While the B52 sequencer selection proposal includes a decentralised proving network as part of it's design,Fernet leaves it out of scope. Our goal here is to arrive at a design for a prover network that can be used by any sequencer selection algorithm.

## Why decentralise provers?

Written by @jaosef

Proving correct execution is orders of magnitude slower than simply executing a transaction. If a sequencer needs access to a very larger proving network simply to make a block, that is a very large centralising force.

Why can't we just post proofs to a worker queue with a bounty?

1. A sequencer who must produce a block within a certain window (as defined by the SSP) will not rely on one prover to do this, as they risk revenue. A rational sequencer will either ask many provers to compute the block, leading to redundant compute and extra costs for users, or run their own compute.

2. A rational sequencer will only pay the fastest prover, which means the second fastest prover does not get paid. Over time this may lead to centralisation with one giant prover. This creates bribe attacks, present, unless the protocol enforces all provers get paid.

## Aspects

Given the above, we can discuss different aspects of proving network, such as payments, selection, participation, etc.

### How are provers selected?

Let's assume there's a pool of valid provers to choose from at any time, already made sybil resistant via staking or a similar mechanism. How is a prover chosen to work on a given proof?

- The sequencer can be free to choose whatever prover(s) they want. This allows sequencers to leverage reputation or other off-protocol information to make their choice. This is similar to how UniswapX allows users to grant exclusiveness in filling an order to a particular executor (albeit for a limited amount of time), or clients pick relayers in the Gas Station Network.

- The chosen prover(s) can be decided by the protocol itself, such as via a VRF. This requires that the pool of provers is vetted in advance to ensure all provers in it can provide a reasonable quality of service. This is similar to how Livepeer picks Transcoders.

On B52 as it is today, provers submit intents to build a proof during a voting phase, and then the sequencer chooses which prover is allocated to which job. It is unclear how provers coordinate between each other, and we could end up in a situation where every prover signals intent to build the entirety of a block, and it's fully on the sequencer to choose how work is spread out.

### How are provers paid?

Provers need to be rewarded for their work. Now, prover payments can be handled in or out of protocol.

- Provers can be paid a fraction of the block fees depending on their contributions to their proofs tree. Given each proof can include the id of the prover that proved it, it should be easy to calculate this rate. This has the benefit that provers get paid only if they submit valid work, and verification is done as part of the protocol itself.

- Payments can be handled out of protocol, as a transaction between sequencer and provers. Sequencers could paid provers upon receipt of a valid proof, and we could require sequencers to submit a bond that is released once the prover sends back the proof, to avoid griefing. We could implement a payment-channel-like structure to avoid gas costs in these interactions, but it requires more work to build. On the other hand, keeping payments out of the main protocol gives us more flexibility in changing it.

Embedding payments into the protocol seems to be the easiest way to go. In this approach, provers could be grieved if their submitted proof is not included in the resulting tree, but we could set up a separate path in which a prover can claim uncle rewards by submitting their proof separately if it was neglected.

### How many provers are selected per unit of work?

We can choose how many provers are selected for each unit of work. Choosing a single one leads to the least wasted effort, but can affect liveness if that prover fails to deliver. Given long proving times, a missing proof delays an entire layer of the proofs tree. However, if we require multiple provers for each unit of work, all provers need to be rewarded (not necessarily equally), leading to higher fees overall.

Note that, if the sequencer is left to choose prover(s) on its own, we cannot enforce it to pick multiple provers unless we have a strong sybil resistance mechanism in place. On the other hand, if we allow the sequencer to choose a single prover each round, we could end up with a single centralised prover monopolising the network, as mentioned by Joe in the first paragraphs.

### How are provers vetted?

We can enforce requirements for a prover to be eligible for participation in the protocol.

- If sequencers are completely free to choose provers, then we could roll without any requirements. Anyone can just act as a prover by helping the sequencer assemble a proof.
- We can require a stake in the form of a fixed amount of tokens for participating, similar to validators in Ethereum. This allows the network to slash malicious behaviour by provers, even if we have not yet identified what that would look like, assuming we have a governance mechanism that permits it. It also helps prevent spam.
- If prover selection is mandated by the protocol and not sequencers, we need a stronger signal than just staking, since we need the network to filter out which provers are able to provide a reasonable service. Here we can require that token holders stake for specific provers, similar to transcoders in Livepeer or indexers in TheGraph. This will lead to a smaller number of provers but with a vetted quality of a service.

### How many provers do we expect in the network?

We want to encourage a scenario where we don't converge to a single prover dominating the market. This could happen if we prioritise provers exclusively based on performance and set up a winner-take-all mechanic, as described here:

Related is the problem of maintaining decentralization. One challenge with proof markets is prover undercutting. A well-funded prover could choose to operate at a loss by bidding below market rates to price other operators out of the market before growing and increasing prices. Another form of undercutting is operating an excessive number of nodes while bidding market rates, allowing random selection to match the operator with a disproportionate amount of proof requests.

This can be mitigated with uncle rewards, or enforcing a minimal proving time (as Mina does; thanks Anon for bringing it up) to level the playing field.

Related to this is hardware requirements. We expect provers not to require inaccessible machines, but we still don't know if they will require something slightly more powerful than consumer-grade machines (eg 32 cores).

## Looking into other rollups

Given we're not the only zk rollup out there with plans for decentralisation, it's worth looking into what others are doing.

### Mina

Mina's prover network is handled through the snarketplace, a marketplace where block producers (chosen through a VRF) purchase SNARK proofs from workers. These purchases are entirely off-protocol, and the market favours the worker that offers the lowest fees, not the fastest one.

There is no protocol involvement in pricing snarks, nor are there any protocol level rewards for snark workers to produce snarks. The incentives are purely peer-to-peer, and dynamically established in a public marketplace, aka the snarketplace.

The risk of centralisation around a single entity is mitigated by the fact that the barrier to entry is low.

Is there any concern about a single large SNARK worker dumping work in the snarketplace, and then raising prices after monopolizing the market?

[…] this strategy is effective only in markets where there are high barriers to entry. Meaning, competitors who were crowded out in the predation stage are unable to rejoin the market.

This is not the case for SNARK work because the barriers to entry are low. Anyone who has spare compute can join the snarketplace and produce as little as one SNARK work and profit on that unit of work. The only barrier to entry is the initial capital expense on hardware, but hardware requirements are low so that users with spare equipment can come online and participate. If any SNARK worker succeeds in driving out the market and increases prices, newcomers are anticipated to reappear and drive prices back down.

### Taiko

Taiko seems to have a very different prover environment than Mina's: while Mina shoots for making it easy for anyone to become a prover since proofs are small, Taiko has a high barrier to participation:

Keep in mind that running a prover on mainnet will be a very competitive space, and those who generate proofs will have very powerful machines. It's unlikely anyone will be able to be a "casual prover".

A previous version of Taiko favored a winner-takes-all approach to submitting proofs, but this was leading to a scenario where a single prover would prevail, affecting liveness. A testnet run seems to have confirmed this. Hence, their new approach is to keep a pool of 32 provers, ranked based on stake and fees requested, and use a VRF to choose among them. Failing to submit a proof by the chosen prover leads to slashing.

## Starknet

Starknet went through a similar process as this one a few months ago, weighing tradeoffs of different approaches for decentralising proving.

How to best enjoy the benefits of competition without risking centralization?

In contrast to competition, turn-based monopolies resolve incentive problems well, but may not discover the best candidate for each job or incentivize improvement. A glaring question is: what are the turns based on? Stake-based turns allow stakers to choose provers. This does not seem problematic in principle, but the "classical" role of stake is to provide Sybil resistance only for the consensus layer. What are some other interesting ways of realizing turn-based monopolies? How to incentivize improvement in such a model?

A turn-based model can also be competitive, e.g a turn-based auction. For example, Ethereum's designation of labor for sequencers is essentially a turn-based auction, with the winner being the first sequencer willing to sell block space to a transaction in exchange for its fee.

And more recently:

When thinking of auctions, we felt an L1 auction contract is too cumbersome and expensive, while off-chain auctions may be vulnerable to censorship. Coupling proof production to block circumvents these issues. Lastly, even if you have some proofs decoupled from block production, it seems a turn-based system based on stake is still simpler to manage than an auction.

## nil

While not a zk rollup in itself, nil offers a proof market where buyers can purchase ZKPs from a set of provers. Given an order with a given maximum price and time to fulfill, the market automatically matches it with a prover given their offered price.

Here are the basic rules of order matching for a request with specified time t_r

and price c_r

:

- Choose proposals with t_p

≤ t_r

and c_p

≤ c_r

.

- Sort them by price and proof generation time. The user's request defines the priority of these parameters.
- Apply proof producer's rating to the sorted list.
- Choose the top proposal from the list.

Provers are required to stake to participate, and are penalised if they fail to fulfill their obligations. The market also keeps a reputation system for provers, though it's under development. Whether this incentive structure could lead to centralisation is unclear from the docs I could find.

## Scroll

Thanks to Toghrul for taking the time to share his thoughts with us. I'll try to capture our discussion here, and hope I'm not getting anything wrong!

Scroll's main assumption is that MEV will be the main driver of revenue of their chain, so their designs goal is that sequencers do not keep all that MEV to themselves and instead share it fairly with provers. This leads to two approaches:

1. Multiple sequencers are elected per round, and they post bids to the proving network. The proving network chooses the best bid for them, forcing sequencers to compete amongst themselves and sharing the value captured. In this model, which prover is elected to manufacture a proof is decided via a VRF, and provers are required to stake to participate.

2. Provers themselves are the block proposal leaders, removing the role of the sequencer as the proposer. The prover is elected by the proposer selection algorithm directly.

There's also emphasis in having a consensus layer at L2 to grant strong ordering guarantees before committing to L1, similar to the Espresso Martini sequencer selection proposal.

## Going back to Aztec

Let's go back to what we can build for Aztec. We can start with this relates toB52 and Fernet, the current candidates for sequencer selection.

### B52

B52 forces block builders to share their captured MEV with provers by putting forward bids to be voted on by provers. The winning builder then selects provers among the set of their voters, and is incentivised to maximise the number of unique provers participating in that block, while choosing provers with good reputation in order not to risk liveness (or they lose their reward). Prover selection is then left entirely to the block builder. Note that prover votes are weighted by a VRF to prevent a centralised actor from taking over the proving network by spawning as many provers as work units on each block.

This approach incentivises builders to compete for extracting the most MEV (either directly or via a mev-boost-like external PBS mechanism) while sharing it with the proving network.

### Fernet

Fernet doesn't define a proving network within its scope. B52's approach is not directly applicable to the proposal, since the winning block proposer has exclusive rights to the block contents, as long as they can produce a proof.

A marketplace, similar to Mina's or nil's, would be a good fit for Fernet, since it's orthogonal to the rest of the protocol. It would require additional work to integrate, and it's unclear how decentralised it would get. As a live example, Mina's Snarketplace seems to lead to a powerlaw distribution of workers.

### Protocol-elected provers

In both scenarios for B52 and Fernet, it's the winning block builder or sequencer who chooses the provers. We can instead propose a protocol that picks the prover for each work unit with a turn-based system, which could be applicable to both.

On each block, we can rely on a VRF for deciding which prover in the pool would be elected to generate each unit of work. We begin by choosing which prover works on which leaf of the tree. Then, for each pair of adjacent nodes, one of the two provers chosen for those nodes is picked to build the merge node, and so forth until the root of the tree. If a given prover fails, its sibling is incentivised to step in and produce their proofs, so their subtree is included; in this scenario, the failing prover gets no reward, and the sibling gets rewarded only for the nodes that were assigned for them.

This approach ensures a fair distribution of work across the proving network, and provers cover for each other in the face of liveness issues. We could also add some form of slashing if a prover fails to deliver, though we risk a prover griefing another by just not including their proof and causing a slash.

In B52, this approach can be used to selecting which provers amongst the voters are selected for building the block, and how the workload is distributed. Rather than having provers vote for specific nodes, provers would then vote for the proposal they like the most, the builder would consolidate all

votes (regardless of participation in proof generation), and from that set pick the provers that will do the actual work through this algorithm.

In Fernet, since there are no competing block proposals in each slot, this approach is applied across the entire prover pool. A missing piece, however, is how fees are shared between sequencer and provers. We could patch the sequencer scoring function (currently only a VRF) to factor in the percentage of fees the sequencer pushes to the provers. We can expect that, as the network evolves, MEV would become more of a driving force in terms of revenue, so sequencers would eventually share 100% of fees to provers to increase their chances of getting elected and extracting MEV. This assumes that protocol fees only are enough for keeping the prover network running.

Work in progress.