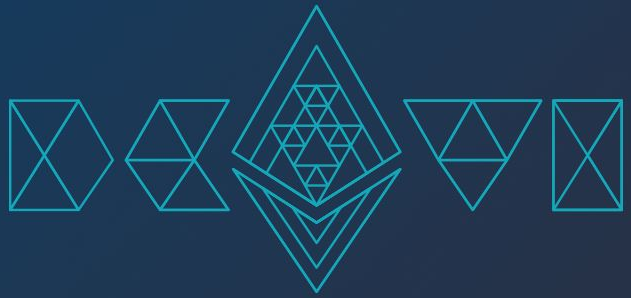


Introduction to Circom2.0

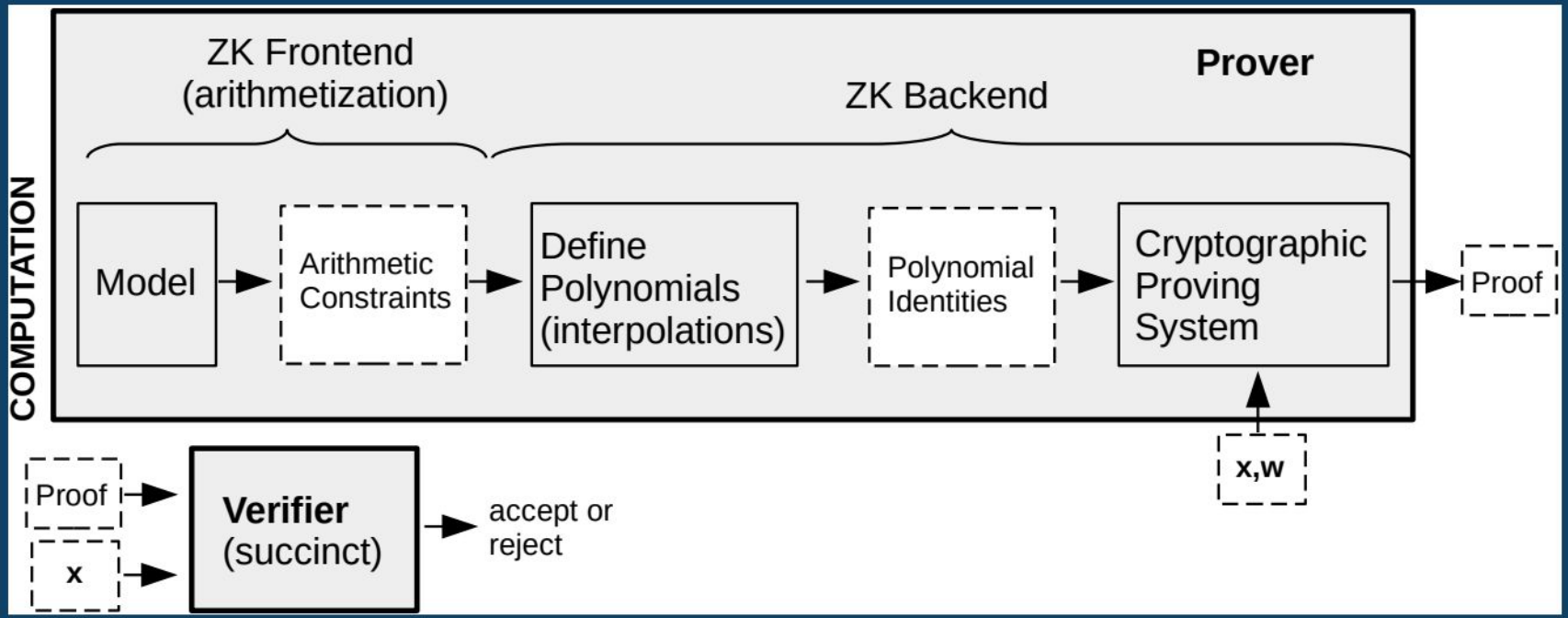
Jesús Ligero & Carlos Matallana

Polygon Hermez

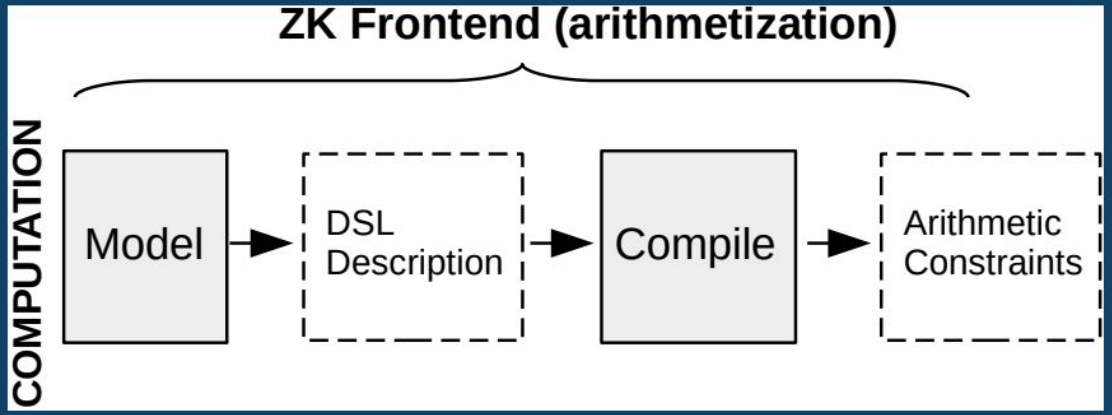
Overview

- Circom language overview
- Introduction to Hermez-zkRollup circuit
  - Account leaf
- Application coding
  - Goal: Prove an account has made more than 10 transactions in Hermez-zkRollup to get an NFT
  - Code circuit in circom 2.0
  - Extract Verifier.sol
  - Overview smart contracts and full flow

High-Level Building Blocks



Extended Frontends

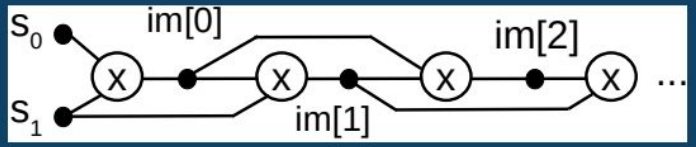


Domain Specific Languages (DSLs) and their corresponding compilers, help in writing high amounts of constraints of big computations.

Circom is a DSL and its corresponding compiler.

Multiplicative Fibonacci Expressed in the Circom DSL i

```
pragma circom 2.0.0;

template MultiplicativeFibonacci(n) {
    signal input s0;    // private by default
    signal input s1;    // private by default
    signal output out;  // public by default
    signal im[n];       // intermediate signals

    for (var i=0; i < n ; i++) {
        if (i==0) {
              im[0] <== s0 * s1; // computation + constraint
              // im[0] <-- s0 * s1; // computation
              // im[0] === s0 * s1; // constraint
        } else if (i==1) {
            im[1] <== s1* im[0];
        } else {
            im[i] <== im[i-1] * im[i-2];
        }
    }
    out <== im[n-2];
}

component main = MultiplicativeFibonacci(1022);
```




```
$ circom mfibonacci.circom --r1cs

template instances: 1
non-linear constraints: 1021
linear constraints: 0
public inputs: 0
public outputs: 1
private inputs: 2
private outputs: 0
wires: 1025
labels: 1026
```

Prime number used:

21888242871839275222246405745257275088548364400416034343698204186575808495617

Multiplicative Fibonacci Expressed in the Circom DSL ii

- Compile to output the witness calculator:

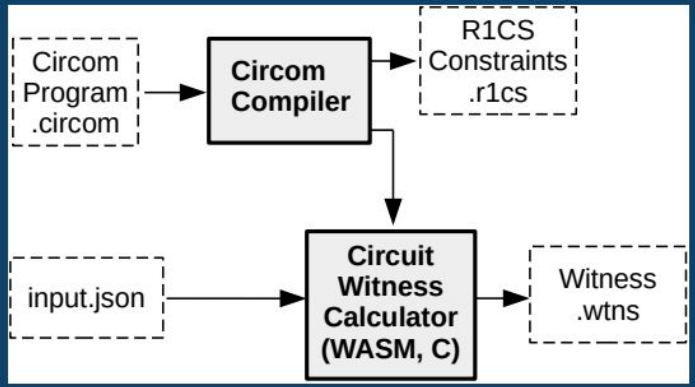
```
$ circom mfibonacci.circom --r1cs --wasm --sym --c

template instances: 1
non-linear constraints: 1021
linear constraints: 0
public inputs: 0
public outputs: 1
private inputs: 2
private outputs: 0
wires: 1025
labels: 1026
```

- Create an `input.json`:

```
1  {"s0": 2, "s1": 1 }
```

- Execute the witness calculator:

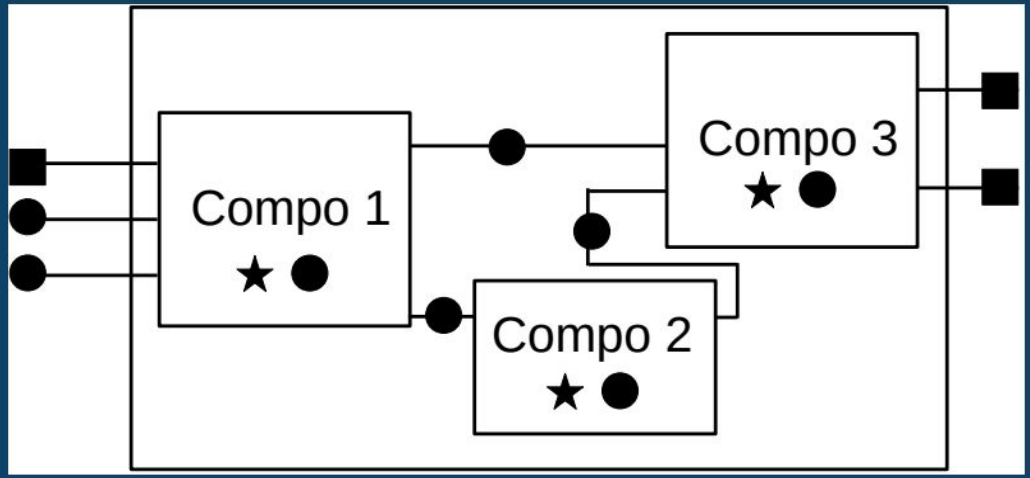
```
$ node generate_witness.js mfibonacci.wasm input.json output.wtns
# or
$ snarkjs wc mfibonacci.wasm input.json output.wtns
```



- The witness calculator is used by the prover to compute all the wires.
- The values of the wires (witness) will not be seen by the verifier.
- The verifier, with polynomial's representatives, checks identities.

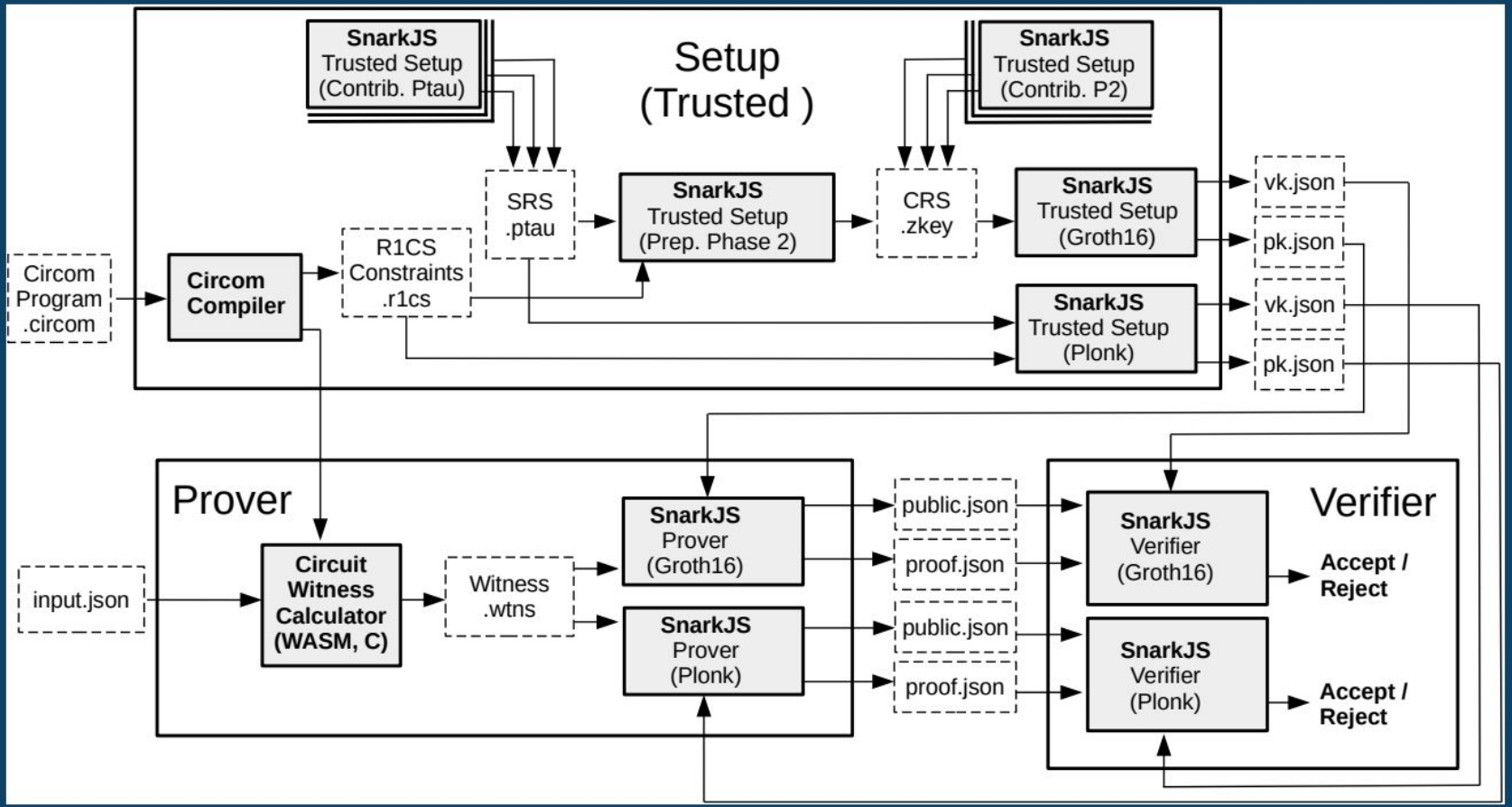
Circuits as Interconnected Components in circom

- Circom is a language similar to the verilog hardware description language: we create components by instantiating templates and interconnect these components to create more complex circuits.
- For this reason, the values of signals cannot be used to create the shape of the circuit.
- Computed values (with `<--`) are indeed private inputs.



- Constraints for connections do not increment the total number of constraints (because we can apply the simplification of linear constraints).

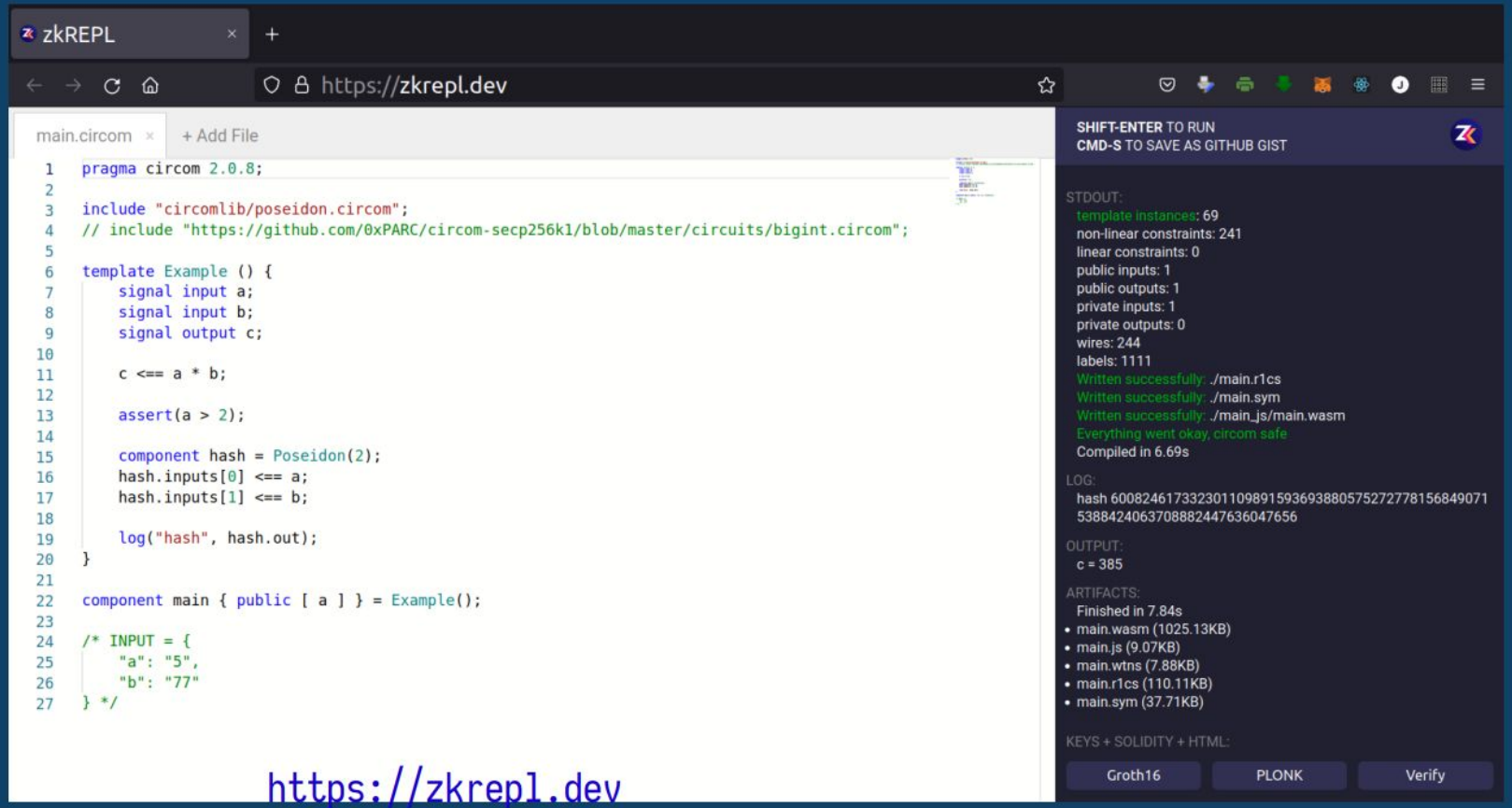
Proofs of Circuits with Circom/SnarkJs (Groth16 and Plonk)



State of the Art & Tools for ZK Circuits ii

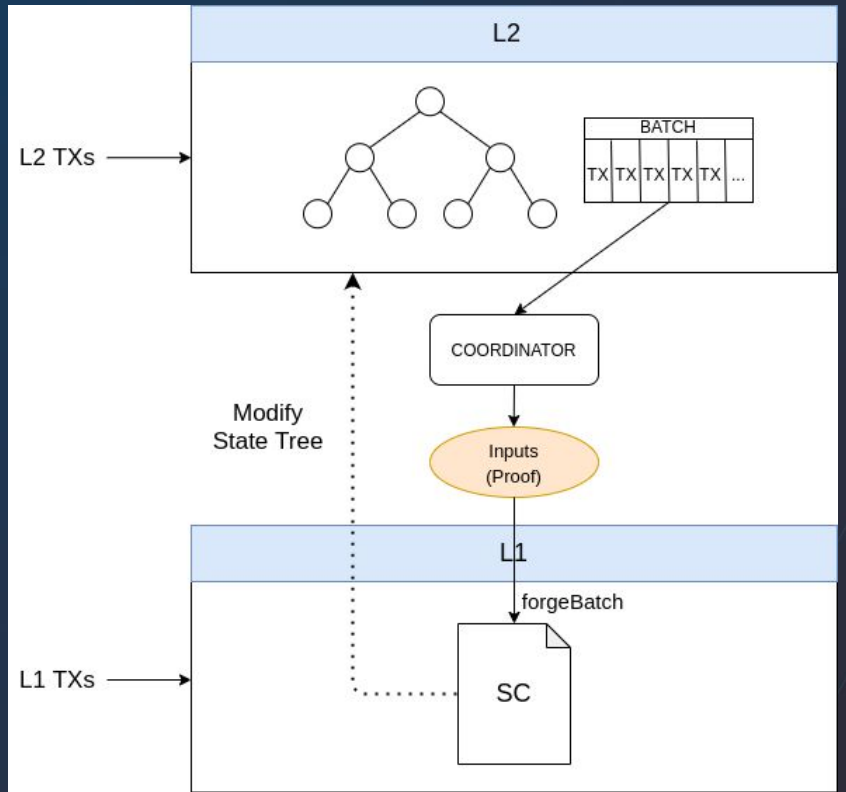
Tool	Description	Language	GitHub repository	Popularity
Libsnark	Desktop/server	C++	scipr-lab/libsnark	1.5k
snarkjs	Browser and server	JavaScript	iden3/snarkjs	983
Bellman	Desktop/server	Rust	zkcrypto/bellman	625
PySNARK	Python gadget library	Python	meilof/pysnark	104
EMP	Interactive protocols	C++	emp-toolkit/emp-zk	35
ZPiE	Library for embedded systems	C	xevisalle/zpie	12
Frontends				
Tool	Type	DSL	Compiler	GitHub repository	Popularity
ZoKrates	Program	Python-like	Rust	Zokrates/ZoKrates	1.3k
circom	Hardware	circom	Rust	iden3/circom	425
snarky	Program	OCaml-like	OCaml	o1-labs/snarky	412
Zinc	Program	Rust-like	Rust	matter-labs/zinc	305
Leo	Program	Rust-like	Rust	AleoHQ/leo	285
xJsnark	Program	Java-like	Java	akosba/xjsnark	154
Buffet	Program	C-like	C, C++	pepper-project	111
Other					
Tool	Description			GitHub repository	Popularity
zkInterface	Standard tool for zero-knowledge interoperability			QED-it/zkinterface	97
CirC	Tool for compilers			circify/circ	79

Using zkrepl.dev



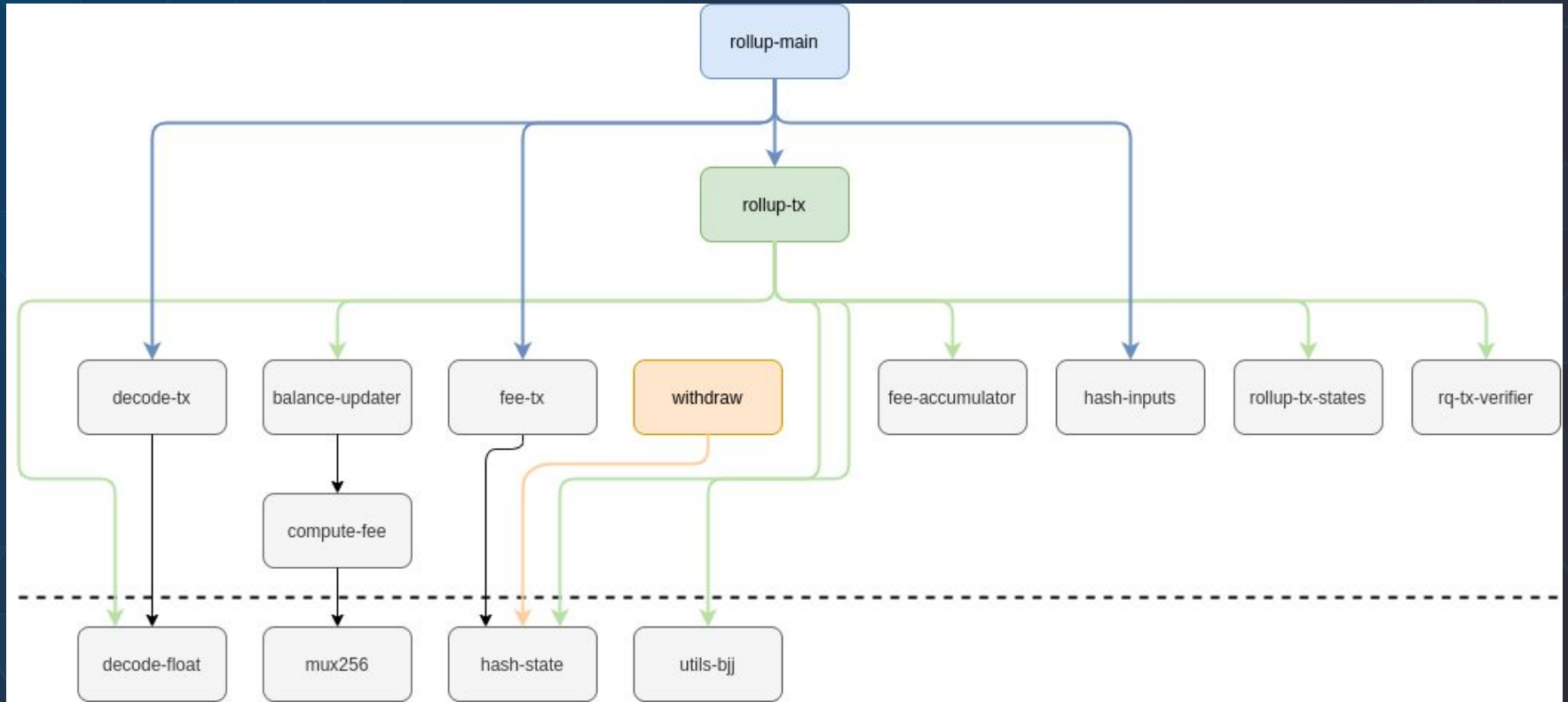
https://zkrepl.dev

Introduction to Hermez-zkRollup circuit



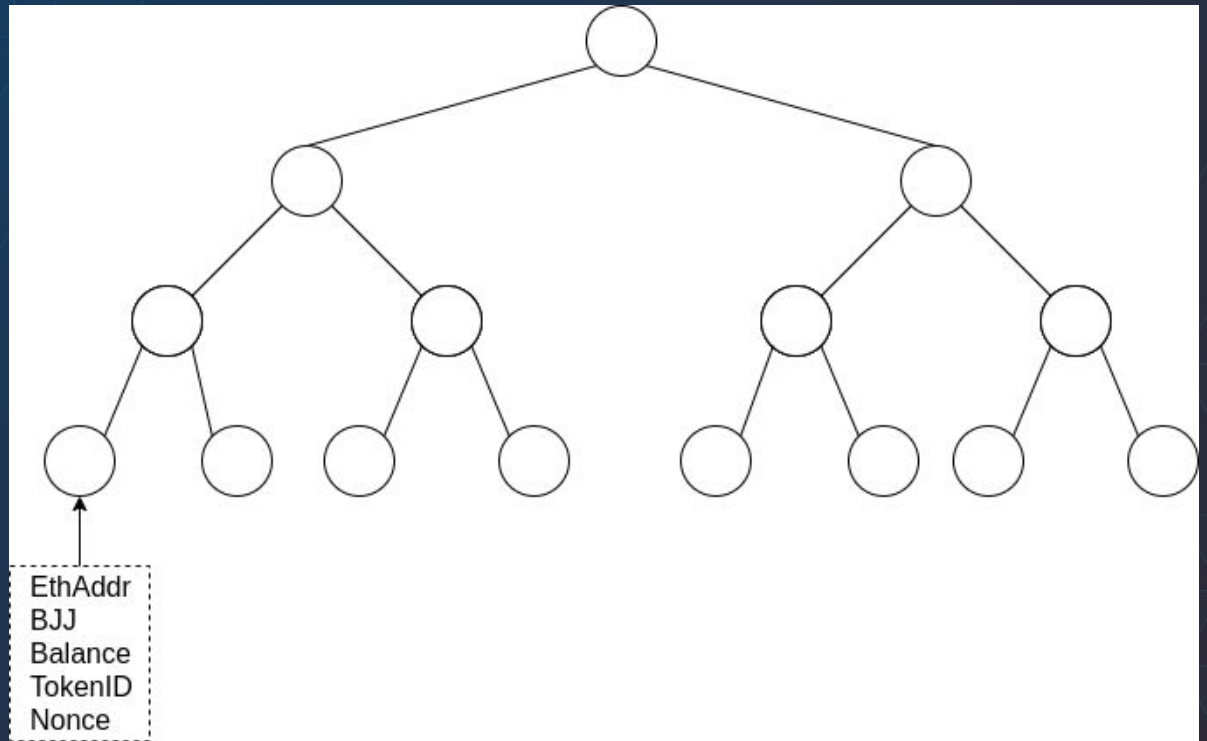
Introduction to Hermez-zkRollup circuit

Circuits involved



Introduction to Hermez-zkRollup circuit

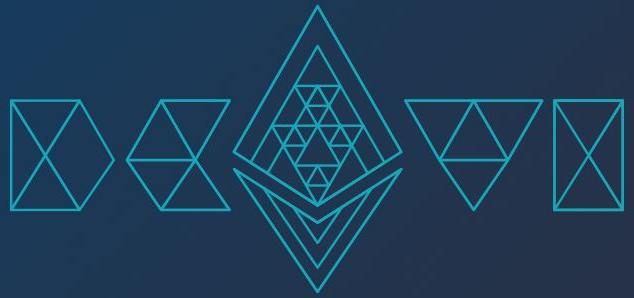
State-tree



Introduction to Hermez-zkRollup circuit

Each leaf of the state tree (account) contains the following data:

- Key: **idx** (merkle tree index)
- Value: **Hash(state)**
  a. State hash = H(e0, e1, e2, e3)
     e_0: [32 bits] tokenID [40 bits] nonce [1 bit] sign
  b. e_1: [192 bits] balance
  c. e_2: [253 bits] ay
  d. e_3: [160 bits] ethAddr



Let’s code!

https://github.com/krlosMata/devconVI-workshop

Jesús Ligero & Carlos Matallana

Polygon Hermez


Thank you!^^