Authors: Norbert, Nilu, Rachit

Special thanks to Barnabé Monnot (RIG/EF) for his review and suggestions.

Summary

We propose a simple mechanism that enables decentralization, permissionless entry, liveness and cost-efficiency. It's an in-protocol mechanism that integrates staking for eligibility and slashing as a security mechanism to disincentivize malicious behavior. It also employs reputation score to measure prover uptime and failures. The provers are selected through a VRF from a pool with the highest reputation score. The design has a backup mechanism for emergencies in times of prover failure and network congestion. The backup mechanism is proof racing in a more confined environment, which promotes competition and liveness. Other features like proof batching and distributed proving can be added on top of this simple design.

Permissionless Entry and Eligibility

Anyone meeting the eligibility criteria can join as a prover and leave at any time. In the exit case, the stake can be withdrawn in a week.

Provers are accepted to the network based on two criteria:

- A minimum computing power — to ensure all provers have sufficient compute capacity to generate a valid proof within the given proof time window. As estimated by Aztec:
- public VM circuit will require no more than 64gb RAM and 32 cores
- public kernel and rollup circuits will require 16gb RAM and 8 cores
- public VM circuit will require no more than 64gb RAM and 32 cores
- public kernel and rollup circuits will require 16gb RAM and 8 cores
- A certain stake amount of ETH or native token — to ensure slashing to disincentivize dishonest behavior.

Decentralization

Since Aztec will be launched as a fully decentralized network, the mechanism should avoid unintentionally leading to centralization or monopolization. Some selection mechanisms might inadvertently favor well-funded or resource-rich participants, leading to centralization.

To detect centralization in a network, the Herfindahl-Hirschman Index (HHI) is used to assess centralization in the distribution of resources among entities in a network. It quantifies the extent to which a few entities dominate the network. Higher HHI values indicate greater centralization, while lower values suggest more distributed networks.

Resource Distribution Inequality (RDI)

HHI method is used in the distribution of resource concentration among provers in the network. It encompasses CPU and RAM of provers' hardware.

RDI = HHI of CPU and RAM Concentration

Geographic Diversity (GD)

Geographic distribution shows the spread of prover network across different physical locations. This metric can be useful for detecting potential Sybil attacks.

[

image.png

1031×125 5.13 KB

](https://europe1.discourse-cdn.com/business20/uploads/aztec/original/1X/6c1eb964dbe76764451881147d2d9fc136cf16b6.png)

These decentralization metrics can be used for risk assessment and monitoring of the network state, and new strategies can be created accordingly. For example, when the network has a higher concentration of provers with expensive hardware, the prover selection mechanism may prioritize provers with cheaper resources to promote equity. This mechanism can be deactived when resource concentration of the network is in equilibrium. Additionally, monitoring and adjusting the allocation of provers to maintain a high GD can be part of your network management strategy.

Prover Selection

The framework for prover selection is based on random selection among the provers with the highest 25% reputation score.

Randomness

For simplicity, we apply the same mechanism as for sequencer randomness inFernet (borrowed from the Espresso team), generating a SNARK of a hash over the prover private key and a public input. The public input is the current block number and a random value from RANDAO.

Reputation Score

The reputation score is dependent on two factors:

- Prover Uptime (PU)

represents the percent of time a prover is available and actively participating in the network.

- Prover Reliability (PR)

represents the reliability of each prover in terms of their ability to generate proofs accurately and without errors.

[

image.png

1000×164 5.35 KB

](https://europe1.discourse-cdn.com/business20/uploads/aztec/original/1X/491b7c18e87092b371726cb46f94e2abe1393c44.png)

Every prover is assigned a base score = 100. The base score is the highest score a prover can have.

Proving Stage

Once the sequencer has revealed the block contents, a prover is selected to prove the block in a specific proof time window. Once the proof is generated, the prover submits the proof to L1 for the block to be finalised. The proof needs to reference the new state root uploaded by the sequencer (similar to how it works in Fernet). Once the proof submitted by the prover to L1 it is verified against the initial block commitment and becomes final. If the prover fails to submit a valid proof on time, the system goes into “emergency mode”.

Emergency Mode

In emergency mode, proof racing mechanism is activated. The proof task for the failed block is opened for competition among N number of provers, randomly selected from the top 25% of provers in the network. This competition can help the system minimize reorgs. The prover who submits the proof the fastest receives the block reward, while other provers receive uncle rewards for their efforts in generating proofs. Rewards for this mechanism can be derived from the slashed stake of the failed prover.

While proof racing leads to some computational waste, it can be considered as network redundancy to maximize liveness and minimize the time/cost lost due to prover failure.

If there are not enough provers in the network and there are too many blocks waiting to be proved, the network again goes into “emergency mode” and runs a proof race among all provers. This time no uncle rewards are distributed. Once the network is stabilized, the network returns to “normal mode”.

Incentives

The primary objective is to incentivize provers to behave honestly and to reward provers in proportion to their computational costs. Reward for honest provers is calculated as follows:

Reward = Prover computation cost based on complexity of the proof + L1 call data cost + Prover profit

Disincentives

Provers are slashed for two reasons:

- Failing to submit a valid proof within the given proof time window.
- The protocol slashes N% of the total staked amount. When a prover’s stake drops to half, the prover is kicked out of the network. In our view the mechanism should not tolerate more than 4-6 proving failures per prover until forced exit

kicks in.

- Part of the slashed amount will be used to cover the rewards in the proof racing mechanism. The rest is burnt.
- The protocol slashes N% of the total staked amount. When a prover's stake drops to half, the prover is kicked out of the network. In our view the mechanism should not tolerate more than 4-6 proving failures per prover until forced exit kicks in.
- Part of the slashed amount will be used to cover the rewards in the proof racing mechanism. The rest is burnt.
- Being inactive for extended time.
- The amount slashed will vary according to the ratio of provers that are active in the network.
- Going offline when the vast majority (2/3) of the provers are still online leads to relatively minor penalties. On the other hand, going offline at the same time as more than 1/3 of the total number of provers leads to more severe penalties.
- Penalties are proportional to the income a prover earns during the time offline. If you go offline under normal circumstances, you will lose an amount of ETH roughly equivalent to the amount of ETH you would have earned during that time.
- Going offline when the vast majority (2/3) of the provers are still online leads to relatively minor penalties. On the other hand, going offline at the same time as more than 1/3 of the total number of provers leads to more severe penalties.
- Penalties are proportional to the income a prover earns during the time offline. If you go offline under normal circumstances, you will lose an amount of ETH roughly equivalent to the amount of ETH you would have earned during that time.
- The amount slashed will vary according to the ratio of provers that are active in the network.
- Going offline when the vast majority (2/3) of the provers are still online leads to relatively minor penalties. On the other hand, going offline at the same time as more than 1/3 of the total number of provers leads to more severe penalties.
- Penalties are proportional to the income a prover earns during the time offline. If you go offline under normal circumstances, you will lose an amount of ETH roughly equivalent to the amount of ETH you would have earned during that time.
- Going offline when the vast majority (2/3) of the provers are still online leads to relatively minor penalties. On the other hand, going offline at the same time as more than 1/3 of the total number of provers leads to more severe penalties.
- Penalties are proportional to the income a prover earns during the time offline. If you go offline under normal circumstances, you will lose an amount of ETH roughly equivalent to the amount of ETH you would have earned during that time.

Transaction Phases

We propose the following transaction states representing the different phases of a transaction lifecycle:

1. Executed: the transaction is approved locally and sent to the mempool,
2. Processed: the transaction is included in a revealed block,
3. Proved: proof has been generated for the block including the transaction,
4. Verified: proof has been verified on L1,
5. Finalized: the transaction has been finalized.

Other Features

Above we defined our simple decentralized prover model. Now we can add more features on top of the model which could be useful in various settings.

Proof Batching

In the case of proof batching, a certain number of proofs generated for individual blocks can be aggregated into a group. Only the aggregated proof is then sent to L1 for verification. This can reduce transaction fees but results in a longer time to finality. Batch proving can be implemented as a mechanism to manage peak periods of incoming transactions. During times of high transaction influx, the proving of individual blocks can run "almost" in parallel and the grouping will cause only minimal delay in finalization at L1.

Distributed Proving

In the context of decentralized proof construction, rather than creating a single proof for a large, monolithic block, we divide it into smaller computational components. Different provers are responsible for generating proofs for these individual pieces. This division of computation also correlates with reduced hardware requirements, especially when compared to handling an entire block. The same principles of prover selection and emergency mode remains consistent in this model.

Allowing Liquid Proving Pools

Similar to the concept of liquid staking, where participants pool their resources, hardware resources can be shared and pooled. This approach lowers entry barriers and increases accessibility for joining the prover network through resource pooling services. The pool leverages these resources to operate a prover and produce proofs. Rewards earned by the prover are distributed among those who contribute computational resources to the pool.

Comparisons

- Sidecar is based on an out-of-protocol proof market where provers submit quotes for specific proving tasks. We are proposing an enshrined prover network with the ability for prover pools to join the network. The proposed in-protocol prover mechanism improves native token utility: the protocol can use and leverage its own native token for staking and prover incentives. It also removes the dependency on an external point of failure, and gives the protocol and its ecosystem greater sovereignty.
- Similar to the Cooperative model our model is also a staking-based mechanism utilizing VRF for prover selection, however in our case the proof for each block is generated by one prover.
- Unlike the Staking Proving Network model and Fernet on the Rocks we propose to use a network of provers separate from the network of sequencers due to eliminating the risk depending on a single actor.

Questions

- What should be the proof time window?
- What is the expected recovery period?
- What is the minimum downtime before a prover gets slashed?
- How can we measure the prover computation cost based on complexity of the proof?
- How do we decide when to enable emergency mode in network congestion?
- How do we decide when to enable proof batching?

Resources

- Decentralized Proving, Proof Markets, and ZK Infrastructure
- Ideas on a proving network
- Understanding rollup economics from first principles
- Taiko Proving Design overview: Grímsvötn and Eldfell cases
- Starknet Decentralized Protocol IV - Proofs in the Protocol
- Provers: To decentralize or not to decentralize?
- Validated, staking on eth2: #1 - Incentives
- awesome-prover-mechanisms