Staking Proving Network for Fernet

Overview

This document proposes a cooperative proving network, with two explicit goals:

1. Ensuring that the role of sequencer and prover are 100% divorced to minimise the centralising forces that can arise from this.

2. Better incentivising stakers, and ensuring sequencer machines are not idle when not proposing

The proposal achieves these goals by re-using the existing staking set proposed in Fernet, and assigning stakers proving work in a deterministic and cooperative manor.

Rational

Fernet relies on a widely decentralised staking set, each with powerful machines, (likely 16 cores ~ 16gb ram) needed to fulfill the role of a sequencer on Aztec.

The design works well very well for randomly selecting a sequencer and ensuring the chain's censorship resistance properties.

However, sequencing is just the tip of the block production iceberg, and when designing for block production as a whole, the staking set needs to perform more functions than just sequencing.

Block Production Life cycle

[

ddd

2760×416 99.1 KB

](https://europe1.discourse-cdn.com/business20/uploads/aztec/original/1X/777f943bae099169bbf0f35e9c35a142c919ae3c.png)

In order to get a block on chain, the current sequencer in Fernet is responsible for all boxes in the diagram. They will be forced to outsource the majority of the work to better suited entities as described below.

Searching & Ordering

Stakers will likely sell the right to order the transactions in a block to a better suited entity via a secondary market e.g Flashbots.

Proving

Without a prover coordination protocol, stakers will either have to control a large proving network, or outsource proving to entities with dedicated hardware, to ensure they can produce a block. This creates large centralising forces on the sequencer staking set, as the sequencer can't fulfil their role without help, as well as weakening the censorship properties of the network.

Proving Committee

Outsourcing proving and searching leaves sequencer machines idle for certain periods. These machines still have large resource requirements, which is wasteful. (minimum 16gb ram 16 cores).

This proposal adds a key role to the staking set, a proving committee. The committee is selected using the same VRF in Fernet to pick a subset of stakers to prove the block. The committee changes each epoch.

The committee is selected from the opposite end of the VRF to sequencers, ensuring no overlap, e.g sequencers have a high VRF, provers have a low VRF.

As members of the proving committees are already sequencers they currently maintain full nodes on the L2, so have access to the full state of the L2, simplifying proving coordination as less data needs to be transferred.

Worked Example

1. The protocol defines a staking system used for selecting sequencers and provers. Stakers register a staking public key on an L1 smart contract and are required to run the staking and proving client.
2. For sequencing, a VRF over the current randao value is used to determine which sequencer can produce a canonical block for a given slot (as per Fernet). The highest VRF will win.
3. The VRF is over the stakers public key

, ensuring other stakers can calculate the winning VRF output. This is important to prevent with-holding attacks.

1. The VRF is over the stakers public key

, ensuring other stakers can calculate the winning VRF output. This is important to prevent with-holding attacks.

1. The block reward is split into three portions B^{Sequencer}

, B^{Provers}

and B^{Submitter}

1. For proving, the same VRF is used to select a proving committee from the lowest n

VRF outputs. (n

is set by the sequencer dependant on desired throughput).

1. The current sequencers block commitments, defines the set of proofs needed for the rollup block. For any given block, stakers, compare their VRF against to see if they are in the proving committee. If so, they submit proofs defined by their VRF and the block transcript using option 1 or option 2 described below.
2. The current sequencer must broadcast all transactions to L2 to allow the proving committee to construct proofs. This is enforced in two ways:
3. The block can only be produced if a subset of eligible provers e.g N/M helps make the proof. (This gives a guarantee that at least N/M stakers have the data for txs in private meme-pool, and preventing withholding attacks unless N/M stakers are the same entity).
4. The B^{Sequencer}

is increased for each distinct prover who helps contribute to a block from within the proving committee.

1. The block can only be produced if a subset of eligible provers e.g N/M helps make the proof. (This gives a guarantee that at least N/M stakers have the data for txs in private meme-pool, and preventing withholding attacks unless N/M stakers are the same entity).
2. The B^{Sequencer}

is increased for each distinct prover who helps contribute to a block from within the proving committee.

1. Provers selected for the proving committee who repeatedly (e.g 3 times in a row) do not contribute to blocks can be slashed via a mechanism similar to the L1 Activity bleed on L1. This will need tuning to allow censorship by a large staker.
2. Once the final proof is made, anybody can submit to L1 and claim the B^{Submitter}

portion of the block reward. This is designed to cover costs associated with submitting e.g gas / call data.

Distributing work to provers based on a VRF

Option 1 Binary tree traversal**

Credit to Dan Lee for this idea.

If we take the proving VRF of any staker eligible, it is possible to define a path through the proof tree based on this.

An example for a rollup of 2^11

VRF = sha256(stakingPublicKey, previousCanonicalBlockHash)

VRF = 0x9d04a3ebfac363861c8d79235cb824f5984732027ab838b6d701de2d21671191

Assuming this is 32 bytes, we truncate the first 11 bits in binary and iterate through the tree to define a proving path.

Prover Transcript = 0011 0010 110

The tree traversal is enforced in the rollup circuits, constraining the user only receives a reward if they are merging proofs on their binary path. This helps ensure a large prover has to do far more work to include a subtree that is not theirs and will be incentivised to coordinate.

Example: take a path through the tree as 1111 1111 111

, for each level of the tree, to get credit for the 7th level of the tree, without including other provers work, a large prover would have to compute 2^6 other proofs for which they are not paid.

Option 2: Sorted VRFs

The same VRF used for sequencer selection is used for prover selection. The highest VRF will become the sequencer and the lowest N sequencers will be eligible for proving.

When committing to a block, the sequencer defines the tree height for the block. e.g 2^n

. This is used to determine the size of the proving set used for each round of proving.

Proving is split into rounds where each eligible prover computes a sub-tree of height m

, lets say 2.

Twice as many provers are selected as necessary to provide redundancy.

Let k

= the height of the tree still to be proved.

For tree of size 10:

Round 1

: The bottom 2^{n+1}

, 2^{11}

provers ranked by lowest VRF output selected. Each computes a subtree of 4, inlcuding VM proofs, and base rollup proofs.

Round 2

: The bottom 2^{k-m +1}

, 2^9

provers ranked by lowest VRF output selected. Each computes a subtree of 4 merge proofs.

Round 3

: The bottom 2^{k-m +1}

, 2^7

provers ranked by lowest VRF output selected. Each computes a subtree of 4 merge proofs.

Round 4

: The bottom 2^{k-m +1}

, 2^5

provers ranked by lowest VRF output selected. Each computes a subtree of 4 merge proofs.

Round 5

: The bottom 2^{k-m +1}

, 2^3

provers ranked by lowest VRF output selected. Each computes a subtree of 4 merge proofs.

Round 6

: The sequencer completes the tree.

Issues solved by this proposal

Proof stealing

A proving network design needs to address the case where Bob steals Alices proof. This is acheived via baking in a secret key into the proof that also controls stake, ensuring only Bob can be paid for this proof.

Liveness

A sequencer with commodity hardware, should be able to get a reasonable guarantee from the proving network, that their block will be produced, or they will not be able to fulfill their role as sequencer. The Aztec block lifecycle is longer than L1, and it is likely that each block, will be worth more in Rewards, Fees, and MEV as they span a longer time span.

Sequencers must be able to trust the proving coordination protocol to deliver, or they will forgo a large revenue opportunity.

This proposal adds a requirement for the entire staking set to help produce a block proof to help guarantee that a block will be proven.

Growing a large staking set

On Ethereum validators will propose a block every ~70days on average with a block time of 12s, and less as the set grows.

On Aztec if the block time is 120s, a single sequencer would only propose a block every 700 days with 500,000 stakers, and potentially every 2800 days if the block time is 10 minutes.

This document tries to add additional revenue opportunities to the staking set to mitigate against infrequent block proposing, when a sequencers VRF is not selected. This better utilises staker resources and increases staker % ROI, even with larger block times.

Ensuring data is available

This proposal presents two methods to ensure proof data is available and prevent with-holding attacks:

1. Modifying the canonical block ranking algorithm to require a signature from M/N of the elected proving committee. This forces a sequencer to broadcast data in order for their block to become canonical.
2. Incentivising sequencers to quickly disseminate data by increasing their reward based on the percentage of the proving committee that contributes to the block.

Why should stakers perform two roles?

Separation of concerns is usually better when it comes to complex systems. However for a system like Aztec if the role of proving is not carefully incentivised by the protocol, centralising forces will arise due to the economic incentives the protocol gives to sequencers via Fernet. (i.e the ability to sell the ordering and proving rights for a fee).

These centralising forces arise from:

1. Proposing a block as a sequencer is a revenue opportunity. Fees + block reward + ordering preference fee.
2. If a sequencer is unable to produce a block when their VRF output is eligible, they will have a worse ROI on their stake.
3. As such, rational economic sequencers will employ methods to maximise ROI including:
4. Running centralised proving infrastructure to ensure they can produce a full block
5. Paying multiple provers via a proving market
6. Selling the right to produce a block to someone better suited
7. Running centralised proving infrastructure to ensure they can produce a full block
8. Paying multiple provers via a proving market
9. Selling the right to produce a block to someone better suited

Learning from Ethereum

This design is very similar to how the beacon chain operates. On L1, stakers have two key roles:

1. Proposing blocks
2. Acting as part of a validating consensus committee

Aztec is a ZK L2, therefore a validating

does not need to exist. However computationally intensive proofs do need to be made.

If Aztec's rollup circuits and public VM circuits are optimised, the machine requirements for a prover should be possible to run at home e.g 16 cores + 32gb of ram.

Comparisons to other proposals

Cooperative proving network for Fernet

This protocol is a cooperative proving network and very similar toCooperative proving network for Fernet.

It's key differences are:

1. This proposal enforces that the staking set for sequencers and provers is the same.
2. This proposal elects a proving committee per block based on the low output VRF's
3. This proposal does not reward uncle proved blocks, instead it enforces an inactivity slash for elected provers who do not participate over a longer period.
4. This proposal does not have a burn model attatched to it (even though burn models are awesome…)
5. This proposal attempts to solve DA issues by either enforcing that a block will not become canonical without signoff from M/N provers, or by incentivising the sequencer more for each unique prover that participates. Both should serve to incentivise the sequencer to widley distribute proof data.

The VRF selection and fee burn model in Cooperative Proving network can be best thought of as a good alternative to Option 1 and Option 2 in this proposal of how to distribute work to provers via a VRF. It is likely the best bits from both proposals should be combined.

SideCar

This proposal is at the other end of the spectrum toSideCar. It removes all responsibility for putting a block on chain from the sequencer, after they have posted a block commitment. This seperation of concerns is beneficial for two reasons:

1. To limit with-holding attacks from the sequencer.
2. To fight against centralising forces that arise from sequencers with large proving power, or rely on a specific proving entity. In this proposal, a small sequencer has access to the same proving power as a large sequencer, in a decnetralised fashion thanks to Fernet. This promotes liveness and censorship resistance compared to a sequencer being reliant on either:
3. a large proving network controlled by the sequencer
4. a proving market that could change / de platform Aztec
5. a small set of large proving entities like L1 builders
6. a large proving network controlled by the sequencer
7. a proving market that could change / de platform Aztec
8. a small set of large proving entities like L1 builders
9. Seeks to better incentivse stakers, who in the current Fernet design will only produce a block every 700-2800 days.

Side car does have some advantages:

1. Sidecar may use compute more efficiently, but at the cost of liveness
2. Sidecar is easier to build
3. Sidecar will be better if the publicVM proof dominates and needs very large machines, i.e 128gb ram, and 64cores, however IMO this would be a failure and not promote decentralisation as Aztec proofs must run in a data centre.

Outstanding questions

Public VM benchmarks

This design, like other designs, has a big unknown on the benchmarks of the public VM. Given public VM proofs will dominate proof construction, machines will need to be sized to match this.

Assumptions

Throughput: 10 tx/s

Blocktime: 600 seconds

VM proofs per tx: 2

VM proof time: 10s

Machines required: ~400

Numbers taken from this model.

Optional Add On : BLS Validation Training Wheel

The proving committee concept can further be extended to define aBLS Validation Training Wheel. This would further re-use the same staking set, while the training wheel is active. The proving committee must also provide a signature over the blocks state transition.