ethereum foundation

appliedzkp.org

Privacy and Scaling Explorations

Explore new use cases for zero-knowledge proofs and other cryptographic primitives through research and proof-of-concept

Semaphore

Prove that you belong to a group without revealing identity

- You also get a “hash” of your {identity || metadata} (nullifier)
- Designed to be simple and generic privacy layer for Ethereum DApps
- Basically is a SNARK that proves a proof-of-inclusion of a Merkle tree
- Implemented in CIRCOM/JS and HALO2/Rust

MACI

Secure voting system where you can replace your vote any time.

- Trusted coordinator
  - Can know the final vote
  - Is not able to censor, forge a vote or impersonate a voter
  - Is not able to produce false tally of votes
- Supports quadratic voting
- No identity system, uses ethereum addresses
- Basically is a SNARK that proves inclusion in MT, vote counting and the last valid user “voting public key”
- Implemented using CIRCOM/JS

RLN

Limit anonymous identities spam in a decentralized environment

- Anonymous identities should belong to a known set
- Basically each time a message is generated an n/M share of a ephemeral secret key is disclosed
- Implemented using CIRCOM/JS and HALO2/RUST

BLS Wallet

Using aggregatable BLS signatures in ethereum

- Batch transactions in L1 to reduce gas costs
- Aggregate signatures offline to do social recovery
- JS (Aggregation service, web wallet) + SC’s

TLSNotary

Generate a proof that a web server generated a file

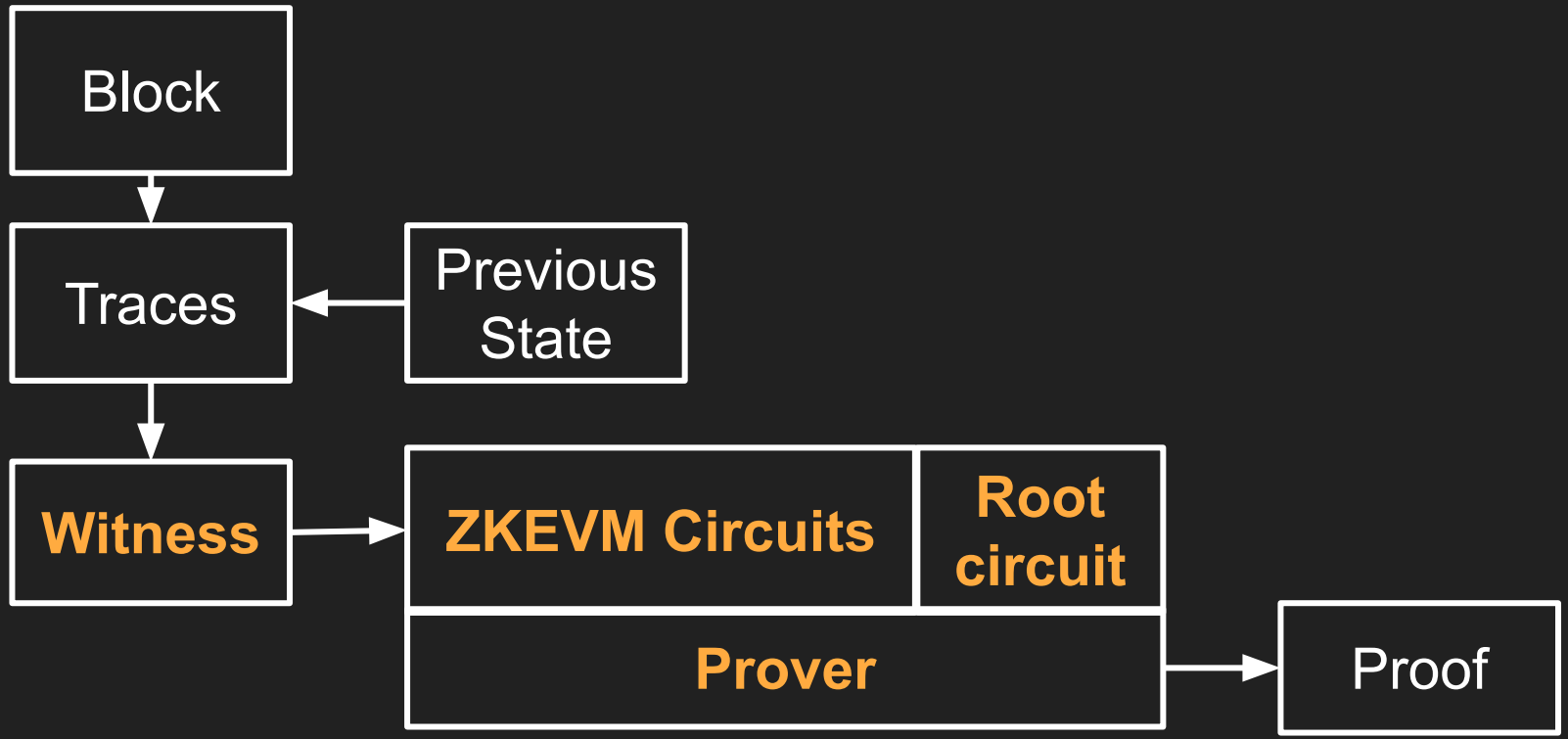
- Gets cryptographic information from the TLS connection.
- The notary service does not know about the content downloaded.
- Browser extension + Developer utilities
- JS + Rust

ZKEVM

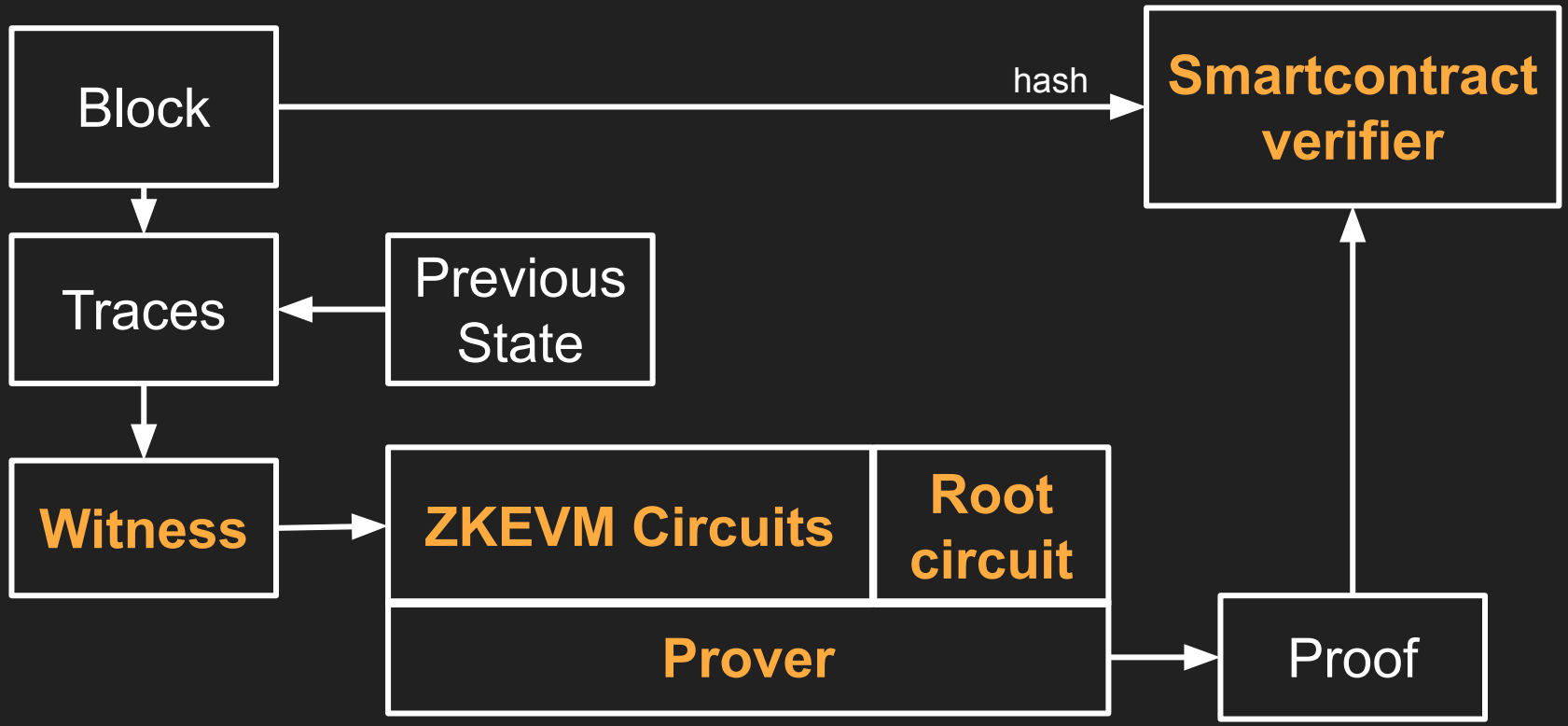
Generate a easy-to-verify proof that a block is correct

- Is NOT zero knowledge
- Obvious use cases are light clients & validity rollups
- PSE goal is to be able to verify all mainnet blocks
- Two big contributors:
  - Taiko (Type1 rollup)
  - Scrolltech (Type2 rollup)
- Let’s drive a little into

ZKEVM


Block
Traces
Previous State
Witness
ZKEVM Circuits
Root circuit
Prover
Proof

ZKEVM


Block
hash
Smartcontract verifier
Traces
Previous State
Witness
ZKEVM Circuits
Root circuit
Prover
Proof

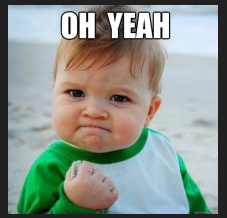
ZKEVM - circuits

Circuits, we use HALO2 arithmetization, that means

- We have to write a block verification only using 3 primitives

- A polynomial of a fixed degree is equal to zero
- A variable belongs to a set of another variables
- A variable is equal to another

- And, in excel-style



ZKEVM - programming model

fib(3)^2 =25

A	B	C	D	E	A[n]= A[n-1]+1	D[n]= B[n]+C[n]	B[n]=C[n-1] C[n]=D[n-1]	X	X^2
1	1	1	2	4	0	1	0	1	1
2	1	2	3	9	1	1	1	2	4
3	2	3	5	25	1	1	1	3	9
.	.	.	.	.	0	0	0	4	16
.	.	.	.	.	0	0	0	5	25

ZKEVM - programming model

fib(**3**)^2 =**25**

A	B	C	D	*E*	*A[n]= A[n-1]+1*	D[n]= B[n]+C[n]	*B[n]=C[n-1] C[n]=D[n-1]*	*X*	*X^2*
1	1	1	2	4	0	1	0	1	1
2	*1*	2	3	9	1	1	1	2	4
3	2	3	5	**25**	1	1	1	3	9
.	.	.	.	.	0	0	0	4	16
.	.	.	.	.	0	0	0	5	25

ZKEVM - programming model

fib(3)^2 =25

A	B	C	D	*E*	***A[n]= A[n-1]+1***	*D[n]= B[n]+C[n]*	*B[n]=C[n-1] C[n]=D[n-1]*	*X*	*X^2*
1	1	1	**2**	4	0	1	0	1	1
2	1	2	**3**	9	**1**	1	1	2	4
3	2	3	**5**	25	1	1	1	3	9
.	.	.	**.**	.	0	0	0	4	16
.	.	.	**.**	.	0	0	0	5	25

ZKEVM - programming model

fib(3)^2 =25

A	B	C	D	*E*	*A[n]= A[n-1]+1*	**D[n]= B[n]+C[n]**	*B[n]=C[n-1] C[n]=D[n-1]*	*X*	*X^2*
1	**1**	**1**	**2**	4	0	**1**	0	1	1
2	1	2	**3**	9	1	1	1	2	4
3	2	3	**5**	25	1	1	1	3	9
.	.	.	**.**	.	0	0	0	4	16
.	.	.	**.**	.	0	0	0	5	25

ZKEVM - programming model

fib(3)^2 =25

A	B	C	D	*E*	*A[n]= A[n-1]+1*	D[n]= B[n]+C[n]	***B[n]=C[n-1] C[n]=D[n-1]***	*X*	*X^2*
1	1	**1**	**2**	4	0	1	0	1	1
2	**1**	**2**	**3**	9	1	1	**1**	2	4
3	2	3	**5**	25	1	1	1	3	9
.	.	.	.	.	0	0	0	4	16
.	.	.	.	.	0	0	0	5	25

ZKEVM - programming model

fib(3)^2 =25

A	B	C	D	*E*	*A[n]= A[n-1]+1*	D[n]= B[n]+C[n]	*B[n]=C[n-1] C[n]=D[n-1]*	***X***	***X^2***
1	1	1	2	4	0	1	0	1	1
2	1	2	3	9	1	1	1	2	4
3	2	3	**5**	**25**	1	1	1	3	9
.	.	.	.	.	0	0	0	4	16
.	.	.	.	.	0	0	0	**5**	**25**

ZKEVM - programming model

fib(**3**)^2 =**25**

A	B	C	D	*E*	*A[n]= A[n-1]+1*	D[n]= B[n]+C[n]	*B[n]=C[n-1] C[n]=D[n-1]*	*X*	*X^2*
1	1	1	2	4	0	1	0	1	1
2	*1*	2	3	9	1	1	1	2	4
3	2	3	5	**25**	1	1	1	3	9
.	.	.	.	.	0	0	0	4	16
.	.	.	.	.	0	0	0	5	25

ZKEVM - circuits

Current Pros/cons

- ↑ Code has no “unknown” side effects
- ↓ A lot of code, hard to audit, witness/constrain split

To explore

- Use a simple DSL
- Use micro opcodes strategy
- Use formal verification

ZKEVM - proving system

Modified HALO2

- Use ZKG instead IPA
- Do lookups on dynamic columns
- Generate internal randomness in circuits

Everything in one big circuit (super circuit)

ZKEVM - proving system

Pros/cons

- ↑ It works, and gained attraction from other teams to build projects
- ↓ Takes a lot of memory (TB) and proving time

To explore

- Use HW acceleration
- Use small field size
- Use hyperplonk / bigg lookup tables
- Massive use of recursion

ZKEVM - current state

Status

- Tests: 16% (bumps into 98% when two opcodes implemented)
- Opcodes: 90%
- Errors: 20%
- Pre-compiles: 0%
- Blocks, transactions, keccak, MPT: 80%
- Proving system: ?

ZKEVM - final thoughts

- Insane amount of tricky engineering, but we need MORE, please do a PhD in Engineering+Math if you have some spare time.
- Ethereum fractal scaling + cross L3 bridges + light clients
- Allows ZK proofs of everything that happens in any L1, L2, L3 chain easily
- Not having an unified way to use validity proofs could break a little the EVM ecosystem

- Fully homomorphic crypto enabled on well-defined EVM (kEVM? Consensys?) on formalized zkWASM compiler, 2s proofs on browser allowing ZK on personal chains (SSB, Holochain)

thanks!

adria0