Why do we need decentralized provers?

Currently, there are multiple ZK-Rollups running on the Ethereum mainnet. However, the design of decentralized ZK-Rollups is still in its early stages. We are currently focused on the issue of decentralized sequencer, but most people overlook the fact that currently, most ZK-Rollup projects have not implemented decentralized provers.

For ZK-Rollups, a centralized prover is still safe and does not bring the same issues of cersorship as a centralized sequencer. However, a centralized prover can also cause many problems. First, if there is only one prover, a single node failure can cause the entire ZK-Rollup to fail to submit its validity proof, thus affecting the finality of transactions. Second, the cost of a centralized prover is high, and it is unable to meet the computational demand for massive ZK-Rollups in the future. Finally, from an economic perspective, a centralized prover alone enjoys a portion of the profits, which from a token economics perspective, is actually unfair.

Challenges of decentralized provers

Decentralized provers can effectively solve the above problems, but it also brings some challenges. This is one of the reasons why several zkEVM schemes recently launched have adopted a centralized prover scheme. For example, the beta mainnet of the Polygon zkEVM relies on a trusted aggregator to submit ZKPs, and zkSync era is similar in this regard.

From a technical perspective, when the smart contract of a ZK-Rollup verifies the ZKP, it needs the original proof data. This can potentially trigger various on-chain attack behaviors. For example, when a certain prover submits the calculated ZKP to the chain-level contract, it needs to send an L1 transaction. When this transaction broadcasts to the transaction pool, attackers can see the original proof data, and they can set a higher gas fee to send a transaction, thus being first to be bundled into a block and earn PoW rewards. In addition, since provers compete with each other based on computational power, there is no reliable identity recognition mechanism, and it is also difficult to establish a communication mechanism. Different miners may perform duplicate work, resulting in wasted computational power.

Two-Step Submission of ZKP

Opside has proposed a two-step submission algorithm for ZKPs to achieve decentralized proof-of-work. This algorithm can prevent ZKP race attacks while allowing more miners to earn rewards, thereby encouraging more miners to stay online and providing stable and continuous ZKP computational power.

[

Two-Step Submission of ZKP

2406×966 124 KB

](https://ethresear.ch/uploads/default/original/2X/f/f9f606ef39400ba262178458b4af8972375a9ae2.png)

Step 1: Submit hash

- After a prover calculates a ZKP for a certain sequence, it first calculates the hash of (proof / address) and submits the hash and address to the chain-level smart contract. Here, proof is a zero-knowledge proof for a certain sequence, and address is the address of the prover.
- Assuming that the first prover submits the hash of the ZKP at the Tth block, it is accepted until the T+10th block without any limit. From the T+11th block, new provers cannot submit the hash anymore.

Step 2: Submit ZKP

- After the T+11th block, any prover can submit a ZKP. As long as one ZKP passes verification, it can be used to verify all the submitted hashes. Validated provers receive PoW rewards based on the ratio of miners' staked amounts.
- If no ZKP passes verification before the T+20th block, all provers who have submitted hashes will be slashed. The sequence is then reopened, and new hashes can be submitted, returning to Step 1.

Here's an example: let's assume that each block has a PoW reward of 128 IDE on the Opside network, and there are currently 64 rollup slots available. Therefore, each rollup sequence is assigned a PoW reward of 2 IDE. If three miners, A, B, and C, successfully submit the correct ZKP for a sequence in succession, and the three miners' miner stakes (IDE) are 200K, 500K, and 300K, respectively. Then, A, B, and C can each earn a PoW reward of 0.4 IDE, 1 IDE, and 0.6 IDE, respectively.

Prover's token stake and punishment

To prevent malicious behavior from the prover, the prover needs to register with a special system contract and stake a certain amount of token. If the current stake amount is less than the threshold, the prover can not submit the hash and ZKP. The reward for the prover's submission of the ZKP will be distributed based on the ratio of the stake amount, preventing the prover from submitting multiple ZKPs.

If the prover commits the following actions, different levels of punishment will be applied:

- The prover submits an incorrect hash.
- For a certain sequence, if no corresponding ZKP passes verification, all provers who have submitted hashes will be punished.

The forfeited token will be burned.

For more details and considerations about the two-step submission mechanism of the ZKP, readers are encouraged to refer to the Opside official docs. The specific numbers of the prover's stake and punishment may be changed in the future.

Several considerations:

- Why allow multiple provers to submit hashes? If only the first prover to submit a hash is rewarded, other provers may not have an incentive to submit a proof after the first prover submits a hash. If a malicious attacker delays submitting the proof for a long time after submitting a hash, it may slow down the verification of the entire sequence. Therefore, it is necessary to allow multiple provers to independently and simultaneously submit hashes to avoid monopoly of ZKP verification by a single attacker.
- Why is there a time window? If anyone can submit a proof immediately after submitting a hash, the proof may still be stoled. Attackers can immediately submit a hash associated with their address and then submit a proof to earn rewards. By setting a time window, provers who have submitted hashes have no incentive to submit proofs within the window, thereby avoiding the possibility of being raced.
- Why is the reward allocated based on stake? Multiple provers can submit hashes for the same sequence within a time window. In fact, miners can submit multiple hashes using their generated proof (only needs multiple addresses). This can lead to the majority or even all of the PoW rewards being taken by miners. To avoid this attack, the reward for a sequence will be allocated based on the ratio of miner's stake amount.

Summary and Planning

The two-step submission algorithm for ZKPs proposed in this post realizes the decentralization of the prover while effectively avoiding race attacks and encouraging more miners to provide stable and continuous ZKP computational power. The initial version of the algorithm will be launched on the Opside pre-alpha testnet. In the future, Opside will also introduce more innovative ideas in the field of ZKP mining, such as:

- Dynamic adjustment of the reward allocation ratio between PoS and PoW based on the demand and supply of ZKP computational power throughout the entire network.
- Personalized pricing mechanism for Rollup batches based on the type of ZK-Rollup, number of Rollup transactions, and gas usage of the Rollup.
- Subsidies for application developers to generate ZKPs for their associated Rollups to encourage miners to provide computational power.