 29 Rami Khalil, Arthur Gervais, and Guillaume Felley. TEX - A securely scalable trustless exchange. *IACR Cryptol. ePrint Arch.*, page 265, 2019. URL: <https://eprint.iacr.org/2019/265>.
- 30 Kshitij Kulkarni, Theo Diamandis, and Tarun Chitra. Towards a theory of maximal extractable value I: constant function market makers. *CoRR*, abs/2207.11835, 2022. [arXiv:2207.11835](https://arxiv.org/abs/2207.11835).
- 31 Klaus Kursawe. Wendy, the good little fairness widget: Achieving order fairness for blockchains. In *AFT '20: 2nd ACM Conference on Advances in Financial Technologies, New York, NY, USA, October 21-23, 2020*, pages 25–36. ACM, 2020.
- 32 Ron Lavi, Or Sattath, and Aviv Zohar. Redesigning bitcoin’s fee market. *ACM Transactions on Economics and Computation*, 10(1):1–31, 2022.
- 33 S. Leonardos, B. Monnot, D. Reijsbergen, S. Skoulakis, and G. Piliouras. Dynamical analysis of the EIP-1559 Ethereum fee market. In *Proceedings of the 3rd ACM Advances in Financial Technologies*, 2021.
- 34 Dahlia Malkhi and Pawel Szalachowski. Maximal extractable value (MEV) protection on a DAG. In Yackolley Amoussou-Guenou, Aggelos Kiayias, and Marianne Verdier, editors, *4th International Conference on Blockchain Economics, Security and Protocols, Tokenomics 2022, December 12-13, 2022, Paris, France*, volume 110 of *OASICs*, pages 6:1–6:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.
- 35 Paul Milgrom and Ilya Segal. Deferred-acceptance auctions and radio spectrum reallocation. In *Proceedings of the fifteenth ACM conference on Economics and computation*, pages 185–186, 2014.
- 36 Robert Miller. MEV-Share: programmably private orderflow to share MEV with users. URL: <https://collective.flashbots.net/t/>

- mev-share-programmably-private-orderflow-to-share-mev-with-users/1264, February 2023.
- 37 Roger B Myerson and Mark A Satterthwaite. Efficient mechanisms for bilateral trading. *Journal of economic theory*, 29(2):265–281, 1983.
  - 38 Noam Nisan. Serial monopoly on blockchains. URL: <https://www.cs.huji.ac.il/~noam/publications/ser-mon.pdf>, April 2023.
  - 39 Alexandre Obadia, Alejo Salles, Lakshman Sankar, Tarun Chitra, Vaibhav Chellani, and Philip Daian. Unity is strength: A formalization of cross-domain maximal extractable value. *CoRR*, abs/2112.01472, 2021. [arXiv:2112.01472](https://arxiv.org/abs/2112.01472).
  - 40 Mallesh Pai and Max Resnick. Structural advantages for integrated builders in mev-boost. *arXiv preprint arXiv:2311.09083*, 2023.
  - 41 Kaihua Qin, Liyi Zhou, and Arthur Gervais. Quantifying blockchain extractable value: How dark is the forest? In *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022*, pages 198–214. IEEE, 2022.
  - 42 Kaihua Qin, Liyi Zhou, Benjamin Livshits, and Arthur Gervais. Attacking the DeFi ecosystem with flash loans for fun and profit. In Nikita Borisov and Claudia Díaz, editors, *Financial Cryptography and Data Security - 25th International Conference, FC 2021, Virtual Event, March 1-5, 2021, Revised Selected Papers, Part I*, volume 12674 of *Lecture Notes in Computer Science*, pages 3–32. Springer, 2021.
  - 43 Max Resnick. Contingent fees in order flow auctions. *arXiv preprint arXiv:2304.04981*, 2023.
  - 44 Tim Roughgarden. *Twenty Lectures on Algorithmic Game Theory*. Cambridge University Press, 2016.
  - 45 Tim Roughgarden. Transaction fee mechanism design. *ACM SIGecom Exchanges*, 19(1):52–55, 2021. Full version at <https://arxiv.org/abs/2106.01340>.
  - 46 Alejo Salles. On the formalization of MEV. URL: <https://collective.flashbots.net/t/on-the-formalization-of-mev/879>, december 2021.
  - 47 Elaine Shi, Hao Chung, and Ke Wu. What can cryptography do for decentralized mechanism design. *arXiv preprint arXiv:2209.14462*, 2022.
  - 48 Christof Ferreira Torres, Ramiro Camino, and Radu State. Frontrunner jones and the raiders of the dark forest: An empirical study of frontrunning on the Ethereum blockchain. In Michael Bailey and Rachel Greenstadt, editors, *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, pages 1343–1359. USENIX Association, 2021.
  - 49 Ke Wu, Elaine Shi, and Hao Chung. Maximizing miner revenue in transaction fee mechanism design. In Venkatesan Guruswami, editor, *15th Innovations in Theoretical Computer Science Conference, ITCS 2024, January 30 to February 2, 2024, Berkeley, CA, USA*, volume 287 of *LIPICs*, pages 98:1–98:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024.
  - 50 Sen Yang, Fan Zhang, Ken Huang, Xi Chen, Youwei Yang, and Feng Zhu. Sok: MEV countermeasures: Theory and practice. *CoRR*, abs/2212.05111, 2022.
  - 51 Andrew C.-C. Yao. An incentive analysis of some Bitcoin fee designs. In *Proceedings of the 47th International Colloquium on Automata, Languages, and Programming (ICALP)*, 2020.
  - 52 Haoqian Zhang, Louis-Henri Merino, Vero Estrada-Galiñanes, and Bryan Ford. Flash freezing flash boys: Countering blockchain front-running. In *42nd IEEE International Conference on Distributed Computing Systems, ICDCS Workshops, Bologna, Italy, July 10, 2022*, pages 90–95. IEEE, 2022.
  - 53 Zishuo Zhao, Xi Chen, and Yuan Zhou. Bayesian-Nash-incentive-compatible mechanism for blockchain transaction fee allocation. *arXiv preprint arXiv:2209.13099*, 2022.



# Profitable Manipulations of Cryptographic Self-Selection Are Statistically Detectable

Linda Cai ✉ 


Princeton University, NJ, USA

Jingyi Liu ✉ 

Princeton University, NJ, USA

S. Matthew Weinberg ✉ 

Princeton University, NJ, USA

Chenghan Zhou ✉ 

Stanford University, Palo Alto, CA, USA

---

## Abstract

---

Cryptographic Self-Selection is a common primitive underlying leader-selection for Proof-of-Stake blockchain protocols. The concept was first popularized in Algorand [7], who also observed that the protocol might be manipulable. [11] provide a concrete manipulation that is strictly profitable for a staker of any size (and also prove upper bounds on the gains from manipulation).

Separately, [3, 23] initiate the study of *undetectable* profitable manipulations of consensus protocols with a focus on the seminal Selfish Mining strategy [9] for Bitcoin’s Proof-of-Work longest-chain protocol. They design a Selfish Mining variant that, for sufficiently large miners, is strictly profitable yet also indistinguishable to an onlooker from routine latency (that is, a sufficiently large profit-maximizing miner could use their strategy to strictly profit over being honest in a way that still appears to the rest of the network as though everyone is honest but experiencing mildly higher latency. This avoids any risk of negatively impacting the value of the underlying cryptocurrency due to attack detection).

We investigate the detectability of profitable manipulations of the canonical cryptographic self-selection leader selection protocol introduced in [7] and studied in [11], and establish that for any player with  $\alpha < \frac{3-\sqrt{5}}{2} \approx 0.38$  fraction of the total stake, *every strictly profitable manipulation is statistically detectable*. Specifically, we consider an onlooker who sees only the random seed of each round (and does not need to see any other broadcasts by any other players). We show that the distribution of the sequence of random seeds when any player is profitably manipulating the protocol is inconsistent with any distribution that could arise by honest stakers being offline or timing out (for a natural stylized model of honest timeouts).

**2012 ACM Subject Classification** Theory of computation → Algorithmic game theory and mechanism design; Applied computing → Digital cash

**Keywords and phrases** Blockchain, Cryptocurrency, Proof-of-Stake, Strategic Mining, Statistical Detection

**Digital Object Identifier** 10.4230/LIPIcs.AFT.2024.30

**Related Version** *Full Version*: <https://arxiv.org/abs/2407.16949> [5]

**Funding** *Linda Cai*: Supported by a Chainlink Fellowship, and a Ripple UBRI grant.

*Jingyi Liu*: Supported by a Ripple UBRI grant.

*S. Matthew Weinberg*: Supported by a Ripple UBRI grant, and NSF CAREER CCF-1942497.

*Chenghan Zhou*: Supported by a Ripple UBRI grant.



© Linda Cai, Jingyi Liu, S. Matthew Weinberg, and Chenghan Zhou;

licensed under Creative Commons License CC-BY 4.0

6th Conference on Advances in Financial Technologies (AFT 2024).

Editors: Rainer Böhme and Lucianna Kiffer; Article No. 30; pp. 30:1–30:23

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1 Introduction

Since Nakamoto introduced Bitcoin in 2008, blockchain technology has made a significant impact on digital transactions by establishing a decentralized system in which transactions are validated through consensus among peers, rather than by a central authority. This innovation, while popularizing decentralized currencies, has also brought to light substantial challenges, particularly the extensive computational and energy demands of its proof-of-work (PoW) consensus mechanism. Notably, the energy consumption associated with Bitcoin mining exceeds that of many countries, raising significant environmental concerns. Furthermore, the necessity for large-scale mining hardware introduces considerable centralization risks to cryptocurrencies [2], many of which are inherently designed to be decentralized.

In response to these challenges, the blockchain community has been exploring Proof of Stake (PoS), which has been implemented in many prominent crypto-currencies (e.g. Ethereum, Algorand, Cardano). In each round, PoS selects block leaders (who get to propose a block to be included) based on the stake, reducing energy usage and aiming to prevent Sybil attacks by randomly assigning leadership chances proportionally to coin holdings. However, the leader selection process in PoS presents additional challenges. For example, the pseudorandomness resulting from PoW is in some sense “external” to the blockchain (the next miner is selected proportionally to their computational power, independently, and nothing in the blockchain itself can influence this). Replicating this property in PoS blockchains has proved challenging without trusting an external randomness beacon (which is often a non-starter in blockchain applications, whose entire purpose is to remove the need for such trust).<sup>1</sup> On the other hand, pseudorandom numbers generated using the blockchain itself can often be predicted by the miners, opening up the possibility of profitable deviations [4].

One promising idea in addressing the leader selection challenge is cryptographic self-selection, initially proposed by Algorand [7]. Cryptographic self-selection is a protocol to select a block-proposer for round  $r + 1$  as a function of communication during round  $r$ . We overview Algorand’s canonical proposal shortly, and briefly note here that it is known to admit profitable deviations for arbitrarily small participants [11].<sup>2</sup> We subsequently discuss cryptographic self-selection in further detail, but at this point merely wish to note that: (a) nonmanipulable randomness sources are a major open problem within the blockchain community, due to applications for PoS, (b) these problems are important to both researchers [4, 7, 12, 11] and practitioners [1]<sup>3</sup>, and (c) the particular approach initially proposed in [7] is a canonical testbed due to its elegance and simplicity (which we overview shortly).

Separately, recent work of [3, 23] propose a novel concern for profitable manipulations: detectability. Specifically, while it may be challenging to *trace* a strategic manipulation to a particular actor in a permissionless system,<sup>4</sup> profitable manipulations would likely be *detect-*

---

<sup>1</sup> To slightly elaborate on this point: trusting a centralized external randomness beacon (such as NIST) is certainly a non-starter, because NIST then has control over the block producers. One could instead have an external distributed process to generate random numbers independent of this blockchain. But if this blockchain has monetary value, then securely implementing that distributed process is its own challenge. The story is getting more subtle with Verifiable Delay Functions that might act as a cryptographic external randomness beacon, although their security assumptions are hardware-based and not as battle-tested as standard cryptography, so there will always be a desire for solutions based on standard cryptography.

<sup>2</sup> This is in contrast to block-withholding manipulations in PoW longest-chain protocols [21, 16, 9], although alternate strategic manipulations of some PoW protocols are profitable for arbitrarily small miners [13, 15, 23].

<sup>3</sup> For example, this blog post by the Ethereum foundation on manipulating its RanDAO: link.

<sup>4</sup> This is not to say that tracing strategic manipulations is impossible – indeed, law enforcement regularly

able. This observation serves as a basis to mitigate concerns with profitable manipulations in practice – perhaps the manipulator will earn moderate additional cryptocurrency via manipulation, but its detection may cause the value of these tokens to tank when measured in USD. Their work highlights that detectability of strategic manipulations plays a significant role in their usability in practice – undetectable deviations avoid the risk of devaluing the underlying cryptocurrency, while detectable ones can be disincentivized through outside-the-model means.

Our paper lies at the intersection of these two agendas: we investigate the detectability of profitable manipulations in cryptographic self-selection. Surprisingly, our main result finds that for any participant with less than  $\frac{3-\sqrt{5}}{2} \approx 0.38$  fraction of the total stake, *all profitable manipulations of Algorand’s canonical cryptographic self-selection protocol are statistically detectable*.

We now provide additional context and details for our result.

**Leader Selection in PoS Blockchains.** PoS consensus protocols typically take one of two forms: they may be a *longest-chain protocol*, or a *Byzantine Fault Tolerant (BFT)-based protocol*. Both formats are well-represented in practice, and Ethereum is in some sense a hybrid of the two. In a longest-chain protocol, strategic manipulations are more straightforward – they typically come by inducing forks, and causing the attacker to have their own blocks represent a greater fraction of blocks in the longest chain [9, 21, 16, 4, 12]. Manipulations of BFT-based protocols are more subtle. BFT-based protocols proceed one block at a time, reaching a strong consensus on block  $r$  and finalizing it forever before getting to work on block  $r + 1$ . As such, these protocols are nonmanipulable at the per-block level (unless the attacker has sufficient stake to cause significantly more damage by violating consensus entirely). Instead, these protocols typically have a randomly-selected “leader” dictate the contents of the block and the per-round BFT protocol aims to reach consensus on the leader’s block. But, these protocols still need an effective method to select a leader for each round independently and proportional to their stake.

Fortunately for mechanism designers, leader selection protocols are often modular components of the broader blockchain protocol, and can be studied in isolation from the (significantly more complex) BFT protocols that handle per-round consensus.

**Algorand’s Canonical Leader Cryptographic Self-Selection.** [7] propose an elegant leader selection protocol, which we describe for simplicity in the case where each account holds the same number of coins (we rigorously overview their protocol in the general case in Section 2, but omit the generalization now in the interest of clarity). First, pick a uniformly random seed,  $Q_1$ , for round one. Then in round  $r$ , ask each account holder  $i$  to first digitally sign  $Q_r$  and then hash<sup>5</sup> their digital signature to get a credential  $\text{CRED}_i^r$ . Whoever broadcasts the smallest credential is the leader for round  $r$ .

Their protocol has several desirable properties. First, assuming that every player honestly digitally signs and hashes in each round (and that the hash function behaves like a random oracle), the leader in each round is indeed a uniformly random coin, independent of all previous rounds. Second, it is not predictable too far into the future: because player  $i$  cannot

---

traces attacks in permissionless systems: link.

<sup>5</sup> The formal concept is a Verifiable Random Function, which we define in Section 2.1. Intuitively, the hash is a uniformly random number drawn specifically for player  $i$  in a manner that no other player can precompute (because they can’t digitally sign on behalf of player  $i$ ).

## 30:4 Detecting Profitable Manipulations in Cryptographic Self-Selection

digitally sign on behalf of player  $j$ , player  $i$  has absolutely no idea what seed might result next round (if another player is the leader). Finally, the manner in which it can possibly be manipulated is extremely structured: the only strategies available to a player are to broadcast or not broadcast their credentials.

Still, their protocol is not perfect – [7] already acknowledge that it might be manipulable, and [11] establish a strictly profitable strategy for arbitrarily small players. [11]’s strategy is fairly simple, and we overview it in Section 4.

**Detecting Strategic Behavior in Cryptographic Self-Selection.** How would one detect that a participant is strategically manipulating a protocol? In PoW longest-chain protocols, [3] propose to look at the pattern of forks – strategic behavior often results in long runs of consecutive forks whereas routine latency instead would result in independently distributed forks. This particular detection method is not applicable to a BFT-based protocol, as BFT-based protocols have no forks once a block is finalized.

In the spirit of [3], we aim to detect strategies using the minimal amount of information possible, and in particular we only use information that is available to anyone following the blockchain. Specifically, anyone following the blockchain *must know who is the leader of round  $r$*  and *must know their credential  $CRED_i^r$*  that proves they are the leader.

If every participant in the network were honest, and there were  $n$  coins in the network, we would expect in a given round that the winning credential is distributed according to the minimum of  $n$  independent draws of the Hash function. So across a large number of rounds, an observer could check the sequence of winning credentials and see if they empirically match i.i.d. draws of the minimum of  $n$  independent draws of the Hash function.

This is perhaps too strong of an assumption on honest parties, however. In particular, it assumes either that every single coin is online and participating in the protocol or that the observer otherwise knows that exactly (say)  $k$  coins are online. An observer instead might know that there exists some number  $k$  of online coins participating in the protocol, but not know  $k$ . Then, they would expect to see sequences of credentials that empirically match i.i.d. draws from the minimum of  $k$  independent draws of the Hash function, for some  $k$ .

So consider an attacker who controls multiple accounts. They can selectively refrain from broadcasting in round  $r$  (and might benefit from doing so, if another account will win round  $r$  anyway and their chosen credential gives them a better shot of winning round  $r + 1$ ), but doing so will skew the distribution of round  $r$ ’s credential larger and the distribution of round  $r + 1$ ’s credential smaller. This is profitable, but when done naively detectable (we analyze [11]’s particularly simple strategy, which follows from this intuition, in Section 3). The challenge for the attacker is whether it is possible to profit (by biasing their winning credential to be lower in some rounds), without being detectable (by biasing the winning credential in other rounds to be higher). Our main result shows that this is impossible: for any  $\alpha < \frac{3-\sqrt{5}}{2} \approx 0.38$ ,<sup>6</sup> and any participant with an  $\alpha$  fraction of the total stake, any strategy that leads a  $> \alpha$  fraction of the rounds produces a distribution over sequences of winning credentials that is not consistent with any number of online honest coins.<sup>7</sup>

Finally, one might even consider having a fixed  $k$  of online coins to be too stringent of a null hypothesis – perhaps the number of active coins fluctuates from round to round. We also establish that our main result degrades smoothly in the deviation an observer is comfortable

---

<sup>6</sup> Note, for example, that an adversary with  $\alpha > \frac{3-\sqrt{5}}{2} > 1/3$  of the stake could alternatively directly violate the underlying consensus protocol, which would do significantly more damage than a strategic manipulation.

<sup>7</sup> Like most prior work (e.g. [16, 4, 12, 3]), we consider an attacker who does not have excessively strong network connectivity – see Section 3 for the formal setup.

attributing to honest-but-occasionally-offline behavior. If the observer believes the online coins to fluctuate within  $1 \pm \delta$  of an unknown baseline, and the true online coins indeed fluctuate within  $1 \pm \delta$  of some ground truth baseline, undetectable manipulations lead at most an additional  $2\delta$  fraction of rounds.

**Roadmap and Discussion.** We study the detectability of strategic manipulations in cryptographic self-selection, Algorand’s canonical leader selection protocol [7]. We establish that any profitable deviation is detectable, and also quantitatively extend our results to even further relaxed null hypotheses of what might result from honest-but-offline behavior.

Detectability of profitable manipulations is a desirable property of consensus protocols, as it provides an outside-the-model avenue to deter deviant behavior. While [3] derive profitable, undetectable deviations from longest-chain PoW consensus protocols, we instead show that cryptographic self-selection admits no profitable manipulations. Our work now establishes that some canonical protocols admit undetectable profitable deviations while others do not, and further motivates detectability of profitable manipulations as a standard question to be asked of novel consensus protocols.

In Section 1.1, we overview related work in further detail. In Section 2 and Section 3 we overview our model and our statistical detection methods in significantly more detail. Section 4 overviews the profitable strategy of [11] through the lens of detectability in order to familiarize the reader with the techniques. Section 5 formally states and proves our main result and its robust extension.

## 1.1 Related Work

**Detection of Strategic Attacks in Proof-of-Work Protocols.** Several methods of detecting selfish mining in proof-of-work protocol have been proposed. [8] presents a heuristic to detect selfish mining based on changes in the height of forks in a blockchain network and their simulation result implies a connection between the presence of selfish mining attack and higher rate of forks, with a mean height of higher than 2.

[18] proposes a statistical test for each miner based on the null hypothesis that under honest mining, the probability of observing two successive blocks mined by the same miner is given by type II binomial distribution of order 2, and the presence of selfish mining will cause deviation from such distribution, causing a higher probability of observing successive blocks mined by the same miner. The authors conduct empirical tests on five cryptocurrencies based on Proof-of-Work – Bitcoin, Litecoin, Ethereum, Monacoin and Bitcoin Cash and claim to be the first research work that reveals the presence of selfish mining in real cryptocurrency systems, although they acknowledge that other reasons can also lead to abnormal successive block discovery rates. We further note that their detection method relies on knowing which addresses or wallets are controlled by the same user, while our detection scheme does not rely on such knowledge.

Other works such as [22] use neural networks that achieve good accuracy of detecting selfish mining on simulated datasets.

In contrast to these detection methods, [3] proves the existence of a statistically undetectable and strictly profitable selfish mining strategy for miners with 38.2% of the total hash rate. Under this strategy, the attacker hides their block with carefully constructed probabilities such that the eventual structure of the blockchain under this selfish mining attack has the same distribution as the structure of the blockchain constructed by only honest miners with a different latency parameter. Thus statistical tests that only look at the pattern of the blockchain itself such as fork heights cannot detect their attack.

**Strategic Manipulation of Consensus Protocols.** Following seminal work of [9], there is now a long body of work studying strategic manipulations in consensus protocols [9, 21, 16, 6, 15, 13, 12, 11, 24, 23, 3]. These works are all thematically related to ours in that we also study strategic manipulation of consensus protocols. Of these, only [11] bears any technical similarities, as the others all study longest-chain variants.

In terms of motivating cryptographic self-selection, [4] establish that longest-chain variants with fully-internal pseudorandomness are all vulnerable to a selfish-mining-style attack based on predicting future randomness.

Research works on the detection of strategic attacks mostly focus on the longest-chain Proof-of-Stake protocols such as [20]. To the best of our knowledge, our work is the first to propose a detection method for manipulating leader selection protocols in BFT-based blockchains.

**Relevant Proof-of-Stake Protocols in Practice.** Several large blockchains employ Proof-of-Stake over Proof-of-Work, and there is not yet convergence on a dominant consensus paradigm. For example, Cardano [17] uses a longest-chain variant considered in [4], Algorand uses cryptographic self-selection [14, 7] considered in [11] (although Algorand seems to have since updated their leader selection to induce a round robin aspect – every  $k$  rounds, the winner’s credential sets the seeds for the subsequent  $k$  rounds. See [14].), and Ethereum uses a hybrid of the two (although manipulations of Ethereum are much closer to manipulations of cryptographic self-selection than of longest-chain protocols – see here). In terms of relevance for practice, our results (a) highlight a desirable property of [7]’s original cryptographic self-selection that is desirable in practice, and (b) serve as a canonical example to highlight manners in which a protocol might avoid undetectable profitable deviations.

## 2 Model and Preliminaries

### 2.1 Proof-of-Stake Consensus Protocols with Finality

Proof-of-stake protocols with finality look more like classical Consensus algorithms from Distributed Systems than Bitcoin’s Longest-Chain protocol. That is, these protocols repeatedly run a secure consensus algorithm to agree on a block of authorized transactions, add this block of transactions to the ledger, and proceed. Unlike the Longest-Chain protocol, these blocks are added to the ledger and remain in the ledger forever. In order to maintain security guarantees, the consensus algorithm for each block is often complex.

To mitigate this complexity (both computational, communication, and conceptual), many protocols select a *leader*  $\ell_t$  who plays a special role in the consensus protocol. Intuitively, all participants try to copy the leader’s proposed block. Similarly to a Longest-Chain protocol, the leader  $\ell_t$  dictates the contents of the block. That is, the contents of Block  $t$  are fully dictated by the leader  $\ell_t$ , just like in a Longest-Chain protocol (and the only difference is how consensus is reached so that the rest of the network agrees on what block was indeed dictated). As a result, we model the payoff of players to be the fraction of rounds in which they are the block leader, since creating a block is the only way they can gain profit (e.g., through block rewards and MEV).

In order to mitigate grinding attacks, the leader-selection protocol typically needs to ensure that when participants are behaving honestly, the probability that each participant (or pool of participants) gets to be the leader in each round is proportional to each participant’s stake (hence the name “proof-of-stake”). Moreover, there should be limited room for a participant to gain extra profit by deviating from the protocol.

Cryptographic self-selection (used by Algorand [7, 14]) is an elegant solution for the leader-selection protocol, which does not rely on the existence of frequent and high quality randomness beacon to generate a random seed for each round. Instead, it uses information about the previous rounds to generate a seed for the current round. We now briefly describe the protocol and the parts that are relevant for constructing a statistical detection method of strategic deviation from the protocol. The reader could refer to [11, 7] for a more detailed description of the protocol and encryption schemes.

There are two key components to the cryptographic self-selection protocol: Verifiable Random Functions (VRFs) and balanced scoring functions.

► **Definition 1** (Verifiable Random Function (VRF) [19]). *A Verifiable Random Function is a public-key cryptographic function that generates public key and secret key pairs, denoted  $(\text{sk}, \text{pk})$ , and efficiently evaluates an input  $x$  using a function  $f_{\text{sk}}$  that is dependent on the secret key. The function produces an output  $y$  and a proof of correctness, which can be verified efficiently by anyone who has the public key. The following security properties are guaranteed:*

- *Pseudorandomness: Given the public key  $\text{pk}$  and a sequence of input-output pairs  $(x_1, y_1), \dots, (x_n, y_n)$  with their corresponding proofs, it is computationally infeasible to predict  $y = f_{\text{sk}}(x)$  for any  $x \neq x_1, \dots, x_n$  without the secret key  $\text{sk}$ . In fact, the distribution of  $y$  looks indistinguishable from the uniform distribution on  $[0, 1]$ .*
- *Unique Provability: For any input  $x$ , there is exactly one output  $y$  that can be verified as the correct computation of  $f_{\text{sk}}(x)$ .*

A balanced scoring function takes in the pseudorandom output generated from the VRF associated with an account (i.e. parametrized by the account's secret key) and the amount of stake in that account, and yields a score. The account with the minimum score is selected as the leader. A balanced scoring function always selects a leader proportional to the account's stake assuming the outputs of the VRFs are truly random. In particular, this implies that splitting one's stake between multiple accounts and/or merging stake with another entity does not impact the probability of being selected as the minimum. This forms the basis for selecting a leader-selection protocol that selects leaders independently in each round proportional to their stake.

► **Definition 2** (Balanced Scoring Function [11]). *A scoring function  $S(\cdot, \cdot)$  takes as input a credential  $X_i$  and a quantity of stake  $\alpha_i$  and outputs a score  $S(X_i, \alpha_i)$ . A scoring function is balanced if for all  $n$  and all player stakes  $\alpha_1, \dots, \alpha_n$ ,*

$$\Pr_{X_1, \dots, X_n \leftarrow U([0,1])} \left[ \underset{i \in [n]}{\operatorname{argmin}} S(X_i, \alpha_i) = j \right] = \frac{\alpha_j}{\sum_{i=1}^n \alpha_i}.$$

► **Proposition 3** ([10]). *Let  $S(\cdot, \cdot)$  be any balanced scoring function. Then, for all  $n \in \mathbb{N}$  and  $(\alpha_i)_{1 \leq i \leq n}$ , the random variables  $S(X, \sum_{i=1}^n \alpha_i)$  and  $\min_{1 \leq i \leq n} \{S(X_i, \alpha_i)\}$  are identically distributed for  $X, X_1, \dots, X_n \sim U([0, 1])$ .*

► **Definition 4** (Cryptographic Self-Selection Protocol (CSSP), [11]). *The Cryptographic Self-Selection Protocol (CSSP) operates as follows:*

1. *Each account  $i$ , with stake  $\alpha_i$ , sets up a VRF  $f_{\text{sk}_i}(\cdot)$  with a pair of secret key and public key  $(\text{sk}_i, \text{pk}_i)$ . Participants agree on some Balanced Scoring Function  $S(\cdot, \cdot)$ .*
2.  *$Q_r$  denotes the seed used during round  $r$ .  $Q_1$  is a uniformly random draw from  $[0, 1]$ , and  $Q_r$  will be determined during round  $r - 1$  (see below).*

3. In round  $r$ , each account  $i$  computes their credential  $CRED_i^r = f_{sk_i}(Q_r)$  using their VRF  $f_{sk_i}$ . Each account-holder should broadcast their credential (this is not enforced – an account-holder may choose not to broadcast, if desired).
4. The leader  $\ell_r$  is the account-holder  $i$  who broadcasts the credential with the lowest score  $S(CRED_i^r, \alpha_i)$ .
5. The seed for the next round,  $Q_{r+1}$ , is set as the credential of the leader of round  $r$ , namely  $CRED_{\ell_r}^r$ .

The actions of a player (who may control multiple accounts) in round  $r$  of a CSSP are simply to decide which (if any) of their credentials to broadcast. The payoff to player  $i$  is the fraction of rounds in which they are the leader. Formally, if  $L_p(r)$  is the indicator variable for whether an account controlled by player  $p$  is the leader in round  $r$ , then the payoff to player  $p$  is  $\liminf_{r \rightarrow \infty} \frac{\sum_{r' \leq r} L_p(r')}{r}$ .

CSSP is a formalization of the leader-selection protocol initiated by Algorand [7]. Note that leader-selection is but one aspect of a Proof-of-Stake protocol (the core of the protocol is reaching consensus on the block proposed by the leader). Fortunately, the leader-selection protocols are modular, and can be studied in isolation from the (significantly more complex) consensus algorithms that use them.

## 2.2 Strategic Play in CSSP

Studies of strategic manipulation in consensus protocols first and foremost aim to understand whether one should expect strategic players to choose to be honest. As such, the overwhelming majority of prior work considers a single strategic player against a profile of honest players. [11] establish that this single strategic player is *not* incentivized to be honest, and we ask whether a strategy that realizes these gains is always detectable. In concurrent and independent work, [10] establish tight bounds on the profitability of manipulations. They do not consider detectability, and therefore the work is orthogonal to ours.

In a CSSP, honest behavior corresponds to broadcasting all credentials in every round. A strategic player may selectively choose which credentials to broadcast in each round. It should initially seem counterintuitive that strategic behavior is profitable – hiding a credential *in round  $r$*  certainly cannot help a player *win round  $r$* . However, hiding a credential in round  $r$  might help a player *win round  $r' > r$*  by influencing the seed  $Q_{r'}$ .

**Strategy Space in CSSP ([11]).** Consider a CSSP parameterized by  $\alpha$ , the fraction of stake controlled by the strategic player, and  $\beta \in [0, 1]$ , the network connectivity strength of the strategic player. The strategic player is called  $\beta$ -strong if it learns  $\beta$  fraction of the credentials broadcast by the honest players before they must broadcast themselves. Specifically,  $\beta = 1$  represents a player that learns all credentials of the honest players before they broadcast (because they are extremely well-connected in the network) and  $\beta = 0$  represents a player that learns none of the credentials of the honest players.

[11] make refinement of the strategy space of the CSSP game by showing that any strategy of the strategic player is equivalent to a strategy that only broadcasts at most one credential per round, splits their stake into as many accounts as possible, and considers only two honest players  $B$  and  $C$ , the former with  $\beta(1 - \alpha)$  fraction of the stake, and the latter with  $(1 - \beta)(1 - \alpha)$  fraction of the stake. Their proof shows that for any strategy  $s$  in CSSP, you can find another strategy  $s'$  in the refined strategy space with the same payoff. Since our focus is on both the profitability and detectability of a strategy, we need to show that such



refinement also preserves the detectability of a strategy. Our detection methods (will be introduced in Section 5) only assume an access to the broadcast credential with the minimum score (i.e. the leader's credential) in each round. Thus two strategies that induce the same minimum broadcast credential in each round are either both detectable or both undetectable. Since for any undetectable strategy  $s$  in CSSP, there is also an undetectable strategy  $s'$  in the refined strategy space, by having  $s'$  broadcast the same credential in each round as the minimum credential that  $s$  broadcasts (and broadcasts none if  $s$  broadcasts none), it is without loss of generality to consider only refined strategies from now on. The refined strategy space is described below:

► **Definition 5** (Refined CSSP, [11]). *The strategic player first splits their stake in as many accounts as possible. This set of accounts, denoted as  $A$ , is then fixed for all rounds. In each round  $r$  of CSSP, the strategic player:*

1. *Is aware of the seed  $Q_r$  and the honesty of player  $B$  and  $C$ .*
2. *Has access to the credential  $CRED_B^r$  of honest player  $B$ , but not to the credential  $CRED_C^r$  of honest player  $C$ . The player only knows the fact that  $CRED_C^r$  is distributed uniformly on  $[0, 1]$ .*
3. *Can compute credentials  $CRED_i^r$  and scores  $S(CRED_i^r, \alpha_i)$  for accounts  $i \in A$ , and can compute the score  $S(CRED_B^r, \beta(1 - \alpha))$ .*
4. *For each  $\ell \in A \cup \{B\}$ , can imagine that perhaps  $Q_{r+1} = CRED_\ell^r$ , and then pre-computes hypothetical credentials  $CRED_i^{r+1}$  for each  $i \in A$  in case we were to have  $\ell_r = \ell$ .*
5. *Can extend this pre-computation to any round  $k$  and sequence of accounts  $i_0, \dots, i_k$  (with  $i_0 \in A \cup \{B\}$  and  $i_\ell \in A$  for  $0 < \ell \leq k$ ), and compute  $CRED_{i_k}^{r+k}$  based on the hypothetical possibility that  $\ell_{r+\ell} = i_\ell$  for each  $\ell \in \{0, \dots, k-1\}$ .*
6. *Selects an account  $i^* \in A$  and broadcasts its credential  $CRED_{i^*}^r$ , or chooses not to broadcast any credential.*

The Refined CSSP is the precise mathematical game we study, for a particular balanced scoring function  $S$ . In addition, Proposition 6 implies that the game induced by CSSP is the same for any balanced scoring function used in the protocol. They further imply that if we have a statistical detection method for strategic behavior under one CSSP protocol using a particular balanced scoring function  $S$ , we can apply the same detection method to a variant of the protocol using another balanced scoring function  $S'$ . Therefore, undetectable profitable strategies exist for any leader-selection protocol based on Algorand's cryptographic self-selection if and only if they exist in the Refined CSSP for any particular  $S$  of our choosing.

► **Proposition 6** ([11, 10]). *The game induced by CSSP with a balanced scoring function is independent of the particular balanced scoring function used. Formally, for two distinct balanced scoring functions  $S, S'$ , the games induced by CSSP are identical. Specifically, for all players  $i$ , there is a bijective mapping  $f$  from strategies of player  $i$  in the CSSP with  $S$  to strategies of player  $i$  in the CSSP with  $S'$ , where all players broadcast the same set of credentials in each round. For all  $i$ , the payoff to player  $i$  in the CSSP with  $S$  under strategy profile  $s$  is exactly the same as the payoff to  $i$  in the CSSP with  $S'$  under strategy profile  $\langle f_i(s_i) \rangle_i$ .*

To simplify our analysis, we choose  $S(X, \alpha) := -\ln(X)/\alpha$ , which induces an exponential distribution with rate  $\alpha$ , i.e. when  $X$  is drawn from  $U([0, 1])$ ,  $S(X, \alpha)$  is drawn from  $\text{Exp}(\alpha)$ . The exponential distribution has the nice property that for a set of random variables  $X_1, \dots, X_n$  drawn from  $\text{Exp}(\alpha_1), \dots, \text{Exp}(\alpha_n)$  respectively, the minimum score  $\min_{i \in [n]} X_n$  is distributed according to  $\text{Exp}(\sum_{i=1}^n \alpha_i)$ . This implies that if the total sum of active stakes is 1 and all players are honest, then the minimum score broadcast is distributed according

to  $\text{Exp}(1)$ . Additionally, Lemma 29 and Lemma 30 imply that the scoring function is a balanced scoring function, and therefore by Proposition 6 there is a bijective mapping from strategies of player  $i$  in the CSSP game with balanced scoring function  $S'$  and strategies of player  $i$  in the CSSP game with balanced scoring function  $S = -\ln(X)/\alpha$ , which achieves the same outcome (i.e., the same leader is selected each round) and the same profit for player  $i$ . Thus if we can detect any profitable strategy under scoring function  $S$ , we can use the same method to detect a profitable strategy under scoring function  $S'$  since the same strategy is also profitable under  $S$ . It is therefore without loss of generality to assume  $S(X, \alpha) = -\ln(X)/\alpha$  for all the analysis that follows. Appendix A contains all the relevant properties of an exponential distribution that we will employ to detect strategic deviation.

### 3 Statistical Detection Methods for Strategic Deviations

Our paper concerns detection of strategic manipulations in CSSP, so we must first clarify what information is available to the onlooker who wishes to distinguish between the case when all participants are honest (but perhaps suffer latency issues), or a strategic player is manipulating the protocol.

We consider the minimal amount of information necessary for an onlooker just to follow the state of the blockchain: the credentials of the leader from each round.<sup>8</sup> We will show that this information alone suffices to detect any profitable manipulation in case the strategic player has  $\beta = 0$ .

Before continuing, we briefly note that the  $\beta = 0$  case corresponds to a “poorly connected” attacker who cannot learn the broadcasts of other players before deciding their own. This matches the  $\gamma = 0$  case when analyzing Proof-of-Work protocols, which is considered standard/canonical. We also note that, if desired, a leader selection protocol could take steps to induce  $\beta = 0$  (for example, participants could cryptographically commit to their credential with a large deposit, and then only receive their deposit back upon revealing). Our main result does leverage  $\beta = 0$  (and we will highlight where), and it is an interesting technical question to understand the case of  $\beta = 1$ .<sup>9</sup> But, we hope this brief note reminds the reader that  $\beta = 0$  is considered the canonical setting. We now proceed with a formal description of the information observed.

► **Definition 7 (Observed distribution).** *Let an observer pick a uniformly random round  $r$  from the set of all rounds  $\{1, \dots, R-1\}$ . Let  $Z, Z_{+1}$  be the random variables denoting the score of the winning credential in consecutive rounds  $r$  and  $r+1$  respectively. i.e.,  $Z = S(\text{CRED}_{\ell_r}^r, \alpha_{\ell_r})$  and  $Z_{+1} = S(\text{CRED}_{\ell_{r+1}}^{r+1}, \alpha_{\ell_{r+1}})$ . Then  $D_Z$  and  $D_{Z_{+1}}$  represent the distributions of the winning credentials in round  $r$  and  $r+1$  when  $R \rightarrow \infty$ , and we define  $F_Z, F_{Z_{+1}}$  to be the cumulative density function (c.d.f.) of  $D_Z, D_{Z_{+1}}$  respectively.*

Let us now briefly discuss a null hypothesis for the observed distribution. One null hypothesis might be that in every round  $r$ , every player is online and suffers no latency issues (that is, every account-holder  $i$  learns of the seed  $Q_r$ , computes  $\text{CRED}_i^r$  and broadcasts it within the allotted time-window), and behaves honestly. If this were the case, we would expect the distribution of winning scores to be i.i.d. from the distribution  $S(U([0, 1]), 1) = \text{Exp}(1)$  (by Lemma 27 and Lemma 28), and in particular we would expect  $(Z, Z_{+1})$  to be distributed according to  $\text{Exp}(1) \times \text{Exp}(1)$ .

<sup>8</sup> Note that the detection method of [3] is tailored to Longest-Chain protocols and in particular looks at the distribution of orphans. As there are no orphans in consensus protocols with finality, we need a fundamentally different detection method.

<sup>9</sup> We further explore [11]’s 1-LOOKAHEAD strategy in Section 4, which leverages  $\beta = 1$  and is detectable.

This is perhaps too strong a null hypothesis, though – some participants may go offline for extended periods of time, and there is no reason the rest of the network should a priori be aware of this. Additionally, some participants may be online but suffer latency issues that prevent them from broadcasting their credential in time. We therefore consider a weaker null hypothesis which instead posits that there exists *some* stake  $\gamma$  which is online and honest each round, *except the precise value of  $\gamma$  is unknown*. Under this null hypothesis, we would expect there to *exist some*  $\gamma$  for which the distribution of winning scores is i.i.d from  $S(U([0, 1]), \gamma) = \text{Exp}(\gamma)$ , and therefore we would expect there to exist some  $\gamma$  for which  $(Z, Z_{+1})$  is distributed according to  $\text{Exp}(\gamma) \times \text{Exp}(\gamma)$ . For simplicity of notation, we w.l.o.g. let 1 denote the “true” online stake, which might be less than the total stake. Therefore, we consider the null hypothesis to also be satisfied when  $\gamma > 1$ .<sup>10</sup>

Our main result establishes that no profitable strategy for a  $\beta = 0$  strategic player induces an observed distribution that passes the null hypothesis. We also consider an even more robust null hypothesis in Section 5 where there exists some unknown  $\gamma$  for which the fraction of active stake in each round lies in  $[(1 - \delta) \cdot \gamma, (1 + \delta) \cdot \gamma]$ , but stick to the simpler null hypothesis first for cleanliness of our main result.

We now elaborate below on two types of statistical tests. Note that in each round, we only have access to the realization, rather than the underlying distribution of the minimum score, so we only have an empirical estimate of  $(F_Z, F_{Z_{+1}})$ . Still, the number of rounds of history for a Proof-of-Stake-with-Finality blockchain protocol is extremely large. For example, Ethereum produces new blocks every twelve seconds, or 7200 blocks/day. We also remind the reader that all prior analysis on profitability, and the unique prior work on detectability, consider profitability and detectability in steady-state. This is sensible given the intended lifespan of a blockchain and the rate at which blocks are produced.

**Detection Method 1: Distribution of Minimum Score.** We first focus simply on the distribution of the winning credential across rounds, without looking at correlation of credentials between rounds. Proposition 8 explicitly confirms that under the null hypothesis,  $D_Z$  should be  $\text{Exp}(\gamma)$  for some  $\gamma$ .

► **Proposition 8.** *When the total online stake is constant across rounds and all players honestly broadcast their credentials, there exists a number  $\gamma$  such that  $D_Z$  is distributed identically to  $\text{Exp}(\gamma)$ .*

**Proof.** Let  $\lambda$  be the actual amount of total online stakes. By Proposition 3,  $\min_i \{S(X_i, \alpha_i)\}$  is distributed identically to  $S(X, \sum_i \alpha_i)$  when  $X, X_i$  are i.i.d. from  $U([0, 1])$ . Since when all players are honest,  $\text{CRED}_i^r$  is distributed identically to  $U([0, 1])$ . Therefore, at each round  $r$ , the score of the leader  $S(\text{CRED}_{\ell_r}^r, \alpha_{\ell_r})$  is distributed identically to  $S(X, \lambda)$ , where  $X \sim U([0, 1])$ . By our choice of the scoring function,  $S(\text{CRED}_{\ell_r}^r, \alpha_{\ell_r})$  is distributed identically to  $\text{Exp}(\lambda)$ . Thus, the c.d.f. of  $D_Z$  is

$$F_Z(z) = \lim_{R \rightarrow \infty} \sum_{r=1}^R \frac{1}{R} F_{S(\text{CRED}_{\ell_r}^r, \alpha_{\ell_r})}(z) = \lim_{R \rightarrow \infty} \sum_{r=1}^R \frac{1}{R} (1 - e^{-\lambda z}) = 1 - e^{-\lambda z}$$

which implies that  $D_Z$  is distributed identically to  $\text{Exp}(\lambda)$ . Taking  $\gamma = \lambda$  concludes our proof. ◀

<sup>10</sup>That is, if the strategic player causes it to appear as though a  $\gamma > 1$  fraction of the total stake is online and honest, then clearly something is wrong and an onlooker should detect this. But if the true online stake is only 1/3 of the total stake and the strategic player causes it to appear as though 2 · 1/3 of the total stake is online and honest, this is plausible to an onlooker who doesn't know the true fraction of online stake.

## 30:12 Detecting Profitable Manipulations in Cryptographic Self-Selection

A more robust null hypothesis allows for the possibility that the fraction of online players varies across rounds, but not by much. In this setting, the total amount of online stake lies in some range  $[(1 - \delta)\lambda, (1 + \delta)\lambda]$  for some small  $\delta > 0$  and  $\lambda > 0$ . The exact distribution  $D_Z$  is impossible to compute without knowing the actual online stake  $\lambda_1, \dots, \lambda_R$  in each round. Nevertheless, the observer expects that the c.d.f. of  $D_Z$  is within a certain range parameterized by  $\gamma$  that represents her estimation of  $\lambda$ .

► **Proposition 9.** *When the fraction of online stake lies within a multiplicative  $1 \pm \delta$  factor across all rounds, and all players honestly broadcast their credentials, there exists a number  $\gamma$  such that  $\text{Exp}((1 + \delta)\gamma) \preceq D_Z \preceq \text{Exp}((1 - \delta)\gamma)$ .<sup>11</sup>*

**Proof.** Let  $\lambda$  be such that the online stake in each round is within  $[(1 - \delta) \cdot \lambda, (1 + \delta) \cdot \lambda]$ . Such  $\lambda$  is guaranteed to exist by hypothesis. At each round  $r$ , since the fraction of online stake in round  $r$  is  $\lambda_r$ , we know that  $S(\text{CRED}_{\ell_r}^r, \alpha_{\ell_r})$  is distributed according to  $\text{Exp}(\lambda_r)$ . Thus, the c.d.f. of  $D_Z$  is

$$F_Z(z) = \lim_{R \rightarrow \infty} \sum_{r=1}^R \frac{1}{R} F_{S(\text{CRED}_{\ell_r}^r, \alpha_{\ell_r})}(z) = \lim_{R \rightarrow \infty} \sum_{r=1}^R \frac{1}{R} (1 - e^{-\lambda_r z})$$

Because  $\lambda_r \in [(1 - \delta)\lambda, (1 + \delta)\lambda]$ , for all  $r$ ,

$$\text{Exp}((1 - \delta)\lambda) \preceq \text{Exp}(\lambda_r) \preceq \text{Exp}((1 + \delta)\lambda)$$

Plugging this in  $F_Z(z)$  and substituting  $\lambda$  with  $\gamma$ , we are able to conclude that

$$\text{Exp}((1 + \delta)\gamma) \preceq D_Z \preceq \text{Exp}((1 - \delta)\gamma) \quad \blacktriangleleft$$

We conclude this detection method by reminding the reader that because the observer does not know the actual amount of online stake,  $\gamma$ , a strategic player could make it appear as though the total online stake is some  $\lambda \neq \gamma$  (and we specifically remind the reader that  $\gamma > \lambda$  would and should still satisfy our null hypothesis). Our main contribution in this paper is to show that it is impossible to be profitable and preserve the distribution of broadcast scores to be consistent with any fraction of online stake.

**Detection Method 2: Correlation of Consecutive Minimum Scores.** Our second detection method leverages the fact that the credentials of the leader are independent from round to round when all players follow the protocol. In particular, we examine the correlation between the minimum scores in consecutive rounds. Under honest mining behavior, all credentials are drawn i.i.d. from  $U([0, 1])$ , thus the probability of seeing the score of the leader's credential to be  $z_{+1}$  in round  $r + 1$  should not be changed given the score of the leader's credential  $z_r$  in round  $r$ . Formally,

$$F_{Z, Z_{+1}}(z, z_{+1}) = F_Z(z) \times F_{Z_{+1}}(z_{+1})$$

### 3.1 Necessary Conditions for Undetectable Strategic Attacks

In this section, we analyze the effect of the adversary's strategy on  $D_Z$  and define the concept of a *statistically undetectable strategy*. The honest players would always broadcast their credentials with the minimum scores (equivalently, broadcast all credentials they have), while the adversary commits to a strategy  $\pi$  that does not necessarily broadcast the credential with minimum score.

<sup>11</sup> Here,  $D_1 \preceq D_2$  denotes that  $D_2$  first-order stochastically dominates  $D_1$ .

► **Definition 10.** Let the scoring function  $S(\text{CRED}, \alpha) = -\ln(\text{CRED})/\alpha$ , where  $\alpha$  is the stake of the adversary. Pick a round  $r$  uniformly at random from the set of all rounds  $[R]$ , where the total active stake in round  $r$  is  $\lambda_r$ . Let  $X_r(\lambda_r), X_{r+1}(\lambda_{r+1})$  be the random variables that denote the minimum score of the honest players' broadcast credentials in round  $r$  and  $r + 1$  respectively; let  $Y_r(\pi), Y_{r+1}(\pi)$  be the random variables that denote the score of the adversary's broadcast credential in round  $r$  and  $r + 1$  respectively when they commit to strategy  $\pi$ . Thus, the score of the leader in round  $r$  and  $r + 1$ , could be written as  $Z_r(\pi, \lambda_r) = \min\{X_r(\lambda_r), Y_r(\pi)\}$  and  $Z_{r+1}(\pi, \lambda_{r+1}) = \min\{X_{r+1}(\lambda_{r+1}), Y_{r+1}(\pi)\}$ .

We also define the distribution of these random variables with respect to a uniformly random round  $r$ .

► **Definition 11.** Let  $\mathcal{X}$  be a random variable over a uniformly random round.  $D_{\mathcal{X}}$  is defined to be the distribution of  $\mathcal{X}$ , where the corresponding cumulative density function

$$F_{\mathcal{X}}(x) = \lim_{R \rightarrow \infty} \Pr_{r \leftarrow U\{1, \dots, R\}}[\mathcal{X} \leq x].$$

An observer may choose to examine the distribution of all possible random variables, and even joint distribution of random variables over a random round. For instance, she might examine the score in the previous round, or the joint distribution of the scores in the next 10 rounds. Formally, let  $\mathcal{Z}$  denote the set of scores of broadcast credentials that the observer chooses to examine. A strategy  $\pi$  is robust to any statistical detection if for any set  $\mathcal{Z}$ , the joint distributions of seeing all scores in  $\mathcal{Z}$  over a random round are identical when the adversary uses strategy  $\pi$  with online stake 1 and when the adversary honestly follows the protocol with online stake  $\gamma$ . That is, no matter which set of scores the observer chooses to examine, she could not distinguish the distribution when the adversary honestly follows the protocol and there is a  $\gamma$  fraction of online stake, or when they use strategy  $\pi$  and there is a 1 fraction of online stake (recall that we w.l.o.g. let 1 denote the fraction of online stake for simplicity of notation).

The two detection methods proposed in the previous section give us two necessary conditions for a strategic attack to be undetectable, since we expect the distribution of the minimum scores and the correlation between consecutive minimum scores to follow certain patterns when all players are honest. The first detection method that uses the distribution of minimum scores corresponds to the case when the observer chooses to examine the score at each specific round, i.e.,  $\mathcal{Z} = \{Z_r\}$ .

► **Definition 12.** Let  $\lambda_r$  be the real participating stake in round  $r$ . The observer knows that the sequence of participating stakes falls into a certain class  $\mathcal{C}_R$  representing a sequence of active stakes (e.g. fluctuation must be within  $1 \pm \delta$  fraction). The strategy  $\pi$  is statistically undetectable to the distribution test if for some  $\{\gamma_r\}_{r \in [R]} \in \mathcal{C}_R$  and all  $z_r$ ,

$$\lim_{R \rightarrow \infty} \Pr_{r \leftarrow U(\{1, \dots, R\})}[Z_r(\pi, \lambda_r) \leq z_r] = \lim_{R \rightarrow \infty} \Pr_{r \leftarrow U(\{1, \dots, R\})}[Z_r(\pi_{\text{honest}}, \gamma_r) \leq z_r].$$

The second test on correlation between consecutive minimum scores corresponds to the case when the observer chooses to examine the scores in two consecutive rounds, i.e.,  $\mathcal{Z} = \{Z_r, Z_{r+1}\}$ . In order to distinguish with the first test, we leave the constraint on  $D_{\mathcal{Z}}$  to Definition 12 and only focus on the correlation between  $D_{\mathcal{Z}}$  and  $D_{\mathcal{Z}_{+1}}$  in Definition 13.

► **Definition 13.** A strategy  $\pi$  is statistically undetectable to the correlation test if  $Z(\pi)$  and  $Z_{+1}(\pi)$  are independent. That is, for any  $z_r$  and  $z_{r+1}$ ,

$$\begin{aligned} & \lim_{R \rightarrow \infty} \Pr_{r \leftarrow U(\{1, \dots, R\})} [Z_r(\pi) \leq z_r \wedge Z_{r+1}(\pi) \leq z_{r+1}] \\ &= \lim_{R \rightarrow \infty} \Pr_{r \leftarrow U(\{1, \dots, R\})} [Z_r(\pi) \leq z_r] \cdot \Pr_{r \leftarrow U[1, R]} [Z_{r+1}(\pi) \leq z_{r+1}] \end{aligned}$$

## 4 A Canonical Example

In CSSP, the winning credential of the current round is used as the seed of the next round. This leaves the possibility that an adversary would be strategic in their winning credentials and effectively bias the distribution of seeds. For instance, [11] demonstrate that such protocols are indeed vulnerable to such deviations. In order to acquaint the reader with both the CSSP and statistical detectability, we will show that [11]’s canonical 1-LOOKAHEAD manipulation is statistically detectable using *either* our distribution test or our correlation test.

Here is some brief intuition for 1-LOOKAHEAD: because the winning credential is the seed of the next round, the adversary is able to compute credentials for all wallets assuming that a credential in this round is the winning credential. Thus, if the adversary has multiple credentials with low scores to choose from, they could choose to broadcast only the one which maximizes the expected number of rounds won among the current and one-after round. We repeat the formal definition of 1-LOOKAHEAD below:

► **Definition 14 (1-LOOKAHEAD strategy).** Let the total stake be fixed and normalized to 1 with the adversary owning an  $\alpha$  fraction of the total stake. The goal of the 1-LOOKAHEAD strategy is to maximize the expected number of rounds won among the present and subsequent rounds, and proceeds as follows:

1. Let  $r$  be the current round and  $A$  be the set of all accounts of the adversary,  $B$  be the lone honest account that is broadcast when the adversary decides with total stake  $\beta(1 - \alpha)$ .
2. Let  $W(Q_r) \subseteq A$  denote the accounts  $i$  satisfying  $S(\text{CRED}_i^r, \alpha_i) < S(\text{CRED}_B^r, \beta(1 - \alpha))$ . Observe that  $W(Q_r)$  might be empty, and that when  $\beta = 0$ ,  $W(Q_r) = A$ .
3. If  $W(Q_r)$  is empty, the adversary cannot win this round, so they move on to the next round and go back to step 1.
4. If  $W(Q_r)$  is non-empty, for all potential winning accounts  $i \in W(Q_r)$  and all potential next-round accounts  $j \in A$ , compute credential  $\text{CRED}_{i,j}^{r+1} = f_{\text{sk}_j}(\text{CRED}_i^r)$ , which is the credential of account  $j$  in round  $r + 1$  in the event that account  $i$  happens to win round  $r$ .
5. Let  $j(i) = \arg \min_{j \in A} S(\text{CRED}_{i,j}^{r+1}, \alpha_j)$  – this is the account whose credential is most likely to win in round  $r + 1$  if account  $i$  wins round  $r$ .
6. For each  $i \in W$ , define  $P_i^{r+1}$  to be the probability that the adversary wins with account  $i$  in round  $r$  and wins with account  $j(i)$  in round  $r + 1$ .<sup>12</sup> That is (below, think of  $X^r := S(\text{CRED}_C^r, (1 - \beta)(1 - \alpha))$ ):

$$P_i^{r+1} = \Pr_{X^r \leftarrow \text{Exp}((1-\beta)(1-\alpha))} [S(\text{CRED}_i^r, \alpha_i) < X^r] \cdot \Pr_{X^{r+1} \leftarrow \text{Exp}(1-\alpha)} [S(\text{CRED}_{i,j(i)}^{r+1}, \alpha_{j(i)}) < X^{r+1}].$$

<sup>12</sup>For example, if  $\beta = 1$ , the probability that the adversary wins with account  $i$  in round  $r$  is 1. No matter  $\beta$ , the probability that the adversary wins with account  $j(i)$  in round  $r + 1$  is just the probability that this credential beats a draw from  $\text{Exp}(1 - \alpha)$ .

7. Let  $i^* = \arg \max_{i \in W} (\Pr_{X^r \leftarrow \text{Exp}((1-\beta)(1-\alpha))}[S(\text{CRED}_i^r, \alpha_i) < X^r] + P_i^{r+1})$ .  $i^*$  is the account that maximizes the expected number of consecutive rounds (among  $r, r+1$ ) that the adversary wins.
8. Broadcast  $\text{CRED}_{i^*}^r$  at round  $r$ . If  $\text{CRED}_{i^*}^r$  is the credential with minimum score in round  $r$ , broadcast  $\text{CRED}_{j(i^*)}^{r+1}$  at round  $r+1$ . If  $\text{CRED}_{i^*}^r$  does not win round  $r$  continue.
9. Return to Step 1.

While the honest strategy always broadcasts the credential with minimum score and maximizes the probability of winning the current round  $r$ , 1-LOOKAHEAD instead optimizes the expected number of consecutive rounds won (but only considering the next round – this is why the strategy is termed 1-LOOKAHEAD). For example, when  $\beta = 1$ , and the adversary has multiple accounts that can win this round, they may as well broadcast the credential whose seed gives them the best chance of winning the subsequent round. For  $\beta < 1$ , the math is trickier, but the strategy always strictly outperforms honesty.

Because the purpose of this section is to gain comfort with the concept of detectability, we focus on the simplest version of 1-LOOKAHEAD, which is when  $\beta = 1$  (which corresponds to the most powerful adversary). The arguments in the subsequent subsections proceed roughly as follows:

- Section 4.1 shows how we might start reasoning about the distribution test (Definition 12). In particular, Section 4.1 identifies that we can view the distribution of minimum score  $D_{Z_r(\pi_{1-\text{LOOKAHEAD}})}$  as a mixture of distributions associated with transitions in a two-state Markov Chain, and reasons through what each of these three distributions are. Intuitively, these three distributions are “what is the minimum broadcast score, conditioned on  $r$  being a reset round (i.e. the adversary did not bias  $Q_r$  in  $r-1$ ) and the adversary having at least two winning accounts?”, “what is the minimum broadcast score, conditioned on  $r$  being a round where the adversary biased  $Q_r$  in  $r-1$ ?”, and “what is the minimum broadcast score, conditioned on  $r$  being a reset round and the adversary has at most one winning account?”.
- Section 4.2 then establishes that no mixture of these distributions can result in an exponential distribution, and therefore 1-LOOKAHEAD fails the distribution test and is detectable. Intuitively, this follows simply because exponential distributions have a precise rate of tail decay, and the above distributions have no reason to match this precise tail, nor to cancel the differences out.
- Section 4.3 considers the correlation test, and establishes that 1-LOOKAHEAD also fails the correlation test. Intuitively, this is because during reset rounds we expect to see a larger than normal winning score (because the adversary may hide coins during a reset round), but during biased rounds we expect to see a lower than normal winning score (because the adversary has biased the seed to make their own score lower than normal). So consecutive rounds are in fact negatively correlated.

Note that failing either of the two tests suffice for a strategy to be statistically detectable – we include both to acquaint the reader with various detection methods (a priori, a strategy might pass one test but fail another).

## 4.1 Broadcast Distribution on a Markov Chain

We observe that at each round  $r$ , the distribution of credentials only depend on the distribution of the seed  $Q_r$ . This allows us to characterize the CSSP as a stationary Markov chain. For instance, when all players follow the protocol of CSSP and broadcast their credentials that result in the lowest score, the distribution of  $Q_r$  is uniformly random from  $[0, 1]$  for all  $r$ .

### 30:16 Detecting Profitable Manipulations in Cryptographic Self-Selection

Thus, the Markov chain describing the honest CSSP has only one state that transits to itself with probability 1. In particular, the game effectively resets when the distribution of  $Q_r$  is unbiased. We call such a round to be a “reset round”.

► **Definition 15** (Reset Round [11]). *A round  $r$  is a reset round if for all possible strategies  $\pi$ , the distribution of  $\{\Pr[Y_{r'}(\pi) \leq X_{r'}]\}_{r' \geq r}$  conditioned on  $Q_{r-1}$  and all historical information prior to round  $r-1$ , is identical to the distribution of  $\{\Pr[Y_{r'}(\pi) \leq X_{r'}]\}_{r' \geq r}$  after replacing  $Q_r$  with a uniformly random draw from  $[0, 1]$ .*

The 1-LOOKAHEAD strategy, on the other hand, effectively biases the distribution of the seed in favor of the adversary by comparing the best credential for next round. Therefore, the Markov chain of CSSP when the adversary plays 1-LOOKAHEAD is different from the Markov chain of CSSP when every player follows the protocol.

► **Lemma 16.** *A CSSP process, with the adversary owning  $\alpha$  fraction of the stakes with  $\beta = 1$  and using 1-LOOKAHEAD, is equivalent to the stationary Markov chain with two states  $\Pi = \{C, H\}$ , where the transition probability is*

$$\begin{aligned} \Pr[\Pi_{r+1} = C | \Pi_r = C] &= 1 - \alpha^2 \\ \Pr[\Pi_{r+1} = H | \Pi_r = C] &= \alpha^2 \\ \Pr[\Pi_{r+1} = C | \Pi_r = H] &= 1 \\ \Pr[\Pi_{r+1} = H | \Pi_r = H] &= 0 \end{aligned}$$

Standard calculation shows that the stationary distribution of the above Markov chain would be  $s_C = \frac{1}{1+\alpha^2}$  and  $s_H = \frac{\alpha^2}{1+\alpha^2}$ . The following Lemma shows the overall distribution of the leader’s credential’s score, which is computed by summing the distribution conditioned on each type of transition respectively.

► **Lemma 17.** *The overall distribution of  $D_{Z_r}$  for 1-LOOKAHEAD strategy is*

$$\begin{aligned} D_{Z_r} &= \frac{1}{1+\alpha^2} \left( \sum_{\ell=1}^{\infty} \text{Exp}_{\ell}(1) \left[ \sum_{\omega \geq \ell} \frac{\alpha^{\omega}(1-\alpha)}{\omega} \right] + (1-\alpha)\text{Exp}(1) \right) + \\ &\quad \frac{1}{1+\alpha^2} \sum_{\omega=2}^{\infty} \alpha^{\omega}(1-\alpha)\text{Exp}(1+(\omega-1)\alpha), \end{aligned} \tag{1}$$

where  $\text{Exp}_{\ell}(1) := \text{Exp}_{\ell-1}(1) + \text{Exp}(1)$  with  $\text{Exp}_0(1) := 0$ .

Equation (1) shows that  $D_{Z_r}$  could be viewed as mixture of exponential and Erlang distributions (sum of identical exponential distributions) with different rates. We briefly sketch the argument in the proof of Lemma 17, which is quite technical. Since the score of credential in each account  $i$  with stake  $\alpha_i$  is distributed identical to an exponential  $\text{Exp}(\alpha_i)$  by the properties of exponential distributions (Lemma 28 and Lemma 29), by Lemma 30, the  $\ell^{\text{th}}$  minimum score of credentials that an adversary owns is distributed identical to a Erlang distribution that is sum of  $\ell$  identical exponential distributions, denoted as  $\text{Exp}_{\ell}$ . Since the adversary’s action in each round is confined to choosing which credential to broadcast, the adversary, and hence the overall score distribution must be a mixture of  $\text{Exp}$  and  $\text{Exp}_{\ell}$ s with different rates.

This key property about  $D_{Z_r}$  leads to our main results in this section. In section 4.2, we show that that 1-LOOKAHEAD is statistically detectable under distribution test because the mixture of distributions is not an exponential distribution; In section 4.3, we show that 1-LOOKAHEAD is detectable under correlation test because of stochastic dominance relationships between exponential distributions with different rates.



## 4.2 Broadcast Distribution of 1-Lookahead Cannot be an Exponential Distribution

It is known that the sum of  $\ell$  independent exponential variables with mean 1 each is an Erlang distribution of parameterized by  $\ell, 1$ . That means, the probability density function (p.d.f.) of  $\text{Exp}_\ell(z; 1)$  is  $\frac{z^{\ell-1}e^{-z}}{(\ell-1)!}$ . Plugging in this and the p.d.f. of exponential distributions, we obtain the p.d.f. of  $D_{Z_r}$  to be

$$f_{Z_r} = \frac{1}{1+\alpha^2} \left( \sum_{\ell=1}^{\infty} \frac{z^{\ell-1}e^{-z}}{(\ell-1)!} \left[ \sum_{\omega \geq \ell} \frac{\alpha^\omega(1-\alpha)}{\omega} \right] + (1-\alpha)e^{-x} \right) + \frac{1}{1+\alpha^2} \sum_{\omega=2}^{\infty} \alpha^\omega(1-\alpha)(1+(\omega-1)\alpha)e^{-(1+(\omega-1)\alpha)} \quad (2)$$

If 1-LOOKAHEAD is a statistically undetectable strategy, there exists a parameter  $\gamma > 0$  such that  $f_Z$  equals to the p.d.f. of  $\text{Exp}(\gamma)$ . However, the following Lemma shows that this is impossible.

► **Lemma 18.** *There is no  $\gamma > 0$  such that  $D_Z(\pi_{1\text{-LOOKAHEAD}}) = \text{Exp}(\gamma)$ .*

**Proof Sketch.** Assume by contradiction that equation (2) is an exponential distribution. i.e.,  $f_Z = \gamma e^{-\gamma z}$  where  $\gamma > 0$  is the amount of active stakes. Rewriting  $e^{(1-\gamma)z}$  and  $e^{(1-\omega)\alpha}$  according to the Taylor expansion of  $e^x = \sum_{\ell=1}^{\infty} \frac{1}{(\ell-1)!} x^{\ell-1}$ , the coefficient for the  $x^{\ell-1}$  must agree on all  $\ell \geq 1$ . This means that for all  $\ell \geq 1$ ,

$$\gamma^2(1-\gamma)^{\ell-1} = \frac{\alpha^\ell(1-\alpha)}{1+\alpha^2} \left[ \frac{1}{\ell} + \sum_{\omega=2}^{\infty} \alpha^{\omega-1}(1-\omega)^{\ell-1}(1+(\omega-1)\alpha)^2 \right]$$

We now take the absolute value on both sides, and show that the absolute value on the left hand side and the right hand side does not grow at the same rate with  $l$ . Therefore, we can conclude that  $f_{Z_r}$  cannot be an exponential distribution. ◀

## 4.3 Distribution of Consecutive Two Rounds are Negatively Correlated in 1-Lookahead

In this section, we apply the correlation test to 1-LOOKAHEAD and show that the distribution of consecutive two rounds are negatively correlated. In a high level, when the adversary successfully hides some credentials in round  $r$  and bias  $Q_{r+1}$  in round  $r$ , they have to do so by strategically hiding credentials with minimum scores. The distribution of scores in such a round stochastically dominates the honest distribution. However, the adversary only chooses to hide credentials because they can obtain credentials with lower scores in round  $r+1$ . Therefore, the distribution of scores in such a round is stochastically dominated by the honest distribution. This establishes a negative correlation between the scores in subsequent rounds. The Lemma states as follows:

► **Lemma 19.** *When the adversary uses 1-LOOKAHEAD strategy, the distribution of consecutive two rounds,  $D_{Z_r(\pi_{1\text{-LOOKAHEAD}})}, D_{Z_{r+1}(\pi_{1\text{-LOOKAHEAD}})}$  are negatively correlated. That is, for any numbers  $a, b$ ,*

$$\Pr_{r \leftarrow U[1, R]} [Z_{r+1} > b | Z_r > a] < \Pr_{r \leftarrow U[1, R]} [Z_{r+1} > b]$$

We defer the formal proof of Lemma 19 to the full version of the paper [5].

## 5 Profitable Strategies are Detectable

In this section, we will show that when the online stake remains constant throughout the protocol, *every* profitable strategy of an adversary with  $\beta = 0$  is detectable (and this holds for all  $\alpha$ ).

Let us first highlight a few complexities of detecting profitable manipulations. First, there are certainly undetectable non-profitable manipulations (for example, the adversary could simply never broadcast – this results in i.i.d. scores across rounds according to  $\text{Exp}(1 - \alpha)$ , and is indistinguishable from if the adversary were ‘non-strategically offline’). Second, note that a strategic adversary can look as far into the (hypothetical) future (assuming they win consecutive rounds) as they like when deciding which accounts to broadcast, and could try to carefully curate them to match a particular distribution. In general, CSSP induces a Markov Decision Process for the adversary, where each state is a countably long list of real numbers. 1-LOOKAHEAD witnesses that the MDP always has a strategy that outperforms honest, and we seek to understand whether any such strategy also satisfies a collection of complex constraints (and more over, there is not a single collection of constraints to satisfy – the adversary can pick any  $\gamma$  and satisfy the undetectability constraints to appear as i.i.d.  $\text{Exp}(\gamma)$ ).

Given the complexity of the strategy space in CSSP, our proof is surprisingly simple. Firstly, we make use of the following observation: since the adversary does not know the credentials owned by the honest miner before broadcasting their own,<sup>13</sup> in order to improve their probability of winning throughout the protocol, the adversary must on average broadcast credentials with smaller scores compared to when they are honest. Simultaneously, as discussed in Section 3, the observer expects the empirical score distribution to follow an exponential distribution (of undetermined rate  $\gamma$ ). Hence, in order to maintain undetectability, the adversary’s credentials must be distributed as an exponential with rate greater than 1.

However, [11] shows that unless the adversary controls almost half of the network, the adversary loses to honest participants in a non-trivial fraction of rounds. After such an event, the adversary loses their advantage gained before from strategic manipulation, and must participate as if the protocol has restarted. We call such rounds where adversary regains the perspective of a uniformly random seed “reset rounds”. In a reset round, we show that the adversary must broadcast credentials with scores at least as large as when honest. This leads to a contradiction – the tail of the “reset round” credential score distribution is already too fat for the credential score distribution of the adversary to be an exponential of rate greater than 1.

Our result also extends to the setting where the active stake fluctuates within  $1 \pm \delta$  factor, where we show that any undetectable strategies can only achieve limited profitability bounded by  $2\delta$ .

### 5.1 Detectability for Steady Online Stake

Throughout Section 5.1, we will assume that the online stake in each round remains constant, and equal to 1 (by normalization). Among all the online stake, the adversary holds  $\alpha$  stake, while the honest participants hold  $1 - \alpha$  stake. The outside observer knows that the total online state is steady across rounds, but does not know how much total stake is online.

We first prove that a profitable and undetectable adversary has a score distribution that is strictly dominated by  $\text{Exp}(\alpha)$ . We will use notations related to the minimum score of broadcast credentials that are formally defined in Definition 10.

---

<sup>13</sup>This is the key simplifying aspect of our proof that leverages  $\beta = 0$ .

► **Theorem 20.** *When the online stake remains constant throughout the protocol, for any adversary who holds  $\alpha$  stake and employs a profitable and undetectable strategy  $\pi$ , the adversary's broadcast score  $Y_r(\pi)$  from a random round  $r$  is distributed identically to  $\text{Exp}(\alpha + \epsilon)$  for some  $\epsilon > 0$ .*

**Proof.** Let  $X_r(1)$  and  $Y_r(\pi)$  be the minimum score of broadcast credential among honest miners and the adversary respectively, at a uniformly random round  $r$ . Then the overall minimum score at that round is  $\min\{X_r(1), Y_r(\pi)\}$ . Since the adversary must broadcast before observing the honest miner's credentials in round  $r$ ,  $X_r(1)$  is independent of  $Y_r(\pi)$ . By Definition 12 and Proposition 8, in order for the adversary's strategic attack to remain undetectable,  $\min\{X_r(1), Y_r(\pi)\}$  must distribute according to  $\text{Exp}(\gamma)$  for some  $\gamma > 0$ . Since  $X_r(1) \sim \text{Exp}(1 - \alpha)$  and by independence between  $X_r(1)$  and  $Y_r(\pi)$ , we have that for any  $z > 0$ ,

$$\begin{aligned} \Pr[\min\{X_r(1), Y_r(\pi)\} \geq z] &= e^{-\gamma z} \\ \implies \Pr[X_r(1) \geq z] \Pr[Y_r(\pi) \geq z] &= e^{-\gamma z} \\ \implies e^{-(1-\alpha)z} \Pr[Y_r(\pi) \geq z] &= e^{-\gamma z} \\ \implies \Pr[Y_r(\pi) \geq z] &= e^{-(\gamma - (1-\alpha))z} = e^{-(\alpha + (\gamma - 1))z}. \end{aligned}$$

Thus  $Y_r(\pi) \sim \text{Exp}(\alpha + (\gamma - 1))$ . The expected fraction of rounds that the adversary wins if they are honest is  $\alpha$ . Thus to be strictly profitable, the adversary needs to win with fraction  $> \alpha$ , which requires  $\Pr[Y_r(\pi) < X_r(1)] > \alpha$ . Since  $X_r(1)$  and  $Y_r(\pi)$  are exponential random variables with rate  $(1 - \alpha)$  and  $\alpha + (\gamma - 1)$ , by Lemma 29,

$$\alpha < \Pr[Y_r(\pi) < X_r(1)] = \frac{\alpha + (\gamma - 1)}{\gamma}.$$

The above equation implies that  $\gamma > 1$ . Thus we conclude  $Y_r(\pi) \sim \text{Exp}(\alpha + (\gamma - 1)) = \text{Exp}(\alpha + \epsilon)$  for some  $\epsilon > 0$ . ◀

Now, we show that for a non-trivial fraction of rounds, the adversary's score is drawn from a distribution that dominates  $\text{Exp}(\alpha)$ . We will need to reason about the adversary's score in reset rounds (defined in Section 4 at Definition 15), where the distribution of the seed  $Q_r$  is unbiased. For the reader's convenience, the formal definition of the reset round is restated here.

► **Definition 15** (Reset Round [11]). *A round  $r$  is a reset round if for all possible strategies  $\pi$ , the distribution of  $\{\Pr[Y_{r'}(\pi) \leq X_{r'}]\}_{r' \geq r}$  conditioned on  $Q_{r-1}$  and all historical information prior to round  $r - 1$ , is identical to the distribution of  $\{\Pr[Y_{r'}(\pi) \leq X_{r'}]\}_{r' \geq r}$  after replacing  $Q_r$  with a uniformly random draw from  $[0, 1]$ .*

[11] shows that the number of reset rounds are non-negligible.

► **Lemma 21** ([11], Theorem 4.1). *For  $\alpha < \frac{3-\sqrt{5}}{2} \approx 0.38$ , the fraction of rounds that is a reset round is strictly greater than 0.*

Meanwhile, we show that in a reset round, the adversary's output score distribution stochastically dominates  $\text{Exp}(\alpha)$ .

▷ **Claim 22.** Given that round  $r$  is a reset round, adversary's output distribution in round  $r$  must (weakly) stochastically dominate  $\text{Exp}(\alpha)$ .

## 30:20 Detecting Profitable Manipulations in Cryptographic Self-Selection

Proof. Let  $Y_r(\pi)$  be the broadcast minimum coin of the adversary using strategy  $\pi$  at a random round  $r$ . Notice that if  $r$  is a reset round, then  $Y_r(\pi)$  would be distributed according to  $\text{Exp}(\alpha)$  had  $\pi$  been an honest strategy. Let  $C_1, \dots, C_j$  be the score of credentials of all accounts that the adversary owns at round  $r$ , where  $C_1 \leq C_2 \leq \dots \leq C_j$ . Then for any  $z > 0$ ,

$$\Pr[Y_r(\pi) < z \mid r \text{ is a reset round}] \leq \Pr[C_1 < z \mid r \text{ is a reset round}] = 1 - e^{-\alpha z}$$

since  $C_1 \sim \text{Exp}(\alpha)$  after a reset round. Thus  $Y_r(\pi)$ 's distribution given that  $r$  is a reset round must (weakly) stochastically dominate  $\text{Exp}(\alpha)$ .  $\triangleleft$

Combining Theorem 20, Lemma 21 and Claim 22, we show a contradiction between above two properties that we have derived about a profitable adversary's score distribution. This shows no profitable adversary strategy is undetectable.

► **Theorem 23.** *When the online stake remains constant throughout the protocol and  $\alpha < \frac{3-\sqrt{5}}{2}$ , there is no profitable and statistically undetectable strategy.*

**Proof.** Given any adversary strategy  $\pi$ , let  $p_\pi$  be the fraction of rounds that is a reset round, by Lemma 21,  $p_\pi > 0$ . Let  $Y_r(\pi)$  be the broadcast minimum coin of the adversary at a random round  $r$ . Let  $Y_{rs}(\pi)$ ,  $Y_{non-rs}(\pi)$  be the broadcast minimum coin of the adversary at a random reset round and at a random non reset round respectively, as defined in Definition 15. Then  $Y_r(\pi)$  is a mixture of random variable  $Y_{rs}(\pi)$  and  $Y_{non-rs}(\pi)$ . Specifically,  $Y_r(\pi) = p_\pi \cdot Y_{rs}(\pi) + (1 - p_\pi) \cdot Y_{non-rs}(\pi)$ . Then for any  $z > 0$ ,

$$\Pr[Y_r(\pi) \geq z] = p \cdot \Pr[Y_{rs}(\pi) \geq z] + (1 - p) \cdot \Pr[Y_{non-rs}(\pi) \geq z].$$

By Theorem 20, for any undetectable strategy, it must be the case that  $\Pr[Y_r(\pi) \geq z] \leq e^{-(\alpha+\epsilon)z}$ . However, by Claim 22 and Lemma 21,  $p > 0$  and  $\Pr[Y_{rs}(\pi) \geq z] \geq e^{-\alpha z}$ , hence  $\Pr[Y_r(\pi) \geq z] \geq p \cdot e^{-\alpha z}$ . Since  $p$  is not dependent on  $z$ , it is impossible for  $\Pr[Y_r(\pi) \geq z]$  to be both at most  $e^{-(\alpha+\epsilon)z}$  and at least  $p \cdot e^{-\alpha z}$  for all  $z > 0$ .  $\blacktriangleleft$

## 5.2 Extension to Fluctuation in Online Stakes

In practice, limited fluctuation in participating stake of the protocol may be expected. In this subsection, we consider the case where the online stake in any round fluctuates within a  $1 \pm \delta$  multiplicative factor of the baseline online stake. By normalization, we assume the ground truth baseline online stake is 1, while the observer anticipates the online stakes to be in  $[(1 - \delta)\gamma, (1 + \delta)\gamma]$  for some  $\gamma > 0$ . We show that any adversary who profits beyond  $2\delta$  probability of winning is detectable. Our key observation is that Theorem 20 can be generalized to adversaries who enjoy profits beyond the online stake fluctuation range (as discussed below in Theorem 25).

► **Definition 24.** *An adversary with stake  $\alpha$  is  $\Delta$ -profitable if their probability of being the leader in a random round  $r$  is more than  $\alpha + \Delta$ .*

► **Theorem 25.** *When the online stake fluctuation is known to lie in all rounds within a multiplicative  $1 \pm \delta$  band of its baseline, for any adversary who holds  $\alpha$  stake and employs a  $2\delta$ -profitable and undetectable strategy  $\pi$ , the adversary's broadcast score  $Y_r(\pi)$  from a random round  $r$  is stochastically dominated by  $\text{Exp}(\alpha + \epsilon)$  for some  $\epsilon > 0$ .*

The proof of Theorem 25 can be found in the full version of the paper [5].

► **Theorem 26.** *When the online stake fluctuation is known to lie in all rounds within a multiplicative  $1 \pm \delta$  band of its baseline, no undetectable strategy is  $2\delta$ -profitable when  $\alpha < \frac{3-\sqrt{5}}{2}$ .*

**Proof.** The proof is identical to that of Theorem 23, where we establish that the adversary cannot produce a strategy whose broadcasts are dominated by  $\text{Exp}(\alpha + \epsilon)$ . The only difference is that we use Theorem 25 instead of Theorem 20 to conclude that this is necessary in order to be undetectable and strictly profitable. ◀

---

## References

- 1 Musab A. Alturki and Grigore Roşu. Statistical model checking of randao’s resilience to pre-computed reveal strategies. In Emil Sekerinski, Nelma Moreira, José N. Oliveira, Daniel Ratiu, Riccardo Guidotti, Marie Farrell, Matt Luckcuck, Diego Marmsoler, José Campos, Troy Astarte, Laure Gonnord, Antonio Cerone, Luis Couto, Brijesh Dongol, Martin Kutrib, Pedro Monteiro, and David Delmas, editors, *Formal Methods. FM 2019 International Workshops*, pages 337–349, Cham, 2020. Springer International Publishing.
- 2 Nick Arnosti and S. Matthew Weinberg. Bitcoin: A natural oligopoly. In Avrim Blum, editor, *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, volume 124 of *LIPICs*, pages 5:1–5:1. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICs.ITCS.2019.5.
- 3 Maryam Bahrani and S. Matthew Weinberg. Undetectable selfish mining. In *EC ’24: The 25rd ACM Conference on Economics and Computation, New Haven, CT, USA, July 8 - 11, 2024*. ACM, 2024.
- 4 Jonah Brown-Cohen, Arvind Narayanan, Alexandros Psomas, and S. Matthew Weinberg. Formal barriers to longest-chain proof-of-stake protocols. In *Proceedings of the 2019 ACM Conference on Economics and Computation, EC 2019, Phoenix, AZ, USA, June 24-28, 2019*, pages 459–473, 2019. doi:10.1145/3328526.3329567.
- 5 Linda Cai, Jingyi Liu, S. Matthew Weinberg, and Chenghan Zhou. Profitable manipulations of cryptographic self-selection are statistically detectable, 2024. arXiv:2407.16949.
- 6 Miles Carlsten, Harry A. Kalodner, S. Matthew Weinberg, and Arvind Narayanan. On the instability of bitcoin without the block reward. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 154–167, 2016. doi:10.1145/2976749.2978408.
- 7 Jing Chen and Silvio Micali. Algorand: A secure and efficient distributed ledger. *Theor. Comput. Sci.*, 777:155–183, 2019. doi:10.1016/J.TCS.2019.02.001.
- 8 Vanessa Chicarino, Célio Albuquerque, Emanuel Jesus, and Antônio Rocha. On the detection of selfish mining and stalker attacks in blockchain networks. *Annals of Telecommunications*, 75(3):143–152, 2020. doi:10.1007/s12243-019-00746-2.
- 9 Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography and Data Security*, pages 436–454. Springer, 2014.
- 10 Matheus V. X. Ferreira, Aadityan Ganesh, Jack Hourigan, Hannah Huh, S. Matthew Weinberg, and Catherine Yu. Computing optimal manipulations in cryptographic self-selection proof-of-stake protocols. In *EC ’24: The 25rd ACM Conference on Economics and Computation, New Haven, CT, USA, July 8 - 11, 2024*. ACM, 2024.
- 11 Matheus V. X. Ferreira, Ye Lin Sally Hahn, S. Matthew Weinberg, and Catherine Yu. Optimal strategic mining against cryptographic self-selection in proof-of-stake. In David M. Pennock, Ilya Segal, and Sven Seuken, editors, *EC ’22: The 23rd ACM Conference on Economics and Computation, Boulder, CO, USA, July 11 - 15, 2022*, pages 89–114. ACM, 2022. doi:10.1145/3490486.3538337.
- 12 Matheus V. X. Ferreira and S. Matthew Weinberg. Proof-of-stake mining games with perfect randomness. In Péter Biró, Shuchi Chawla, and Federico Echenique, editors, *EC ’21: The 22nd ACM Conference on Economics and Computation, Budapest, Hungary, July 18-23, 2021*, pages 433–453. ACM, 2021. doi:10.1145/3465456.3467636.

- 13 Amos Fiat, Anna Karlin, Elias Koutsoupias, and Christos H. Papadimitriou. Energy equilibria in proof-of-work mining. In *Proceedings of the 2019 ACM Conference on Economics and Computation, EC 2019, Phoenix, AZ, USA, June 24-28, 2019.*, pages 489–502, 2019. doi:10.1145/3328526.3329630.
- 14 Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, October 28-31, 2017*, pages 51–68. ACM, 2017. doi:10.1145/3132747.3132757.
- 15 Guy Goren and Alexander Spiegelman. Mind the mining. In *Proceedings of the 2019 ACM Conference on Economics and Computation, EC 2019, Phoenix, AZ, USA, June 24-28, 2019.*, pages 475–487, 2019. doi:10.1145/3328526.3329566.
- 16 Aggelos Kiayias, Elias Koutsoupias, Maria Kyropoulou, and Yiannis Tselekounis. Blockchain mining games. In *Proceedings of the 2016 ACM Conference on Economics and Computation, EC '16, Maastricht, The Netherlands, July 24-28, 2016*, pages 365–382, 2016. doi:10.1145/2940716.2940773.
- 17 Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, pages 357–388, 2017. doi:10.1007/978-3-319-63688-7\_12.
- 18 Sheng-Nan Li, Carlo Campajola, and Claudio J. Tessone. Twisted by the pools: Detection of selfish anomalies in proof-of-work mining. *CoRR*, abs/2208.05748, 2022. doi:10.48550/arXiv.2208.05748.
- 19 Silvio Micali, Michael Rabin, and Salil Vadhan. Verifiable random functions. In *40th annual symposium on foundations of computer science (cat. No. 99CB37039)*, pages 120–130. IEEE, 1999.
- 20 Michael Neuder, Daniel J. Moroz, Rithvik Rao, and David C. Parkes. Defending against malicious reorgs in tezos proof-of-stake. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies, AFT '20*, pages 46–58, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3419614.3423265.
- 21 Ayelet Sapirshstein, Yonatan Sompolinsky, and Aviv Zohar. Optimal selfish mining strategies in bitcoin. In *Financial Cryptography and Data Security - 20th International Conference, FC 2016, Christ Church, Barbados, February 22-26, 2016, Revised Selected Papers*, pages 515–532, 2016. doi:10.1007/978-3-662-54970-4\_30.
- 22 Zhaojie Wang, Qingzhe Lv, Zhaobo Lu, Yilei Wang, and Shengjie Yue. Forkdec: Accurate detection for selfish mining attacks. *Security and Communication Networks*, 2021:5959698, 2021. doi:10.1155/2021/5959698.
- 23 Aviv Yaish, Gilad Stern, and Aviv Zohar. Uncle maker: (time)stamping out the competition in ethereum. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023*, pages 135–149. ACM, 2023. doi:10.1145/3576915.3616674.
- 24 Aviv Yaish, Saar Tochner, and Aviv Zohar. Blockchain stretching & squeezing: Manipulating time for your best interest. In David M. Pennock, Ilya Segal, and Sven Seuken, editors, *EC '22: The 23rd ACM Conference on Economics and Computation, Boulder, CO, USA, July 11 - 15, 2022*, pages 65–88. ACM, 2022. doi:10.1145/3490486.3538250.

## **A** Relavent Properties for Exponential Distributions

► **Lemma 27** ([11], Lemma 2.1.). Let  $S(x, \alpha_i) := \frac{-\ln(x)}{\alpha_i}$ . When  $x$  is drawn uniformly from  $[0, 1]$ ,  $S(x, \alpha_i)$  is identically distributed to  $\text{Exp}(\alpha_i)$ .

► **Lemma 28** ([11], Lemma A.1.). *Let  $X_1, \dots, X_n$  be independent random variables where  $X_i$  is drawn from  $\text{Exp}(\alpha_i)$  for some  $\alpha_i > 0$ . Then  $\min_{i \in [n]} \{X_n\}$  is identically distributed to  $\text{Exp}(\sum_{i=1}^n \alpha_i)$ .*

► **Lemma 29** ([11], Lemma A.2.). *Let  $X_1, X_2$  be two independent random variables drawn from  $\text{Exp}(\alpha_1), \text{Exp}(\alpha_2)$  respectively, where  $\alpha_1, \alpha_2 > 0$ . Then  $\Pr[X_1 < X_2] = \frac{\alpha_1}{\alpha_1 + \alpha_2}$ .*

► **Lemma 30** ([11], Lemma 4.3.). *Let  $X_1, X_2, \dots$  be i.i.d. copies of an exponentially distributed random variable such that  $\min_{n \in \mathbb{N}} X_n$  is exponentially distributed with rate  $\alpha$ . Then, for all  $i \in \mathbb{N}$ , the random variable  $Y_i = \min_{n \in \mathbb{N}}^{(i)} X_n$  is identically distributed to  $Z_i = Z_{i-1} + \text{Exp}(\alpha)$  where  $Z_0 := 0$ .*

► **Lemma 31** ([11], Lemma 4.4.). *Let  $Y_1, Y_2, \dots$  be i.i.d. copies of an exponentially random variable such that  $\min_{n \in \mathbb{N}} Y_n$  is exponentially distributed with rate  $\alpha$ . Let  $X$  be exponentially distributed with rate  $1 - \alpha$ . Let  $W = \{i \in \mathbb{N} : Y_i < X\}$ . Then  $\Pr[|W| = \ell] = \alpha^\ell (1 - \alpha)$ .*

