Sequencer

The sequencer is a module responsible for creating and publishing new rollup blocks. This involves fetching txs from the P2P pool, ordering them, executing any public functions, running them through the rollup circuits, assembling the L2 block, and posting it to the L1 rollup contract along with any contract deployment public data.

On every new block assembled, it modifies the world state database to reflect the txs processed, but these changes are only committed once the world state synchronizer sees the new block on L1.

Components

Theblock builder is responsible for assembling an L2 block out of a set of processed transactions (we say a tx has been processed if all its function calls have been executed). This involves running the txs through the base, merge, and rollup circuits, updating the world state trees, and building the L2 block object.

Theprover generates proofs for every circuit used. For the time being, no proofs are being actually generated, so the only implementation is an empty one.

Thepublisher deals with sending L1 transactions to the rollup and contract deployment emitter contracts. It is responsible for assembling the Ethereum tx, choosing reasonable gas settings, and monitoring the tx until it gets mined. Note that the current implementation does not handle unstable network conditions (gas price spikes, reorgs, etc).

Thepublic processor executes any public function calls in the transactions. Unlike private function calls, which are resolved in the client, public functions require access to the latest data trees, so they are executed by the sequencer, much like in any non-private L2.

Thesimulator is an interface to the wasm implementations of the circuits used by the sequencer.

Thesequencer pulls txs from the P2P pool, orchestrates all the components above to assemble and publish a block, and updates the world state database.

Circuits

What circuits does the sequencer depend on?

Thepublic circuit is responsible for proving the execution of Brillig (public function bytecode). At the moment, we are using a fake version that actually runs ACIR (intermediate representation for private functions) and does not emit any proofs.

Thepublic kernel circuit then validates the output of the public circuit, and outputs a set of changes to the world state in the same format as the private kernel circuit, meaning we get a standard representation for all txs, regardless of whether public or private functions (or both) were run. The kernel circuits are run iteratively for every recursive call in the transaction.

Thebase rollup circuit aggregates the changes from two txs (more precisely, the outputs from their kernel circuits once all call stacks are emptied) into a single output.

Themerge rollup circuit aggregates two outputs from base rollup circuits into a single one. This circuit is executed recursively until only two outputs are left. This setup means that an L2 block needs to contain always a power-of-two number of txs; if there are not enough, then empty txs are added.

Theroot rollup circuit consumes two outputs from base or merge rollups and outputs the data to assemble an L2 block. The L1 rollup contract then verifies the proof from this circuit, which implies that all txs included in it were correct.

Source code

You can view the current implementation on Github here .

Further Reading

- Sequencer Selection Edit this page

Previous Rollup Circuits Next Sequencer Selection