sciendo

Proceedings on Privacy Enhancing Technologies ; 2020 (3):175–203

Thien-Nam Dinh*, Florentin Rochet*, Olivier Pereira, and Dan S. Wallach

Scaling Up Anonymous Communication with Efficient Nanopayment Channels

Abstract: Tor, the most widely used and well-studied traffic anonymization network in the world, suffers from limitations in its network diversity and performance. We propose to mitigate both problems simultaneously through the introduction of a premium bandwidth market between clients and relays. To this end, we present moneTor: incentivizing nodes to join and support Tor by giving them anonymous payments from Tor users. Our approach uses efficient cryptographic nanopayments delivered alongside regular Tor traffic. Our approach also gives a degree of centralized control, allowing Tor's managers to shape the economy created by these payments. In this paper, we present a novel payment algorithm as well as a data-driven simulation and evaluation of its costs and benefits. The results show that moneTor is both feasible and flexible, offering upwards of 100% improvements in differentiated bandwidth for paying users with near-optimal throughput and latency overheads.

Keywords: Tor, cryptocurrency, payment channels

DOI 10.2478/popets-2020-0048
Received 2019-11-30; revised 2020-03-15; accepted 2020-03-16.

1 Introduction

Anonymous traffic routing through Tor remains one of the most popular low-latency methods for censorship evasion and privacy protection [1]. In this setup, clients protect both their TCP/IP metadata and content by routing their traffic through an onion-encrypted path with three randomly selected volunteer relay nodes, referred to as a circuit. The Tor network presently consists of $\approx 6,400$ relays contributing over 160 Gbit/s of bandwidth globally [2]. While Tor has proven to be a highly effective option for privacy-seeking users, it suffers from two important issues that are relevant to this work. First, Tor is vulnerable to a broad variety of traffic correlation attacks [3, 4], where an attacker, controlling multiple nodes or network vantage points, will have a significant chance of occupying key roles in a circuit, making deanonymization possible. Second, Tor has scalability issues, leading to traffic congestion [2, 5].

While it may be possible to improve the engineering of Tor's cryptographic protocols [6] or scheduling [7], such measures are only a stopgap if the Tor network cannot add capacity quickly enough to support its users' growing demands. Consequently, it is straightforward to see Tor's issues as an economic question: how do we *incentivize* people to add more servers to the Tor network? Running a network server fundamentally entails real-world costs for the hardware, electricity, and bandwidth. That money has to come from somewhere.

To address this, we present *moneTor*: a monetary design which allows relays to offer a *premium bandwidth* product to Tor users in exchange for cryptographic currency tokens. These payments create incentives for operators to add additional capacity to the Tor network. Of course, the whole concept of onion routing requires Tor relay and exit nodes not to know the identity of the sender, so moneTor adds mechanisms to maintain the anonymity of these payments. We will demonstrate that this is possible while maintaining standard properties that should be expected of any cryptocurrency (e.g., scarcity, fungibility, divisibility, durability, and transferability) [8, p.3].

Tor also has the curious property that it is not fully decentralized. Even though Tor nodes, themselves, are located around the world and operated by many different organizations, The Tor Project centrally tracks the health of the Tor network, provides software distributions, and publishes lists of active Tor nodes. This partial centralization creates opportunities for a "fiscal policy" to manage these cryptographic payments. For example, the Tor Project could impose "taxes" for a variety of purposes, such as providing a baseline of financial support to all Tor node operators, and could evolve these rules over time in response to changing needs.

***Co-Primary and Corresponding Author: Thien-Nam Dinh:** Sandia National labs, E-mail: thidinh@sandia.gov
***Co-Primary and Corresponding Author: Florentin Rochet:** UCLouvain Crypto Group, E-mail: florentin.rochet@uclouvain.be
Olivier Pereira: UCLouvain Crypto Group, E-mail: olivier.pereira@uclouvain.be
Dan S. Wallach: Rice University, E-mail: dwallach@rice.edu

Our work aims to solve the challenge of building moneTor in the presence of the following constraints:

- *Anonymity*: The payment system may not, in any way, compromise Tor's primary mission to protect user anonymity.
- *Payment Security*: The payment system must satisfy common security properties expected of any cryptographic currency scheme.
- *Efficiency*: The payment system must be lightweight, low-latency, and scalable, to accommodate the dynamic and bursty nature of Tor traffic.

We recognize that the addition of monetary incentives into Tor would involve sensitive legal, economic, and sociopolitical considerations. Exact fiscal policy recommendations are beyond the scope of our work, but we do discuss the risk, merits, and versatility that moneTor could provide in Section 7.

Contributions. This work describes a full-stack framework for tokenized Tor incentives. MoneTor addresses the challenge of applying theoretical advances in cryptocurrency research to the concrete constraints and complexities of the live Tor network. We introduce highly-efficient payment protocols which facilitate the novel concept of *locally transparent nanopayment channels*. During data exchange, our distributed payment processing procedure completely shifts expensive CPU operations off of the critical path, incurring negligible computational costs (a single hash operation) per payment. State-of-the-art throughput is made possible by a global payment infrastructure that utilizes *trustless intermediaries* to handle the added CPU load, potentially in exchange for monetary rewards. The moneTor scheme adheres to the standard Tor security model and conforms to our domain-specific constraints of *Anonymity*, *Payment Security*, and *Efficiency*.

We provide a prototype of our payment layer and an extension to the existing routing protocol, resulting in approximately 15k lines of new C code within the Tor codebase. Our networking experiments demonstrate thousands of transactions per second, avoiding any additional latency through mechanisms including preemptive channel creation. We also discovered, through experimental simulations, that scheduling approaches [9, 10] from previous Tor incentives systems do not adequately provide differentiated service for prioritized traffic. Consequently, we present a new prioritization mechanism that achieves our traffic shaping objectives by changing the size of control-flow windows.

Our Github repository [11] contains all of the data and code necessary to reproduce our research results.

2 Background

Traffic Analysis. Tor's threat model assumes an adversary who passively observes some fraction of the encrypted Tor network traffic as well as operating some fraction of Tor's onion routers, creating opportunities to observe and manipulate user streams. Introducing delays, in particular, can give these adversaries significant power to deanonymizing traffic flows (see, e.g., [12, 13]). Tor does not implement explicit countermeasures for this class of attack, since most countermeasures would introduce additional latency.

Increasing the number and diversity of Tor nodes reduces the power of these adversaries. While our work is not explicitly targeted at defending Tor against traffic analysis, if it creates incentives for more nodes to participate, it would additionally improve Tor against this class of attacks.

Circuit Handling on Tor Clients. An important engineering goal of Tor is to reduce latency. In engineering moneTor, we face the same challenge—ensuring that any communication or computation added by moneTor has a minimal impact on latency. Tor, itself, addresses these issues by doing work in advance of when it's needed. For example, Tor will construct its overlay circuits in advance and keep those circuits idle. When a user application wants to communicate, an idle circuit will be ready to go, leading to a fast user experience. MoneTor piggybacks on this design, building *preemptive payment channels* to ensure that payments can begin immediately.

Flow Control. Another engineering goal of Tor is to manage and optimize flow rates to ensure fair sharing of the available bandwidth. Tor uses a sliding-window flow control system, similar to that used by TCP/IP, but the window size is fixed. This helps control Tor's maximum use of bandwidth, but can have a substantial impact on network performance. A variety of research efforts have worked to characterize and improve this situation (see, e.g., [14, 15]). MoneTor must also make changes to Tor's flow control system to provide prioritized service to paid traffic, yet still preserve fairness.

Payment Channels. MoneTor requires efficient and anonymous micropayment channels, potentially making tiny incremental payments alongside every transmitted network cell. Many common cryptocurrency protocols can support only tens of transactions per second [16], which is clearly inadequate for our use case. One popular workaround is an off-chain approach popularly known as "Lightning Networks" [17]. In this setup, two parties—A and B—each create a special escrow transactions on the ledger to setup a simple payment channel between them. These parties may then proceed to make bidirectional micropayments to each other *without ledger interaction* through the exchange of signed "I Owe You" tokens. A related variant, called a "tripartite payment channel" adds a mutually-trusted *intermediary* I, avoiding any need for A to trust B or vice-versa. Such schemes are secure if they satisfy the following requirements:

1. At every step of the protocol, all parties possess proof of execution of the last finalized payment.
2. Given two proofs of payment state, the network can unambiguously identify the more recent state.
3. When A pays B through I, the payment is atomic. That is, there is never a situation in which I pays B but is unable to extract the agreed-upon payment from A.

Lightning network designs have useful scalability properties, but moneTor also needs anonymity, since the Tor middle and exit relays should have no way to leverage the payment system to identify the payer associated with each Tor circuit.

Several recent cryptocurrency designs have both scalability and anonymity features. Tumblebit is a channel-like mixing protocol for Bitcoin that allows fast and anonymous off-chain payments [18]. Malvavolta et al. [19] also describe a variant on payment channels providing Tor-like privacy. Their scheme preserves both sender and receiver privacy, assuming at least one trusted intermediary. Neither of these schemes is ideal for our purposes. Tumblebit requires unrealistic synchronization between payment parties for the Tor environment. Malvavolta et al. introduces additional parties who could collude to compromise user privacy.

To satisfy the needs of moneTor, we instead started with Green and Miers's Bolt protocol [20]. Bolt is a tripartite anonymous channel with efficient zero-knowledge proofs that provide sufficient privacy and an adequate starting point for scalability. As we describe in Section 3.2, moneTor introduces an additional layer to achieve our full scalability requirements.

3 Payment Design

3.1 Ledger

In our payment design, we follow the Bitcoin paradigm in which all users produce signed statements, using public key cryptography, as the basis of making operations with their wealth [21]. However, our system does not need to rely on Bitcoin's decentralized blockchain, or its energy-intensive proof of work system, to reach a consensus. Instead, we note that Tor already relies on a centralized set of authorities to manage Tor, for example publishing a list of Tor nodes. Our design extends this role, having Tor's central authorities also maintain the global payment state as a public tamper-evident database (see, e.g.,Crosby et al. [22]). This design can also be extended in a variety of ways, for example, distributing it across several trusted authorities (see, e.g., RSCoin [23]). Ultimately, any transition from moneTor (this research effort) to broader use by real Tor users, would be able to adopt any of a variety of different cryptographic payment systems. Instead, the focus of our research is on designing efficient "off-ledger" payments, which can be processed at the granularity of data packets flowing through Tor, and only later reconciled against the "real" payment system.

3.2 Payment Protocols Overview

In this section, we specify the protocols that comprise the moneTor payment infrastructure. As first discussed in Section 2, our chosen model is an implementation of a "tripartite anonymous payment channel." Compared to purely centralized schemes, such channels are a critical method for scaling, allowing for a theoretically unbounded number of off-ledger transactions between any two parties. However, in a naïve two-party implementation, the total channel management complexity is on the order of $O(n \times m)$ where n is the number of Tor clients and m is the number of relays. Our solution is to introduce the Intermediary Relay, a Tor node whose only role is to provide *atomic* payment channel services between clients and relays. By acting as trustless payment hubs maintaining persistent channels to many users, their services reduce the channel complexity to $O(n + m)$. Notably, we can adjust the target number of intermediaries to balance the performance of the payment infrastructure and the size of the anonymity set for connected premium users.

Bolt. The basis of our scheme is an extension of Bolt's anonymous micropayment channel protocol, which is itself a privacy-focused adaption of the Lightning Network [17]. Due to its importance as a starting point for our work, we first provide a brief outline of the prerequisite micropayment channel procedures defined in Bolt. All protocols are either two or three party interactions between a subset of the following roles: C (client), R (relay), E (end-user: either a client or relay), I (intermediary), and L (ledger). For our use case, assume that all communications are anonymously routed through Tor circuits [20].

KeyGen: Generates a cryptographic keypair.
Init-E: E initializes half of a micropayment channel by escrowing funds on L.
Init-I: I initializes half of a micropayment channel by escrowing funds on L.
Establish: E and I interact to establish a new micropayment channel from their respective halves.
Pay: C interacts with I and R to send a single micropayment to R.
Refund: E closes a channel on L and makes a claim on the escrowed funds.
Refute: I closes a channel on L and makes a claim on the escrowed funds.
Resolve: L determines the final balance of funds awarded to each party.

While anonymous micropayment channels present a tremendous advance for many applications, the relatively heavy cryptography (37-100 ms) and communication (7 messages) is prohibitively expensive, especially if done for every message transmission[1]. To overcome this barrier, we present a new payment layer design enabling far more efficient *nanopayments*.

moneTor. The moneTor design uses the existing anonymous micropayment structure to build *locally transparent nanopayments*. In this model, clients and relays extend single micropayment operations into *nanopayment channels* through an intermediary (see Figure 1). These lightweight channels allow the client to send n unidirectional nanopayments to the relay through the established Tor circuit. Each payment represents a fixed value δ, established at the start of the channel. By *locally transparent*, we mean that each nanopayment is trivially linkable to all other payments in the same channel. However, the nanopayment channels themselves are unlinkable to other nanopayment channels and micropayment operations. It is by design that this channel anonymity guarantee fits with Tor's existing circuit framework and security model, which similarly stipulates that messages within circuits are linkable internally but not externally to other circuits [2]. In essence, the overarching motivation of our work is to relax the costly anonymity guarantees provided by Bolt toward the design of a new set of protocols specifically adapted for Tor. Finally, note that establishing and closing nanopayment channels do not require the clients or relays to interact with the ledger, resulting in a far more scalable design than systems which depends on resource-intensive centralized entities.

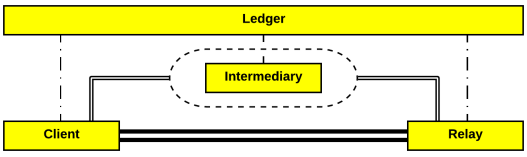


Fig. 1. Payment Roles — Dashed lines represent periodic transactions (rare), thin double lines indicate micropayment channels (used at the beginning and end of circuit lifetime), and thick double lines indicate a nanopayment channel (handling nanopayments during the lifetime of the circuit). The dashed outline around the intermediary represents a notion of payment anonymity for the end-users. Connections to the ledger and to the intermediary are protected by an internal Tor circuit.

We now briefly describe our new protocols. Any two parties C and R can construct a nanopayment channel once both have completed Bolt's *Establish* with a common intermediary I. We define the following set of protocols needed to manage nanopayments:

Nano-Setup: C and I interact to prepare an incomplete half of a nanopayment channel on top of their existing micropayment channel.
Nano-Establish: C sends her nanopayment channel information to R, who interacts with I to complete the second half of the nanopayment channel on top of R's existing micropayment channel.
Nano-Pay: C sends a single nanopayment to R. This is repeatable for up to n operations.

1 In addition to concerns regarding global network overhead, it is also desirable to keep the barrier of entry low for smaller Tor relay operators.

2 In contrast, as discussed in Section 6, some prior works implemented payments schemes which were needlessly unlinkable from the relay's viewpoint.

Nano-Close-R: R closes his nanopayment channel with I.
Nano-Close-C: C closes her nanopayment channel with I. This step must happen after *Nano-Close-R*.

We also need a conflict resolution procedure to ensure secure closure properties for the nanopayment scheme, because a malicious party can force a closure (i.e., abort) at any time. We must ensure that these actions result in a consistent result (i.e., never penalizing honest parties). As such, we must extend our protocol to allow L to resolve disputes:

Nano-Refund: E closes the channel on L.
Nano-Refute: I closes the channel on L.
Nano-Resolve: L makes a final determination on both outstanding micropayment and nanopayment balances.

Our nanopayment scheme is inspired by the classic *Payword* two-party micropayment scheme in which payments are encoded by successively revealed preimages in a precomputed hash chain [24].

The challenge in using Payword is to securely integrate the hash chain concept into an existing three-party anonymous micropayment channel setup such that all parties maintain secure cryptographic ownership of their funds at all steps. At the same time, we must ensure the scheme does not leak deanonymizing information outside of the nanopayment channel context, a nontrivial task that requires significant restructuring of the Bolt protocol to achieve our constraints. Our final solution incurs an overhead penalty of approximately two micropayment operations per nanopayment channel, one at the beginning and one at the end of the channel life cycle.

3.3 Nanopayment Protocol Details

In this section, we summarize the intuition for the basic steps in the payment protocol. A more formal, detailed description of the algorithms is provided in Appendix B. Security considerations are detailed next in Section 3.4 and formally described in Appendix C.

Nano-Setup At the start of this protocol, C has access to a micropayment wallet w obtained from Bolt's **Establish** that enables her to operate her micropayment channel with the Intermediary I as well as a refund token rt that entitles her to claim her current funds on the ledger L in the event that I misbehave or goes offline. To construct a nanopayment channel, C first generates an array of values hc of length n where $hc_i = H(hc_{i+1})$ and hc_n is a random number. The root of the hash chain hc_0 is used to create a globally unique nanopayment token nT that encodes the public parameters of the channel including the length n and the per-payment value δ. C sends I a commitment to a fresh nanopayment channel parametrized by nT along with a zero-knowledge proof of the following statements:

1. The nanopayment wallet nw is well-formed from w.
2. C has ownership of a micropayment channel containing at least $n \times \delta$ funds.

I verifies these messages and supplies C with a new signed refund token nrt that entitles C to cash out the full balance of the micropayment channel using **Nano-Refund** if needed. C, now protected against misbehavior by I, agrees to send a revocation token σ_w, which revokes her right to use w to create other nanopayment channels from this micropayment wallet or to use rt to cash out the micropayment channel before the **Nano-Close** is run. I is now protected against double spending by C and can safely inform C that the nanopayment channel has been set up successfully.

Nano-Establish: At this point, C sends R the same nT token used to setup the channel with I. R uses the token to initiate her end of the nanopayment channel with I by executing essentially the same procedure that C used in *Nano-Setup*. The nanopayment channel is now fully established and ready to be used. A key observation is that both ends of the channel (C-I and R-I) are rooted at the same hash chain root hc_0. This design ensures that any attempts to prematurely close either channel—which requires revealing hc_0 to the network—would enable parties to close the other channel as well.

Nano-Pay: To make the i^{th} payment, C simply sends the next hash preimage hc_i to R. Knowledge of this preimage hc_i is sufficient for R to prove possession of a nanopayment. At any given time, R can broadcast the tuple (nrt, hc_i) to L to prove ownership of the correct balance of funds. Notice that this action simultaneously reveals hc_i to I, who can then claim an equivalent value of funds from C. As a result, the scheme satisfies a correct-by-construction property of *atomicity* whereby both legs of the protocol are finalized at the same time.

Nano-Close: After some number of payments $k < n$ has transpired and C wants to close the Tor circuit, both C and R will generally prefer to close their nanopayment channels through I. In this process, the R-I leg must be

closed before the C-I leg. This is due to the unidirectional nature of nanopayment channels. Since payments are flowing from C to R, I must first determine its debt to R in order to know how much it can claim from C.

R first sends to I a commitment to a new micropayment wallet w' and a zero-knowledge proof of the following statements:

1. w' is well-formed from w (w was either created by Bolt's establish phase or by a previous moneTor Nano-Close).
2. The balance of w' is equal to the sum of the balance from the previous wallet w and $\delta \times k$.

Once verified, I issues a refund token rt' on the new funds. R agrees to invalidate the nanopayment channel by issuing a revocation token σ_{nw} to I. I and R proceed to create a blind signature on w' thus validating the wallet for future use.

Once I has closed his nanopayment channel leg with R, I and C are free to complete the close protocol. All parties now revert to the original state preceding *Nano-Setup* save for a securely updated balance.

Nano-Refund, Nano-Refute, Nano-Resolve: Honest parties will not typically close active nanopayment channels on the ledger, opting instead to run Bolt micropayment closure procedures when they wish to cash out. However, in the event of malicious behavior or premature termination, *Nano-Refund* and *Nano-Refute* enable E and I to withdraw funds on the ledger with the latest payment information at any time. After a set amount of time has passed allowing the counterparty to reciprocate, the ledger runs *Nano-Resolve* to make a final publicly verifiable determination on the final balance. Correct execution of these procedures allows all honest parties to claim the correct amount of funds according to the payment history. In some cases, it might even be desirable to penalize clearly malicious users by transfering the full amount of their initial balance to the counterparty.

3.4 Payment Security and Anonymity

Our security model accounts for both privacy and payment security. The privacy threat model derives from the local active adversary paradigm ubiquitously studied in Tor research [1]. Like all cells in Tor, Nanopayment messages are locally linkable by relays participating in the circuit. However, since each circuit is only ever associated with one anonymous nanopayment channel at any given time, relays and intermediaries cannot link two nanopayment channels with the same user. Furthermore, Tor circuits protect all communication in the tripartite nanopayment protocol. Hence, the relationship between client to intermediary, client to ledger, relay to intermediary, and relay to ledger are themselves anonymized. We provide formal definitions and proofs for the following theorem in Appendix C.

Theorem 1. *The nanopayment channel scheme offers anonymity (C.1.1, C.1.2) and secure balance (C.1.3) under the assumptions that the commitment scheme is secure, the zero-knowledge system is simulation extractable and zero-knowledge, and the hash function used to create the hashchain and verify the preimage during Nano-Pay is modeled as a random oracle.*

This theorem fully covers the security and anonymity characteristics of the protocol leading up to the closing of a micropayment channel. However, two points must be made with consideration to potential deanonymization after the conclusion of the protocol. First, the price revealed to the ledger at micropayment close enables a subtle passive attack by I. By examining the final number of payments made on each channel in conjunction with the globally fixed nanopayment cost, I may potentially link all of C's nanopayment channels.[3] To mitigate this vulnerability, we stipulate that C must make at least one micropayment, which has a monetary value hidden from I, before closing a micropayment channel. We stipulate that this micropayment should contain a random value not greater than the channel escrow maximum value as stated in the Tor consensus and may be made to another account owned by C.

Secondly, in the event of a dispute resolution on the ledger, the two parties on either end of the micropayment channel (C-I or I-R), must disclose their on-ledger identities. However, given an adequately privacy-preserving choice of an on-ledger transaction protocol such as Zerocash, even this result would be meaningless for R [25]. Thanks to the built-in ledger privacy, R can

3 This attack is best illustrated with a trivial example. Suppose that I facilitates a number of nanopayment channels with the following number of payments, each of which is known to represent one unit of money: $[58, 839, 356, 881, 23, 89, 561]$. Now C closes her micropayment channel and terminates with exactly $58 + 356 = 414$ units of money. Once the micropayment channel is closed, I must necessarily gain knowledge of the final balance of funds and can easily link the first and third nanopayment channels as belonging to the same C.

only link the ledger interaction to its circuit, but not to the client's real-world identity.

Having accounted for all privacy considerations both during and after the protocol, we informally state the following anonymity guarantees relative to unmodified Tor:

1. Additional parties needed to operate the moneTor system (i.e., ledgers and intermediaries) cannot extract any additional information about a given client than any middle relay.
2. Excluding side channels, circuits do not leak any information other than the single bit needed to differentiate premium and nonpremium users.

Our threat model for payment security is similar to those found in prior work related to blockchain micropayment channels [17]. In such models, the user is protected from malicious intermediaries by the ability to prove misbehavior to a global ledger. Our protocols also guarantee that the client would not risk more money than initially agreed-upon (Balance property, see Section C.1.3).

Side-channels on micropayment events. The moneTor protocol prescribes unlinkable micropayment events by protocol design. However, a naïve implementation of this scheme may be susceptible to side-channel vulnerabilities, most notably, timing attacks. For instance, a channel-creation policy that immediately mints a new nanopayment channel after closing of the previous channel would allow I to trivially link all channels belonging to C. To mitigate this particular risk, we require a random delay t_r uniform $\in [0, r]$. Since our scheme specifies the availability of preemptive channels, such a delay should not adversely impact the user experience. User privacy with respect to side-channels and other statistical techniques depend on the number of users concurrently building channels to the same intermediary. If we require an anonymity set of size N, then the maximum number of Intermediaries allowed in the network is $\frac{\mathbb{E}[W]}{N}$ where W is a discrete random variable representing the number of payment events between any t_i, the time at **Nano-Close** and t_{i+1}, the time of the next **Nano-Setup** or **Nano-Establish**. The number of these events should be counted within a sliding time window of size r. Consequently, the number of active Intermediaries should be a parameter of the system that the Tor project determines based on the activity of premium users. Furthermore, clients should maintain channels with an number of different intermediaries to further mitigate any other unexpected side-channel leaks.

3.5 Economic Considerations

In creating the technical scheme, we explicitly considered several key economic factors. For instance, if deployed, the Tor Project would need to decide how to assign value to the moneTor tokens, a choice that entails fundamental trade-offs between control, liability, and social perception. Here, we enumerate three broad categories of options along with some comments about the technical implementations:

1. The Tor Project reserves control of monetary supply for the purpose of enforcing a publicly declared policy, for instance, to peg the value of tokens against one or more fiat currencies it that it holds in reserve. This option would require the Tor Project to reserve a privileged signing key for minting new moneTor tokens (e.g., when a user makes a fiat deposit).
2. The moneTor tokens are instantiated as a standard cryptocurrency whose value fluctuates as a function of market pressures and the chosen distribution schedule. As with any cryptocurrency, a node that rejects the distribution mechanism (e.g., mining in Bitcoin) would violate the protocol.
3. The moneTor tokens act as a secure wrapper for an external cryptocurrency such as Bitcoin, Ethereum, or a future state-backed currency. This option is made possible by ongoing work in the field of ledger interoperability protocols [26, 27].

In all three cases, note that the mechanism required to implement the policy takes place entirely on the ledger. Consequently, the off-ledger moneTor payment layer, which is the main contribution of this work, is compatible with all of these three options.

While the Tor Project provides several acceptable options for monetary policy, the pricing mechanism for premium bandwidth should be standardized, since any price differentiation between circuits will inevitably leak information. This privacy issue becomes more significant with higher granularity payment options. Therefore, we impose the constraint that all users should pay a single uniform price for premium bandwidth at any time t.

It is of course central to make sure that moneTor serves the goal of advancing of human rights and freedom of the Tor Project, which may be quite differ-

ent from those that would result from a purely profit-seeking environment. To this end, moneTor includes an explicit taxation mechanism, which is described in Section 3.6. Under this scheme, any time a payment takes place within the moneTor layer, a tunable fraction of the revenue is diverted into a special account controlled by the Tor Project. This is analogous to a secure and automatic sales tax on premium traffic payments. The Tor Project can use this fund to shape the topology of the network towards some notion of desirable diversity and performance via a transparent policy. The exact content of such a policy is an active subject of research that is orthogonal to our paper.[4] The tax rate also dictates incentives for nodes seeking to game the system, perhaps by inserting dummy traffic to collect tokens or faking bandwidth measurements. We explore such questions and trade-offs in Section 7.

3.6 Tax Integration

The zero-knowledge setup provides an elegant way to anonymously handle the tax collection policy. Thus far, we have treated the nanopayment value δ as symmetric for both the client and relay leg. In practice, it requires only a trivial modification to specify separate values of δ_C and δ_R such that the following equality is satisfied.

$$\delta_C - \delta_R = tax + fee \tag{1}$$

Here, *tax* is the portion of every payment that is redirected to the Tor tax authority while *fee* represents compensation for I's services. I gradually accumulates these overhead charges in his balance over the course of running many nanopayment channels. When it is time for I to cash out the full micropayment channel, L simply divides the funds between the I and the tax authority. Note that this does not mean that L can arbitrarily control money, as this process is well-defined in the setup of the network protocol.

3.7 Integration in Tor Circuits

Up to this point, we have described payments that occur between a single client and a single relay. In practice, it is typical for each client to maintain several active circuits concurrently,[5] each of which requires three streams of payments to the guard, middle, and exit relays. These channels must be actively managed to optimize computational overhead as well as money flow. Furthermore, connections between the client and the guard relay are transparent and persistent across the timescale of several months. We optimize for this situation by enabling transparent and direct payment channels between the client and guard, which considerably reduces the time needed to establish or close the channel. In contrast, middle or exit relay channels require the flexibility of our full tripartite scheme.

4 Network Design

4.1 Pre-built Channels

By default, Tor attempts to pre-build circuits to reduce latency once a user wishes to create a data stream. Much like circuits, moneTor payment channels are high in initial latency because of the multiple in-out messages in the protocol. To solve this problem, we exploit the same circuit build strategy by preemptively setting up and establishing payment channels on clean pre-built circuits. This approach dramatically reduces the time-to-first-payment. Unfortunately, the excessive establishment of preemptive channels introduces network overhead. Our implementation features a basic prediction strategy to balance this trade-off by using historical usage date to anticipate the required number of channels. The approach is similar to the way in which Tor anticipates the need for a fresh circuit. In Section 5, we analyze moneTor's preemptive channels approach and show that the payment confirmation time matches the round-trip-time of the client-relay connection, as expected.

4.2 Prioritized Traffic

Traffic scheduling is perhaps the most intuitive mechanism with which to implement prioritization. However, our preliminary experiments found that local scheduling decisions on each relay for priority do not work well

4 E.g., Waterfilling [28] argues for security by maximum diversity in endpoints of user paths, and TAPS [29] argues for security by trust policies.

5 For instance, the popular Tor Browser user application typically does not share circuits with streams targeting a different destination address unless those streams come from the same SOCKS connection.

with the current Tor network, which precludes the use of out-of-the-box scheduling approaches based on DiffServ [9] and EWMA [10]. Intuitively, this behavior is a direct result of the evolution of the Tor network capacity in recent years. The growth in bandwidth across guard and middle relays produces more congestion between the exit relay and the final destination. We simulated Tor's topology to analyze scheduling and found that relays were able to instantaneously flush their queues at each "write" event, rendering any attempt at local scheduling to be ineffective. These results, detailed in Appendix D, may suggest the need for a separate comprehensive study of network prioritization mechanisms.

Consequently, we turn to the alternative strategy of prioritizing traffic through Tor's internal control-flow window sizes. Unlike scheduling-based approaches, a window-based approach might be more accurate under conditions with lower internal congestion [30, 31]. Indeed, since local decisions inside the scheduler at a particular relay may fail to achieve priority, we need to design priority as a global function of the circuit. Recall that edge nodes regulate the traffic flux in either direction using a set of flow control windows. Roughly speaking, these windows determine the space allotted to each circuit on a relay's scheduling queue, which in turn positively correlates with effective bandwidth. We implement our prioritization scheme by statically readjusting the window maximum sizes once according to the following formula for both *Circ window* and *Stream window*.

$$window' = window(1 + \alpha(premium/pr\% - 1)) \quad (2)$$

Here, a circuit is marked as prioritized by the bit $premium \in \{0, 1\}$. The tunable priority benefit $\alpha \in [0, 1]$ defines the proportion of the non-premium capacity that we wish to transfer to premium clients. By accounting for $pr\% \in [0, 1]$, the fraction of premium to nonpremium clients, we can keep the total flow capacity constant. It follows that the relay memory consumption induced by processing cells should stay constant too.

Even if most relays can flush all queues at each "write" event, some relays may still suffer from congestion within the Tor network. In this case, modifying Tor's overlay flow control will not achieve priority since the cells are stuck within the congested relay's queues. To overcome this issue, we modify EWMA with a linear scaling factor that favors paid circuits.

$$A_{t+\Delta t} = A_t \times 0.5^{\Delta t/H} \quad (3)$$

$$A'_{t+\Delta t} = A_{t+\Delta t}/\beta + C_{t,t+\Delta t} \quad (4)$$

Defined in Tang and Goldberg's original paper [10], A is a variable score used to sort circuits such that the circuit with the lowest A is always next on the scheduling queue. C is the number of cells relayed within Δt, the time that has passed since the previous observation, and H is a global parameter representing the half-life decay interval. Our added term, $\beta \in [1, \text{inf})$, is a tunable parameter such that $Bandwidth_{premium} = Bandwidth_{nonpremium} \times \beta$ for any given circuit under ideal conditions.

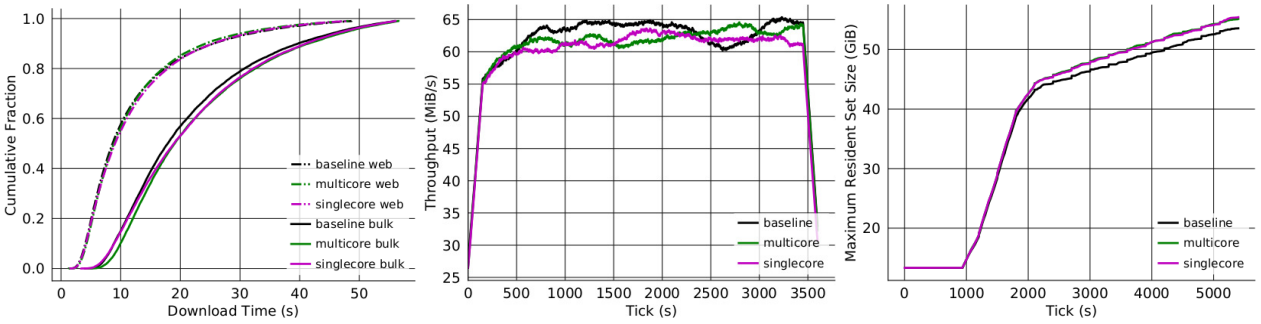
Finally, note that our design focuses on the conditions of the current Tor network, where the vast majority of traffic exits the network and congestion occurs primarily at exits. Although it is not inherently incompatible with our scheme, we leave the prioritization of internal onion services, which do not pass through exits, for future work.

5 Experimental Validation

Understanding typical Tor usage and assessing it benefits from our priority scheme is a crucial requirement. Appendix A covers a real-world Tor measurement study that illustrates the importance of token exchange within the first few seconds of the data stream. Having established the empirical context for a channel payment scheme, we validated our technical design via experiments performed on a prototype software implementation within the native Tor codebase. The objective is to prove that we can deliver a qualitatively "significant" advantage to paid premium users while incurring minimal overhead costs for throughput, memory usage, and latency within a realistic network environment. Due to the pre-built payment channel setup and low payment verification cost, we determine that our scheme supports the majority of observed short-lived and bursty Tor circuits in a near-fair-exchange setting.

5.1 Prototype

A substantial contribution of our research is embedded within our implementation of the moneTor framework. The modifications, applied to Tor release version 0.3.2.10, cover approximately fifteen thousand lines of new code across Tor's core C software. We emphasize that the implementation is engineered solely for our experiments. Most notably, expensive cryptographic operations such as ZKPs and commitments were simulated



(a) Download Time Overhead - Web + Bulk **(b)** Throughput - Compared to baseline **(c)** Simulation Memory

Fig. 2. Global Overhead — Comparison of overhead in pure multicore and singlecore network. Figure 2a shows two sets of time CDF curves for each file size (2 MiB and 5 MiB), Figure 2b shoes the 5 minute moving average the simulation 10 consensus file '2018-02-03-00-00-00-consensus'.

using methods that account for Shadow's unique virtual time management [32]. We consider both scenarios in which the simulated Tor process is running on a multicore or singlecore processor. In the multicore case, the cryptographic operations are replaced by an "idle" command that allows the virtual node to complete other tasks in parallel. In the singlecore case, cryptographic operations are simulated by looping through a series of dummy SHA256 hash operations. Using these methods, duration of the delays were tuned to conservatively reflect real measurements published in prior background work [20].[6] Note that our prototype does not implement anything that does not help us to answer our research goals, such as coin/wallet management, extension of the Tor control protocol to manage addition asset and options, Intermediary information recovery in case of crash, etc. Instead, the prototype serves the following purposes:

1. Our implementation handles nuances missing from the theoretical protocol specification. We show that there are no unexpected or prohibitive practical conflicts with the existing Tor design.
2. Our platform allows us to study the feasibility of premium circuit prioritization from a networking perspective.
3. Our platform allows us to obtain a rough factor-of-two approximation for all bandwidth, computation, and memory requirements of a real deployment, both globally and at individual nodes.

The first design purpose is clearly qualitative and we did not discover any insurmountable logical flaws in the design. To analyze the networking dynamics and resource consumption, we studied our implementations through a set of experiments described in the next section.

5.2 Methodology

Experiments were conducted using the Tor shadow simulator tool [32, 33]. We ran two sets of experiments at different scales from a consensus document published in early February 2018. The first set featured 100 relays, 1000 clients, 10 intermediaries, and ran for a total of 90 minutes. These experiments were used to gather information concerning the system overhead and protocol execution times. The second set featured 250 relays, 2500 clients, 25 intermediaries, 80 minutes of total run time, and was used to measure the performance benefits conferred to premium clients. In both cases, simulated traffic consists of 8% *bulk* clients who continuously download 5 MiB files and 92% *web* clients who periodically download 2 MiB files.[7] The number and behavior of clients were chosen to satisfy (A) realistic congestion rates measured by a transfer timeout percentage of approximately 4% [2] and a historical bulk/web global traffic ratio of about 1:3 [35, 36]. Neither the scale of our experiments nor the precise configuration of client nodes are intended to be precise replicas of real-world conditions. Tor networking is itself a complex area of re-

6 Extracted values are conservative in the sense that our zero-knowledge proofs require proving only a subset of the statements required in each corresponding Bolt zero-knowledge proof.

7 While 5 MiB bulk files are a common standard in Tor benchmarking [2], 2 MiB web files reflect the approximate size of modern web pages [34].

search and, for our purposes, we are content to adopt the simplest model that will highlight the relatively crude networking needs of our incentivization scheme.

5.3 Experiments

Our experiments are separated into three groups: global overhead, payment latency, and network priority. Each captures a separate characteristic of the scheme.

Global Overhead. First, we attempt to show the total cost of the moneTor scheme in terms of total network throughput. To study worst-case performance, we configured a medium-scale experiment consisting of 100% premium clients which we compared to a baseline trial with 0% premium clients. The purpose of this experiment was to measure overhead imposed by the payment scheme *without* applying any network prioritization in either control-flow or EWMA. Since our protocol can benefit from concurrently executed cryptographic operations, a key parameter to the simulation is the number of CPU cores available on each relay. Unfortunately, this information is not publicly available. As a result, we conducted two trials: one in which all nodes are running on multi-core hardware and one in which all nodes are running on single-core hardware. Figure 2 summarizes the results.

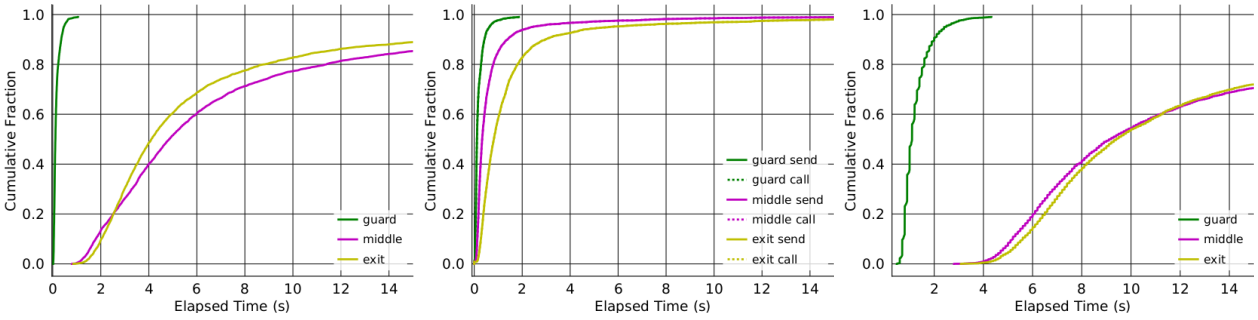
Our findings indicate that even in the worst case scenario, our system incurs statistically negligible overhead at these scales across the measures of download time (e.g., less than 2% increase on the mean web download for the singlecore experiment), throughput, and memory usage. When examining the raw network messages, we found corroborating evidence that moneTor contributes to only a small fraction, less than 1%, of the total network traffic in our experiment, a result which holds true across all of our trials. By default, we configured a payment rate of one payment cell for every 1000 data cells exchanged in either direction. If the network requires more fairness, it is also possible to increase the payment rate with negligible CPU cost as long as the network overhead introduced by the control cells remains under an acceptable fraction of the overall bandwidth.

Payment Latency. Given the results from our experiments, we surmise that payment latency is a crucial factor in servicing front-loaded clients. To this end, we measure the distribution of completion times for various steps in the protocol. To highlight the effects of native latency in the Tor network, we show payments split across each relay role of guard, middle, and exit. Recall that moneTor makes use of high-overhead, low-marginal cost payment channels (i.e., the channels take time to build but the client needs them long after they are built). In other words, the bulk of the cost in our scheme lies in the execution of **Nano-Establish** and **Nano-Close** protocols as shown in Figure 3a and Figure 3c.

Notice that nano-close operations take roughly twice as long to complete as the nano-establish operations due to the need for the relay to close his half of the nanopayment channel before the client can complete hers. Figure 3b illustrates the time to first payment, our most revealing latency metric. This measure includes the overhead in channel establishment when we do not have available preemptive channels. In the best case scenario, when all three payment channels have been correctly pre-built for the circuit, this measure is equivalent to a single trip toward each relay. Comparing this Figure 3b to Figure 3a, we observe the effectiveness of preemptive channel building. The other observation supporting the effectiveness of the pre-built strategy is the recorded time for the "call" versus "send" lines; if no discrepancy is observed between them, it means that the pre-build successfully led to fully established channels.

In all protocol phases, we observe that latencies for guard relays are negligible in comparison to the middle and exit relays, which is a result of our design decision to implement directly-paid guard channels. Again, this is a Tor-specific optimization made possible by the fact that guards maintain a semi-persistent, transparent relationship with only a small subset of clients.

Network Priority. Our final set of experiments studies the success of our scheme in delivering prioritized traffic for premium users. To perform this analysis, we prepared sets of three small experiments with varying modifier priorities: $\alpha \in \{0, 0.25, 0.5\}$ and $\beta \in \{1, 5, 10\}$. From a networking perspective, $\alpha = 0, \beta = 1$ is equivalent to unmodified Tor. We set the fraction premium users to be 25%. From Figures 4a, 4b, and 4c, we observe that, first, variations in our network-wide tunable parameters do offer differentiation in download speed. Yet, as we detail in Appendix D, offering bandwidth differentiation for the Tor network is more complex than previously assumed. Indeed, local scheduling priority, which was historically effective for past Tor topologies, appears to be ineffective under current conditions where congestion is concentrated at the exit interface. Second, the differentiation in bandwidth for $\alpha = .25, \beta = 5$ "averages out" to approximately mirror the baseline experiment, indicating little loss in overall network per-



(a) Nano-Establish - Built after the circuit construction, but before the circuit usage!

(b) First Payment - Should match 1 half RTT if the preemptive Nano-Establish is perfect

(c) Nano-Close - Happens just before the circuit is destroyed

Fig. 3. Protocol Execution Time — Time to finish each protocol step split across interactions with each of the three relays. The simulation includes 100 relays, 2 authorities, 1 ledger authority, 10 intermediaries and 1000 Tor clients scaled down from the public consensus file '2018-02-03-00-00-00-consensus'.

formance, and confirming our overhead experiment (recall 25% of premium users). Nevertheless, our result for $\alpha = 0.5, \beta = 10$ indicates that the performance degrades faster for nonpremium users when we become too aggressive in procuring gains for premium users. The data also highlights the complexity in selecting a set of parameters and techniques to offer efficient prioritization without degrading the overall throughput of the network. The overarching takeaway is that the network prioritization mechanism appears to be an even more complex challenge than the design of the anonymous payment layer itself.

Note that our analysis of the scheme holds the total network capacity static. However, the motivation for any Tor incentivization scheme is to attract new relays to grow the network which would, in principle, improve anonymity and censorship resistance for all users. The effect on performance is less clear. Although adding new relays will increase the throughput, a faster Tor would likely attract more users as well. In the absence of a reliable economic model, it is unclear how incentives would affect the experience of the average user, and so we opted to forgo modeling the added capacity.

6 Related Work

This section categorizes previously proposed incentivization schemes into three groups.

Non-transferable benefits. These schemes aim to recruit relays by offering some privileged status intended for personal use which cannot be *securely* sold for reimbursement of financial investment [37–39].

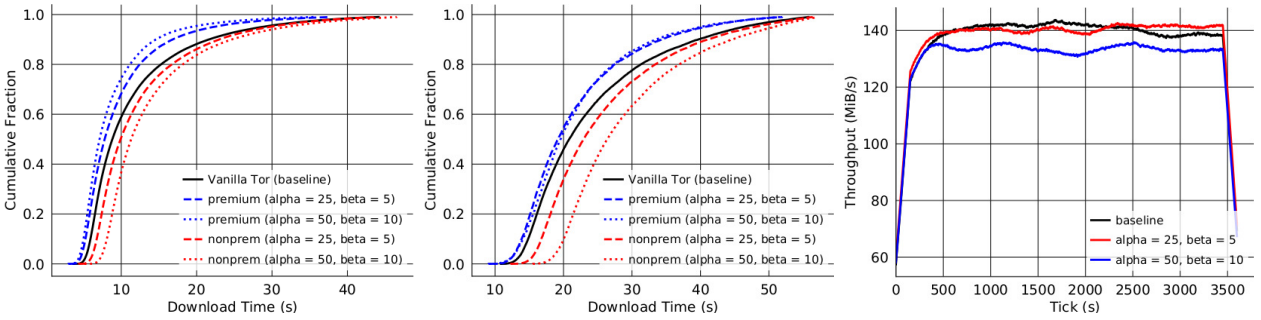
Transferable benefits. These schemes offer privileged service or products intended to hold value on a secondary resale market. These indirect financial incentives presume to attract a broader demand than in the non-transferable case. Partial proof-of-work tokens that can then be redeemed by relays for profit in real cryptocurrency mining pools [40] and exchangeable shallots in TEARS [41] are part of this category.

Monetary payments. These schemes offer rewards with what might be considered real money that holds external value. This category would include moneTor. PAR [42] allows clients to send direct payments to relays in a hybrid payment scheme which makes use of inefficient but anonymous Chaumian e-cash protocols [43] and efficient but transparent probabilistic micropayments. PAR introduces the *honest but curious* bank paradigm in which the bank cannot deanonymize clients but is in control of their deposited financial assets. PAR suffers from scalability issues owing to its strongly centralized architecture. Other monetary payments schemes such as XPay [44] and the proposal of Carbunar *et al.* [45], which combine an e-cash base with bipartite hashchains, are limited by the same centralized banking requirements.

In general, most prior works in this field, including BRAIDS, LIRA, and each of the cited monetary payment protocols, suffer from two issues:

1. Scalability limits arising from the need for all nodes to connect to a single central bank
2. The existence of a trusted banking authority which can opaquely manipulate the money supply.

Our scheme mitigates 1) by making use of payment channel networks, eliminating the need for nodes to con-



(a) Web Download Time **(b)** Bulk Download Time **(c)** Total throughput, all clients

Fig. 4. Prioritization Benefit — Performance differentiation between paid and unpaid users. We display results for 25% premium users. Simulations feature 250 relays, 2 authorities, 1 ledger authority, 25 intermediaries and 2500 Tor clients scaled down from the public consensus file '2018-02-03-00-00-00-consensus'.

nect to the central ledger for every relay interaction. The second issue is intrinsic to the standard Chaumian e-cash paradigm adopted by these prior works, in which "deposit" operations take the form an unblinded token signed by the trusted bank [43]. Even if all interactions were committed to a tamper-proof ledger, the network at large has no way to verify that signed tokens correspond to a valid "withdrawal" operation by another user. In other words, a malicious bank can mint fraudulent tokens at will. Our system eliminates this concern by replacing the active *honest but curious* bank with the passive public ledger model ubiquitous to modern cryptocurrencies. Even in the most centralized configuration described by Option 1 of Section 3.5, the Tor Project can only perform minting operations which are transparently verifiable on the ledger.

These improvements adequately capture comparisons between moneTor and other monetary payment schemes. However, trade-offs with state-of-the-art solutions in the first two categories are more nuanced. We address several specific works next.

BRAIDS. The BRAIDS scheme introduces *tickets* to represent premium status. Users may transfer tickets to other users, but this transfer must be done through a trusted-third party. Small numbers of ephemeral tickets are freely distributed by a central *bank* to any client upon request or to relays that have accumulated tickets spent by clients. Crucially, tickets can only be spent at a single relay defined at the time of their minting, in order to circumvent the double spending problem [38]. However, users may exchange accumulated tickets for tickets from other relays through a central bank within a set time interval. In addition to the general concerns listed earlier, BRAIDS raises two more:

3. Verifying a blind signature on the relay side for each payment is computationally intensive if the rate of payment is high, and current high-bandwidth Tor relays are already CPU-bound.
4. Tickets are relay-specific, implying that users must frequently exchange them for the right relays.

Consequently, users have two costly options as the network increases: stockpile a large number of tickets for each relay or interact frequently with the central bank to exchange tickets.

LIRA. LIRA is an ideological successor to BRAIDS which improves scalability as well as the efficiency on the relay-side verification of tickets by a factor ≈ 80. Clients in LIRA probabilistically "win" premium tickets without any interaction with the bank. While LIRA improves the efficiency of BRAIDS, and could even offer high fairness (by supporting high payment rates), it incentivizes *client* cheating by continuously building circuits to try to win premium tickets [39, 46]. Depending on the chosen value for the payment rate, this problem alone could prevent the scheme from being a realistic option. If the payment rate is too low, then the incentive to cheat increases since the scheme awards a large priority bandwidth between guesses to the cheater. If the payment rate is too high, then guessers would have difficulty maintaining good guesses through their circuit lifetime, since the probability of maintaining winning guesses exponentially decreases with payment rate. As a result, LIRA's goal of increasing buyer anonymity through successful guessers loses its efficacy. Moreover, LIRA does not address Concerns 2) and 4) inherited from BRAIDS.

In contrast to BRAIDS and LIRA, moneTor possesses the ability to prevent double-spending without suffering from a centralized exchange process

(BRAIDS) [46], or from incentives to cheat to gain premium access and stockpile relay-specific information (LIRA). Moreover, moneTor offers high fairness with on-the-fly payment verification that presents an improvement factor of ≈ 6 compared to LIRA (one H operation as compared to six H + a few XORs and multiplications) and ≈ 500 to BRAIDS. This procedure does not account for opening and closing the nanopayment channel, which happens outside of the data transfer time window. There, the relay's costly operation is to perform one ZKP (opening) and generate one blind signature (closing). Finally, and more importantly, moneTor tokens are not relay-specific, which solves the scalability problem of either stockpiling or interacting more with the central bank as the network grows.

TEARS. TEARS introduces a two-layer approach whereby *shallot* tokens are awarded by a distributed semi-trusted bank to participating relays. These shallots can then be redeemed for BRAIDS-style *Priority Passes*. While fully exchangeable shallots are an improvement over non-transferable privileges in a narrow economic sense, these tokens are conceptually discrete, indivisible assets that are not as easily exchanged as true currency. We expect our ecosystem to be more user friendly than TEARS, offering arbitrarily high transferability and divisibility of priority tokens without changing the underlying Tor architecture. In contrast to TEARS, which requires a blind signature for each payment, our granular nanopayment transactions approximate fair-exchange, a critical property for the low-bandwidth and short-lived premium Tor circuits documented in Appendix A. Finally, a major difference with previous work (TEARS, LIRA, BRAIDS) is that moneTor does not depend on a relay's bandwidth audit to distribute tokens. Relays receive some revenue directly from each client and the rest from the Tor Project's tax redistribution. Although the tax redistribution does require bandwidth audits, the incentive to game the system is only proportional to the tunable tax rate.

We cannot ignore the fact that direct payments introduce a separate mode of abuse. In principle, exit relays in moneTor have an incentive to inflate network traffic by injecting junk traffic (e.g., padding cells) or to conspire with the destination server to send useless data to the client. Monetor can mitigate this risk by implementing junk traffic monitoring into the existing measurement infrastructure. This can be done by running premium and non-premium circuits through measured relays. Junk traffic produced by the exit node is already a significant issue [13] that has prompted the release of several patches by the Tor project. [8] However, the conspirator problem seems intractable since the junk data appears legitimate to the Tor circuit layer. As a coarse-grain mitigation strategy, we can cap the number of nanopayments available to each circuit.

7 Discussion

In this discussion, we frame the social nuances of incentivization that accompany moneTor. Today, the Tor network owes its success to voluntary contributors. These relay operators incur hardware, bandwidth, and labor expenses to participate in the Tor network often for intrinsic reasons, among them: political, philosophical, and philanthropic. In implementing a system like moneTor, the Tor Project must consider the consequences of financial incentives on these potentially fragile value systems. Critics have called attention to the social consequences of incentives [46]. For instance, the "crowding out effect", describes a psychological phenomenon whereby the introduction of extrinsic motivations displaces previously dependable intrinsic motivations [47]. At the sociological level, empirical studies of prosocial behaviour [48] have shown that explicit incentives can reduce participation. This means that in our application, they risk degrading the average social quality of Tor nodes without necessarily growing the network. None of this is to say that extrinsic incentives are always ineffective, [48], only that choosing the right solution is nontrivial. We do not presume to offer an authoritative opinion on the best social incentive design. Instead, we show that the versatility of our token-centric technical design is sufficient to support a wide range of potential strategies.

By design, we have left many variables to be decided by the Tor Project. Chief among these are the premium bandwidth price, premium bandwidth advantage, tax rate, monetary policy, and redistribution policy. Enforcement mechanisms include a combination of network monitoring, parameter broadcast, ledger constraints, and organizational policy. Together, these options give the Tor Project the freedom to implement a

8 Independently, there may be a need for Tor to centrally monitor exit nodes' behavior, so abusive exit nodes can be detected and removed from the network altogether.

large variety of incentivization strategies. We present a few theoretical paradigms below.

1. **Benevolent Ruler**: The Tor Project is a nonprofit with altruistic organizational intentions. In this hypothetical approach, the authorities set the tax rate to 100%, turning moneTor into a revenue stream for purely centralized efforts to improve the network.
2. **Crypto-Libertarian**: On the other end of the spectrum, a simple naïve strategy is to implement moneTor as unregulated market where relays sell bandwidth for money, perhaps trustlessly backed by a cryptocurrency like Bitcoin or Ethereum.
3. **In-Kind Rewards**: If financial rewards turn out to be prohibitive, moneTor can mimic several of the purely in-kind non-transferable and transferable schemes described in Section 6 by limiting the number of on-ledger transactions to zero or one, respectively. For instance, at zero allowed transactions, relays would be unable to use their tokens for any purpose other than to buy premium traffic from other relays.
4. **Subsistence Relaying**: Here, although relays would receive real money, the expected value rewards would be limited to the break-even cost of running a relay. In this paradigm, every relay operator would effectively act as an individual NGO to avoid the potential liabilities that come with financial profit. In practice, moneTor could support this design by adjusting global parameters such that the income of any relay does not exceed the cost-to-bandwidth ratio in the cheapest geographic region. The tax redistribution policy could cover any discrepancy from this income floor due to location or other factors. Previously, studies have indicated that incentives can be useful when sparingly applied to "concrete" task [47, 49]. Provided careful public communication of intent, the same results would be expected here.
5. **Decentralized Grant-making**: Finally, the Tor Project can implement moneTor as a form of *altruistic money*, which payees can donate to an approved list of prosocial projects, reminiscent of participatory grant-making and participatory budgeting [50]. Rather than cashing out tokens, the ledger would only allow relays to contribute to their choice of special programs that, for instance, increase the number of relays in region X, fund research on topic Y, or advocate for privacy-related issue Z. The result is a fully fungible asset that is extrinsically worthless to profit-driven entities but intrinsically rewarding for parties invested in Tor's core mission.

Much future work remains to implement some of these methods. On the technical side, robust monitoring must be implemented to mitigate against inflated bandwidth for tax redistribution and junk traffic insertion for direct payments. Economically, we require a better understanding of location-based incentives and the elasticity of supply and demand for various types of relay incentives and client premium bandwidth.

The purpose of these paradigms is to illustrate the versatility moneTor's technical infrastructure provides. In practice, the Tor Project can implement nearly any combination of our listed approaches, and many others as well. The mindset we wish to instill is this: an overly flexible design can always be constrained afterward. Until there is a consensus for the optimal social approach to incentivization, the most useful technical base is one that can be adapted to many models of human behavior.

8 Conclusion

This work introduces moneTor, a practical system that uses tokenized payments to incentivize participation in the Tor network. Like any network, Tor can benefit from improving its performance and diversifying its participants. While Tor incentivization has been studied for almost 10 years, existing implementations have many technical limitations.

Our work directly addresses these shortcomings. At the payment level, we leveraged recent advances from the cryptocurrency research space to inform the design of our payment system. At the network level, we investigated a critical issue in the existing scheduling mechanism and introduce a new flow-control-based approach for network prioritization. The resulting scheme features sufficiently anonymous transactions in the absence of a trusted third party and considerably greater scalability. We tested moneTor through extensive simulation efforts to demonstrate its technical feasibility.

An essential set of questions remain concerning the legal, political, and social aspects of incentives for Tor. We certainly do not dismiss the importance of these issues, which existing literature have discussed at length [46]. Nevertheless, the ability to show what *can* be done, in terms of its technical impact on networking and security, is a central step toward determining what *should* be done.

Acknowledgments

We would like to thanks the anonymous reviewers for their helpful guidance on the socio-economical questions raised by our work and for the profoundly positive impact which their thorough reviews had on our work. We thanks Rob Jansen for providing us helpful comments on issues with the initial results we obtained with Shadow. We thank the Tor project ethical board for their guideline towards safer live experimentations. This work has been funded in part by the Walloon Region (competitiveness pole Logistics in Wallonia) through the project Digitrans (convention number 7618). Part of this work was performed while the third author was a Fulbright Scholar at Rice University.

References

[1] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second generation onion router," in *Usenix Security*, 2004.

[2] "Tor Performance Metrics," metrics.torproject.org, accessed: 2018.

[3] M. K. Wright, M. Adler, B. N. Levine, and C. Shields, "The predecessor attack: An analysis of a threat to anonymous communications systems," *ACM Transactions on Information and System Security (TISSEC)*, vol. 7, no. 4, pp. 489–522, 2004.

[4] S. J. Murdoch and G. Danezis, "Low-cost traffic analysis of Tor," in *Security and Privacy, 2005 IEEE Symposium on*. IEEE, 2005, pp. 183–195.

[5] M. AlSabah and I. Goldberg, "Performance and security improvements for Tor: A survey," *ACM Computing Surveys (CSUR)*, vol. 49, no. 2, p. 32, 2016.

[6] J. Reardon and I. Goldberg, "Improving Tor using a TCP-over-DTLS tunnel," in *Proceedings of the 18th conference on USENIX security symposium*. USENIX Association, 2009, pp. 119–134.

[7] R. Jansen, J. Geddes, C. Wacek, M. Sherr, and P. F. Syverson, "Never Been KIST: Tor's Congestion Management Blossoms with Kernel-Informed Socket Transport," in *USENIX Security Symposium*, 2014, pp. 127–142.

[8] T. Crump *et al.*, *The Phenomenon of Money (Routledge Revivals)*. Routledge, 2011.

[9] C. Dovrolis and P. Ramanathan, "A case for relative differentiated services and the proportional differentiation model," *IEEE network*, vol. 13, no. 5, pp. 26–34, 1999.

[10] C. Tang and I. Goldberg, "An improved algorithm for Tor circuit scheduling," in *Proceedings of the 17th ACM conference on Computer and communications security*. ACM, 2010, pp. 329–339.

[11] "Account holding monetor code," https://github.com/monetor, 2018.

[12] X. Fu, Z. Ling, J. Luo, W. Yu, W. Jia, and W. Zhao, "One cell is enough to break Tor's anonymity," in *Proceedings of Black Hat Technical Security Conference*, 2009, pp. 578–589.

[13] F. Rochet and O. Pereira, "Dropping on the Edge: Flexibility and Traffic Confirmation in Onion Routing Protocols," *Proceedings on Privacy Enhancing Technologies*, vol. 2018, no. 2, pp. 27–46, 2018.

[14] M. AlSabah, K. Bauer, I. Goldberg, D. Grunwald, D. McCoy, S. Savage, and G. Voelker, "DefenestraTor: Throwing out Windows in Tor," in *Proceedings of the 11th Privacy Enhancing Technologies Symposium (PETS 2011)*, July 2011.

[15] F. Tschorsch and B. Scheuermann, "Mind the gap: Towards a backpressure-based transport protocol for the Tor network," in *Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI '16)*, Santa Clara, CA, Mar. 2016.

[16] "Blockchain info," https://blockchain.info/stats, 2018, accessed: April 2018.

[17] J. Poon and T. Dryja, "The Bitcoin lightning network: Scalable off-chain instant payments," *draft version 0.5*, vol. 9, p. 14, 2016.

[18] E. Heilman, L. Alshenibr, F. Baldimtsi, A. Scafuro, and S. Goldberg, "Tumblebit: An untrusted Bitcoin-compatible anonymous payment hub," in *Network and Distributed System Security Symposium*, 2017.

[19] G. Malavolta, P. Moreno-Sanchez, A. Kate, M. Maffei, and S. Ravi, "Concurrency and privacy with payment-channel networks," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 455–471.

[20] M. Green and I. Miers, "Bolt: Anonymous payment channels for decentralized currencies," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 473–489.

[21] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *URL: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.221.9986&rep=rep1&type=pdf*, 2008.

[22] S. A. Crosby and D. S. Wallach, "Efficient Data Structures For Tamper-Evident Logging," in *USENIX Security Symposium*, 2009, pp. 317–334.

[23] G. Danezis and S. Meiklejohn, "Centrally banked cryptocurrencies," *Proceedings on Privacy Enhancing Technologies*, 2016.

[24] R. L. Rivest and A. Shamir, "PayWord and MicroMint: Two simple micropayment schemes," in *International workshop on security protocols*. Springer, 1996, pp. 69–87.

[25] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from Bitcoin," in *Security and Privacy (SP), 2014 IEEE Symposium on*. IEEE, 2014, pp. 459–474.

[26] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timon, and P. Wuille, "Enabling Blockchain Innovations with Pegged Sidechains," *URL: https://blockstream.com/sidechains.pdf*, 2014.

[27] J. Poon and V. Buterin, "Plasma: Scalable Autonomous Smart Contracts," *White paper*, 2017.

[28] F. Rochet and O. Pereira, "Waterfilling: Balancing the Tor network with maximum diversity," *Proceedings on Privacy Enhancing Technologies*, vol. 2017, no. 2, April 2017.

[29] A. Johnson, R. Jansen, A. D. Jaggard, J. Feigenbaum, and P. Syverson, "Avoiding the man on the wire: Improving Tor's security with trust-aware path selection," in *Proceedings of the Network and Distributed Security Symposium - NDSS '17*. Internet Society, February 2017.

[30] C. Kiraly, R. Dingledine, G. Bianchi, R. Lo Cigno, and A. M. Scattolo, "Effect of Tor window size on performance," http://archives.seul.org/or/dev/Feb-2009/msg00000.html, 2009.

[31] C. Kiraly, G. Bianchi, and R. Lo Cigno, "Solving performance issues in anonymization overlays with a l3 approach," *URL: http://disi.unitn.it/locigno/preprints/TR-DISI-08-041.pdf*, 2008.

[32] R. Jansen and N. Hopper, "Shadow: Running Tor in a Box for Accurate and Efficient Experimentation," in *Proceedings of the Network and Distributed System Security Symposium - NDSS'12*. Internet Society, February 2012.

[33] J. Tracey, R. Jansen, and I. Goldberg, "High performance Tor experimentation from the magic of dynamic ELFs," in *11th USENIX Workshop on Cyber Security Experimentation and Test (CSET 18)*, 2018.

[34] "HTTP Archive," https://httparchive.org/, 2018.

[35] R. Jansen and A. Johnson, "Safely Measuring Tor," in *Proceedings of the 23rd ACM Conference on Computer and Communications Security (CCS '16)*, October 2016.

[36] R. Jansen, M. Traudt, and N. Hopper, "Privacy-preserving Dynamic Learning of Tor Network Traffic," in *Proceedings of the 24th ACM Conference on Computer and Communications Security (CCS '18)*, November 2018.

[37] R. Dingledine, D. S. Wallach *et al.*, "Building incentives into Tor," in *International Conference on Financial Cryptography and Data Security*. Springer, 2010, pp. 238–256.

[38] R. Jansen, N. Hopper, and Y. Kim, "Recruiting new Tor relays with BRAIDS," in *Proceedings of the 17th ACM conference on Computer and communications security*. ACM, 2010, pp. 319–328.

[39] R. Jansen, A. Johnson, and P. Syverson, "LIRA: Lightweight incentivized routing for anonymity," in *Proceedings of the 20th Network and Distributed System Security Symposium.*, 2013.

[40] A. Biryukov and I. Pustogarov, "Proof-of-work as anonymous micropayment: Rewarding a Tor relay," in *International Conference on Financial Cryptography and Data Security*. Springer, 2015, pp. 445–455.

[41] R. Jansen, A. Miller, P. Syverson, and B. Ford, "From onions to shallots: Rewarding Tor relays with TEARS," in *7th Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs 2014)*, 2014.

[42] E. Androulaki, M. Raykova, S. Srivatsan, A. Stavrou, and S. M. Bellovin, "PAR: Payment for anonymous routing," in *International Symposium on Privacy Enhancing Technologies Symposium*. Springer, 2008, pp. 219–236.

[43] D. Chaum, A. Fiat, and M. Naor, "Untraceable electronic cash," in *Conference on the Theory and Application of Cryptography*. Springer, 1988, pp. 319–327.

[44] Y. Chen, R. Sion, and B. Carbunar, "XPay: Practical anonymous payments for Tor routing and other networked services," in *Proceedings of the 8th ACM workshop on Privacy in the electronic society*. ACM, 2009, pp. 41–50.

[45] B. Carbunar, Y. Chen, and R. Sion, "Tipping pennies? privately practical anonymous micropayments," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 5, pp. 1628–1637, 2012.

[46] R. Jansen, "Tor incentives research roundup: Goldstar, PAR, BRAIDS, LIRA, TEARS, and TorCoin," https://blog.torproject.org/tor-incentives-research-roundup-goldstar-par-braids-lira-tears-and-torcoin, 2014.

[47] U. Gneezy, S. Meier, and P. Rey-Biel, "When and Why Incentives (Don't) Work to Modify Behavior," *Journal of Economic Perspectives*, vol. 25, no. 4, pp. 191–210, December 2011.

[48] R. Bénabou and J. Tirole, "Incentives and Prosocial Behavior," *American Economic Review*, vol. 96, no. 5, pp. 1652–1678, December 2006.

[49] J. R. Behrman, P. Sengupta, and P. Todd, "Progressing through progresa: An impact assessment of a school subsidy experiment in rural mexico," *Economic Development and Cultural Change*, vol. 54, no. 1, pp. 237–275, 2005. [Online]. Available: http://www.jstor.org/stable/10.1086/431263

[50] T.-N. Dinh, "Universal basic philanthropy: A scalable model to democratize social impact," https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3503969, 2020.

[51] L. Daigle, "WHOIS protocol specification," *URL: https://tools.ietf.org/html/rfc3912*, 2004.

[52] S. Williamson and M. Kosters, "Referral whois protocol (rwhois)," *URL: https://tools.ietf.org/html/rfc3912*, 1994.

[53] "Tor research safety board," https://research.torproject.org/safetyboard.html, 2018, members: https://research.torproject.org/safetyboard.html#who.

[54] M. Green and I. Miers, "Bolt: Anonymous payment channels for decentralized currencies," https://eprint.iacr.org/2016/701, 2016.

[55] R. Jansen, M. Traudt, J. Geddes, C. Wacek, M. Sherr, and P. Syverson, "KIST: Kernel-Informed Socket Transport for Tor," *ACM Transactions on Privacy and Security (TOPS)*, vol. 22, no. 1, p. 3, 2018.

[56] R. Dingledine and S. J. Murdoch, "Performance improvements on Tor or, why Tor is slow and what we're going to do about it," *Online: http://www. torproject. org/press/presskit/2009-03-11-performance. pdf*, 2009.

[57] R. Jansen, "BRAIDS github repository," https://github.com/robgjansen/braids-tor-simulator, 2010.

[58] R. Dingledine and N. Mathewson, "Tor directory specifications," https://gitweb.torproject.org/torspec.git/tree/dir-spec.txt, accessed: February 2019.

[59] F. Rochet and O. Pereira, "Waterfilling proposal," https://github.com/frochet/wf_proposal, 2017.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. The views expressed in this article do not necessarily represent the views of Sandia National Laboratories, NTESS, the U.S. Department of Energy or the United States Government.

A Empirical Analysis

A.1 Data Collection

For over two weeks, we deployed a data collection system to observe empirical temporal information about lifetime and bandwidth consumption in Tor circuits. Our objective was to have a deeper understanding of typical Tor usage and whether such usage could benefit from our channel-based payment system. For example, these measurements might capture the distribution of type and magnitude of potential premium traffic. We classify the traffic type based on the service connection port. In addition to the standard ports 80 and 443 used for web traffic, we aggregated data from some other families, including the WHOIS protocol [51] and RWHOIS [52] from ports 43 and 4321, respectively. By specifying a reduced exit policy, we can identify traffic patterns according to known types.

Efforts to preserve users privacy

To ensure ethical experimentation, we contacted the Tor research safety board [53] and used their feedback to inform our data collection process. We collected data from five old and stable exit relays with a total bandwidth of 50 MiB/s, stripped the origin metadata, aggregated it in memory, and stored it on a central server into bins of configurable size for each traffic type. The data only covers circuits that are "active" from the perspective of the exit relay, in other words, circuits that have received a connection request to an IP address on the internet. Once we collected enough data (1600 circuits) from a single type, we saved the information on the disk, cleared the relay memory, and resumed a new session. Crucially, we did not record information linked to any single particular user flow on disk. The final result contained only aggregated data with following two pieces of information for each pool:

- **Time Profile**: The total number of inbound and outbound cells in each 5-second time interval. Tracking begins after the conclusion of a successful DNS request.
- **Total Counts**: The total amount of cells processed by a circuit. We aggregate this information by taking the mean of fixed-size nearest neighbor bins.

E.g. avoid writing metadata from any specific flow or circuit to disk.

Observations

Our measurements successfully captured several essential pieces of information for the design and justification of moneTor. For example, one crucial task is to determine the number of potential users that could benefit from paid traffic. From Figure 5a, we observe that $\approx 82\%$ of circuits carrying only web traffic exchanged less than 1000 cells. While we cannot deduce any statements about users, we can speak to the fraction of circuits that may benefit from a payment channel in the Tor network, since around 50% of them do not carry data and less than 17% of them carry at least one web page. The remaining 18% would appear to be better candidates for moneTor.

It is also evident from Figure 5b that most of the traffic is usually carried within the first few tens of seconds regardless of traffic type. From that result, we believe that the reliability of payment is critical within the first few seconds, especially from a relay perspective. This result highlights our choice to extend Bolt to offer high fairness. Furthermore, the front-loaded distribution curve indicates a need to establish available payment channels as soon as the user opens a circuit. As a result, we base our design upon a preemptive circuit build strategy, which effectively eliminates channel setup/establish latency in the average case.

B Algorithms

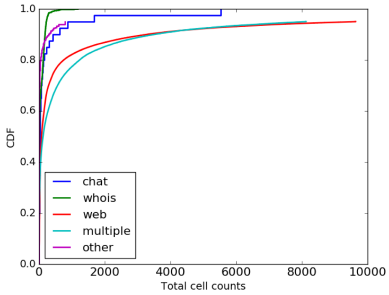
B.1 Conventions

We adopt the following conventions in our algorithms.

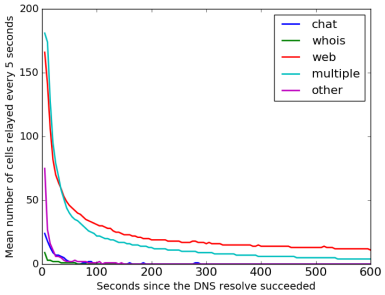
- Variable subscripts denote a party or role ((I)ntermediary, (C)lient, (R)elay, (E)nd user).
- New nanopayment variables have the character (n) as a prefix. All other variables reference a value from the original Bolt scheme, although the name might be altered somewhat.
- Payment values (ϵ, δ) are signed integers with respect to the end-user. For example, δ_C is negative and δ_R is positive when a client is paying a relay.

B.2 Variable Index

This section describes the symbols we use to define our algorithms. In the following list, the left-hand symbol names a "high-level" bundle of logically related values that the nodes keep in persistent storage to maintain



(a) Total Counts — Distribution of circuit size with respect to the total number of cells processed



(b) Time Profile — Average distribution of traffic across circuit lifetime beginning with the first DNS request.

Fig. 5. Data collection from 5 old and stable exit relays with a cumulative bandwidth of ≈50 MiB/s

control of their assets. All other symbols name "low-level" values, which serve as inputs to cryptographic and accounting operations.

$nT = (\delta_C, \delta_R, n, hc^0)$ — Nanopayment Channel Token — Stores static, public information that defines a nanopayment channel including the payment values on both legs, the maximum number of payments, and the hashchain head. This can be passed around publicly.

$ncsk_C = (nwpk_C, nwsk_C, HC)$ — Client Nanopayment Secrets — Includes a public/private key pair which allows the client to setup and close a nanopayment channel. It also includes a pre-computed hash chain to make incremental nanopayments

$nS_C = (k, hc^k)$ — Client Nanopayment State — Stores the mutable, public state of the nanopayment. Includes the current number of completed nanopayments and the latest sent hash preimage

nrt_C — Client Nanopayment Refund — Allows the client to make a claim to the ledger at any time for the escrowed money. This refund is signed by the intermediary and conditioned on revealing the latest hash preimage that the client claims to have sent.

nrc_C — Client Channel Closure Message — This is the final message that the client posts to the ledger to claim all funds of the micropayment channel. It includes any completed nanopayments.

$nS_I = \{nT : nanopayment\_state\}$ — Intermediary Nanopayment State — Map of all past and present nanopayment channels and the corresponding channel state. Possible states are: $\perp$ (failed attempt to setup a nanopayment channel), *setup* (channel has been set up by C), *established* (channel has been established with R), or $closed||hc^k$ (channel has been closed and no further payments are allowed).

$ncsk_R = (nwpk_C, nwsk_C, \perp)$ — Relay Nanopayment Secrets — Includes a public/private key pair allowing the relay to setup and close a nanopayment channel. Similar to $ncsk_C$ except the last field is blank, since relays cannot make payments.

$nS_R = (k, hc^k)$ — Relay Nanopayment State — See nS_C

nrt_R — Relay Nanopayment Refund — See nrt_C

nrc_C — Relay Channel Closure Message — See nrc_C

$S_I = \{[wpk_E|nwpk_E] : micropayment_state\}$ — Intermediary Micropayment State — Map of all past and present micropayment channels.

B.3 Algorithms

Algorithm 1 Create Wallet Helper function for creating a new wallet

function WAL($pp, pk_{payee}, w, \epsilon$)
  parse w as $(B, wpk, wsk, r, \sigma^w)$
  $(wsk', wpk') \leftarrow$KeyGen(pp)
  $r' \leftarrow$Random()
  $wCom' \leftarrow$Commit$(wpk', B + \epsilon, r')$
  $\pi \leftarrow PK\{(wpk', B, r', \sigma^w)$:
    $wCom' =$ Commit$(wpk', B + \epsilon, r') \wedge$
    Verify$(pk_{payee}, (wpk, B), \sigma^w) = 1 \wedge$
    $B + \epsilon \geq 0\}$
  return $(wsk', wpk', wCom', \pi)$

Algorithm 2 Nano-Setup Protocol between a client and intermediary to create a new nanopayment channel from an existing micropayment wallet. Run prior to circuit setup.

procedure CLIENT($pp, pk_I, w_C, \delta_C, n$)
  parse w_C as $(B_C, wpk_C, wsk_C, r_C, \sigma_C^w)$
  if $B_C + (\delta_C \times n) < 0$ **then**
    Abort() ▷ consider new micropayment channel
  $\epsilon_C \leftarrow \delta_C \times n$
  $(nwpk_C, nwsk_C, nwCom_C, n\pi_C) \leftarrow$Wal($pp, pk_I, w_C, \epsilon_C$)
  $\delta_R \leftarrow -(\delta_C + tax)$ ▷ the tax is a net profit for I
  $HC \leftarrow$MakeHC(Random(), n)
  $nT \leftarrow (\delta_C, \delta_R, n, HC[0])$
  Intermediary.Send($wpk_C, nwpk_C, nwCom_C, n\pi_C, nT$)
procedure INTERMEDIARY(pp, sk_I, S_I, nS_I)
  $(wpk_C, nwpk_C, nwCom_C, n\pi_C, nT) \leftarrow$Client.Receive()
  parse nT as $(\delta_C, \delta_R, n, hc^0)$
  if $wpk_C \in S_I \vee nwpk_c \in S_I \vee \neg$Verify($n\pi_C$) **then**
    Abort() ▷ invalid wallets
  if $-\delta_C \neq price \vee \delta_R + \delta_C + tax \neq 0$ **then**
    Abort() ▷ incorrect payment prices
  $S_I \leftarrow S_I \cup \{wpk_C : \perp, nwpk_C : \perp\}$
  $nS_I \leftarrow nS_I \cup \{nT : \perp\}$
  $nrt_C \leftarrow$Sign($sk_I, refund||nT||nwCom_C$)
  Client.Send(nrt_C)
procedure CLIENT
  $nrt_C \leftarrow$Intermediary.Receive()
  if $\neg$Verify($pk_I, refund||nT||nwCom_C, nrt_C$) $= 1$ **then**
    Abort() ▷ invalid refund token
  $nS_C \leftarrow (0, HC[0])$
  $ncsk_C \leftarrow (nwpk_C, nwsk_C, HC)$
  $\sigma_C^{revc(w)} \leftarrow$Sign($wsk_C, revoke||wpk_C$)
  Intermediary.Send($\sigma_C^{recv(w)}$)
procedure INTERMEDIARY
  $\sigma_C^{recv(w)} \leftarrow$Client.Receive()
  if $\neg$Verify($wpk, revoke||wpk_C, \sigma_C^{recv(w)}$) $= 1$ **then**
    Abort() ▷ invalid revocation token
  $S_I[wpk_C] \leftarrow \sigma_C^{recv(w)}$
  $nS_I[nT] \leftarrow setup$
  Client.Send($established$)

Algorithm 3 Nano-Establish Protocol between a client, intermediary, and relay to establish the nanopayment channel between the client and relay. Run at the start of circuit setup.

procedure CLIENT(nT)
  Relay.Send(nT)
procedure RELAY($pp, pk_I, B_{I:B}, w_R$)
  $nT \leftarrow$Client.Receive()
  parse w_R as $(B_R, wpk_R, wsk_R, r_R, \sigma_R^w)$
  parse nT as $(\delta_C, \delta_R, n, hc^0)$
  if $B_{I:B} - (\delta_B \times n) < 0$ **then**
    Abort() ▷ consider new micropayment channel
  $\epsilon_R \leftarrow \delta_R \times n$
  $(nwpk_R, nwsk_R, nwCom_R, n\pi_R) \leftarrow$Wal($pp, pk_I, w_R, \epsilon_R$)
  Intermediary.Send($wpk_R, nwpk_R, nwCom_R, n\pi_R, nT$)
procedure INTERMEDIARY(pp, sk_I, S_I, nS_I)
  $(wpk_R, nwpk_R, nwCom_R, n\pi_R, nT) \leftarrow$Relay.Receive()
  parse nT as $(\delta_C, \delta_R, n, hc^0)$
  if $wpk_R \in S_I \vee nwpk_R \in S_I \vee \neg$Verify($n\pi_R$) **then**
    Abort() ▷ invalid wallets
  if $nS_I[nT] \neq setup$ **then**
    Abort (unregistered nanopayment channel)
  $S_I \leftarrow S_I \cup \{nwpk_R : \perp\}$
  $nS_I[nT] \leftarrow established$
  $nrt_R \leftarrow$Sign($sk_I, refund||nT||nwCom_R$)
  Relay.Send(nrt_R)
procedure RELAY
  $nrt_R \leftarrow$Intermediary.Receive()
  if $\neg$Verify($pk_I, refund||nT||nwCom_R, nrt_R$) $= 1$ **then**
    Abort() ▷ invalid refund token
  $ncsk_R \leftarrow (nwpk_R, nwsk_R, \perp)$ ▷ match client format
  $nS_R \leftarrow (0, hc^0)$

Algorithm 4 Nano-Pay Protocol between the client and relay to forward a single nanopayment. Run periodically throughout the lifetime of the circuit.

procedure CLIENT($nT, ncsk_C, nS_C$)
  parse nT as $(\delta_C, \delta_R, n, hc^0)$
  parse $ncsk_C$ as $(nwpk_C, nwsk_C, HC)$
  parse nS_C as (k, hc^k)
  if $k >= n$ **then**
    Abort() ▷ out of nanopayments, setup a new channel
  $nS_C \leftarrow (k+1, HC[k+1])$
  Relay.Send($HC[k+1]$)
procedure RELAY(nT, nS_R)
  $hc^{k+1} \leftarrow$Client.Receive()
  parse nS_R as (k, hs^k)
  if $k+1 >= n \vee Hash(hc^{k+1}) \neq hc^k$ **then**
    Abort() ▷ invalid nanopayment
  $nS_R \leftarrow (hs^{k+1}, k+1)$

Algorithm 5 Nano-Close Protocol between an end-user and an intermediary to close a channel.

$\forall E \in \{Client, Relay\}$
procedure EndUser($pp, pk_I, w_E, nT, ncsk_E, nS_E$)
  parse w_E as $(B_E, wpk_E, wsk_E, r, \sigma_E^w)$
  parse nT as $(\delta_C, \delta_R, n, hc^0)$
  parse $ncsk_E$ as $(nwpk_E, nwsk_E, \_)$
  parse nS_E as (k, hc^k)
  $\epsilon_E \leftarrow \delta_C \times k$ if (EndUser = Client) else $\delta_R \times k$
  $(wpk'_E, wsk'_E, wCom'_E, \pi'_E) \leftarrow Wal(pp, pk_I, wpk_B, \sigma_E^w, B_E, \epsilon_E)$
  Intermediary.Send$(wpk_E, wCom'_E, \pi'_E, nT, \epsilon_E, k, hc^k)$
procedure Intermediary(pp, sk_I, S_I, nS_I)
  $(wpk_E, wCom'_E, \pi_E, nT, \epsilon_E, k, hc^k) \leftarrow$EndUser.Receive()
  parse nT as $(\delta_C, \delta_R, n, hc^0)$
  if $\epsilon_E < 0 \wedge closed \notin nS_I[nT]$ **then**
    Abort() ▷ client attempting to close before relay
  if $\neg$Verify$(\pi_E) \vee nS_I[nT] \neq established$ **then**
    Abort() ▷ invalid wallet or channel
  if $k > n \vee \neg$VerifyHC(hc^0, k, hc^k) **then**
    Abort() ▷ invalid payment hash chain
  $nS_I[nT] \leftarrow closed||hc^k$
  $rt'_E \leftarrow$Sign$(sk_I, refund||wCom'_E)$
  EndUser.Send(rt'_E)
procedure EndUser
  $rt'_E \leftarrow$Intermediary.Receive()
  if $\neg$Verify$(pk_I, refund||wCom'_E, rt'_E) = 1$ **then**
    Abort() ▷ invalid refund token
  parse $ncsk_E$ as $(nwpk_E, nwsk_E, \perp)$
  $\sigma_E^{rev(nrt)} \leftarrow$Sign$(nwsk_E, revoke||nwpk_E)$
  Intermediary.Send$(nwpk_E, \sigma^{rev(nrt)})$
procedure Intermediary
  $(nwpk_E, \sigma_E^{rev(nrt)}) \leftarrow$EndUser.Receive()
  if $nwpk_E \in S_I \vee \neg$Verify$(nwpk_E, \sigma^{rev(nrt)})$ **then**
    Abort() ▷ unregistered channel or revocation token
  $S_I[nwpk_E] \leftarrow \sigma^{rev(nrt)}$
  EndUser.Send$(verified)$
procedure EndUser
  $ver \leftarrow$Intermediary.Receive()
  $w'_E \leftarrow$Intermediary.Blindsig$(ver, wpk'_E||B_E + \epsilon_E)$

Algorithm 6 Nano-Refund Algorithm by an end-user to close a micropayment channel and claim ledger funds.

$\forall E \in \{Client, Relay\}$
function EndUser($pp, csk_E, w_E, nT, ncsk_E, nS_E, nrt_E$)
  parse csk_E as $(\_, sk_E, \_, \_, \_, \_)$
  parse w_E as $(B_E, \_, \_, \_, \_)$
  parse nT as $(\delta_C, \delta_B, \_, \_)$
  parse $ncsk_E$ as $(nwpk_E, \_, \_)$
  parse nS_E as (k, hc^k)
  $\delta_E \leftarrow \delta_C$ if (EndUser = Client) else δ_R
  $m_E \leftarrow (refund||nT||nwpk_E||B_E + \delta_E \times n, nrt_E, hc_E^k, k_E)$
  $nrc_E \leftarrow (m_E, Sign(sk_E, m_E))$
  return nrc_E

Algorithm 7 Nano-Refute Algorithm by an intermediary to respond to an end-user's refund claim by posting its own channel closure message to the ledger

$\forall E \in \{Client, Relay\}$
function Intermediary($pp, T_E, S_I, nS_I, nrc_E$)
  parse nrc_E as (m_E, σ_E^m)
  parse m_E as $(refund||nT||nwpk_E||B_E^{full}, nrt_E, k_E, hc_E^k)$
  ▷ $B_E^{full} \leftarrow$ balance if nanopayment channel were saturated
  parse T_E as $(pk_E, \_)$
  if $\neg$Verify(pk_E, m_E, σ_E^m) **then**
    Abort() ▷ bad signature, well be rejeced by ledger
  if $\neg$Verify$(pk_I, (refund||nT||nwpk_E||B_E^{full}), nrt_E)$ **then**
    Abort() ▷ unapproved refund token
  if $S_I[nwpk_E] \neq \perp$ **then**
    ▷ E is posting an old token, I should refute
    $\sigma_E^{rev(nrt)} \leftarrow S_I[nwpk_E]$
    $nrc_I \leftarrow ((revoked, \sigma_E^{rev(nrt)}), Sign((revoked, \sigma^{rev(nrt)})))$
  ▷ Everything checks out; accept the closure
  $hc^k \leftarrow nS_I[nT]$
  $nrc_I \leftarrow ((accepted, k_I, hc_I^k), Sign(accepted, k_I, hc_E^k))$
  return nrc_I

Algorithm 8 Nano-Resolve Algorithm run by the ledger (and everyone verifying the ledger) to resolve all channel closure messages and allocate the appropriate final balances

▷ Returns the tuple $(B_E^{final}, B_I^{final})$
function LEDGER($pp, T_E, T_I, nrc_E, nrc_I$)
  $B^{total} = B_E^{init} + B_I^{init}$
  parse nrc_E as (m_E, σ_E^m)
  parse nrc_I as (m_I, σ_I^m)
  parse m_E as $(refund||nT||nwpk_E||B_E^{full}, nrt_E, k_E, hc_E^k)$
  ▷ $B_E^{full} \leftarrow$ balance if nanopayment channel were saturated
  parse nT as $(\delta_C, \delta_R, n, hc^0)$
  $\delta_E \leftarrow \delta_C$ if (EndUser = Client) else δ_R
  if $nrc_E = \perp$ **then**
    ▷ E failed to respond closure request in time
    return $(0, B_{total})$
  if $\neg$Verify$(pk_E, m_E, \sigma_E^m) \vee \neg$Verify$(pk_I, m_I, \sigma_I^m)$ **then**
    return $\perp$ ▷ messages could not be authenticated
  if $\neg$Verify$(pk_I, refund||nT||nwpk_E||B_E^{full}, nrt_E)$ **then**
    return $(0, B_{total})$ ▷ E is attempting to use invalid token
  if $revoked \in m_I$ **then**
    parse m_I as $(revoked, \sigma_E^{rev(nrt)})$
    if Verify$(nwpk_E, \sigma_E^{rev(nrt)})$ **then**
      return $(0, B_{total})$ ▷ E is trying to use old channel
    else
      return $(B_{total}, 0)$ ▷ invalid revocation from I
  ▷ micropayments settled, now resolve nanopayments
  parse m_I as $(accepted, k_I, hc_I^0)$
  if $k_I \leq k_E \leq n \wedge$VerifyHC$(hc^0, k_E, hc_E^k))$ **then**
    ▷ E has the highest hash preimage
    $B_E^{final} = B_E^{full} - \delta_E \times (n - k_E)$
    $B_I^{final} = B_{total} - B_E^{full} + \delta_E \times (n - k_E)$
  if $k_E \leq k_I \leq n \wedge$VerifyHC$(hc^0, k_I, hc_I^k))$ **then**
    ▷ I has the highest hash preimage
    $B_E^{final} = B_E^{full} - \delta_E \times (n - k_I)$
    $B_I^{final} = B_{total} - B_E^{full} + \delta_E \times (n - k_I)$
  return $(B_E^{final}, B_I^{final})$

C Formal definitions and proofs

C.1 Definitions

Anonymity and balance for nanopayment channel
Our objective in this work is to provide an efficient, correct, and privacy-preserving payment system for Tor network bandwidth. Our nanopayment channel is built on the top of an existing micropayment channel as designed by Green and Miers [20]. Intuitively, we replace the Pay protocol of their bidirectional channel with our set of Nano-Setup, Nano-Establish, Nano-Pay and Nano-Close, protocols to allow high-granularity payments of up to n iterations at the cost of roughly two Pay protocols. We require that the intermediary does not learn more than the number of nanopayments realized between an unknown Tor client and an unknown relay. Moreover, we require that the nanopayment protocol always produce a correct outcome for each valid execution of the protocol. Informally, the anonymity guarantees provided by the nanopayment channel states that any relay (except the guard relay) of a circuit learns no information except that a valid nanopayment channel establishment, payment, or closure has occurred over an open micropayment channel with some intermediary. A particular relay should not be able to link any two nanopayment channels for separate circuits that it operates.

We reuse the payment anonymity and balance properties of Green and Miers [54] for an Anonymous Payment Channel (APC scheme), but we adapt them for our tripartite protocol. The scheme requires a privacy property that holds against the intermediary, a privacy property that holds against a relay, and a balance property to define monetary security. We prove that if there exists an adversary able to break the anonymity property, then this adversary can distinguish the Real experiment from the Ideal experiment of an APC scheme with a non-negligible advantage. Furthermore, we prove that the only adversary able to break the balance property is an adversary able to break preliminary security assumptions.

Due to the fact that moneTor nanopayment channels are inherently transparent, we do not require unlinkability between Nano-Setup/Nano-Establish and Nano-Close from the perspective of the relay and the intermediary.

C.1.1 Payment anonymity with respect to the intermediary:

Let $\mathcal{A}$ be an adversary playing the role of the intermediary. We consider an experiment involving P customers (a.k.a. Tor client) and Q relays, each interacting with the intermediary as follows. First, $\mathcal{A}$ is given pp, then outputs $T_{\mathcal{M}}$. Next $\mathcal{A}$ issues the following queries in any order:

Initialize channel for $\mathcal{C}_i$ and $\mathcal{R}_j$. When $\mathcal{A}$ makes this query on input B^{cust}, B^{inter}, it obtains the commitment $T^i_{\mathcal{C}}$ generated as

$$(T^i_{\mathcal{C}}, csk^i_{\mathcal{C}}) \leftarrow\!\!\$\ Init_{\mathcal{C}}(pp, B^{cust}, B^{inter})$$

where the customer might also be a relay. In this case, the intermediary obtains the commitment $T^j_{\mathcal{R}}$ generated as

$$(T^j_{\mathcal{R}}, csk^j_{\mathcal{R}}) \leftarrow\!\!\$\ Init_{\mathcal{R}}(pp, B^{relay}, B^{inter})$$

Establish channel with $\mathcal{C}_i$ and $\mathcal{R}_j$. In this query, $\mathcal{A}$ executes the Establish protocol with $\mathcal{C}_i$ (resp. $\mathcal{R}_j$) as

$$Establish(\{\mathcal{C}(pp, T_{\mathcal{M}}, csk^i_{\mathcal{C}})\}, \{\mathcal{A}(state)\})$$

Where $state$ is the adversary's state. We denote the customer's output as w_i, where w_i may be $\perp$.

Nano-Setup from $\mathcal{C}_i$. In this query, if $w_i \neq \perp$, then $\mathcal{A}$ executes the Nano-Setup to escrow ϵ with $\mathcal{C}_i$ as:

$$\text{Nano-Setup}(\{\mathcal{C}(pp, \epsilon, w^i_{\mathcal{C}})\}, \{\mathcal{A}(state)\})$$

Where $state$ is the adversary's state. We denote the customer's output as $w^i_{\mathcal{C}}$, the hashchain root hc^0, the customer's nanopayment secret $ncsk_{\mathcal{C}}$, the customer's state $nS_{\mathcal{C}}$ and the refund token $nrt_{\mathcal{C}}$, where any may be $\perp$.

Nano-Establish from $\mathcal{R}_j$. In this query, if $w^j_{\mathcal{R}}$ and $nT \neq \perp$, then $\mathcal{A}$ executes the Nano-Establish to register the nanopayment channel with the relay $\mathcal{R}_j$ as:

$$\text{Nano-Establish}(\{\mathcal{R}(pp, w^j_{\mathcal{R}}, nT)\}, \{\mathcal{A}(state)\})$$

Where $state$ is the adversary state. We denote the relay's output as $w^j_{\mathcal{R}}$, the refund token $nrt_{\mathcal{R}}$, the relay's nanopayment secret $ncsk_{\mathcal{R}}$ and the state of the relay's nanopayment channel $nS_{\mathcal{R}}$.

Nano-Close from $\mathcal{C}_i$ and $\mathcal{R}_j$. In this query, if $\epsilon^i_{\mathcal{C}}$, nT, $ncsk_{\mathcal{C}}$ and $nS_{\mathcal{C}} \neq \perp$, then $\mathcal{A}$ executes the Nano-Close to close the nanopayment channel and update the

micropayment wallet with $\mathcal{C}_i$ (resp. $\mathcal{R}_j$).

$$\text{Nano-Close}(\{\mathcal{C}(pp, \epsilon^i_{\mathcal{C}}, nT, ncsk_{\mathcal{C}}, nS_{\mathcal{C}})\}, \{\mathcal{A}(state)\}) \rightarrow w^i_{\mathcal{C}}$$

Where *state* is the adversary's state. We denote the customer's and relay's output as $w^i_{\mathcal{C}}$ (resp. $w^j_{\mathcal{R}}$), where it may be $\perp$.

Finalize with $\mathcal{C}_i$ (resp. $\mathcal{R}_j$). When $\mathcal{A}$ makes this query, it obtains $rc^i_{\mathcal{C}}$, computed as $rc_{\mathcal{C}} \leftarrow\!\!\$\ Refund(pp, w^i_{\mathcal{C}})$.

We say that $\mathcal{A}$ is *legal* if $\mathcal{A}$ never asks to spend from a wallet where $w^i_{\mathcal{C}}$ or $w^j_{\mathcal{R}}$ is $\perp$ or undefined, and where $\mathcal{A}$ never asks C_i to spend unless the customer has sufficient balance to complete the spend.

Let pp' be an auxiliary trapdoor not available to the participants of the real protocol. We require the existence of a simulator $\mathcal{S}^{X-Y(\cdot)}(pp, pp', \cdot)$ such that for all $T_{\mathcal{M}}$, no allowed adversary $\mathcal{A}$ can distinguish the following two experiments with non-negligible advantage:

Real experiment. In this experiment, all responses are computed as described in our Algorithms.

Ideal experiment. In this experiment, the micropayment operations are handled using the procedure above. However, for the nanopayment procedures, $\mathcal{A}$ does not interact with $\mathcal{C}_i$ and $\mathcal{R}_j$ but instead interacts with a simulator $\mathcal{S}^{X-Y(\cdot)}(pp, pp', \cdot)$.

C.1.2 Payment anonymity with respect to the relay.

Let $\mathcal{A}$ be an adversary playing the role of the relay. We consider an experiment involving P customers (a.k.a. Tor clients), each interacting with the relay as follows. First, $\mathcal{A}$ establishes a micropayment channel with the intermediary. Next, $\mathcal{A}$ issues the following queries in any order:

Nano-Establish from $\mathcal{C}_i$. In this query, nT may be $\perp$, then $\mathcal{A}$ executes only the part of Nano-Establish which interacts with $\mathcal{C}_i$:

$$\text{Nano-Establish}(\{\mathcal{C}(pp, nT)\}, \{\mathcal{A}(state)\})$$

Where *state* is the adversary state. We denote the customer's output nT, which may be $\perp$.

Nano-Pay from $\mathcal{C}_i$. In this query, $nT \neq \perp$ and p_k may be $\perp$, then $\mathcal{A}$ executes the Nano-Pay protocol for an amount δ with $\mathcal{C}_i$ as:

$$\text{Nano-Pay}(\{\mathcal{C}(pp, \delta, p_k)\}, \{\mathcal{A}(state)\})$$

Where *state* is the adversary's state and p_k is the preimage of the current hash stored in the adversary's state or $\perp$.

We say that $\mathcal{A}$ is *legal* if $\mathcal{A}$ never asks to spend more than $n \times \delta$.

Let pp' be an auxiliary trapdoor not available to the participants of the real protocol. We say that a payment scheme offers anonymity if, for every legal $\mathcal{A}$, there is a simulator $\mathcal{S}^{X-Y(\cdot)}(pp, pp', \cdot)$ such that the following two experiments cannot be distinguished with a non-negligible advantage:

Real experiment. In this experiment, all responses are computed as described in our Algorithms.

Ideal experiment. In this experiment, the micropayment operations and nanopayment operations with the intermediary are handled using our algorithms. However, for the nanopayment procedures between the Tor client and the adversary relay, $\mathcal{A}$ does not interact with $\mathcal{C}_i$ but instead interacts with $\mathcal{S}^{X-Y(\cdot)}(pp, pp', \cdot)$.

C.1.3 Balance

Let $\mathcal{A}$ be an adversary playing the role of the relay. We consider an experiment involving P honest Tor clients $\mathcal{C}_1, ..., \mathcal{C}_P$ interacting with the relay. We assume the micropayment channels are properly setup and established with the intermediary and that the intermediary continues to interact honestly with the client and relay.

Given the micropayment channel setup and established, parties hold funds valued at B^{cust} and $B^{\mathcal{A}}$. Let $bal_{\mathcal{A}} \leftarrow 0$ be the amount of funds the adversary may claim. Now $\mathcal{A}$ may issue the following queries in any order:

Nano-Establish from $\mathcal{C}_i$. In this query, nT may be $\perp$, then $\mathcal{A}$ executes only the part of Nano-Establish which interacts with $\mathcal{C}_i$:

$$\text{Nano-Establish}(\{\mathcal{C}(pp, nT)\}, \{\mathcal{A}(state)\})$$

Where *state* is the adversary state. The adversary obtains nT and establishes the nanopayment channel with the intermediary.

Nano-Pay from $\mathcal{C}_i$. The nanopayment channel has been correctly established before. This query can executed up to n times before **Nano-close** is called. For each execution, $nT \neq \perp$ and p_k may be $\perp$. $\mathcal{A}$ executes the Nano-Pay protocol for an amount δ with $\mathcal{C}_i$ as:

$$\text{Nano-Pay}(\{\mathcal{C}(pp, \delta, p_k)\}, \{\mathcal{A}(state)\}) \rightarrow p_k$$

If $H(p_k)$ matches the hash stored in the adversary's state, then $bal_{\mathcal{A}} = bal_{\mathcal{A}} + \delta$ and $H(p_k)$ is stored in the internal state. If it does not match, we output $\perp$.

Nano-Close with intermediary. In this query, $\epsilon_{\mathcal{A}} \leftarrow k \times \delta$ for k Nano-Pay executions. $nT, ncsk_{\mathcal{A}}, nS_{\mathcal{A}} \neq \perp$, then $\mathcal{A}$ executes the Nano-Close protocol to close its leg of the nanopayment channel and claim funds to the intermediary.

$$Nanoclose(\{\mathcal{A}(pp, \epsilon^i_{\mathcal{A}}, nT, ncsk_{\mathcal{A}}, nS_{\mathcal{A}})\},$$
$$\{Intermediary(state)\}) \rightarrow w^i_{\mathcal{A}}$$

We denote the adversary output $w^i_{\mathcal{A}}$, where it may be $\perp$. The Tor client also closes its leg of the nanopayment channel with the intermediary to transfer $k \times \delta$ and update its wallet accordingly. At any point, all parties have the option to call Nano-Refund to initiate a partial or full refund of their escrowed fund and close the nanopayment channel. We say that $\mathcal{A}$ is *legal* if it never agrees to execute the Nano-Pay protocol upon $nT = \perp$. We further restrict $\mathcal{A}$ to establish one nanopayment channel per micropayment channel established with any Tor client. We say that a scheme guarantees a correct balance if no $\mathcal{A}$ can complete, with non-neglible probability, the game described above in such a way that $bal_{\mathcal{A}} > k \times \delta$.

C.2 Theorem

The nanopayment channel scheme satisfies the properties of anonymity (C.1.1, C.1.2) and security (C.1.3) under the restriction that the adversary does not abort before Nano-Close finishes, the restrictions that at most one nanopayment channel can be open per micropayment channel, the assumptions that the commitment scheme is secure, the zero-knowledge system is simulation extractable and zero-knowledge, and the hash function used to create the hashchain and verify the preimage during the Nano-Pay is a cryptographic hash function.

C.3 Proofs

C.3.1 Anonymity

We prove that the nanopayment channel scheme satisfies our anonymity properties using a simulator $\mathcal{S}^{X-Y(\cdot)}(pp, pp', \cdot)$ such that no allowed adversary $\mathcal{A}$ can distinguish the Real experiment from the Ideal experiment with non-negligible advantage. The way this proof proceeds requires honest runs of the appropriate algorithms for the micropayment channel. When Nanopayment channel operations are called, the client side or relay side of the protocol is emulated by the simulator for the Ideal experiment. To prove that they are indistinguishable, we borrow Green and Miers's proof and extend it to our notion of payment anonymity to the intermediary and the relay. We start with the Real experiment, and we create Games that modify elements of the protocol until we match the Ideal experiment conducted with the simulator $\mathcal{S}$. When $\mathcal{A}$ calls the simulator $\mathcal{S}$ on legal interactions, the simulator emulates the Tor client or relay part of the protocol, depending on which step of the protocol we perform.

Let be ν_1, ν_2 be negligible functions and let **Adv[Game i]** be $\mathcal{A}$'s advantage in distinguishing the output of **Game i** from the Real Distribution.

Game 0. This is the Real experiment: Nano-Setup, Nano-Establish, and Nano-Close between customers (Tor clients) and the intermediary.

Game 1. This game is identical to **Game 0** except that we replace NIZK proofs generated by the customer at the Nano-Setup and Nano-Close with simulated proofs (we assume the existence of a ZK simulation algorithm which can extract a simulated proof). If the proof system is zero-knowledge, then **Adv[Game 1]** $\leq \nu_1$.

Game 2. This game is identical to **Game 1** except that we replace the commitments $nwCom_C$, $nwCom_R$, $wCom'_C$ and $wCom'_R$ by commitments on random messages. If the commitment scheme is computationally hiding, then **Adv[Game 2]** $-$ **Adv[Game 1]** $\leq \nu_2$.

Game 3. This game is identical to **Game 2** except that we replace the root of the hashchain $HC[0]$ with a value generated from Random(). Note that Random() was also used for the original value, therefore **Adv[Game 3]** $-$ **Adv[Game 2]** $= 0$.

Game 4. This game is identical to **Game 3** except that we replace $wpk_C, nwpk_C, wpk_R, nwpk_R$ with random keys using the KeyGen algorithm described for anonymous micropayment channels. Since the distribution is identical to the distribution of original values, **Adv[Game 4]** $-$ **Adv[Game 2]** $= 0$

We have started with the Real experiment and modified elements of the protocols from a series of Games to come up with a computationally indistinguishable experiment conducted by $\mathcal{S}$ from the Real experiment.

Since $\mathcal{A}$ cannot distinguish the real experiment from the Ideal experiment obtained in **Game 4.** with non-negligible advantage, the interaction between customers and intermediary is anonymous.

Now, we have to prove the indistinguishablility between the Real experiment and the Ideal experiment for the relay's payment anonymity property. We proceed with the same logic:

Game 0'. This is the Real experiment: Nano-Establish and Nano-Pay between Tor clients and relays.

Game 1'. This game is identical to **Game 0'** except that we replace the root of the hashchain $HC[0]$ with a value generated from Random() in the Nano-Establish interaction. Note that Random() was also used for the original value, therefore **Adv[Game 1']** − **Adv[Game 0']** $= 0$

Game 2'. This game is identical to **Game 1'** except that we replace the preimage p_k sent to the relay by a value generated from Random(). In the random oracle model, both the original value and the simulated one provide from the same distribution, hence **Adv[Game 2']** − **Adv[Game 1']** $= 0$

Since **Game 2'** is identical in the Ideal experiment, the interaction between Tor clients and relays is anonymous.

By showing that the interaction with the intermediary and the interaction with the relay through the nanopayment algorithms are anonymous, we conclude that our nanopayment channel is anonymous.

C.3.2 Balance

We prove that the Nanopayment channel guarantees correct balance if the micropayment channel is itself secure, the hash function behaves like a random oracle, and the signature scheme is EU-CMA secure (i.e. Existential Unforgeability under a Chosen Message Attack).

To win, $\mathcal{A}$ must claim more money than the agreed-upon price between an honest client and the adversary. The adversary must make this claim while running a protocol that is indistinguishable from the honest protocol. The Nano-Setup protocol borrows the same structure as the provably secure Pay protocol. Properties included the soundness of the zero-knowledge proof, the binding property of the commitment scheme, and the unforgeability of the signature scheme. At this step, the adversary cannot win against the Tor client and claim more than $k \times \delta$ where k is 0 since no Nano-Pay has been executed yet. For the following steps of the protocol, we proceed by showing that $\mathcal{A}$ cannot diverge from the protocol and claim more than $k \times \delta$ with our classical security assumptions. If $\mathcal{A}$ could succeed this game, it would mean that there exists an indistinguishable experiment from the Real experiment where $\mathcal{A}$ ends up with more than $k \times \delta$.

Game 0. This is the Real experiment.

Game 1. This game is identical to Game 0 except that we replace hc^0 in nT by a value chosen by $\mathcal{A}$ from a hashchain created by $\mathcal{A}$. From this hashchain, $\mathcal{A}$ creates nT'. If the intermediary is honest, the nanopayment cannot be established because nT' is unknown to the intermediary for this micropayment channel. If the intermediary is dishonest, then it can accept nT' but cannot prove, under the assumption that the signature scheme is unforgeable in the usual sense (EU-CMA secure), that the client holds a refund token with nT' instead of nT. Hence, **Adv[Game 1]** $\leq \nu_1$.

Game 2. This game is identical to Game 1 except that $\mathcal{A}$ tries in the Nano-Pay protocol to find herself a preimage to the stored hashchain, and claim more than δ. Assuming the hash function is a cryptographic hash function, then the adversary cannot find a preimage unless the Tor client sends it to issue a payment. Hence, **Adv[Game 2]** $\leq \nu_2$.

Finally, the Nano-Close protocol borrows the micropayment Pay protocol to update the micropayment wallet according to the number k of preimages the adversary received from the Tor client. The Pay protocol has been proved secure by Green and Miers; hence we observe that the adversary cannot win the game with a non-negligible probability (claim more than $k \times \delta$).

D Scheduling insights

D.1 background: scheduling

Tor handles multiple queues of cells for each circuit and writes cells in the outbound connection while favoring "bursty" traffic, involving many smaller files, over "bulky" traffic, involving a few large files. The network's goal is to prioritize circuits handling interactive data streams, like chats or web browsing. Tor uses a heuristic called EWMA [10] (i.e., computes the exponentially weighted moving average for the number of cells sent on each circuit) to decide which circuit to prioritize. Recently, the Kist [7] scheduler improved the efficiency of EWMA by reducing the congestion on the kernel out-

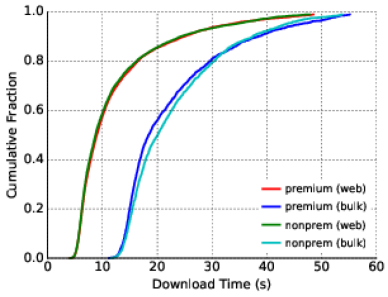


Fig. 6. Prioritized Scheduling — CDF download times for superimposed web and bulk clients where premium status is enforced only via scheduling. Almost no priority is observed.

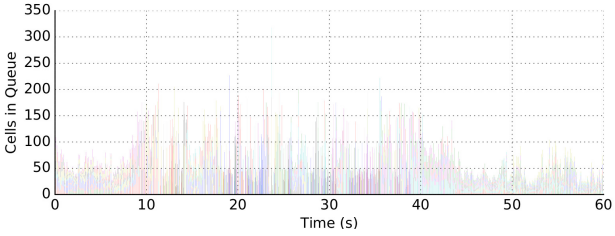


Fig. 7. Queue Temporal Profile (60 seconds) — Size of the scheduling buffer over time at a single exit relay in terms of number of cells. Colors group cells belonging to the same circuit.

bound queue and pushing back this delicate problem onto the Tor layer.

In an ideal network, we might expect that traffic movement is an exclusive function of the raw bandwidth capacity in each edge connection and the scheduling algorithm implemented at each node. The Tor network employs EWMA to favor interactive web clients over continuous bulk clients.

D.2 Obstacles

In moneTor, we originally attempted to modify EWMA with a simple linear scaling factor that would favor paid circuits. However, obtaining positive results from the modified algorithm has proven to be a complex challenge. Upon failing to achieve meaningful differentiation with low values of β, we adopted a more blunt policy which *always* services premium circuits first and implemented it in a zero-overhead version of moneTor. The results are displayed in Figure 6. Although we observed some moderate differentiation, the difference falls well short of the benefit needed to incentivize paid users. The same observation holds even under heavy levels of induced congestion.

This result is a serious issue since all previous works use strategies that make local scheduling decisions on each relay serve premium bandwidth. Our experiments indicate that offering bandwidth priority based on local decisions would be ineffective.

D.3 Investigation and discussion

Our negative results indicate that scheduling is not the most decisive factor in determining performance. To verify this hypothesis, we studied the incoming queue from which the scheduler can select new active circuits. Figure 7 illustrates the temporal load in the queue at a single exit relay over one minute. The height of the curve represents the total number of cells waiting to be serviced at each continuous point in time, while the colors group quantities of cells that belong to the same circuit. Figure 8 displays a subset of the same information within a smaller time interval.

In the figures mentioned earlier, notice that the queue is only populated for a period of 10 ms before it is completely flushed, implying the queue spends the vast majority of its time empty. This 10 ms window is a product of Tor's internal event handling framework and is consistent with data from Jansen et al [55]. We found in an analysis of the line-by-line observations of the queue activity that while cells get flushed in the correct order, they appear in the queue at roughly equal proportions. In effect, bandwidth in our simulation is not constrained by the ordering policy of the scheduler but rather by the rate at which they arrive from the network. As a consequence, local decisions at a particular relay for scheduling fall short of offering the expected priority. Note that Jansen *et al.* [55] discovered the same problem, which motivated the deployment of KIST. Our results show that despite the use of KIST, relays may still be able to flush all queues at once, eliminating the effect of the scheduler's choice.

We now proceed to investigate why local relay scheduling does not work as expected. Why do we fail to reproduce the positive results from previous works such

This ideal version of moneTor strips away all payment operations and instead passes a single signal through the network to distinguish premium circuits.

While the graph has the visual appearance of a bar graph, this is just a function of the striking data pattern. In actuality, the plot displays a stacked area graph.

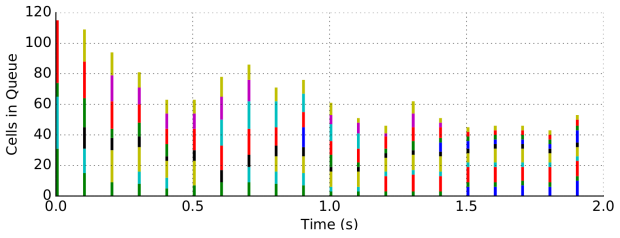


Fig. 8. Queue Temporal Profile (2 seconds) — Size of the scheduling buffer over time at a single exit relay in terms of number of cells. Colors group cells belonging to the same circuit.

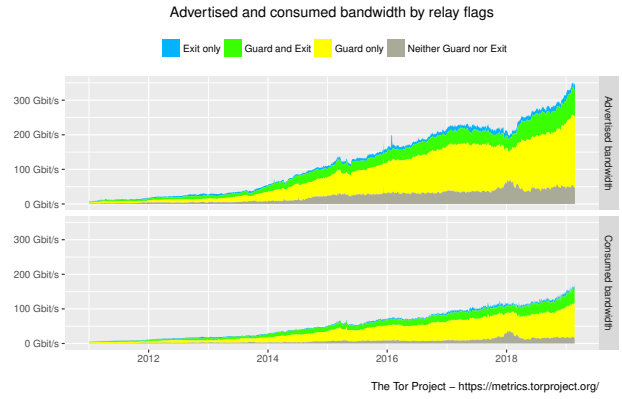


Fig. 9. Evolution of bandwidth aggregated by relay flags.

as BRAIDS and LIRA despite using the same methodology? One possible explanation is that other network control mechanisms within the Tor codebase constrain the flow of cells, rendering the mostly idle scheduler to be ineffective. Such mechanisms include point-to-point flow control, connection throttling, or some minimally-documented threshold embedded in the code. However, we do not believe that this fully explains our results, since the Tor code implementing these components has not meaningfully changed over the years.

Our positive results in Section 5.3, which show that the performance increases when we increase the circuit windows, points to a more promising direction. This finding contradicts previous works exploring the effect of window sizes [30, 31, 56]. To explore this lead, we found and verified the following explanation: the constraining network bottleneck has moved from the Tor network itself to the exit relay interface with external servers on the web. In this scenario, cell queuing within Tor is not nearly as important as the TCP/IP packet handling at each exit relay. Both approaches to prioritize flows are complementary: when the congestion is inside the Tor network, applying local scheduling policies makes an efficient priority mechanism, as demonstrated by previous works. Also, in such a situation, priority based on the flow-control (that is, a function of the global circuit) would not be efficient because all cells would spend the majority of the time waiting in relays' FIFO queues. Conversely, if the congestion is outside of the Tor network (between the exit relays and the destination), then local scheduling policies would fall short of making any prioritization as shown in Figure 6. Decisions based on the flow-control would achieve prioritized flows in this case, as demonstrated in Section 5.3.

The shift of congestion from the internal Tor network to the exit gateways explains why our scheduling results in Figure 6 are different from BRAIDS and LIRA. Indeed, BRAIDS ran experiments with Tor version `0.2.0.35` and a network history from January 2010 [57]. This version of Tor was released before Mike Perry implemented a major change in the path selection mechanism that reduced Tor's internal congestion problem and greatly improved the performance. The bandwidth-weights are a set of weights specified in the directory specifications, section 3.8.4 [58], that aim at balancing the overall network usage. Those weights are critical for network performance and also for anonymity [28, 59]. Importantly, the benefit of Mike Perry's bandwidth-weights is proportional to the inequalities between the overall bandwidth in-circuit positions, which has grown rapidly over the years, as shown in Figure 9.

LIRA's experiments ran on `tor-0.2.3.13-alpha` from March 2012, which benefits from Mike Perry's bandwidth-weights. This timeline may explain why LIRA has a less impressive priority advantage compared to one exposed in BRAIDS. Experiments in LIRA are probably less internally congested than the ones from BRAIDS, a change that appears to be caused by the significant change in the path selection algorithm. However, the different congestion status may also be due to other factors such as a different client usage model and the local scheduling policies. LIRA used a simulated environment scaled down from an April 2012 consensus where $\approx 42\%$ of relays had an `Exit` flag. Meanwhile, only $\approx 13\%$ of relays in our experiment had an `Exit` flag. Figure 10 plots the distribution of bandwidth allocated to each position and illustrates the different state of the Tor network, which in turn is reflected in the Shadow Simulations. More recent consensus data, which we use in moneTor's experiments, indicate network conditions where exit bandwidth is much scarcer [28].

Commit `0ff86042ac16` in September 2010.

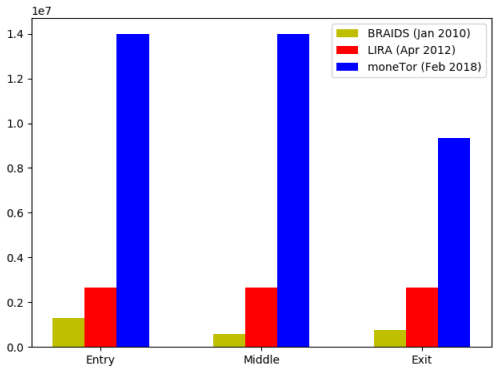


Fig. 10. Bandwidth distribution in consensuses used for BRAIDS, LIRA and moneTor experiments - BRAIDS does not benefit from bandwidth-weights to refill Middle, LIRA benefits from bandwidth-weights to offer balance between all positions, moneTor benefits from bandwidth-weights to balance entry and middle, yet does not have enough exit nodes to achieve the full balance

In summary, as the network grows to offer greater internal bandwidth (Figure 9, Figure 10), LIRA's and BRAIDS's schedulers tend to become less efficient for providing priority, as demonstrated in our experiments and shown in Figure 8. We should emphasize that this observation describes a coarse-grained trend. Since our experiments ran on a scaled-down topology with simplistic models for user behavior, the results do not necessarily describe the state affairs for all relays in the real Tor network. What can be said is that networking as a whole is an immensely complex and unpredictable domain and that attaining a simulation environment conducive to effective scheduling is, at the very least, nontrivial. Any serious deployment of an incentivization scheme would require further research into robust prioritization mechanisms. We leave this task for future work.