# PARoLE: Profitable Arbitrage in Optimistic Rollup with ERC-721 Token Transactions

Alvi Ataur Khalil* and Mohammad Ashiqur Rahman*†
*Electrical and Computer Engineering, Florida International University, USA
†Knight Foundation School of Computing and Information Sciences, Florida International University, USA
akhal042@fiu.edu, marahman@fiu.edu

***Abstract*—Optimistic rollup has emerged as a promising Layer 2 (L2) scaling solution for blockchain; however, its existing protocols are vulnerable to front/back-running activities, where an opportunistic rollup operator can strategically alter the transactions' order to create an arbitrage opportunity. Specifically, in the limited edition ERC-721 standardized non-fungible tokens (NFTs), the re-ordering of transactions introduces a lucrative threat landscape due to its scarcity-driven pricing and market volatility. In this work, we introduce PARoLE, a novel attack technique on optimistic rollup systems, where an adversarial aggregator re-orders the NFT transactions in an optimal way, leveraging model-free deep reinforcement learning (DRL) to maximize the balance of a target account. We create our own NFT called the "PARoLE Token" (PT) and deploy it in the OpenSea marketplace via Optimism Goerli to validate the attack impact. Furthermore, we collect NFT snapshots from rollup mainchains to analyze the impact in real-world NFT marketplaces.**

***Index Terms*—Blockchain, optimistic rollups, profitable arbitrage, mempool, non-fungible tokens**

## I. Introduction

Cryptocurrencies and blockchain technology have revolutionized the financial industry, with remarkable efficiencies and applications from 2009 to the present day [1], [2]. However, mass adoption remains unattainable since no existing blockchain can efficiently manage global-scale operations [3], [4]. In the rapidly evolving landscape of blockchain technologies, optimistic rollup has emerged as a pioneering innovation, which presents a Layer 2 (L2) scaling solution. Optimistic rollup shifts a significant portion of transaction processing and smart contract execution off the main blockchain, similar to the state-of-the-art L2 scaling solutions, e.g., side chains [5], plasma [6], payment channel networks [7], [8], etc. This L2 solution optimizes efficiency by batching transactions, reducing on-chain operations, and minimizing transaction fees. While it enhances the overall throughput, the confirmation time also decreases, making blockchain applications more cost-effective for users. This approach maintains a robust level of security by leveraging the security model of the underlying main chain [9].

Although perceived as a solution for quicker transaction processing, rollup has centralization concerns related to the entity called "sequencer," which is responsible for transaction ordering [10]. This centralization grants the sequencer significant power, enabling it to potentially censor transactions and exploit maximal extractable value (MEV), impacting users financially. Moreover, the reliance on a central sequencer poses a systemic risk — if it fails, the entire L2 rollup system can collapse. Aggregators in the L2 rollup ecosystem can potentially address some of the concerns associated with a single sequencer, who collect and bundle transactions from multiple sources (ideally from Bedrock's [11] Mempool) before submitting them to the Ethereum main chain [12]. This approach aims to introduce a degree of decentralization and mitigate the concentration.

However, this distributed solution is not immune to vulnerabilities, particularly within its transaction processing protocols. A significant risk is the potential for front-running and back-running activities, allowing opportunistic aggregators to manipulate transaction orders for profitable arbitrage opportunities tailored to specific users. While aggregators are expected to process transactions based on their base and priority fees [13], they have the flexibility to arrange them differently if sufficient incentives are present. We exploit this vulnerability and investigate the kinds of transactions that frequently provide arbitrage opportunities.

Arbitrage malpractice within the decentralized ecosystem is a well-established phenomenon. Mclaughlin et al. identified 3.8 million arbitrages in the Ethereum decentralized exchange (DEXes) ecosystem, generating $321 million in profit [14]. Zhou et al. introduced an SMT-based automatic profit generation tool for decentralized finance (DeFi) ecosystem [15], which had an estimated average weekly revenue of 72.44 ETH, demonstrating its effectiveness in arbitrage and complex settings. Wang et al. presented a theoretical framework for analyzing cyclic arbitrages in DEXes, which revealed that traders executed 292,606 cyclic arbitrages, exploiting over $138 million in revenue over 11 months [16].

While ERC-20 tokens have been extensively studied for predatory trading practices, the realm of ERC-721, known as the non-fungible tokens (NFTs), lacks sufficient attention and research despite its high-value assets. A study [17] revealed that as high as 3.93% of addresses, processing 2.04% of sale transactions, trigger suspicions of market abuse through illicit trading patterns of NFTs. For instance, an individual generated $100,000 through speculative trading of an NFT and approximately $8,000 through the operation of an arbitrage bot [18]. In another instance, a sniper bot executed a flashloan attack by front-running a bid transaction on an NFT, securing 26.25 ETH [19]. Along with this trend of negligence to unethical behaviors, the NFT market is susceptible to volatile price spikes [20], due to its uniqueness and scarcity. Specifically, in the case of the limited edition NFTs, whose value grows

2158-3927/24/$31.00 ©2024 IEEE
DOI 10.1109/DSN58291.2024.00026

proportional to its digital scarcity [21], the arbitrage opportunity through front/back runnings becomes more prominent.

Exploiting the vulnerability of the optimistic rollup systems and potential arbitrage scope within limited edition NFT markets, we propose **PARoLE** (**P**rofitable **A**rbitrage in Optimistic **Rol**lup with **E**RC-721 token transactions) attack, where an adversarial aggregator colludes with a user who participates in NFT transactions. The attacker aggregator checks for arbitrage opportunities with the set of transactions he receives from Bedrock's Mempool and re-orders the sequence in a way that maximizes the colluding user's balance. However, given the nature of the limited edition NFT market and the non-linear patterns of their pricing, traditional deterministic trading algorithms will fail, both to capture the economic dynamics and produce a profitable order. On the other hand, deep reinforcement learning (DRL) is well-suited for solving problems involving decision-making, and sequential actions, where an agent needs to learn optimal strategies through trial and error [22]. Due to DRL's capacity to adapt to dynamic environments, handle complex decision-making, and learn from experience, it has become a very effective tool for solving non-linear optimization problems. Thus, we utilize DRL within the attack framework to optimize the re-ordering task and maximize the profit for the intended colluding user.

Our primary contributions in this work are fourfold:

- We identify an existing vulnerability of an optimistic rollup system and propose a novel attack technique with limited edition NFT transactions.
- We introduce the GenTranSeq module, designed to determine the most effective transaction order to optimize the attack advantages by leveraging DRL.
- We create our own NFT called the ParoleToken (PT) [23], which is deployed at the **OpenSea** testnet via **Optimism Goerli**. We perform various token transactions using PT and conducted simulations based on its data and traffic to validate the effectiveness and impact of the attack.
- We collected NFT snapshots from the optimistic rollups (**Arbitrum** and **Optimism**) and analyzed them to further validate the attack impact in a real-world NFT market.

We discuss necessary preliminary information in Section II. The related works are discussed in Section III. We introduce our proposed PARoLE attack in Section IV. In Section V, we discuss the technical details of the attack. We analyze three case studies in VI. In Section VII, we explain the empirical analysis and findings. Finally, we discuss a potential defense technique against the PARoLE attack in Section VIII. At last, we conclude the work in Section IX.

## II. Background

In this section, we will discuss a few preliminary concepts that will help explain the proposed PARoLE attack later.

### A. Optimistic Rollup

Blockchain systems currently fall significantly short in providing a service quality equivalent to centralized systems, especially concerning transactions per second [24], [25]. Optimistic rollup is an L2 scaling solution for blockchain networks that addresses scalability concerns by processing transactions off-chain and ensuring their validity through an optimistic approach backed by a challenge mechanism [26]. Rollup systems are deployed using Layer 1 (L1) smart contracts, where participants, such as aggregators and verifiers, engage with this contract. The basic workflow of optimistic rollups is illustrated in Figure 1, which involves several steps. At first, a user will need L2 tokens ($t^{L2}$) to interact with the rollup operators, which can be exchanged with other cryptocurrencies ($C^{L1}$) via the L1 smart contract. The L2 transactions are sent to Bedrock's Mempool, from where the aggregators collect and execute them. A user can send his transactions to the Bedrock's Mempool directly (User 1 sending $TX_{1,t^{L2}}$), or he can send the transaction via the L1 smart contract (User 2 sending $TX_{2,C^{L1}}$). In this work, we primarily consider the second approach to generalize the proposed attack for L1 NFT transactions. Then, the aggregator processes the transactions, generates a cryptographic aggregate of these transactions along with the Merkle state root of the L2 chain, and submits them to the verifiers. The verifiers on L1 monitor these submissions and can raise challenges if they detect any invalid or fraudulent transactions within the batch. A challenge period follows, during which the fraud-proof is inspected to dispute the optimistic assumption. The disputed transactions are reverted if fraud is proven, and the fraudster loses a bonded security deposit. However, if no valid challenge is raised, the transactions are considered finalized and are added to blockchain [27].

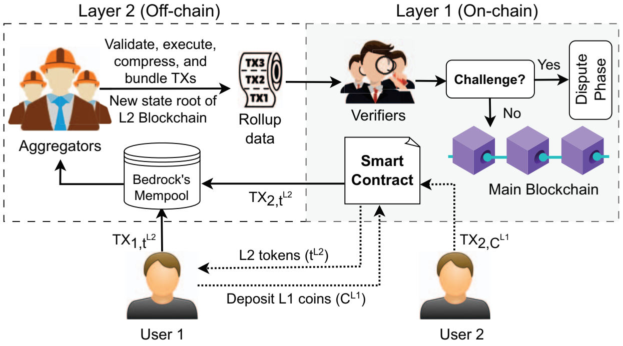


Fig. 1. Optimistic rollup workflow.

### B. Ethereum Tokens: ERC-20 and ERC-721

Ethereum tokens are digital assets created and managed on the Ethereum blockchain through the use of smart contracts. While resembling the cryptocurrency coins, the tokens lack an independent blockchain of their own; instead, they are constructed atop an existing one [28]. The predominant standard, ERC-20, facilitates the creation of fungible tokens, which are interchangeable and widely employed for purposes like crowdfunding through initial coin offerings and representing ownership of various assets [29]. The ERC-721 standard is a set of rules and protocols for the creation and management of NFTs. Unlike ERC-20 tokens, which can be exchanged on a one-to-one basis, ERC-721 tokens are non-fungible, meaning

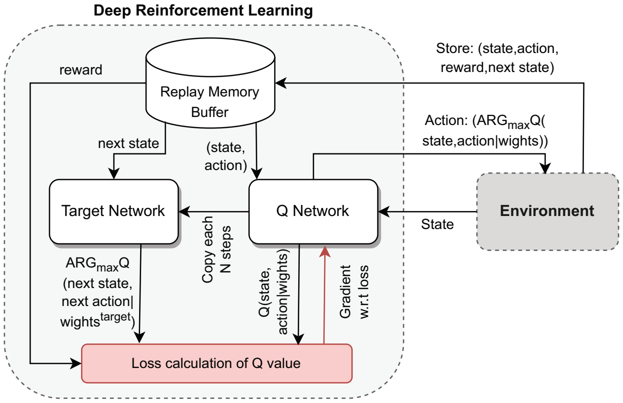

Deep Reinforcement Learning
reward
Replay Memory Buffer
Store: (state,action, reward,next state)
next state
(state, action)
Action: ($ARG_{max}Q$( state,action|wights))
Target Network
Q Network
Environment
State
Copy each N steps
$ARG_{max}Q$ (next state, next action| $wights^{target}$)
Q(state, action|wights)
Gradient w.r.t loss
Loss calculation of Q value

Fig. 2. DQN architecture.

each token is distinct and cannot be exchanged on a like-for-like basis, rather there will at least be a price difference [20]. The ERC-721 standard ensures that each token has a unique identifier, making it easily distinguishable from other tokens, and includes features for managing ownership, transferring tokens, and retrieving token metadata.

### *C. Reinforcement Learning*

Reinforcement Learning (RL) is a type of machine learning paradigm where an agent learns to make decisions by interacting with an environment [30]. The agent takes actions in the environment, receives feedback in the form of rewards or punishments, and uses this information to improve its decision-making over time. The goal of the agent is to learn a policy, a mapping from states to actions, that maximizes the cumulative expected reward [31]. RL algorithms can be categorized into model-free and model-based approaches, where model-free methods, like Q-learning, directly learn the optimal policy or value function from the data. DRL combines RL with deep learning, using neural networks to approximate complex policies or value functions [32]. Although a significant portion of DRL research has been focused on applications in video games and simulated control scenarios, DRL has demonstrated potential in acquiring sophisticated real-world skills (i.e., trading [33]). Unlike traditional Q-learning, where an agent manages a Q-table, the DRL algorithm deep Q-network (DQN) utilizes a neural network (Q-network) for learning. This approach, effective in complex real-world environments, is illustrated in Figure 2. To train the Q-network, the agent's experiences are stored as training data in a repository known as the replay memory buffer. To enhance the stability of learning, a supplementary neural network, called the 'Target network,' is included in the DQN framework. Weights from the Q-network are periodically copied to the target network, and the latter predicts future Q-values for subsequent states, contributing to the loss calculation in the Q-network's predictions.

## III. Related Work

The emergence of **MEV** in DeFi is becoming more prominent [34], where opportunistic traders secure better prices for their trades and profit by causing the victim's trade to execute at a disadvantageous price. Numerous studies explore various high-frequency trading strategies for ERC-20 tokens. Research by Qin et al. quantified blockchain extractable value (BEV) in DeFi smart contracts [35], yielding $540.54M in profit over 32 months. Bartoletti et al. proposed an Automated Market Makers (AMMs) attack strategy [36], exploiting transaction-ordering issues. **Frontrunning**, a prevalent adversarial strategy in DEXes, was explored by Daian et al. [37] and Eskandari et al. [38], highlighting risks in decentralized applications (DApps) and proposing mitigating strategies. Zhou et al. presented an AMM defense strategy, unifying multiple AMMs for security and financial benefits [39]. Another noteworthy attack technique is **Cryptocurrency pump and dump** (CPD), which involves manipulating cryptocurrency prices through false promotions, creating a temporary surge (pump) to attract unsuspecting investors. This is followed by a quick sell-off (dump) by orchestrators to capitalize on inflated prices, leading to significant losses for buyers. Xu et al. conducted an empirical analysis of 412 pump-and-dump events, developing a predictive model with high precision [40]. Gandal et al. exposed deceptive activity on the Mt. Gox Bitcoin exchange (revealing 600,000 fraudulently acquired bitcoins), resulting in a spike in the USD-BTC exchange rate in late 2013 [41]. Kamps et al. used anomaly detection to identify potential pump-and-dump activities, revealing clustering on specific exchanges and coins, contributing to research in crime prevention for this emerging fraud problem [42]. None of these works, however, concentrated on NFTs, where the dynamics of the economic behavior resemble real-world assets.

The manipulation of ERC-721 token markets is becoming more widespread with the emergence of various innovative attack strategies. One of the more common techniques is the **NFT rug pull** (NRP), where the creators or sellers abandon an NFT project after attracting attention and investments. Roy et al. found that over 36% of promoted projects on Twitter were fraudulent, with a majority involving bot activity [43]. To address these issues, a machine learning classifier tool was introduced to proactively detect 382 new fraudulent NFT projects. Huang et al. conducted a comprehensive study of NRPs [44], identifying 253 cases and implementing a rule-based method to flag 7,487 rug pulls. Sharma et al. analyzed structural and behavioral properties of NRPs, examining 758 cases across 10 NFT marketplaces [45].

Another prevalent attack strategy is the **NFT Ponzi schemes**, where returns to earlier investors come from the contributions of new investors rather than business profits. Bartoletti et al. discussed the transition of Ponzi schemes from offline to digital spaces, particularly their adaptation to cryptocurrency platforms like Ethereum [46]. Chen et al. proposed a detection approach and an advanced classification model, identifying over 500 smart Ponzi schemes on Ethereum [47].

There exist other unethical activities in NFT trading, such as **wash trading** on the Ethereum NFT market, which affects 5.66% of all NFT collections with a total artificial volume of $3.4 billion, emphasizing the profitability and prevalence of wash trading and suggesting the need for protective mecha-

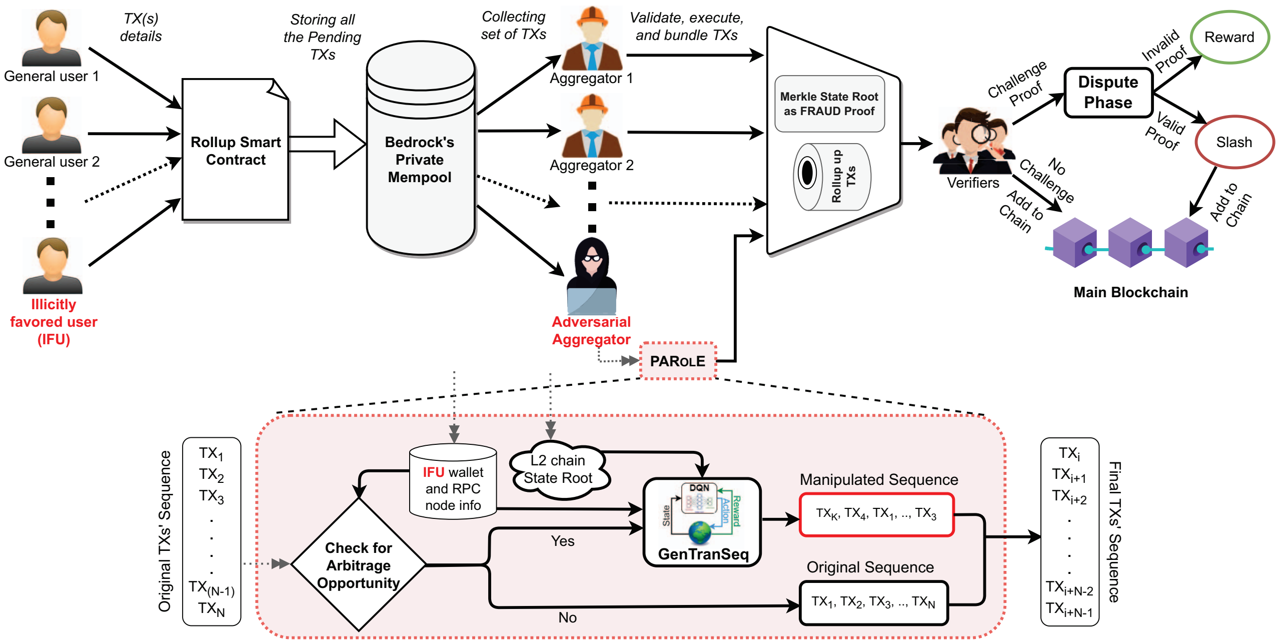


Fig. 3. Proposed PARoLE attack technique.

nisms in NFT markets [48]. Bonifazi et al. provide a novel "ex-post" analysis of wash trading activities in the NFT market, offering insights into the profitability of past illicit actions [49]. Serneels et al. proposed 3 innovative strategies for flagging suspicious wash trading activities in the NFT market, addressing the challenge of distorted token valuations [50]. Liu et al. proposed an approach to detect wash trading transactions using an algorithm by AnChain.AI [51], while Wen et al. introduced NFTDisk [52], a novel visualization tool designed for investors to identify wash trading activities, offering an intuitive approach through radial and flow-based visualization.

None of the above-discussed works exploited the vulnerability of optimistic rollup transaction processing, where an adversarial aggregator could alter the execution order of limited edition NFT transactions, providing illicit advantages to a user, without resorting to NFT market manipulation as seen in earlier research.

In terms of defense against transaction order manipulation attacks in the blockchain system, a range of approaches exist. Optimized trade execution methods, such as A2MM [39], offer targeted protection against known attacks like sandwich attacks by optimizing transaction parameters. However, their applicability is limited to specific applications, necessitating broader mitigation approaches. Professional Market Maker (PMM) models [53] introduce fairness to DEXes but face challenges in security and feasibility. Trusted third-party ordering schemes like Flashbots [54] and Gnosis [55] Protocol prioritize privacy and efficiency but compromise decentralization and security by relying on trusted entities. The eUTXO model [56] prevents front-running attempts but struggles with throughput limitations. While each approach offers benefits, none of them consider the threat of an adversarial validator (i.e., aggregator), who has the sequencing preference before the execution of the transaction. As such, these approaches lack the capability to detect and prevent sophisticated manipulation orchestrated by adversarial aggregators in L2 optimistic rollup systems.

## IV. PARoLE Attack: Key Idea

Here, we explain the main theoretical idea of the attack.

### A. *Theoretical Definition of* PARoLE

The PARoLE attack occurs through a collusion between an adversarial aggregator and an illicitly favored user (IFU), where the inability of Bedrock's Mempool in ensuring MEV resilient transaction sequencing is exploited. The Bedrock's Mempool serves as a temporary storage space for pending transactions before they are validated and added to the L2 chain. The legacy network generates a block for each transaction, processing them in a first-come-first-serve order, while Bedrock creates blocks at fixed intervals, necessitating a Mempool to hold pending transactions until they're incorporated into a block. Since Bedrock's Mempool is kept private to mitigate MEV [13], an adversarial aggregator does not have the opportunity to choose the eligible set of transactions from the Mempool to fabricate a profitable arbitrage for the IFU. Rather, each aggregator has to collect a set of transactions from the private Mempool according to priority sequence. At this stage, the aggregator uses the PARoLE module to analyze the transactions' order and, if possible, generate a new order of transactions, which will yield a higher final balance for the IFU than otherwise. Thus, in spite of Bedrock's efforts to mitigate MEV, aggregators' collection of transactions still presents opportunities for arbitrage, which poses substantial risks to market efficiency, liquidity providers, and overall system trust. Afterward, the transactions are validated and

TABLE I
LIST OF NOTATIONS

| ***Symbol*** | ***Definition*** |
|---|---|
| $M_k^{i,t}$ | Token with ID '$i$' is minted by user '$k$' as '$t$'-th TX |
| $B_k^t$ | Balance of user '$k$' after '$t$'-th TX confirmation |
| $\mathcal{P}^t$ | Price of NFT after '$t$'-th TX confirmation |
| $\mathcal{P}^0$ | Initial price of NFT |
| $T_{j,k}^{i,t}$ | Token with ID '$i$' is transferred from user '$j$' to user '$k$' as '$t$'-th TX |
| $O_k^{i,t}$ | Token with ID '$i$' is owned by user '$k$' after '$t$'-th TX confirmation |
| $\mathcal{S}^t$ | Available # of tokens to be minted after '$t$'-th TX confirmation |
| $\mathcal{S}^0$ | Total supply of tokens specified in the NFT smart contract |
| $D_k^{i,t}$ | Token with ID '$i$' is burnt by user '$k$' as '$t$'-th TX |
| $\mathcal{U}$ | Set of optimistic rollup users |
| $U_k$ | The '$k$'-th rollup user |
| $\mathcal{A}$ | Set of rollup aggregators |
| $A_k$ | The '$k$'-th rollup aggregator |
| $A_{\mathcal{P}}$ | The adversarial rollup aggregator committing PARoLE attack |
| $\mathcal{V}$ | Set of rollup verifiers |
| $V_k$ | The '$k$'-th rollup verifier |

executed in the generated *profitable order*, meaning there is no further manipulation of data and/or adversarial activity during execution by the adversarial aggregator. Thus, all the aggregators bundle their collected transactions and calculate the Merkle state root as fraud-proof. Finally, the verifiers will inspect the fraud-proof and should not initiate the dispute phase unless there is some adversarial manipulation from the rest of the aggregators (unrelated to PARoLE attack).

### B. Attack Workflow

In this part, we discuss the detailed workflow of the PARoLE attack, which is presented in Figure 3. The users in L1 send their transactions to the rollup smart contract residing in the L1. Bedrock generates blocks at regular intervals, necessitating a buffer to store pending transactions until they are incorporated into a block [13]. As mentioned before, the pending transactions of optimistic rollup are stored in a private Mempool. As shown in the figure, the aggregators collect the transactions and are supposed to execute them in order of their base and priority fees. However, the adversarial aggregator sends the transactions to the PARoLE module, which at first checks if there is an arbitrage opportunity for the IFU. The adversarial aggregator provides the private wallet information of the IFU, along with the remote procedure call (RPC) node URL used by the IFU at that time. Given the set of transactions collected by the aggregator and the IFU information, the potential arbitrage opportunity is checked. If there is a profitable arbitrage opportunity, then the GenTranSeq module is utilized. The same IFU information, along with the state root of L2's blockchain, is fed into the GenTranSeq module. This module is comprised of one of the most prevalent model-free DRL algorithms called the DQN.

The DQN module considers the current sequence of transactions as the initial observation of the environment and trains itself to predict the order of transactions that will maximize the balance of the IFU/IFUs. It takes into consideration the current state of the L2 chain to execute each candidate solution using an optimistic virtual machine (OVM) [57] and observe the balance update of the IFU. It also tracks the price update of the limited edition ERC-721 tokens, which are minted/transferred/burned during the transactions in consideration. Finally, if there is at least one alternate order that results in a higher balance for the IFU, the GenTranSeq module returns that order. If multiple candidate orders exist, then it tries to optimize the sequence to maximize the IFU's balance. A detailed technical analysis of the GenTranSeq module is provided later in Section V.

Finally, the adversarial aggregator validates and executes the transactions in the GenTranSeq module produced order. Then, they are bundled together to create the rollup of transactions. No further adversarial activities are performed after the altering of the transaction order. Later, the verifiers check the fraud-proof, and if none of the other aggregators performed any illegal action, the batch of rollup transactions is added directly to the main L1 blockchain.

## V. Technical Details

In this section, we present the technical details of the proposed attack by first formally defining the optimistic rollup system. Then, we discuss the arbitrage assessment technique utilized in the PARoLE module, followed by the detailed analysis of the GenTranSeq module's Markov Decision Process (MDP) design and the architecture of the DQN model.

### A. Formally defining Optimistic Rollup System

To interact with an optimistic rollup system, users must borrow L2 tokens from the optimistic rollup smart contract (ORSC) residing in L1. A user can exchange Ethereum coins (ETH) to get an equivalent amount of the L2 tokens. Each user $U_k$ ($\in \mathcal{U}$) sends its transactions to the ORSC for processing in the off-chain system:

$$U_k.\mathit{SubmitTX}(\mathit{TX}_k) \rightarrow \mathit{ORSC};\ \mathit{TX}_k \in \{M_k^{i,t}, T_{k,l}^{i,t}, D_k^{i,t}\}$$

The terms $M_k^{i,t}, T_{k,l}^{i,t}, D_k^{i,t}$ represent different types of NFT transactions (minting, transfer, and burning, respectively), which are defined in Table I. The ORSC stores the unconfirmed transactions in the Bedrock's Mempool:

$$\mathit{ORSC.PrivateStore}(T_{x,y}^{j,1}, ...., D_z^{l,N}) \rightarrow \mathit{BedRockMemPool}$$

A set of aggregators ($\mathcal{A}$) work as the primary operators of the rollup system, where they are responsible for transaction validation and execution. Each aggregator $A_k$ collects a set of transactions from Bedrock's Mempool and processes them in the order of their base and priority fees. Then, the executed transactions from $\mathcal{A}$'s aggregators are bundled together, and the new Merkle state root of the L2 chain is computed as the *fraud-proof*:

$$\mathcal{A}.\mathit{AggregateTX}(\mathit{BedRockMemPool}) \rightarrow \mathit{RollupTX},\ \mathit{Proof}$$

Afterward, the set of verifiers ($\mathcal{V}$) is responsible for checking the validity of the executed transactions by checking the generated *fraud-proof*, within a predefined challenge period. If no verifier challenges the proof, then:

$$\mathcal{A}.\mathit{SubmitBlock}(\mathit{RollupTX}) \rightarrow \mathit{ORSC}$$
$$\mathit{ORSC.ConfirmBlock}(\mathit{RollupTX}) \rightarrow \mathit{Etherium}$$

If at least one verifier $V_k$ challenges the *fraud-proof*, and the proof was indeed invalid, then the responsible aggregator $A_k$'s bond will be slashed:

$$V_k.Challenge(\mathcal{A}.Proof) \rightarrow Success$$
$$A_k.Bond = A_k.Bond - A_k.SlashBond()$$

If $V_k$ challenges the *fraud-proof*, however, the proof was valid, then $V_k$'s bond will be slashed:

$$V_k.Challenge(\mathcal{A}.Proof) \rightarrow Fail$$
$$V_k.Bond = V_k.Bond - V_k.SlashBond()$$

The above discussion formally summarizes the workflow of an optimistic rollup system. In the next section, we will discuss PARoLE attack using the notations introduced here.

### B. Assessment of Potential Arbitrage Opportunity

As mentioned in Section IV, the adversarial aggregator $A_{\mathcal{P}}$ processes the collected set of transactions through the PARoLE module. At first, the potential arbitrage opportunity is checked by analyzing the different types of transactions and IFU's involvement in those transactions. There can be three types of NFT transactions:

*1) Minting:* Refers to the creation of a new NFT using the NFT's smart contract. Each NFT has a unique identifier or ID, that distinguishes it from other tokens from the same NFT. Minting of NFT with ID $i$ is possible only if:

$$M_k^{i,t} \rightarrow (B_k^{t-1} \geq \mathcal{P}^{t-1}) \wedge (\mathcal{S}^{t-1} \geq 1) \tag{1}$$

which means user $U_k$'s minting request of $i$-th NFT can be executed as the $t$-th transaction, only if: (i) following the confirmation of $(t-1)$-th transaction, $U_k$'s balance $B_k^{t-1}$ is greater or equal to the price of the NFT ($\mathcal{P}^{t-1}$), and (ii) available NFTs to be minted $\mathcal{S}^{t-1}$ is greater or equal to 1. If both the constraints are satisfied, then:

$$\begin{aligned} O_k^{i,t} &= True \\ B_k^t &= B_k^{t-1} - \mathcal{P}^{t-1} \\ \mathcal{S}^t &= \mathcal{S}^{t-1} - 1 \end{aligned} \tag{2}$$

The above operations are performed when the minting transaction gets executed.

*2) Transfer:* Refers to the transfer of ownership of a particular instance of the NFT. In simple terms, one user sells an NFT that he owns to another user. The transfer of NFT with ID $i$ from user $U_k$ to user $U_j$ is possible, only if:

$$T_{k,j}^{i,t} \rightarrow (B_j^{t-1} \geq \mathcal{P}^{t-1}) \wedge O_k^{i,t-1} \tag{3}$$

meaning the transfer request of the $i$-th NFT can be executed as the $t$-th transaction, only if: (i) following the confirmation of $(t-1)$-th transaction, $U_j$'s balance $B_j^{t-1}$ is greater or equal to the price of the NFT ($\mathcal{P}^{t-1}$), and (ii) the token with ID $i$ is owned by user $U_k$ ($O_k^{i,t-1}$). If both are satisfied, then:

$$\begin{aligned} O_j^{i,t} &= True \\ B_j^t &= B_j^{t-1} - \mathcal{P}^{t-1} \\ B_k^t &= B_k^{t-1} + \mathcal{P}^{t-1} \end{aligned} \tag{4}$$

The above operations are performed when the transfer transaction gets executed.

**Algorithm 1:** PARoLE Algorithm

```
Function PAROLE ($U_{IFU}$, $Chain^{L2}$, $TxSeq^{Original}$):
    if Arbitrage($U_{IFU}$, $TxSeq^{Original}$) then
        $(\gamma, \epsilon, d, \alpha) \leftarrow$ Set
        for $ep \in Episodes$ do
            $State \leftarrow TxSeq^{Original}$; $Env \leftarrow \{U_{IFU}, Chain^{L2}\}$
            for $sp \in MaxSteps$ do
                if $rand(0,1) \geq \epsilon$ then
                    $action \leftarrow QNet(State)$
                end
                else
                    $action \leftarrow rand(Actions)$
                end
                $\{r_{sp}, State', \text{Profit}\} \leftarrow Env.Act(action, State)$
                $QNet.update(Loss^{TD} = TargetNet(State))$
                $State \leftarrow State'$; $\mathcal{R}^{ep} \leftarrow \mathcal{R}^{ep} + r_{sp}$
                $TargetNet.copy(QNet)$ if Profit
            end
            $TxSeq^{Final} \leftarrow State$
        end
    end
    else
        $TxSeq^{Final} \leftarrow TxSeq^{Original}$
    end
return $TxSeq^{Final}$;
```

*3) Burning:* Refers to the destroying of an already minted NFT using the NFT smart contract. A user $U_k$ can burn its token with ID $i$, only if:

$$D_k^{i,t} \rightarrow O_k^{i,t-1} \tag{5}$$

which means if $U_k$ owned $i$-th NFT after the confirmation of $(t-1)$-th transaction, then he can burn the NFT as the $t$-th transaction. If the above constraint is satisfied, then:

$$\begin{aligned} O_k^{i,t} &= False \\ \mathcal{S}^t &= \mathcal{S}^{t-1} + 1 \end{aligned} \tag{6}$$

The above operations are performed when the burning transaction gets executed.

By analyzing the set of transactions collected by $A_{\mathcal{P}}$, the potential for arbitrage opportunity is assessed. Since there can be three kinds of transactions, the order of their execution can significantly influence the final outcome of the system. The minting and burning transactions change the price per unit NFT, since these transactions decrease/increase the available number of NFTs still to be minted (Equation 2 and 6), in the case of the limited edition variant of ERC-721 token. Importantly, specific transactions can only be executed when positioned at a particular point in the sequence, and their execution is precluded if placed at other points due to the transaction constraints being unsatisfied (Equation 1, 3, and 5). Thus, during the assessment, it is crucial to verify the execution of specific transactions, all of which would have satisfied the constraints in the original sequence.

There is potential for profitable arbitrage for the IFU, if he is involved in multiple transactions within the set of transactions collected by $A_{\mathcal{P}}$. Ideally, he should at least be involved in a pair of minting and transfer transactions, while being involved in more transactions increases the chance for potential arbitrage opportunities. In the next section, we will discuss how the GENTRANSEQ module processes the original sequence, given the IFU and L2 chain state information.

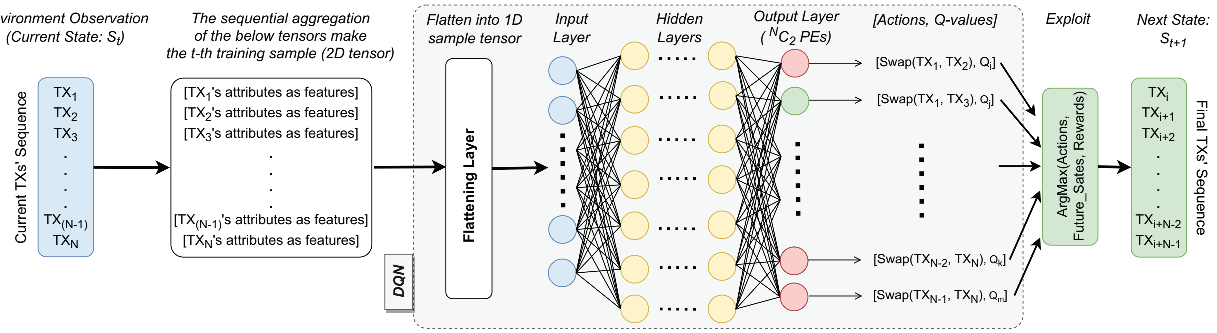

Environment Observation
(Current State: $S_t$)
Current TXs' Sequence
$TX_1$
$TX_2$
$TX_3$
$TX_{(N-1)}$
$TX_N$
The sequential aggregation
of the below tensors make
the t-th training sample (2D tensor)
[$TX_1$'s attributes as features]
[$TX_2$'s attributes as features]
[$TX_3$'s attributes as features]
[$TX_{(N-1)}$'s attributes as features]
[$TX_N$'s attributes as features]
DQN
Flatten into 1D
sample tensor
Flattening Layer
Input
Layer
Hidden
Layers
Output Layer
( $^NC_2$ PEs)
[Actions, Q-values]
[Swap($TX_1$, $TX_2$), $Q_i$]
[Swap($TX_1$, $TX_3$), $Q_j$]
[Swap($TX_{N-2}$, $TX_N$), $Q_k$]
[Swap($TX_{N-1}$, $TX_N$), $Q_m$]
Exploit
ArgMax(Actions,
Future_Sates, Rewards)
Next State:
$S_{t+1}$
$TX_i$
$TX_{i+1}$
$TX_{i+2}$
$TX_{i+N-2}$
$TX_{i+N-1}$
Final TXs' Sequence

Fig. 4. The Pre-processing and deep Q-network architecture of the GenTranSeq module.

### C. Analysis of the GenTranSeq Module

In this part, we discuss the GenTranSeq module in detail by first illustrating the modeling of the MDP for the transaction re-ordering problem and then discussing the DQN architecture.

*1) Designing the MDP:* We conceptualize the problem as a Markov game, an extension of the MDP. Specifically, in DRL, an MDP is typically represented as a tuple comprising states, actions, transition probabilities, rewards, and a discount factor. We designed an advanced MDP to represent the transaction re-ordering environment, where a DQN agent will be trained for a certain number of episodes (an "episode" referring to a single run or instance of the agent interacting with the environment from start to finish). In each episode, the agent receives a fresh set of transactions in their original sequence, and the agent is tasked with executing a certain number of actions (limited by a predefined maximum bound, i.e., maximum steps) to produce an altered sequence that meets the attack objective. This MDP is expressed as a six-tuple: {States, Actions, Reward, Policy ($\pi$), Discount Factor ($\gamma$), Exploration ($\epsilon$)}. Each component of this tuple will be explored in detail in the following sections.

***States:*** The set of states represents all the possible observations an agent can experience within the environment. In this case, an observation is a sequence of transactions. Thus, the DQN agent observes the current state, i.e., the current order of the transactions collected by $A_\mathcal{P}$ and takes an appropriate action. We consider the number of transactions an aggregator collects for processing purposes as the individual "Mempool" size of that aggregator (distinct from Bedrock's Mempool). In the evaluation section (Section VII), we use the term "Mempool" to refer to the aggregator's Mempool.

The environment consists of all the possible states an agent can be in. If the "Mempool" size of the aggregator is $N$ (collects $N$ number of transactions from Bedrock's Mempool), then the total number of possible states for the agent will be '***N!***'. There can be multiple candidate states (i.e., order of transactions) that result in a higher final balance for the IFU. However, the maximum possible balance can be achieved through a subset of those candidate states. The DQN agent, by solving this non-linear optimization problem, learns the intelligent ordering of those transactions and ensures maximum final balance for the IFU.

***Action:*** The set of actions represents the range of activities an agent can perform to interact with the environment. Since a DRL agent solves problems by making sequential decisions over time, a solution involves taking a set of actions. For this problem, an action is swapping two transactions from the current sequence of transactions (i.e., current state). During the initial stage of training, an agent might perform a lot of swappings to reach the goal state; however, with a sufficient amount of training, the number of swaps (i.e., actions) will be optimized. For example, in a grid world, to reach a goal cell, an untrained agent will take many moves (action: moving in all possible directions in the grid world); however, as it is more trained, the path length (number of actions) will be optimized.

If the "Mempool" size of the aggregator is $N$ (collects $N$ number of transactions from Bedrock's Mempool), then the **total number of possible actions** for the DQN agent will be $^NC_2$, i.e., choice of any two transactions form the $N$ transactions.

***Reward:*** The reward function, shaped by the rewards, governs the learning process of the agent. Achieving an order that results in a better final balance for the IFU results in positive rewards, while producing an order that results in a worse final balance for the IFU results in penalties. Additionally, an agent is penalized if it takes an action that fails to guide the agent towards an increasing final balance for the IFU. If the reward of $i$-th episode is $\mathcal{R}^i$, then:

$$\mathcal{R}^i = \sum_{k=1}^{m} r_k \text{ units} \tag{7}$$

here, $m$ is the total number of steps per episode, and $r_k$ is the reward achieved at that step. The reward for the $k$-th step is calculated by:

$$r_k = \mathcal{W} \times (B_{IFU}^{N,k} - B_{IFU}^{N,0}) \tag{8}$$

where, $B_{IFU}^{N,0}$ and $B_{IFU}^{N,k}$ presents the balance of IFU after $N$-th transaction execution with the *original sequence* and the *altered sequence after $k$-th action*, respectively. The weight factor $\mathcal{W}$ is set to a high positive value for penalizable action, while it is set to '1' for other cases.

***Policy ($\pi$):*** Policy is the learning of agents through interaction and exploration within the environment, specifying the action an agent will undertake in a given state. In this scenario, the policy is simply determined by the weights and biases of the Q-network, dictating the action an agent takes based on observations of the state.

***Discount Factor ($\gamma$):*** The discount factor plays a crucial role in determining the extent to which DQN agents prioritize rewards in the distant future relative to those in the immediate future. Ranging from zero to one, this parameter influences the agent's level of foresight. Setting $\gamma$ to zero renders the agent fully myopic, focusing solely on immediate rewards. A reward $r_i$ in the future that occurs after $X$ steps will be discounted by a factor of $X$ (i.e., the reward will be $(r_i)^X$).

***Exploration ($\epsilon$):*** The exploration parameter governs the agent's action behavior, typically initialized close to 1, indicating that there is an almost certain probability that the agent will take random actions to interact with the environment as exploration behavior. Those actions shape the learning of the Q-network. As the episode advances through additional steps, the value of $\epsilon$ decays, reducing the probability of random actions. A lower $\epsilon$ value signifies a higher probability of the agent choosing actions based on the Q-network, representing exploitation behavior. The $\epsilon$ in GenTranSeq module is reduced each episode with the following equation:

$$\epsilon_i = \epsilon_{min} + (\epsilon_{max} - \epsilon_{min})^{-(d \times i)} \tag{9}$$

where, $\epsilon_i$ is exploration parameter of $i$-th episode, and $d$ represents the decay parameter.

*2) Architecture of the DQN:* This part discusses the DQN model (Figure 4) utilized in the GenTranSeq module. Whenever the exploitation behavior is chosen for a step, the DQN is utilized for determining the next action. At first, the original transaction set is passed as the initial input to the GenTranSeq module, where each transaction is converted into a 1-dimensional (1D) tensor by encoding each attribute of the transaction. Generally, it is an eight-element tensor, including flags like the involvement of IFU in the transaction, the type of the transaction (i.e., transfer/minting/burning), and values like current token price, available tokens to be minted, etc. Aggregating all the tensors from the transaction list makes the 2D tensor input to the DQN model. The first layer is the flattening layer, converting the input into a 1D vector and passing it to the input layer, where there will be $8 \times N$ processing elements (PEs), given the transactions' sequence length is $N$. After multiple hidden layers, the output layer predicts the action and the associated Q-value (given the next state). As mentioned earlier, ${}^{N}C_2$ actions are possible with transactions' sequence length of $N$, indicating that the output layer will have ${}^{N}C_2$ PEs. Finally, the action with the maximum Q-value is selected, and corresponding transactions are swapped to create an alternate sequence as the next state. The same steps are performed for the next step.

TABLE II
Modeling Parameters of GenTranSeq module

| ***Parameter Name*** | ***Assigned Values*** |
|---|---|
| *Exploration parameter ($\epsilon$)* | 0.95 |
| *Epsilon decay (d)* | 0.05 |
| *Discount factor ($\gamma$)* | 0.618 |
| *Episodes* | 100 |
| *Steps (Each episode)* | 200 |
| *Learning rate ($\alpha$)* | 0.7 |
| *Reply memory buffer size* | 5,000 |
| *Q-network update* | Every 5 steps |
| *Target network update* | Every 30 steps |

For updating the Q-network through backpropagation, the temporal difference (TD) error is calculated as the prediction loss. This is actually the difference between the true Q-value of a state-action pair and the value estimated by the Q-network. The target network is utilized for the estimated true Q-value prediction, which utilizes the Bellman equation [58] and the discounted future rewards. All the values of different hyperparameters used in the DQN are summarized in Table II. Initially, the agent explores extensively with a high exploration rate ($\epsilon$) of 0.95, gradually decreasing exploration over time (with $d$ set to 0.05) for maintaining adaptability. Experimentation with learning rates ($\alpha$) ranging from 0.05 to 0.75 shows 0.7 as favorable for rapid learning and stability. After investigating, we found a discount factor of 0.618 balances short-term and long-term rewards effectively. Limiting the training to 100 episodes prevents overfitting, while setting 200 steps within an episode are found to be adequate. A large reply memory buffer aids generalization, and updating the Q-network every 5 steps and target network every 30 steps stabilizes training and accelerates learning. The PARoLE attack's steps are shown in Algorithm 1 in a simplified way using the notations introduced in this section.

## VI. Case Studies

In this section, we quantify the attack benefit for the IFU through three case studies, as shown in Figure 5. The first case illustrates the original order of the transaction execution, while the latter cases show two alternative orders of transaction execution and the corresponding attack benefits.

### *A. Status of the System:*

For all the case studies, the IFU of the adversarial aggregator has an initial L2 token balance of 1.5 ETH and owns 2 PARoLE tokens. The current price of one unit limited edition PARoLE token is 0.4 ETH, which varies according to scarcity (demand/supply). Accordingly, only the minting and burning transactions update the price of each unit of PARoLE token, while transfer transactions keep the price as is. The price of the PARoLE token is updated according to the following equation:

$$\mathcal{P}^{t}_{PT} = \frac{\mathcal{S}^{0}_{PT}}{\mathcal{S}^{t}_{PT}} \times \mathcal{P}^{0}_{PT} \tag{10}$$

Here, $\mathcal{P}^{t}_{PT}$ represents the price of the PARoLE token after the $t$-th transaction is executed, and $\mathcal{P}^{0}_{PT}$ represents the initial price of the same. The maximum number of PARoLE tokens that could be minted, i.e., $\mathcal{S}^{0}_{PT}$ is set to 10, and the initial price, $\mathcal{P}^{0}_{PT}$ is set to 0.2 ETH. Among the maximum supply of 10, 5 PARoLE tokens are already minted (i.e., remaining tokens to be minted, $\mathcal{S}^{t}_{PT}$ is 5), and the price per unit has increased to 0.4 ETH, according to Equation 10. Only the balance of IFU is calculated in the case studies since the GenTranSeq module alters the order of transactions for maximizing the IFU

| | *Original TX Sequence* | *PT Price* | *IFU Total Balance* |
|---|---|---|---|
| | *(TXs involving PT)* | *(1 unit)* | *L2 balance+(PTs owned)*Price* |
| $TX_1$ | Transfer PT: $U_1 \rightarrow U_2$ | 0.4 ETH | 1.5 + (2*0.4) = 2.3 ETH |
| $TX_2$ | Mint PT: $U_{19}$ | 0.5 ETH | 1.5 + (2*0.5) = 2.5 ETH |
| $TX_3$ | Transfer PT: IFU $\rightarrow U_{11}$ | 0.5 ETH | 2.0 + (1*0.5) = 2.5 ETH |
| $TX_4$ | Transfer PT: $U_{19} \rightarrow U_6$ | 0.5 ETH | 2.0 + (1*0.5) = 2.5 ETH |
| $TX_5$ | Mint PT: IFU | 0.66 ETH | 1.5 + (2*0.66) = 2.82 ETH |
| $TX_6$ | Transfer PT: $U_{13} \rightarrow U_3$ | 0.66 ETH | 1.5 + (2*0.66) = 2.82 ETH |
| $TX_7$ | Burn PT: $U_2$ | 0.5 ETH | 1.5 + (2*0.5) = 2.5 ETH |
| $TX_8$ | Transfer PT: $U_1 \rightarrow$ IFU | 0.5 ETH | 1.0 + (3*0.5) = 2.5 ETH |

(a)

| | *Altered TX Sequence* | *PT Price* | *IFU Total Balance* |
|---|---|---|---|
| | *(TXs involving PT)* | *(1 unit)* | *L2 balance+(PTs owned)*Price* |
| $TX_1$ | Transfer PT: $U_1 \rightarrow U_2$ | 0.4 ETH | 1.5 + (2*0.4) = 2.3 ETH |
| $TX_7$ | Burn PT: $U_2$ | 0.33 ETH | 1.5 + (2*0.33) = 2.16 ETH |
| $TX_5$ | Mint PT: IFU | 0.4 ETH | 1.17 + (3*0.4) = 2.37 ETH |
| $TX_4$ | Transfer PT: $U_{19} \rightarrow U_6$ | 0.4 ETH | 1.17 + (3*0.4) = 2.37 ETH |
| $TX_3$ | Transfer PT: IFU $\rightarrow U_{11}$ | 0.4 ETH | 1.57 + (2*0.4) = 2.37 ETH |
| $TX_6$ | Transfer PT: $U_{13} \rightarrow U_3$ | 0.4 ETH | 1.57 + (2*0.4) = 2.37 ETH |
| $TX_2$ | Mint PT: $U_{19}$ | 0.5 ETH | 1.57 + (2*0.5) = 2.57 ETH |
| $TX_8$ | Transfer PT: $U_1 \rightarrow$ IFU | 0.5 ETH | 1.07 + (3*0.5) = 2.57 ETH |

(b)

| | *Altered TX Sequence* | *PT Price* | *IFU Total Balance* |
|---|---|---|---|
| | *(TXs involving PT)* | *(1 unit)* | *L2 balance+(PTs owned)*Price* |
| $TX_1$ | Transfer PT: $U_1 \rightarrow U_2$ | 0.4 ETH | 1.5 + (2*0.4) = 2.3 ETH |
| $TX_7$ | Burn PT: $U_2$ | 0.33 ETH | 1.5 + (2*0.33) = 2.16 ETH |
| $TX_8$ | Transfer PT: $U_1 \rightarrow$ IFU | 0.33 ETH | 1.17 + (3*0.33) = 2.16 ETH |
| $TX_5$ | Mint PT: IFU | 0.4 ETH | 0.84 + (4*0.4) = 2.44 ETH |
| $TX_4$ | Transfer PT: $U_{19} \rightarrow U_6$ | 0.4 ETH | 0.84 + (4*0.4) = 2.44 ETH |
| $TX_3$ | Transfer PT: IFU $\rightarrow U_{11}$ | 0.4 ETH | 1.24 + (3*0.4) = 2.44 ETH |
| $TX_6$ | Transfer PT: $U_{13} \rightarrow U_3$ | 0.4 ETH | 1.24 + (3*0.4) = 2.44 ETH |
| $TX_2$ | Mint PT: $U_{19}$ | 0.5 ETH | 1.24 + (3*0.5) = 2.74 ETH |

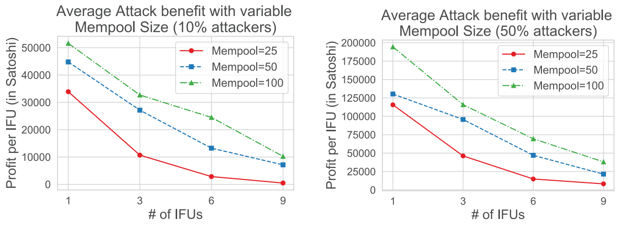
(c)

Fig. 5. Case Studies: a) Case 1: The resultant PT price and IFU balance update with the original transaction sequence. b) Case 2: The resultant PT price and IFU balance update with a candidate altered transaction sequence yielding better IFU final balance. c) Case 3: The resultant PT price and IFU balance update with optimally altered transaction sequence yielding maximum IFU final balance.

balance. The total balance of the IFU is the summation of the L2 token balance and the price of the PAROLE tokens he owns. In all the cases in the subsequent sections, the transactions are numbered in the order of the original sequence.

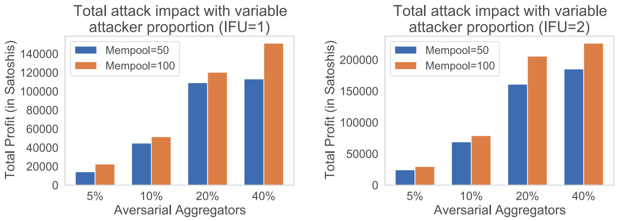
***Case 1:*** In the first case, as shown in Figure 5(a), the IFU has an initial balance of 2.3 ETH. He is involved in three transactions, two of which are transfer transactions, and one is a minting transaction (all colored red). After the first transaction, the IFU's balance remains unchanged since it was a transfer transaction. The second transaction is a minting transaction performed by user $U_{19}$, which changes the price of a unit PAROLE token from 0.4 ETH to 0.5 ETH (as per Equation 10). The IFU's balance is updated since the price of PAROLE token has increased. His L2 token balance remains the same, while PAROLE portion valuation increases from 0.8 ETH to 1.0 ETH. Although the third transaction involved the IFU, it did not update the IFU's balance since it was a transfer transaction. As he sold one of his PAROLE tokens, he now has an L2 token balance of 2.0 ETH, and PAROLE portion valuation decreases to 0.5 ETH. The fourth transaction keeps the IFU's balance the same since it was a transfer transaction. The IFU mints in the fifth transaction, which increases the per unit price of PAROLE token. His balance has also increased to 2.82 ETH due to the price update. The sixth transaction also did not update the IFU's balance since it is a transfer transaction. The seventh transaction is a burning transaction by user $U_2$, which increases the supply of PAROLE token. As a result, the per unit price reduces to 0.5 ETH from 0.66 ETH (Equation 10). Thus, IFU's balance is reduced to 2.5 ETH, where his L2 token balance remains the same (1.5 ETH), while PAROLE portion valuation decreases to 1.0 ETH from 1.32 ETH. Finally, in the eighth transaction, IFU buys a PAROLE token from user $U_1$, which does not update his total balance; however, his L2 token balance becomes 1.0 ETH, and PAROLE portion valuation becomes 1.5 ETH.

***Case 2:*** The second case, as presented in Figure 5(b), demonstrates an altered order of transactions generated by the GENTRANSEQ module, resulting in a higher final total balance for the IFU. Like the previous case, the IFU has an initial balance of 2.3 ETH. After the first transaction, the IFU's balance remains unchanged since it was a transfer transaction. However, this time, the burning transaction by user $U_2$ takes place as the second transaction, which increases the supply of PAROLE token. As a result, the per unit price reduces to 0.33 ETH from 0.4 ETH. Thus, IFU's balance is reduced to 2.16 ETH from 2.3 ETH, where his L2 token balance remains the same (1.5 ETH), while PAROLE portion valuation decreases. This time, the IFU mints at the third transaction, which increases the per unit price of PAROLE token. His balance has also increased to 2.37 ETH due to the price update, where his L2 token balance becomes 1.17 ETH, and the PAROLE portion valuation increases from 0.66 ETH to 1.2 ETH. The fourth transaction, as before, keeps the IFU's balance the same since it was a transfer transaction. Although the fifth transaction involved the IFU, it did not update the IFU's balance since it was a transfer transaction. As he sold one of his PAROLE tokens, he now has an L2 token balance of 1.57 ETH, and PAROLE portion valuation decreases to 0.8 ETH. The sixth transaction also did not update the IFU's balance since it is a transfer transaction. The seventh transaction is a minting transaction performed by user $U_{19}$, which changes the price of a unit PAROLE token from 0.4 ETH to 0.5 ETH (as per Equation 10). The IFU's balance is updated since the price of PAROLE token has increased. His L2 token balance remains



(a) (b)

Fig. 6. Comparing the average attack profit while serving different numbers of IFUs, with variable Mempool sizes: (a) 10% of the aggregators were adversarial, and (b) 50% of the aggregators were adversarial.



(a) (b)

Fig. 7. Contrasting the attack's impact in terms of total profit augmentation with different percentages of aggregators as adversary (with variable Mempool sizes): (a) Serving 1 IFU, and (b) Serving 2 IFUs.

the same, while PARoLE portion valuation increases from 0.8 ETH to 1.0 ETH. Finally, in the eighth transaction, IFU buys a PARoLE token from user $U_1$, which does not update his total balance; however, his L2 token balance becomes 1.07 ETH, and PARoLE portion valuation becomes 1.5 ETH.

***Case 3:*** The third case, as presented in Figure 5(c), demonstrates the optimal altered order of transactions generated by the GenTranSeq module, resulting in the maximum final total balance for the IFU. Like case 1, the IFU has an initial balance of 2.3 ETH. After the first transaction, the IFU's balance remains unchanged since it was a transfer transaction. Similar to case 2, the burning transaction by user $U_2$ takes place as the second transaction, which increases the supply of PARoLE token. As a result, the per unit price reduces to 0.33 ETH from 0.4 ETH. Thus, IFU's balance is reduced to 2.16 ETH from 2.3 ETH, where his L2 token balance remains the same (1.5 ETH), while PARoLE portion valuation decreases. In the third transaction, IFU buys a PARoLE token from user $U_1$, which does not update his total balance; however, his L2 token balance becomes 1.17 ETH, and PARoLE portion valuation becomes 0.99 ETH. This time, the IFU mints at the fourth transaction, which increases the per unit price of PARoLE token. His balance has also increased to 2.44 ETH due to the price update, where his L2 token balance becomes 0.84 ETH, and the PARoLE portion valuation increases from 0.99 ETH to 1.6 ETH. The fifth transaction, as before, keeps the IFU's balance the same since it was a transfer transaction. Although the sixth transaction involved the IFU, it did not update the IFU's balance since it was a transfer transaction. As he sold one of his PARoLE tokens, he now has an L2 token balance of 1.24 ETH, and PARoLE portion valuation decreases to 1.2 ETH. The seventh transaction also did not update the IFU's balance since it is a transfer transaction. Finally, the eighth transaction is a minting transaction performed by user $U_{19}$, which changes the price of a unit PARoLE token from 0.4 ETH to 0.5 ETH (as per Equation 10). The IFU's balance is updated since the price of PARoLE token has increased. His L2 token balance remains the same, while PARoLE portion valuation increases from 1.2 ETH to 1.5 ETH.

### B. Discussion on Findings

We observe that, in all three cases, the IFU's PARoLE token portion of the balance has a valuation of 1.5 ETH (three tokens priced at 0.5 ETH each). However, the beauty of the GenTranSeq module is displayed through the increased L2 token balance, which is increased in Case 2 (by 7%) and maximized in Case 3 (increased by 24%). The L2 token portion of the balance is the non-volatile part of the balance, meaning that it will not decrease with the burning of tokens (increased supply). Again, Case 2 will occur in the initial phase of the training, where the DQN agent only looks for a better final balance for the IFU. As the agent moves to new episodes, the rewards function is tuned toward maximizing profit for the IFU. Consequently, Case 3 takes place after the model is fully trained to optimize the altered sequence of transactions.

TABLE III
Behavior of PARoLE token in OpenSea Transactions

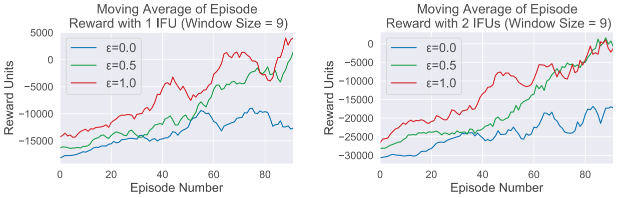
| *TX Type* | *TX Hash* | *Block Number* | *L1 state index* | *Gas usage* | *TX fees* |
|---|---|---|---|---|---|
| Minting | 0x8..f56 | 17934499 | 115922 | 90.91% | 253 Gwei |
| Transfer | 0x3..5f3 | 18183117 | 117994 | 69.84% | 142k Gwei |
| Burning | 0xe..cf2 | 18184325 | 118004 | 69.82% | 141k Gwei |

## VII. Evaluation

In this section, we conduct experimental analysis using our own real ERC-721 token PT [23] deployed at the **OpenSea** testnet via the **Optimism Goerli**. We conducted different token transactions with PT (one instance from each type of transaction is presented in Table III) and performed its data and traffic-based simulations to validate the effectiveness and impact of the attack. First, we analyze the model's efficiency in augmenting the IFU's balance. Then, we investigate the learning performance of the DQN model. Later, we assess the attack's impact in the real-world via NFT snapshots. Finally, we contrast the performance of the DQN model with well-known non-linear programming (NLP) solvers. All the experiments were performed on a computer with an 11th Gen Intel(R) Core (TM) i7-1195G7 @2.90GHz processor and 16 GB of memory.

### A. Influence of IFUs on Attack Profit

In this part, we investigate the influence of the number of IFUs served on the profit achieved through the PARoLE attack, as shown in Figure 6. Specifically, we show the average amount of profit achieved for each IFU. In the first case (Figure 6(a)), we assume that 10% of the aggregators are adversarial, i.e., launch the PARoLE attack. As mentioned



(a) (b)

Fig. 8. The moving average of the episode rewards accumulated by the DQN agent with variable $\epsilon$ parameter values, serving (a) 1 IFU, and (b) 2 IFUs.

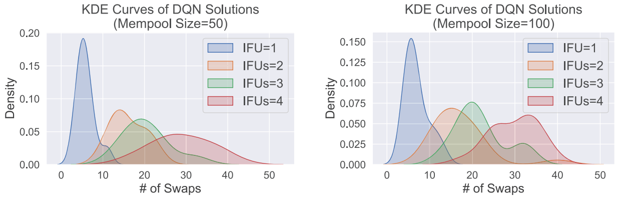
before in Section V, "Mempool" size indicates the number of transactions an aggregator collects for processing (different from Bedrock's Mempool). We observe that serving less number of IFUs incurs better results in terms of average profit per IFU, which is due to the fact that the transactions can only be re-ordered in '$N!$' number of ways, and very few alternate orders could increase the final balance for multiple IFUs. Thus, as we increase the number of IFUs to be served, the average profit amount keeps decreasing. We also vary the size of the Mempool and observe that a larger Mempool helps achieve better average profit. This is due to the fact that having more transactions to be ordered gives the DQN agent more flexibility in terms of possible alternate sequencing to serve the intended number of IFUs. However, the profit does not linearly increase as we keep enlarging the Mempool. The profit convergence can be observed through the lesser difference in profit for Mempool sizes 50 and 100, compared to the difference with Mempool sizes 25 and 50. We increase the portion of the adversarial aggregators in Figure 6(b) to 50% and observe that the profit per IFU has increased substantially. However, the first case would be more realistic in real-world optimistic rollup system. We will further analyze the effect of different adversarial aggregators' portions in the next section.

### B. *Validating Attack Impact w.r.t. Adversarial Proportion*

In this section, we experiment with different percentages of the aggregators as adversarial and observe the corresponding summations of total profits for all the IFUs, as shown in Figure 7. Initially, we consider 1 IFU to be served by the adversarial aggregators, illustrated in Figure 7(a). It is observed that the total profit increases as the proportion of adversarial aggregators is increased; however, in the case of Mempool size of 50, the trend of increase somewhat converges around 110k Satoshis (from 20% to 40%). Conversely, a linear increase is observed with a Mempool size of 100. The reason behind this is with a smaller Mempool, there are fewer alternate sequences to increase the total profit, even if there is a higher percentage of attackers. With a larger Mempool, each attacker finds more solutions and moves towards the optimal one. In Figure 7(b), 2 IFUs are served; however, the total profit increase is not linear. This refers back to the observation we achieved in the previous section.

### C. *Performance Assessment of the DQN Model*

In this section, we observe the reward accumulation throughout the training episodes of the DQN agent, as shown in Figure 8. We specifically present the moving average of the reward units with a window size of 9. In the first case (Figure 8(a)), we set the number of IFUs to one and observe the reward accumulation with different $\epsilon$ values. It is observed that when training starts with $\epsilon$ being set to '0', the average moving reward does not increase that much, even after being close to termination episodes. This is due to the fact that the agent always exploits the Q-network for action decisions. As a result, much of the environment (e.g., potential solutions with higher rewards) is left unexplored. The agent learns a few swapping behaviors and gets trapped in a local optimum. On the other hand, with $\epsilon$ being set to '1', the agent explores the solution space and performs significantly better. Even the first moving average (of the first nine episodes) is higher than the previous case. The choice of $\epsilon$ being set to '0.5' also derives intelligent behavior; however, the learning is a bit slow in terms of episode number compared to the '1' case. We further experiment with IFUs being set to two in Figure 8(b), and observe similar results. However, in this case, the rewards are in the range of approximately -30K to 1K units, which is worse compared to the one IFU case (approximately -18K to 5K units). It is evident that finding the sequence of transactions that serve more IFUs requires more exploration of the environment, causing more penalizable actions. Further, it is observed that the choice of '1' does not increase the reward prominently after around 70 episodes since the maximum achievable reward is reached.



(a) (b)

Fig. 9. Kernel Density Estimate curves for solution size in terms of the number of swaps performed by the DQN agent to find the first candidate solution: with (a) Mempool size = 50, and (b) Mempool size = 100.

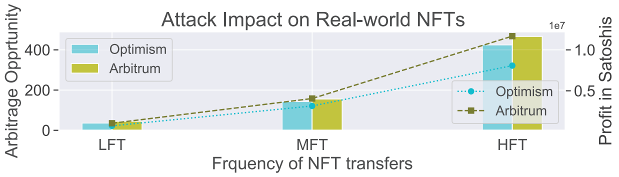
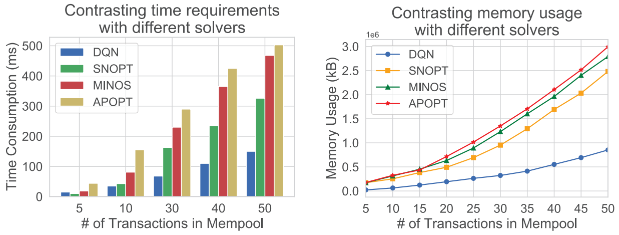


Fig. 10. Analyzing real-world monetary impact via NFT snapshots.

### D. *Distribution of Solution Sizes*

In this section, we discuss the distribution of solution sizes with different numbers of IFUs to serve, using Kernel Density Estimate (KDE) curves (Figure 9). In the first case (Figure 9(a)), we set the Mempool size to 50 and observe that the solutions with approximately five actions have the highest probability when only one IFU is served. This means that with a trained DQN agent, in most cases, five swaps will outcome a sequence that will maximize the balance of the only IFU.



(a) (b)

Fig. 11. Contrasting the DQN inference performance with non-linear programming solvers in terms of, (a) the execution times of the solvers, and (b) the memory usage by the solvers.

As we keep increasing the number of IFUs to be served, we observe the distribution to be more spread in larger ranges. We observe similar trends in Figure 9(b), where the Mempool size is increased to 100. However, this time, the three IFUs and four IFUs are concentrated on multiple peaked regions, which is due to the fact that with the increased Mempool size, the agent finds multiple candidate strategies with potentially larger solution space.

### *E. Attack impact in Real-world NFTs*

In this section, we analyze the impact of the PARoLE attack in real-world NFT marketplaces in terms of profit gain. We investigate historical snapshots of NFTs (deployed through optimistic rollup mainchains), including details such as prices, transaction volumes, and other relevant information (through wallet address and NFT minting contract address lookups via websites such as “holders.at” [59]). We searched for instances where the same NFT was priced differently at different times and looked for arbitrage opportunities among the transactions. As shown in Figure 10, we divide the NFTs into three categories according to the frequency of their transactions (FT): i. less than 100 ownerships (e.g., 0x7A..c8e deployed via Optimism) as low FT (LFT), ii. ownerships between 101 to 3000 (e.g., 0xCE..791 deployed via Arbitrum) as medium FT (MFT), and iii. more than 3000 ownerships as high FT (HFT). We observe that there is a higher arbitrage opportunity with the NFTs deployed via the Arbitrum chain compared to Optimism. We also calculate the total profit opportunity by deriving the relation we obtained through our simulation-based experiments. We want to emphasize that this analysis is solely based on our observation of the snapshots and experiments.

### *F. Contrasting DQN performance with NLP Solvers*

In this part, we contrast the DQN's inference performance (since IFU trains the model offline) with other existing solvers. As the NFT transaction re-ordering task is a non-linear optimization problem (discussed in Section I), we contrast with the well-known NLP solvers: Advanced Process OPTimizer (APOPT), Modular In-core Non-linear Optimization System (MINOS), and Sparse Non-linear OPTimizer (SNOPT). First, we compare the execution times of each solver with the DQN inference time, as shown in Figure 11(a). We observe that DQN consumes the minimal time among all the solvers compared. Although, for 5 transactions in the Mempool, we observe SNOPT performed slightly better than DQN, as we increased the number of transactions resembling real-world scenarios, SNOPT performance got worse, following a similar trend as the other NLP solvers. DQN, on the other hand, showed a rather linear increase with the larger sizes of Mempool. We also compare the memory usage of the solvers (Figure 11(b)) and observe that DQN inference consumed minimal memory, even if we keep increasing the Mempool size. These experiments validate the choice of utilizing DQN instead of the NLP solvers since time is critical in off-chain transaction processing, while memory usage directly influences the attack cost.

## VIII. Potential Defense Technique

The aggregators periodically interact with the Bedrock nodes to gather transactions from Bedrock's Mempool [13]. The problem with the current sequencing technique of Bedrock's Mempool is it prioritizes the transactions according to only the base and priority fees, which allows the opportunity for arbitrage. While in the pending state, if newer transactions with higher fees are received, the general approach that Mempools follow is to send the transactions with the lowest fees to the block behind [60].

As a potential defense technique to address the PARoLE attack, we propose to include the GenTranSeq module as an attack detection mechanism in Bedrock's Mempool. Initially, the order with the base and priority fee will be considered and sent to the GenTranSeq module to observe the worst case (maximum profit for any of the users involved in the pending transactions). If the worst case is below a predefined threshold (depending on the priority fee), then no action will be performed since the arbitrage is negligible, considering the priority of the transactions. However, if the worst case is above the calculated threshold, then the minimal number of involved transactions to avoid arbitrage will be sent to the block behind. In our future work, we will demonstrate the approach in detail and validate the effectiveness of the proposed defense technique.

## IX. Conclusion

In this work, we introduced a novel attack technique in optimistic rollup systems, enabling an adversarial aggregator to achieve profitable arbitrage by rearranging NFT transactions. We conducted a comprehensive analysis of the attack along with a detailed explanation of the DQN model utilized by the GenTranSeq module. We discussed a few case studies that explained how the PARoLE attack augments the non-volatile part of the IFU balance. Through our own test NFT-based simulations, we demonstrated the specific impacts of the attack, including monetary gains with variable environment setups. We found that serving less number of IFUs incurs better profit per IFU while having a larger Mempool makes it convenient for the attacker to look for an optimal solution with a potentially better solution space. Finally, we validated the attack's impact using real-world rollup NFT snapshots, analyzing the monetary benefits through the PARoLE attack.

# References

[1] S. Nakamoto, "Bitcoin whitepaper," *URL: https://bitcoin. org/bitcoin. pdf-(: 17.07. 2019)*, 2008.

[2] A. A. Khalil, J. Franco, I. Parvez, S. Uluagac, H. Shahriar, and M. A. Rahman, "A literature review on blockchain-enabled security and operation of cyber-physical systems," in *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 2022, pp. 1774–1779.

[3] D. Yang, C. Long, H. Xu, and S. Peng, "A review on scalability of blockchain," in *Proceedings of the 2020 the 2nd International Conference on Blockchain Technology*, 2020, pp. 1–6.

[4] T. Rajabi, A. A. Khalil, M. H. Manshaei, M. A. Rahman, M. Dakhilalian, M. Ngouen, M. Jadliwala, and A. S. Uluagac, "Feasibility analysis for sybil attacks in shard-based permissionless blockchains," *Distributed Ledger Technologies: Research and Practice*, vol. 2, no. 4, pp. 1–21, 2023.

[5] A. Singh *et al.*, "Sidechain technologies in blockchain networks: An examination and state-of-the-art review," *Journal of Network and Computer Applications*.

[6] J. Poon and V. Buterin, "Plasma: Scalable autonomous smart contracts," *White paper*, pp. 1–47, 2017.

[7] A. A. Khalil, M. A. Rahman, and H. A. Kholidy, "Fakey: Fake hashed key attack on payment channel networks," in *2023 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2023, pp. 1–9.

[8] A. A. Khalil and M. A. Rahman, "Ship: Securing hashed timelock contracts in payment channel networks," in *2023 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2023, pp. 1–2.

[9] L. T. Thibault, T. Sarry, and A. S. Hafid, "Blockchain scaling using rollups: A comprehensive survey," *IEEE Access*, 2022.

[10] J. Hamid, "Binance: All ethereum rollups are centralized," https://www.cryptopolitan.com/binance-all-ethereum-rollups-are-centralized/.

[11] Optimism-Documentation, "Bedrock explainer," https://community.optimism.io/docs/developers/bedrock/explainer/.

[12] R. Coll Aumatell, "Analysis and development of blockchain rollups," Master's thesis, Universitat Politècnica de Catalunya, 2021.

[13] Optimism-Documentation, "Mempool," https://community.optimism.io/docs/developers/bedrock/differences/#mempool.

[14] R. McLaughlin *et al.*, "A large scale study of the ethereum arbitrage ecosystem," in *USENIX Security 2023*.

[15] L. Zhou *et al.*, "On the just-in-time discovery of profit-generating transactions in defi protocols," in *2021 IEEE SP*.

[16] Y. Wang *et al.*, "Cyclic arbitrage in decentralized exchanges," in *Companion Proceedings of the Web Conference 2022*.

[17] V. von Wachter *et al.*, "Nft wash trading: Quantifying suspicious behaviour in nft markets," *arXiv preprint arXiv:2202.03866*, 2022.

[18] I. Bogatyy, "How we made $100k trading cryptokitties." https://medium.com/@ivanbogatyy/how-we-made-100k-trading-cryptokitties-2d69aebe715b.

[19] S. Malwa, "Did an "art heist" just happen on an ethereum cryptopunks nft?" https://cryptoslate.com/did-an-art-heist-just-happen-on-an-ethereum-cryptopunks-nft.

[20] B. White, A. Mahanti, and K. Passi, "Characterizing the opensea nft marketplace," in *Companion Proceedings of the Web Conference 2022*.

[21] R. Chohan and J. Paschen, "Nft marketing: How marketers can use nonfungible tokens in their campaigns," *Business Horizons*, 2023.

[22] V. François-Lavet *et al.*, "An introduction to deep reinforcement learning," *Foundations and Trends® in Machine Learning*, 2018.

[23] PARolE, "Paroletoken (pt) at opensea marketplace," https://testnets.opensea.io/assets/optimism-goerli/0xd27e14457926495bcf06a3c029ef3ef43a2c4a93/0.

[24] K. Croman *et al.*, "On scaling decentralized blockchains: (a position paper)," in *International conference on financial cryptography and data security*. Springer, 2016.

[25] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," *white paper*, vol. 3, no. 37, pp. 2–1, 2014.

[26] T. Yu *et al.*, "Dual-blockchain-based p2p energy trading system with an improved optimistic rollup mechanism," *IET Smart Grid*, 2022.

[27] T. Schaffner, "Scaling public blockchains," *A comprehensive analysis of optimistic and zero-knowledge rollups. University of Basel*, 2021.

[28] M. Di Angelo and G. Salzer, "Tokens, types, and standards: identification and utilization in ethereum," in *2020 IEEE DAPP*.

[29] P. Cuffe, "The role of the erc-20 token standard in a financial revolution: the case of initial coin offerings," 2018.

[30] M. A. Wiering and M. Van Otterlo, "Reinforcement learning," *Adaptation, learning, and optimization*, vol. 12, no. 3, p. 729, 2012.

[31] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[32] Y. Li, "Deep reinforcement learning: An overview," *arXiv preprint arXiv:1701.07274*, 2017.

[33] Z. Xiong *et al.*, "Practical deep reinforcement learning approach for stock trading," *arXiv preprint arXiv:1811.07522*, 2018.

[34] J. Piet *et al.*, "Extracting godl [sic] from the salt mines: Ethereum miners extracting value," *arXiv preprint arXiv:2203.15930*, 2022.

[35] K. Qin *et al.*, "Quantifying blockchain extractable value: How dark is the forest?" in *2022 IEEE SP*.

[36] M. Bartoletti, J. H.-y. Chiang, and A. Lluch Lafuente, "Maximizing extractable value from automated market makers," in *Conference on Financial Cryptography and Data Security*. Springer 2022, 2022.

[37] P. Daian, S. Goldfeder, T. Kell, Y. Li, X. Zhao, I. Bentov, L. Breidenbach, and A. Juels, "Flash boys 2.0: Frontrunning, transaction reordering, and consensus instability in decentralized exchanges," *arXiv preprint arXiv:1904.05234*, 2019.

[38] S. Eskandari, S. Moosavi, and J. Clark, "Sok: Transparent dishonesty: front-running attacks on blockchain," in *Financial Cryptography and Data Security: FC 2019 International Workshops, VOTING and WTSC, St. Kitts, St. Kitts and Nevis, February 18–22, 2019, Revised Selected Papers 23*. Springer, 2020, pp. 170–189.

[39] L. Zhou, K. Qin, and A. Gervais, "A2mm: Mitigating frontrunning, transaction reordering and consensus instability in decentralized exchanges," *arXiv preprint arXiv:2106.07371*, 2021.

[40] J. Xu and B. Livshits, "The anatomy of a cryptocurrency {Pump-and-Dump} scheme," in *28th USENIX Security Symposium (USENIX Security 19)*, 2019, pp. 1609–1625.

[41] N. Gandal, J. Hamrick, T. Moore, and T. Oberman, "Price manipulation in the bitcoin ecosystem," *Journal of Monetary Economics*, 2018.

[42] J. Kamps and B. Kleinberg, "To the moon: defining and detecting cryptocurrency pump-and-dumps," *Crime Science*, 2018.

[43] S. S. Roy, D. Das, P. Bose, C. Kruegel, G. Vigna, and S. Nilizadeh, "Demystifying nft promotion and phishing scams," *arXiv preprint arXiv:2301.09806*, 2023.

[44] J. Huang, N. He, K. Ma, J. Xiao, and H. Wang, "A deep dive into nft rug pulls," *arXiv preprint arXiv:2305.06108*, 2023.

[45] T. Sharma, R. Agarwal, and S. K. Shukla, "Understanding rug pulls: An in-depth behavioral analysis of fraudulent nft creators," *arXiv preprint arXiv:2304.07598*, 2023.

[46] M. Bartoletti, S. Carta, T. Cimoli, and R. Saia, "Dissecting ponzi schemes on ethereum: identification, analysis, and impact," *Future Generation Computer Systems*, vol. 102, pp. 259–277, 2020.

[47] W. Chen, Z. Zheng, E. C.-H. Ngai, P. Zheng, and Y. Zhou, "Exploiting blockchain data to detect smart ponzi schemes on ethereum," *IEEE Access*, vol. 7, pp. 37 575–37 586, 2019.

[48] M. La Morgia *et al.*, "A game of nfts: Characterizing nft wash trading in the ethereum blockchain," in *2023 IEEE ICDCS*.

[49] G. Bonifazi *et al.*, "Performing wash trading on nfts: Is the game worth the candle?" *Big Data and Cognitive Computing*, 2023.

[50] S. Serneels, "Detecting wash trading for nonfungible tokens," *Finance Research Letters*, vol. 52, p. 103374, 2023.

[51] D. Liu, F. Piccoli, K. Chen, A. Tang, and V. Fang, "Nft wash trading detection," *arXiv preprint arXiv:2305.01543*, 2023.

[52] X. Wen, Y. Wang, X. Yue, F. Zhu, and M. Zhu, "Nftdisk: Visual detection of wash trading in nft markets," in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, 2023.

[53] M. Ciampi *et al.*, "Fairmm: A fast and frontrunning-resistant crypto market-maker," in *International Symposium on Cyber Security, Cryptology, and Machine Learning*. Springer, 2022.

[54] 2022, "Flashbots," https://docs.flashbots.net/.

[55] Gnosis, "Gnosis protocol," https://gnosis.io/.

[56] M. M. Chakravarty *et al.*, "The extended utxo model," in *Financial Cryptography and Data Security: FC 2020 International Workshops*.

[57] Alchemy-Documentation, "What is the optimistic virtual machine (ovm)?" https://www.alchemy.com/overviews/optimistic-virtual-machine.

[58] E. Barron and H. Ishii, "The bellman equation for minimizing the maximum cost." *NONLIN. ANAL. THEORY METHODS APPLIC.*, 1989.

[59] J. Quack, "holders.at," https://holders.at/.

[60] M. O. S. Project, "mempool.space," https://mempool.space/.