Zero-Knowledge Proof for Attack Prevention in The Ethereum Blockchain

Anders Malta Jakobsen*, Oliver Holmgaard†

Abstract—This is a placeholder abstract test. The whole template is used in semester projects at Aalborg University (AAU).

1 Introduction

TODO

2 Background

In this section, we will go through some of the concepts that will be used in the rest of the paper as well as some surrounding context like attacks performed.

2.1 Ethereum and Proof of Stake

Ethereum is a blockchain platform that allows developers to create decentralized applications using smart contracts. Previously operating with a Proof of Work (PoW) consensus algorithm, Ethereum transitioned to a Proof of Stake (POS) consensus algorithm in 2022. This transition was done to reduce the energy consumption of the network and to increase the scalability of the network. The transition was done in a series of upgrades called the Ethereum 2.0 upgrade. POS works by selecting validators to create new blocks based on the amount of cryptocurrency they have staked.

2.2 Zero-Knowledge Proofs

A Zero-Knowledge Proof (ZKP) is a cryptographic method that allows one party to prove to another party that something is true without revealing any information.

Two of the subcategories of ZKPs are Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (ZK-SNARK) and Zero-Knowledge Scalable Transparent Argument of Knowledge (ZK-STARK). ZK-SNARK is the more common of the two. It uses elliptic curve cryptography to create proofs based on the assumption that it is hard to find the discrete logarithm form the publicly known base point. ZK-SNARK has a start-up ritual which requires a trusted setup by all parties involved that the original proving key is destroyed as to not be able to create fake proofs. A criticism of ZK-SNARK is that it is not quantum resistant because of the reliance on elliptic curve cryptography.

ZK-STARK on the other hand, is a newer and more complex ZKP. Despite not having non-interactive in its name, ZK-STARK is also non-interactive. Different form ZK-SNARK, ZK-STARK uses hashing functions to create proofs instead of

- *All authors are affiliated with the Dept. of Computer Science, Aalborg University, Aalborg, Denmark*
- *E-mails: *amja23, †oholmg20 @student.aau.dk*

elliptic curves. This method is post-quantum secure and does not require a trusted setup. It does, however, come with a higher computational cost and is not as widely used and documented as ZK-SNARK.

3 Related Work

The usage of ZKPs in Ethereum is not a new concept. In fact, it currently uses them both on- and off-chain. The following provides a short overview of some of the already existing solutions as well as one still being in development.

3.1 Off-chain roll-ups

As a result of Ethereum's popularity, the network could easily get congested if developers were not actively trying to distribute computations [1].

One of the first ideas was to introduce an on-chain technique called sharding. It is a technique where the database would be split into different parts between subsets of validators. Sharding was never deployed on the blockchain though, and instead Ethereum uses off-chain solutions to off-load computations. The idea of off-chain solutions, called L2-roll-ups, is that users commit their work to off-chain nodes. These perform the computations, thereafter they submit the work to the Ethereum Mainnet chain.

One of these solutions is called a Zero-Knowledge Rollup (ZK-rollup).

3.2 Whisk

How do we simulate the attack? What variables do we want to control? Severity of attack - Slow down node or make node disappear?

4 Experimental protocol

5 Discussion

This is a potential discussion section.

6 Conclusion

This is a potential conclusion section.

References

[1] ethereum.org, "Scaling," 2024, Accessed: 15-10-2024.

[2] H. Chen, M. Pendleton, L. Njilla, and S. Xu, "A survey on ethereum systems security: Vulnerabilities, attacks, and defenses," *ACM Comput. Surv.*, vol. 53, no. 3, Jun. 2020, ISSN: 0360-0300. DOI: 10.1145/3391195.

[3] A. H. H. Kabla, M. Anbar, S. Manickam, T. A. Al-Amiedy, P. B. Cruspe, A. K. Al-Ani, and S. Karuppayah, "Applicability of intrusion detection system on ethereum attacks: A comprehensive review," *IEEE Access*, vol. 10, pp. 71 632–71 655, 2022. DOI: 10.1109/ACCESS.2022.3188637.

[4] V. Buterin, "Eip-150: Gas cost changes for io-heavy operations," *Ethereum Improvement Proposals*, no. 150, September 2016, Accessed: 18-10-2024.

[5] V. Buterin, "A state clearing faq," November 2016, Accessed: 18-10-2024.

[6] G. Wood, "Eip-161: State trie clearing (invariant-preserving alternative)," *Ethereum Improvement Proposals*, no. 161, October 2016, Accessed: 18-10-2024.

[7] ethereum.org, "Secret leader election," 2024, Accessed: 22-10-2024.

[8] ethereum.org, "Ethereum proof-of-stake attack and defense," 2024, Accessed: 22-10-2024.

[9] V. Buterin, "Secret non-single leader election," 2024, Accessed: 22-10-2024.

[10] G. Kadianakis, "Whisk: A practical shuffle-based ssle protocol for ethereum," 2024, Accessed: 22-10-2024.

[11] J. Neu, E. N. Tas, and D. Tse, "Two more attacks on proof-of-stake ghost/ethereum," in *Proceedings of the 2022 ACM Workshop on Developments in Consensus*, ser. ConsensusDay '22, Los Angeles, CA, USA: Association for Computing Machinery, 2022, pp. 43–52, ISBN: 9781450398794. DOI: 10.1145/3560829.3563560.

[12] C. Schwarz-Schilling, J. Neu, B. Monnot, A. Asgaonkar, E. N. Tas, and D. Tse, "Three attacks on proof-of-stake ethereum," in *Financial Cryptography and Data Security*, I. Eyal and J. Garay, Eds., Cham: Springer International Publishing, 2022, pp. 560–576, ISBN: 978-3-031-18283-9.

[13] U. Pavloff, Y. Amoussou-Guenou, and S. Tucci-Piergiovanni, "Byzantine attacks exploiting penalties in ethereum pos," in *2024 54th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2024, pp. 53–65. DOI: 10.1109/DSN58291.2024.00020.

[14] ethereum.org, "What are zero-knowledge proofs?," 2024, Accessed: 15-10-2024.

[15] K. Charbonnet, "A technical introduction to maci 1.0," 2022, Accessed: 15-10-2024.

[16] M. Zhang, R. Li, and S. Duan, *Max attestation matters: Making honest parties lose their incentives in ethereum PoS*, Cryptology ePrint Archive, Paper 2023/1622, 2023.

Appendix A

Attacks on Ethereum

In the following section is an explanation of different attacks that either can or could happen to the ethereum blockchain. There is a description of both the attack and its possible mitigation. If found, there will be a link to a Proof-of-Concept (PoC) of the attack and/or the proposed/implemented mitigation.

A.1 Reorg

A.2 DoS

We've found three different kinds of DoS attacks that either were or are possible to perform on Ethereum.

One of the attacks is called *under-priced opcodes* [2, 3]. This attack works because Ethereum has a gas

mechanism to reduce abuse of computing resources. Though when a contract has a lot of underpriced opcodes, they will consume many resources. Execution of contracts requires a lot of resources.

To mitigate this, Ethereum has raised the gas cost of opcodes to preserve the number of transactions-per-second [4] [1].

Another attack, which is closely related to the former, is *empty account in the state trie* [2, 3]. This attack was possible because the existence of empty accounts increases the transaction processing time and synchronization. An empty account is an account with zero balance and no code. The attack required the proposer to select only the transactions of the adversary, which could be insured by offering a higher gas price.

The mitigation is a combination of the one explained for *under-priced opcodes* as well as a mitigation for clearing empty accounts [4–6] [2].

The last example of a DoS attack is called *Proposer DoS* [7, 8]. The background to making this attack possible is that the consensus mechanism uses a publicly known function for choosing the upcoming block proposers. The adversary is therefore able to compute this in slight advance of the blockchain, s.t. each proposer is now known. After this, the adversary can map the proposer's IP addresses and overload their connection. A successful attack would leave a proposer unable to propose their block in time.

To prevent this kind of attack, Ethereum plans to use something they call Single Secret Leader Election (SSLE) which ensures that only the selected validator knows that they have been selected [7, 9].

Specifically, a proposal has been made to use an election protocol called Whisk, which is a type of SSLE [10][3][4][5]. It works by each validator submitting a commitment to a secret shared by all validators. The commitments are shuffled s.t. no-one can map commitments to the validators, but each validator knows what commitment belongs to them. This shuffle-phase goes on for a day, 256 epochs, before using the shuffled proposer list the following day. Commitments are chosen at random, and the selected proposer will detect its commitment to know when to propose a block.

The shuffling phase requires validators to occasionally shuffle a subset of candidate proposers. Using a subset is a measure to reduce computation for the validators, as 256 epochs correspond to 8912 proposers. The shuffle requires a validator to construct a ZKP to confirm that the shuffle was performed correctly.

A.3 Balancing Attack

For this type, we found two attacks.

The first attack we call *LMD-specific balancing attack* [11]. This attack exploits the Latest Message Driven (LMD) *proposer boosting* by sending out two competing blocks but giving half the validators one block before the other and the opposite for the other half. This would create a fork in the blockchain.

Although no mitigation is mentioned, it requires $W_p/b + 1$ adversarial slots [6].

Another balancing attack has the adversary exploiting adversarial network delay and strategic voting by a vanishing fraction of adversarial validators to stall the protocol indefinitely [12] [7].

1. EIPs/EIPS/eip-150.md at master ethereum/EIPs GitHub

2. EIPs/EIPS/eip-161.md at master ethereum/EIPs GitHub

3. Shuffle_SSLE/rust_code/src at master ethresearch/Shuffle_SSLE GitHub

4. [WIP] Introduce consensus code for Whisk (SSLE) by asnd6 Pull Request #2800 ethereum/consensus-specs GitHub

5. GitHub - dapplion/lighthouse at whisk

6. where W_p is the proposer boost weight (fx 100 validators/slot: $W_p = 0.7 \cdot 100 = 70$) and b is the fraction of adversaries in the committee in each slot

7. GitHub - tse-group/gasper-gossip-attack

Though this attack does depend on networking assumptions that are highly contrived in practice; those being the attacker having fine-grained control over latencies of individual validators.

A.4 Finality Attack (Bouncing Attack)

For the Ethereum, there exist attacks called finality attacks, also known as bouncing attacks, which have the purpose of denying the blockchain to finalize its block, halting its functionality.

The first attack is a *double finality* attack [8, 13]. It is theoretically possible for an attacker that wants to risk 34% of the total staked ether. Two forks finalize simultaneously, creating a permanent split of the chain.

A way to see a mitigation of this is that it is practically impossible given the current value of 34% of the total staked ether. Also, voting on two different chains, called double voting, is a slashable offense in the Ethereum chain, so the adversary would get their staked ETH slashed.

Another finality attack is called *33% finality attack* [8]. Here, all adversarial ($\geq 33\%$) validators can simply go inactive, meaning that a block cannot get $2/3$ attestations which is required to achieve finality. Therefore, the blockchain would not be able to go further, as it would not be able to achieve finality.

The mitigation for this is called *the inactivity leak*. The Ethereum chain penalizes the validators who resist voting or are inactive. Their ETH gets burned until the majority vote has a $2/3$ majority. This makes the attack impractical for the attacker, as it is costly to have $\geq 33\%$ of the total staked ETH and their ETH gets burned as well.

A.5 Avalanche Attack

This type only includes a single attack that we call *avalanche attack on proof-of-stake ghost* [11] [8]. The attack uses withheld blocks to make wide subtrees to displace an honest chain. It does this by exploiting the reuse of uncle blocks.

The mitigation for this attack is already a part of the Ethereum blockchain [9]. It is mitigated by LMD, which works together with Greedy Heaviest Observed Subtree (GHOST). The protocol only counts a vote from a validator if the vote is strictly later than the current entry. If two equivocating votes were sent from the same validator at the same time slot, only the earlier message would be counted.

A.6 Bribery

An adversary using bribery attacks on the Ethereum chain could be interested in dictating a choice in some sort of voting mechanism. This is exactly what happens in a described *quadratic funding* attack [14]. The adversary researches votes on the chain and bribes users to vote for what the adversary wants.

In defense of a potential bribery attack, the Ethereum blockchain implements a private voting system called Minimum Anti-Collusion Infrastructure (MACI) [14, 15] [10].

What MACI does is essentially hiding what each person has voted for. It does so by demanding the voters to send their votes encrypted to a central coordinator. This coordinator constructs ZK-SNARK proofs, which verifies that all messages were processed correctly, and that the final result corresponds to the sum of all valid votes.

As votes are now hidden, the adversary is not able, by oneself, to prove that the bribee voted in way of said bribery. Though the bribee could decrypt their own message and show the vote to the adversary.

MACI has fixed this problem by implementing public key switching. This means that a voter can request a new public key. In addition to this, a vote

8. GitHub - tse-group/pos-ghost-attack

9. Proposer LMD Score Boosting by adiasg Pull Request #2730 ethereum/consensus-specs GitHub

10. GitHub - privacy-scaling-explorations/maci: Minimal Anti-Collusion Infrastructure (MACI)

is only valid if it uses the most recent public key of the voter. Therefore, a bribee can show its first vote obeying the adversary, generate a new public key, and send a new, now honest, vote. The old vote will then become invalid as it uses a deprecated public key.

A.7 Staircase Attack

The last type of attack that we cover is called a *staircase attack*. This is a two-part attack, with a warm-up attack and a full attack [16] [11].

First, we will cover the *warm-up attack*. The attack works when the adversary, called a, has to propose the first block of an epoch. From there, the attack works in four parts:

1) The adversary withholds its block at slot t
2) The honest attestors in slot t create attestation with the last block of the previous epoch, called b.
3) Adversary a releases block b_t and honest validators update their checkpoint to block b_t.
4) Attestations with targets different from b_t will be discarded, and honest attestors in slot t will be penalized eventually

This attack is mitigated by what Ethereum calls *honest reorg*, which should prevent intentionally withheld blocks [16].

A block proposed in slot t with fewer than 20% attestations is considered invalid. Though it can be theoretically avoided with network timing, ensuring at least 20% of the attestors get the block.

With the *warm-up attack*, the author goes on to describe a full *staircase attack*. The attack starts with the *warm-up attack*. Then the plan is to manipulate the source of the honest validators. Half the validators should get an outdated *last justified checkpoint* and be penalized. All byzantine validators withhold their blocks and publish them in the middle of the epoch. It should be able to be done as a one-time attack with probability 98.84% if adversary controls N/3 validators [16]. The byzantine validators don't get penalized, but will get a smaller reward than the fair share.

What would make this attack infeasible is that it would require a lot of controlled validators. The author states that for Ethereum to be vulnerable, there would need to be < 16384 validators, which makes the attack as good as impossible given Ethereum has $1.073.406$ daily active validators [16] [12]. Also, Ethereum released a patch fixing this attack after the author pointed it out [13].

11. GitHub - tsinghua-cel/Staircase-Attack: This is the staircase attack implement for the paper "Max Attestation Matters: Making Honest Parties Lose Their Incentives in Ethereum PoS."

12. According to beaconcha.in as of 22-10-2024

13. Confirmation Rule by saltiniroberto Pull Request #3339 ethereum/consensus-specs GitHub