TL;DR:

Ethereum's scalability and security hinge on addressing the burgeoning state size issue and mitigating the risks posed by consensus bugs. Choosing less prevalent clients becomes crucial for stakers to minimize correlated failures. Transitioning to stateless clients with compact proof sizes offers a promising solution, decoupling historical state storage from transaction processing and enhancing scalability while preserving network integrity.

Introduction:

To address Ethereum's scalability and security imperatives, a paradigm shift towards stateless clients with compact proof sizes is imperative. Stateless clients offer a pragmatic approach to alleviating the burden of state size on network participants by decoupling historical state storage from transaction processing.

Furthermore, picking the right Ethereum client is really important. If many validators using the same client mess up, the consequences get worse. So, people who stake Ethereum and use less popular clients reduce their risk of losing everything. But if they stick with the popular clients, they could lose everything if things go wrong. So, it's important for responsible stakers to choose less common clients to protect against big problems in the system.

The different Ethereum clients add another layer of difficulty, especially because of the constant worry about "consensus bugs." These bugs, like the one that caused the "infinite money supply," can mess up the network and make Ether less valuable. Although people are trying to make the clients stronger, there's still a risk of everything going wrong because there's no way to directly fix the problems caused by too much data.

So, picking the right Ethereum client is a big deal for staking. If lots of validators using the same client make mistakes, the punishments get worse. That's why people need to choose less popular clients to reduce the risks and keep the network running smoothly.

Dankrad Feist has an inisghtful article explains the concerns in details.

Evolution Towards Stateless Clients with Compact Proof Sizes:

Addressing Ethereum's scalability and resilience imperatives demands a transformative shift towards stateless clients augmented by compact proof sizes. Stateless clients, by decoupling historical state data storage from transaction processing, offer a promising avenue to alleviate the burden of burgeoning state size on network participants.

At the core of this transition lies the imperative to devise mechanisms for succinctly representing transaction validity, while minimizing computational overhead using Zero-Knowledge proofs. we recognize that the slashing protection history is a critical database that records a validator's local signing history. This database plays a vital role in preventing validators from signing slashable messages by keeping track of previously signed messages. The failure to migrate this database during system upgrades or client switches puts validators at risk of duplicative actions, potentially leading to severe penalties under the slashing protocol.

Our EIP-X proposal centers around Creating a peer-to-peer (P2P) network of light clients leveraging Zero-Knowledge Proofs (ZKPs) for the validation of Ethereum block headers, specifically the last finalized block at the beacon chain and presents a novel approach to enhancing the scalability, security, and efficiency of the Ethereum network without changing the core structure. It can be considered as a PIC of Verkle tree plans. This proposal outlines the technical foundation and goals for implementing such a system, with an emphasis on the Ethereum Portal Network and its potential to support this innovative solution.

Integrating Zero-Knowledge Proofs (ZKPs) can significantly enhance this framework by offering a more scalable and secure approach to managing and verifying the slashing protection history without compromising on privacy or the integrity of validators' signing history. ZKPs can allow validators to prove that they have not signed any slashable messages in the past without revealing the specific details of the messages they have signed. This capability is particularly relevant in the context of stateless clients, where the emphasis is on minimizing the storage requirements and computational overhead on validators.

Integrating zk-SNARK proofs of witness could potentially save both gas and disk space:

There's a significant reduction in gas costs, which can lead to lower transaction fees and improved efficiency on the network.

The disk space savings, while smaller, contribute to reducing the storage requirements, which can be particularly beneficial for nodes with limited resources.

This analysis doesn't account for the computational cost and time of generating zk-SNARK proofs, which is significant (around 3 seconds and 106 MB of memory for generation). These factors would need to be considered when evaluating the overall benefits and trade-offs of implementing such a system.

EIP-2929 introduces a simpler gas cost schedule that directly charges for accessing subtree and an element within the subtree. however Its interesting to explore the pptential optimization of gas cost used and disc resources for ZKP of blockwitness in the meanwhile where Verkle tries are under development.

Comparing with Verkle tries, which aim to optimize storage and access through a different approach, zk-SNARKs offer a complementary strategy by providing privacy and compression benefits. The choice between using Verkle tries and zk-SNARKs (or a combination of both) would depend on the specific requirements of the application, including the need for privacy, the amount of data to be stored, and the computational resources available for proof generation.

Vitalik has a brilliant and insightful proposal on redesign how witness gas costs would work in a more principled way that covers accounts, storage slots and contract code and the witness lengths of each one after a switch to Verkle trees.

I've used that insight to estiamte **potential improvement in gas cost and resrouce usage of ZKP**(snark) of witness for EIP-x proposal:

Step 1:

Calculate Traditional Witness Gas Costs

Average Gas Per Byte (Traditional): Given as 9.5 gas.

Average Witness Size: Given as 200 bytes.

Calculate Gas Cost for Traditional Witness:

Formula: Average Gas Per Byte (Traditional) * Average Witness Size

Calculation: 9.5 gas/byte * 200 bytes = 1,900 gas

Step 2:

Estimate zk-SNARKs Gas Costs

For zk-SNARKs, the direct gas cost per byte wasn't provided, so we used the same average gas per byte as the traditional method for comparison purposes.

Assumed Average Gas Per Byte (zk-SNARKs): 9.5 gas (same as traditional for comparison).

zk-SNARK Proof Size: Given as 128 bytes.

Calculate Gas Cost for zk-SNARKs:

Formula: Average Gas Per Byte (zk-SNARKs) * zk-SNARK Proof Size

Calculation: 9.5 gas/byte * 128 bytes = 1,216 gas

Step 3:

Calculate Gas Savings with zk-SNARKs

Difference in Gas Costs: Subtract the zk-SNARKs gas cost from the traditional witness gas cost.

Calculation: 1,900 gas (Traditional) - 1,216 gas (zk-SNARKs) = 684 gas saved

Step 4: Calculate Disk Space Savings

Difference in Disk Space Requirements: Subtract the zk-SNARK proof size from the average traditional witness size.

Calculation: 200 bytes (Traditional) - 128 bytes (zk-SNARKs) = 72 bytes saved

[

Bildschirmfoto 2024-03-24 um 13.43.51

1462×880 126 KB

](https://ethresear.ch/uploads/default/original/2X/4/4f908d60e6a78d6ba84791bd463325d3c679036c.jpeg)

Here's a summary of the comparison between traditional witness data and using zk-SNARKs, displayed in both a table and a chart:

[

Bildschirmfoto 2024-03-24 um 13.44.03

1424×228 13.1 KB

](https://ethresear.ch/uploads/default/original/2X/b/b0033976833d63ca0a55b8da6eb6fdf840cc1ef1.png)

**Futhre Comprehension between ZKP and Verkle trie: **

Verkle trees provide a more efficient method for generating and verifying proofs in scenarios involving a large number of operations, with their verification process being fast and cost-effective for adding additional proofs. zk-SNARKs, while offering strong privacy and succinct proofs, require more time for proof generation and incur a significant cost for verification. The choice between the two would depend on the specific application requirements, including the importance of proof generation time, verification costs, and the scalability needs for handling multiple proofs or commitments.To compare the computational and verification costs between Verkle trees and zk-SNARKs, we'll outline the key metrics for each method based on the provided data. The comparison will focus on the proof generation time, verification time, and the verification cost where applicable.

Verkle Trees

Prover Cost:

- Operations per Opening:

256 * 4 field operations.

- Total for 25,000 Openings:

Based on 5,000 proofs leading to 25,000 openings (an extrapolation from the data provided), the total operations would be significantly higher. However, the exact operations for 25,000 openings aren't directly provided but can be inferred to require substantial computational effort.

- Proof Generation Time:

Approximately 750 milliseconds for 5,000 proofs.

Verifier Cost:

- Multi-Scalar Multiplications (MSM) Size:

For 5,000 proofs leading to 15,000 commitments, the MSM size is 15,000.

- Verification Time:

Approximately 50-150 milliseconds.

- Marginal Cost for Another Opening:

Very low, in the order of hundreds of nanoseconds.

zk-SNARKs

Prover Cost:

- Proof Generation Time:

On average, 10-15 minutes (a significant difference compared to Verkle trees).

- Note:

The computational cost in terms of field operations isn't directly provided for zk-SNARKs but is known to be substantial due to the complex cryptographic computations involved.

Verifier Cost:

- Verification Cost:

500,000 GWEI.

Comparison Summary

- Proof Generation Time:

Verkle trees are significantly faster in proof generation (milliseconds vs. minutes for zk-SNARKs).

- Verifier Efficiency:

Verkle trees offer a fast verification process (50-150 ms) without specifying a cost in GWEI, whereas zk-SNARK verification costs 500,000 GWEI. The time for zk-SNARK verification isn't provided but is generally considered to be fast, often in the milliseconds range.

- Scalability for Multiple Proofs:

Verkle trees have a very low marginal cost for additional openings, enhancing scalability within the same commitment. zk-SNARKs have a fixed proof size and verification cost, independent of the number of operations proved.

Objectives

The Ethereum network is evolving to enhance scalability and minimize the burden on nodes by reducing the volume of data they need to handle. Traditional approaches where light clients individually request data from full nodes can significantly strain the network, especially when multiple clients make concurrent requests. To address this, a new method proposed in EIP-X focuses on leveraging the beacon chain's events. When a new block is finalized, a witness of the previous block is generated, and a zero-knowledge proof (ZKP) of that witness is created. This ZKP is then sent in a single message over the Ethereum communication protocol, Discv5, effectively disseminating all the necessary data to all participating peer-to-peer (P2P) nodes in the Trin portal network with a single call. Stateless clients, which rely on proofs rather than storing the entire state for transaction execution, are fundamental to this advancement. This solution aims to utilize ZKPs to authenticate block headers and the transitions of state they imply, within a P2P network of light clients operating on the Portal Network, thereby streamlining data verification and transmission across the network.

Technical Proposal Overview

- Portal Network as the Foundation: The Portal Network is designed to improve the scalability and decentralization of the Ethereum network by allowing nodes to store and serve only a subset of the entire blockchain state. It's a network of light clients that support Ethereum's vision of a more accessible and efficient blockchain. By leveraging the Portal Network, we can facilitate a distributed environment where light clients efficiently verify and relay block information without needing the complete blockchain state【Portal Network Introduction】.

- Integration with Zero-Knowledge Proofs: At the heart of this proposal is the use of ZKPs to ensure the integrity and correctness of the block headers shared among the light clients. ZKPs can compactly prove that a block header is correct and that the transactions it contains have been accurately processed, without revealing the entire transaction data or state changes. This mechanism significantly enhances privacy and efficiency, as it allows for the verification of transactions with minimal data requirements. (Using geth) challenge: new fork

Communication protocol:

To effectively utilize the Discv5 messaging protocol for propagating Zero-Knowledge Proofs (ZKPs) across all light client recipients on the Portal Network, it is necessary to introduce a custom message type specifically designed for ZKP dissemination. The Discv5 protocol accommodates various message types, facilitating a range of network activities including peer discovery, information sharing, and data synchronization. Each message type is associated with a request and a corresponding response, ensuring a coherent communication flow.

Proposed Enhancement:

Custom ZKP Message Type Implementation:

Definition of a Custom Message Type for ZKPs: To streamline the propagation of ZKPs, a new custom message type within the Discv5 protocol must be defined. This message type will be exclusively used for transmitting encrypted ZKP data to light clients on the Portal Network.

Unique Message ID for ZKP Messages: Upon receiving a ZKP message, nodes will identify the message type through a distinct message ID designated for ZKP content. This identification process is crucial for nodes to recognize the received data as a ZKP and to respond appropriately.

Network Adaptation to Support New Message Type: For the seamless integration and support of the new ZKP message type, the entire P2P network must update to recognize and process the new message ID. This adaptation ensures that ZKP messages are correctly handled and propagated across the network.

Role of Custom Messages in Network Connectivity:

These custom messages are pivotal in enhancing the network's connectivity and efficiency. By allowing nodes to accurately locate peers, exchange ZKPs, and maintain up-to-date state information, the network can achieve higher levels of security and scalability. The integration of ZKPs, coupled with the Discv5 messaging protocol, enables a robust framework for stateless client operation within the Ethereum ecosystem.

- P2P Network of Light Clients Using ZKPs: Light clients on this network will operate by receiving ZKPs of the last finalized block's header. This header includes a root hash that represents the state changes made by transactions within the block. The clients will use these proofs to verify the integrity and correctness of the state transitions, ensuring they are consistent with the Ethereum protocol rules.

- Portal Subnetwork for State Verification: To facilitate consensus and state verification in a stateless manner, the plan includes the creation of a Portal subnetwork dedicated to comparing ZKPs of the last known state with the current state. This comparison verifies that state transitions are legitimate and that the block being proposed for inclusion in the blockchain is valid. This subnetwork will play a crucial role in maintaining the security and integrity of the blockchain while minimizing the computational and storage overhead for participants.

For more details on the Portal Network and its potential, refer to the Ethereum Portal Network documentation and resources:

- Ethereum Portal Network Introduction
- Ethereum Developers Portal Network
- EIP-x github
- Ethereum Merge: Run the majority client at your own peril!-Dankrad Feist