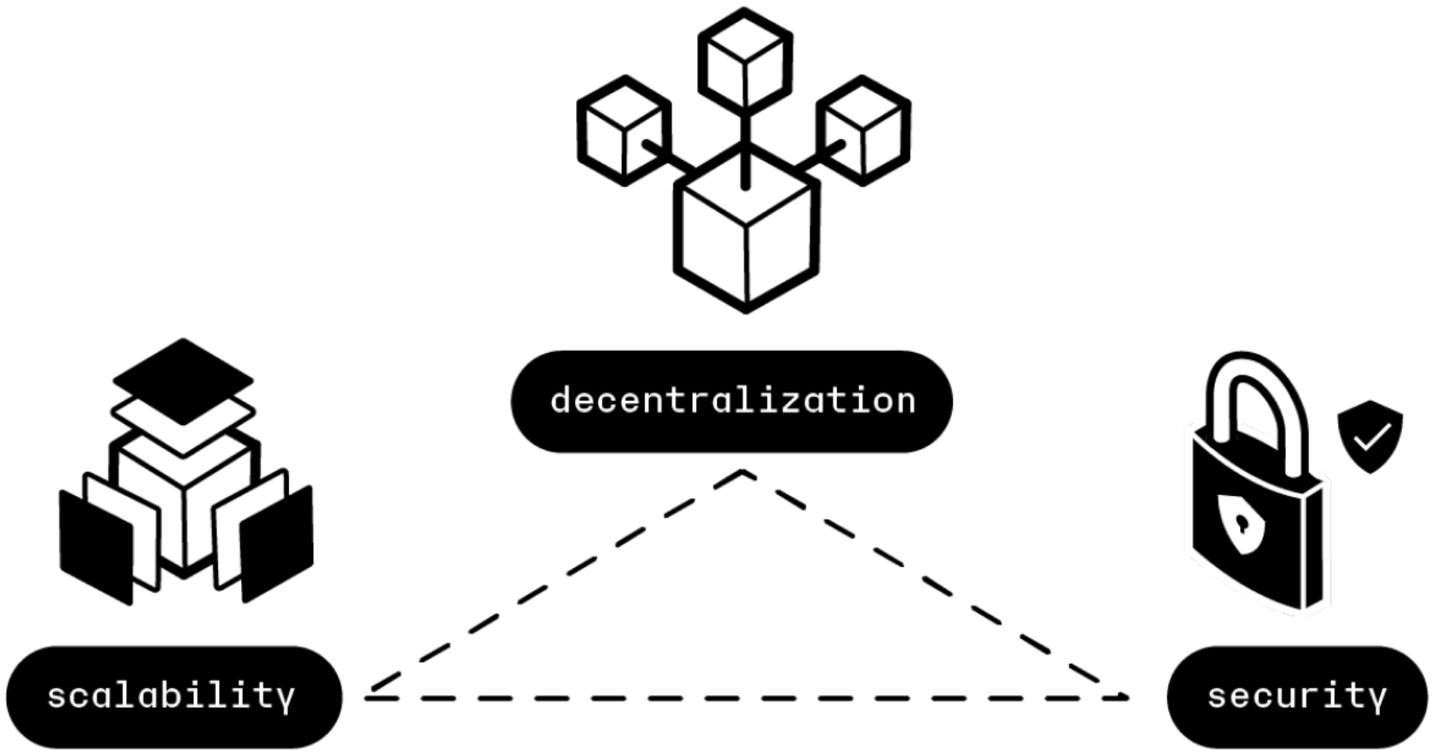
zkEVM

1조 - 유인선, 박보현, 노종찬

Table of Content

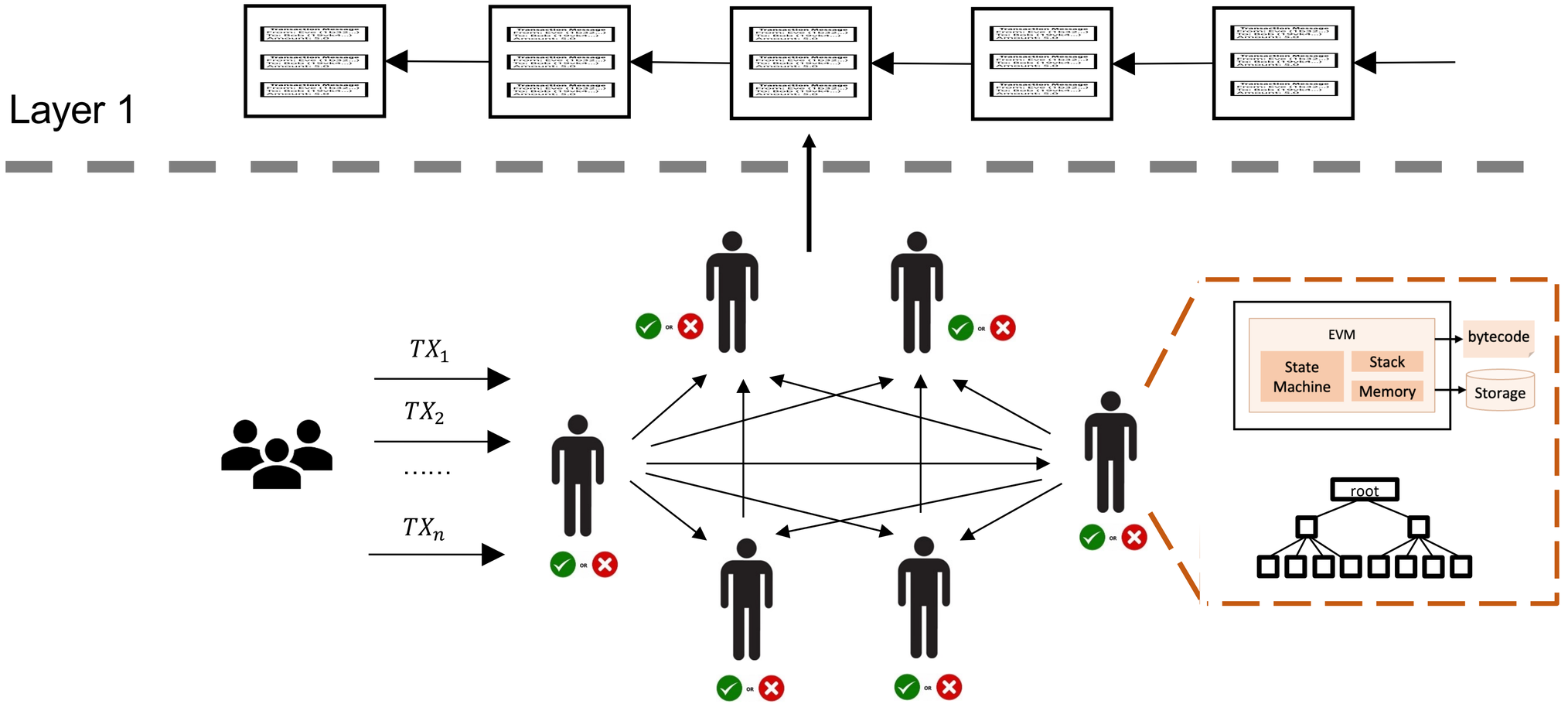
1. Motivation

Blockchain Trilemma



Ref) https://coinloan.io/blog/what-is-blockchain-trilemma/

Scaling Etheruem



Ref - ZKP MOOC

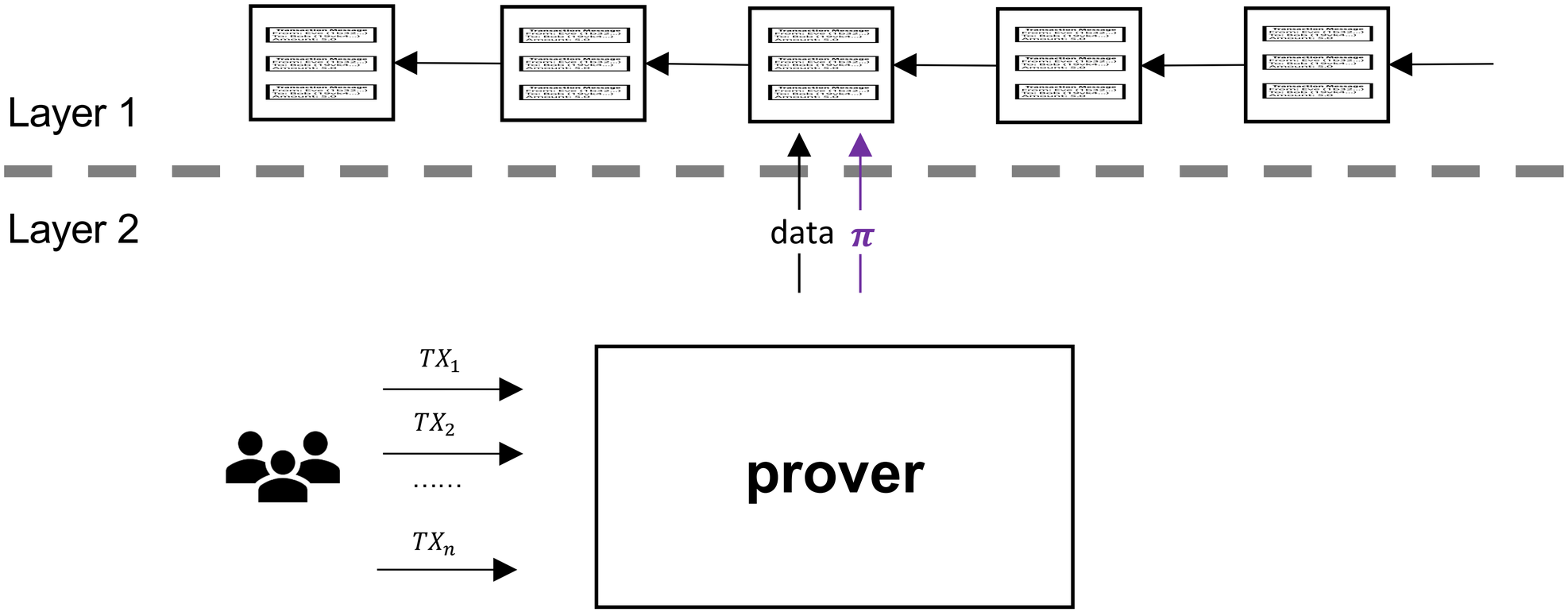
https://www.youtube.com/watch?v=vuQGdbpDWcs

Scaling solution in L2

Execute STF in off-chain

	Fault Proof (Fraud Proof)	Validity(ZK) Proof
Offchain	Plasma / Optimistic Layer	Validium
Onchain	Optimistic Rollup	ZK-Rollup

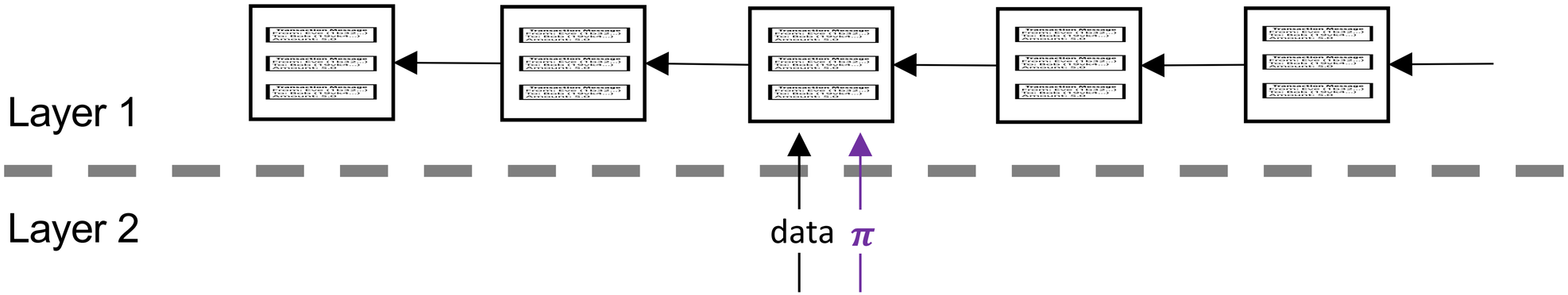
ZK-Rollup

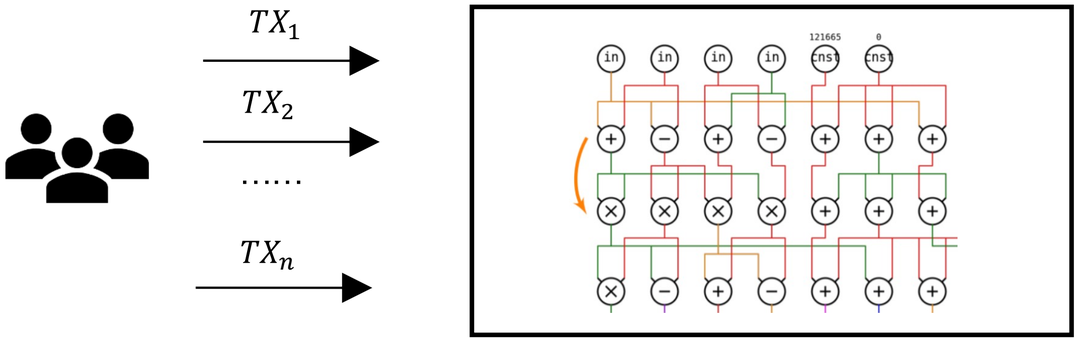


Ref - ZKP MOOC

https://www.youtube.com/watch?v=vuQGdbpDWcs

Challenge

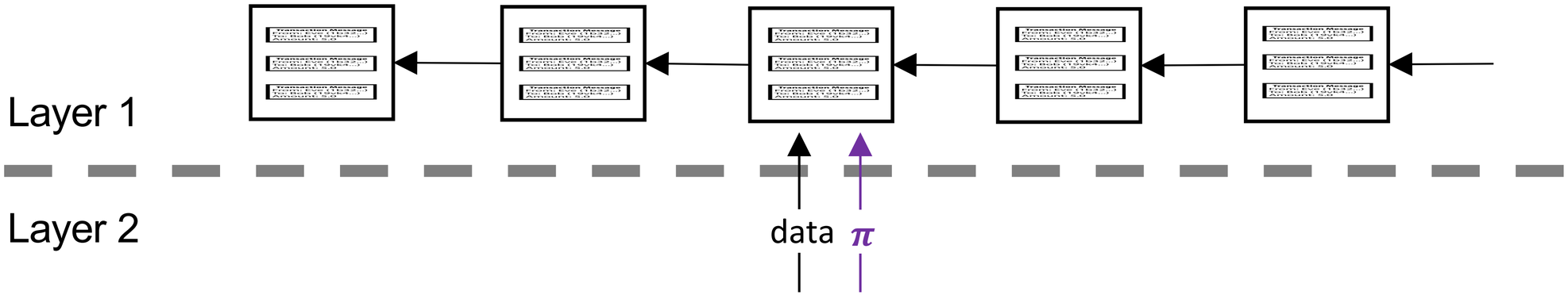


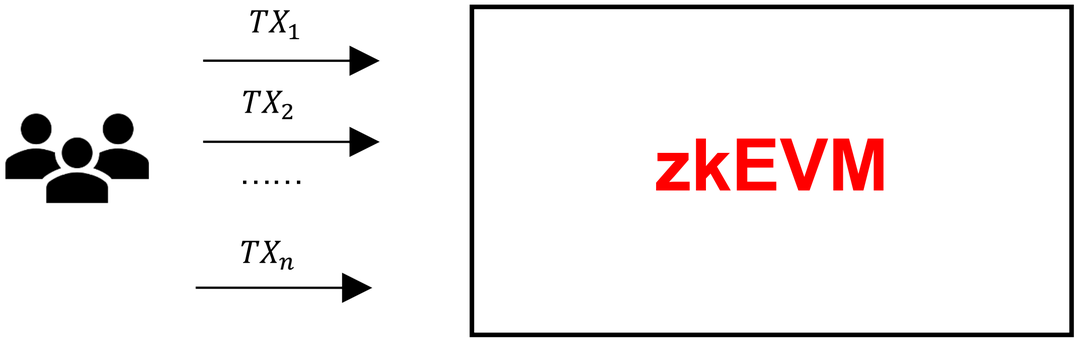


Ref - ZKP MOOC

https://www.youtube.com/watch?v=vuQGdbpDWcs

zkEVM



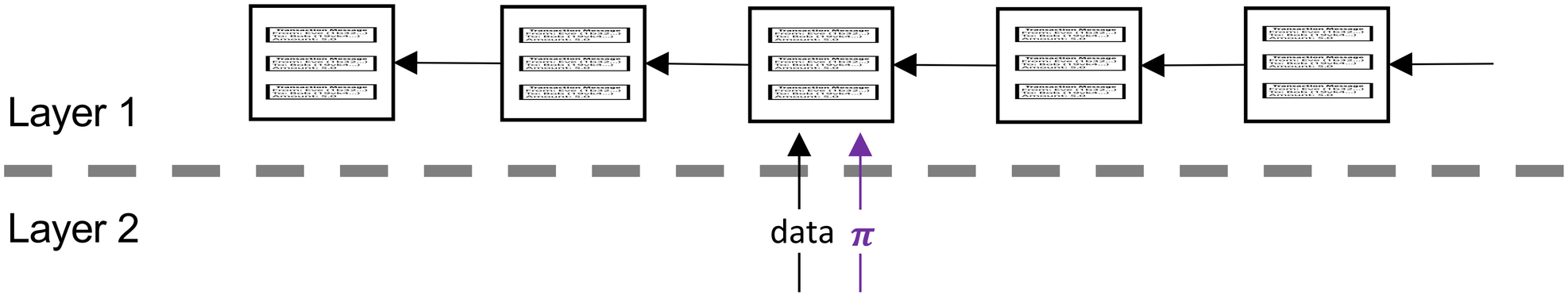


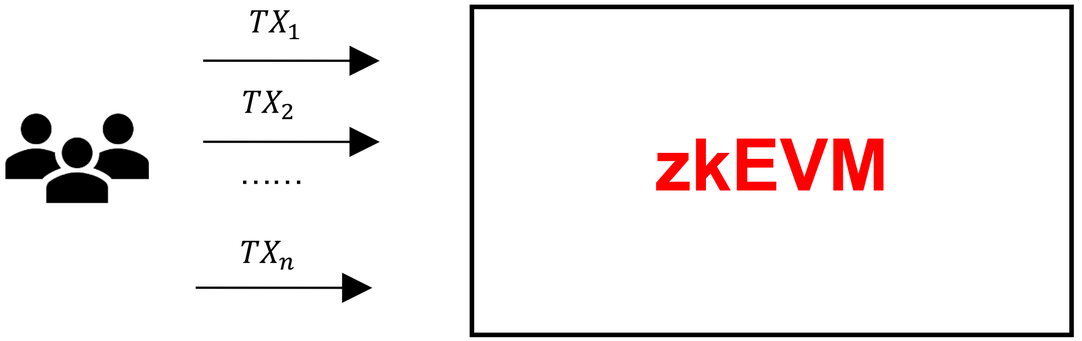
- Developer friendly
- Composability
- Hard to build ☹
- Large proving overhead ☹

Ref - ZKP MOOC

https://www.youtube.com/watch?v=vuQGdbpDWcs

zkEVM





- Polynomial commitment
- Lookup + Custom gate
- Hardware acceleration
- Recursive proof

Ref - ZKP MOOC

https://www.youtube.com/watch?v=vuQGdbpDWcs

2. Background

ZK-SNARK

- **ZK**: **Z**ero **K**nowledge

  어떤 참인 명제에 대해 검증자(Verifier)가 상호작용 이후에 할 수 있는 연산 = 이전에 할 수 있었던 연산

- **SNARK**: **S**uccinct **N**on-Interactive **AR**gument of **K**nowledge

  Succinct: 간결한

  Non-Interactive: 비대화형

  Argument of Knowledge: 증명

ZK-SNARK workflow

Program

```
def hcf(x, y):
   if x > y:
       smaller = y
   else:
       smaller = x

   for i in range(1,smaller + 1):
       if((x % i == 0) and (y % i == 0)):
           hcf = i

   return hcf
```

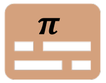
R1CS
Plonkish
AIR

Constraints

```
x * x == var1
var1 * x ==  y
(y+x) * 1 == var2
(var2+5) * 1 == out
```

Polynomial IOP
+
PCS

Proof

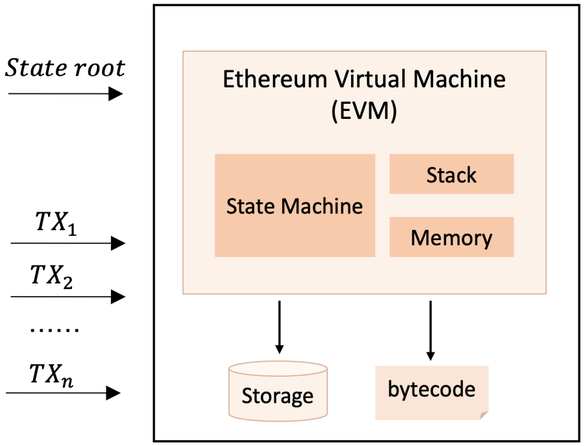


Ref - ZKP MOOC

https://www.youtube.com/watch?v=vuQGdbpDWcs

ZK-SNARK workflow

Program



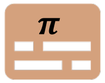
Constraints

R1CS
Plonkish
AIR

```
x * x == var1
var1 * x == y
(y+x) * 1 == var2
(var2+5) * 1 == out
```

Plonk IOP
+
KZG

Proof

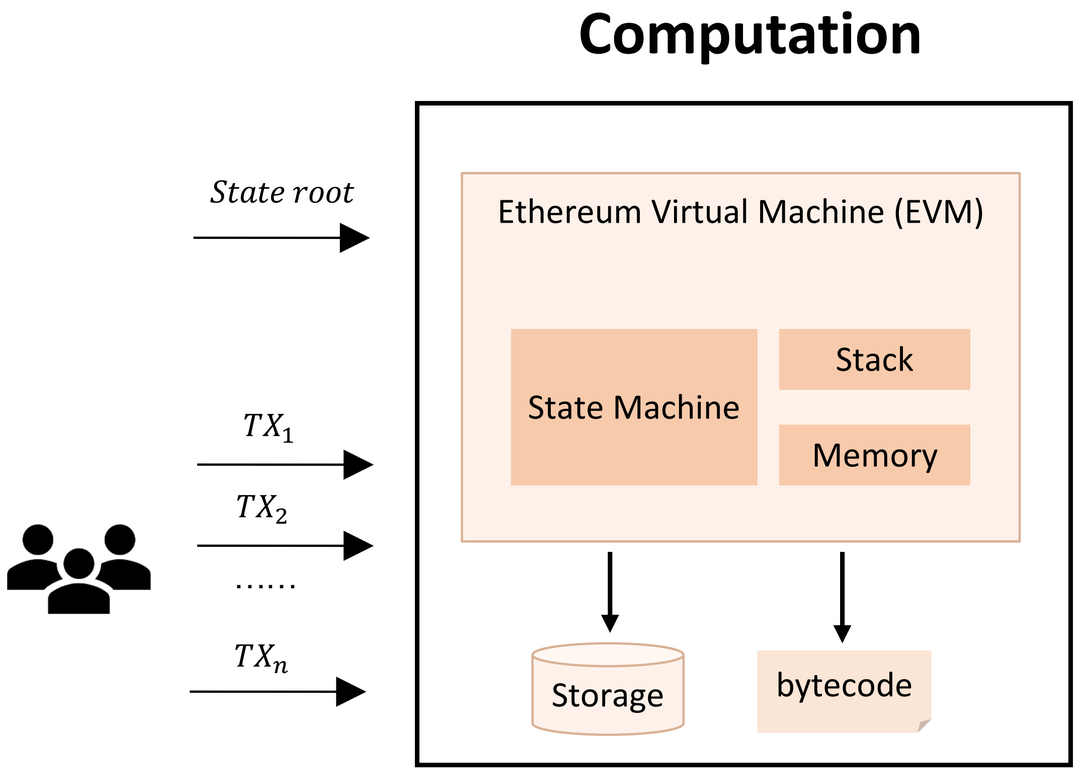


Ref - ZKP MOOC

https://www.youtube.com/watch?v=vuQGdbpDWcs

3. Architecture of zkEVM

What zkEVM has to prove

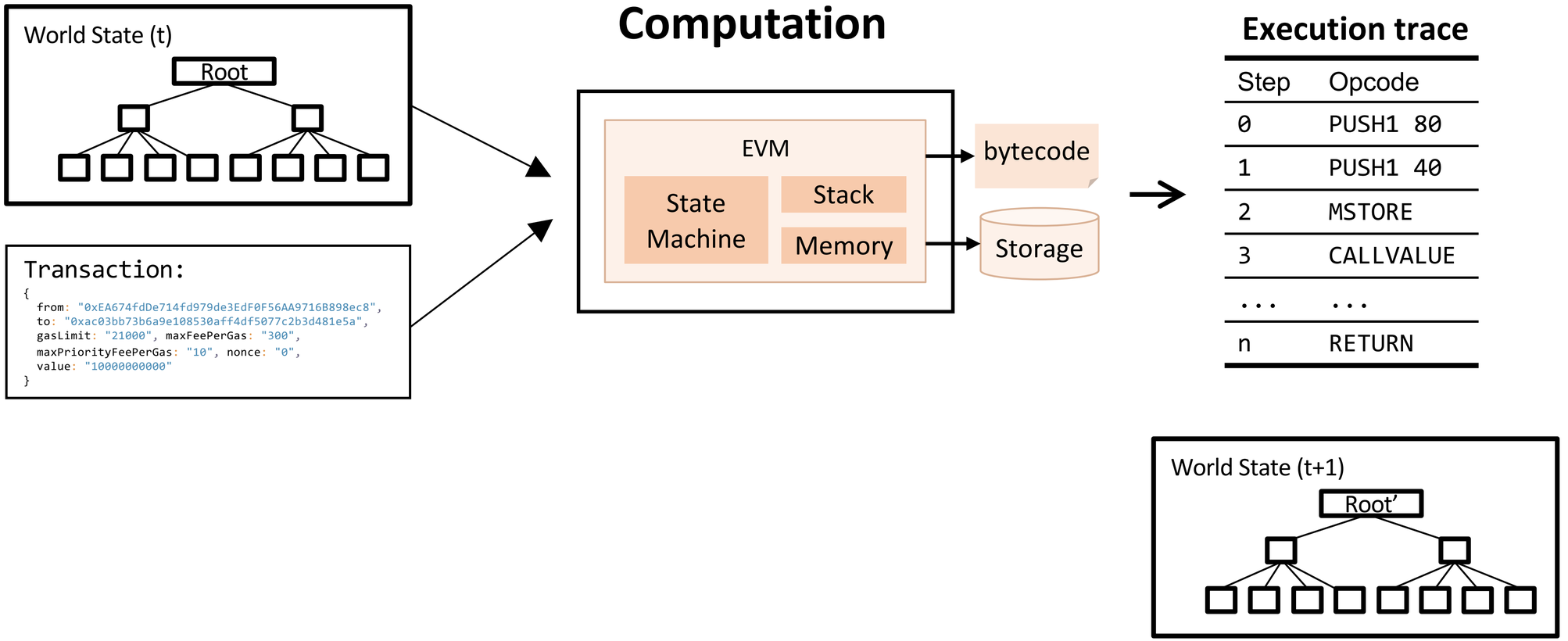


- EVM word size is 256bit
  - Efficient range proof
- EVM has zk-unfriendly opcodes
  - Efficient way to connect circuits
- Read & Write consistency
  - Efficient mapping
- EVM has a dynamic execution trace
  - Efficient on/off selectors

Ref - ZKP MOOC

https://www.youtube.com/watch?v=vuQGdbpDWcs

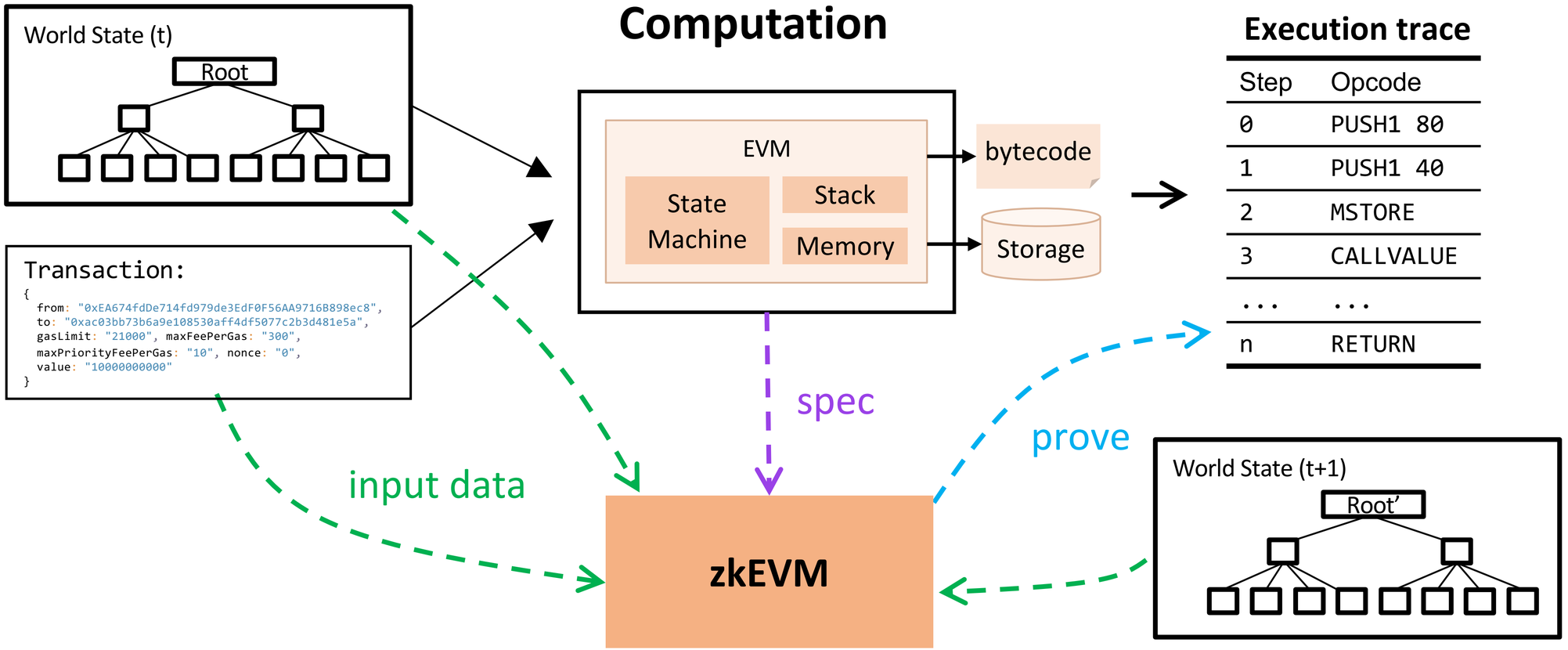
What zkEVM has to prove



Ref - ZKP MOOC

https://www.youtube.com/watch?v=vuQGdbpDWcs

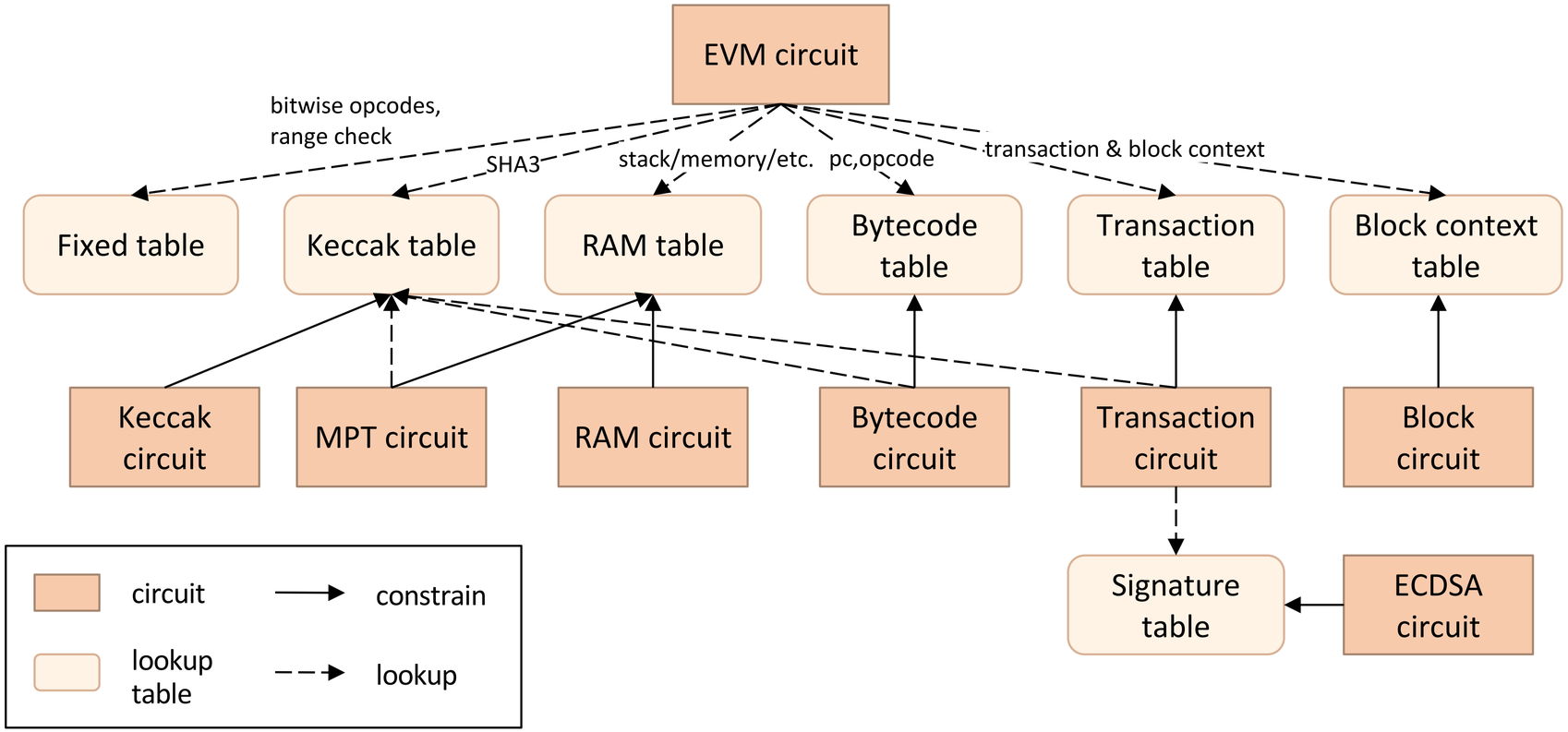
What zkEVM has to prove



Ref - ZKP MOOC

https://www.youtube.com/watch?v=vuQGdbpDWcs

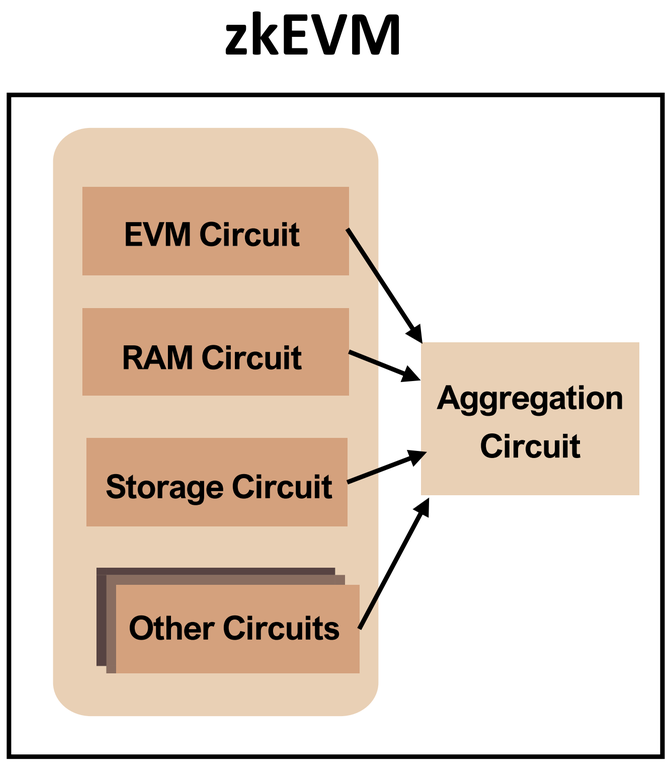
The architecture of zkEVM circuits



Ref - ZKP MOOC

https://www.youtube.com/watch?v=vuQGdbpDWcs

zkEVM workflow



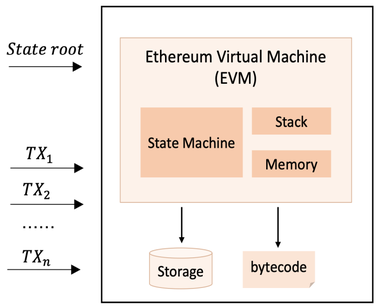
- The first layer needs to be “expressive”
  - EVM circuit has **116 columns, 2496 custom gates, 50 lookups**
  - Highest custom gate degree: 9
  - For 1M gas, EVM circuit needs **2^18 rows** (more gas, more rows)
- The second layer needs to aggregate proofs into one proof
  - Aggregation circuit has **23 columns, 1 custom gate, 7 lookups**
  - Highest custom gate degree: 5
  - For aggregating EVM, RAM, Storage circuits, it needs **2^25 rows**

Ref - ZKP MOOC

https://www.youtube.com/watch?v=vuQGdbpDWcs

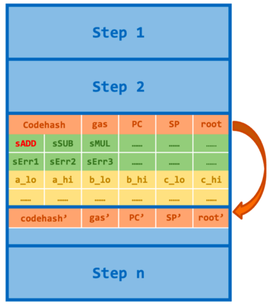
zkEVM workflow

Program



R1CS
Plonkish
AIR

Constraints



- **Step context**

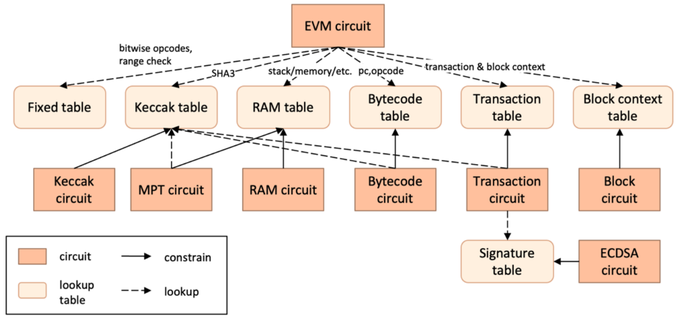
```
sADD * (pc' - pc - 1) == 0
sADD * (sp' - sp - 1) == 0
sADD * (gas' - gas - 3) == 0
```

- **Case switch**

```
sADD * (1 - sADD) == 0
sMUL * (1 - sMUL) == 0
        ...
sADD + sMUL + ... + sERRk == 1
```

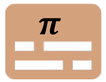
- **Opcode specific witness**

```
sADD*(a_lo+b_lo-c_lo - carry0* 2^128)== 0
sADD*(a_hi+b_hi+carry0-c_hi - carry1*2^128)== 0
```



Proof

Plonk IOP
+
KZG



Ref - ZKP MOOC

https://www.youtube.com/watch?v=vuQGdbpDWcs

zkEVM flavors (by Justin Drake)

- **Language level**
  Transpile an EVM-friendly language (Solidity or Yul) to a SNARK-friendly VM which differs from the EVM. This is the approach of zkSync and Starware.

- **Bytecode level**
  Interprete EVM bytecode directly, though potentially producing different state roots than the EVM, e.g. if certain implementation-level data structures are replaced with SNARK-friendly alternatives. This is the approach taken by Scroll, Hermez, and Consensys.

- **Consensus level**
  Target full equivalence with EVM as used by Ethereum L1 consensus. That is, it proves the validity of L1 Ethereum state roots. This is part of the “zk-SNARK everything” roadmap for Ethereum.

Ref - ZKP MOOC

https://www.youtube.com/watch?v=vuQGdbpDWcs

Questions?

Thank you!