Excel@FIT 2025
http://excel.fit.vutbr.cz

# DoS-Resistant SSLE Protocols in Ethereum: WHISK and Homomorphic Sortition

Tomáš Krajčí

**Abstract**
Denial-of-Service (DoS) vulnerabilities threaten Ethereum's block producers, risking network disruption and losses. This paper analyzes WHISK and Homomorphic Sortition protocols as countermeasures. Simulations and analysis assess their effectiveness and overhead. Both protocols enhance security but introduce costs and remain penetrable, guiding future blockchain defenses.

*xkrajc25@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

## 1. Introduction

**[Motivation]** Ethereum's shift to Proof-of-Stake (PoS) enhances efficiency but exposes block producers to DoS attacks, which can halt block production, degrade performance, and impose economic losses. Securing these validators is vital for Ethereum's reliability and adoption.

**[Problem Definition]** The research tackles DoS attacks targeting Ethereum's block producers, aiming to obscure their identities until block proposal time. A solution must ensure proposer secrecy, fairness, and minimal network overhead, evaluated via security and performance metrics.

**[Existing Solutions]** WHISK and Homomorphic Sortition are Single Secret Leader Election (SSLE) protocols addressing this issue. WHISK uses shuffling to hide proposers, while Homomorphic Sortition employs encryption. Prior works [1, 2, 3] highlight their potential, yet overheads persist.

**[Our Solution]** This study analyzes WHISK and Homomorphic Sortition through simulations and theoretical evaluation, detailing their mechanisms, strengths, and limitations in protecting Ethereum validators.

## 2. WHISK Protocol

WHISK is a Single Secret Leader Election (SSLE) protocol designed to safeguard Ethereum validators from Denial-of-Service (DoS) attacks by concealing their identities until block proposal time. The protocol shuffles validator trackers across 8192 slots (one day) through a multi-phase process (Figure 2). Initially, a candidate list of 16,384 validator trackers is selected. These trackers are then shuffled using a Feistelshuffle algorithm, with zero-knowledge proofs ensuring the process remains honest. Finally, a proposer list of 8,192 validators is determined using RANDAO. Validators reveal their roles only when proposing a block, making targeted DoS attacks significantly harder.

### 2.1 Shuffling Mechanism in WHISK

The shuffling mechanism in WHISK ensures proposer secrecy through three distinct phases:

1. **Candidate Selection:** A list of 16,384 validator trackers is selected from the validator pool using public randomness from RANDAO. These trackers represent validator identities and form the initial candidate list.

2. **Shuffling Phase:** The candidate list is randomized over 8192 slots (i.e. one day) using a Feistelshuffle algorithm. The list is organized as a $128 \times 128$ matrix and shuffled across 64 rounds, with each round covering 128 slots. During each round: - A single row is shuffled per slot using private randomness provided by the slot's block producer. - A Feistel transformation ($y \to y^3 \mod 128$) is applied to enhance dispersion and unpredictability. - Block producers submit zero-knowledge proofs to confirm honest shuffling and commit to their permutations in advance, preventing manipulation.

3. **Proposer Selection:** Post-shuffling, RANDAO selects 8,192 validators from the shuffled candidate

list to form the proposer list, dictating the block proposal order for the next 8192 slots.

This structured process guarantees a securely randomized proposer list, obscuring validator identities until the moment of proposal and thwarting potential attackers.

## 3. Homomorphic Sortition Protocol

Homomorphic Sortition uses threshold fully homomorphic encryption (ThFHE) to privately select leaders based on stake. Validators privately check eligibility using encrypted data by evaluating whether a pseudorandom value falls within their encrypted stake range. Once a winner is determined, an encrypted voucher is generated, partially decrypted by participants, and fully reconstructed once enough shares are collected. Communication overhead remains low, scaling with the number of participants, while computational costs are high due to complex homomorphic operations, limiting scalability for large networks.

### 3.1 Homomorphic Sortition Mechanism

Homomorphic Sortition employs threshold fully homomorphic encryption (ThFHE) to enable private leader selection in a decentralized system, such as Ethereum, based on validators' stakes while preserving confidentiality. The mechanism unfolds in three key phases:

1. **Eligibility Determination:** A pseudorandom value $x$ is derived from a publicly available seed and the current round number. Each validator's stake is encrypted and scaled relative to the total system stake (Figure 4). Privately, the validators assess whether $x$ is within their scaled stake range, determining eligibility without disclosing individual stake amounts.

2. **Leader Selection:** Through homomorphic operations, a binary selection vector $E$ is constructed, identifying the chosen validator (Figure 5). The encrypted stake, ticket, and identity of the selected validator are processed to produce a voucher $v_r$, serving as a cryptographic proof of selection (Figure 6).

3. **Proof Sharing and Verification:** The selected leader distributes partial decryptions of $v_r$. Once sufficient valid shares are gathered—surpassing the system's predefined fault tolerance threshold—the voucher is fully decrypted, enabling public verification of the leader's identity without compromising private data.

This approach guarantees fairness and privacy in leader selection, while maintaining robust security. However, the reliance on intricate homomorphic computations introduces significant computational overhead.

## 4. Our Contributions

As part of this work, a proof-of-concept implementation of two Single Secret Leader Election (SSLE) protocols — **WHISK** and **Homomorphic Sortition** — was developed. The WHISK protocol was realized with SNARK-based shuffle proofs using libsnark library [4] to verify correct randomization of tracker commitments, achieving efficient and verifiable validator selection.

The Homomorphic Sortition protocol was implemented using threshold Fully Homomorphic Encryption (ThFHE) using openFHE library [5], enabling private, stake-proportional leader election through encrypted computation and threshold decryption.

The implementation includes core cryptographic primitives, custom SNARK circuits for verifiable shuffling, and TFHE-based circuits for encrypted comparison, selection, and voucher generation.

Validator workflows were emulated in a controlled environment to evaluate the behavior of leader election processes.

Performance characteristics, including SNARK proof generation time, proof size, verification speed, and the overhead of encrypted circuit evaluation, were measured to assess the practicality of deploying these privacy-preserving protocols in blockchain systems.

## 5. Conclusions

This study, backed by analysis and Proof-of-Concept (PoC) implementations, reveals that WHISK and Homomorphic Sortition markedly strengthen Ethereum's defenses against DoS attacks by shielding block proposer identities. WHISK delivers a workable yet resource-heavy option for today's blockchain systems, while Homomorphic Sortition offers superior privacy at a steep computational cost. Yet, with weaknesses persisting in both, the door remains open for smarter, more resilient solutions.

## References

[1] Luciano Freitas, Andrei Tonkikh, Adda-Akram Bendoukha, Sara Tucci-Piergiovanni, Renaud Sirdey, Oana Stan, and Petr Kuznetsov. Homomorphic sortition – single secret leader election for PoS blockchains. Cryptology ePrint Archive, Paper 2023/113, 2023.

[2] Dan Boneh, Saba Eskandarian, Lucjan Hanzlik, and Nicola Greco. Single secret leader election, 2020.

[3] Antoine Toulme, Justin Drake, Dankrad Feist, Gottfried Herold, Dmitry Khovratovich, Mary Maller, and Mark Simkin. Whisk: A practical shuffle-based ssle protocol for ethereum. https://hackmd.io/@asn-d6/HyD3Yjp2Y, 2020.

[4] SCI-PR Lab. libsnark: a c++ library for zksnark proofs. https://github.com/scipr-lab/libsnark.

[5] OpenFHE Development Team. Openfhe [software]: Open-source fully homomorphic encryption library. https://github.com/openfheorg/openfhe-development, 2022.