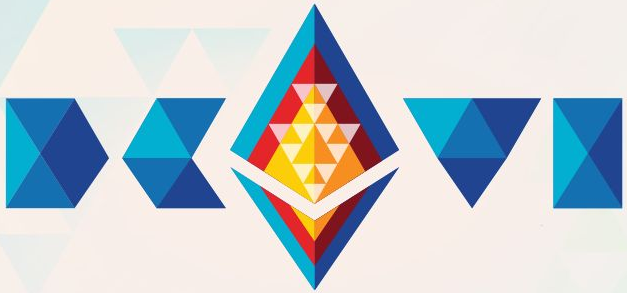
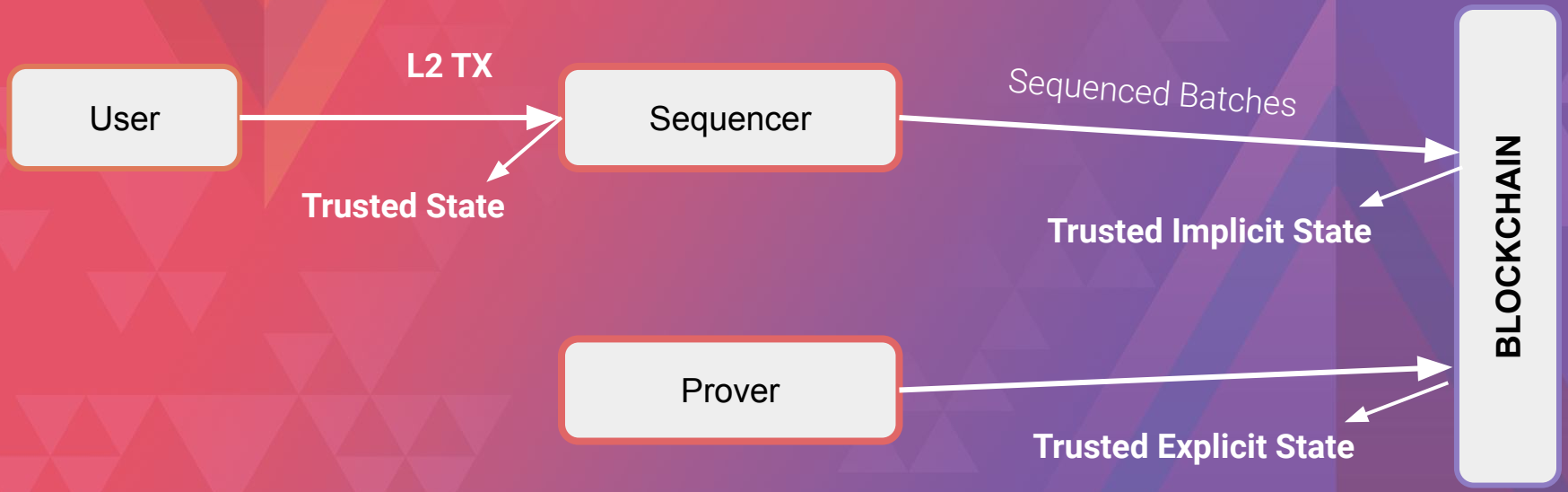


Technical Details of the Opcode Compatible zkEVM

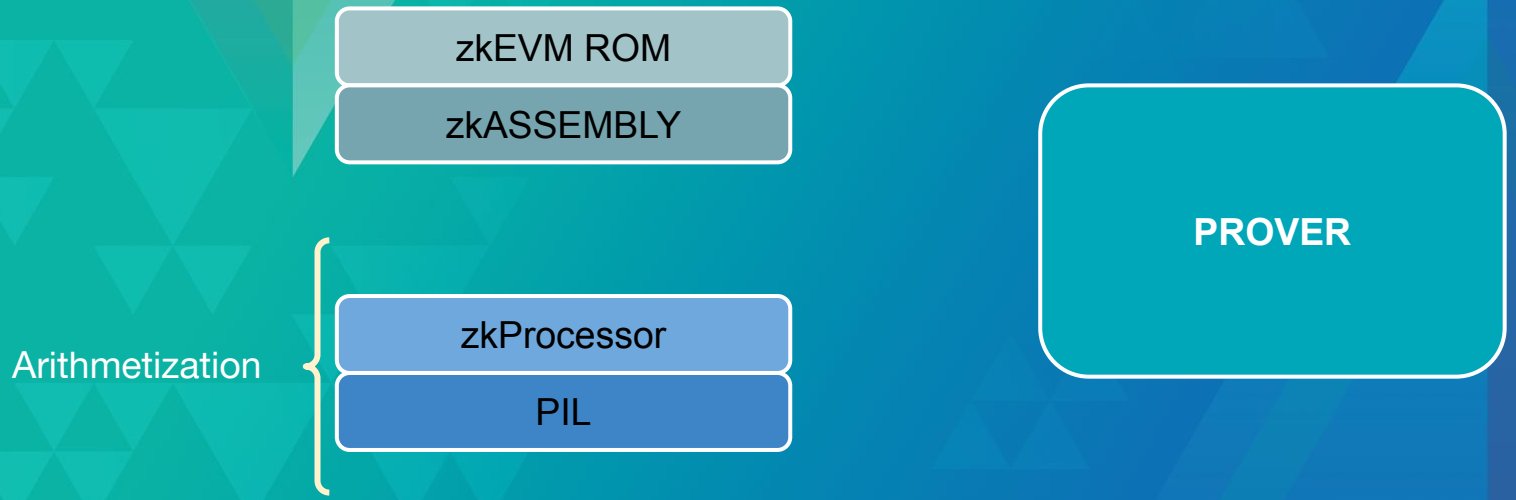
Introducing  polygon zkEVM Testnet

Jordi Baylina
Co-founder and Technical Lead
Polygon Hermez

Introduction to zkRollup



Structure of the Proof



EVM Processor

RAM

- Multiple R/W
- 1 Access per CLOCK
- Paged for handling Ethereum CALL contexts
- 32 byte alignment sub stat machine.

ROM

- The Code that always execute the prover
- It cannot be modified.

STORAGE

- Sparse Merkle Tree
- Goldilocks Poseidon hash function
- Single tree for the system
- Hashes of the smart contract codes are in the tree.

EVM Processor

RAM

BINARY

- Operations done byte to byte with a carry from a plookup table
  - ADD
  - SUB
  - LT & SLT
  - EQ
  - AND
  - OR
  - XOR

ROM

ARITHMETIC

- 256 bits arithmetic operations
- A*B + C = D*2^256 + E
- Range check of inputs and outputs
- 32 CLOCKs per operation
- Includes EC addition formulas for ECDSA multiplication.

STORAGE

HASH

- Binary circuit of ANDn and XOR
- We use a plookup to do various circuits in parallel
- We currently can do 468 keccakf's in the current circuit. (N = 2^23).

ZkASM-ROM

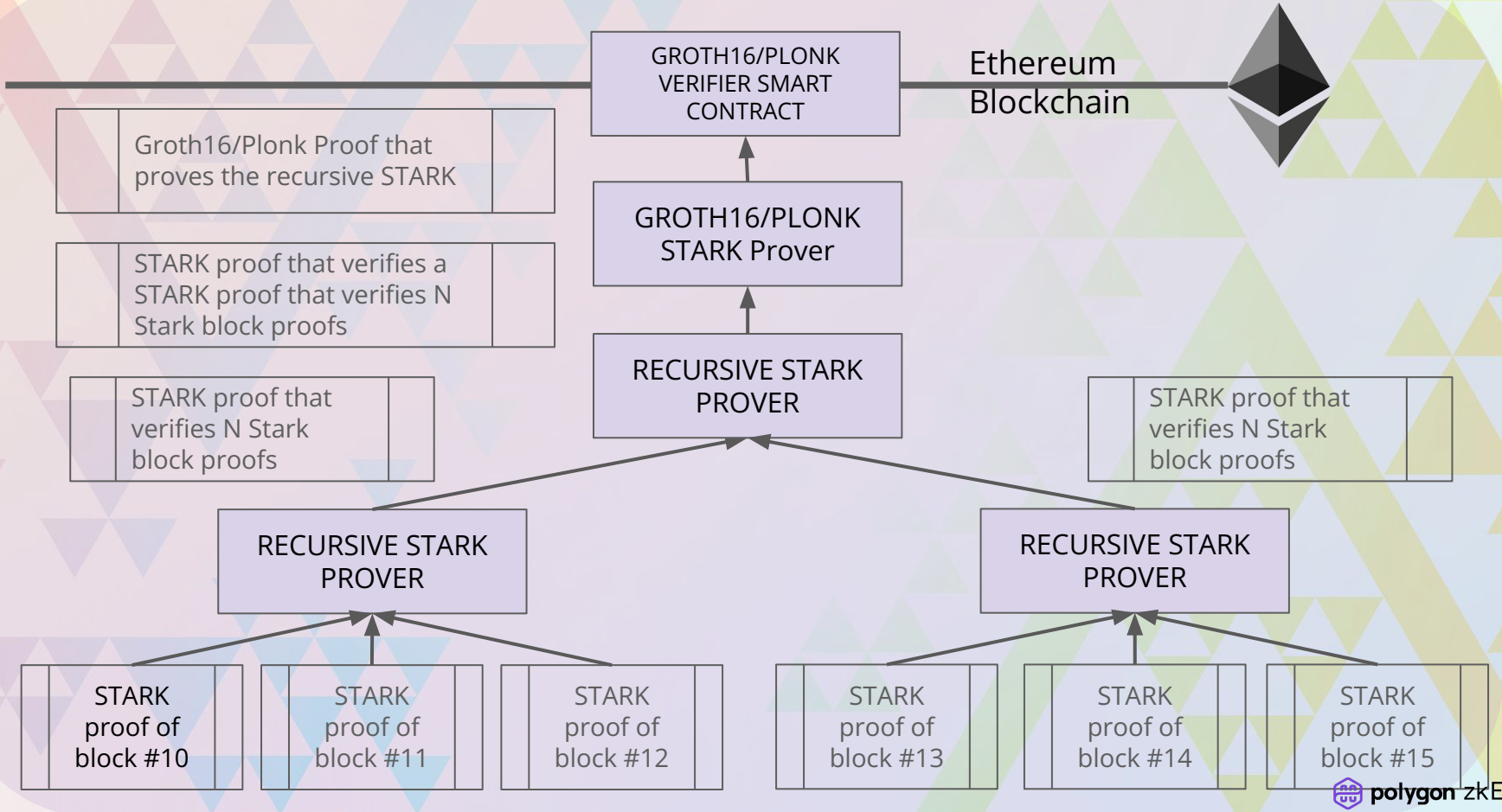
- Ethereum Transaction processor
- FREE Input, the Transactions and the hash must match
- About 16 clocks per gas unit
- zkCounters to prevent the proof to fail (DoS)

Some examples:

- Opcodes
- RLP Processing

```
2109  opPUSH31:
2110      31 => D
2111      $ => B                          :MLOAD(isCreateContract)
2112      0 - B                           :JMPN(opAuxPUSHB)
2113                                      :JMP(opAuxPUSHA)
2114
2115  opPUSH32:
2116      32 => D
2117      $ => B                          :MLOAD(isCreateContract)
2118      0 - B                           :JMPN(opAuxPUSHB)
2119                                      :JMP(opAuxPUSHA)
2120
2121  opDUP1:
2122
2123      %MAX_CNT_STEPS - STEP - 120 :JMPN(outOfCounters)
2124
2125      SP - 1 => SP    :JMPN(stackUnderflow)
2126      $ => A          :MLOAD(SP++)
2127      1024 - SP       :JMPN(stackOverflow)
2128      A               :MSTORE(SP++)
2129      1024 - SP       :JMPN(stackOverflow)
2130      GAS-3 => GAS    :JMPN(outOfGas)
2131                      :JMP(readCode)
2132
2133  opDUP2:
2134
2135      %MAX_CNT_STEPS - STEP - 120 :JMPN(outOfCounters)
2136
2137      SP - 2 => SP    :JMPN(stackUnderflow)
2138      $ => A          :MLOAD(SP)
```

Recursion and on-chain verification



polygon zkEVM

Time for the zkEVM Testnet DEMO

public.zkevm-test.net

1

Transfer goerliETH L1 -> L2

2

Deploy a simple smart contract.

3

Call the smart contract

4

Deploy a Withdraw L2 -> L1

A Scaling Solution fully compatible with the Ethereum Ecosystem

Our community of dApp Developers should not be able to notice any difference between developing on Ethereum L1 and Polygon zkEVM.

Our Commitment is Security and Zero friction for the dApp Developer and for Users to have transparent smart contract execution with off chain validity proofs.

The zkEVM design offers:

- Same tooling
- Same language (Solidity)
- Same gas model

Fast finality (Centralized Sequencer).

Maintaining Security backed by Ethereum.

Vision and Design Goals

polygon zkEVM

IT's a TESTNET not a showcase

Let's Test it!

Input and constructive feedback Welcomed!

Every problem we manage to find and fix in Testnet will be avoided in Mainnet

What to expect?

- Expect some restarts
- Expect bugs
- Expect that it will not be available for some periods

Prover Costs

- In a single CPU with 192 cores ($9/h in AWS), a more than 4M gas proof takes about 9 minutes to be generated
- Cost per TX is less than $0.007
- Margins we are working to improve:
  - Coding optimisations
  - Mathematical optimisations
  - GPU / FPGA
  - Design improvements

What is missing today

- Support for pre EIP-155 TXs and modern EIP-2718 Transaction types

- Support for SHA256, BLAKE and PAIRINGS In our roadmap for development

PREPARING FOR AUDIT

- Developing an Audit that serves all rollups and leveraging community effort
- Audit structure
  - Smart contracts: rollup and bridge
  - Arithmetization (PIL)
  - ROM
  - Proving System
- STARK
  Recursive STARK
  SNARK
- Tooling zkASM and PILCOM
- UI safety and General Infrastructure security

- Optimizing, bug fixing, testing, observing and ready to provide support for the public testnet
- Implementing the key elements and differences to become a type 2 rollup according to Vitalik's classification
- Aggregating proofs
- Getting us ready for Audit
  a. We already open source all the repos.
  b. Next auditors' training program.
- EIP-4844 Data availability in Danksharding.

We are collaborating with EF and other rollups in a community effort for auditing Smart Contracts

What is the zkEVM team working on?

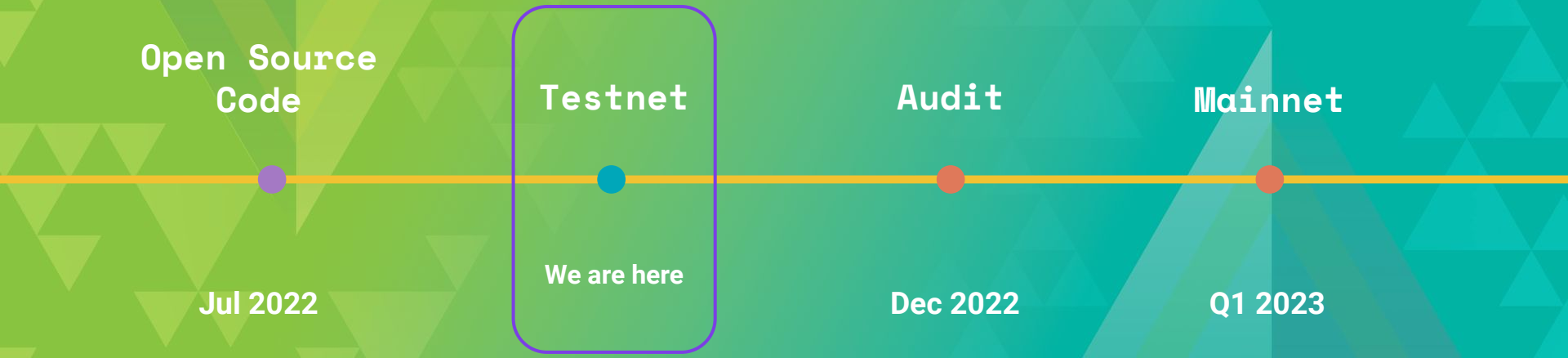
TESTING UPDATES

We are using Ethereum test suit to verify the zkEVM

The Goal is to maintain a high level of equivalence

We are passing 97% of the Ethereum test suite

zkEVM Roadmap



polygon zkEVM

Open Source

https://github.com/0xPolygonHermez

Core repos:

- zkevm-proverjs
- zkevm-rom
- zkevm-prover
- zkevm-node
- zkevm-contracts
- zkevm-bridge-service
- zkevm-bridge-ui
- zkevm-doc

zkEVM specific tools and libraries

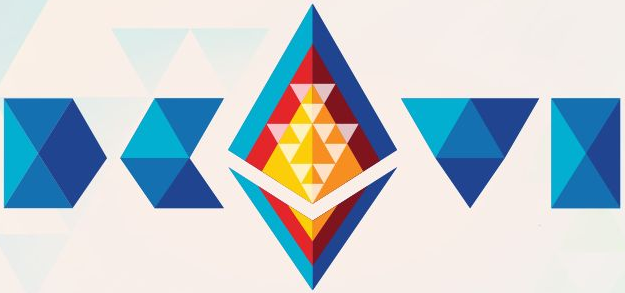
- zkevm-commonjs
- zkasmcom
- zkevm-testvectors
- zkevm-storagerom

generic tools and libraries

- pilcom
- pil-stark

zkEVM is no longer a myth

It's now here



Thank you!

Jordi Baylina

Technical Lead and Co-founder
Polygon Hermez

@jbaylina