Sidecar Proving Proposal

Summary

I propose that we facilitate a very basic commitment and slashing scheme to delegate and coordinate proving rights for a given Aztec block. It's expected that the sequencer chooses among a variety of 3rd party proving quotes, facilitated via an out of protocol marketplace. The Prover gets decided and signaled to the community by a staked proverDeposit on L1, paid for by the current sequencer. Proving then runs as a "sidecar" to the protocol, eventually published to Ethereum for verification and finalization. In the event that the block doesn't get proven, the collateralized proverDeposit will get slashed.

Design goals

1. Enable the community to iterate on proving marketplace designs outside of the scope of Aztec governance or upgradability.
2. Enable the network to dynamically adjust it's proving capacity, e.g. if a large portion of the proving network stops working at block N, or many wish to join, any number of new provers can start working at block N+1.

Details

[

Aztec-Prover-RFP-Sidecar-Proposal

2257×800 246 KB

](https://europe1.discourse-cdn.com/business20/uploads/aztec/original/1X/719d06f6e76fe13a407ea6c8776863acc09426d9.png)

Expanding Fernet's proposal phase

In order for this proposal to distribute clear incentives to participants, it suggests the sequencer should add a tip and percentage of block rewards that they will share with the prover of this particular block. The tip could be $ETH, or any ERC-20 token. These would be deposited and defined via the same L1 transaction, e.g. a proverTip

, and proverRewardShare

, in which sequencers put up block proposals. In the event that a sequencer with a higher VRF output does reveal their data and their block proposal does end up becoming canonical, the proverTip

gets refunded.

Estimated duration: 3-5 Ethereum blocks

Prover Commitment Phase

This proposal suggests adding another phase, after Fernet's proposal and reveal phases, called the Prover Commitment

Phase. During the Prover Commitment

Phase, it's expected that the sequencer participates in an out of protocol decision making process, which would be provided via open source software that can be run ontop of sequencer software (sort of like mev-boost).

By the end of the Prover Commitment Phase, any sequencer that believes they have a chance to win block proposal rights must signal via an L1 transaction a particular Ethereum address (EOA or otherwise), called the proverAddress

, and specify a proverDeposit

in a particular ERC-20 token for the right to prove. The prover deposit is necessarily large, e.g. $1,000 - $10,000+ of an ERC-20 token, however the exact amount is tbd. This is due to the fact that the proverDeposit may practically becomes the cost to reorg or stall the network for the duration of the block (e.g. 10m), via the act of withholding or not producing proofs. For more information please refer to items #1

& #2

in the known issues and mitigations section.

Who pays the proverDeposit?

TLDR; the sequencer, given it's their decision which marketplace or service provider to use.

The out of protocol decision making process would be up to the sequencers and opt-in. So, in a naive case, sequencers can build blocks of a size that they can self-prove during the proving window, and put down their own proverDeposits. It is likely that all

clients would implement this as a fallback option, similar to geth and other proposer client implementations. It's possible this is just a block of a handful of transactions, given proving costs. In a more sophisitcted case, sequencers can participate in an out-of-protocol first price auction, and make the participants of the auction put up the proverDeposits for the sequencers to publish to L1. It is expected that 3rd party proving marketplaces or services, such as Nil or Gevulot, compete directly in this process either via native integration, or a relayer.

To restate, as defined the current sequencer has discretion to decide who gets to prove the current block. This means that they have to put up the collateralized proverDeposit, which could get slashed if the sequencer makes a poor decision delegating proving rights. If they do not put up a proverDeposit by the end of the Prover Commitment Phase, they lose their block production rights to the next highest ranking sequencer who does.

In practice, I imagine a few categories or persona's of sequencer operators, who would potentially handle this prover deposit differently. For a few examples: a larger entity would likely vertically integrate and self-prove the current block, and therefore put up the proverDeposit themselves. An average consumer sequencer may not be interested in taking further slashing risk and therefore may choose from a reasonable trusted list of service provers, relayers, etc. and require them to send the funds required to fulfill the proverDeposit out of protocol. A sophisticated sequencer may participate in the RFQ and due to their confidence in proof fulfillment take the slashing risk themselves and not worry about a proactive payment to cover the proverDeposit. It is likely there are other scenarios, but ultimately this should help paint the spectrum of options for paying the proverDeposit.

Estimated duration: 3-5 Ethereum blocks

Proving Phase

In this proposal, the actual "work of proving" happens outside of the Aztec network & protocol. Therefore there's nothing else that needs to be done at the moment. The protocol simply waits a prespecified amount of time for the proof submission phase to begin.

Estimated duration: 40 Ethereum blocks (8 min)

Enforcing a maximum upper bound on proving times

One design consideration is whether or not to explicitly define the duration of the proving phase, or alternatively only enforce a maximum upper bound (e.g. 10 minutes). This would lead to variable or dynamical block sizes, proving durations, and block times. Which means the number of sequencers per day is no longer well known in advance, but rather can only be roughly estimated or approximated. Notably this may be aligned with other proposals and/or expectations already.

Proof Submission Phase

After the proofs have been completed, they are assumed to be shared via the L2 p2p network. Anyone can submit these proofs to L1 during the submission phase. It is expected that this will be done by either the sequencer, &/or the proving entity/network, however, this could be anyone. There is a token reward given to the first address to do so, to offset the L1 gas costs of submission. The specific quantity of the reward is tbd, but should reflect the cost of submission and the trouble of operating the relevant infrastructure.

Estimated duration: 3-5 Ethereum blocks

Notably the duration here is generally dependent on the ability to submit4844 blobs. It is possible that allowing a prover to upload proofs as they go, i.e extending the proof submission phase or overlapping it with others, or some kind of proof batching may be necessary.

Rewards & Incentives

At a high level, rewards would work as follows: once the block has been validated on L1, the ERC-20 tokens (proverTip

- proverRewardShare

) that were "committed" to before during the proposal phase will be distributed via the Ethereum address defined earlier. The Sequencer will get a marjority of newly minted the block rewards, minus those that they're sharing with the address that did the proving (which could be themselves, if vertically integrated). Lastly, there are additionally incentives to the ethereum address that published the completed proofs to L1, to offset submission costs Again this could be the randomly elected sequencer themselves, and in a very

vertically integrated case, they could earn 100% of rewards and MEV for a block.

Estimated duration: 1 Ethereum block.

Total estimated block times (for finality): 48 - 72 Ethereum blocks

Another fancy diagram

[

block-prod-lifecycle-sidecar

1920×527 127 KB

](https://europe1.discourse-cdn.com/business20/uploads/aztec/original/1X/3ec7d56ea34b7a531a6ea20f65a9b9cda6549f21.jpeg)

Confirmation or finality rules

Users could potentially get different levels of “transaction status updates” via user interfaces, or broadly “finality guarantees” at various phases throughout this lifecycle. I think it can be generally understood as:

Executing

1. Transaction is signed &/or executed locally

Sent &/or Processing

1. Transaction is in the p2p mempool
2. Transaction is included in a proposed block
3. Transaction is in a revealed block
4. Transaction is in a revealed block with a valid proving commitment
5. (reasonably long delay for proving…)
6. Transaction is included in a completed proof on the L2 p2p network

Published &/or Verifying

1. Transaction is published to L1
2. Transaction is verified on L1

Finalized

1. Transaction is finalized on L1

Users in this case could get application specific (depending on trust assumptions & user experience needs) notices of their transactions making progress within 30 seconds - 1 minute. Then further progress alerts via confirmation or increased probability of their transaction’s successfully finalizing throughout these other various points in the block production lifecycle, that may or may not be relevant. In extreme cases, users and application developers may want to wait until the rollup’s block has been verified “deep enough” in the Ethereum blockchain, to accmoodate for potential L1 reorgs, as well. e.g. 1-2 further epochs.

Comparisons

To existing Aztec proposals

Unlike Palla’s suggestion of a cooperative prover network, this design specifically attempts to create a competitive & dynamic economic marketplace for proving rights. This is also true for anon’s proposal, as well, which implements an in-protocol “bonded” auction. Generally, the design goals for Sidecar to ensure that the latest proving marketplace dynamics can be iterated on outside of the protocol, similar to MEV auctions, and can continue at the rapid pace of innovation within the industry.

To other projects

This design is similar to Taiko.

Some high level differences (among many others) between this & Taiko’s model include:

- Sidecar does not (currently) attempt to define the specific interfaces that must be implemented by “Prover Pools” (or those participating in the prover commitment phase/out of protocol auction). This could be a nice addition, however, and is worth further consideration!
- Sidecar suggests a generally different economic and incentive model. In Taiko, in the event that proofs are not completed 3/4 of their “bond” or insurance collateral is permanently burned. In this model, a specific staked proverDeposit is slashed and distributed to other network participants (dynamics of which are slightly tbd, but generally redistributed to the sequencers affected by the likely reorg that was caused).
- Their documentation is worth reading.

Outstanding questions + known issues & mitigations

1. Should the right to prove be all or nothing?
2. Currently the RFQ specifically delegates the entirety

of proving to a single ethereum address (i.e prover or proving marketplace). It seems potentially desirable to be able to dynamically configure the portions of the proof tree (or subtree) that individual “provers” work on. But again, this could be done out of the protocol… * If going down this path, it becomes significantly closer to a cooperative proving network or the orignal B52 design of providing a block transcript. This leads to more coordination complexity and potential feasability of a withholding attack. * I would consider this a viable candidate for an alternative proposal.

- I would consider this a viable candidate for an alternative proposal.
- If going down this path, it becomes significantly closer to acooperative proving network or the orignal B52 design of providing a block transcript. This leads to more coordination complexity and potential feasability of a withholding attack.
- I would consider this a viable candidate for an alternative proposal.
- I would consider this a viable candidate for an alternative proposal.
- High price of proverDeposits
- This is related to the above issue, as well. Currently this proposal delegates 100% of a block’s proving rights to a single ethereum address. It is then up to that entity to out of protocol further divide and conquor the necessary work, have a very large

set of machines that can produce the proofs necessary, or alternatively, build very small blocks. One option to reduce this is to standardize the size of a block and the number of subtrees that need to be proven, e.g N subtrees would require N commitments by the end of the prover commitment phrase, reducing the cost by the same number. However, as I currently understand, this does increase cooridination complexity and the feasability of a (malicious or unintentional) withholding attack that could cause a reorg. Therefore I currently think a single out of protocol delegation as currently defined may be sufficient to begin with, and hope others iterate on the initial idea.

1. Explicitly defining what happens in the event of a reorg.
2. The short answer is that the network would invalidate all blocks that were proposed during the reorgWindow

and start a new proposal phase. The sequencers affected would be proportionately distributed the slashed stake, from the proverDeposit

provided by the sequencer who caused the reorg.

- TO DO: Need to spend some time on diagrams and explaining the possible scenarios.
- Impact on upgrade mechanisms
- This design intends to ensure that the proving systems and marketplaces can continue iterating outside of the scope of the Aztec protocol and therefore the Aztec upgrade mechanism. Which is great and should satisfy that specific requirement. However, in the event that Aztec chooses to implement an upgrade mechanism like the republic or the return of the jedi I’d highlight that the “prover RFQ &/or commitment phase” should remains somewhat upgradable (as much as possible, likely via a separate contract referenced in the version registry). This is in the event that out of protocol proving RFQ’s have unintended or unknown consequences, cryptography innovation slows down, or it becomes desirable to further enshrine for whatever hypothetical reason (e.g. network and incentive alignment).
- Is it worth normalizing &/or predefining block times?
- As discussed briefly above, “the protocol could not define the duration of the proof submission phase but rather a maximum upper bound. This would lead to variable or dynamical block sizes, proving durations, and therefore variable

block times." There are some tradeoffs here worth further consideration. Generally, dynamic blocks could lead to smaller blocks that could be produced quicker, and therefore potentially provide better user experience. It additionally may better reflect the ability to dynamically adjust and scale compute resources. Enforcing via an upper bound seems like a reasonable solution, and application developers can rely on that as a fallback/guarantee of when to query the network and update their state. This is something that should likely be considered in network simulations or modeling.

- Another valid design consideration would be to remove preconfirmations, and have new proposal phases begin immediately after the last block is confirmed.
- The obvious question … enshrine the marketplace?
- This design is predicated on some kind of out of protocol marketplace &/or decision making process for the sequencer. This proposal specifically does not attempt to design a marketplace! But many will likely ask, why not? And how do you know an out of protocol marketplace will be sufficient? My answer to this generally is… I don't currently know what a good marketplace design would be, and if we start working towards implementing sidecar, we can continue working on marketplace designs all the way up to mainnet (or beyond). We follow the spirit of Vitalik's "minimum viable enshrinment".
- Factoring proverDeposits into sequencer scores
- The currrent sequencer is ranked based on the RNG they generated via the VRF output inFernet. It could make sense to consider the proverDeposit as a ranking/scoring factor as well, to enable a more competitive marketplace for the rights to block production, therefore potentially increasing the cost of stalling or reorging the network.

References

1. https://boost.flashbots.net/
2. https://nil.foundation/
3. https://www.gevulot.com/
4. [Proposal] Cooperative proving network for Fernet
5. [Proposal] Provers: Bonded Prover Auction
6. Proving Taiko blocks – Taiko
7. Request for Proposals: Upgrade Mechanisms
8. The Republic: A Flexible, Optional Governance Proposal with Self-Governed Portals
9. The Return Of The Jedi