Sequencer Selection Proposal: PBS with a federated prover network

Summary

This proposal for sequencer selection uses concepts from MEV Auctions and Sequencer Auctions and PBS to remove the need for a cryptographic sequencer selection process for zkrollup L2s.

Instead, the proposal describes an economic algorithm, administered by an Ethereum smart contract on L1. Communication is facilitated by the gossip network of the L2.

Goals:

1. Separate block building from proving.
2. Ensure that no single actor controls the network.
3. Reduce infrastructure centralisation by incentivising many provers.
4. Facilitate MEV auctions.
5. Ensure that provers know which block to build proofs for, ahead of time.
6. Simple to build and audit.

Details

This proposal assumes the following:

1. Users pay transaction fees on L2 and there is extractable MEV on the public part of their transaction on the L2. The winning block proposer receives the fees and can extract MEV from ordering.
2. A successful proposer pays for the right to extract MEV by burning native tokens.
3. For each block, there is a valid zero-knowledge proof validating all state transitions and transactions. This proof is verified on L1.
4. Provers stake native tokens for the right to prove.
5. There is a block reward paid in native tokens to the winning block and to N uncle blocks.
6. For each proposed block, the proposer decides a percentage of the block reward that will be paid to provers, pro-rata to their contributions in the block.
7. The winning block is determined by ranking proposal on a smart contract with two factors:
8. the amount the proposer is willing to burn.
9. the number of provers who contributed to a block.
10. the amount the proposer is willing to burn.
11. the number of provers who contributed to a block.

Lifecycle of a block

Block Proposal Window

Block proposing, block voting and block commitments occur during the BLOCK_PROPOSAL_WINDOW

. This window lasts M seconds after the last rollup block was confirmed on L1.

Phase 1: Block proposal

A proposer orders pending transactions from the mempool into a block. A proposal consists of:

1. A commitment to the order of transactions in the block (a random secret is used as this will be revealed later).
2. The % of the block reward that the proposer will give to the provers (ratable to their contributions in the block)

3. The amount of native token that the proposer is willing to burn for the block.

A proposal is sent to the L2 gossip network.

Phase 2: Block Voting

Provers (who have previously staked) vote for the blocks that they will construct proofs for.

Economic incentives suggest that they will likely pick based on the most profitable proposals, based on the % of the block reward they will receive.

A vote consists of:

A signature over 1. The block hash 2. The index of the binary proof tree they will make e.g Depth 10:Index 4

Provers broadcast this intent to the L2 gossip network. They will likely broadcast multiple intents, for multiple parts of the block and different blocks.

If a prover does not submit a proof after declaring their intent to do so, they can be slashed.

Phase 3: Aggregation

The block proposer assembles block votes into a block transcript, detailing which provers will create which part of the proof tree from the available votes on the L2 gossip network.

The proposer commits the block transcript to L1 via an Ethereum transaction.

A block will become the head of the chain if BOTH

the following conditions are met:

1. A valid proof is submitted to L1 within the PROOF_ACCEPTANCE_WINDOW

(M seconds after the end of the BLOCK_PROPOSAL_WINDOW

);

AND

1. It has the highest score for the current block height:
2. SCORE(y) = NUM_PROVERS (x)^3 * BURN_BID(z) ^ 2

– needs modelling

1. SCORE(y) = NUM_PROVERS (x)^3 * BURN_BID(z) ^ 2

– needs modelling

As the NUM_PROVERS

is cubed, the proposer will try to create a block transcript with a very diverse prover set, in an effort to maximise their score and win the block reward, fees, and right to extract the MEV.

The proposer could also opt to burn more native token, but this will be less profitable.

Plot of the relationship between provers and burn.

In an effort to ensure the number of provers is greater than a TARGET_PROVERS

, the surface could be updated per block, much like the gas prices / burn inEIP-1559.

Dynamically updating the slope would ensure that the cost for a small set of provers to sustain any censorship attack would increase with each block.

Block Acceptance Window

Phase 4: Proving

Once the block proposal window ends, a proposer will reveal the contents of a block to the L2 gossip network. Provers can look up high ranking blocks on L1 via the smart contract and start building any assigned parts of the proof tree.

Example Proof Tree

[

2948×1338 329 KB

](https://europe1.discourse-cdn.com/business20/uploads/aztec/original/1X/ed831608546c6f2e4bd4de141ce979ad7830ea59.png)

Note: Aztec is targeting a block size of 2^14, or 16,384 txs, so in this example, with each prover computing a 2^3 tree, there would be 2,048 provers.

Completed proofs are submitted back to the L2 gossip network, so they can be built on top of for higher levels of the proof tree. Provers are incentivised to contribute proofs back to ensure that the block is created and they get paid.

The MergeRollup circuit must aggregate provers public keys at each level of the tree by verifiying the prover transcript, and expose this as a public output.

Phase 5: Proof Submission

Proof submission must occur within the PROOF_ACCEPTANCE_WINDOW

Within the window, the proposer (or anyone else) can submit a completed block proof to the rollup contract. Submitting a proof will do the following:

1. Verify the proof
2. Rank the proposal on the rollup contract by its score
3. We will need to store a reference to the messages that need processing for any L1 > L2 messages, and process these once a block is picked.

State Advances

Once the PROOF_ACCEPTANCE_WINDOW

has ended, anyone can finalise the current set of proof submissions and advance the block state to the winning block. This is achieved by calling a function on the L1 contract.

Fees are dispersed as follows:

1. The highest scoring block becomes the head of the chain
2. Fees get paid to the proposer of the winning block
3. The block reward gets paid to the proposer / provers in the agreed split.
4. Provers that contributed to any uncle blocks split an uncle block reward pro-rata.

Diagram

[

Diagram

2048×4418 484 KB

](https://europe1.discourse-cdn.com/business20/uploads/aztec/original/1X/dd2ba6a8bd7f671fdfe2f1f4452d25beda518a2f.png)

Staking

Provers must stake N native tokens to participate in block production. Every prover who contributes to a block gets paid a pro-rata % of the block reward.

Slashing

A prover can be slashed if they commit to producing part of a proof, but don’t.

Slashing occurs on L1, out of the prover’s stake.

The L1 smart contract knows (from the block transcript) who agreed to produce a block. This can be compared to the set of provers in the accepted proof that actually produced the block. This is the basis for a slashing algorithim.

There are two issues here:

What if the proposer tries to slash the prover by not including their proof?

Remedies:

1. Allow a prover to submit their proof directly to L1 within a time window to reverse a malicious slashing.
2. Implement slashing as a rate, so it effects the provers future ability to with-hold a proof.
3. Add a mechanism to incentivise inclusion of proofs
4. e.g if 100 provers attest but the proposer only includes proofs made by 55, the block rewards would be 55% of total rewards.
5. e.g if 100 provers attest but the proposer only includes proofs made by 55, the block rewards would be 55% of total rewards.

What if the prover tries to DDOS the proposer by not creating a proof?

1. The prover will be slashed so this attack is costly.
2. Allow non committed proofs to be included. The reward payout will be lower in this situation, but the block will succeed.

How does a prover know which block to prove?

During the voting and proposing stage, the L2 gossip network will make available a list of the current proposals and the current votes for each by provers.

In the PROOF_ACCEPTANCE_WINDOW

, the provers can read the committed blocks from the L1 Rollup contract, and only prove blocks they think will win.

Redundant prover work

This proposal incentivises redundant prover work. This is a deliberate choice to ensure that the prover network is diverse and that blocks can always be produced.

There are scenarios where it may not be profitable to submit a block for the proposer of a proved block that does not win

.

However, as it is not guaranteed that other proposers will submit a proof for their block in the PROOF_ACCEPTANCE_WINDOW

, it is likely a proposer may gamble and submit.

Other Questions

What stops a proof being taken and used in a different block proposal?

Proofs are signed and committed to in advance. Inside a proof, a signature verification will occur. A nullifier could also be generated here that includes secret information from the committed prover to stop a proof being re-used.

What happens if no proofs are submitted?

If no proofs are submitted in the PROOF_ACCEPTANCE_WINDOW

, the network will restart the BLOCK_PROPOSAL_WINDOW

from the point in time that the first proposal is submitted.

Feasibility

This proposal does not require any new cryptographic primatives and is entirely administered on L1 via a smart contract. This means it is easy to build and audit.

The hardest components to architect are:

1. How does a prover select which parts of the proof tree to vote on. (Likely they will just select all)

2. How does a proposer select provers and do they track prior performance?
3. How does L1 / L2 messaging work.
4. Small changes to the merge rollup circuits, but these are not large changes.
5. As staking occurs on L1, it will likely require a storage proof in each root rollup to prove each prover is a valid prover.
6. Simulations to establish the trade offs of different BLOCK_PROPOSAL_WINDOWS

and PROOF_ACCEPTANCE_WINDOWS

, as well as the number of provers vs burn.

Requirements

Decentralization

:

Sequencer selection must be sufficiently sybil resistant

Sequencer selection should not prioritize the best hardware or largest actors

As this proposal has PBS, it is likely that the actor who can extract the most MEV from the current transaction pool, may be a large actor as MEV simulations are computational intensive. However for this actor to win, they must pay the proving network fairly, and could lead to a more efficient network.

Hardware requirements for sequencers must be similar to those of Ethereum validators

Liveness

:

Network participants must know in advance who the sequencer is for a given time slot

A rollup should be created in every given slot to reduce network latency even in periods of low transaction activity

This proposal accepts a period of network down time during BLOCK_PROPOSAL_WINDOW

to facilitate coordination and liveness. Other proposals that run SSLE protocols will not face this downtime, but come with different trade offs.

To mitigate this, the BLOCK_PROPOSAL_WINDOW

can be on the order of seconds. e.g on Ethereum, MEV Boost is able to propose blocks in sub 15 seconds.

The exact values chosen will need to be simulated.

Downtime can be further mitigated by allowing the proposer to specify a block hash, from which block proposal builds from.

The BLOCK_PROPOSAL_WINDOW

would then start at the same time as the start of the PROOF_ACCEPTANCE

window of the last confirmed rollup block, ensuring zero downtime.

Censorship Resistance:

Ensure the sequencer selection process is censorship resistant

The economic value of winning a block increases with network activity. A dishonest actor must pay more than an honest actor in this proposal to halt the entire network, which is unfeasible to maintain.

However, the cost to censor a single transaction is significantly less.Here a dishonest actor can just forgoe the fees associated with that transaction to censor. Forgoeing these fees every block would add up over time, so this attack is costly, and more of a delay than a full censorship.

Should this be an issue, censorship resistance can be further improved by addingInclude Lists on L1 to allow users to force

a transaction to be included.

Ensure transaction inclusion from a particular sequencer is censorship resistant

Privacy:

Should allow sequencers the option of anonymity during selection and block submission

This proposal will need to be tweaked slightly to afford better privacy to network participants.

Currently:

1. Provers must have public identities to stake / be paid rewards / vote
2. Identities can be setup with fresh accounts
3. Identities can be setup with fresh accounts
4. Proposers need public identities to commit to blocks and get paid. The proposal could be tweaked to allow others to submit completed blocks for privacy.

Other Considerations

Separating ordering and block building from execution

Using L1 versus L2 for the sequencer selection protocol

Proposer-builder separations

The good bits

1. Proposer builder separation reduces censorship risk due to global regulatory differences, and helps with MEV mitigation.
2. Sophisticated actors (e.g., proposers) extract MEV, order transactions and take native token FX risk; non-sophisticated actors just stake and prove.
3. No complicated cryptography; relies on economics and game theory.
4. Easy to implement.
5. Incentivises liveness - no missed slots.
6. No one controls the network for any period of time.

Prior Work

Some of these ideas are based on Ethereum PBS and MEV auction research by Flashbots (e.g., Suave). Thank you to all your work in this space.

dba.mirror.xyz

SUAVE, Anoma, Shared Sequencers, & Super Builders

Thank you to Rajiv Patel-O'Connor, Tarun Chitra, getsqt, Ben Fisch, Evan Forbes, Christopher Goes, and apriori for incredibly valuable insights, feedback, and review of this post.

HackMD

State of research: increasing censorship resistance of transactions under...

State of research: increasing censorship resistance of transactions under proposer/builder separat

Ethereum Research – 9 Dec 19

Spam resistant block creator selection via burn auction

Is this about a consensus algorithm that replaces PoS? I’m curious that what is key difference of concept between this and Proof of Burn (PoB). In PoB, by burning the coin, nodes can prove their contribution/identity to the network, they can get...

DISCLAIMER

The information set out herein is only conceptual and describes Aztec’s future development goals. In particular, the network roadmap is being shared in order to outline some of the plans for Aztec and is provided solely for informational purposes only and does not constitute any binding commitment. Please do not rely on this information for any purpose - the development, release, and timing of any products, features or functionality remains subject to change.