Rollup Circuits

The primary purpose of the Rollup Circuits is to 'squish' all of the many thousands of transactions in a rollup into a single SNARK, which can then be efficiently and verified on Ethereum.

These circuits are executed by a Sequencer, since their primary role is to order transactions. A Sequencer might choose to delegate proof generation to the Prover pool.

The way we 'squish' all this data is in a 'binary tree of proofs' topology.

Example: If there were 16 txs in a rollup, we'd arrange the 16 kernel proofs into 8 pairs and merge each pair into a single proof (using zk-snark recursion techniques), resulting in 8 output proofs. We'd then arrange those 8 proofs into pairs and again merge each pair into a single proof, resulting in 4 output proofs. And so on until we'd be left with a single proof, which represents the correctness of the original 16 txs. This 'binary tree of proofs' topology allows proof generation to be greatly parallelized across prover instances. Each layer of the tree can be computed in parallel. Or alternatively, subtrees can be coordinated to be computed in parallel. Note: 'binary tree of proofs' is actually an over simplification. The Rollup Circuits are designed so that a Sequencer can actually deviate from a neat, symmetrical tree, for the purposes of efficiency, and instead sometimes create wonky trees. Some of the Rollup Circuits also do some protocol checks and computations, for efficiency reasons. We might rearrange which circuit does what computation, as we discover opportunities for efficiency.

Base Rollup Circuit

The Base Rollup Circuit takes two kernel circuit proofs (from two distinct transactions) and compresses them into a single snark.

For both transactions, it:

- Performs public state read membership checks.
- Updates the public data tree in line with the requested state transitions.
- Checks that the nullifiers haven't previously been inserted into the[indexed nullifier tree]
- .
- Batch-inserts new nullifiers into the nullifier tree.
- Batch-inserts new note hashes into the note hash tree
- Batch-inserts any new contract deployments into the contract tree.
- Hashes all the new nullifiers, commitments, public state transitions, and new contract deployments, to prevent exponential growth in public inputs with each later layer of recursion.
- Verifies the input kernel proof.

Merge Rollup Circuit

The Merge Rollup Circuit takes two Base or Merge circuit proofs and compresses them into a single snark.

It:

- Hashes the two input hashes into a single output hash.
- Verifies the two input base or merge proofs.

Root Rollup Circuit

The Root Rollup Circuit takes two Base or Merge circuit proofs and compresses them into a single snark.

It:

- Hashes the two input hashes into a single output hash.
- Verifies the two input base or merge proofs.Edit this page

Previous Public Kernel Circuit Next Sequencer