Title

Sequencer Schmequencer

Proposed by

@Mike

Summary

Our proposal draws inspiration from Ouroboros. By using Verifiable Random Functions (VRFs) to rank sequencers, the participant with the lowest VRF output gets designated as the leader for block/rollup production. Other participants function as backups, ensuring network liveness and reliability. This approach fosters fairness, decentralization, and network liveness.

Details

The proposed protocol revolves around the following steps:

1. Time is divided into 'epochs', which are subdivided into 'slots'.
2. Epochs would be 'aligned' with Ethereum blocks, insofar as an epoch will span some n ethereum blocks.
3. A sequencer will be chosen for every slot.
4. This proposal would depend on some canonical token, so that user balances can be measured.
5. No staking is needed.
6. At the end of each epoch (the very last Aztec Rollup of an epoch), user balances are 'snapshotted'. All users with a nonzero balance will be eligible to become a sequencer (although the likelihood will be weighted by the user's balance). Again, no staking.
7. Also at the end of each epoch, a global randomness variable is 'snapshotted'. (This randomness could be derived from historic VRF outputs, or maybe from something like randao).
8. Participants who wish to become sequencers for the epoch in two-epochs' time will execute this sequencer selection protocol.
9. Each sequencer generates a random number (using a VRF – see Appendix) for every time 'slot' in the target epoch.
10. If the random number produced for a particular slot is less than their token balance (as a fraction of the total token supply) they are eligible to produce a rollup in this slot.
11. Multiple sequencers might turn out to be eligible for a slot, so all eligible sequencers must submit (via a zk-snark) their VRF outputs to an L1 'Ranking Contract', which will rank them - lowest VRF wins.
12. Submissions will hide sequencer identities, if we assume the use of a mixnet/Tor for submissions. Or they could submit via Aztec directly (although censorship by other sequencers could be an issue).
13. Sequencers may observe the currently-winning VRF output for a particular slot, and so don't need to submit anything to L1 (to save money) if they can already see they won't win (due to having too-high a VRF output relative to other submissions).
14. All VRF submissions must be made to the Ranking Contract in the epoch after snapshotting. I.e. in epoch e - 2. This ensures users can see other VRF outputs (in order to save money on submitting low-ranking VRFs), and to ensure high-ranking sequencers can prepare for their slot.
15. If no sequencers turn out to be eligible for a slot, we have a fallback to ensure there will always be a sequencer for every slot:
16. The VRF's input will include a fallback_nonce; a counter (starting at 0) which may be incremented to increase the probability of at least someone winning a slot. (With this mechanism, the probability of finding a slot winner is 1, if all users participate).
17. Of the VRF outputs submitted to the ranking contract, the lowest valid VRF with the lowest fallback_nonce will win.
18. To safeguard against the winning Sequencer not actually producing a rollup in time, any of the highest-ranking x sequencers (actual number tbd) may submit L2 blocks during the slot. This results in a small amount of redundant compute, with the benefit of greatly reducing the likelihood of missed slots.

19. This will cause some noise in the proof pool, but it wouldn't be too bad. Provers' proofs can contain an identifier of the particular L2 block they're building for. And if proof pool activity is visible to all, a 'more popular' L2 block might emerge early in a slot, at which time all provers will likely swarm around the favourite. The prover protocol can be tailored towards this sequencer selection protocol.
20. Of course, a Sequencer might not need help, or might not communicate with provers via a public pool, in which case there will be a small amount of redundant roll up proving.
21. Of course, a Sequencer might not need help, or might not communicate with provers via a public pool, in which case there will be a small amount of redundant roll up proving.
22. Whichever of the top x sequencers is first to be included on L1, will be the canonical L2 block for that slot.
23. (Take this bullet point with a pinch of salt… it's a bit of a yolo suggestion) There's still a chance none of the x sequencers submit L2 blocks during their slot. To protect against this ever further, in the very final L1 block of a slot (assuming a slot will span many L1 blocks), any of the eligible sequencers (who successfully submitted VRF outputs during epoch e - 2 — ie not just the 'top x') may submit L2 blocks in one big free-for-all.
24. We might wish to burn some of the canonical token, each L2 block, for reasons similar to ethereum, but that's separate.

Note: It might be valuable to enable users to delegate their balance to a sequencer delegate, too.

A pretty diagram

Timeline

t-3 - | epoch | e-3 | | t-2 - <- Snapshot of state root at the end of epoch e-3. | ^ Used to prove balance in VRF. | | epoch | | e-2 | | Proofs of VRF eligibility for a particular slot of epoch e | | MUST be submitted (to the Ranking Contract) during this epoch. | | After this epoch, rankings are fixed. | v t-1. - | | | | t - | ^ | | For a given slot, a rollup may be submitted by whomever is in the top x of the epoch | | Ranking Contract's rankings for that slot. e | | | | | v t+1 -

Psuedocode

my_balance_proportion[epoch_num - 2] = my_balance[epoch_num - 2] / total_supply[epoch_num - 2];

let results = []; let fallback_nonce = 0; // initially set to 0, and increment as a fallback, if no sequencer found.

// Iterate over each slot number: for (let i = 0; i < num_slots[epoch_num]; ++i) { const my_test_vrf = hash(private_key, i, fallback_nonce, rand[epoch_num], // needed to prevent someone cleverly selecting a nice private key ahead of time. "TEST");

```
if ( my_balance_proportion[epoch_num - 2] < my_test_vrf / field_modulus ) {

    // Congrats! You're a sequencer for this epoch_num, slot i !

    // Then this is needed to feed future randomness.
    // my_test_vrf was too biased towards small values to be usable for future randomness.
    const my_randomness_contribution = hash(
        private_key,
        i,
        fallback_nonce
        rand[epoch_num],
        "RAND"
    );

    results.push({
        epoch_num,
        slot_num: i,
        fallback_nonce,
        my_test_vrf,
        my_randomness_contribution,
    })
}
```

}

Comparisons

This protocol combines the best features from a few blockchain protocols and other proposals, ensuring a fair, decentralized selection process with minimized bias. Unlike proposals favoring superior hardware or sophisticated search algorithms, this approach encourages sequencer diversity. When compared to Joe's PBS proposal that presents a market-like ecosystem, this proposal is simpler, doesn't require token lock-up, and more closely aligns with Whisk-y or Cookie Jar proposals in terms of prioritizing inability to centralize around a few sophisticated MEV searchers. It is worth considering what starting with a simple proposal like this and migrating to a more complicated, and perhaps privacy preserving solution, overtime would look like.

Feasibility

The proposal's simplicity and its reliance on established technologies make implementation within 4-6 months feasible. However, certain aspects like the specifics of unbiased VRF generation, generation frequency, and token holding requirements need more research and exploration.

Questions

1. How will potential VRF bias be mitigated to maintain fairness in the sequencer selection process?

2. What factors will determine the token holding requirements for sequencer selection eligibility?

2.a. No restrictions. Any balance is eligible. (Although tiny balances are unlikely to 'win').

1. Can the proposed protocol be applied on Layer 2 (L2) without posing significant censorship concerns?

Note: the L1 Ranking contract could instead be done in an Aztec public L2 contract instead, if there were a way to combat censorship. This is a current area of investigation on discourse.

1. How will backup sequencers be selected and incentivized to maintain network reliability?

4.a. They'll stick around for the chance they might win, and because they might be higher-ranking in future blocks.

1. How will Aztec Tokens being burned upon processing a rollup impact network economy and functionality?