Authors: Stanislav Vozarik, Gleb Urvanov, Shoeb Siddiqui, Peter Kris,

We present a master-rollup architecture that is designed to facilitate cross-L1 information transfers. This architecture solves problems introduced with bidirectional cross-L1 communication, such as problems related with bridges, block finality, and the correctness of information ingestion. The architecture leverages security guarantees of the Eigenlayer.

Existing Solutions

	Decentralized	Efficient	Secure	Multichain
BRIDGE	questionable	yes	questionable	no
ROLLUP	questionable	yes	yes	no
MASTER-ROLLUP	yes	yes	yes	yes

Bridges

– While they operate directly on blockchains itself, thus to some degree decentralized, the decentralization itself is questionable for bridges, due to often employing various (albeit sophisticated) multi-signature mechanisms. We cannot really evaluate the efficiency of a bridge itself, but the presumption is that the secondary chain has lower traffic, transaction costs and thus higher available bandwidth. Bridge security relies on the security assumptions of the secondary chain, which is usually much lower than the primary chain. This difference in security levels might lead to unauthorized bridge withdrawals, rendering bridged assets invaluable. Since bridges operate as pair connections, securing multichain connections require deployment of multiple bridge instances and jumps.

Rollups

- Are a new addition to the ecosystem which leverage the security of Ethereum or other L1 networks. They do not rely on the inferior security of secondary chains, but decentralization is often questionable. This is because sequencers – the communication actors / layers between chains - are usually centralized in a single entity or a small group. A major advantage of rollups is efficiency, they offer cost savings or bandwidth of several thousandfold compared to the main chain. Similar to bridges, rollups also work in pairs, requiring multiple instances and jumps for communication between different chains.

Proposition

Master-Rollup

- A rollup that's simultaneously connected to multiple L1s and using Eigenlayer as settlement layer to coordinate communication between respective L1s. Efficiency and security are inherited from rollup architecture itself. However, enhancing decentralization and upgrading to multichain comes with another set of challenges which we are actively working to solve.

Problem 1: L1 data ingestion correctness

Single chain rollup, problem not present

In every cross-chain communication, there are actors that transport information from one chain to another, as the chains themselves do not really see each other in a “blockchain” way. The final security is derived from the security of Ethereum L1. In the case of a wrong or malicious read from L1, the rollup contract can verify its previous state and requests towards L2, preventing an update including this incorrect read. L1 stays secured.

[

1142×860 63.5 KB

](https://global.discourse-cdn.com/standard14/uploads/eigenlayer/original/2X/e/e68f12079619235dfc6500a8295a05c18e57da50.png)

In the example above, a malicious sequencer could fake a user depositing value. After some L2 blocks, the final state is brought back to L1. The contract on L1 will check whether L2 operations had a valid starting point, comparing it to its previous state. In this case, L1 knows that user1 did not deposit any value, thus withdrawal of it (or the rest after L2 operations) is not possible. Of course, this would come as well with the necessity to rollback any L2 blocks upwards the incorrect starting point – reversing the malicious/incorrect read.

For funds that only exit the way they entered and where this can be checked by L1 contract, having one or a set of trusted sequencers to read from L1 is sufficient. However, untrusted sequencers could spam the network with incorrect information and lead to constant rollbacks. Due to the inability of anyone or everyone to participate in this sequencing, we approach decentralization inadequacy.

Master-rollup problem

Since master-rollup funds can enter from one L1 chain and exit via another, there is no way for other L1 to back check whether this entry was correct.

[

1142×1251 99.1 KB

](https://global.discourse-cdn.com/standard14/uploads/eigenlayer/original/2X/a/ac7637353c2ee1ed758cb85e888e57649f9a1618.png)

This makes it crucial to confirm that the original read from L1 is correct before using it on L2, instead of just relying on a trusted set of sequencers.

Solution: RollDown (ensuring decentralization).

To ensure that updates between L1 and L2 are correct, both layers need to be treated the same way; just as L2 rolls up to L1, L1 needs to roll down to L2. In master-rollup we decided to address the rollup solution with a zero-knowledge approach. For everything happening on L2, there will be zero-knowledge proof calculated and provided to L1 for verification. This ensures that L1 knows that the state on L2 is correct. The master-rollup architecture also needs to do the reverse: assure L2 that the state provided by L1 is accurate However, computing ZK proof due to computational intensity and related costs is not practically feasible.

Optimistic approach

Similarly to optimistic rollups, updates from L1 to L2 are not automatically accepted, but are waiting to be ingested by L2 in a queue for a dispute period. During this period, any other sequencer can simply cancel this read as false, as they are also aware of the L1 state. Dispute is later brought back to L1, which decided which party is correct and slashes or rewards accordingly.

[

A diagram of a flowchart Description automatically generated

1400×771 91.9 KB

](https://global.discourse-cdn.com/standard14/uploads/eigenlayer/original/2X/5/5a8f2ec76bad678a89635acb5c6e0c6fe2ba6424.jpeg)

Caveats:

- Sequencers must maintain a stake on L1.
- Sequencers will have their L1 stake slashed if they are caught posting incorrect reads or unnecessary cancellations to the rollup.
- Each sequencer is granted one read right and n cancellation rights per time frame, where n equals the number of the other sequencers in the active set.

Even if all but one sequencer acts maliciously, a single honest sequencer has the power to cancel all incorrect reads. Eventually, the honest sequencer can post proof of misbehavior to L1. The result would be that all misbehaving sequencers gets slashed while the honest ones receive a reward.

This creates a security of a correct read to a level “at least one fair actor” instead of “at least 51%”, which is usually on blockchains and creates a vulnerability in bridge solutions. It also allows for anyone willing to stake funds to become a sequencer, ensuring that the set of sequencers remains decentralized.

Problem 2: multi L1 communication and state awareness

Master-rollup, being an app-specific rollup, serves as an L2 communication hub between different L1s. Usually rollups need to mirror the whole business logic of L2 on L1, for L1 to be able to “replay” operations happening on L2. This becomes a problem of linear complexity when connecting other L1 to master-rollup.

Any operation on a rollup is divided into 3 parts:

-Requests can originate on L1 and be transferred to L2, in our case with roll-down mechanics explained earlier, or they can originate directly on L2.

-Updates are beyond the scope of this article. Master-rollup uses zero-knowledge proofs to ensure that operations on both L2 and L1 can be securely updated. As each L1 chain connected to master-rollup might operate differently, the problem is the scope of an update. Its relevancy and even possibility. This problem starts at the execution phase.

-Execution. Different chains might have different functionalities. Thus it might not be possible to update one L1 with an end state resulting from an operation logic present on another L1. A common functionality understood by all is the transfer. Instead of mirroring the whole business logic on L2 for each connected L1, every operation can be decomposed or translated to a series of transfers relevant to each L1, as the end state of an operation.

[

1329×859 53.9 KB

](https://global.discourse-cdn.com/standard14/uploads/eigenlayer/original/2X/7/74d06559d7a6019881f7e7d74c4c9c2405192150.png)

Example: simple swap on a DEX is internally just a series of transfers between a user account and an account holding pool liquidity. Master-rollup marks these transfers by their relevancy, to be a part of operations resulting in a new state, which is part of a state update for L1. In the case of a swap, the relevancy is determined by the asset’s origin. Such transfers are easily zero-knowledge provable, and the end state, along with the asset ownership gets updated on each L1. This produces a specific case of atomic swap. The same principle can be used for a plethora of different information transfer use cases.

Problem 3: correctness execution phase, correctness of L2 state itself

Rollups solve some of the security issues with bridges, as the “bridged” assets waiting on L1 side cannot be withdrawn due to same security shortcomings. However, most L2’s still suffer smaller scale challenges compared to L1. Low number of nodes means lower costs for a 51% attack. While this presents challenges in many areas, there is an additional one in master-rollup architecture. If an operation originates and finalizes on the same L1, as mentioned previously, L1 can check the whole history of the result, up to the initial request. The situation changes if an operation originates on L2, or on a different L1 chain. In this case, we need to ensure all intermediate L2 operations are correct and finalized…

All operations in the execution phase are included in blocks that are sooner or later finalized. Depending on how and when blocks are finalized, we have different kinds of finality. Probabilistic finality does not provide immediate result, and economic finality suffers from smaller scale and lower cost of becoming majority voter, which is usually the case for most L2. Due to the nature of master rollup, which allows withdrawals to different L1 chains, immediate and deterministic finality is required. It is not enough to prove operations on L2 are valid, like zero knowledge proofs are capable, it must be proven they were irreversibly included in blockchain.

[

1142×577 33.9 KB

](https://global.discourse-cdn.com/standard14/uploads/eigenlayer/original/2X/d/dbbb257cb56a4e3382e4095b199e6df2b8ad736e.png)

Deterministic finality for master-rollup is achieved by connection to re-execution and finalization chain. Execution phase is completed in L2's node runtime in a predetermined and immutable manner. Before block finalization, every operation is re-executed and verified by this security chain, with the same deterministic rules. Block finality is guaranteed by signatures of stakers participating in block approval. To prove the block is finalized it is enough to verify these signatures. Onchain signature verification is a computationally intensive process. Instead of verifying these signatures on L1 directly, the security chain computes zero-knowledge proof of correctness of the signature verification process as proof of finality. This deterministic finality of master-rollup blocks and its proof is incorporated in all L2 to L1 state updates, providing guarantee of L2 correctness and finality. Eigenlayer is the most promising candidate for the re-execution chain.

Other practical considerations and applications:

L1-L2 communication speedup

Since information transfer from L1 to L2 is covered by a decentralized set of sequencers and mechanism to ensure correctness is based on a dispute period, there is a considerable delay between publishing request on L1 and secure ingestion on L2. This brings an opportunity for actors with access to both L1 and L2 to relay this information faster. They can do this by staking their own funds on L2, proportional to the size of the transferred information and its correctness… For example, in a regular deposit case without speedup, the deposit has to wait for the dispute period to conclude, so L2 can be certain of the deposit's legitimacy from the L1 side. Only after this period, L2 can be sure and mint this deposit on L2 user account. Any actor with funds of the type and amount of this deposit, can release these funds to the depositor immediately and be reimbursed with later minted tokens to his account instead of depositors. If the deposit is rendered invalid during the dispute period, no tokens will be minted and transferred funds are lost for the speed up service provider. These actors "ferry" the information across the chain gap at their own risk for a fee. We refer to them as Ferries.

Escape Hatch

An escape hatch is a necessary component in any rollup, to allow users to withdraw funds if L2 loses liveliness. In master-rollup infrastructure escape hatches cannot be always active, since the withdrawn value could be moved to another L1 if the chain hasn't actually lost its liveliness.

Uniswap X

The Master-rollup structure has the potential to enable cross-chain liquidity for native pairs that might serve well for cross-chain settlement of protocols like UniswapX or others.