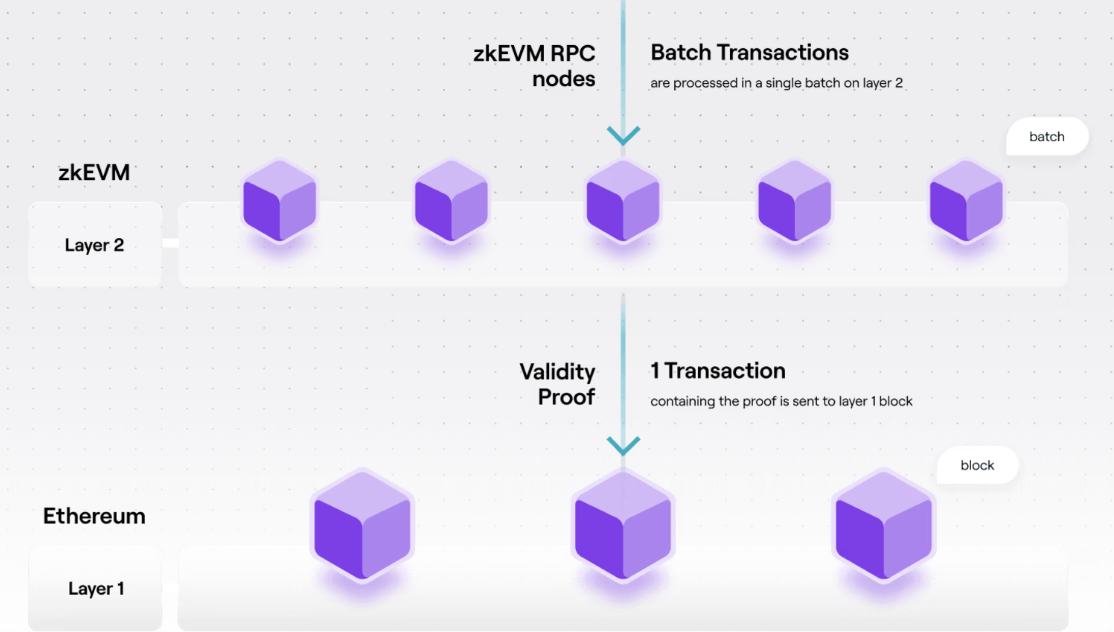
# Polygon zkEVM

===============

The Ethereum blockchain network has been suffering from scalability issues over the past few years.The Polygon zkEVM built by polygon is for anyone who needs a cheaper, faster way to use Ethereum without compromising security and decentralization

Polygon zkEVM is a decentralized Ethereum Layer 2 (L2) scalability solution which is the first EVM-equivalent Zero Knowledge(ZK) L2.

EVM-equivalence lets you deploy your existing solidity code without going through any extra steps to compile your code to get it to work on this network. This means that any smart contract or development tool used on Ethereum can be used on Polygon zkEVM. The zkEVM with its features such as Ethereum compatibility, scalability and security is expected to assist in the mass adoption of Ethereum in the near future.

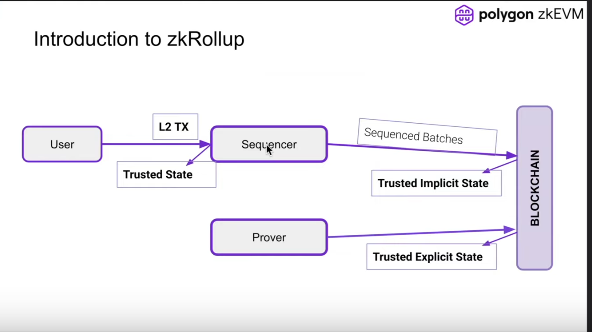


Source: Polygon

The zkEVM rolls up a batch of transactions and uses cryptographic zero-knowledge proofs (ZKP) to validate those transactions on Ethereum.

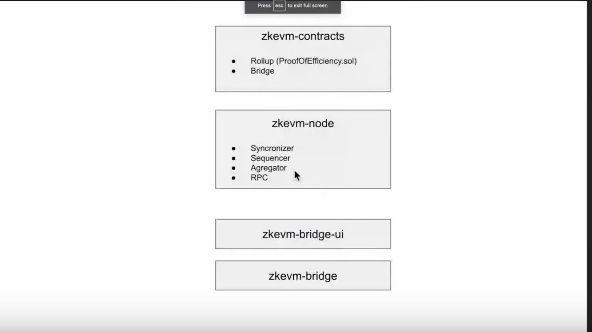
Polygon zkEVM Architecture

================================



The user sends transactions to the sequencer . The sequencer packs the transactions into batches and places them into Blockchain. At this point the funds cannot be withdrawn.

Later on the prover takes the transactions, computes the proof and places the proof (state) on the Blockchain. At this point the funds can be withdrawn from the Blockchain.



Basic architecture

**Zkevm-contracts**  -There are mainly two smart contracts one related to Rollup and one Related to the Bridge

**Zkevm-node** -

zkNode is the software needed to run any zkEVM node. It is a client that the network requires to implement the Synchronization and govern the roles of the participants (Sequencers or Aggregators). Polygon zkEVM participants will choose how they participate:

* As a node to know the state of the network, or
* As a participant in the process of batch production in any of the two roles: **Sequencer** or **Aggregator**

The two permissionless participants of the zkEVM network are: **Sequencers** and **Aggregators**.

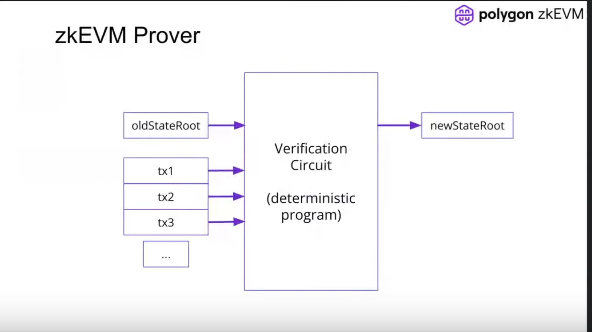
**Sequencer**

* Collect transactions and publish them in a batch
* Receive fees from the published transactions
* Pay L1 transaction fees + MATIC (depends on pending batches)

**Aggregators**

* Process transactions published by Sequencers
* Build zkProof
* Receive MATIC from Sequencer

**ZKevm-bridge-** The bridge handles the interface between smart contracts .An **LX-LY bridge** is a Smart Contract that lets users transfer their assets between two layers, LX and LY. It is a combination of two smart contracts, one deployed on one chain and the second on the other.The L1-L2 in zkEVM is a decentralised bridge for secure deposits and withdrawal of assets. It is a combination of two smart contracts, one deployed on one chain and the second on the other.



-Proving and verification of transactions in **Polygon zkEVM** are all handled by a zero-knowledge prover component called the **zkProver**. All the rules for a transaction to be valid are implemented and enforced in the zkProver.

The prover takes oldStateRoot and a set of transactions and computes the newStateRoot.

The **zkProver** performs complex mathematical computations in the form of polynomials and assembly language which are later verified on a smart contract.

## 

## 

## 

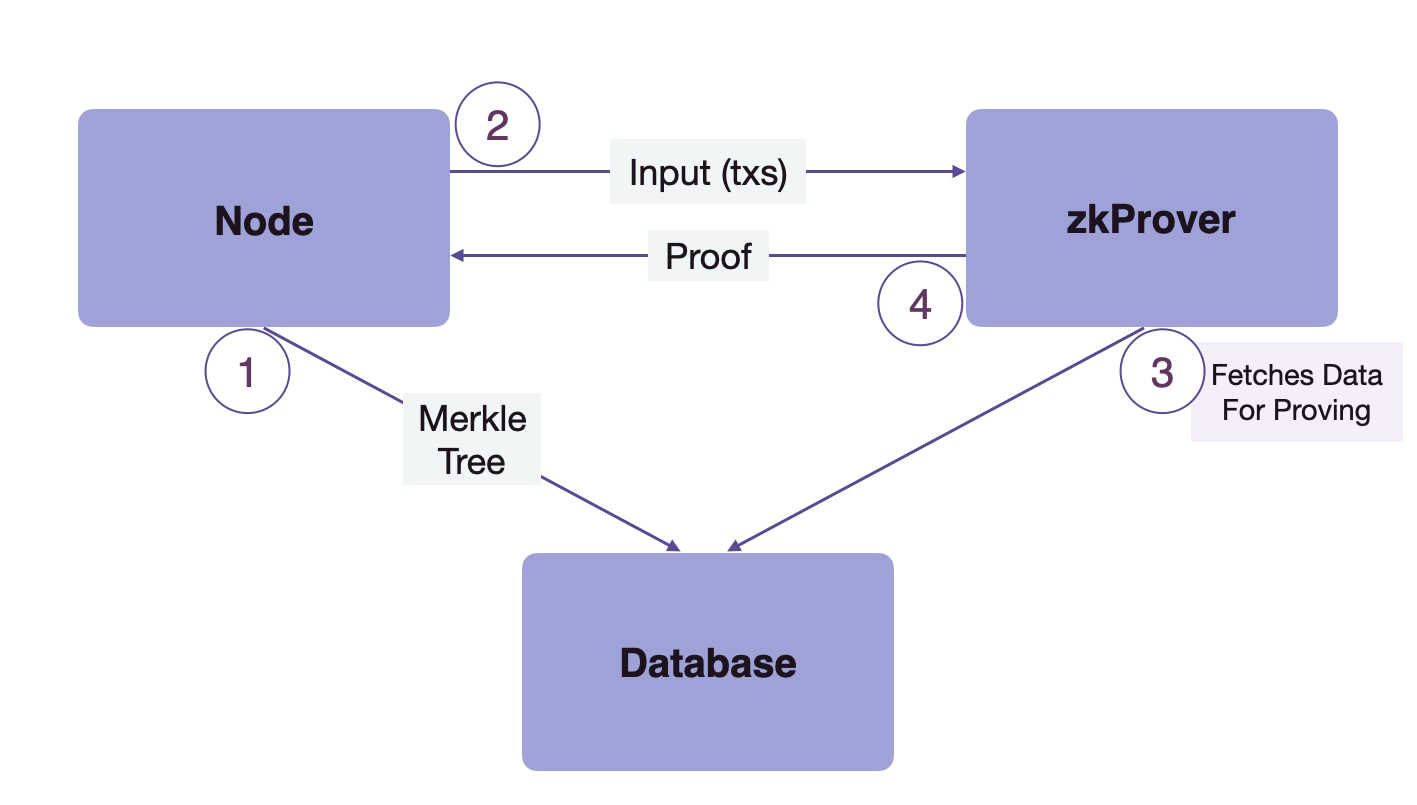
## 

## 

## 

## **Interaction of zkProver with Node and Database**

The zkProver mainly interacts with two components, i.e. the Node and the Database (DB).



**1 →** The Node sends the content of Merkle trees to the Database to be stored there

**2 →** The Node then sends the input transactions to the zkProver

**3 →** The zkProver accesses the Database and fetches the info needed to produce verifiable proofs of the transactions sent by the Node. This information consists of the Merkle roots, the keys and hashes of relevant siblings, and more

**4 →** The zkProver then generates the proofs of transactions, and sends these proofs back to the Node

## Transaction Process of Polygon zkEVM

=======================================

Before getting into a transaction flow in L2, users need some funds to perform any L2 transaction. In order to do so, users need to transfer some ether from L1 to L2 through the zkEVM Bridge dApp.

* Bridge

======

* Deposit ether
* Perform claim on L2 and receive the funds
* L2 Transactions

=============

* User initiates tx in a Wallet (e.g. Metamask) and sends it to a Sequencer
* It gets finalized on L2 once Sequencer commits to add his transaction
* Transaction has finalized on L2, but not on L1 (simply put, L2 state is not yet on L1). Also known as **Trusted State**
* Sequencer sends the batch data to L1 smart contract, enabling any node to synchronize from L1 (known as **Virtual State**)
* Aggregator will take pending transactions to be verified and build a Proof.
* Once the Proof is validated, user's transactions will attain L1 finality (important for withdrawals). This is called the **consolidated state**.

zkEVM characteristics

=================

**Proof of Efficiency (PoE) consensus mechanism** -The earlier version, **Polygon Hermez 1.0**, was based on the **Proof of Donation (PoD)** consensus mechanism.**Proof of Efficiency (PoE)** consensus mechanism leverages the experience of the existing **PoD** in v1.0 and add support for the permissionless participation of multiple coordinators to produce batches in L2.

**Permissionless-ness** **-** The aim is to allow anyone with the zkEVM software to participate in the network.The consensus algorithm will give everyone the opportunity to be a Sequencer or an Aggregator.

**Decentralization-** The aim is to ensure that there is no censorship and that no one party can control the network.

**Security -**Since this is an L2 solution, most of the security is inherited from Ethereum.Smart contracts will ensure that everyone who executes state changes does so appropriately, creates a proof that attests to the validity of a state change, and makes validity proofs available on-chain for verification.