Decentralized Sequencing Proposal: Espresso Martini

Proposed by: Espresso Systems team

Summary

This proposal is an extension to any base decentralized sequencing design (e.g., total anarchy, Whisk-y, Cookie Jar, Irish Coffee, etc.) that preserves the same security properties but adds pre-confirmations to improve the user experience. More precisely, clients who trust that less than a threshold of the sequencer staking set is corrupt (e.g., 33%) can obtain fast pre-confirmations of transaction success long before the L1 rollup contract is updated. On the other hand, clients who do not wish to make additional trust assumptions can simply wait on the L1 contract for finality. Importantly, no matter what fraction of the sequencer staking set is corrupt, the safety and liveness of the rollup reduces to that of the base decentralized sequencing design.

Motivation

Users who only trust Ethereum for finality need to wait around 15 minutes, which cannot be improved unless Ethereum's consensus protocol is modified (e.g., SSF is still in a research phase and is not expected to ship for several years). Moreover, since rollup blocks are posted every 10 minutes, users who submit early in this window may have to wait up to 25 minutes for finality.

Note that 10 minutes is an upper bound on the rollup block time. Proposals like Whisk-y and PBS with a federated prover network could enable an interval of 1-2 minutes, for finality of 15-20 minutes.

Decentralized sequencing designs in which a random beacon and/or SSLE protocol is used to elect a sequencer/prover node once every 10 minutes result in a poor user experience. Namely, users have to wait 15-25 minutes for confirmation that their transaction will be included in the rollup. This stands in contrast to a reputable trusted centralized sequencer

(e.g., run by Aztec itself), which can provide users who trust the centralized sequencer

with instant confirmation of not only transaction inclusion but also execution results long before posting the rollup block to the Ethereum smart contract. Ironically, it is much harder for users to trust individual nodes elected by a decentralized sequencing protocol than a centralized sequencer. On the other hand, it is reasonable for users to believe that less than 33% of the entire set of nodes are corrupt, without knowing which specific

subset of sequencer/prover nodes they can trust.

The goal of this proposal is to achieve a decentralized sequencing design whereby users who believe that at least 67% of the staking set is uncorrupted

can obtain almost-instant pre-confirmations of rollup transactions. For many users, this assumption may be even easier to believe than trusting an Aztec centralized sequencer. Moreover, with a sufficiently scalable consensus protocol and the advent of restaking contracts pioneered by EigenLayer, this sequencer staking set could ultimately be identical to Ethereum's validator staking set. Furthermore, our proposed design is compatible with base decentralized sequencing designs such as Whisk-y, Cookie Jar, Irish Coffee. This design will also respect two key constraints:

1. The Ethereum (L1) contract provides ultimate finality on rollup transactions. Users who do not want to trust the staking set can thus wait for confirmation from Ethereum.
2. Even if the entire sequencing staking set becomes corrupt, the rollup's safety and liveness reduces to that of the base decentralized sequencing design (i.e., will eventually make progress if the L1 contract is functional and there exists a prover willing to submit a rollup block).

Design sketch

- Let T denote a timeout parameter, e.g. T = 200 minutes. It is assumed that the rollup proof system is configured to produce proofs for a batch of transactions (with continuous arrival times) every 10 minutes. This can also be left as a parameter.
- Let R denote a recovery window time parameter, which starts at R = 10 minutes.
- Let DS denote the base decentralized sequencing system (e.g., a random beacon is used to elect a prover for a 10 minute period). DS may or may not have an associated staking set (e.g., total anarchy does not).
- Let CP denote a responsive consensus protocol that has fast finality under best-case conditions but is partition tolerant in worst-case conditions. Examples include HotStuff and its enhancements HotStuff-2 and HotShot, which are being developed and productionized by Espresso Systems. CP will be run by a sequencer staking set and can tolerate no more than 33% corruption of the stake.

- The L1 contract has a timer that gets reset every time it receives a valid rollup block updating the state root of the rollup together with a valid proof.
- Under normal circumstances, users submit their rollup transactions to the protocol CP to be ordered and executed. CP filters for bad transactions that cannot be proven in a rollup block (e.g., for ZK transactions they must be correctly formatted and have a valid proof). CP also provides availability for the data required to build a rollup block from these transactions. The DS protocol is used to determine a sequencer/prover node that ultimately builds a rollup block out of these transactions and submits it in the next L1 update.

For CP to execute transactions it must be stateful. An alternative is to have a lazy CP that does not execute and only orders. It may still implement Aztec-specific transaction filtering logic. Ideally the proof system would not require any filtering, but this may be a practical requirement today.

- If T time has not yet elapsed (since the last update), then the L1 contract will only accept valid rollup blocks extending the last state root with a transaction list that has been finalized by CP. To do so, the L1 contract has a light-client verifier for the state of CP (typically just requires threshold signature verification, but can also be done with snarks). The rollup block contains both a proof and data used to verify the proof, which implicitly includes a cryptographic commitment to a list of transactions that the L1 contract can verify as identical to those finalized by CP.
- If T time has elapsed and no new rollup block has been received then the L1 contract enters a recovery phase for a period of R time. During this time the L1 contract will suspend the requirement that rollup blocks need to be authorized by CP. The behavior of the system during this period is thus identical to running DS on its own.
- Following the recovery phase, the L1 contract reinstates its requirement that rollup blocks are authorized by CP. However, with each consecutive occurrence of a T-length timeout the parameter R is doubled.

Protocol analysis

Under normal circumstances a new valid rollup block is received by the L1 contract at intervals of length at most T. These blocks are submitted by DS but also authorized by CP. It thus inherits the sequencer/prover decentralization of DS. It also remains consistent with pre-confirmations given by CP to users (under the assumption that CP is safe). During a recovery phase the system behavior is identical to running DS without pre-confirmations from CP. The recovery phase may be entered, i.e. T time has elapsed but no new rollup block has been submitted, for the following reasons:

- Provers collude to circumvent CP:

It is possible that the staking set is not actually corrupt, but provers collude to circumvent CP. Making T large (significantly greater than 10 minutes) ensures that many provers need to collude to do so and a single prover cannot. Furthermore, because the system returns to CP after each recovery phase, provers would need to collude for several consecutive phases (doubling R each time until it gets very large) and would also need to sustain this attack in order to entirely circumvent CP.

- CP experiences a temporary liveness failure:

If CP experiences a temporary liveness failure then the recovery phase allows for the rollup to progress using only DS as a fallback mechanism, before reverting to normal operation. Exponential backoff on the duration of R is a natural way to handle such temporary failures.

- CP is corrupt:

If the CP staking set is adversarial then it may attempt to halt the rollup, such as by withholding data necessary for proving. If this behavior is detectable/accountable then there may exist a governance mechanism to fully remove CP. Even in the absence of such a governance mechanism the exponential backoff on the duration of R means that, in the long run, the protocol only loses liveness for relatively brief periods of time.

User flow

The following is an example user flow under the Espresso Martini proposal:

1. A user creates an Aztec transaction, optionally including a fee. There are many ways to design the fees (compatible with any TFM), and fees could even be negative to return potential MEV to the user.

Suppose the DS already has a TFM and a way of paying fees. Then with Espresso Martini, one option is to have users pay fees as usual through Aztec VM using the given TFM, but also pay a percentage to the CP staking set. Another option is for the DS sequencer/prover to act as a relayer that covers the CP fee for users.

1. The client submits the signed transaction to CP.
2. The user receives a pre-confirmation from CP. Under optimistic conditions, pre-confirmations are almost instant. Clients may optionally choose to act on these pre-confirmations and display the transaction as finalized.

3. Optionally, clients may wait the full time period for the transaction to be proven as part of a rollup and posted to the L1.

HotShot

One option for CP is the Espresso Sequencer, which implements HotShot, an optimistically responsive protocol with linear communication that can confirm transactions as fast as the network will allow. With optimistic responsiveness, the network can leverage highly scalable "Web2" architectures (e.g., high bandwidth nodes operating NGINX instances) to optimistically achieve extremely high throughput, while making no compromises on safety. HotShot guarantees that even if this architecture goes down, liveness is sustained over a high-resilience gossip-based fallback path. This achieves the best of both worlds: Web2 performance with Web3 security.

HotShot is designed to achieve similar performance to a centralized sequencer, while simultaneously being able to support the entire Ethereum validator set. Our initial testnet provides promising results, reaching up to 20,000 ERC-20 txns/s, though note that this is prior to the integration of many techniques that further improve throughput. The protocol remains secure and live even in adversarial networking conditions, which makes it more resilient and trustworthy than a centralized sequencer. Espresso also incorporates data availability using verifiable information dispersal techniques, such as those described in Succinct Erasure Coding Proof Systems, adapted to HotShot with linear communication for highly scalable and fast data availability.

HotShot improves upon HotStuff and Hotstuff-2. It is a leader-based protocol, which means that in each round a random leader is elected to propose a new block of transactions. A supermajority of nodes then confirms the block by signing off on it. The random leader election utilizes an unbiasable VDF-based random beacon. Protocols in the Hotstuff family have optimistically linear communication (unlike PBFT) and responsiveness (unlike Tendermint).

Restaking

The introduction of restaking to Ethereum means that the Espresso Sequencer can be run by the same validator set as Ethereum. This is compelling for a number of reasons:

1. The sequencer is reliant on the same set of nodes as Ethereum for security (consistency/liveness). Only the protocol changes, and with it the properties achieved (e.g., finality vs availability). In fact, Ethereum already effectively runs two protocols, Casper FFG and LMD Ghost, which work in concert to maintain a dynamically available chain with a finalized prefix. Engaging the Ethereum validators in running the Espresso Sequencer thus modularly adds to the components of Ethereum itself in order to better support the rollup ecosystem without changing the security model. Otherwise, users would need to trust a new set of validators for both the liveness of the rollup ecosystem and for fast finality.

2. If none (or relatively little) of the value generated by a rollup is shared with the L1 validators, then this risks destabilizing the security of the rollup. For example, the L1 validators could be bribed to fork the rollup smart contract state for greater profit than they would earn managing the rollup contract honestly.

Proposer-builder separation

By design, the Espresso Sequencer is compatible with both proposer-builder separation and protocol-owned builders. This allows projects to decide how they wish to approach MEV-related activity within their rollup. Moreover, PBS further facilitates interoperability and atomic composability of transactions across all rollups that are using the Espresso Sequencer.

Requirements

The following requirements are satisfied as long as the base decentralized sequencing design satisfies them.

Decentralization

Sequencer selection must be sufficiently Sybil resistant

Sequencer selection should not prioritize the best hardware or largest actors

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

Censorship Resistance

Ensure the sequencer selection process is censorship resistant

Ensure transaction inclusion from a particular sequencer is censorship resistant

Privacy

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

Further reading

- The Espresso Sequencer
- Shared Sequencing: Defragmenting the Rollup Ecosystem