Transactions

On this page you'll learn:

- The step-by-step process of sending a transaction on Aztec
- The role of components like PXE, Aztec Node, ACIR simulator, and the sequencer
- The Aztec Kernel and its two circuits: private and public, and how they execute function calls
- The call stacks for private & public functions and how they determine a transaction's completion

Simple Example of the (Private) Transaction Lifecycle

The transaction lifecycle for an Aztec transaction is fundamentally different from the lifecycle of an Ethereum transaction.

The introduction of the Private eXecution Environment (PXE) provides a safe environment for the execution of sensitive operations, ensuring that decrypted data are not accessible to unauthorized applications. However, the PXE exists client-side on user devices, which creates a different model for imagining what the lifecycle of a typical transaction might look like. The existence of a sequencing network also introduces some key differences between the Aztec transaction model and the transaction model used for other networks.

The accompanying diagram illustrates the flow of interactions between a user, their wallet, the PXE, the node operators (sequencers / provers), and the L1 chain.

1. The user initiates a transaction
2. – In this example, the user decides to privately send 10 DAI to gudcause.eth. After inputting the amount and the receiving address, the user clicks the confirmation button on their wallet.

The transaction has not been broadcasted to the sequencer network yet. For now, the transaction exists solely within the context of the PXE.

1. The PXE executes transfer locally
2. – The PXE, running locally on the user's device, executes the transfer method on the DAI token contract on Aztec and computes the state difference based on the user’s intention.

The transaction has still not been broadcasted to the sequencer network yet and continues to live solely within the context of the PXE.

1. The PXE proves correct execution
2. – At this point, the PXE proves correct execution (via zero-knowledge proofs) of the authorization and of the private transfer method. Once the proofs have been generated, the PXE sends the proofs and required inputs (inputs are new note commitments, stored in thenote hash tree
3. and nullifiers stored in thenullifiers tree
4.) to the sequencer. Nullifiers are data that invalidate old commitments, ensuring that commitments can only be used once.

The sequencer has received the transaction proof and can begin to process the transaction - verifying proofs and applying updates to the relevantdata trees - alongside other public and private transactions.

1. The sequencer has the necessary information to act
2. – the randomly-selected sequencer (based on the Fernet sequencer selection protocol) validates the transaction proofs along with required inputs (e.g. the note commitments and nullifiers) for this private transfer. The sequencer also executes public functions and requests proofs of public execution from a prover network. The sequencer updates the corresponding data trees and does the same for other private transactions. When the sequencer receives proofs from the prover network, the proofs will be bundled into a final rollup proof.

The sequencer has passed the transaction information – proofs of correct execution and authorization, or public function execution information – to the prover, who will submit the new state root to Ethereum.

1. The transaction settles to L1
2. – the verifier contract on Ethereum can now validate the rollup proof and record a new state root. The state root is submitted to the rollup smart contract. Once the state root is verified in an Ethereum transaction, the private transfer has settled and the transaction is considered final.

Going deeper

Transactions on Aztec start with a call from Aztec.js or the Aztec CLI, which creates a request containing transaction details. This request moves to the Private Execution Environment (PXE) which simulates and processes it. Then the PXE interacts with the Aztec Node which uses the sequencer to ensure that all the transaction details are enqueued properly. The sequencer then submits the block to the rollup contract, and the transaction is successfully mined.

Seethis diagram for a more detailed overview of the transaction execution process. It highlights 3 different types of transaction execution: contract deployments, private transactions and public transactions.

See the page oncontract communication for more context on transaction execution.

Enabling Transaction Semantics: The Aztec Kernel

There are two kernel circuits in Aztec, the private kernel and the public kernel. Each circuit validates the correct execution of a particular function call.

A transaction is built up by generating proofs for multiple recursive iterations of kernel circuits. Each call in the call stack is modeled as new iteration of the kernel circuit and are managed by aFIFO queue containing pending function calls. There are two call stacks, one for private calls and one for public calls.

One iteration of a kernel circuit will pop a call off of the stack and execute the call. If the call triggers subsequent contract calls, these are pushed onto the stack.

Private kernel proofs are generated first. The transaction is ready to move to the next phase when the private call stack is empty.

The public kernel circuit takes in proof of a public/private kernel circuit with an empty private call stack, and operates recursively until the public call stack is also empty.

A transaction is considered complete when both call stacks are empty.

The only information leaked about the transaction is:

1. The number of private state updates triggered
2. The set of public calls generated

The addresses of all private calls are hidden from observers.Edit this page

Previous Authentication Witness Next Smart Contracts