Overview

This document outlines a proposal for Aztec's block production, integrating immutable smart contracts on Ethereum's mainnet (L1) to establish Aztec as a Layer-2 Ethereum network. It aims to serve as the latest source of truth for the Fernet sequencer selection protocol, and reflect the decision to implement the Sidecar prover coordination protocol.

Notably, this is written as a proposal within the context of the first instance or deployment of Aztec. Therefore, the definitions and protocol may evolve over time with each version or release. The "latest source of truth" once an initial implementation is available will be either the GitHub codebase &/or the documentation website.

How it works

Sequencers can register permissionlessly via Aztec's L1 contracts, entering a queue before becoming eligible for a random leader election ("Fernet"). Sequencers are free to leave, adhering to an exit queue or period. Roughly every 7-10 minutes (subject to reduction as proving and execution speeds stabilize and/or improve) sequencers create a random hash using RANDAO and their public keys. The highest-ranking hash determines block proposal eligibility. Selected sequencers either collaborate with third-party proving services or self-prove their block. They commit to a prover's L1 address, which stakes an economic deposit. Failure to submit proofs on time results in deposit forfeiture. Once L1 contracts validate proofs and state transitions, the cycle repeats for subsequent block production (forever, and ever…).

Get involved

Please leave comments, suggested improvements, or questions in this forum by Friday, February 16, 2024. Contact cooper@aztecprotocol.com if you wish to schedule a meeting.

Participants

Full Nodes

Aztec full nodes are nodes that maintain a copy of the current state of the network. They fetch blocks from the data layer, verify and apply them to its local view of the state. They also participate in the P2P network to disburse transactions and their proofs. Can be connected to a Private Execution Environment (PXE

) which can build transaction witness using the data provided by the node (data membership proofs).

Estimated hardware requirements

Minimum

Recommended

CPU

16cores

32cores

Network

32 mb/s

128 mb/s

Storage

3TB

5TB

RAM

32gb

64gb

These hardware requirements are current best guesses

and subject to change.

Sequencers

Aztec Sequencer's are full nodes that propose blocks, execute public functions and choose provers, within the Aztec Network. It is the actor coordinating state transitions and proof production, helping the rollup progress it's state forward. Aztec is currently planning on implementing a protocol called Fernet (Fair Election Randomized Natively on Ethereum trustlessly), which is permissionless and anyone can participate. Additionally, sequencers play a role participating within Aztec Governance, determining how to manage protocol upgrades. The details of Fernet are further articulated below.

Estimated hardware requirements

Minimum

Recommended

CPU

16cores

32cores

Network

64 mb/s

256 mb/s

Storage

3TB

5TB

RAM

32gb

64gb

These hardware requirements are current best guesses

and subject to change.

Provers

An Aztec Prover is most often a full node that is producing Aztec-specific zero knowledge (zk) proofs (rollup proofs). Notably, they do not have to be full nodes in thee vent that they have reliable access to a full node, or the latest Aztec state. The current protocol, called Sidecar, suggests facilitating out of protocol proving, similar to out of protocol PBS. Provers in this case are fully permissonless and could be anyone - such as a vertically integrated sequencer, or a proving marketplace such as nil, gevulot, or kalypso, as long as they choose to support the latest version of Aztec's full nodes and proving system.

Estimated hardware requirements

Minimum

Recommended

CPU

16 cores

32cores

Network

64 mb/s

256 mb/s

Storage

3TB

5TB

RAM

32gb

64gb

These hardware requirements are current best guesses

and subject to change.

Other types of network nodes

- Validating Light nodes
- Maintain a state root and process block headers (validate proofs), but do not store the full state.
- The L1 bridge is a validating light node.
- Can be used to validate correctness of information received from a data provider.
- Maintain a state root and process block headers (validate proofs), but do not store the full state.
- The L1 bridge is a validating light node.
- Can be used to validate correctness of information received from a data provider.
- Transaction Pool Nodes
- Maintain a pool of transactions that are ready to be included in a block.
- Maintain a pool of transactions that are ready to be included in a block.
- Archival nodes
- A full node that also stores the full history of the network
- Used to provide historical membership proofs, e.g., prove that x

was included in block n

.

- In the current model, it is expected that there are standardized interfaces by which well known sequencers, i.e., those operated by well respected community members or service providers, are frequently and regularly uploading historical copies of the Aztec state to immutable and decentralized storage providers such as: IPFS, Filecoin, Arweave, etc.
- A full node that also stores the full history of the network
- Used to provide historical membership proofs, e.g., prove that x

was included in block n

.

- In the current model, it is expected that there are standardized interfaces by which well known sequencers, i.e., those operated by well respected community members or service providers, are frequently and regularly uploading historical copies of the Aztec state to immutable and decentralized storage providers such as: IPFS, Filecoin, Arweave, etc.
- The specific details of such additional node types/participant roles is TBD and likely to be facilitated via RFP, similar to how sequencer selection or prover coordination was determined.

Registration

Sequencers must stake a to be determined amount of a native token on Layer-1 to join the protocol. Within the initial Fernet implementation, an entryPeriod

can be used for limiting the ability to quickly get outsized influence over governance decisions, but is not strictly necessary. It’s expected this entryPeriod is initially set at 7 days.

In the initial implementation, Provers don’t need to register via staking on L1 contracts like Sequencers do, but must commit a bond during the prover commitment phase

articulated below. This ensures economic guarantees for timely proof generation, and therefore short-term liveness. If the prover is unable or unwilling to produce a proof for which they committed to in the allotted time their bond will be slashed.

sequenceDiagram

participant Anyone participant Contract as Aztec L1 Contract participant Network as Aztec Network

Anyone ->> Contract: register as a sequencer Anyone --> Anyone: Wait 7 days Anyone ->> Network: eligible as a sequencer

Looking beyond the initial implementation, Aztec may choose to implement a consensus network within the Layer-2 sequencers, and in that case would likely need to add a registration queue to ensure stability of the sequencer set while producing blocks (please refer to the future improvements section for more information). It is also possible that registration for provers could be added in the future.

Block production overview

[

image

1760×518 182 KB

](https://europe1.discourse-cdn.com/business20/uploads/aztec/original/1X/24092abd01e13c4aec20e591d5687068075d9787.png)

Every staked sequencers participate in the following phases, comprising an Aztec slot:

1. Proposal:

Sequencers generate a hash of every other sequencer’s public keys and RANDAO values. They then compare and rank these, seeing if they possibly have a “high” ranking random hash. If they do, they may choose to submit a block proposal to Layer-1. The highest ranking proposal will become canonical.

1. Prover commitment:

After an off-protocol negotiation with the winning sequencer, a prover submits a commitment to a particular Ethereum address that has intentions to prove the block. This commitment includes a signature from the sequencer and an amount X of funds that get slashed if the block is not finalized.

1. Reveal:

Sequencer uploads the block contents required for progressing the chain to whatever DA layer is decided to be implemented, e.g., Ethereum’s 4844 blobs. * It is an active area of debate and research whether or not this phase is necessary, without intentions to implement “build ahead” or the ability to propose multiple blocks prior to the previous block being finalized. A possible implementation includes a block reward that incentivizes early reveal, but does not necessarily require it - turning the ability to reveal the block’s data into another form of potential timing game.

1. It is an active area of debate and research whether or not this phase is necessary, without intentions to implement “build ahead” or the ability to propose multiple blocks prior to the previous block being finalized. A possible implementation includes a block reward that incentivizes early reveal, but does not necessarily require it - turning the ability to reveal the block’s data into another form of potential timing game.

2. Proving:

The prover or prover network coordinates out of protocol to build therecursive proof tree. After getting to the last, singular proof that reflects the entire blocks’s state transitions they then upload the proof of the block to the L1 smart contracts.

1. Finalization:

The smart contracts verify the block’s proof, which triggers payouts to sequencer and prover, and the address which submits the proofs (likely the prover, but could be anyone such as a relay). Once finalized, the cycle continues! * For data layers that is not on the host, the host must have learned of the publication from the Reveal

before the Finalization

can begin.

1. For data layers that is not on the host, the host must have learned of the publication from the Reveal

before the Finalization

can begin.

1. Backup:

Should no prover commitment be put down, or should the block not get finalized, then an additional phase is opened where anyone can submit a block with its proof, in a “based-rollup” or backup mode. In the backup phase, the first rollup verified will become canonical, and the cycle will begin with the next slot’s proposal phase.

sequenceDiagram

participant Contract as Aztec L1 Contract participant Network as Aztec Network participant Sequencers participant Provers

loop Happy Path Block Production Sequencers --> Sequencers: Generate random hashes and rank them Sequencers ->> Contract: Highest ranking sequencers propose blocks Note right of Contract: Proposal phase is over! Contract ->> Network: calculates highest ranking proposal Sequencers ->> Provers: negotiates the cost to prove Sequencers ->> Contract: commits to a proverAddress Provers ->> Contract: proverAddress deposits slashable stake Note right of Network: "preconfirmed" this block is going to be next! Sequencers --> Sequencers: executes public functions Provers --> Provers: generates rollup proofs Provers ->> Contract: submits proofs Contract --> Contract: validates proofs and state transition Note right of Contract: "block confirmed!" end

Exiting

In order to leave the protocol sequencers can exit via another L1 transaction. After signaling their desire to exit, they will no longer be considered active

and move to an exiting

status.

When a sequencer moves to exiting

they are no longer eligible for block proposal elections. They additionally might have to await for an additional delay before they can exit., eg 3-7 days. Notably this delay may be in addition to any delays that are required to exit stake from governance or the network’s upgrade mechanism.

sequenceDiagram

participant Anyone as Sequencer participant Contract as Aztec L1 Contract participant Network as Aztec Network

Anyone ->> Contract: exit() from being a sequencer Note left of Contract: Sequencer no longer eligible for Fernet elections Anyone --> Anyone: Wait 3-7 days Anyone ->> Network: exit successful, stake unlocked

Confirmation rules

There are various stages in the block production lifecycle that a user and/or application developer can gain insights into where their transaction is, and when it is considered confirmed.

Notably there are no consistent, industry wide definitions for confirmation rules. Articulated here is an initial proposal for what the Aztec community could align on in order to best set expectations and built towards a consistent set of user experiences/interactions. Alternative suggestions encouraged!

Below, we outline the stages of transaction confirmation:

1. Executed locally
2. Submitted to the network
3. At this point, users no longer need to actively do anything
4. At this point, users no longer need to actively do anything
5. In the highest ranking proposed block
6. In the highest ranking proposed block, with a valid prover commitment
7. In the highest ranking proposed block with effects available on the DA Layer
8. In a proven block that has been verified / validated by the L1 rollup contracts

9. In a proven block that has been finalized on the L1

sequenceDiagram

participant Anyone as User participant P2P Network participant Sequencer participant Network as Aztec Network participant Contract as Ethereum

Anyone --> Anyone: generates proof locally Anyone ->> P2P Network: send transaction P2P Network --> Sequencer: picks up tx Sequencer ->> Network: sequencer puts tx in a block Network --> Network: executes and proves block Network ->> Contract: submits to L1 for verification Contract --> Contract: verifies block Contract --> Contract: waits N more blocks Contract --> Contract: finalizes block Network --> Contract: updates state to reflect finality Anyone ->> Network: confirms on their own node or block explorer

Economics

In the current Aztec design, it's expected that block rewards in the native token are allocated to the sequencer, the prover, and the entity/address submitting the rollup to L1 for verification. Sequencers retain the block's fees and MEV (Maximal Extractable Value). A potential addition in consideration is the implementation of MEV or fee burn, discussed at the end of this post. The ratio of the distribution is to be determined, via modeling and simulation.

Future Aztec versions will receive rewards based on their staked amount, as determined by the Aztec upgrade mechanism. This ensures that early versions remain eligible for rewards, provided they have active stake and usage. Changes to the network's economic structure, especially those affecting block production and sequencer burden, require thorough consideration due to the network's upgrade and governance model relying on an honest majority assumption and at a credibly neutral sequencer set for "easy" proposals.

With the rest of the protocol mostly

well defined, Aztec Labs now expects to begin a series of sprints dedicated towards economic analysis and modeling with Blockscience throughout Q1-Q2 2024. This will result in a public report and associated changes to this documentation to reflect the latest thinking.

Mev-boost

Within the Aztec Network, "MEV" (Maximal Extractable Value) can be considered "mitigated", compared to "public" blockchains where all transaction contents and their resulting state transitions are publicly visible or computable by all network participants. In Aztec's case, MEV is generally

only applicable to public functions and those transactions that touch publicly viewable state.

It is expected that any Aztec sequencer client software will initially ship with some form of first price or priority gas auction for transaction ordering, ie a "naive" or unsophisticated mechanism. Meaning that in general, transactions paying higher fees will get included earlier in the network's transaction history. Similar to Ethereum's Layer-1 ecosystem, eventually an opt-in, open source implementation of "out of protocol proposer builder separation" (PBS) such as mev-boost will likely emerge within the community, giving sequencers an easier to access mechanism to earn more money during their periods as sequencers. This is an active area of research.

Proof-boost

It is likely that this proving ecosystem will emerge around a [flashbots mev-boost][https://boost.flashbots.net/] like ecosystem, specifically tailored towards the needs of sequencers negotiating the cost for a specific proof or set of proofs. Currently referred to as proof-boost

or goblin-boost

(due to goblin plonk…).

Specifically, Proof boost is expected to be open source software sequencers can optionally run alongside their clients that will facilitate a negotiation for the rights to prove this block, therefore earning block rewards in the form of the native protocol token. After the negotiation, the sequencer will commit to an address, and that address will need to put up an economic commitment (deposit) that will be slashed in the event that the block's proofs are not produced within the alloted timeframe.

[

image

956×677 66.4 KB

](https://europe1.discourse-

cdn.com/business20/uploads/aztec/original/1X/e1bed038fddfeb27f2b9e77b8486981b83ee64b3.png)

Initially it's expected that the negotiations and commitment could be facilitated by a trusted relay, similar to L1 block building, but options such as onchain proving pools are under consideration. Due to the out of protocol nature of Sidecar, these designs can be iterated and improved upon outside the scope of other Aztec related governance or upgrades - as long as they maintain compatibility with the currently utilized proving system(s). Eventually, any upgrade or governance mechanism may choose to enshrine a specific well adopted proving protocol, if it makes sense to do so.

Constraining Randao

The RANDAO

values used in the score as part of the Proposal phase must be constrained by the L1 contract to ensure that the computation is stable throughout a block. This is to prevent a sequencer from proposing the same L2 block at multiple L1 blocks to increase their probability of being chosen. Furthermore, we wish to constrain the RANDAO

ahead of time, such that sequencers will know whether they need to produce blocks or not. This is to ensure that the sequencer can ramp up their hardware in time for the block production.

As only the last RANDAO

value is available to Ethereum contracts we cannot simply read an old value. Instead, we must compute update it as storage in the contract. The simplest way to do so is by storing the RANDAO

at every block, and then use the RANDAO

for block number - n

when computing the score for block number n

. For the first n

blocks, the value could pre-defined.

Known issue: RANDAO's Biasability. Further reading: 1, 2, 3. At the moment we believe that this is not a serious issue, however improvements would likely be considered &/or prioritized.

Diagrams

Happy path

sequenceDiagram

participant Anyone participant Contract as Aztec L1 Contract participant Network as Aztec Network participant Sequencers participant Provers

Anyone ->> Contract: register() Anyone --> Anyone: Wait 7 days Anyone ->> Network: eligible as a sequencer loop Happy Path Block Production Sequencers --> Sequencers: Generate random hashes and rank them Sequencers ->> Contract: Highest ranking sequencers propose blocks Note right of Contract: Proposal phase is over! Contract ->> Network: calculates highest ranking proposal Sequencers ->> Provers: negotiates the cost to prove Sequencers ->> Contract: commits to a proverAddress Provers ->> Contract: proverAddress deposits slashable stake Note right of Network: "preconfirmed" this block is going to be next! Sequencers --> Sequencers: executes public functions Provers --> Provers: generates rollup proofs Provers ->> Contract: submits proofs Contract --> Contract: validates proofs and state transition Note right of Contract: "block confirmed!" end Sequencers ->> Contract: exit() Sequencers --> Sequencers: wait 7 days

Voting on upgrades

In the initial implementation of Aztec, the L1 smart contracts are planned to be fully immutable. However, sequencers are expected vote on upgrades (ie. what should be the next immutable instance of the network) alongside block proposals, in order to reflect social consensus as closely as possible on L2. If sequencers wish to vote alongside an upgrade, they signal by updating their client software &/or an environment configuration variable. If they wish to vote no or abstain, they do nothing. Because the "election" is randomized through the hash ranking algorithm in Fernet, the voting acts as a random sampling throughout the current sequencer set. This implies that the specific duration of the vote must be sufficiently long (and RANDAO sufficiently randomized) to ensure that the sampling is reasonably distributed, eg a majority of sequencers must vote yes by upgrading their client implementation over a minimum duration of 7 days.

More information about the Aztec networkupgrade mechanism is expected to be published alongside it's own RFC within Q1-Q2 of 2024. In this context, it's important to know that sequencers will actively be involved in the upgrade mechanism via voting, and it's intentions to try and mirror L1 social consensus as closely as possible.

sequenceDiagram

participant Contract as Aztec L1 Contract participant Network as Aztec Network participant Sequencers participant Provers

loop Happy Path Block Production Sequencers --> Sequencers: Generate random hashes and rank them Sequencers ->> Contract: Propose block + indicate that they desire to upgrade Note right of Contract: Proposal phase is over! Contract ->> Network: calculates highest ranking proposal + vote Sequencers ->> Provers: negotiates the cost to prove Sequencers ->> Contract: commits to a proverAddress Provers ->> Contract: proverAddress deposits slashable stake Note right of Network: "preconfirmed" this block is going to be next! Sequencers --> Sequencers: executes public functions Provers --> Provers: generates rollup proofs Provers ->> Contract: submits proofs Contract --> Contract: validates proofs and state transition Note right of Contract: "block confirmed! votes counted for upgrade!" end

Backup mode

In the event that no one submits a valid block proposal, we introduce a “backup” mode which enables a first come first serve race to submit the first proof to the L1 smart contracts.

sequenceDiagram

participant Anyone participant Contract as Aztec L1 Contract participant Network as Aztec Network participant Sequencers

loop Happy Path Block Production Sequencers --> Sequencers: Generate random hashes and rank them Contract --> Contract: Waited n L1 blocks.... Proposal phase is over Contract --> Network: calculates highest ranking proposal Note left of Network: No one proposed a block... backup mode enabled! Anyone ->> Contract: submits a rollup... Contract --> Contract: validates proofs and state transition Note right of Contract: "block confirmed!" end

We also introduce a similar backup mode in the event that there is a valid proposal, but no valid prover commitment (deposit) by the end of the prover commitment phase.

sequenceDiagram

participant Anyone participant Contract as Aztec L1 Contract participant Network as Aztec Network participant Sequencers participant Provers

loop Happy Path Block Production Sequencers --> Sequencers: Generate random hashes and rank them Sequencers ->> Contract: Highest ranking sequencers propose blocks Note right of Contract: Proposal phase is over! Contract ->> Network: calculates highest ranking proposal Sequencers ->> Provers: negotiates the cost to prove Contract --> Contract: Waited 5 L1 blocks.... Prover commitment phase is over Note left of Network: No one committed to prove this block... backup mode enabled! Anyone ->> Contract: submits a rollup... Contract --> Contract: validates proofs and state transition Note right of Contract: "block confirmed!" end

Known potential issue: L1 censorship. L1 builders may choose to not allow block proposals to land on the L1 contracts within a sufficient amount of time, triggering “backup” mode - where they could have a block pre-built and proven, waiting L1 submission at their leisure. This scenario requires some careful consideration and modeling. A known and potential mitigation includes a longer proposal phase, with a relatively long upper bounds to submit a proposal - at the cost of slowing down Aztec’s block production.

Future improvements and outstanding questions

Aztec is an incredibly ambitious engineering project, and a number of tradeoffs have been made in the designs articulated above in order to ensure that the network can be iterated on and shipped as quickly as possible. Currently, it’s expected an initial reference implementation should be completed within the next 3-4 months. Due to these tradeoffs, there are a number of “obvious improvements” to the network that will be prioritized based on the completion of other critical network functionality.

1. Should Aztec introduce a consensus network?
2. Generally, the answer here seems to be yes. The question has historically been ROI versus the engineering time/effort spent, as a complex implementation could theoretically delay a network launch.
3. Should Aztec implement a version of MEV or fee burn?
4. Generally, the answer here seems to be yes. It seems like a fee burn model may make more sense than MEV burn, due to the inability to easily quantify MEV on private functions/transactions. Additionally it must be carefully considered how to ensure that the sequencer set doesn’t consolidate on a few sophisticated actors, due to their involvement in the upgrade mechanisms.
5. Should Aztec consolidate the proposal & prover commitment phases? What about other phases?
6. Maybe! This is less clearly understood, at the moment. Reducing L1 interactions is usually nice as it saves some cost

and complexity, however, this could potentially make “MEV-stealing” easier to achieve and warrants further considerations.

7. How long should each phase be? Should they be enforced, or incentivized?
8. The current thinking is that guaranteeing a block gets produced on a regular cadence is a nice to have feature of the network for infrastructure providers, application developers, and end users. However, it may not be pragmatic to enforce, and may make more sense to have a “decaying reward system” that tries to incentivize block production happening as fast or regular as possible. This is an open area of consideration, and is related to item #3

above.

1. How should mev-boost & prover-boost get integrated?
2. MEV-boost is open source and off the shelf, however, Aztec’s environment may be sufficiently different that it warrants a fork or reimplementation of MEV-boost. Prover or proof-boost does not yet have any well known analogues and needs significant design + implementation considerations, which will likely be prioritized later in the year. If you’re reading this and feel strongly about a path forward here, please get in touch.

In the event that other core network functionality is completed sooner than expected, some (or all of these) improvements may be candidates for the initial network release. While these ideas and options are open for improvement, suggestion, debate, etc. they will be considered/prioritized through the lens of engineering feasibility and changes to deliverable timelines.

Disclaimer