Motivation:

Ethereum's state grows rapidly, burdening consensus nodes with 35 GB state and 100 GB with proofs, increasing every six months. The increasing state size hampers scalability, imposing storage and processing burdens on nodes. Participants pay one-time costs perpetually, raising economic concerns. Additionally, ERC-4337 wallets incur elevated gas costs (~42000 gas for basic operations) due to multiple storage read/write costs, business logic overhead, and log payments not required in EOAs. A stateless-based solution is essential to maintain Ethereum's efficiency and long-term viability. Additionally, the light clients, which do not store the entire state but rely on simplified verification mechanisms, struggle to efficiently access and validate the state data against the mainnet. The lack of a concise and constant-sized proof of the current state limits light clients' ability to interact with the blockchain seamlessly. Therefore we Propose a stateless Account abstraction model which doesn't force any change in the current Ethereum core but can solve the above problems at the account level. The advantage of having succinct zk proof of the state is that it is possible to broadcast it across the nodes.

Project description

The challenge at hand is to enable stateless verification of blocks in the blockchain system. Currently, to validate a block, light clients must request the state piece by piece, and this results in many light clients frequently burdening the full node altogether. The goal is to allow clients to verify the correctness of individual blocks without needing a full state. Instead, clients can rely on compact witness files generated by state-holding nodes, which contain the portion of the state accessed by the block along with proof of correctness. This stateless verification offers several benefits, including supporting sharding setups and reducing reliance on validators.

Issue 1: introducing new Entity for Stateless AA, trustless State provider #1

this entity consists of Ethereum light clients providing the latest state, peer-to-peer network and ZK circuit. We introduce a decentralized peer-to-peer network with a state provider entity, collaborating to validate and generate ZK-SNARKs proofs for state information.A lightweight client verifies proofs and validates state data from the state provider.

The state provider entity is at the core of stateless AA, and leverages theHelios light client to query the latest block state in a fully trustless manner. By implying

a peer-to-peer network, we can ensure the latest state of the block in a fully trustless and decentralized manner. Then we are able to provide succinct and constant-sized proof of the current state and propagate the ZK-proof of witness needed for execution. The current zk-proof of the state can then be propagated to all light clients at once. Light clients can effortlessly verify this proof against the mainnet, improving scalability, and reducing resource requirements for light clients. Users and applications can efficiently interact with the blockchain, relying on the security and integrity of zk-proofs to verify the state. This approach enables more participants to engage with the network without needing the computational resources and storage capacity required for full nodes.

Issue 2: introducing new Entity for Stateless AA,The stateless verifier

:

The stateless verifier acting as a light client that receives a zero-knowledge proof (zk proof) of a block state and the new state after a transaction is executed. This verifier will provide verification of the transaction and include the new state on the chain. The verifier receives a new block containing the transaction, the zk proof of the witness including the latest state (before execution), and the resulting new state after the transaction. The verifier extracts the transaction and the zk proof from the received block. It can then verify the correctness of the transaction execution without needing to access the entire state.

- If the transaction is valid, the verifier applies the transaction to the state it currently holds, thus deriving the new state.

P.S: Im working on this concept druing theEthereum fellowship cohort 4 and represent my progress in my Github.