Your Loss is My Gain: Low Stake Attacks on Liquid Staking Pools

Sen Yang[*,1] Aviv Yaish[*,1,2] Arthur Gervais[3] Fan Zhang[1]

[*]Equal contribution
[1]Yale University & IC3
[2]Complexity Science Hub, Vienna
[3]University College London & UC Berkeley RDI

Abstract

Permissionless Proof-of-Stake (PoS) economic security is predicated on the high cost of violating consensus safety or liveness. We show that liquid staking introduces additional risks that are not captured by standard PoS economic security arguments. Through an empirical study of Ethereum data, we find that the operational performance of liquid staking pools is positively associated with subsequent normalized liquid staking token (LST) returns. Motivated by this, we present a cross-layer attack: a low-stake adversary can manipulate the consensus protocol to degrade a target pool's performance and take application-layer positions that profit if the market reprices the corresponding liquid staking token (LST) in-line with the historically observed association.

To make the consensus layer manipulation concrete, we develop a deep reinforcement learning (DRL) framework to automatically discover attack strategies. Our evaluation shows that the learned strategies can recover near-optimal theoretical attacks and uncover new manipulation behaviors that significantly degrade target pool performance. We further characterize feasible application-layer monetization channels and analyze leveraged shorting in detail using Monte Carlo simulations, showing that such attacks can be profitable with over one-half probability for LSTs of major staking pools. Our findings reveal a previously overlooked attack surface in PoS systems with liquid staking and expose a gap between consensus and economic security.

1 Introduction

Permissionless Proof-of-stake (PoS) blockchains allow users to become validators, i.e., take part in processing transactions, by locking tokens as illiquid stake. To prevent an overproliferation of validators, blockchains can set a minimum stake amount, thus also lowering communication overhead. However, some worry that this entry barrier mays harm decentralization [12, 13]; consider that Ethereum, the most popular PoS blockchain, mandates a minimal stake of 32 ETH, currently worth over 100,000 USD. Market forces addressed this concern by (1) forming staking *pools* where users (short of the minimal amount or not) collectively stake their funds, and (2) issuing tradeable *LSTs* to pool members which can be

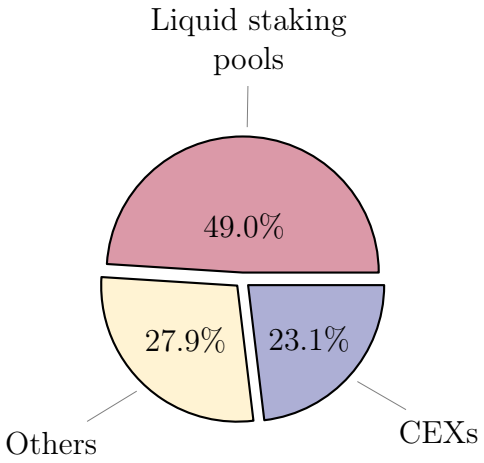


Figure 1: The distribution of stake in Ethereum by staker category, based on data from [43] as of Feb 2026, is shown. As is evident, liquid staking is the leading staking category.

redeemed for their share of the stake. Pooled and liquid staking have become Ethereum's main staking methods, consistently comprising about 50% of stake since Ethereum's transition to PoS (see Figs. 1 and 10). Given this prominence, it is natural to ask:

Do staking pools and liquid staking contribute to the security of the system?

1.1 This work

To answer this question, we present attacks that target liquid staking pools and allow an adversary with a small fraction of the stake to increase its profit. Our attacks work *even* when the underlying consensus mechanism is provably economically secure. Such mechanisms typically ensure that the cost of violating safety (honest nodes agree on the identity and order of processed transactions) and liveness (new transactions are eventually processed) is higher than the possible profits for attackers without a large fraction of stake (e.g., over 33%). Our attacks remain viable even in the face of such strong guarantees due to several key ideas: (1) Instead of violating safety or liveness to generate a profit, attacks can influence staking market dynamics to the benefit of the adversary; (2) Market dynamics are affected by pool performance: profit-maximizing users go with the best-performing pools, and the value of a pool's LSTs is positively associated with its performance, using the pool's total annual percentage rate (APR), i.e., the realized annualized return delivered to stakers; (3) Pool performance can be manipulated by adversaries inside and outside the pool.

Understanding market dynamics. To motivate our attacks, we perform the first analysis of how staking pool performance affects LST prices. Using each pool's recent staking APR, we find that better performance is consistently associated with subsequent movements in normalized LST prices across six liquid staking pools, with *moderate correlations* for most pools and *strong correlations* for Coinbase and Binance. For example, Lido shows a moderate Pearson correlation of 0.391, while Coinbase shows a strong Pearson correlation of 0.645.

Cross-layer attacks. Motivated by this observation, we analyze a class of cross-layer attacks against liquid staking pools which separate the *attack surface* from the *profit channel*: the adversary manipulates consensus-layer performance, but extracts profit at the application layer. As illustrated in Figure 2, under observed market dynamics, the adversary can monetize the attack through application-layer actions (e.g., taking a short position) that make a profit when LST prices decrease.

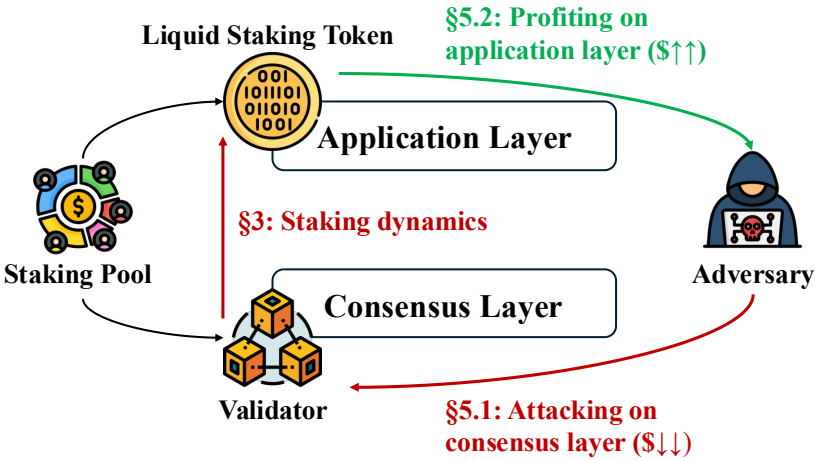


Figure 2: An overview of our attacks. The adversary first targets the consensus layer by attacking validators controlled by a liquid staking pool, which degrades the pool's performance. If the corresponding LST is repriced according to the target pool's performance (as analyzed in Section 3), the price decrease can be monetized via financial positions.

To make these attacks concrete, we first focus on a simple and realistic class of consensus-layer attacks available to a low-stakes adversary: manipulation of the PoS leader-election process [33, 3]. We develop a deep reinforcement learning (DRL) framework that automatically discovers such strategies, allowing the adversary to bias the mechanism, reduce a victim pool's realized block allocation, and thereby degrade its performance. Through carefully designed reward functions, our framework can steer the adversary to either maximize its own block allocation or, when desired, trade off its own allocation to minimize the victim's realized block allocation. We further extend the environment to model heterogeneous block values induced by Maximal Extractable Value (MEV), allowing us to study how MEV opportunity costs affect learned manipulation strategies. We use self-allocation maximization mainly as a validation baseline and focus on victim-targeted degradation.

We then turn to the application layer. Given the observed LST price relationship with pool performance, we provide a taxonomy of monetization strategies. As a representative case, we analyze leveraged shorting, which is simple to execute and supported by deep liquidity, to estimate the adversary's potential profits and resulting attack incentives.

Evaluation. We evaluate the framework across three goals.

First, as a baseline validation, we configure the reward to maximize the adversary's block allocation and show that learned policies recover known self-optimization strategies [3, 52].

Second, we change the objective to victim-targeted degradation, which differs from prior work that focused on increasing the adversary's allocation. Under this objective, the learned

policies trade-off limited adversarial loss for disproportionate victim loss. For example, an adversary with 20% stake can reduce the realized block allocation of another 20% stake pool by about 8.3%. These degradation-oriented policies also impose a more significant impact on chain quality: adversary more often sacrifice or displace blocks to harm the victim pool.

Third, we evaluate an MEV-aware extension with heterogeneous block values. We find that MEV changes the learned behavior: self-optimization policies become more conservative about sacrificing the adversary's high-value slots, while self-optimization and victim-targeted policies become more willing to fork-displace victim blocks with valuable MEV opportunities.

Finally, using Monte Carlo simulations calibrated to real-world data and parameters, we find that leveraged shorting can yield a likelihood of profit above one-half for three representative pools: Coinbase, Ether.fi, and Rocket Pool. Compared with the roughly 3% APR from honest staking, the expected profit from successful executions could be 11× higher, providing a strong incentive even when attacks are executed infrequently.

Contributions. We summarize our contributions below.

- We are the first to empirically study the relation between liquid staking pool performance and LST prices; our analysis finds consistent positive associations, with moderate correlations for several pools and a strong correlation for Coinbase and Binance.
- Building on the empirical link, we identify a class of cross-layer attacks that separate the attack surface from the profit channel: the adversary degrades a target pool's consensus-layer performance and monetizes LST price movements through application-layer strategies.
- We implement a DRL framework for discovering PoS leader-election manipulation strategies under different objectives. It recovers known self-optimization strategies as a validation baseline and identifies degradation-oriented strategies that impose disproportionate loss on a specific victim pool.
- We propose a comprehensive taxonomy of application-layer monetization channels and analyze shorting as a representative case; Monte Carlo simulations with real-world protocol parameters show that such attacks can be economically attractive.

2 Background

2.1 Terminology

Proof-of-Stake. Proof-of-Stake (PoS) blockchains rely on stake-weighted participation to maintain consensus safety and liveness [46, 22, 14, 54, 94, 91]. To participate in consensus, an entity must lock a protocol-defined amount of capital as *stake* to become a *validator*, which gives it both the right and obligation to perform consensus duties such as block proposal and attestation; in return, validators earn protocol-defined rewards for doing so correctly.

Liquid Staking. Liquid staking allows users to participate in PoS systems without directly operating validators. Instead, users deposit their stake into a liquid staking pool, which delegates the stake to a set of operators that run validators on behalf of the pool. In return, the pool issues a liquid staking token (LST) that represents a claim on the underlying staked assets and future consensus rewards. Liquid staking tokens typically follow two accounting designs: some staking pools distribute rewards via *rebasing* token balances, aiming to maintain an approximate 1:1 peg to the underlying asset (e.g., ETH), whereas others keep token balances fixed and reflect rewards through token price appreciation. For simplicity, unless stated otherwise, we use the term LST to refer to *non-rebasing* LST.

2.2 Related work

Staking pools and LSTs. As far as the authors are aware, no prior empirical work systematically studies the correlation between staking pool performance and the LST prices. Meng [50] analyzes drivers of staking inflows and withdrawals in Ethereum. Two empirical studies [85, 86] analyze leveraged staking, and price discrepancies between primary and secondary LST markets and related arbitrage frictions, respectively, while Scharnowski and Jahanshahloo [73] examine the liquid staking basis, its determinants, and its contribution to price discovery. In parallel, [38] proposes a game-theoretic model of staking pool formation that examines how reward-sharing rules shape incentives and security.

Consensus-layer manipulation in PoS. Chen and Micali [17] present the Algorand PoS protocol and bound the biasability of its leader selection mechanism. Ferreira et al. [33] construct explicit biasing attacks and quantify the adversary's advantage, with tighter bounds in [84]. Cai et al. [15] further show that, under certain timing assumptions, profitable attacks on Algorand are always detectable.

A parallel line of work analyzes the vulnerability of Ethereum's leader election mechanism to similar biasing attacks. Buterin [10, 11] observes that RANDAO-based randomness beacons allow actors to manipulate outcomes by choosing whether or not to reveal their inputs. Dworzanski [25] formalizes this as a last-mover advantage, where only the final contributor can perfectly predict the effect of its action on the beacon. Alpturer and Weinberg [3] derive optimal biasing strategies across stake levels, showing that a 20% attacker can increase its block share to 20.68%. Nagy et al. [52] further show that biasing strategies can be combined with forking, expanding the adversary's action space and increasing the achievable block share to 20.96%. Both works formulate leader-election manipulation as an explicit Markov decision process (MDP) and derive strategies for increasing the adversary's block share via dynamic programming. In this work, we instead adopt a DRL approach that supports multiple manipulation objectives, including maximizing the adversary's own block share and minimizing the victim pool's realized block share; our main focus is the latter. A broad selection of randomness beacon protocols are surveyed and compared in [18].

A complementary line of work studies strategic forking in Ethereum PoS, including ex-ante reorg attacks [56, 75, 72], balancing attacks [54, 55, 75], bouncing attacks [59], justification-withholding attacks [61], and staircase attacks [93, 49]. Most of these attacks have since been mitigated by protocol updates [94]. In principle, they could also be adapted for consensus-layer manipulation, but typically require a much larger stake share.

Pool-insider attacks. This line of work studies attacks initiated by pool participants rather than external adversaries. Rosenfeld [68] identifies a block-withholding attack against PoW mining pools, where an adversarial participant earns pool rewards by submitting "locally valid" blocks while withholding "globally valid" blocks. For staking pools, Tzinas and Zindros [77] show that a legitimate node operator can use standard participation privileges to degrade a pool's performance without changing consensus rules. Similar ideas can also support our cross-layer attacks, as discussed in Section D.

Shorting-based attacks. Prior work theoretically discussed profiting from price drops [36, 77]. We go further in several ways. First, we empirically analyze the association between pool performance and token value. Second, we provide a manipulation vector for a concrete blockchain protocol, which moreover does not require the adversary to operate the victim's node. We follow with a taxonomy of available monetization channels, and then estimate an adversary's potential profit via Monte Carlo simulations calibrated using real-world data.

3 Empirical Pool Performance-LST Price Relationship

We now empirically study the correlation between liquid staking pool performance and associated LST prices using real-world data. We focus on Ethereum, the largest PoS system with the most active liquid staking ecosystem.

3.1 Data collection

To measure staking dynamics, we collect four types of data: (1) LST token prices; (2) liquid staking pool tags; (3) the execution-layer and consensus-layer rewards of liquid staking pools; and (4) the effective balances of liquid staking pools. We collect these data from block 15,537,393 to block 22,677,606, covering the period from the Merge (September 15, 2022) [12] to June 10, 2025, which corresponds to 1,000 days under Ethereum PoS.

ETH and LST prices. We collect historical USD-denominated prices of ETH and LST tokens from CoinGecko [21]. The dataset covers daily prices of six LSTs, corresponding to six liquid staking pools: Lido, Binance, Coinbase, Ether.fi, Rocket Pool, and Mantle, which collectively include both large and smaller liquid staking pools (see Section A for details).

Liquid staking pool tags. We collect validator public keys and pool tags to map validators to liquid staking pools. We begin with a dataset that maps Ethereum validators to specific entities [43], covering 1,969,253 validators, of which 1,887,716 have pool tags. We then supplement this dataset with an address-to-pool-tag mapping based on prior work [52], which includes 65 validator addresses associated with 43 liquid staking pools.

Liquid staking pool performance. For each pool, we measure performance using its *APR*, a standard annualized measure of staking return. Intuitively, APR summarizes the realized return generated by the pool's validators over a given period, scaled by stake and annualized, and thus captures the end-to-end outcome of the pool's staking operations.

To compute APR of each liquid staking pool, we combine rewards from both the consensus layer and the execution layer. For the consensus layer, we obtain each validator's realized consensus rewards by querying a local Ethereum node via its consensus-layer API [30]. Specifically, we collect per-epoch attestation rewards, proposer rewards, and sync committee rewards for each validator and aggregate them based on liquid staking pool tags.

For the execution layer, we distinguish between blocks built through MEV-Boost [34] and blocks built locally. If a block is built through MEV-Boost, we obtain the validator's reward from the corresponding MEV-Boost relay API [35]. Otherwise, we follow prior work [90, 57] and compute the block value directly from on-chain block. We aggregate these execution-layer rewards using the validator addresses associated with each liquid staking pool.

Finally, we collect daily effective balances of validators associated with each pool from the consensus-layer API and aggregate it to obtain pools' daily effective balances. We compute a pool's daily staking return as the sum of its consensus-layer and execution-layer rewards divided by its daily effective balance, and annualize this quantity to obtain the pool's APR.

3.2 Empirical analysis

Price normalization. Since LST prices can move with market dynamics, comparing raw USD-denominated prices can mix in a common market factor. To better study the correlation between pool performance and the corresponding LST price (P_{LST}), we normalize each LST price by the ETH price (P_{ETH}). Specifically, we use $P_{\text{LST}}/P_{\text{ETH}}$ as our price measure.

Relation between performance and future returns. To assess whether staking performance is incorporated into LST valuation, we use a unified specification that relates contemporaneous performance to subsequent token returns. We focus on 180-day forward log returns rather than same-day prices, as same-day co-movement does not distinguish whether price changes follow performance or merely occur concurrently. A forward-return specification better captures whether performance is gradually reflected in market valuation.

To avoid selective sample choices, we use each pool's full available sample and apply the same specification across pools. Specifically, we report Pearson and Spearman correlations, and regressions of 180-day forward log returns on contemporaneous pool APR. Standard errors are computed using Heteroskedasticity and Autocorrelation Consistent (HAC) [4] adjustment to account for serial correlation and heteroskedasticity in daily price series.

As Table 1 shows, both Pearson and Spearman correlations are positive across all six pools, suggesting that better pool performance is generally associated with higher subsequent LST returns. The association is strongest for Coinbase, with Pearson and Spearman correlations of 0.645 and 0.708, respectively, indicating a strong positive relation under common effect-size guidelines [20]. Binance shows a relatively strong association, with correlations of 0.434 and 0.516, while the remaining pools show positive associations of varying magnitudes.

The regression coefficient β measures how strongly contemporaneous pool APR associates with future 180-day forward log returns. All regression coefficients are positive, with the largest estimates observed for Ether.fi and Coinbase. Together with the HAC standard errors and corresponding p-values, the results are suggestive of a consistent positive association between higher pool performance and higher future returns.

Table 1: Analysis results for the relationship between pool performance and LST prices. Pearson captures linear association, and Spearman captures rank-based monotone association. Correlations of 0.3-0.5 are moderate, and those above 0.5 are strong [20]. Larger β values indicate stronger return sensitivity to APR. Smaller HAC Standard Errors (HACSE) relative to β indicate more precise estimates. Smaller p-values indicate stronger evidence.

Pool	Pearson	Spearman	β	HAC SE	p-value
Lido	0.391	0.387	0.1049	0.0204	$<$0.0001
Binance	0.434	0.516	0.1738	0.0270	$<$0.0001
Coinbase	0.645	0.708	0.8719	0.1359	$<$0.0001
Ether.fi	0.407	0.302	1.7247	0.7692	0.0250
Rocket Pool	0.176	0.260	0.1142	0.0407	0.0050
Mantle	0.112	0.315	0.1151	0.0550	0.0365

Taken together, these results show that LST prices and pool performance are meaningfully linked, providing an empirical foundation for our subsequent analysis of cross-layer attacks. Robustness checks with 90-day and 60-day horizons yield qualitatively similar positive relations across pools, as reported in Appendix B.

4 Model

We now formally define our model. Notations are summarized in Appendix C.

We consider a PoS consensus mechanism $\mathcal{M}$ with a validator set $\mathcal{V}$, which is partitioned into two disjoint subsets: a strategic adversary $\mathcal{A}$ and a set of honest validators $\mathcal{H}$, such that $\mathcal{V} = \mathcal{A} \cup \mathcal{H}$. The adversary $\mathcal{A}$ controls cumulative relative stake $\alpha_{\mathcal{A}} \in (0, 1)$, while honest validators $\mathcal{H}$ possess the remaining stake $1 - \alpha_{\mathcal{A}}$. As with previous work, stake distribution is constant within the scope of an attack. Finally, we consider adversaries with sufficient capital to execute application-layer financial strategies, e.g., take short positions.

The mechanism $\mathcal{M}$ is defined by two components that govern the evolution of the chain:

- **Leader Election ($\mathcal{M}_L$):** This component defines the proposer selection process. Ideally, a validator $v \in \mathcal{V}$ is selected with probability proportional to its stake; intuitively, it determines *who is allowed to propose a block*.
- **Fork Choice Rule ($\mathcal{M}_F$):** This determines the blockchain's canonical head by resolving potential conflicts between multiple branches according to a rule that aggregates validators' staking-weighted signals; intuitively, it determines *which proposed block becomes canonical*.

Example: Ethereum. Take Ethereum as an example. The leader election component $\mathcal{M}_L$ in Ethereum is driven by a randomness beacon based on RANDAO [27]. Time in Ethereum is divided into epochs, each consisting of 32 slots. In each slot, a single validator is selected as the block proposer, and each valid proposed block contains a 96-byte value, called `randao_reveal`. This value is contributed by the proposer and incorporated into a running randomness mix. The mix is updated incrementally across slots and acts as a

source of randomness for the protocol. The RANDAO mix at the end of an epoch e is then used as an input for proposer selection and committee assignment in subsequent epochs, together with validators' effective stake balances. Strictly speaking, this affects epoch $e+2$ in Ethereum, but for simplicity we refer to it as affecting epoch $e+1$ in the remainder of the paper; this convention does not alter any of our results.

The fork choice rule $\mathcal{M}_F$ in Ethereum is implemented by LMD-GHOST [14]. Validators issue attestations that vote for blocks, and the canonical head of the chain is determined as the block (together with its ancestor chain) that accumulates the largest staking-weighted attestations, considering only the latest message from each validator. To prevent ex-ante reorg attacks [75], proposer boost p_{boost} is introduced to temporarily increase the effective weight of the proposer's block during fork choice, ensuring that a newly proposed block is favored over competing blocks that have accumulated adversarial attestations. In the current specification, p_{boost} is set to 40% of the total effective stake balance.

5 Our Attacks

Liquid staking pools span both the consensus layer, where their validators earn rewards through block proposals, and the application layer, where their LSTs are traded and used as financial assets. This structure allows an adversary to attack one layer while monetizing the effect in another, making cross-layer attacks a general class of vulnerabilities in liquid staking pools. In this section, we show how such attacks can be constructed by combining consensus-layer manipulation policies with application-layer financial positions.

5.1 Consensus-layer manipulation

5.1.1 Manipulation landscape

As discussed in Section 2.2, an adversary can leverage various approaches to degrade the target pool's performance. We summarize these approaches and their corresponding adversary stake requirements in Table 2.

Table 2: Stake requirements for different strategies.

Attack strategy	**Adversary stakes**
Leader election manipulation [3, 52]	Low-High
Reorganization attacks [55, 93]	Medium-High
Insider attacks [68, 77]	Low
Network attacks [40]	Zero

While all these strategies can degrade the victim pool's performance, for example, reorganization attacks can invalidate proposed blocks or attestations via strategic forking, they typically have different requirements on the adversary's stake share and guarantees. Reorganization attacks can be effective, but they often require a relatively large stake share to succeed. By contrast, insider and network-layer attacks are harder to quantify and lack clear performance guarantees. The former involves an adversary sabotaging the pool from within

as a participant, whereas the latter relies on disrupting connectivity to hinder operational correctness. We provide more details on these attack vectors in Section D.

In this section, we focus on consensus-layer manipulation via leader election, because prior work shows that even low-stakes adversaries can bias leader election [3, 52], and such manipulation has more predictable implications.

5.1.2 Straw-man solution and limitation

Manipulating the leader election mechanism allows the adversary to degrade a target liquid staking pool's performance by reducing its realized share of block proposals relative to its ideal, stake-proportional allocation.

A natural approach to modeling such manipulations is by casting the leader election and fork choice process as an MDP. Formally, an MDP is a tuple (S, A, P, R), where S and A denote the states and actions available to the agent, while P and R define the transition model and rewards. Specifically, the probability of transitioning from state $s \in S$ to state $s' \in S$ when taking action $a \in A$ is given by $P_a(s, s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$, and the agent's reward for the transition is given by $R_a(s, s')$. A policy $\pi : S \to A$ is a function that specifies the action taken by the agent in each state.

Previous research [3, 52] has modeled manipulation strategies targeting Ethereum's leader election and fork choice rules as an MDP, with this approach being standard for consensus attacks [32, 39, 89, 88, 7]. In this context, the adversary seeks to derive a policy that maximizes its long-term expected utility, specifically by augmenting its relative share of block proposals at the expense of honest participants. This can be performed through the application of classical techniques, such as policy or value iteration [62].

However, this approach also has a well-known limitation: the explosion of state space [44]. To model the attack accurately, a state $s \in \mathcal{S}$ must at least capture the 256-bit RANDAO mix, the proposer allocation for the current and look-ahead epochs, and the relative fork status (i.e., private vs. public chain heights). In particular, [52] shows that even in a single-adversary setting, the state space can grow rapidly, making an explicit MDP formulation infeasible. This limitation motivates us to adopt a learning-based approach that does not require explicit enumeration of the state space. In particular, prior work has shown that deep reinforcement learning can be effective in studying selfish mining strategies [44, 5, 71], suggesting its suitability for problems with a similar structure.

5.1.3 Deep reinforcement learning

In this paper, we adopt a DRL approach to study consensus-layer manipulation against a target liquid staking pool. Our DRL formulation differs from prior work in two ways. Methodologically, it avoids explicit state-space enumeration and instead learns policies directly from interaction with the simulated protocol environment. Substantively, it optimizes for targeted degradation of a victim pool's realized performance, rather than only for the adversary's own proposer gains.

Figure 3 presents the overview of our DRL framework. First, according to the consensus $\mathcal{M}$, we build an environment that simulates the execution of its leader election and fork choice components. Then, the adversary is modeled as an RL agent that controls only its

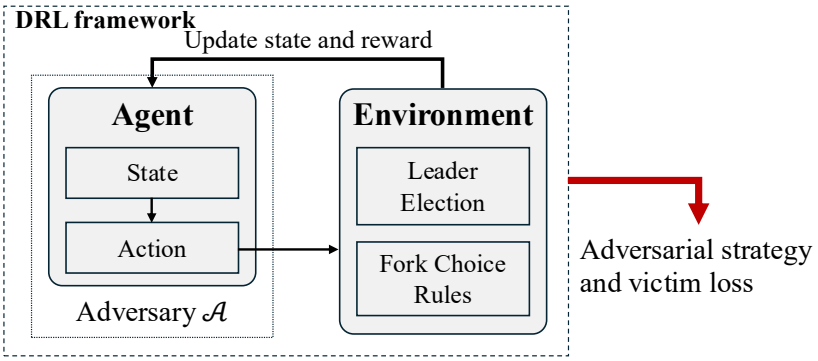


Figure 3: DRL framework for automatic discovery of consensus-layer adversarial strategies.

own validator actions, while honest validators follow the consensus mechanism $\mathcal{M}$ and act according to the protocol without learning or adaptation. At each slot, the agent observes the system state, takes an action, and receives a reward that reflects the effect of its strategy on both adversarial gain and honest-validator loss under protocol-compliant execution.

Environment. The environment serves as a formal abstraction of the consensus layer, specifically isolating the operational logic of the leader election mechanism and the fork choice rule (i.e., LMD-GHOST). To maintain computational tractability, we assume a synchronous network with instantaneous broadcasting. Within this framework, the environment dictates the system's evolution: it assigns block proposers for each slot and evaluates the canonical chain according to the prescribed fork choice rule. The environment acts as the reactive entity that processes the adversary's actions to determine the next state and the corresponding reward R, reflecting the collective protocol response under these idealized conditions.

Action space A. For Ethereum, we follow prior work [52] to define a discrete action space A, which consists of the following actions in a given slot. (1) **Propose ($a_{\textbf{prop}}$):** the agent honestly publishes a valid block extending the current public canonical head. (2) **Miss ($a_{\textbf{miss}}$):** the agent intentionally skips its assigned proposal, thus not contributing entropy to the beacon. (3) **Hide ($a_{\textbf{hide}}$):** the agent builds a block privately without broadcasting it to the network, withholding its entropy from public view. (4) **Fork ($a_{\textbf{fork}}$):** the agent broadcasts a previously hidden private chain to trigger a reorganization (reorg), which has two effects: excluding honest blocks (particularly those by the target pool $\mathcal{T}$) from the canonical chain, and altering their entropy contributions to the beacon. (5) **Regret ($a_{\textbf{regret}}$):** the agent abandons its private fork and resynchronizes with the public canonical chain.

While the protocol allows for these actions, we employ action consolidation to a compact space $A \in \{0, 1, 2\}$. Specifically, $a = 0$ corresponds to a_{miss}, in which the agent intentionally skips its block proposal. Action $a = 1$ represents honest behavior: when no private chain exists, it maps to a_{prop}; otherwise, it maps to a_{regret}, in which the adversary abandons the private chain and proposes honestly. Action $a = 2$ corresponds to private chain manipulation: when no private chain exists, it maps to a_{hide}; when one exists and a competing honest block is observed, it maps to a_{fork}, initiating an ex-ante reorganization to displace the honest block.

This reduction follows from protocol feasibility constraints and the timing of meaningful

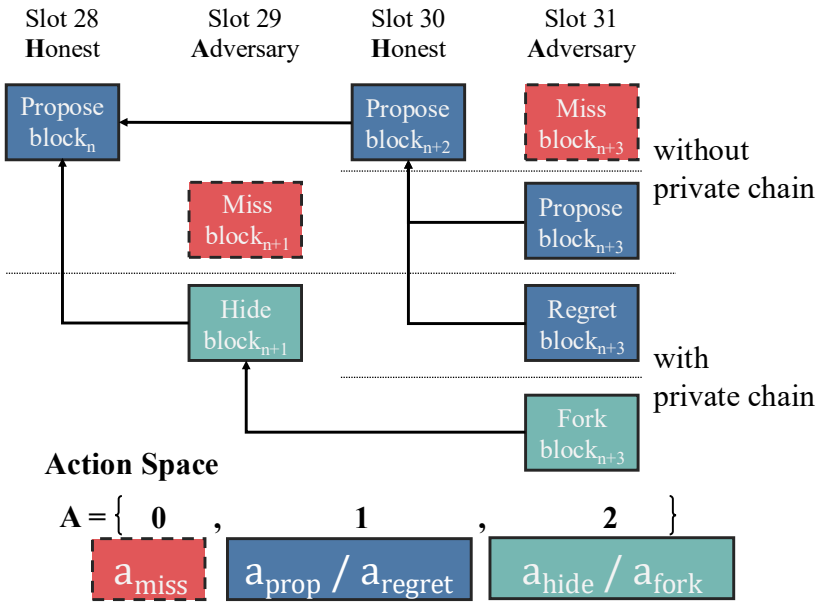


Figure 4: The action space of the adversary.

adversarial decisions. Once the adversary initiates a private chain, it must repeatedly execute the action a_{hide} to accumulate sufficient attestation weight; deviating earlier is infeasible if a reorganization is to succeed. As a result, the adversary's only non-trivial decision point arises after a private chain exists and an honest validator publishes a competing block, at which moment the adversary chooses whether to adopt the action a_{fork} or a_{regret}.

State representation and observation design. To make learning tractable, we transform the high-dimensional environment state S into a compact, 7-dimensional observation vector $\phi(s) \in \mathcal{O}$, which can be categorized into three functional groups:

- *Branch:* A binary indicator signaling the presence of a withheld private chain, which defines the agent's permitted actions by determining the feasibility of a_{fork} or a_{regret}.
- *Counterfactual strategic oracles:* A key difficulty in leader manipulation is that rewards are sparse and delayed: the effect of an action is only realized in the block allocation of the subsequent epoch. To provide the agent with denser, action-specific feedback, we augment the observation space with counterfactual oracles. An oracle helps the adversary estimate the expected block allocation in the next epoch under different hypothetical actions at the current slot. To make these estimates tractable, when evaluating a hypothetical action at a given slot, we assume that all subsequent adversarial slots in the remaining tail of the current epoch follow the default honest strategy (i.e., propose the block).

  Concretely, we define three oracles $\{\Omega_{prop}, \Omega_{skip}, \Omega_{fork}\}$. Each oracle returns two numbers: the expected block allocation for the adversary $\mathcal{A}$ and the target pool $\mathcal{T}$ in epoch $e{+}1$ under the corresponding hypothetical action, denoted as $\mathbb{E}[B_{e+1}^{(\mathcal{A})}]$ and $\mathbb{E}[B_{e+1}^{(\mathcal{T})}]$), respectively.

Reward R. The design of the reward function R is central, as it must capture the adversary's strategic objective beyond simple profit maximization. Our formulation differs from

prior work [3, 52] in that the agent is not trained merely to maximize its own block share, but to induce a long-term degradation of the target pool $\mathcal{T}$'s performance. We therefore construct a composite reward function $\mathcal{R}$ evaluated at epoch boundaries, ensuring that protocol causalities, especially the finality of the RANDAO mix, are respected:

$$R = \beta \cdot \left(\mathbb{E}[B_{e+1}^{(\mathcal{A})}] - B_{\text{loss},e}^{(\mathcal{A})} \right) - \omega \cdot \left(\mathbb{E}[B_{e+1}^{(\mathcal{T})}] - B_{\text{loss},e}^{(\mathcal{T})} \right) + \gamma \cdot L_{\text{tail}}^{e+1}.$$

Here, $\mathbb{E}[B_{e+1}^{(\mathcal{A})}]$ and $\mathbb{E}[B_{e+1}^{(\mathcal{T})}]$ denote the expected number of slots assigned to the adversary and the target pool $\mathcal{T}$ in the subsequent epoch, respectively. The loss terms $B_{\text{loss},e}^{(\mathcal{A})}$ and $B_{\text{loss},e}^{(\mathcal{T})}$ capture forfeited blocks caused by withholding decisions in epoch e, while L_{tail}^{e+1} measures whether the adversary gains influence over the tail of the next epoch.

The weighting coefficients $\{\beta, \omega, \gamma\}$ allow the framework to represent different objectives. By tuning these parameters, we can recover classical attacks or define new ones. When $\omega = 0$, the reward reduces to the adversary's own expected value accumulation, aligning with classical selfish-mixing objectives. Conversely, setting $\beta = 0$ yields a griefing adversary that ignores its own economic losses and acts solely to suppress the target pool $\mathcal{T}$ while preserving strategic continuity. Under this profile, the adversary is willing to sacrifice all of its own block value in order to minimize the target pool's performance.

5.1.4 MEV-aware extension

Prior analytical models usually assume homogeneous block values. To capture high-value opportunities arising from mempool fluctuations and MEV, we extend the environment with a reduced-form shared-opportunity model inspired by prior work [5, 7].

Specifically, we maintain a bounded system-wide pool $P_s \in \{0, \ldots, C-1\}$ of pending high-value opportunities at slot s. At each slot, a new opportunity arrives independently with probability δ and is added to the pool up to capacity C:

$$P_{s+1} = \min(P_s + X_s, C), \qquad X_s \sim \text{Bernoulli}(\delta).$$

Each block can consume at most one pending opportunity. Given $F > 0$ denotes the additional value of a high-MEV opportunity, a canonical block's realized value is

$$v_s = \begin{cases} 1 + F, & P_s > 0, \\ 1, & P_s = 0, \end{cases}$$

After consumption, the pool decreases by one whenever $P_s > 0$. If a high-value block is displaced by the fork choice rule, the corresponding opportunity is returned to the pool and may be inherited by the next canonical block. This captures the fact that many high-value opportunities persist briefly after displacement and can still be captured by later canonical proposers, rather than being exhausted by the first proposer alone.

State representation. In the MEV-aware extension, we augment the observation with value-aware state variables: the realized value carried by the adversary's withheld private branch $V_s^{(\text{priv})}$, the realized value of the competing public segment $V_s^{(\text{pub})}$, and the current-slot value $V_s^{(\text{cur})}$ (i.e., the value obtainable if the current slot were made canonical immediately).

As branches evolve, adversarial withheld blocks contribute to $V_s^{(\text{priv})}$, while newly appended honest blocks contribute to $V_s^{(\text{pub})}$ for as long as the corresponding public segment remains contestable. This reflects our Ethereum-specific setting, where the adversary withholds first and forks later, while future slot values remain unobserved at decision time.

Reward adaptation. We incorporate these value-aware states into the reward through the net realized value effect on each party within the current fork window. Let $\Delta V_e^{(\mathcal{A})}$ and $\Delta V_e^{(\mathcal{T})}$ denote the net value gained or lost by the adversary and the target pool, respectively, within the current epoch. These terms capture both positive value retained through successful fork outcomes, including MEV opportunities that are realized by displacing competing blocks, and negative value lost due to sacrifice, failed withholding, reversion, or displacement from the canonical chain. Because future high-value opportunities are unknown at decision time, the next-epoch allocation terms remain expectation-based. Thus, the reward is defined as

$$R = \beta \cdot \left(\mathbb{E}[B_{e+1}^{(\mathcal{A})}] + \Delta V_e^{(\mathcal{A})}\right) - \omega \cdot \left(\mathbb{E}[B_{e+1}^{(\mathcal{T})}] + \Delta V_e^{(\mathcal{T})}\right) + \gamma \cdot L_{\text{tail}}^{e+1}.$$

This preserves next-epoch allocation effects while making current fork decisions value-aware.

Strategy discovery. Overall, our DRL framework allows the adversary to learn different manipulation strategies, ranging from maximizing its own slot allocation to minimizing the victim pool's slot allocation. We evaluate these behaviors in Section 6.1.

5.2 Application-layer manipulation

The empirical relation between pool performance and LST valuation creates several possible application-layer profit mechanisms for an adversary. At a high level, the adversary benefits from private knowledge about the timing and direction of a performance-induced price movement: it can open directional positions before the market fully incorporates the degradation, or compete for downstream opportunities that arise once prices and oracle values adjust.

We group the adversary's application-layer monetization strategies into four classes. First, the adversary can short the target LST through lending markets, leveraged credit accounts, or derivatives. Second, it can use prediction markets when contracts exist for pool-specific or token-price events. Third, if the price drop pushes existing collateralized positions below liquidation thresholds, the adversary can compete to capture liquidation opportunities. Fourth, it can arbitrage temporary price differences across different exchanges as the LST price adjusts. These strategies differ in capital requirements, execution risk, and exposure to MEV competition [92]; we provide a detailed taxonomy in Section E.

In the remainder of the paper, we use *leveraged shorting* through lending protocols as a representative monetization channel. This choice is conservative in the sense that it does not require winning competition for a one-block MEV opportunity, unlike liquidation or atomic arbitrage, and it exposes the adversary to explicit frictions such as liquidation risk, slippage, and borrowing cost. Concretely, the adversary deposits neutral collateral C, borrows and sells the target LST, and later repurchases the LST to repay the debt. With effective loan-to-value (LTV) ratio ρ and m recursive leverage rounds, this creates a notional short exposure of $N = \rho C \cdot \frac{1-\rho^{m+1}}{1-\rho}$. Ignoring frictions, a normalized price drop Δ_p yields gross

payoff proportional to $N\Delta_p$. Our evaluation in Section 6.2 then adds frictions that determine whether this exposure is profitable: market noise, slippage, liquidation risk, and borrowing costs. The detailed lending-based short construction is deferred to Section E.2.

6 Evaluation Results

In this section, we evaluate the effectiveness of our proposed attacks. To avoid any impact on real-world systems, all evaluations are conducted in a simulated environment.

6.1 Effectiveness of DRL framework

To evaluate whether our DRL framework can automatically discover effective leader-election manipulation strategies, we design the following research questions.

- **RQ1:** Can our framework recover theoretically optimal strategies identified in prior work?
- **RQ2:** How do adversarial profiles ranging from pure profit-seeking to pure griefing impact the performance of the target liquid staking pool?
- **RQ3:** How does MEV affect the manipulation strategies learned by our framework?

Throughout the evaluation, we focus on sub-$\frac{1}{3}$ stake adversaries. This is motivated by two considerations. First, the adversary with higher stake shares can launch catastrophic consensus-level attacks [29]. Second, the current Ethereum staking distribution [43] shows that the largest staking entity, Lido, controls less than 30% of the total stake.

Implementation. We built a simulated Ethereum environment using Gymnasium [76] to model Ethereum's leader election mechanism and fork choice rules. Specifically, it includes three validator groups: the attacker, the victim, and the remaining honest validators, parameterized by their stake shares. Proposer schedules are deterministically derived from the RANDAO mix using a hash-based pseudorandom process with stake-weighted sampling.

We train the attacker policy using Maskable PPO, a variant of Proximal Policy Optimization (PPO) [74, 45, 65]. We use Maskable PPO because our environment contains state-dependent invalid actions, and action masking allows the policy to respect protocol constraints during training while improving sample efficiency. For each adversary-victim stake configuration, we train a separate policy on episodes generated from epoch-level proposer allocations derived from independent random seeds. Each training run lasts up to 10^6 timesteps, and performance is evaluated through repeated simulation runs. More implementation details and hyperparameter settings are deferred to Section F.

6.1.1 RQ1: Discovering optimal strategy

To answer RQ1, we set $\omega = 0$ so that our DRL framework prioritizes the adversary's block share, disable the MEV-aware extension and assume homogeneous block values, aligning with self-optimization strategies studied previously [3, 52].

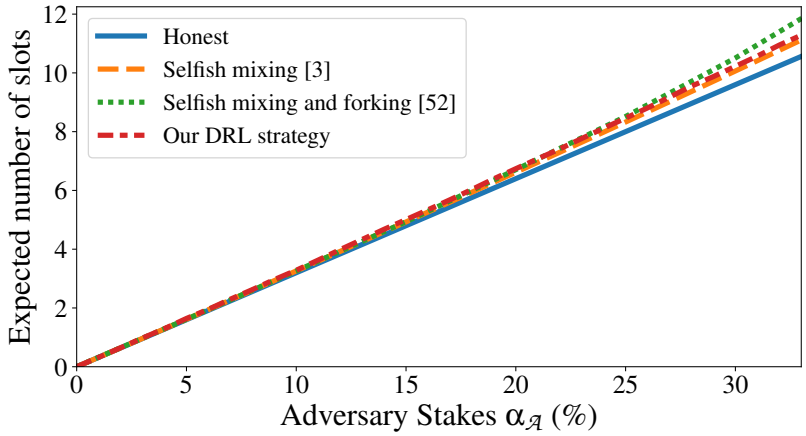


Figure 5: Expected adversarial slots under different strategies and adversary stake.

To determine the best weighting coefficients β, γ, we perform a grid search over an 8×8 parameter grid. For each coefficient, we consider the values 0.1, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, and 5.0. We set the adversary's stake share to 33.3% to calibrate the weighting coefficients. After the grid search, we find that when $\beta = 1.0, \gamma = 0.5$, the adversary achieves the highest block share, measured as the average block share over 10,000 simulation runs after training.

We then vary the adversary's stake share from 1% to 33% in steps of 1%. For each stake share, we train five DRL models with different random seeds and evaluate each learned strategy over 10,000 independent randomized simulations. We measure effectiveness using the same metric as before, namely the average number of slots assigned to the adversary in the next epoch, and report the mean across the five training seeds.

As Figure 5 shows, we compare the effectiveness of different strategies: the honest strategy, the selfish-mixing strategy of [3] (where the adversary only chooses between honest proposing and deliberately missing), and the selfish-mixing-and-forking strategy of [52] (which additionally allows forking). The strategy learned by our DRL framework consistently outperforms selfish-mixing at all adversary stake levels. Notably, when the stake is below 25%, it performs comparably to the selfish-mixing-and-forking strategy, suggesting that our framework can automatically recover near-optimal strategies from prior work.

We observe a small performance gap between our learned strategy and the analytically optimal strategy when the adversary's stake share exceeds 25%. This is expected: prior work gives theoretical optima under idealized modeling assumptions, whereas our DRL framework approximates optimal behavior through finite training in a stochastic environment, with a larger decision space at higher stake shares. Since this experiment aims to validate whether our framework can automatically discover effective strategies rather than exactly reproduce the closed-form optimum, this small gap does not affect our main conclusion.

Answer to RQ-1

Our DRL framework can automatically recover strategies that maximize the adversary's own block allocation, with performance that remains very close to the theoretical optimum established in prior work.

6.1.2 RQ2: Pure-griefing strategy discovery

Having validated the efficacy of our DRL framework in replicating theoretically optimal strategies (RQ1), we now focus on its capacity for autonomous strategy discovery in a different scenario. Specifically, whether it can automatically discover effective strategies against a particular liquid staking pool, hereafter referred to as the *victim*. To isolate this strategic effect, we continue to assume homogeneous block values and disable MEV-aware extension.

To bias our DRL framework toward "pure griefing," we set $\beta = 0$ and perform the same 8×8 grid search to determine the optimal weighting coefficients ω and γ. Specifically, we fix the adversary's and victim's stake shares at $\alpha_{\mathcal{A}} = 20\%$ and $\alpha_{\mathcal{T}} = 20\%$, respectively, to calibrate these coefficients. The grid search shows that $\omega = 5.0$ and $\gamma = 1.5$ minimize the victim's average number of slots assigned in the subsequent epoch.

Upon calibrating the coefficients, we extend our evaluation to a broad range of stake distributions that reflect current liquid staking pools. Specifically, we vary both the adversary's and the victim's stake shares from 5% to 30% in 5% increments. This range is chosen to approximate the current distribution of major staking entities, such as Lido, which controls nearly 30% of all stake, and Ether.fi, which controls about 5% of the total stake.

For each stake configuration, we train a separate DRL agent under the calibrated reward setting. We follow the same training protocol as in RQ1, using five independent training seeds per configuration. For each seed, we evaluate the learned policy over 10,000 independent randomized simulations and report the mean of the resulting metrics across seeds.

Victim loss. To quantify the adversary's impact, we use two complementary metrics: *victim loss* and *adversary loss*, defined as

$$\textit{victim loss} = \frac{B_{e+1}^{(\mathcal{T})} - 32 \cdot \alpha_{\mathcal{T}}}{32 \cdot \alpha_{\mathcal{T}}}, \qquad \textit{adversary loss} = \frac{B_{e+1}^{(\mathcal{A})} - 32 \cdot \alpha_{\mathcal{A}}}{32 \cdot \alpha_{\mathcal{A}}}.$$

These metrics measure the reduction in victim and adversary slot allocations relative to their respective ideal allocations. Note that lower values indicate larger reductions in allocation.

Figure 6 summarizes attack outcomes across different adversary and victim stake shares. Adversary loss is often small and negative, indicating a strategic trade-off in which the attacker incurs limited cost to amplify the victim's performance degradation. Notably, victim loss consistently exceeds adversary loss across nearly all stake combinations.

For example, when $\alpha_{\mathcal{A}} = 10\%$ and $\alpha_{\mathcal{T}} = 30\%$, the adversary incurs only a 2.6% loss while causing a 2.9% loss for the victim. This shows that even a relatively small adversary can use limited self-sacrifice to amplify the victim's performance degradation. The asymmetry is even more pronounced when both the adversary and the victim control 30% of the stake. In this case, the adversary suffers only a 0.4% loss, whereas the victim suffers a 10.8% loss.

Another observation we can make from Figure 6 is that the victim loss generally decreases as the victim's stake increases; however, this does not mean the attack becomes less effective. Because larger staking pools control more slots, even minor relative losses result in significant absolute damage. For instance, a fivefold increase in victim stake (from 5% to 25%) only yields a one-third reduction in relative loss, falling short of a proportional one-fifth improvement. This suggests that large liquid staking pools remain vulnerable to disproportionate absolute damage from this strategy.

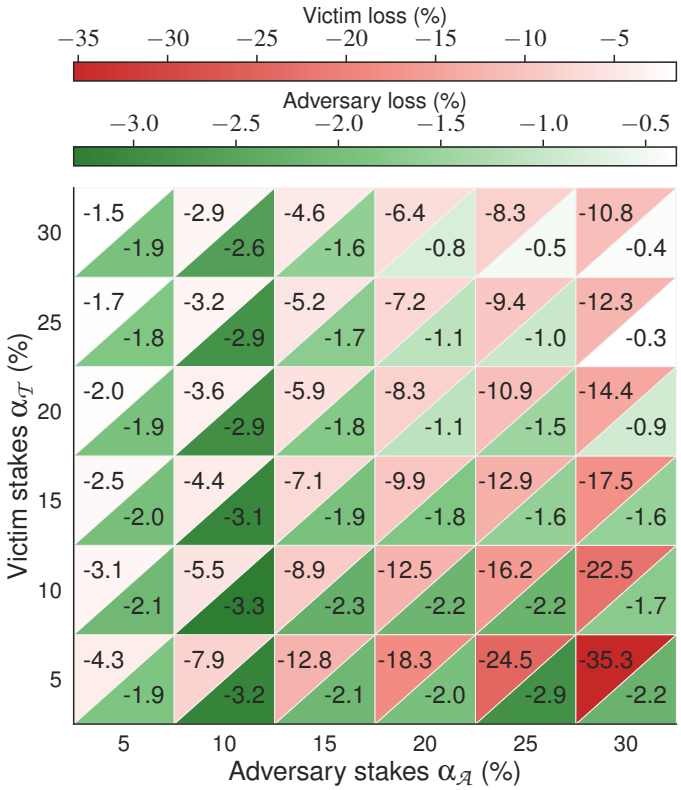


Figure 6: A heatmap of the impact of learned pure griefing strategy under varying stake distributions. Each cell is divided into two triangles: the upper-left triangle (red) shows the victim's relative slot loss, while the lower-right triangle (green) shows the adversary's relative slot loss (or increase). More negative values are worse for the victim (red), while less negative values are better for the adversary (green).

Chain quality impact. Pure griefing can be more damaging to chain quality than self-optimization because the adversary may accept additional block loss to reduce the victim pool's performance. We measure this effect using *chain-quality impact*: the percentage of epoch slots in which an otherwise-canonical block is either sacrificed by the adversary or displaced by an adversarial fork. Higher values indicate stronger degradation of canonical-chain utilization: more slots are wasted or replaced by the attack, reducing throughput and weakening chain-quality guarantees.

Figure 7 shows that pure-griefing strategies consistently cause higher chain-quality impact than self-optimization. Under this objective, the learned strategy leads to a clearer degradation in chain utilization: with 30% stake, more than 1.7% of epoch slots are sacrificed or displaced by the adversary. The effect is slightly stronger for smaller victim pools, but this variation is modest compared with the gap from the self-optimization baseline.

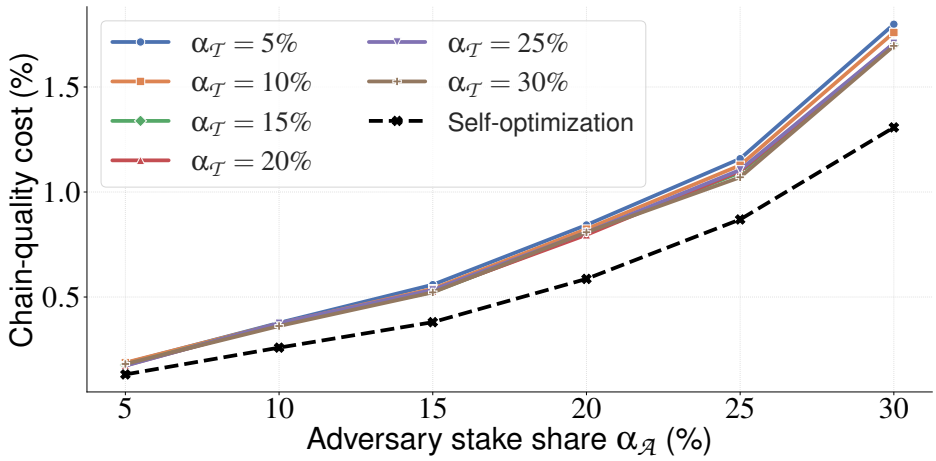


Figure 7: Chain-quality impact of pure-griefing strategies.

Answer to **RQ-2**

Our DRL framework can also discover pure-griefing strategies that cause disproportionate victim loss with limited adversarial self-loss, while also imposing a larger chain-quality cost than self-optimization.

6.1.3 RQ3: Effect of MEV

RQ2 shows that, under homogeneous block values, our DRL framework can automatically discover effective griefing strategies. We now ask whether the learned strategies differ once block values become heterogeneous due to MEV.

To study this effect, we fix a representative adversary-victim stake pair with $\alpha_{\mathcal{A}} = \alpha_{\mathcal{T}} = 20\%$ and vary the frequency and magnitude of MEV opportunities. This allows us to examine whether MEV shifts the policy from pure allocation manipulation toward value-aware behavior, and how this shift affects adversary cost, victim loss, and chain-quality cost.

MEV settings. We instantiate the MEV-aware extension by varying two parameters: the arrival probability δ, which controls how often high-value opportunities enter the pool, and the bonus value F, which controls the additional value of a high-MEV block. We sweep $\delta \in \{0.05, 0.10, 0.15, 0.20\}$ and $F \in \{1, 2, 5, 10\}$, and fix $C = 3$ as in prior work [5].

For comparability with RQ1 and RQ2, for each (δ, F) choice, we train separate DRL agents using the calibrated self-optimization and pure-griefing reward weights from RQ1 and RQ2, and evaluate learned policies over the same number of independent simulations.

Results. We first compare the learned policies with the RQ1 and RQ2 baselines to check whether our framework recovers effective self-optimization and pure-griefing strategies under heterogeneous block values. Due to space limits, we report the detailed validation results in Section F.4. The takeaway is that the learned policies achieve adversary value gains and victim losses comparable to the corresponding RQ1 and RQ2 baselines, suggesting that the framework learns effective strategies in the MEV-aware setting.

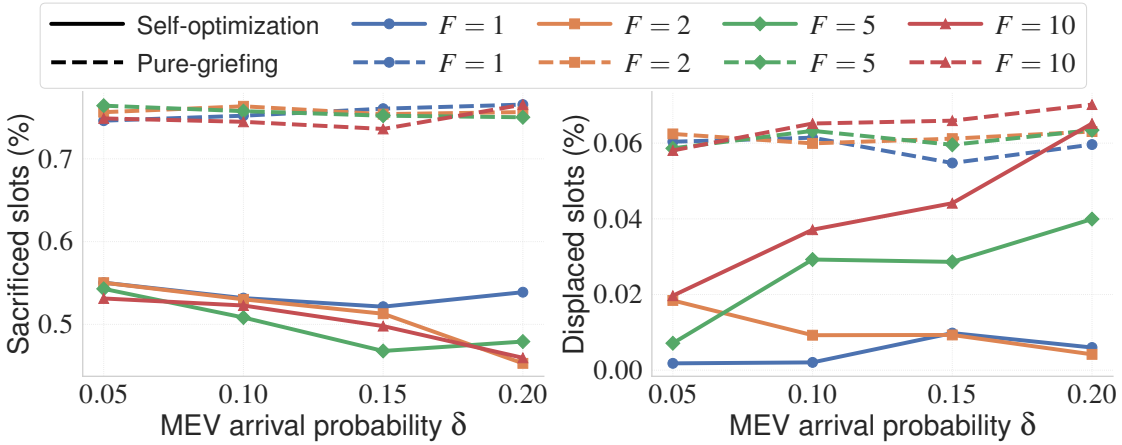


Figure 8: Effect of MEV on the chain-quality cost of learned policies. The left panel reports sacrificed slots, and the right panel reports fork-displaced slots.

We next examine how MEV shifts the learned policy and how this affects chain quality. Figure 8 decomposes chain-quality impact into two sources: sacrificed slots and fork-displaced slots. The two objectives respond differently to MEV. For sacrificed slots, pure-griefing policies remain nearly unchanged across tested MEV settings: the dashed lines stay around the same level as δ and F vary. This is consistent with the objective of pure griefing, where the agent is rewarded for harming victim performance rather than preserving its own high-value proposal opportunities. In contrast, self-optimization policies become more conservative when MEV opportunities are more valuable or frequent. As F and δ increase, the sacrificed-slot fraction generally decreases, indicating that the agent gives up fewer proposal opportunities when those opportunities are more likely to carry high MEV value.

The pattern for fork-displaced slots is different. Displacing the victim's block can be attractive under both objectives: it can both increase the adversary's own payoff by recapturing the MEV opportunity in the displaced block and reduce the victim's rewards. Therefore, the displaced-slot fraction tends to rise as MEV becomes stronger, that is, as δ or F increases. This increase is visible under both self-optimization and pure-griefing objectives, suggesting that MEV strengthens the incentive to fork out the victim's blocks regardless of whether the reward emphasizes the adversary's own gains or the victim's losses.

Taken together, these two components suggest that MEV has different chain-quality impacts across objectives. Because sacrificed slots account for a large share of the chain-quality impact, stronger MEV can make self-optimization policies less harmful to chain quality by discouraging the adversary from giving up its own high-value proposal opportunities. Under pure griefing, this self-preservation effect is absent, so stronger MEV creates additional fork opportunities that can slightly increase the overall chain-quality impact.

Answer to **RQ-3**

MEV makes learned policies value-aware: self-optimization policies sacrifice fewer high-value attacker slots, while both objectives fork-displace more victim blocks. Consequently, stronger MEV can reduce chain-quality impact under self-optimization but slightly increase it under pure griefing.

6.2 Application-layer profit estimation

Following prior work [66, 53, 44, 79, 16, 6, 5], we adopt Monte Carlo-based simulations to estimate the adversary's potential profits via leveraged short positions. Our goal is not to predict exact real-world profits, but to quantify whether the empirically estimated relation between pool performance and LST price movements creates economic incentives. We further use the simulation to assess how robust these incentives remain under realistic trading frictions and alternative parameter settings.

Simulation framework. Let δ^{APR} denote the attack-induced APR-equivalent degradation of the target pool. Let H denote the return horizon, and let $\hat{\beta}_H$ be the regression coefficient at that horizon. Because the empirical specification in Section 3 relates pool APR to forward log returns, we model the induced log-return shock as

$$z_H = -\hat{\beta}_H \delta^{\mathrm{APR}} + \varepsilon_H, \qquad \varepsilon_H \sim \mathcal{N}(0, \sigma_H^2),$$

where σ_H is calibrated from the residual variation in the same regression. We then convert this log-return shock into a normalized LST price drop, $\Delta_p = 1 - \exp(z_H)$, which for small deviations gives $\Delta_p \approx \hat{\beta}_H \delta^{\mathrm{APR}} - \varepsilon_H$. Following Section 5.2, the adversary constructs a leveraged short position through a lending protocol. Let C denote the initial collateral (in ETH), ρ the effective LTV ratio, and m the number of leverage amplification rounds. The resulting short exposure is $N = \rho C(1 - \rho^{m+1})/(1 - \rho)$.

To account for finite market liquidity, we distinguish between the mid-price move implied by Δ_p and the execution prices at entry and exit. Let P_0 denote the pre-attack LST price and $P_1 = P_0(1-\Delta_p)$ the post-attack mid price. If the adversary shorts $q = N/P_0$ units of the LST, the realized sell and buy-back prices are $P_{\mathrm{sell}} = P_0(1 - s_{\mathrm{sell}}(q))$ and $P_{\mathrm{buy}} = P_1(1 + s_{\mathrm{buy}}(q))$, where $s_{\mathrm{sell}}(q)$ and $s_{\mathrm{buy}}(q)$ are proportional slippage functions that depend on trade size and market depth. The gross trading profit is $\Pi_{\mathrm{trade}} = q(P_{\mathrm{sell}} - P_{\mathrm{buy}})$.

As before, recursive leverage amplifies both upside and downside. If adverse price movements trigger the protocol's liquidation threshold, we conservatively assume that the adversary may lose the entire initial collateral C, given the high leverage ratio and the competitive nature of MEV-driven liquidations. Formally, liquidation occurs when the normalized price change satisfies $\Delta_p \le \Delta_p^{\mathrm{liq}}$, where $\Delta_p^{\mathrm{liq}} < 0$ denotes the liquidation threshold corresponding to an adverse price movement, as implied by the protocol's maximum LTV constraint and the effective leverage of the position.

We further account for borrowing costs under time-varying lending conditions. Let r_t denote the instantaneous borrowing rate of the borrowed LST during the holding period. For a position of size N held over a duration T, we model the borrowing cost as $C_{\mathrm{borrow}} = N \int_0^T r_t \, dt \approx N\bar{r}T$, where $\bar{r}$ is the average realized borrowing rate over the holding window.

This formulation allows the borrowing cost to increase when utilization rises or when the short position remains open for longer. We ignore transaction fees in the analysis, as they are second-order compared to the potential profits and losses induced by leveraged positions. Consistent with this assumption, gas costs during our evaluation period are typically small (e.g., below \$0.2 per transaction as of April 2026 [31]). We also ignore costs at the consensus layer, as our previous evaluations show that they are close to zero in many cases.

Accordingly, the final net profit (or total loss) Π is defined as

$$\Pi = \begin{cases} -C, & \text{if liquidation occurs,} \\ \Pi_{\text{trade}} - C_{\text{borrow}}, & \text{otherwise.} \end{cases}$$

Parameter instantiation and profit estimation. Given an attack scenario characterized by $(\delta^{\text{APR}}, H, \hat{\beta}_H, \sigma_H, C, \rho, m, T)$ together with a borrowing regime and a market-depth regime, we estimate the profit distribution via Monte Carlo simulation. In each trial, we (i) sample the market noise ε and compute the normalized price change Δ_p, (ii) determine whether liquidation occurs by checking $\Delta_p \leq \Delta_p^{\text{liq}}$, (iii) compute the realized execution prices under the corresponding slippage assumptions, and (iv) compute the net profit Π using the piecewise definition above. We repeat this process for 10,000 independent trials and report the estimated expected profit $\widehat{\mathbb{E}}[\Pi]$ and the probability of a profitable outcome $\widehat{\Pr}(\Pi > 0)$.

We instantiate $\rho = 93\%$ and $\Delta_p^{\text{liq}} = -2.15\%$ using real-world market data from the Aave protocol. We set $m = 6$ to model a reasonable leverage level, consider holding windows from intraday to multi-day horizons, and use borrowing costs from observed protocol rates, with $C = 100$ ETH as a representative capital allocation. For the borrowing cost, we set the annualized borrowing rate to $\bar{r} = 2.17\%$ according to the Aave protocol.

We capture external execution frictions through three reduced-form slippage regimes with bounds of 0.05%, 0.1%, and 0.2%. These values are conservative relative to slippage settings discussed for deep-liquidity swaps in Uniswap's user guidance [78]. If market depth is worse, the adversary can abstain from opening the position, so larger slippage mainly removes the opportunity rather than imposing an additional realized trading loss.

We set $\delta^{\text{APR}} = 4\%$ as a calibrated APR-equivalent degradation. This is based on our consensus-layer attack with $\alpha_{\mathcal{A}} = 25\%$, comparable to a large staking pool such as Lido. For victims with stake share at most 10%, the attack reduces realized proposer-slot allocation by more than 16% (see Figure 6). Since block rewards are only part of total staking APR, this corresponds to an approximately 4% APR-equivalent performance decline (see Section G.1).

We use the full-sample 60-day estimates from Table 5 as our main calibration of $(\hat{\beta}_H, \sigma_H)$. Compared with longer horizons, this choice better matches the holding period considered in our shorting simulation, while still providing a stable estimate of the performance-price relation. We report alternative calibrations based on other horizons and larger slippage bounds as robustness checks in Section G.2.

We select Coinbase, Ether.fi, and Rocket Pool, whose stake shares span medium ($\approx 5\%$) to small ($\approx 0.5\%$), as representative examples to study profits across their LSTs. Figure 9 plots the empirical cumulative distribution function (ECDF) of net profit from shorting the three LSTs under the main calibration, with a slippage bound of 0.05%. For each curve, the ECDF value at zero gives the probability of a non-profitable outcome. Notably, Coinbase

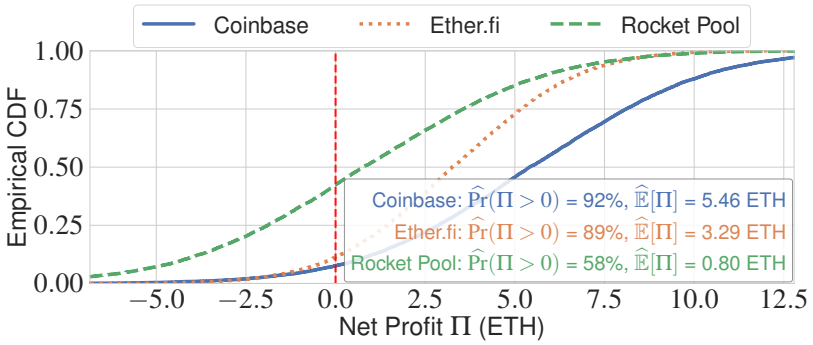


Figure 9: ECDFs of net profit from application-layer attacks on three LSTs. For a given value x, the ECDF reports the fraction of Monte Carlo trials with $\Pi \leq x$.

and Ether.fi exhibit broader profit distributions with large positive means, while Rocket Pool remains profitable in slightly more than half of the trials.

Across the three LSTs, the estimated probability of profit ranges from 58% to 92%, and the expected net profit ranges from 0.80 ETH to 5.46 ETH. For an adversary allocating $C = 100$ ETH, these profits are large relative to honest staking: the same capital would earn roughly 0.49 ETH over 60 days at a 3% APR. Thus, even under a conservative slippage bound, the application-layer profit channel can dominate honest staking rewards.

Break-even degradation threshold. To compare pools' economic exposure, we compute a break-even degradation threshold δ_i^{BE} for each pool i. We vary the APR-equivalent performance degradation δ^{APR} and define δ_i^{BE} as the smallest value for which the adversary's expected profit against pool i exceeds the honest staking reward over 60 days. A lower δ_i^{BE} indicates higher exposure, since a smaller consensus-layer performance loss is sufficient to make the attack economically preferable to honest participation.

Using our empirical calibration, we estimate δ_i^{BE} for each pool. Coinbase, Ether.fi, Rocket Pool, and Binance have low thresholds: 1.39%, 1.94%, 3.59%, and 5.35%, respectively, all below the APR-equivalent degradation achieved by our learned strategy. Mantle and Lido have higher thresholds, suggesting lower exposure: 22.06% and 33.38%, respectively.

7 Discussion

7.1 Mitigation

Note that the cross-layer attacks we study represent a general vulnerability rather than a single concrete attack. Therefore, in our mitigation analysis, we focus on defenses against different types of consensus-layer manipulation.

Impossibility of preventing leader election manipulation. We first argue that randomness beacons cannot fully eliminate leader-election manipulation without sacrificing some reasonable properties. Before stating the formal impossibility result in Theorem 7.1, we give the intuition. The attack works by letting the adversary intentionally reduce the election

chance of one account. Any mechanism that tries to repair this effect must either reassign the lost probability mass or leave it unassigned. Reassignment is unsafe because the beacon cannot distinguish honest accounts from additional attacker-controlled accounts. Leaving it unassigned means that, with some probability, some slots will not have an elected leader. This tension yields the result below.

Theorem 7.1. *Consider accounts $\{1, \ldots, n\}$ participating in a leader election mechanism, where at least two are adversary-controlled. Denote the probability assigned by the mechanism for account i to be elected in slot s by $p_i(s)$. Suppose that after adversary-controlled account $a \in [n]$ performs a manipulation in slot s, the mechanism reduces a's election probability for some subsequent slot s' by $\Delta > 0$ without lowering the election probability of any other account: $p_a(s') = p_a(s) - \Delta < p_a(s)$ and $\forall i \neq a : p_i(s') \geq p_i(s)$. Then, without knowing the identities of all accounts' owners, the mechanism cannot simultaneously satisfy the following properties: (i) no adversary-controlled account has higher election probability in slot s', (ii) a leader is guaranteed to be elected in every slot: $\forall s : \sum_{i \in [n]} p_i(s) = 1$.*

This suggests that robust defenses may require a combination of beacon design choices and external mechanisms, rather than standalone modifications of the randomness beacon. Promising directions include delaying the availability of beacon outputs until after all relevant commitments have been fixed (e.g., via verifiable delay functions (VDFs) [9]), and introducing stronger links between accounts and the entities that control them.

Preventing other attacks. For insider attacks, one mitigation idea is to require collateral, which can change the adversary's payoff structure. If a node operator must lock up collateral that can be slashed when it misses proposals or attestations, then deliberately skipping slots or withholding attestations becomes more costly. Thus, collateral can deter opportunistic insiders by making performance degradation economically unattractive. In practice, Rocket Pool has adopted a similar idea to amplify the penalty for misbehaving node operators [67].

With respect to strategic forking manipulation, we observe that most existing attacks stem from specific flaws in prior protocol designs and have been addressed by subsequent protocol upgrades. For this class of attacks, the most effective mitigation is therefore either (i) for liquid staking pool operators or regular validators to deploy the latest patched protocol or (ii) for protocol designers and the community to transition to a more secure consensus mechanism [72, 94]. The same mitigation idea also applies to network-layer or Denial-of-Service (DoS) attacks if these attacks are caused or amplified by incorrect protocol design or vulnerable software implementations. We note that eclipse attacks and mempool DoS attacks have been addressed through software upgrades [24]. For regular DoS attacks, standard network-layer mitigation is typically available from cloud services [19].

7.2 Scope and future work

DRL approach. As shown in Section 6.1, there is a residual gap between the learned policies and the theoretical optimum in the cases with high adversary stakes. This suggests that further improvements can be made to close this gap in future work. Moreover, the current framework models a single strategic agent on Ethereum PoS. This abstraction is

standard in prior work [32, 70, 39, 88, 5, 7]; modeling multiple interacting adversaries remains largely unexplored and is a natural direction for future work.

Scope of economic modeling. Because fully modeling each LST's market microstructure is difficult, we use a reduced-form profit model calibrated from real historical data. Profitability can still depend on execution conditions and parameters, especially slippage, borrowing costs, holding window T, and the estimated price-response parameters. We therefore vary these parameters in robustness checks rather than relying on a single favorable setting. Across these checks, the cross-layer channel can still create incentives larger than ordinary staking rewards in many settings.

8 Conclusions

In this paper, we propose a cross-layer attack: consensus manipulations can create targeted pool performance shocks which can be monetized via LST markets. We make the attack concrete by establishing an association between performance and LST prices, and showing how a low-stakes adversary can manipulate the PoS leader election mechanism to reduce the victim's realized block share and then use leveraged shorting to profit from the associated price drop. Our evaluations show that our DRL framework discovers strategies that significantly degrade victim performance, e.g., a 20% adversary can reduce the realized block allocation of another 20% pool by about 8.3%. Furthermore, Monte Carlo simulations reveal that such attacks are economically attractive, with a probability of profitability exceeding one-half for major staking pools and, in some cases, returns of roughly 11× the honest staking APR. Finally, we discuss possible directions for mitigating the risks of such cross-layer attacks.

References

[1] Aave. *Aave Protocol*. `https://aave.com/`. Accessed: 2026-01-24. 2026.

[2] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. "Optuna: A Next-generation Hyperparameter Optimization Framework". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD '19. Anchorage, AK, USA: Association for Computing Machinery, 2019, pp. 2623–2631. ISBN: 9781450362016. DOI: `10.1145/3292500.3330701`. URL: `https://doi.org/10.1145/3292500.3330701`.

[3] Kaya Alpturer and S. Matthew Weinberg. "Optimal RANDAO Manipulation in Ethereum". en. In: *6th Conference on Advances in Financial Technologies (AFT 2024)*. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024, 10:1–10:21. DOI: `10.4230/LIPIcs.AFT.2024.10`. URL: `https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.AFT.2024.10`.

[4] Donald WK Andrews. "Heteroskedasticity and autocorrelation consistent covariance matrix estimation". In: *Econometrica: Journal of the Econometric Society* (1991), pp. 817–858.

[5] Roi Bar-Zur, Ameer Abu-Hanna, Ittay Eyal, and Aviv Tamar. " WeRLman: To Tackle Whale (Transactions), Go Deep (RL) ". In: *2023 IEEE Symposium on Security and Privacy (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, 2023, pp. 93–110. DOI: 10.1109/SP46215.2023.10179444. URL: https://doi.ieeecomputersociety.org/10.1109/SP46215.2023.10179444.

[6] Roi Bar-Zur, Danielle Dori, Sharon Vardi, Ittay Eyal, and Aviv Tamar. "Deep Bribe: Predicting the Rise of Bribery in Blockchain Mining with Deep RL". In: *2023 IEEE Security and Privacy Workshops (SPW)*. 2023, pp. 29–37. DOI: 10.1109/SPW59333.2023.00008. URL: https://ieeexplore.ieee.org/abstract/document/10188634.

[7] Roi Bar-Zur, Aviv Tamar, and Ittay Eyal. "MAD-DAG: Protecting Blockchain Consensus from MEV". In: *arXiv preprint arXiv:2511.21552* (2025).

[8] Binance. *Ethereum Staking on Binance*. https://www.binance.com/en/earn/ethereum-staking. Accessed: 2026-02-03. Binance, 2026.

[9] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. "Verifiable Delay Functions". In: *Annual International Cryptology Conference*. Springer. 2018, pp. 757–788.

[10] Vitalik Buterin. *Validator Ordering and Randomness in PoS*. https://web.archive.org/web/20160521011553/https://vitalik.ca/files/randomness.html. 2016.

[11] Vitalik Buterin. *RANDAO beacon exploitability analysis, round 2*. https://ethresear.ch/t/randao-beacon-exploitability-analysis-round-2/1980. 2018.

[12] Vitalik Buterin. *Possible futures of the Ethereum protocol, part 1: The Merge*. 2024. URL: https://web.archive.org/web/20241201230703/https://vitalik.eth.limo/general/2024/10/14/futures1.html.

[13] Vitalik Buterin. *Some thoughts on solo staking, what realistic value solo (+ small-business and community) stakers could provide to the network, and what changes L1 can make to better support solo stakers*. 2024. URL: https://x.com/VitalikButerin/status/1838461704771907726.

[14] Vitalik Buterin, Diego Hernandez, Thor Kamphefner, Khiem Pham, Zhi Qiao, Danny Ryan, Juhyeok Sin, Ying Wang, and Yan X Zhang. "Combining ghost and casper". In: *arXiv preprint arXiv:2003.03052* (2020).

[15] Linda Cai, Jingyi Liu, S. Matthew Weinberg, and Chenghan Zhou. "Profitable Manipulations of Cryptographic Self-Selection Are Statistically Detectable". In: *6th Conference on Advances in Financial Technologies (AFT 2024)*. Ed. by Rainer Böhme and Lucianna Kiffer. Vol. 316. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024, 30:1–30:23. ISBN: 978-3-95977-345-4. DOI: 10.4230/LIPIcs.AFT.2024.30. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.AFT.2024.30.

[16] Tong Cao, Jérémie Decouchant, and Jiangshan Yu. "Leveraging the verifier's dilemma to double spend in Bitcoin". In: *Financial Cryptography and Data Security: 27th International Conference, FC 2023, Bol, Brač, Croatia, May 1–5, 2023, Revised Selected Papers, Part II*. Bol, Brač, Croatia: Springer-Verlag, 2023, pp. 149–165. ISBN: 978-3-031-47750-8. DOI: 10.1007/978-3-031-47751-5_9. URL: https://doi.org/10.1007/978-3-031-47751-5_9.

[17] Jing Chen and Silvio Micali. "Algorand: A secure and efficient distributed ledger". In: *Theoretical Computer Science*. In memory of Maurice Nivat, a founding father of Theoretical Computer Science - Part I 777 (2019), pp. 155–183. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2019.02.001. URL: https://www.sciencedirect.com/science/article/pii/S030439751930091X.

[18] Kevin Choi, Aathira Manoj, and Joseph Bonneau. " SoK: Distributed Randomness Beacons ". In: *2023 IEEE Symposium on Security and Privacy (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, 2023, pp. 75–92. DOI: 10.1109/SP46215.2023.10179419. URL: https://doi.ieeecomputersociety.org/10.1109/SP46215.2023.10179419.

[19] Cloudflare, Inc. *DDoS Protection & Mitigation Solutions*. https://www.cloudflare.com/ddos/. Accessed: 2026-02-04. 2026.

[20] Jacob Cohen. *Statistical power analysis for the behavioral sciences*. routledge, 2013.

[21] CoinGecko. *CoinGecko Crypto Data API*. https://www.coingecko.com/en/api. Accessed: 2025-12-11. 2025.

[22] Phil Daian, Rafael Pass, and Elaine Shi. "Snow White: Robustly Reconfigurable Consensus and Applications to Provably Secure Proof of Stake". In: *Financial Cryptography and Data Security*. Ed. by Ian Goldberg and Tyler Moore. Cham: Springer International Publishing, 2019, pp. 23–41. ISBN: 978-3-030-32101-7.

[23] Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. "Flash Boys 2.0: Frontrunning in Decentralized Exchanges, Miner Extractable Value, and Consensus Instability". In: *2020 IEEE Symposium on Security and Privacy (SP)*. San Francisco, CA, USA: IEEE, 2020, pp. 910–927. DOI: 10.1109/SP40000.2020.00040.

[24] Wanning Ding, Yuzhe Tang, and Yibo Wang. "Asymmetric Mempool DoS Security: Formal Definitions and Provable Secure Designs". English. In: *2025 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 2025, pp. 1584–1602. ISBN: 9798331522360. DOI: 10.1109/SP61157.2025.00061. URL: https://www.computer.org/csdl/proceedings-article/sp/2025/223600a061/21B7R6HnWWk.

[25] Paul Dworzanski. *A note on committee random number generation, commit-reveal, and last-revealer attacks*. 2018. URL: https://web.archive.org/web/20210224073927/http://paul.oemm.org/commit_reveal_subcommittees.pdf.

[26] dYdX. *dYdX Protocol*. https://www.dydx.xyz/. Accessed: 2026-01-24. 2026.

[27] Ben Edgington. *Upgrading Ethereum | 2.9.2 Randomness*. 2023. URL: https://web.archive.org/web/20240204094612/https://eth2book.info/capella/part2/building_blocks/randomness/.

[28] Ethereum. *The Beacon Chain*. 2024. URL: https://web.archive.org/web/20241128055106/https://ethereum.org/en/roadmap/beacon-chain/.

[29] Ethereum Foundation. *Ethereum Proof-of-Stake Attack and Defense*. https://ethereum.org/developers/docs/consensus-mechanisms/pos/attack-and-defense. Accessed: 2026-01-27. 2025.

[30] Ethereum Foundation. *Beacon-API*. https://ethereum.github.io/beacon-APIs/. Accessed: 2026-04-22. 2026.

[31] Etherscan. *Ethereum Gas Tracker*. https://etherscan.io/gastracker. Accessed January 30, 2026. 2026.

[32] Ittay Eyal and Emin Gün Sirer. "Majority is not enough: Bitcoin mining is vulnerable". In: *International conference on financial cryptography and data security*. Vol. 61. Association for Computing Machinery (ACM), 2014, pp. 436–454. DOI: 10.1145/3212998.

[33] Matheus V.X. Ferreira, Ye Lin Sally Hahn, S. Matthew Weinberg, and Catherine Yu. "Optimal Strategic Mining Against Cryptographic Self-Selection in Proof-of-Stake". In: *Proceedings of the 23rd ACM Conference on Economics and Computation*. EC '22. Boulder, CO, USA: Association for Computing Machinery, 2022, pp. 89–114. ISBN: 9781450391504. DOI: 10.1145/3490486.3538337.

[34] Flashbots. *MEV-Boost in a Nutshell*. https://boost.flashbots.net/. Accessed: 2026-04-22. 2026.

[35] Flashbots. *MEV-Boost Relay API*. https://flashbots.github.io/relay-specs/. Accessed: 2026-04-22. 2026.

[36] Bryan Ford and Rainer Böhme. *Rationality is Self-Defeating in Permissionless Systems*. Oct. 2019. DOI: 10.48550/arXiv.1910.08820. URL: http://arxiv.org/abs/1910.08820.

[37] Gearbox. *Gearbox Protocol*. https://gearbox.finance/. Accessed: 2026-01-24. 2026.

[38] Hans Gersbach, Akaki Mamageishvili, and Manvir Schneider. *Staking Pools on Blockchains*. 2022. DOI: 10.48550/arXiv.2203.05838. arXiv: 2203.05838 [cs.GT]. URL: https://arxiv.org/abs/2203.05838.

[39] Arthur Gervais, Ghassan O. Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. "On the Security and Performance of Proof of Work Blockchains". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. CCS '16. Vienna, Austria: Association for Computing Machinery, 2016, pp. 3–16. ISBN: 9781450341394. DOI: 10.1145/2976749.2978341.

[40] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. "Eclipse Attacks on Bitcoin's Peer-to-Peer Network". In: *24th USENIX Security Symposium (USENIX Security 15)*. Washington, D.C.: USENIX Association, Aug. 2015, pp. 129–144. ISBN: 978-1-939133-11-3. URL: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/heilman.

[41] Lioba Heimbach, Vabuk Pahari, and Eric Schertenleib. "Non-Atomic Arbitrage in Decentralized Finance". In: *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2024, pp. 3866–3884. DOI: 10.1109/SP54263.2024.00256.

[42] Lioba Heimbach, Yann Vonlanthen, Juan Villacis, Lucianna Kiffer, and Roger Wattenhofer. *Deanonymizing Ethereum Validators: The P2P Network Has a Privacy Issue*. 2024. DOI: 10.48550/arXiv.2409.04366. arXiv: 2409.04366 [cs.CR]. URL: https://arxiv.org/abs/2409.04366.

[43] hildobby. *Ethereum Staking Overview*. https://dune.com/hildobby/eth2-staking. Accessed: 2026-01-27. 2026.

[44] Charlie Hou, Mingxun Zhou, Yan Ji, Phil Daian, Florian Tramer, Giulia Fanti, and Ari Juels. "SquirRL: Automating Attack Analysis on Blockchain Incentive Mechanisms with Deep Reinforcement Learning". In: *28th Annual Network and Distributed System Security Symposium, NDSS 2021, virtually, February 21-25, 2021*. Reston, VA: The Internet Society, 2021. DOI: 10.14722/ndss.2021.24188. URL: https://www.ndss-symposium.org/ndss-paper/squirrl-automating-attack-analysis-on-blockchain-incentive-mechanisms-with-deep-reinforcement-learning/.

[45] Shengyi Huang and Santiago Ontañón. "A closer look at invalid action masking in policy gradient algorithms". In: *arXiv preprint arXiv:2006.14171* (2020).

[46] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. "Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol". In: *Advances in Cryptology – CRYPTO 2017*. Ed. by Jonathan Katz and Hovav Shacham. Cham: Springer International Publishing, 2017, pp. 357–388. ISBN: 978-3-319-63688-7.

[47] Arad Kotzer, Tom Azoulay, Yoad Abels, Aviv Yaish, and Ori Rottenstreich. "SoK: DeFi Lending and Yield Aggregation Protocol Taxonomy, Empirical Measurements, and Security Challenges". In: *IEEE Transactions on Network and Service Management* (2026), pp. 1–1. ISSN: 1932-4537. DOI: 10.1109/TNSM.2026.3682174.

[48] Kai Li, Yibo Wang, and Yuzhe Tang. "DETER: Denial of Ethereum Txpool SERvices". In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. CCS '21. Virtual Event, Republic of Korea: Association for Computing Machinery, 2021, pp. 1645–1667. ISBN: 9781450384544. DOI: 10.1145/3460120.3485369.

[49] Rujia Li, Mingfei Zhang, Xueqian Lu, Wenbo Xu, Ying Yan, and Sisi Duan. "BunnyFinder: Finding Incentive Flaws for Ethereum Consensus". In: *Cryptology ePrint Archive* (2025).

[50] William Meng. "Analysis of Factors that Drive Ethereum Staking Deposit and Withdrawal Trends across Various Staking Participant Groups". In: *Available at SSRN 4949780* (2024). DOI: 10.2139/ssrn.4949780.

[51] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. "Asynchronous methods for deep reinforcement learning". In: *International conference on machine learning*. PmLR. 2016, pp. 1928–1937.

[52] Ábel Nagy, János Tapolcai, István András Seres, and Bence Ladóczki. "Forking the RANDAO: Manipulating Ethereum's Distributed Randomness Beacon". In: *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*. CCS '25. Taipei, Taiwan: Association for Computing Machinery, 2025, pp. 873–887. ISBN: 9798400715259. DOI: 10.1145/3719027.3744852. URL: https://doi.org/10.1145/3719027.3744852.

[53] Kevin Alarcón Negy, Peter R. Rizun, and Emin Gün Sirer. "Selfish Mining Re-Examined". In: *Financial Cryptography and Data Security*. Ed. by Joseph Bonneau and Nadia Heninger. Cham: Springer International Publishing, 2020, pp. 61–78. ISBN: 978-3-030-51280-4.

[54] Joachim Neu, Ertem Nusret Tas, and David Tse. "Ebb-and-Flow Protocols: A Resolution of the Availability-Finality Dilemma". In: *2021 IEEE Symposium on Security and Privacy (SP)*. 2021, pp. 446–465. DOI: 10.1109/SP40001.2021.00045.

[55] Joachim Neu, Ertem Nusret Tas, and David Tse. "Two More Attacks on Proof-of-Stake GHOST/Ethereum". In: *Proceedings of the 2022 ACM Workshop on Developments in Consensus*. ConsensusDay '22. Los Angeles, CA, USA: Association for Computing Machinery, 2022, pp. 43–52. ISBN: 9781450398794. DOI: 10.1145/3560829.3563560. URL: https://doi.org/10.1145/3560829.3563560.

[56] Michael Neuder, Daniel J Moroz, Rithvik Rao, and David C Parkes. "Low-cost attacks on Ethereum 2.0 by sub-1/3 stakeholders". In: *arXiv preprint arXiv:2102.02247* (2021).

[57] Burak Öz, Danning Sui, Thomas Thiery, and Florian Matthes. "Who Wins Ethereum Block Building Auctions and Why?" In: *6th Conference on Advances in Financial Technologies (AFT 2024)*. Ed. by Rainer Böhme and Lucianna Kiffer. Vol. 316. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024, 22:1–22:25. ISBN: 978-3-95977-345-4. DOI: 10.4230/LIPIcs.AFT.2024.22. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.AFT.2024.22.

[58] Burak Öz, Christof Ferreira Torres, Christoph Schlegel, Bruno Mazorra, Jonas Gebele, Filip Rezabek, and Florian Matthes. "Cross-Chain Arbitrage: The Next Frontier of MEV in Decentralized Finance". In: *Proc. ACM Meas. Anal. Comput. Syst.* 9.3 (Dec. 2025). DOI: 10.1145/3771566. URL: https://doi.org/10.1145/3771566.

[59] Ulysse Pavloff, Yackolley Amoussou-Guenou, and Sara Tucci-Piergiovanni. "Ethereum Proof-of-Stake under Scrutiny". In: *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing*. SAC '23. Tallinn, Estonia: Association for Computing Machinery, 2023, pp. 212–221. ISBN: 9781450395175. DOI: 10.1145/3555776.3577655. URL: https://doi.org/10.1145/3555776.3577655.

[60] Polymarket. *Polymarket: The World's Largest Prediction Market*. https://polymarket.com/. Accessed: 2026-01-24. 2026.

[61] Potuz. *Justification withholding attacks*. Online HackMD note. Accessed: 2026-01-24. 2024. URL: https://hackmd.io/o9tGPQL2Q4iH3Mg7Mma9wQ.

[62] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[63] Kaihua Qin, Liyi Zhou, Pablo Gamito, Philipp Jovanovic, and Arthur Gervais. "An empirical study of DeFi liquidations: incentives, risks, and instabilities". In: *Proceedings of the 21st ACM Internet Measurement Conference*. IMC '21. Virtual Event: Association for Computing Machinery, 2021, pp. 336–350. ISBN: 9781450391290. DOI: 10.1145/3487552.3487811. URL: https://doi.org/10.1145/3487552.3487811.

[64] Kaihua Qin, Liyi Zhou, Benjamin Livshits, and Arthur Gervais. "Attacking the defi ecosystem with flash loans for fun and profit". In: *International conference on financial cryptography and data security*. Springer. 2021, pp. 3–32.

[65] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. "Stable-Baselines3: Reliable Reinforcement Learning Implementations". In: *Journal of Machine Learning Research* 22.268 (2021), pp. 1–8. URL: http://jmlr.org/papers/v22/20-1364.html.

[66] F. Ritz and A. Zugenmaier. "The Impact of Uncle Rewards on Selfish Mining in Ethereum". In: *2018 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. Los Alamitos, CA, USA: IEEE Computer Society, Apr. 2018, pp. 50–57. DOI: 10.1109/eurospw.2018.00013.

[67] Rocket Pool. *8-ETH Bonded Minipools | RPL Collateral Requirements*. https://docs.rocketpool.net/legacy/atlas/lebs#rpl-collateral. Accessed: 2026-02-04. 2023.

[68] M. Rosenfeld. *Analysis of Bitcoin Pooled Mining Reward Systems*. Dec. 2011. DOI: 10.48550/arxiv.1112.4980. arXiv: 1112.4980 [cs.DC].

[69] Oriol Saguillo, Vahid Ghafouri, Lucianna Kiffer, and Guillermo Suarez-Tangil. "Unravelling the Probabilistic Forest: Arbitrage in Prediction Markets". In: *7th Conference on Advances in Financial Technologies (AFT 2025)*. Ed. by Zeta Avarikioti and Nicolas Christin. Vol. 354. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025, 27:1–27:24. ISBN: 978-3-95977-400-0. DOI: 10.4230/LIPIcs.AFT.2025.27. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.AFT.2025.27.

[70] Ayelet Sapirshtein, Yonatan Sompolinsky, and Aviv Zohar. "Optimal Selfish Mining Strategies in Bitcoin". In: *Financial Cryptography and Data Security*. Ed. by Jens Grossklags and Bart Preneel. Berlin, Heidelberg: Springer, 2017, pp. 515–532. ISBN: 978-3-662-54970-4.

[71] Roozbeh Sarenche, Alireza Aghabagherloo, Svetla Nikova, and Bart Preneel. "Bitcoin Under Volatile Block Rewards: How Mempool Statistics Can Influence Bitcoin Mining". In: CCS '25. New York, NY, USA: Association for Computing Machinery, 2025, pp. 903–917. ISBN: 9798400715259. DOI: 10.1145/3719027.3744875. URL: https://doi.org/10.1145/3719027.3744875.

[72] Roozbeh Sarenche, Ertem Nusret Tas, Barnabe Monnot, Caspar Schwarz-Schilling, and Bart Preneel. " Commitment Attacks on Ethereum's Reward Mechanism ". In: *2025 IEEE 10th European Symposium on Security and Privacy (EuroS&P)*. Los Alamitos, CA, USA: IEEE Computer Society, July 2025, pp. 504–526. DOI: 10.1109/EuroSP63326.2025.00036. URL: https://doi.ieeecomputersociety.org/10.1109/EuroSP63326.2025.00036.

[73] Stefan Scharnowski and Hossein Jahanshahloo. "The Economics of Liquid Staking Derivatives: Basis Determinants and Price Discovery". In: *Journal of Futures Markets* 45.2 (2025), pp. 91–117. DOI: https://doi.org/10.1002/fut.22556. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/fut.22556.

[74] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[75] Caspar Schwarz-Schilling, Joachim Neu, Barnabé Monnot, Aditya Asgaonkar, Ertem Nusret Tas, and David Tse. "Three Attacks on Proof-of-Stake Ethereum". In: *Financial Cryptography and Data Security: 26th International Conference, FC 2022, Grenada, May 2–6, 2022, Revised Selected Papers*. Grenada, Grenada: Springer-Verlag, 2022, pp. 560–576. ISBN: 978-3-031-18282-2. DOI: 10.1007/978-3-031-18283-9_28. URL: https://doi.org/10.1007/978-3-031-18283-9_28.

[76] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. "Gymnasium: A Standard Interface for Reinforcement Learning Environments". In: *arXiv preprint arXiv:2407.17032* (2024).

[77] Apostolos Tzinas and Dionysis Zindros. "The Principal–Agent Problem in Liquid Staking". en. In: *Financial Cryptography and Data Security. FC 2023 International Workshops*. Ed. by Aleksander Essex, Shin'ichiro Matsuo, Oksana Kulyk, Lewis Gudgeon, Ariah Klages-Mundt, Daniel Perez, Sam Werner, Andrea Bracciali, and Geoff Goodell. Cham: Springer Nature Switzerland, 2024, pp. 456–469. ISBN: 9783031488061. DOI: 10.1007/978-3-031-48806-1_29.

[78] Uniswap Labs. *API Integration Guide*. https://api-docs.uniswap.org/guides/integration_guide. Accessed: 2026-04-23. 2026.

[79] Taotao Wang, Soung Chang Liew, and Shengli Zhang. "When Blockchain Meets AI: Optimal Mining Strategy Achieved by Machine Learning". In: *International Journal of Intelligent Systems* 36.5 (2021), pp. 2183–2207. ISSN: 1098-111X. DOI: 10.1002/int.22375.

[80] Ye Wang, Yan Chen, Haotian Wu, Liyi Zhou, Shuiguang Deng, and Roger Wattenhofer. "Cyclic Arbitrage in Decentralized Exchanges". In: *Companion Proceedings of the Web Conference 2022*. WWW '22. Virtual Event, Lyon, France: Association for Computing Machinery, 2022, pp. 12–19. ISBN: 9781450391306. DOI: 10.1145/3487553.3524201. URL: https://doi.org/10.1145/3487553.3524201.

[81] Yibo Wang, Yuzhe Tang, Kai Li, Wanning Ding, and Zhihua Yang. "Understanding Ethereum Mempool Security under Asymmetric DoS by Symbolized Stateful Fuzzing". In: *33rd USENIX Security Symposium (USENIX Security 24)*. Philadelphia, PA: USENIX Association, Aug. 2024, pp. 4747–4764. ISBN: 978-1-939133-44-1. URL: https://www.usenix.org/conference/usenixsecurity24/presentation/wang-yibo.

[82] Fei Wu, Danning Sui, Thomas Thiery, and Mallesh Pai. "Measuring CEX-DEX Extracted Value and Searcher Profitability: The Darkest of the MEV Dark Forest". In: *7th Conference on Advances in Financial Technologies (AFT 2025)*. Ed. by Zeta Avarikioti and Nicolas Christin. Vol. 354. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025, 26:1–26:23. ISBN: 978-3-95977-400-0. DOI: 10.4230/LIPIcs.AFT.2025.26. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.AFT.2025.26.

[83] Karl Wüst and Arthur Gervais. *Ethereum Eclipse Attacks*. Tech. rep. ETH Zurich, 2016.

[84] Matheus Xavier Ferreira, Aadityan Ganesh, Jack Hourigan, Hannah Huh, S. Weinberg, and Catherine Yu. "Computing Optimal Manipulations in Cryptographic Self-Selection Proof-of-Stake Protocols". In: *Proceedings of the 25th ACM Conference on Economics and Computation*. EC '24. New York, NY, USA: Association for Computing Machinery, 2024. URL: https://doi.org/10.1145/3670865.3673602.

[85] Xihan Xiong, Zhipeng Wang, Xi Chen, William Knottenbelt, and Michael Huth. *Leverage Staking with Liquid Staking Derivatives (LSDs): Opportunities and Risks*. 2024. DOI: 10.48550/arXiv.2401.08610. arXiv: 2401.08610 [q-fin.GN]. URL: https://arxiv.org/abs/2401.08610.

[86] Xihan Xiong, Zhipeng Wang, and Qin Wang. "Exploring the Market Dynamics of Liquid Staking Derivatives (LSDs)". In: *2024 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. 2024, pp. 334–342. DOI: 10.1109/ICBC59979.2024.10634417.

[87] Aviv Yaish, Kaihua Qin, Liyi Zhou, Aviv Zohar, and Arthur Gervais. "Speculative Denial-of-Service Attacks in Ethereum". In: *33rd USENIX Security Symposium (USENIX Security 24)*. USENIXSEC '24. Philadelphia, PA: USENIX Association, 2024. URL: https://www.usenix.org/conference/usenixsecurity24/presentation/yaish.

[88] Aviv Yaish, Gilad Stern, and Aviv Zohar. "Uncle Maker: (Time)Stamping Out The Competition in Ethereum". In: *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23)*. CCS '23. Copenhagen, Denmark: Association for Computing Machinery, 2023. ISBN: 9798400700507. DOI: 10.1145/3576915.3616674.

[89] Aviv Yaish, Saar Tochner, and Aviv Zohar. "Blockchain Stretching & Squeezing: Manipulating Time for Your Best Interest". In: *Proceedings of the 23rd ACM Conference on Economics and Computation*. EC '22. Boulder, CO, USA: Association for Computing Machinery, 2022, pp. 65–88. ISBN: 9781450391504. DOI: 10.1145/3490486.3538250. URL: https://doi.org/10.1145/3490486.3538250.

[90] Sen Yang, Kartik Nayak, and Fan Zhang. "Decentralization of Ethereum's Builder Market". In: *2025 IEEE Symposium on Security and Privacy (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, May 2025, pp. 1512–1530. DOI: 10.1109/SP61157.2025.00157. URL: https://doi.ieeecomputersociety.org/10.1109/SP61157.2025.00157.

[91] Sen Yang, Burak Öz, Fei Wu, and Fan Zhang. "Geographical Centralization Resilience in Ethereum's Block-Building Paradigms". In: *Proc. ACM Meas. Anal. Comput. Syst.* 10.2 (June 2026). DOI: 10.1145/3805637. URL: https://doi.org/10.1145/3805637.

[92] Sen Yang, Fan Zhang, Ken Huang, Xi Chen, Youwei Yang, and Feng Zhu. "SoK: MEV Countermeasures". In: *Proceedings of the Workshop on Decentralized Finance and Security*. DeFi '24. Salt Lake City, UT, USA: Association for Computing Machinery, 2024, pp. 21–30. ISBN: 9798400712272. DOI: 10.1145/3689931.3694911. URL: https://doi.org/10.1145/3689931.3694911.

[93] Mingfei Zhang, Rujia Li, and Sisi Duan. "Max Attestation Matters: Making Honest Parties Lose Their Incentives in Ethereum PoS". In: *33rd USENIX Security Symposium (USENIX Security 24)*. USENIXSEC '24. Philadelphia, PA: USENIX Association, 2024.

[94] Mingfei Zhang, Rujia Li, Xueqian Lu, and Sisi Duan. "Available Attestation: Towards a Reorg-Resilient Solution for Ethereum Proof-of-Stake". In: *Proceedings of the 34th USENIX Conference on Security Symposium*. Seattle, WA, USA: USENIX Association, 2025. ISBN: 978-1-939133-52-6.

[95] Liyi Zhou, Kaihua Qin, Antoine Cully, Benjamin Livshits, and Arthur Gervais. "On the Just-In-Time Discovery of Profit-Generating Transactions in DeFi Protocols". In: *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2021, pp. 919–936. DOI: 10.1109/SP40001.2021.00113.

A Staking Pool Distribution and Trends

In this section, we provide more details about the liquid staking pool distribution. All data are sourced from an online dashboard [43]. Figure 10 shows how the distribution of stake in Ethereum has evolved over time across different categories of stakers. Liquid staking-related categories, including Liquid staking, Liquid restaking, and Staking pools, account for a large and relatively stable fraction of total stake, exceeding 50% most of the time. Note that this represents a lower bound, as some stakers labeled as "CEXs," such as Binance, also provide liquid staking services and issue their LSTs [8]. More generally, each staker is assigned a single primary label in the dashboard; this labeling scheme does not capture the full spectrum of services provided by a staker, but instead reflects its most salient category.

Table 3 summarizes Ethereum's top 50 stakers as of February 2026, ranked by stake. At the top of the list, Lido accounts for close to 25% of the total stake and representing the largest ETH staking provider. Following are Binance, Ether.fi, and Coinbase, each with approximately 5% to 10% of the total stake. The next tiers consist of stakers holding 1% to 5% and 0.5% to 1%, representing relatively small-sized ETH stakers. The remaining stakers fall into the tail of the stake distribution, each holding less than 0.5% of the total stake.

Table 3: The top 50 stakers currently operating on Ethereum.

Staker	Stake (ETH)	Share (%)	Category
Lido	8680063	23.3992	Liquid Staking
Binance	3409104	9.1901	CEXs
ether.fi	2390809	6.445	Liquid Restaking
Coinbase	1795077	4.8391	CEXs
Figment	1471104	3.9657	Staking Pools
Kraken	1401410	3.7778	CEXs
Blockdaemon	989696	2.668	Staking Pools
Everstake	693280	1.8689	Staking Pools
Kiln	621410	1.6752	Staking Pools
Rocket Pool	558621	1.5059	Liquid Staking
Bitcoin Suisse	432856	1.1669	CEXs
OKX	419198	1.13	CEXs
Upbit	408352	1.1008	CEXs
StakeWise	357405	0.9635	Staking Pools
P2P.org	325710	0.878	Staking Pools
stakefish	289738	0.7811	Staking Pools
Liquid Collective	281888	0.7599	Liquid Staking
DARMA Capital	239424	0.6454	Staking Pools
Renzo	193281	0.521	Liquid Restaking
Bitstamp	188064	0.507	CEXs
Mantle	186080	0.5016	Staking Pools
Kelp DAO	164128	0.4425	Liquid Restaking
Stader	139204	0.3753	Liquid Staking
CoinSpot	137664	0.3711	CEXs
Diva (Pre-launch)	133484	0.3598	Liquid Staking
Consensys	103616	0.2793	Staking Pools
Octant	100352	0.2705	Staking Pools
Revolut	60864	0.1641	CEXs
Frax Finance	59520	0.1605	Liquid Staking
imToken	55712	0.1502	Staking Pools
Gate.io	54816	0.1478	CEXs
Bitfinex	52099	0.1404	CEXs
Cluster 0x82df	49376	0.1331	Unidentified
XHash	41376	0.1115	CEXs
StakeHound	37504	0.1011	Liquid Staking
MyEtherWallet	35395	0.0954	Staking Pools
Puffer Finance	30341	0.0818	Liquid Restaking
Bitget	26944	0.0726	CEXs
GMO Coin	21280	0.0574	CEXs
KuCoin	20000	0.0539	CEXs
Swell	19201	0.0518	Liquid Restaking
RockX	18913	0.051	Staking Pools
Mercado Bitcoin	17888	0.0482	CEXs
WEX Exchange	16000	0.0431	CEXs
Origin Protocol	15520	0.0418	Staking Pools
Cluster 0xc372	11968	0.0323	Unidentified
CoinEx	9952	0.0268	CEXs
Stkr (Ankr)	9228	0.0249	Liquid Staking
Swell (Pre-launch)	7723	0.0208	Liquid Staking
was.eth	7690	0.0207	Solo Stakers

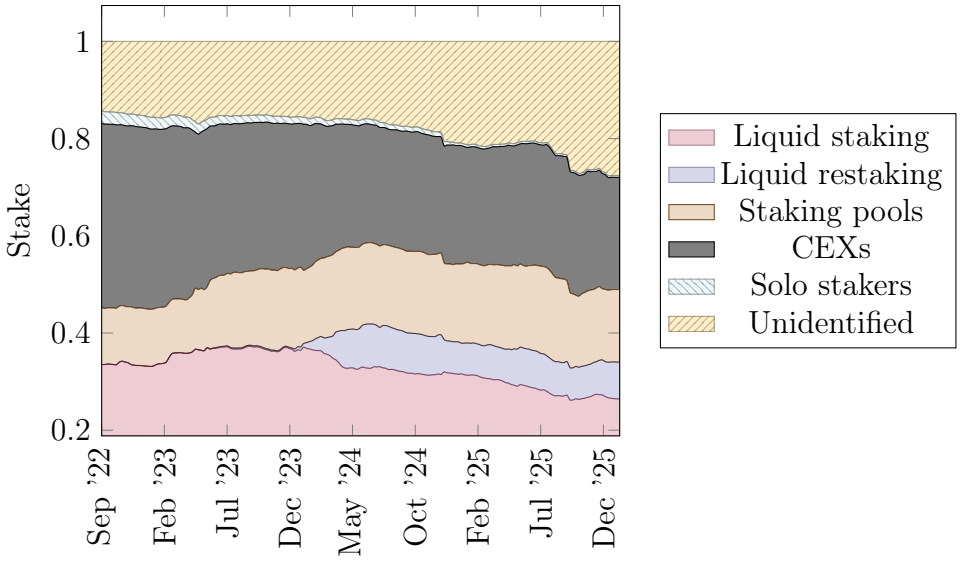


Figure 10: Ethereum's stake distribution over time, according to staker category. Ethereum launched a preliminary version of its PoS protocol in 2020, running it alongside its proof-of-work (PoW) protocol until officially transitioning to PoS on September 15th, 2022 [28].

Table 4: Correlation and regression results for the relationship between pool performance and 90-day forward LST returns.

Pool	Pearson	Spearman	β	HAC SE	p-value
Lido	0.251	0.181	0.0636	0.0185	0.0006
Binance	0.319	0.365	0.1097	0.0215	$<$0.0001
Coinbase	0.662	0.602	0.5472	0.0731	$<$0.0001
Ether.fi	0.228	0.233	0.3877	0.1145	0.0007
Rocket Pool	0.179	0.103	0.1296	0.0507	0.0106
Mantle	0.017	0.089	0.0155	0.0295	0.5990

B Robustness of the Price-Performance Analysis

To examine whether the relation between pool performance and LST valuation depends on the return horizon, we repeat the same specification using 90-day and 60-day forward normalized returns. All other definitions and estimation settings follow the main analysis.

Tables 4 and 5 report the corresponding results. The positive relation is relatively robust across horizons: most estimated coefficients remain positive, and Binance, Coinbase, Ether.fi, and Rocket Pool remain statistically significant under both shorter horizons. This supports the main finding that pool performance and LST valuation are meaningfully related, while suggesting that the relation becomes clearer over longer horizons.

C Glossary

Following is a list of the acronyms used in the paper.

Table 5: Correlation and regression results for the relationship between pool performance and 60-day forward LST returns.

Pool	Pearson	Spearman	β	HAC SE	p-value
Lido	0.053	0.092	0.0149	0.0210	0.4788
Binance	0.295	0.336	0.0932	0.0216	$<$0.0001
Coinbase	0.533	0.452	0.3587	0.0503	$<$0.0001
Ether.fi	0.170	0.204	0.2564	0.0989	0.0095
Rocket Pool	0.192	0.060	0.1389	0.0542	0.0103
Mantle	0.029	0.137	0.0225	0.0280	0.4208

APR annual percentage rate
CEX centralized exchange
DEX decentralized exchange
DoS Denial-of-Service
DRL deep reinforcement learning
ECDF empirical cumulative distribution function
HAC Heteroskedasticity and Autocorrelation Consistent
LST liquid staking token
LTV loan-to-value
MDP Markov decision process
MEV Maximal Extractable Value
PoS proof-of-stake
PoW proof-of-work
PPO Proximal Policy Optimization
VDF verifiable delay function

D Other Consensus-layer Manipulations

In this section, we provide further details on other consensus-layer manipulation approaches that are available to low-stakes adversaries.

Insider attacks. In practice, pools may allow external participants to join as node operators rather than requiring all operators to be internal to the pool [67]. Instead of manipulating consensus rules, the adversary can participate in the target pool as an operator and strategically miss slots or withhold attestations to degrade the pool's performance [68, 77]. We refer to this class of attacks as *insider attacks*. Notably, such attacks require only standard participation privileges and do not rely on access to internal protocol implementations.

Network-layer attacks. Another way to cause the target pool's validators to miss block proposals or attestations is to disrupt their network connectivity during their assigned slots. This requires the adversary to first identify the target's network endpoints, e.g., using the methods of Heimbach et al. [42]. Accordingly, the adversary can launch network-layer attacks to achieve this goal. For example, eclipse attacks [40, 83] isolate targeted validators by controlling their network connections, causing them to observe delayed or incomplete blockchain states. Another possibility is to launch DoS attacks, which render the validator unavailable during its assigned slots and prevent it from performing as expected. Examples include Mempool DoS attacks [48, 87, 81], which can disrupt block building and propagation.

E Application-layer Monetization Details

We now provide details for the application-layer channels summarized in Section 5.

E.1 Taxonomy of monetization channels

Given the established correlation between validator performance and the LST price, an adversary can proactively trigger price deviations through consensus-layer attacks. This capability transforms the LST from a passive asset into a manipulatable financial instrument. Building upon the premise of predictable price depreciation, we systematically explore the DeFi landscape to identify potential monetization channels.

Table 6: Taxonomy of Monetization Channels

Channel	Mechanism	Protocols	Complexity	Capital	Time Window	Key Risk
Shorting	Plain lending Leveraged lending Derivatives	Aave, Spark Gearbox, Morpho dYdX, Binance	Low Medium Low	High Medium Low	Days to months	Liquidation, and the risk increases with leverage
Prediction		Polymarket, Kalshi	Low	Low	Days to months	Oracle Manipulation
Liquidation		Aave, Spark	High	Low	1 block (seconds)	MEV Competition
Arbitrage	Atomic Non-atomic	Uniswap, Curve	High High	Low High	1 block (seconds) Seconds to minutes	MEV Competition

As summarized in Table 6, the application layer offers a diverse array of monetization channels for an adversary capable of inducing a decrease in LST prices. These channels vary significantly in terms of capital efficiency, risk exposure, and dependence on external market conditions. We categorize them into four primary classes: *shorting*, *prediction*, *liquidation*, and *arbitrage*, which we analyze in detail below.

Shorting. One strategy is to short the LST associated with the target liquid staking pool. We consider three mechanisms for executing short positions.

- One can use *lending protocols* [47], such as Aave [1]. The adversary deposits neutral collateral (e.g., ETH) into the protocol to borrow the target LST and then the borrowed tokens are subsequently sold on a decentralized exchange (DEX) to establish a short position. This approach requires upfront capital, since the position must be over-collateralized, and

incurs ongoing borrowing interest as an operational cost. The primary risk is liquidation[1], which is triggered when a price rebound causes the debt value to exceed the protocol's liquidation threshold.

- *Leveraging protocols* like Gearbox [37], the adversary can open "Credit Accounts": smart contract containers that allow for higher leverage within whitelisted DeFi protocols. This approach reduces the initial capital by utilizing the protocol's credit line; however, it also introduces high liquidation sensitivity, as the increased leverage makes the account highly vulnerable to even minor price fluctuations.

- The third mechanism is *perpetual futures*. This approach uses decentralized derivatives platforms (e.g., dYdX [26]) or centralized exchanges (CEXs) to bet on price decreases without requiring physical delivery of the LST. Instead of borrowing the asset, the adversary provides a collateral deposit, known as *margin*, to back a synthetic short position. This mechanism imposes no strict capital floor, allowing the adversary to selectively determine their leverage ratio. While a higher leverage ratio enables significant capital efficiency, it proportionally increases the risk of liquidation.

Prediction. Adversaries can further leverage prediction markets [69], such as Polymarket [60], to monetize their internal knowledge of an impending attack. By betting on unfavorable outcomes (e.g., a price decrease) that they intentionally induce, they exploit information asymmetry between informed adversaries and uninformed participants. Unlike short selling, prediction markets offer a relatively low capital barrier, allowing adversaries to place bets without significant collateral requirements. However, this channel is constrained by market depth (i.e., the availability of counterparties) and settlement risk, since the eventual payout depends on the correctness and potential manipulability of the oracle.

Liquidation. As the LST price declines, many existing collateralized positions may hit their liquidation thresholds. Since the adversary has the privileged knowledge regarding the timing and impact of the attack, they can preemptively prepare liquidation transactions to capture these opportunities the moment the price drops. In terms of capital requirements, this channel is highly efficient as it can be facilitated via flash loans [64]. However, the primary risk is the intense MEV competition [23]: while the adversary has a head start, other searchers monitor the mempool for oracle updates. As a result, the adversary may either lose the liquidation opportunity or have to share most of the potential profit with the block builder to secure inclusion.

Arbitrage. The token price adjustment is often non-simultaneous across CEXs and DEXs, which creates opportunities for arbitrage, i.e., profiting from cross-market price differences. The adversary can exploit this information asymmetry to execute *atomic arbitrage* across DEXs [95, 80], *non-atomic arbitrage* between CEXs and DEXs, as well as across multiple Layer-2 blockchains [41, 82, 58]. Atomic arbitrage requires little capital due to flash loans, whereas other forms require capital on CEXs or across chains. Nevertheless, the primary

[1]Liquidation [63] refers to the protocol mechanism that forcibly closes undercollateralized positions, allowing liquidators to repay outstanding debt and seize collateral at a discount.

risk mirrors that of liquidation: the MEV competition introduces execution uncertainty and potentially prohibitive costs for the adversary.

E.2 Shorting LST through lending protocols

We describe a canonical mechanism for constructing a short position on an LST via DeFi lending protocols, as this strategy is simple and can be readily executed by the adversary.

To establish a short position on an LST, the adversary deposits a neutral asset (ETH) as collateral in a lending protocol, borrows LST, and immediately sells the borrowed LST on the secondary market. Let C denote the adversary's initial ETH collateral, $\rho \in (0, 1)$ the effective LTV ratio, and P_t the LST/ETH market price at time t. The adversary can therefore borrow $Q_t = \frac{\rho C}{P_t}$ units of LST. By selling the borrowed LST, the adversary obtains ρC units of ETH. If the LST/ETH price subsequently decreases to $P_{t'} < P_t$, the adversary can repurchase the borrowed LST at a lower cost of

$$Q_t \cdot P_{t'} = \rho C \cdot \frac{P_{t'}}{P_t}.$$

So the adversary's gross profit from the short position is

$$\Pi_{\text{gross}} = \rho \cdot C \cdot \Delta_p = \rho C(1 - \frac{P_{t'}}{P_t}).$$

Leveraged short. We model leverage amplification in discrete rounds $r = 0, 1, \ldots, k$ starting from the time t, where each round reuses the ETH proceeds from selling borrowed LST as additional collateral. At round t, the adversary borrows

$$Q_r = \frac{\rho C_r}{P_r}$$

units of LST and immediately sells them for ρC_r units of ETH. We assume $P_r = P_t$ within each round, i.e., the borrowing and swap operations are executed within a single short execution window, so price drift is negligible. This can be justified by the fact that these operations can be done automatically by a script. The adversary then deposits these ETH proceeds as collateral for the next round, yielding the recursion

$$C_{r+1} = \rho C_r, \qquad C_0 = C.$$

After k rounds, the total borrowed (and sold) LST amount is

$$Q^{(k)} = \sum_{r=0}^{k} Q_r = \frac{\rho C}{P_t} \sum_{r=0}^{k} \rho^r = \frac{\rho C}{P_t} \cdot \frac{1 - \rho^{k+1}}{1 - \rho}.$$

If the LST/ETH price subsequently decreases from P_t to $P_{t'} < P_t$, the adversary repurchases $Q^{(k)}$ units of LST to repay the debt, yielding the gross profit

$$\Pi^{(k)}_{\text{gross}} = Q^{(k)}(P_t - P_{t'}) = \rho C \left(1 - \frac{P_{t'}}{P_t}\right) \cdot \frac{1 - \rho^{k+1}}{1 - \rho}.$$

F Additional Details for DRL Evaluation

F.1 Hyperparameter

Given that the performance of Proximal Policy Optimization (PPO) is known to be sensitive to hyperparameter configurations, we employed Optuna [2], an automatic hyperparameter optimization framework, to tune the agent. We used the Tree-structured Parzen Estimator (TPE) sampler to search for an optimal set of hyperparameters.

For RQ1, we tune the hyperparameters using the self-optimization setting with 30% adversarial stake. For RQ2, we tune them using the pure-griefing setting with a 20% adversarial stake and a 20% victim stake. The search objective is to maximize the adversary's realized slot allocation for self-optimization policies, and to minimize the victim's realized slot allocation for pure-griefing policies. For RQ3, we keep the hyperparameters aligned with the corresponding objective: the value-maximizing policies use the RQ1-tuned hyperparameters, while the victim-value-minimizing policies use the RQ2-tuned hyperparameters. The selected hyperparameters are presented in Table 7.

Table 7: Optimized Hyperparameters via Optuna.

Parameter	**Search Space / Range**	**Optimized Value (RQ1)**	**Optimized Value (RQ2)**
Learning Rate	$[1 \times 10^{-5}, 1 \times 10^{-3}]$	1.47×10^{-5}	3.99×10^{-5}
n_steps (Rollout Buffer)	$\{256, 512, 1024, 2048, 4096\}$	1024	256
Batch Size	$\{32, 64, 128, 256, 512\}$	512	64
n_epochs (Optimization)	$\{5, 10, 20\}$	10	20
GAE Lambda (λ)	$[0.80, 0.98]$	0.929	0.964
Clip Range (ϵ)	$[0.10, 0.30]$	0.184	0.133
Entropy Coef (c_1)	$[1 \times 10^{-5}, 1 \times 10^{-2}]$	1.51×10^{-3}	4.08×10^{-5}
Value Function Coef (c_2)	$[0.30, 1.00]$	0.393	0.409

F.2 Ablation study

We conduct an ablation study to evaluate whether our observation features help the agent learn effective strategies. Specifically, we conduct two experiments in which both the adversary and victim stakes are 20%, i.e., $\alpha_{\mathcal{A}} = 20\%$, $\alpha_{\mathcal{T}} = 20\%$. In the full observation space, the discovered strategy achieves a 8.3% reduction in the victim's slot allocation. For the first experiment, we disable the counterfactual strategic oracles and train and evaluate the model using the same process as in Section 6.1 (RQ2). The results show that, in this setting, the discovered strategy can reduce only 0.8% of the victim's slot allocation, which is significantly lower than 8.3% in our evaluation. In the second experiment, we disable the branch identifier and repeat the same process, with the results showing that the discovered strategy reduces 7.5% of the victim's allocation, suggesting that this feature is also helpful.

We further evaluate the value-aware observation features in the MEV-aware setting by masking one feature at a time during evaluation while keeping the trained policy fixed. The current slot value $V_s^{(\text{cur})}$ accounts for 83.26% of the clipped reward drop, while $V_s^{(\text{priv})}$ and $V_s^{(\text{pub})}$ account for 9.70% and 7.04%, respectively, suggesting that the policy mainly relies on current MEV value and also uses branch-level value information.

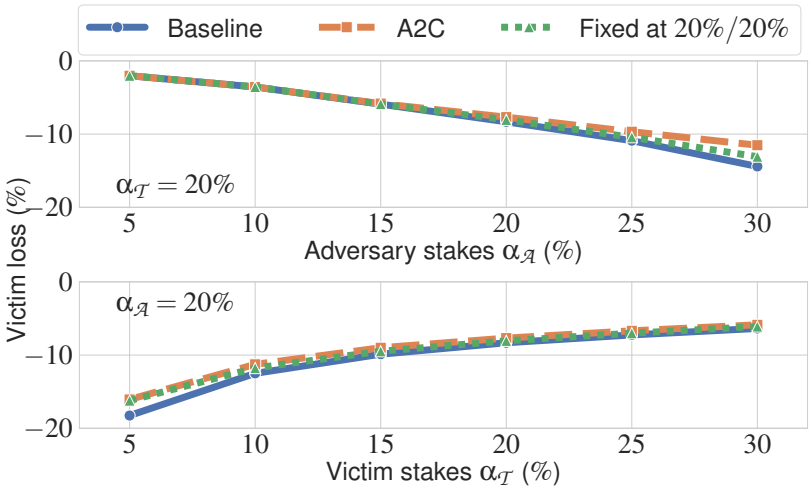


Figure 11: Victim loss under three settings: Baseline (RQ2 training per stake), A2C (the same procedure using A2C), and Fixed (policy trained at $\alpha_{\mathcal{A}} = 20\%$, $\alpha_{\mathcal{I}} = 20\%$).

F.3 Robustness and generalizability

After validating the effectiveness of the discovered strategies in Section 6.1, we examine how sensitive their effectiveness is to the choice of learning algorithm and training configuration, rather than assuming that it solely reflects properties of the consensus protocol. To this end, we evaluate whether strategies learned under one algorithmic setting or stake configuration generalize across different learning algorithms and unseen stake distributions. We repeat the RQ2 training using the A2C algorithm [51] to assess algorithm independence. Separately, to evaluate zero-shot generalization across stake distributions, we test the fixed policy trained with $\alpha_{\mathcal{A}} = 20\%$ and $\alpha_{\mathcal{I}} = 20\%$ on other stake configurations without retraining.

We evaluate the effectiveness of each setting using the same metric (victim loss) as in RQ2. As shown in Figure 11, the three curves closely track each other across all stake configurations. Even when the Fixed or A2C policies are slightly weaker than the Baseline at some stakes, e.g., $\alpha_{\mathcal{A}} = 20\%, \alpha_{\mathcal{I}} = 5\%$, their effectiveness in performance degradation remains stronger than the selfish-mixing strategies [3, 52], the estimated victim loss is about -2% under the same stake configuration.

Together, these results indicate limited dependence on the choice of learning algorithm, as well as meaningful zero-shot generalization across stake distributions.

F.4 Policy validation under heterogeneous block values

In this section, we compare the adversary's gain and the victim's loss in the MEV-aware setting with the corresponding results in RQ1 and RQ2 to assess whether our DRL framework still learns effective strategies under heterogeneous block values.

As shown in Figure 12, we measure the impact of the learned self-optimization strategy in terms of the adversary's value gain. For each pair of MEV parameters (δ, F), we compute the expected value captured by the adversary and compare it with the realized value captured during evaluation. The results show that the adversary's value gain (about 6% above the

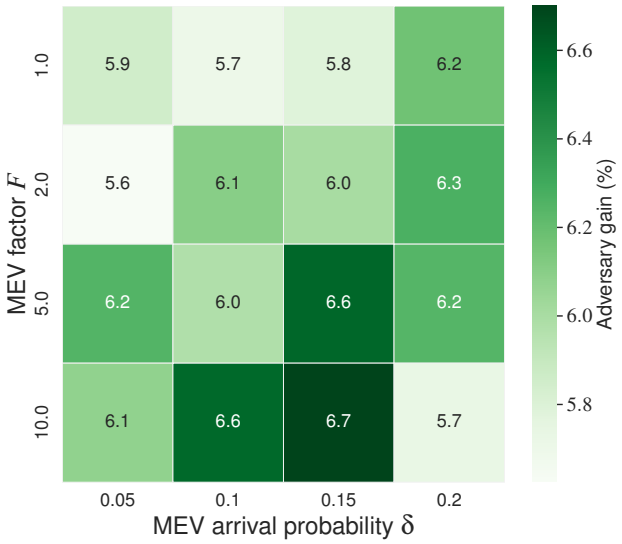


Figure 12: Heatmap of the impact of the learned self-optimization strategy under varying MEV arrival probability δ and additional value F.

expected value) is slightly higher than in the baseline setting, where the adversary increases its slot allocation by 4.8%. This suggests that our DRL framework not only learns an effective self-optimization strategy, but also exploits heterogeneous block values by more effectively capturing high-value blocks when MEV opportunities are present.

As shown in Figure 13, we measure the impact of the learned pure-griefing strategy in terms of value loss. For each pair of MEV parameters (δ, F), we compute the expected value captured by the adversary and the victim based on the expected per-slot value induced by δ and F. We then compare these baselines with the realized values captured during evaluation, and report the resulting adversary loss and victim loss. The results show that the victim's loss (about 8%) remains close to that in RQ2 across different MEV settings, while the adversary's own loss is even lower. This confirms that our DRL framework can still learn effective pure-griefing strategies under heterogeneous block values.

G Additional Profit Simulation Details

G.1 APR-equivalent degradation calibration

The consensus-layer attack directly reduces the target pool's realized proposal allocation, while the application-layer simulation requires an APR-equivalent degradation parameter. We calibrate this mapping using Lido's reward decomposition on 2025-06-09. On that day, block proposal rewards, including both execution-layer rewards and consensus-layer proposer rewards, were about 167 ETH, while total staking revenue was about 683 ETH. Thus, proposal rewards accounted for approximately $167/683 \approx 24.5\%$ of total staking revenue. A 16% reduction in realized proposer allocation therefore corresponds to

$$\delta^{\mathrm{APR}} \approx 0.16 \times \frac{167}{683} \approx 4\%,$$

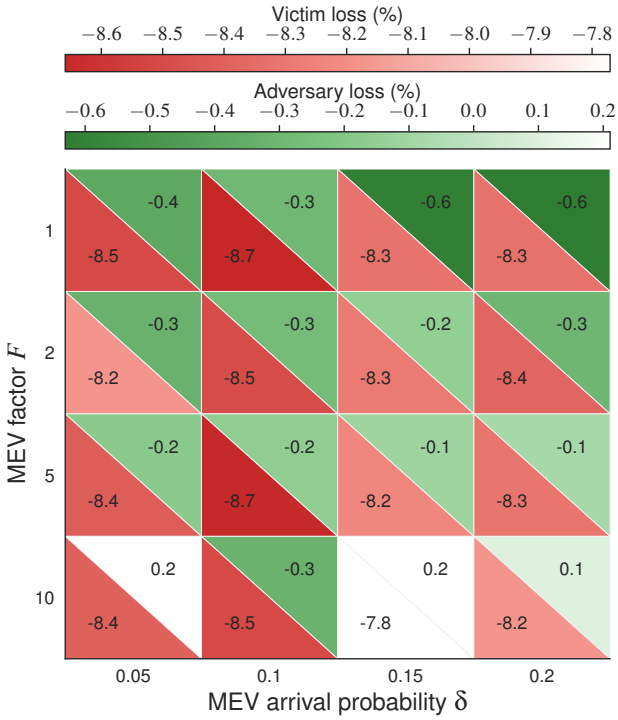


Figure 13: Heatmap of the impact of the learned pure-griefing strategy under varying MEV arrival probability δ and additional value F.

which we round to 4% in the main simulation.

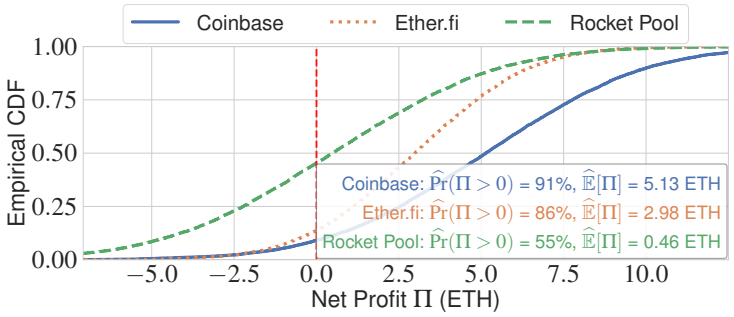
G.2 Alternative market calibrations

To test whether the main conclusion depends on the main 60-day calibration or favorable execution conditions, we repeat the profit simulation using the 90-day estimates from Table 4 and larger slippage bounds. These checks test whether profitability is driven by the longer return horizon or by overly optimistic execution assumptions.

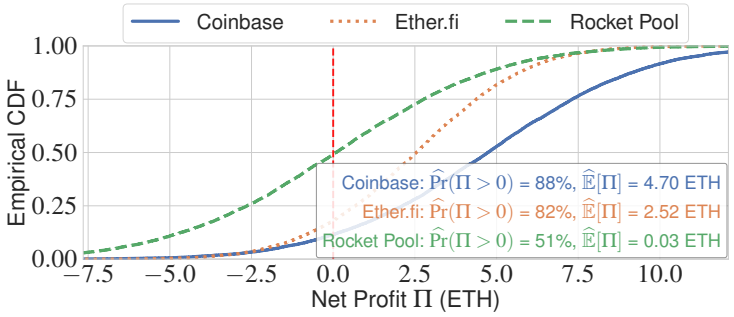
Under the 90-day calibration, the sensitivity coefficient $\hat{\beta}_H$ is 0.5472 for Coinbase, 0.3877 for Ether.fi, and 0.1296 for Rocket Pool, with corresponding σ_H values of 0.0080, 0.0054, and 0.0077. Under the 60-day calibration, $\hat{\beta}_H$ changes to 0.3587, 0.2564, and 0.1389, with corresponding σ_H values of 0.0073, 0.0052, and 0.0076.

Figure 14 shows that increasing slippage shifts the profit distributions left across all three LSTs, indicating lower expected profits and lower probabilities of profitable outcomes. However, the expected profit remains positive under both higher-slippage regimes. Coinbase and Ether.fi remain clearly profitable, with only moderate reductions in both expected profit and profit probability. Rocket Pool is more sensitive to execution frictions: its expected profit becomes close to zero under the 0.2% slippage bound, indicating a smaller profit margin.

As shown in Figure 15, under the 90-day calibration, Coinbase and Ether.fi exhibit higher expected profits than under the 60-day baseline across all slippage regimes. By contrast, Rocket Pool remains more sensitive to execution frictions, with negative expected profit under all three slippage bounds. This suggests that a longer calibration horizon mainly



(a) 0.1% slippage



(b) 0.2% slippage

Figure 14: Profit distributions under higher slippage bounds using the 60-day calibration.

strengthens profitability when the empirical performance–price relation is stronger, while pools with smaller margins remain vulnerable to execution frictions.

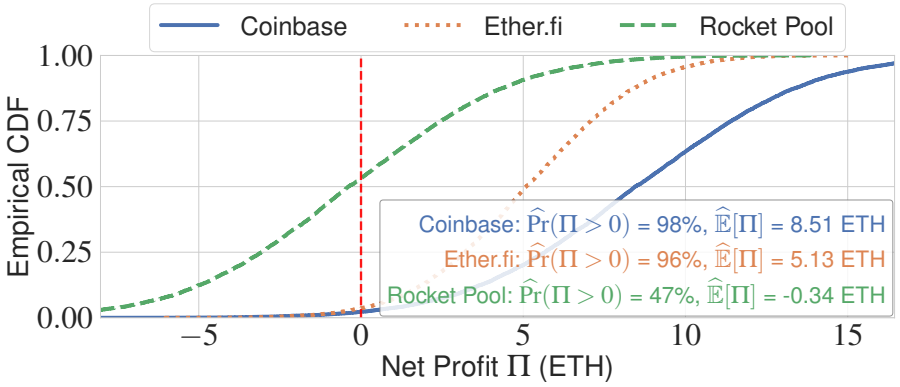
Overall, these results show that the attack is sensitive to market depth and the strength of the performance–price relation, as expected. Nevertheless, the qualitative conclusion remains robust: across alternative calibrations and higher slippage bounds, many configurations still yield positive expected profits. These profits often exceed honest staking rewards for the same capital, creating a strong economic incentive for the adversary.

H Proof of Theorem 7.1

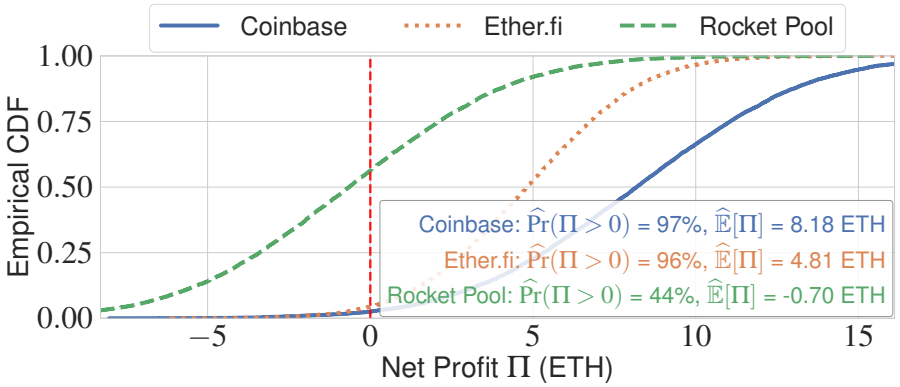
We restate and prove Theorem 7.1 below.

Theorem 7.1. *Consider accounts $\{1, \ldots, n\}$ participating in a leader election mechanism, where at least two are adversary-controlled. Denote the probability assigned by the mechanism for account i to be elected in slot s by $p_i(s)$. Suppose that after adversary-controlled account $a \in [n]$ performs a manipulation in slot s, the mechanism reduces a's election probability for some subsequent slot s' by $\Delta > 0$ without lowering the election probability of any other account: $p_a(s') = p_a(s) - \Delta < p_a(s)$ and $\forall i \neq a : p_i(s') \geq p_i(s)$. Then, without knowing the identities of all accounts' owners, the mechanism cannot simultaneously satisfy the following properties: (i) no adversary-controlled account has higher election probability in slot s', (ii) a leader is guaranteed to be elected in every slot: $\forall s : \sum_{i \in [n]} p_i(s) = 1$.*

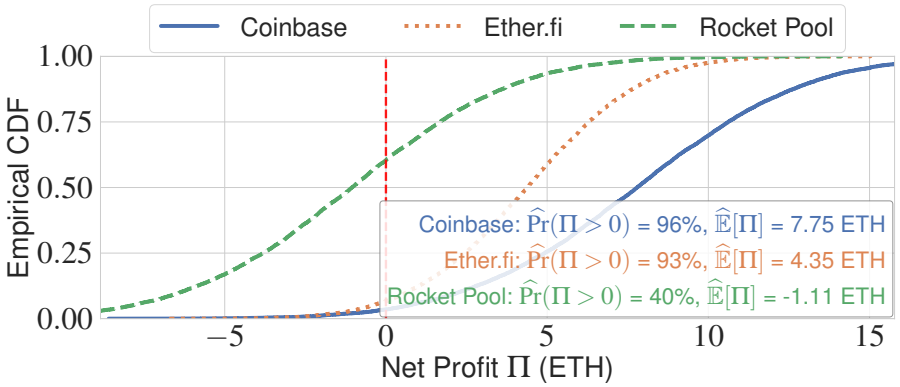
Proof. Assume towards contradiction that such a mechanism exists and consider two cases.



(a) 0.05% slippage



(b) 0.1% slippage



(c) 0.2% slippage

Figure 15: Profit distributions under the 90-day calibration across slippage bounds.

There exists some account $i \neq a$ such that $p_i(s') > p_i(s)$. Consider two executions that are identical from the mechanism's point of view and differ only in the ownership of account i. In one execution, i is controlled by an honest participant. In the other, i is controlled by the attacker. By assumption, the mechanism cannot distinguish these two executions, and therefore must assign the same election probabilities in both. It follows that any reassignment of election probability to i that appears safe in the first execution also benefits the attacker in the second, contradicting property (i).

There does not exist an account $i \neq a$ such that $p_i(s') > p_i(s)$. As the mechanism is assumed to not harm any account but a, then we can rule out the possibility that $\exists i \neq a$ with $p_i(s') < p_i(s)$. If so, it must be that $\forall i \neq a : p_i(s') = p_i(s)$, so we obtain:

$$\begin{aligned}\sum_{i\in[n]} p_i(s') &= p_a(s') + \sum_{i\neq a} p_i(s') \\ &= p_a(s) - \Delta + p_i(s) \\ &= \left(\sum_{i\in[n]} p_i(s)\right) - \Delta\end{aligned}$$

By the assumption that $\Delta > 0$:

$$\sum_{i\in[n]} p_i(s') = \left(\sum_{i\in[n]} p_i(s)\right) - \Delta < \sum_{i\in[n]} p_i(s)$$

Moreover, due to property (ii):

$$1 = \sum_{i\in[n]} p_i(s') < \sum_{i\in[n]} p_i(s) = 1$$

A contradiction is reached, thus completing the proof. □