polygon zkEVM

Knowledge Layer

Architecture

Feeding the Proving System with Inputs

v.1.0

June 5, 2024

1 Inputs for the Proving System

1.1 Introduction

In order to generate the proof of the correct processing of a single batch, various inputs are necessary. While the transaction data is an obvious requirement, there are additional inputs to consider, some of which need to be sent to the Smart Contract, while others are already present. The central figures in this aspect of the zkEVM architecture are the **Verifier** (Smart Contract) and the **Prover**, along with their interaction. This section will explore what and why the Prover has to transmit to the Smart Contract.

In the context of zkEVM, our focus is on succinct computation verification rather than privacy concerns. Consequently, L2 transactions and L2 state data are public. Thus, let's delve into the initial design of the proving system, specifically one that operates **without private inputs**.

1.2 Public Inputs

Recall that the Prover sends to the Smart contract the proof and an array of public inputs (`[publics]`) as we can see in the following figure.

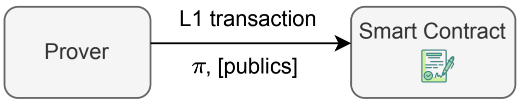

Figure 1: Prover to Smart Contract

About the public inputs:

- `batchData`: All the L2 transactions in the batch that are going to be proved.
- `currentStateRoot`: Current L2 state root.
- `proverAccount`: To receive rewards. Also is attached to the proof so that no one can plagiarise it.
- `timestamp`: Time at which the proof is generated.
- `forkId`: L2 EVM version used.
- `chainId`: The chain for which the proof is being generated. Has the capability to host multiple Layer 2 (L2) networks.

After applying all this we obtain a public output which is `newStateRoot` (new L2 state root).

Not all of these public inputs need to be sent by the Prover to the L1 Smart Contract; some of them are already stored there. These include: `currentStateRoot`, `forkId`, and `chainId`.

Henceforth, in the `calldata` of the transaction sent to the L1's smart contract, we need to include the rest of inputs for the batch verification: `batchData`, `timestamp`, `newStateRoot`. Note that the `proverAccount` is already in the L1 transaction's signature, so we need not provide it explicitly and finally, **the proof** of the correct execution of the L2 transactions within the batch.