Circuits

Central to Aztec's operations are 'circuits' derived both from the core protocol and the developer-written Aztec.nr contracts.

The core circuits enhance privacy by adding additional security checks and preserving transaction details - a characteristic Ethereum lacks.

On this page, you'll learn a bit more about these circuits and their integral role in promoting secure and efficient transactions within Aztec's privacy-centric framework.

Motivation

In Aztec, circuits come from two sources:

1. Core protocol circuits
2. User-written circuits (written as Aztec.nr Contracts and deployed to the network)

This page focusses on the core protocol circuits. These circuits check that the rules of the protocol are being adhered to.

When a function in an Ethereum smart contract is executed, the EVM performs checks to ensure that Ethereum's transaction rules are being adhered-to correctly. Stuff like:

- "Does this tx have a valid signature?"
- "Does this contract address contain deployed code?
- "Does this function exist in the requested contract?"
- "Is this function allowed to call this function?"
- "How much gas has been paid, and how much is left?"
- "Is this contract allowed to read/update this state variable?"
- "Perform the state read / state write"
- "Execute these opcodes"

All of these checks have a computational cost, for which users are charged gas.

Many existing L2s move this logic off-chain, as a way of saving their users gas costs, and as a way of increasing tx throughput.

zk-Rollups, in particular, move these checks off-chain by encoding them in zk-S(N/T)ARK circuits. Rather than paying a committee of Ethereum validators to perform the above kinds of checks, L2 users instead pay a sequencer to execute these checks via the circuit(s) which encode them. The sequencer can then generate a zero-knowledge proof of having executed the circuit(s) correctly, which they can send to a rollup contract on Ethereum. The Ethereum validators then verify this zk-S(N/T)ARK. It often turns out to be much cheaper for users to pay the sequencer to do this, than to execute a smart contract on Ethereum directly.

But there's a problem.

Ethereum (and the EVM) doesn't have a notion of privacy.

- There is no notion of a private state variable in the EVM.
- There is no notion of a private function in the EVM.

So users cannot keep private state variables' values private from Ethereum validators, nor from existing (non-private) L2 sequencers. Nor can users keep the details of which function they've executed private from validators or sequencers.

How does Aztec add privacy?

Well, we just encodeextra checks in our zk-Rollup's zk-SNARK circuits! These extra checks introduce the notions of private state and private functions, and enforce privacy-preserving constraints on every transaction being sent to the network.

In other words, since neither the EVM nor other rollups have rules for how to preserve privacy, we've written a new rollup which introduces such rules, and we've written circuits to enforce those rules!

What kind of extra rules / checks does a rollup need, to enforce notions of private states and private functions? Stuff like:

- "Perform state reads and writes using new tree structures which prevent tx linkability" (see trees
-).
- "Hide which function was just executed, by wrapping it in a zk-snark"
- "Hide all functions which were executed as part of this tx's stack trace, by wrapping the whole tx in a zk-snark"

Aztec core protocol circuits

So what kinds of core protocol circuits does Aztec have?

Kernel Circuits

- Private Kernel Circuit
- Public Kernel Circuit

Rollup Circuits

- Rollup Circuits

Squisher Circuits

We haven't fully spec'ed these out, as Honk and Goblin Plonk schemes are still being improved! But we'll need some extra circuit(s) to squish a Honk proof (as produced by the Root Rollup Circuit) into a Standard Plonk or Fflonk proof, for cheap verification on Ethereum. Edit this page

Previous ACIR Simulator Next Private Kernel Circuit