Secret Leader Election in Ethereum PoS: An Empirical Security Analysis of Whisk and Homomorphic Sortition under DoS on the Leader and Censorship Attacks

Tereza Burianová, Martin Perešíni, and Ivan Homoliak

Brno University of Technology, Faculty of Information Technology, Brno, Czech Republic
terezaburianova@seznam.cz, {iperesini,homoliak}@fit.vut.cz

Abstract. Proposer anonymity in Proof-of-Stake (PoS) blockchains is a critical concern due to the risk of targeted attacks such as malicious denial-of-service (DoS) and censorship attacks. While several Secret Single Leader Election (SSLE) mechanisms have been proposed to address these threats, their practical impact and trade-offs remain insufficiently explored. In this work, we present a unified experimental framework for evaluating SSLE mechanisms under adversarial conditions, grounded in a simplified yet representative model of Ethereum's PoS consensus layer. The framework includes configurable adversaries capable of launching targeted DoS and censorship attacks, including coordinated strategies that simultaneously compromise groups of validators. We simulate and compare key protection mechanisms – Whisk, and homomorphic sortition. To the best of our knowledge, this is the first comparative study to examine adversarial DoS scenarios involving multiple attackers under diverse protection mechanisms. Our results show that while both designs offer strong protection against targeted DoS attacks on the leader, neither defends effectively against coordinated attacks on validator groups. Moreover, Whisk simplifies a DoS attack by narrowing the target set from all validators to a smaller list of known candidates. Homomorphic sortition, despite its theoretical strength, remains impractical due to the complexity of cryptographic operations over large validator sets.

Keywords: Blockchain · Ethereum · Consensus · Proof-of-Stake · Cryptography · Proposer Protection · Secret Single Leader Election · Attack Vectors · Denial-of-Service · Censorship · Whisk · Homomorphic Sortition · Simulation.

1 Introduction

The transition of Ethereum from a Proof-of-Work (PoW) [31] to a PoS [19] consensus mechanism marked a significant milestone in the evolution of blockchain technology. Unlike PoW systems, where computational effort is used to determine block proposers, PoS selects proposers based on staked value, aiming to

reduce energy consumption and increase scalability. However, this architectural shift introduces new attack surfaces, particularly concerning the predictability and exposure of block proposers [36,8,18].

In Ethereum's PoS, validators are selected in advance to propose blocks using the RANDAO [13] randomness beacon. Although this design enables validators to prepare for their duties, it inadvertently exposes their schedule of block proposals to adversaries. Since proposer identities are publicly known prior to their assigned slots, attackers can target them with Denial-of-Service (DoS) attacks or apply censorship tactics [23,3]. These attacks can degrade network performance, cause economic loss, or even lead to validator centralization by selectively penalizing certain participants [23]. By targeting specific proposers who might include undesirable transactions (e.g., OFAC non-compliant relays in 2022 [28]), adversaries can enforce censorship, weaken liveness, or economically coerce validators, threatening the integrity of the network [3]. The visibility of proposers is particularly problematic in scenarios involving Maximum Extractable Value (MEV) [16], as it allows attackers to target block proposers from previous slots, expanding their MEV opportunities and gaining financial advantages [12].

A variety of mechanisms have been proposed to protect proposers in PoS-based systems, ranging from SSLE mechanisms [1] like Whisk [25] and Blind-and-Swap [5], to homomorphic sortition [17], VRF-based protocols such as Algorand [18], and decentralization through Distributed Validator Technology (DVT) [15]. While approaches like DVT, network-layer solutions [34], and SnSLE [4,18] face drawbacks related to complexity, latency, or compatibility, SSLE mechanisms offer protocol-level proposer anonymity with minimal assumptions, making it the most promising fit for Ethereum's consensus. Complementary mechanisms like Proposer-Builder Separation (PBS) [21,32] further mitigate incentives for targeting proposers, though they do not provide strict anonymity.

This paper focuses on two SSLE mechanisms: (1) Whisk [25], a shuffle-based SSLE protocol leveraging zero-knowledge proofs (ZKPs) and (2) homomorphic sortition [17], which leverages fully homomorphic encryption (FHE) for collaborative encrypted leader election. While conceptually different, both approaches share the goal of mitigating adversarial visibility of proposers.

To evaluate these mechanisms, we developed a simulation framework that models the consensus behavior of Ethereum [14] and integrates Whisk and homomorphic sortition. Several adversarial scenarios were designed to test the effectiveness of these protections under conditions such as DoS attacks and targeted censorship. The performance of each mechanism was assessed not only in terms of efficacy but also with respect to computational overhead and feasibility within the strict 12-second slot timing of Ethereum.

Contributions The contributions of this paper are as follows:

- We made a threat analysis of Ethereum's PoS proposal selection, identifying specific vectors such as block proposer censorship and DoS.
- We made a comparative study and simulation-based evaluation of two proposer protection mechanisms (Whisk and homomorphic sortition).

- We elaborated on results, trade-offs, future directions, including integration into the existing Ethereum protocol, and open challenges in scalability.

Organization The paper is structured as follows. Section 2 provides background on blockchain architecture, consensus mechanisms, and Ethereum's PoS. Section 3 describes attacks we dealt with. Section 4 describes the chosen protection mechanisms, focusing on their cryptographic foundations and operational logic. Section 5 outlines the simulation framework and the attack scenarios used for evaluation. Section 6 presents the results of our experiments. Section 7 discusses the results, their implications, and outlines directions for future research. Section 8 provides an overview of related work and Section 9 concludes the paper.

2 Background on Ethereum

Ethereum is a general-purpose, permissionless blockchain platform that supports decentralized applications and smart contracts [40]. It operates on a layered architecture composed of a network/data layer, consensus layer, execution layer and application layer. Communication between participants occurs over a peer-to-peer network, while consensus mechanisms ensure agreement on the global state of the chain.

Proof-of-Stake in Ethereum. In Ethereum's PoS consensus, validator identities are represented by a staked amount of Ether (ETH). Validators participate by proposing blocks, attesting to others' proposals, and finalizing the chain state. Instead of competing for block production using computational power (as in PoW), validators are deterministically selected to perform these duties based on a shared source of randomness and their staked value [19].

RANDAO and Proposer Selection. Proposer selection in Ethereum relies on RANDAO, a collaborative randomness beacon. With each block, proposers contribute unpredictable and verifiable values. For each epoch, these values are aggregated into a RANDAO mix and stored in the beacon state [13]. The randomness seed used for proposer selection is derived from the RANDAO mix two epochs prior, allowing validators to know in advance whether they are selected for the role. Since the proposer selection is deterministic and public, anyone can predict the proposer for any upcoming slot once the relevant RANDAO mix is available [6]. Such a transparency introduces a vulnerability: adversaries can target known proposers. Note that we will refer to this version of leader election as the status quo throughout the paper.

Slot and Epoch Timeline. Ethereum divides time into fixed intervals: slots (12 seconds) and epochs (32 slots). Each slot ideally includes a new block proposal by a designated validator and attestation duties by a validator subset. Validators are informed of roles two epochs in advance. Epoch-bound events include RANDAO mix updates, new committees, rewards, slashings, and Casper FFG [7] justification/finalization checks. This rigid timeline, shown in Figure 1, supports fast block finality and predictable scheduling.

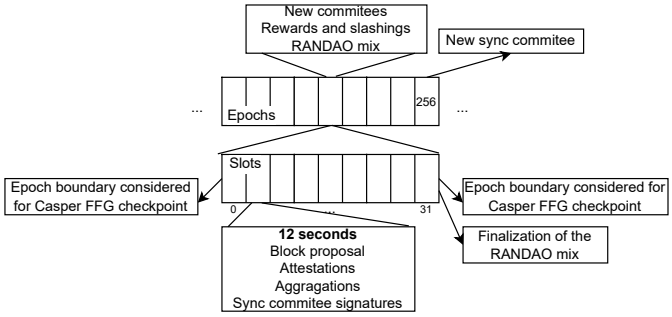


Fig. 1: An overview of the Ethereum consensus timeline.

3 Threat Model

Since proposer identities in Ethereum are deterministically derived and announced ahead of time, attackers can tailor targeted actions towards specific validators. The most relevant threats include:

- **Denial-of-Service (DoS) Attacks.** An attacker disrupts a validator's ability to propose a block by flooding its node with traffic. This can be highly effective if the attacker can identify the proposer in advance and link their identity to an IP address [23]. Some attacks do not require IP mapping but instead exploit known bugs or vulnerabilities in execution clients (e.g., Geth), allowing disruption of proposers at scale [29,11]. DoS attacks in Ethereum can take different forms depending on their target and strategy:
  1. A **large-scale DoS attack** targeting the Ethereum network as a whole, rather than individual proposers, potentially motivated by competing platforms seeking to undermine Ethereum, regulatory bodies attempting to restrict its operation, or actors aiming to manipulate cryptocurrency markets by eroding trust through network disruptions.
  2. A **targeted DoS attack** aimed at validators based on their assigned slots. A notable example occurs with MEV opportunities: by attacking validators in specific slots, such as those immediately before the attacker's own, an adversary can extend the MEV available to them while reducing fairness and decentralization [12].
- **Censorship.** Adversaries can prevent specific proposers from including certain transactions, particularly those involving sanctioned addresses [3].

These threats motivate stronger proposer protection mechanisms that preserve anonymity and limit such attacks [12].

4 Investigated Approaches

Several mechanisms for proposer protection in PoS protocols have been proposed, including VRF-based sortition as in Algorand [18], network-layer obfuscation such as Dandelion++ with RLN [34], and various SSLE schemes [1]. Algorand

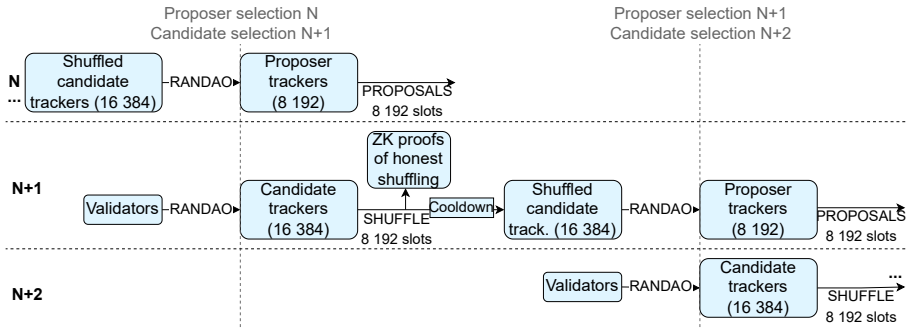


Fig. 2: The pipeline of the Whisk proposer protection mechanism.

hides proposer identities but selects multiple leaders per round, adding coordination overhead, while network-layer solutions require fundamental changes to Ethereum's stack. In contrast, SSLE mechanisms conceal the leader's identity until proposal time, unlike Ethereum's status quo of publicly revealing a leader in advance (see Section 2), and does so without major overhead or incompatibility. SSLE mechanisms differ in approach from shuffle-based anonymization to encrypted computation, and in this work we focus on two promising instances: Whisk [25] and homomorphic sortition [17].

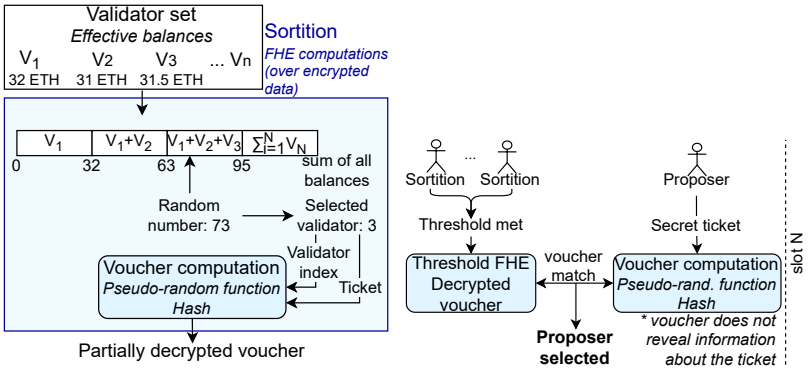
4.1 Whisk: Shuffle-Based Secret Leader Election

Whisk is an SSLE mechanism designed for Ethereum's PoS [25]. It conceals block proposers' identities until their assigned slot by shuffling a candidate list using verifiable random permutations and ZKPs. Operating in a pipelined mode (shown in Figure 2), current-round proposers prepare the shuffle for the next round, enabling scalability while preserving pre-slot anonymity.

Commitment and Anonymity. In Whisk, validators commit to a long-term secret k using a tracker (rG, krG), which can be re-randomized by third parties as $(zrG, zkrG)$ without linking to the original. The proposer proves ownership of the tracker using k and ZKPs, handled via Curdleproofs [38].

Candidate Selection and Shuffling. The process starts by selecting 16,384 candidates from the active validator set using RANDAO, whose identities are initially linkable since trackers are not yet re-randomized. Over 8,192 slots, proposers iteratively shuffle and re-randomize 128 candidate trackers per iteration, with ZKPs via Curdleproofs [38] ensuring correctness.

Proposer Selection. After the iterative shuffling is complete, a second round of RANDAO-based selection is used to choose 8,192 proposers from the shuffled and randomized candidate list, mapping them to the next 8,192 slots. Concurrently, a new candidate selection and shuffling phase is initiated to prepare the proposer set for the following round, ensuring a continuous pipeline of securely randomized leader selection (Figure 2).



(a) The sortition operation. (b) Proposer selection for a single slot.

Fig. 3: The processes used in the homomorphic sortition protocol.

4.2 Homomorphic Sortition: Encrypted Collaborative Selection

Homomorphic sortition is a cryptographic mechanism that leverages threshold fully homomorphic encryption (ThFHE) to perform proposer selection over encrypted data [17]. This approach ensures that proposers cannot be identified until a joint decryption is completed.

Sortition. In homomorphic sortition, leader election begins with sortition (shown in Figure 3a), where a prefix sum over participants' stakes forms weighted intervals. An encrypted random number is compared against this encrypted array, and the leader is selected based on the interval in which the number falls. All computations use fully homomorphic encryption (FHE) to preserve privacy. An encrypted voucher is then generated from the proposer's encrypted ticket.

Circuits and Parallel Computation. Homomorphic sortition relies on a set of encrypted circuits capable of performing basic computations–such as addition, comparisons, and hashing directly over ciphertexts. These circuits are optimized for parallel execution.

Verifiability and Voucher Matching. Once the voucher is decrypted by a threshold of participants and published, any validator claiming to be the elected leader can present a proof of ownership by locally computing the voucher using their secret ticket and demonstrating that it matches the revealed voucher without revealing any secret values, as shown in Figure 3b.

5 Simulation Framework & Implementation

To assess the viability of proposer protection mechanisms in Ethereum's PoS consensus, both Whisk and homomorphic sortition were implemented in a custom-built simulation framework. The simulation was designed to emulate Ethereum's block proposer workflow under various adversarial conditions, enabling both functional validation and security evaluation under attack scenarios.

5.1 Simulation Framework Design

The simulation framework closely mirrors the Ethereum consensus as described in its specifications [14] and partially in Section 2. Importantly, it is designed to be modular and extensible, enabling integration of additional proposer anonymity mechanisms for future evaluation.

Simulation Parameters. The number of validators and the duration of the simulation are configurable, allowing for flexible experimentation. The framework supports running validators under both the default Ethereum proposer selection mechanism and the two implemented protection schemes. Furthermore, the framework models adversarial behavior through configurable attack modules capable of launching denial-of-service and censorship attacks, allowing for controlled testing of each mechanism's resilience. The attack configuration includes parameters that define how likely an attacker is to correctly link a validator's identity to their IP address and how well each validator is protected against attacks. By adjusting these, the model can represent different attacker strategies – such as those relying on standard DoS methods with uncertain targeting, more sophisticated attackers exploiting client vulnerabilities [29,11], or adversaries with varying levels of resources and precision.

Simulation Model Design. The simulation framework was designed to be as simple as possible while still retaining the functionality required for a meaningful analysis. The final model captures the essential elements of Ethereum's consensus layer: honest validators proposing blocks according to the specification, slot and epoch processing (including missed slots), proposer selection, basic block proposal processing and RANDAO.

For clarity and focus, we abstract away components of Ethereum consensus and networking that are not relevant to our objectives. In particular, this includes details of the communication model, such as bandwidth limitations, latencies, and message propagation. For the same reason, we exclude finality, fork choice, transactions and their interaction with the execution layer, balance changes (rewards and slashing), deposits and exits. By abstracting from the network layer, we obtain a synchronous consensus model that isolates the effect of proposer protection mechanisms at the consensus level. Detailed images illustrating the framework's design and implementation are shown in Section A.

5.2 Simulation Framework Implementation

We implemented a simulation framework in Rust, leveraging its performance and rich ecosystem of Ethereum-related libraries. Core beacon structures were abstracted using generic traits to support multiple proposer selection mechanisms and attacks within a unified simulation environment. The implementation includes support for cryptographic primitives such as BLS signatures, with serialization handled via `ssz_rs` to preserve the Ethereum consensus structure compatibility. The framework accepts configuration via JSON, allowing flexible

specification of simulation parameters and outputs detailed results for analysis. We provide the source code for our implementation at https://anonymous.4open.science/r/leader-election-sim-anon-FCFD/.

Implementation of Whisk. The Whisk protocol was implemented with its full pipeline design, where each round of proposers performs the shuffling for the next round of proposer selection, based on the Whisk EIP-7441 specification draft [39]. The number of validators selected for each phase was proportionally reduced to handle reduced validator set size.

Validators were represented using trackers which could be re-randomized, producing unlinkable identities. These could later be claimed using a proof that connects the randomized tracker to their original long-term commitment. The simulation included ZKPs to verify honest shuffles. This was achieved by incorporating an existing Curdleproofs implementation [24] into the pipeline. The pipelined structure was successfully tested, ensuring that the protocol could maintain continuous proposer assignment while preserving pre-slot anonymity.

Implementation of Homomorphic Sortition. Homomorphic sortition was implemented using simplified Fully Homomorphic Encryption (FHE) constructs, eliminating the threshold functionality to simulate the behavior of a single proposer. Each validator was represented by a ticket, a random encrypted number. A list with intervals based on validators' effective balances was created, and a voucher, representing the proof of the selected proposer, was computed based on the random selection from the list. The proposer then computes the voucher without encrypted operations, comparing it to the now decrypted voucher computed in the sortition process.

The implementation focused on maintaining the structure and computational steps of the protocol. To implement the FHE circuits, TFHE-rs by Zama [41] was used. To implement the pseudo-random function and the hash, a simplified Prince block cipher [2] was implemented. A hash function based on Hirose's construction [22], utilizing two instances of the PRINCE block cipher, was used to compute the hash. There are two options to launch the homomorphic sortition simulation. The simplified version omits the voucher verification and uses the pseudorandomness provided by the TFHE-rs implementation for longer simulations. The full version utilizes the implemented cipher and performs all checks. This version serves as a frame for functionality checks and cost measurements.

Model Validation. We validated the functionality of the implemented models by running baseline experiments (i.e., without attacks). These confirmed that proposers were correctly assigned to their slots, the selection process remained fair, and validator identifiers were linkable to their identities under the status quo. In contrast, under the implemented protection mechanisms, the identifiers remained secure and unlinkable to the validators' real identities.

5.3 Simulated Attack Types

This section describes how the threats detailed in Section 3 are incorporated into the framework implementation.

DoS Attack. Our simulation framework implements several potential attack vectors targeting Ethereum proposers. The implemented **targeted Denial-of-Service** (DoS) attack aims to disrupt block production by targeting validators based on identified IP-public key pairs, with success rates influenced by correct pairing probability [8] and individual node protection measures like VPNs, firewalls or the more advanced `sentry` nodes [33,23]. An **advanced DoS** variant circumvents the proposer identification by indiscriminately attacking a subset of validators, thereby enabling scalable disruption.

Censorship Attack. In the censorship attack, we selectively target proposers belonging to a victim set, performing DoS only when a victim is scheduled to propose a block. This strategy also includes an **advanced censorship** variant, where the attackers continuously DoS validators from the victim group, irrespective of their immediate proposer status. These configurable attack modes enable the precise simulation of targeted disruptions and support the assessment of proposer protection mechanisms under diverse threat scenarios.

6 Evaluation

Experiments were conducted to model a DoS scenario and a censorship scenario in which the attacker was able to correctly associate 80 % of validators with their IP addresses, while 20 % of validators had additional protection mechanisms in place against such attacks. The simulation was run with a reduced validator set – 1,000 validators for Whisk and 100 validators for homomorphic sortition. We simulated a resourceful attacker capable of targeting 10 % of the validator set using an advanced DoS strategy. In the censorship scenario, 10 % of the validators were designated as censorship targets. The duration of the simulation was 6 epochs with several randomized runs for more statistically significant results.

6.1 Security Effectiveness

DoS Attacks on Leader. The results of the simulation experiments indicate that while current proposer protection mechanisms, such as Whisk and homomorphic sortition, offer promising improvements over the default Ethereum selection, they are not universally robust against all forms of attack. Figure 4 presents data from a single simulation instance, serving as an illustrative example of possible effects. The figure shows that the targeted DoS attack severely impacts block proposal rates when no protection is in place, with 64 % of blocks missed and 67 % of proposers affected. Both Whisk and homomorphic sortition mitigate targeted attacks by obscuring the identity of the proposer until slot execution,

(a) Targeted DoS (no protection). (b) Targeted DoS (with protection).

(c) Advanced DoS using Whisk. (d) Advanced DoS using h. sortition.

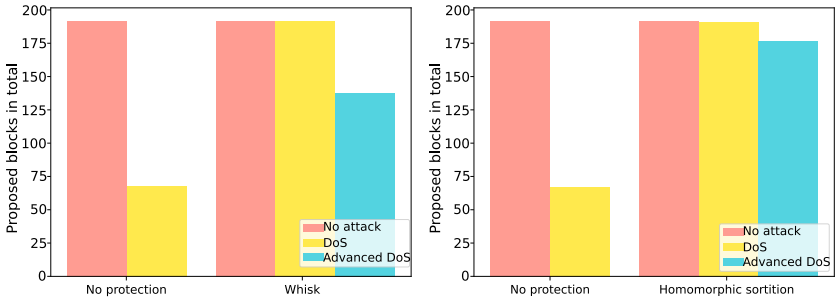
Fig. 4: Comparison of beacon chain activity under different proposer protection mechanisms and attack scenarios, where green indicates successfully proposed blocks and red indicates missed slots.

reducing the impact of the targeted DoS to a negligible level close to 0 %. However, advanced DoS attacks remain problematic, particularly for Whisk, which introduces a vulnerability in its candidate selection phase. Because the candidate set is known in advance and the candidates' identities are not concealed at this stage, an attacker with sufficient resources can target a large number of candidates, leading to partial or complete disruption. This makes the advanced DoS attack potentially more damaging for Whisk, resulting in 28 % of blocks missed, compared to 6–8 % under no protection or homomorphic sortition, where the attacker selects victims from the entire validator set.

Figure 5 highlights the difference in the number of successfully proposed blocks under different protection mechanisms and attack scenarios, providing a statistical view of the system's behavior. For the "No protection" and "Whisk" configurations, results are aggregated over five simulation runs with different random seeds. Due to the significantly higher computational cost of homomorphic sortition, that configuration was simulated only once. Whisk significantly reduces the success rate of standard DoS attacks, but its effectiveness declines under advanced adversarial conditions. Homomorphic sortition performs similarly to the status quo during advanced DoS since the computations are performed on the whole set of validators, indicating that while it introduces no new vulnerabilities, it also does not improve resilience. In the case of targeted DoS, the attack was only effective when the adversary randomly guessed the proposer and successfully eliminated them – a strategy with negligible impact at realistic validator scales, such as one million participants.

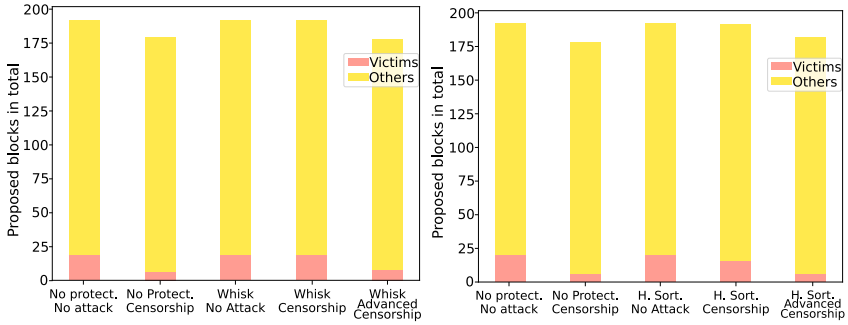
Censorship Attack. Figure 6 provides a statistical overview of the outcomes, comparable to the structure of the DoS attack analysis in Figure 5. However, in contrast to DoS, where the impact is readily observable through increased missed slots, censorship manifests more subtly and requires deeper analysis of slot behavior and victim targeting.

In terms of observable proposal activity, the difference between scenarios appears negligible. Only around 6 % to 7 % of slots are missed during the censorship



(a) Whisk. (b) Homomorphic sortition.

Fig. 5: Comparison of the number of proposed blocks using different selection mechanisms under various circumstances during a DoS attack.



(a) Whisk. (b) Homomorphic sortition.

Fig. 6: Comparison of the number of proposed blocks using different selection mechanisms under various circumstances during a censorship attack.

attack, a deviation that could be attributed to typical validator inactivity, network delays, or benign causes. This makes censorship nearly indistinguishable from regular network fluctuations when viewed at a system-wide level.

However, when examining the effect on the censorship victims set, a stark contrast emerges: approx. 69 % to 71 % of targeted validators are prevented from proposing their blocks. This highlights the effectiveness of the censorship strategy, although its footprint on overall slot availability remains minimal.

Both Whisk and homomorphic sortition demonstrated strong protection against the targeted censorship strategy by concealing proposer identities until the slot execution. As a result, targeted censorship was rendered largely ineffective under these mechanisms. In contrast, under the advanced censorship strategy, where the attacker persistently targets the entire victim group regardless of their current proposer status, the protective effect diminishes and the outcomes remain close to the no-protection baseline, indicating that the protection is insufficient.

Table 1: Time measurements of various consensus processing phases.

		Block Processing [ms]			Epoch processing [ms]
Mechanisms		**10 val.**	**100 val.**	**1 000 val.**	**-**
Base	avg.	8	10	13	<1
	max.	12	13	19	<1
Whisk	avg.	261 10 (cooldown)	264 10 (cooldown)	263 10 (cooldown)	<1 171 (shuffle end)
	max.	263 10 (cooldown)	266 10 (cooldown)	265 10 (cooldown)	<1 185 (shuffle end)
H. Sortition	avg.	277 375	513 750	2 013 006	441
	max.	278 594	518 697	2 100 439	442

6.2 Cost

To evaluate the practical feasibility of the mechanisms, measurements were taken for block processing time, epoch processing time, and block size, shown in Table 1. The results show that Whisk is approx. 20 to 30 times slower than the base selection due to cryptographic operations and periodic epoch shuffling; however, the overall duration remains generally negligible. In contrast, homomorphic sortition exhibited a dramatic increase in processing time, with costs rising steeply alongside the number of validators even with parallelization, making it unsuitable for Ethereum's scale. While Whisk introduces moderate overhead, the extreme computational demands of homomorphic sortition render it impractical with current technology. In terms of network latency, block transmission remains negligible for Whisk and the base mechanism, but increases significantly for homomorphic sortition due to the large block size, especially in possible mechanism modifications selecting multiple proposers in advance, and the need for extended information sharing among participants required by the threshold FHE process.

7 Discussion and Future Work

Our findings underscore that while protection mechanisms increase the cost and complexity of an attack, they do not guarantee immunity. In particular, mechanisms that rely on a known candidate set are inherently vulnerable to attackers with sufficient foresight and resources. Additionally, client vulnerabilities, such as flaws in Ethereum execution clients [29], may reduce the effort needed to perform advanced DoS attacks, bypassing the requirement of IP-identity mapping.

In summary, while Whisk and similar SSLE mechanisms show clear security benefits, they must be treated as a layer of deterrence rather than a foolproof defence. Their presence makes long-term attacks more expensive, less practical, and easier to detect over time. Moreover, by hiding the identity of proposers until the slot, these mechanisms significantly hinder or outright prevent MEV-based attacks, which typically rely on advance targeting and precise timing–conditions

Table 2: Comparison of Proposer Protection Mechanisms

Mechanism	DoS Protection	Computation Cost	Real-Time Feasibility	Cons
Whisk	Moderate–High	Low	High	Known small candidate set
Homomorphic Sortition	Very High	Very High	Low	Scalability, latency

that are difficult to meet without executing a targeted DoS. However, more robust protection is still needed in scenarios where attackers can invest long-term resources and remain covert.

While both mechanisms improve proposer anonymity, they differ significantly in computational cost. Whisk introduces moderate overhead (approx. 20–30 times the status quo) yet remains feasible within Ethereum's timing constraints. In contrast, homomorphic sortition incurs extreme processing delays and scalability issues, making it impractical for real-world deployment without substantial optimization. However, efficiency could be improved by electing multiple leaders in advance using the Secret Leader Permutation (SLP) variant of the protocol, which could reduce the need to perform a full SSLE round for every slot. A high-level overview of these mechanisms and their properties is shown in Table 2.

Future Work. Future research should focus on attempting to mitigate the vulnerability of the known candidate list in Whisk that makes Ethereum even more prone to large-scale DoS attacks. Integrating network-layer anonymization or adaptive shuffling strategies may help improve unpredictability. Additionally, refining the performance of homomorphic sortition–potentially via circuit optimizations or hybrid encryption schemes–could enable its real-world adoption. Given that some attacks exploit client-level vulnerabilities, future work should also investigate validator client hardening and the incorporation of failover strategies at the software level. Future work should also explore adapting the protocol for Ethereum by using the SLP variant to preselect multiple leaders once and investigating the use of committees to improve the scalability of threshold FHE operations. Lastly, expanding the simulation framework to include more complex adversarial behavior and a network layer view would provide a more holistic view of protection mechanisms in adversarial environments.

8 Related Work

We organize the related work into four categories, described below.

DoS Attacks & Privacy. Previous studies have speculated on the feasibility and implications of validator deanonymization and targeted attacks in Ethereum's PoS consensus. Bürgel et al. [8] highlighted the realistic threat of validator sniping, demonstrating that IP addresses can be linked to public keys with

high accuracy, and that minimal downtime is sufficient for a successor validator to extract MEV rewards. Similarly, the "Packetology" analysis [36] provided empirical evidence that network-layer data leaks enable associating validator indices with specific peer identities using timing patterns and gossip propagation. These works emphasized the potential for targeted DoS attacks and stressed the need for privacy-preserving mechanisms.

SSLE Approaches. Whisk [25] introduces a shuffle-based SSLE mechanism for Ethereum that conceals proposers until block creation. It preserves strong anonymity despite partial node compromise and mitigates RANDAO [13] biasing attacks. Whisk counters selling attacks, where validators prove future roles to MEV searchers, without adding complexity. Its bandwidth and latency overhead are modest, enabling feasible deployment. The authors compare Whisk with Dandelion++ [34] and earlier SSLE mechanisms, noting these often suffer from excessive overhead or weak MEV resistance. Whisk is tailored to Ethereum's architecture, achieving high deployability with minimal coordination. Their evaluation positions Whisk as a leading SSLE candidate. Buterin's Simplified SSLE mechanism [5] uses repeated size-2 blind-and-swap operations for low-overhead proposer privacy. It avoids complex cryptography and effectively obscures proposer identities under ideal conditions. Khovratovich [26] shows the scheme's anonymity degrades in adversarial settings, where swaps can be traced or disrupted. Compared to Whisk, it is more vulnerable and requires more rounds to match privacy. The Ouroboros [27] family, underlying Cardano [10], offers SSLE via private cryptographic lotteries. Each validator privately checks slot eligibility based on stake and public randomness. This prevents targeted attacks without public leader announcements, using cryptographic sortition instead of shuffling.

Homomorphic sortition [17] enables encrypted leader selection to enhance proposer privacy without early identity disclosure. Though promising in theory, it remains unanalyzed or unadapted for Ethereum. Its impact on Ethereum's fork-choice rule, latency, and network design is still unexplored. LAKSA [35] targets secure, high-throughput PoS via stochastic leader election and committee confirmation. Inspired by Algorand [18], DFINITY, and Randhound [20,37], it enhances lightweight voting. Randomly sampled committees periodically vote on chain views [35], improving robustness and scalability.

SnSLE Approaches. Secret Non-Single Leader Election (SnSLE) protocols, such as Algorand [18], introduce inherent proposer privacy by allowing multiple potential proposers to be selected for a given slot, through the use of Verifiable Random Functions (VRFs) [30]. However, this approach requires an additional round to determine the best block proposer from eligible candidates, which introduces additional communication overhead with linear complexity. Vitalik Buterin's SnSLE proposal [4] explores an alternative to SSLE based on probabilistic multi-proposer selection, inspired in part by Algorand [18]. In this approach, each validator independently learns if she is eligible to propose in a given slot, offering proposer privacy without requiring shuffling or encryption. While conceptually simpler, the design brings significant practical trade-offs. As noted by Whisk

authors [25], non-single leader election complicates fork-choice rules, makes the protocol vulnerable to MEV time-buying attacks, and significantly increases networking overhead, especially in a sharded environment. Algorand addresses these challenges with protocol-level optimizations like smart gossip and timeouts but adapting such solutions to Ethereum would be nontrivial.

Network-Level Anonymization. A proposal to enhance validator anonymity using Dandelion++ and RLN [34] aimed to obfuscate the origin of consensus messages by routing them through a private sub-network. While conceptually promising, the final analysis concluded that the approach is not feasible for Ethereum's consensus layer due to strict latency constraints and high implementation complexity, especially under the tightened timing rules. The trade-offs ultimately limited its applicability despite its strong anonymity model.

Polkadot's Sassafras [9] combines an SSLE-based leader election with network-layer anonymity. It complements consensus-layer protection with mechanisms to conceal proposer network traffic, offering a more comprehensive defense against deanonymization and targeted attacks. This strategy of providing privacy at the network level while selecting a single leader at the consensus layer is also utilized by other protocols, such as LAKSA [35].

Comparison with Our Work. While prior work [25,26,5] has provided brief analyses and comparisons of individual proposer privacy mechanisms, there has not yet been a unified framework capable of evaluating multiple approaches under adversarial conditions. Our work addresses this gap by introducing a simplified Ethereum consensus model that enables systematic testing and comparison of SSLE mechanisms in hostile environments.

9 Conclusion

This work introduced a modular simulation framework for evaluating proposer privacy mechanisms in Ethereum under adversarial conditions. Using a simplified yet representative consensus model, we analyzed two SSLE mechanisms, Whisk and homomorphic sortition, against targeted and coordinated attacks. While both mechanisms show resilience to simple DoS attacks, neither withstands coordinated adversarial strategies and each presents practical limitations. Whisk introduces a vulnerability in which a subset of validators, known as the candidate set, is publicly exposed, making advanced attacks easier. In contrast, homomorphic sortition operates over the entire validator set, providing stronger security, but the computational requirements are currently too demanding for practical deployment. These findings offer a clearer understanding of the limitations and trade-offs in current SSLE designs and support a more informed development of proposer privacy mechanisms for Ethereum. The framework's extensibility enables future testing of additional designs, supporting continued development of effective and deployable proposer anonymity solutions for Ethereum.

References

1. Boneh, D., Eskandarian, S., Hanzlik, L., Greco, N.: Single secret leader election. Cryptology ePrint Archive, Paper 2020/025 (2020), https://eprint.iacr.org/2020/025
2. Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E.B., Knežević, M., Knudsen, L.R., Leander, G., Nikov, V., Paar, C., Rechberger, C., Rombouts, P., Thomsen, S.S., Yalçın, T.: PRINCE - a low-latency block cipher for pervasive computing applications (full version). Cryptology ePrint Archive, Paper 2012/529 (2012), https://eprint.iacr.org/2012/529
3. Brownworth, A., Durfee, J., Lee, M.J., Martin, A.: Regulating decentralized systems: Evidence from sanctions on Tornado Cash. Staff Reports 1112, Federal Reserve Bank of New York (Aug 2024). https://doi.org/10.59576/sr.1112, https://ideas.repec.org/p/fip/fednsr/98635.html
4. Buterin, V.: Secret non-single leader election (2022), https://ethresear.ch/t/simplified-ssle/12315
5. Buterin, V.: Simplified SSLE (2022), https://ethresear.ch/t/simplified-ssle/12315
6. Buterin, V.: Vitalik's annotated eth2 spec (2024), https://github.com/ethereum/annotated-spec/
7. Buterin, V., Griffith, V.: Casper the friendly finality gadget (2019), https://arxiv.org/abs/1710.09437
8. Bürgel, S.: Proof-of-stake validator sniping research (2022), https://medium.com/hoprnet/proof-of-stake-validator-sniping-research-8670c4a88a1c
9. Caracheo, A., Crites, E., Shirazi, F.: Sassafras part 1: A novel single secret leader election protocol (2022), https://research.web3.foundation/Polkadot/protocols/block-production/Sassafras-Part-1
10. Cardano Foundation: Cardano blockchain platform (2017), https://cardano.org/, accessed 20-07-2025
11. Department Of DOGE Efficiency: Public Disclosure of "DogeReaper", a critical vulnerability in Dogecoin (2024), https://x.com/EfficiencyDOGE/status/1864357823163060316
12. Drake, J.: Justin Drake - MEV & Cryptography (Flashbots Research Talks) (2022), https://www.youtube.com/watch?v=jLHf6yw7b5Y
13. Edgington, B.: Randomness. In: Upgrading Ethereum, pp. 147–166. 6517e04 edn. (2025), https://eth2book.info/latest/book.pdf, [Accessed 15-07-2025]
14. Ethereum: Ethereum proof-of-stake consensus specifications (Deneb) (2025), https://github.com/ethereum/consensus-specs/tree/v1.5.0/specs/
15. Ethereum.org: Distributed validator technology (2023), https://ethereum.org/en/staking/dvt/
16. Ethereum.org: Maximal extractable value (MEV) (2025), https://ethereum.org/en/developers/docs/mev/
17. Freitas, L., Tonkikh, A., Bendoukha, A.A., Tucci-Piergiovanni, S., Sirdey, R., Stan, O., Kuznetsov, P.: Homomorphic sortition – secret leader election for PoS blockchains (2023)
18. Gilad, Y., Hemo, R., Micali, S., Vlachos, G., Zeldovich, N.: Algorand: Scaling byzantine agreements for cryptocurrencies. Cryptology ePrint Archive, Paper 2017/454 (2017), https://eprint.iacr.org/2017/454
19. Grandjean, D., Heimbach, L., Wattenhofer, R.: Ethereum proof-of-stake consensus layer: Participation and decentralization (2023), https://arxiv.org/abs/2306.10777

20. Hanke, T., Movahedi, M., Williams, D.: Dfinity technology overview series, consensus system. arXiv preprint arXiv:1805.04548 (2018)
21. Heimbach, L., Kiffer, L., Torres, C.F., Wattenhofer, R.: Ethereum's proposer-builder separation: Promises and realities (2023), https://arxiv.org/abs/2305.19037
22. Hirose, S.: Provably secure double-block-length hash functions in a black-box model. In: Park, C.s., Chee, S. (eds.) Information Security and Cryptology – ICISC 2004. pp. 330–342. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
23. Kadianakis, G.: How I learned to stop worrying about the DoS and love the chain (2022), https://ethereum-magicians.org/t/how-i-learned-to-stop-worrying-about-the-dos-and-love-the-chain/9941
24. Kadianakis, G., Bhardwaj, N., kevaundray, dapplion: Curdleproofs: A shuffle argument protocol (c2025), https://github.com/asn-d6/curdleproofs/
25. Kadianakis, G., Drake, J., Feist, D., Herold, G., Khovratovich, D., Maller, M., Simkin, M.: Whisk: A practical shuffle-based SSLE protocol for Ethereum (2022), https://ethresear.ch/t/whisk-a-practical-shuffle-based-ssle-protocol-for-ethereum/11763
26. Khovratovich, D.: Analysis of Swap-or-Not SSLE proposal (2022), https://ethresear.ch/t/analysis-of-swap-or-not-ssle-proposal/12700
27. Kiayias, A., Russell, A., David, B., Oliynykov, R.: Ouroboros: A provably secure proof-of-stake blockchain protocol. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology – CRYPTO 2017. pp. 357–388. Springer International Publishing, Cham (2017)
28. Labrys: MEV Watch, https://www.mevwatch.info/, [Accessed 13-09-2025]
29. McSheehan, K.: Remote Geth P2P DoS (Oct 2024), https://envadr.io/research/report_eth_dos.pdf
30. Micali, S., Vadhan, S., Rabin, M.: Verifiable random functions. In: Proceedings of the 40th Annual Symposium on Foundations of Computer Science. p. 120. FOCS '99, IEEE Computer Society, USA (1999)
31. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Tech. rep., bitcoin.org (2008)
32. Neuder, M., Drake, J.: Why enshrine Proposer-Builder Separation? A viable path to ePBS (2023), https://ethresear.ch/t/why-enshrine-proposer-builder-separation-a-viable-path-to-epbs/15710
33. Nielson, B.: Hardening blockchain nodes against attack (2023), https://theblockchainacademy.com/hardening-blockchain-nodes-against-attack, [Accessed 09-09-2025]
34. PSE team: Ethereum consensus layer validator privacy and feasibility analysis using Dandelion++ and RLN (2022), https://www.notion.so/Ethereum-consensus-layer-validator-privacy-and-feasibility-analysis-using-Dandelion-and-RLN-4674432febdc43979a67f043961442e6
35. Reijsbergen, D., Szalachowski, P., Ke, J., Li, Z., Zhou, J.: LaKSA: A probabilistic proof-of-stake protocol. arXiv preprint arXiv:2006.01427 (2020)
36. Rhea, J.: Packetology: Validator privacy (2020), https://ethresear.ch/t/packetology-validator-privacy/7547
37. Syta, E., Jovanovic, P., Kogias, E.K., Gailly, N., Gasser, L., Khoffi, I., Fischer, M.J., Ford, B.: Scalable bias-resistant distributed randomness. In: 2017 IEEE Symposium on Security and Privacy (SP). pp. 444–460. Ieee (2017)
38. The Ethereum Foundation Cryptography Research Team: Curdleproofs: A shuffle argument protocol (2022), https://github.com/asn-d6/curdleproofs/blob/main/doc/curdleproofs.pdf

39. Traglia, J., Kadianakis, G., Drake, J., Bhardwaj, N.: EIP-7441 draft (2025), https://github.com/dapplion/consensus-specs/tree/dev/specs/_features/eip7441
40. Wood, G., et al.: Ethereum: A secure decentralised generalised transaction ledger. Ethereum project yellow paper **151**(2014), 1–32 (2014)
41. Zama: TFHE-rs: A Pure Rust Implementation of the TFHE Scheme for Boolean and Integer Arithmetics Over Encrypted Data (2022), https://github.com/zama-ai/tfhe-rs

A Framework Implementation Details

A.1 Status Quo

Figure 7 describes the implementation of the simulation framework design as per the current consensus specification [14] with the structure of the beacon state and the beacon block shown in Figure 8.

A.2 Whisk

Figure 9 and Figure 10 show the changed framework, beacon state and beacon block structure in Whisk implementation as per the Whisk specification draft [39].

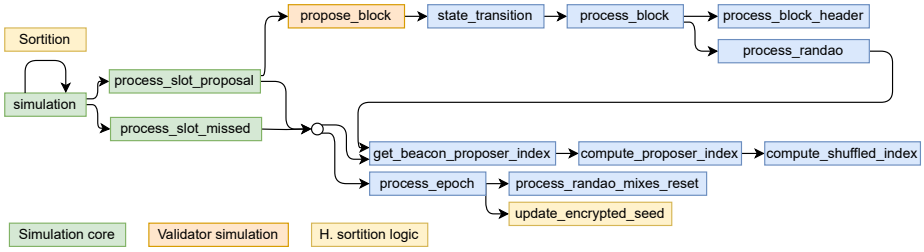


Fig. 7: The function call hierarchy in the consensus simulation as per the current consensus specification [14].

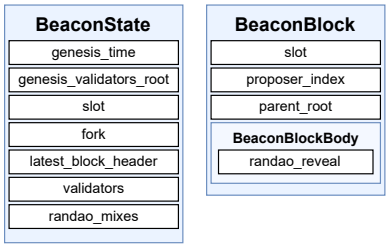


Fig. 8: The structure of the simplified beacon state and beacon block as per the current consensus specification [14].

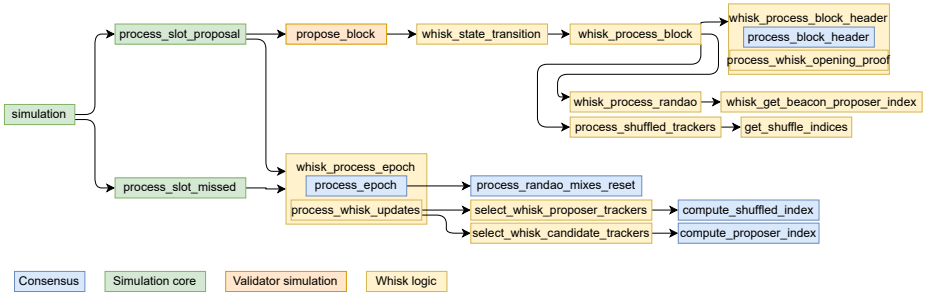


Fig. 9: The function call hierarchy in the consensus simulation as per the Whisk specification [39].

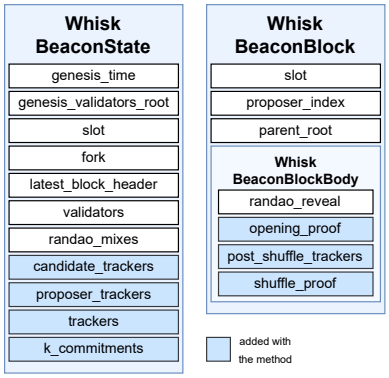


Fig. 10: The structure of the simplified beacon state and beacon block as per the Whisk specification [39].

A.3 Homomorphic Sortition

Figure 11 and Figure 12 show the changed framework, beacon state and beacon block structure in homomorphic sortition implementation, as designed for the simplified Ethereum consensus based on the article by Freitas et al. [17].

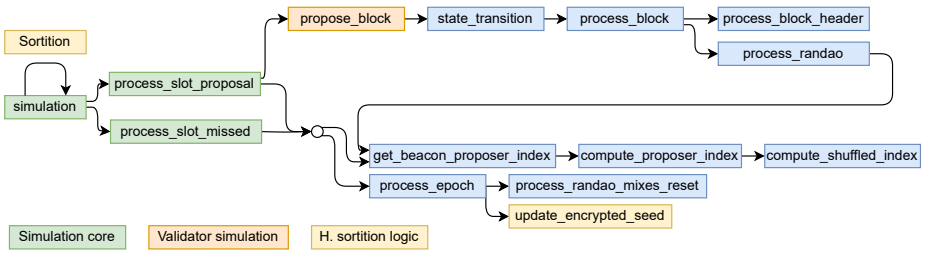


Fig. 11: The function call hierarchy in the consensus simulation using h. sortition.

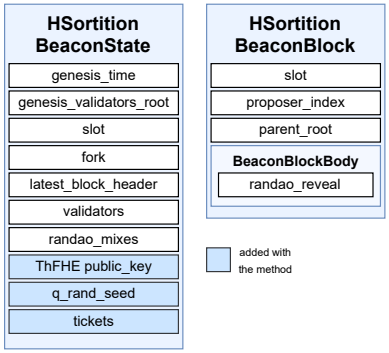


Fig. 12: The structure of the simplified beacon state and beacon block based on the own proposed homomorphic sortition design for Ethereum.