



Hyperledger
FOUNDATION



WORKSHOP

Zero Knowledge Proofs and ZK Programming in Blockchain Application Development



Andras
Szabolcsi



Daniel
Szego



Wednesday, April 24




8AM Pacific

Content



- Zero knowledge basics without math
- Zero knowledge basics with math
- Zero knowledge programming with circom
- Programming examples
- ZK rollup
- Demo
- Q & A



Zero knowledge proof and SNARK programming workshop

Daniel Szego
András Szabolcsi



Zero knowledge proof

"Proof" of a statement

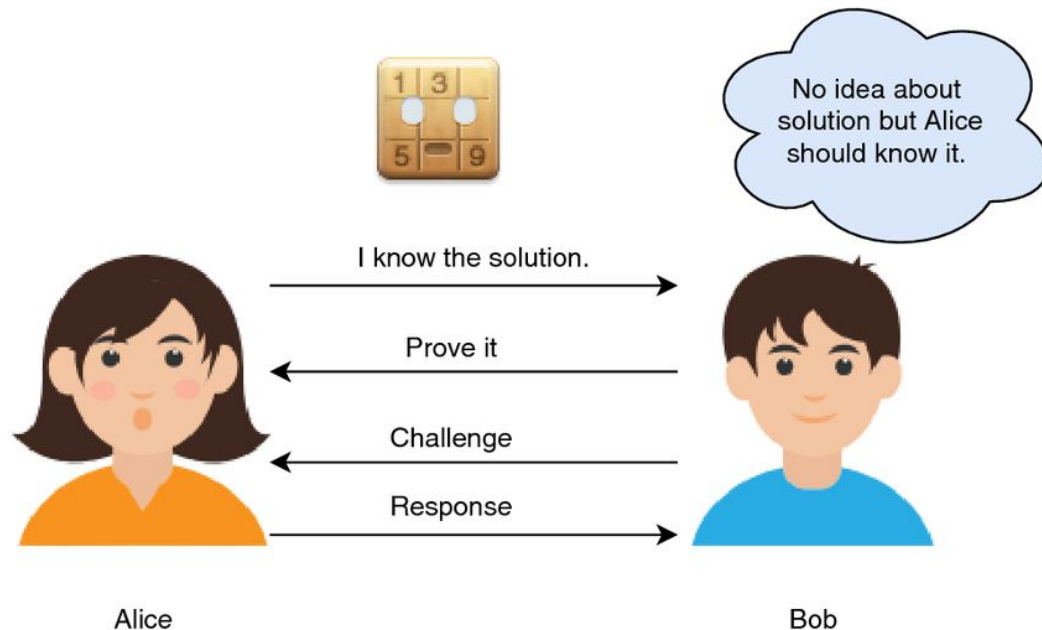
It's not a "classic" mathematical proof, it's stochastic, I know with high probability

I know some kind of secret information, I "prove" that I know without saying it

Roles:

- Prover: prover
- Verifier: verifier, validator

Interactive / non-interactive



Example - Waldo



Example - Cave of Ali Baba

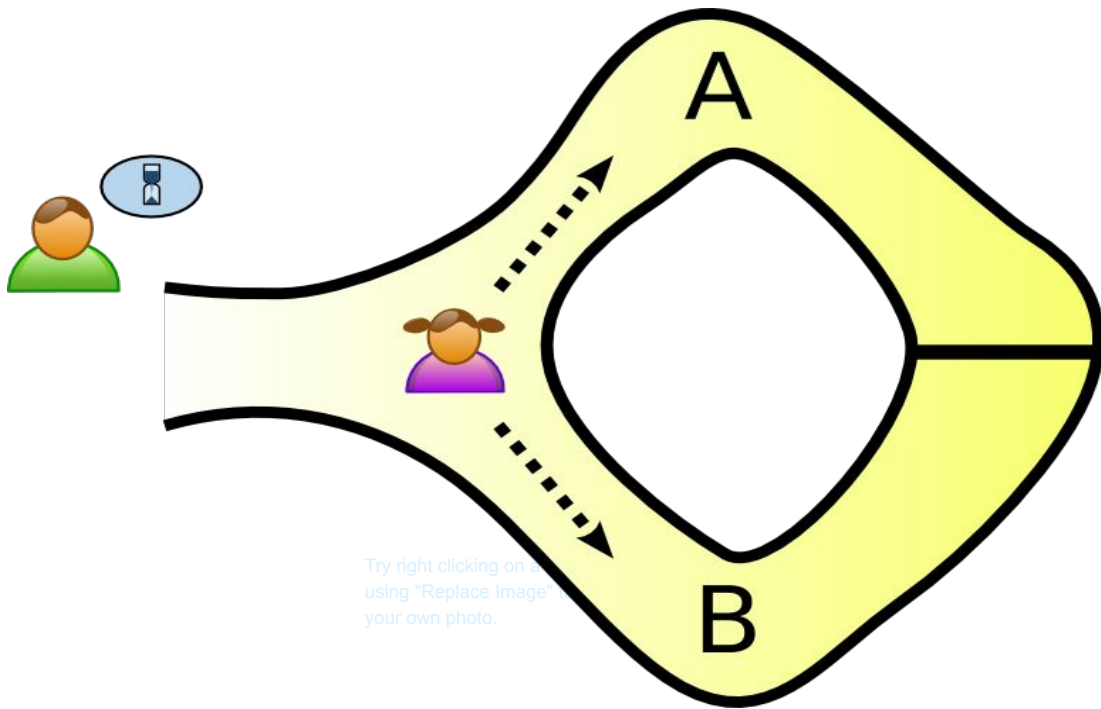
Alice and Bob

Alibaba's cave, which is closed in the middle with a door that opens with a "magic word".

Alice enters the cave and chooses a side to enter.

Bob goes into the cave and decides which side Alice should come back from.

He can cheat 50% of the time -> when repeated many times, he approaches 0%



Try right clicking on a photo
using "Replace Image" to
upload your own photo.

Application - Authenticated but edited photos

Fake pictures, articles, information online.

Camera with digital signature, location and time information.

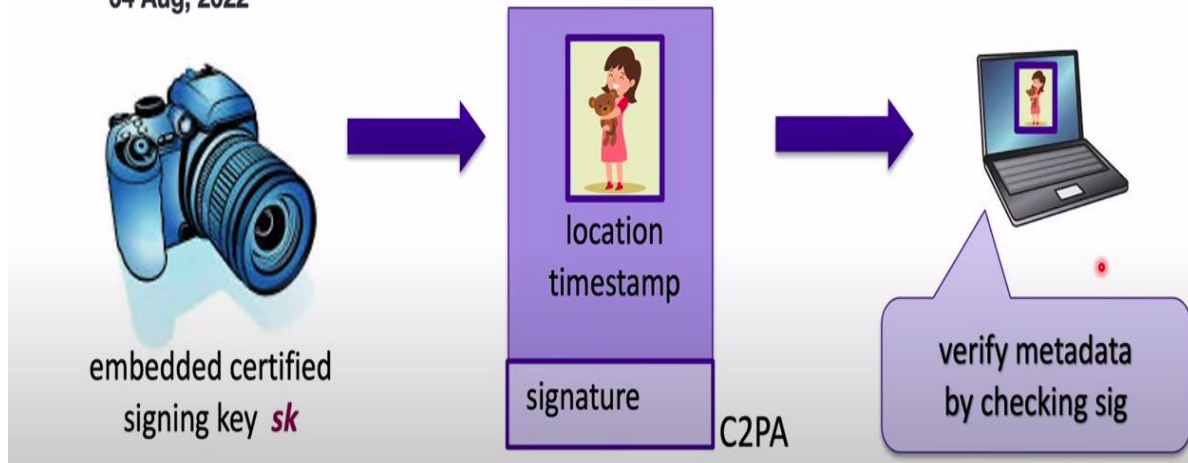
Image transformation, resizing, cropping, sharpening, changing the color palette, etc.

The digital signature is not valid after transformation

Digital signature + transformation + SNARK

Sony Unlocks In-Camera Forgery-Proof Technology

04 Aug, 2022



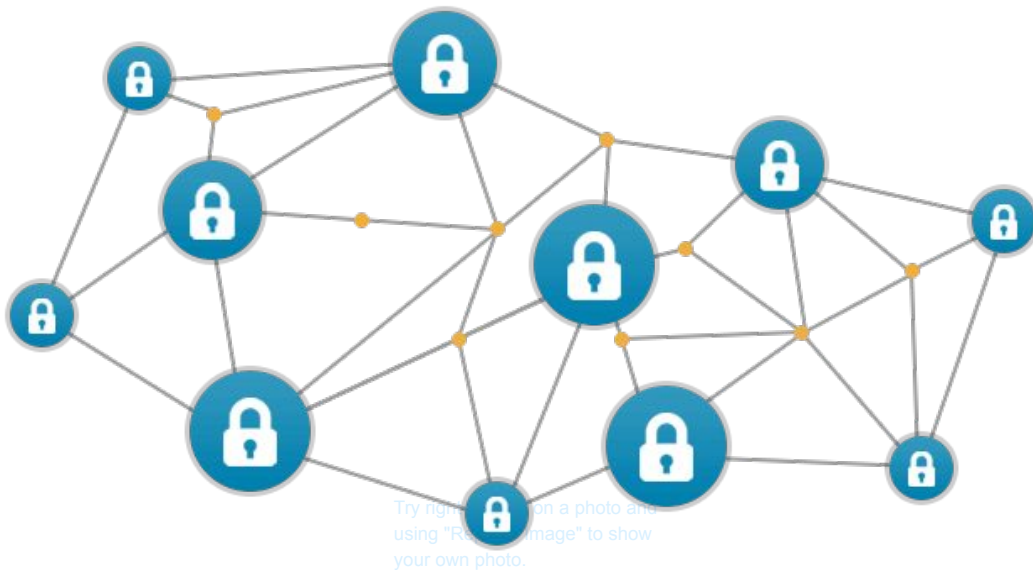
Application - Private blockchain transaction

Blockchain: independent nodes that authenticate transactions.

To authenticate, the node must see the transaction.

Privacy issues.

Instead of full transactions, transactions with secret input and zkSNARK proof.



Application - Rollup

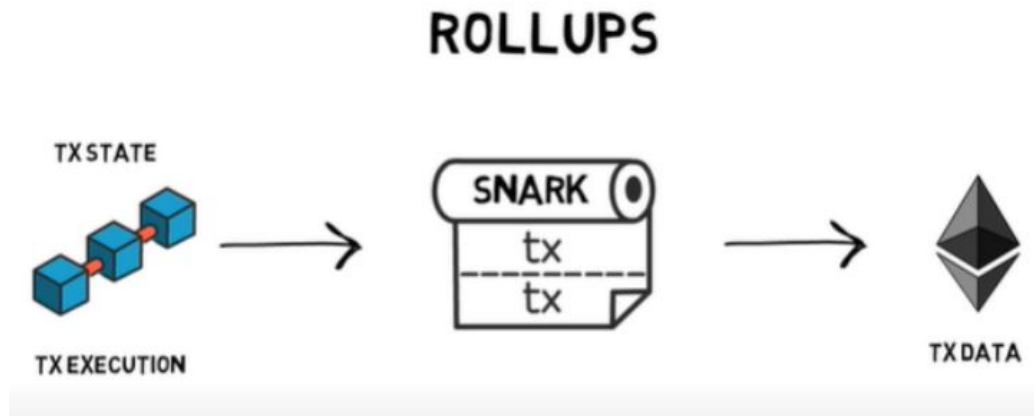
Scalability problem: Most blockchains can perform few transactions and that too slowly.

Reason: independent validators.

Batch execution of transactions outside the blockchain..

The problem is reliability, correct execution, exclusion of fraud.

SNARK / zkSNARK proof of correct execution



your own photo.

Conclusions

An exciting emerging field,

Dynamic development both on the research - theoretical side and on the application side.

Min. 15 years of active development.

Tools and frameworks are also available on the engineering / programming side.

The area can be at least as important as the digital signature.

Zero Knowledge Proofs





Zero knowledge programming



SNARK / zkSNARK

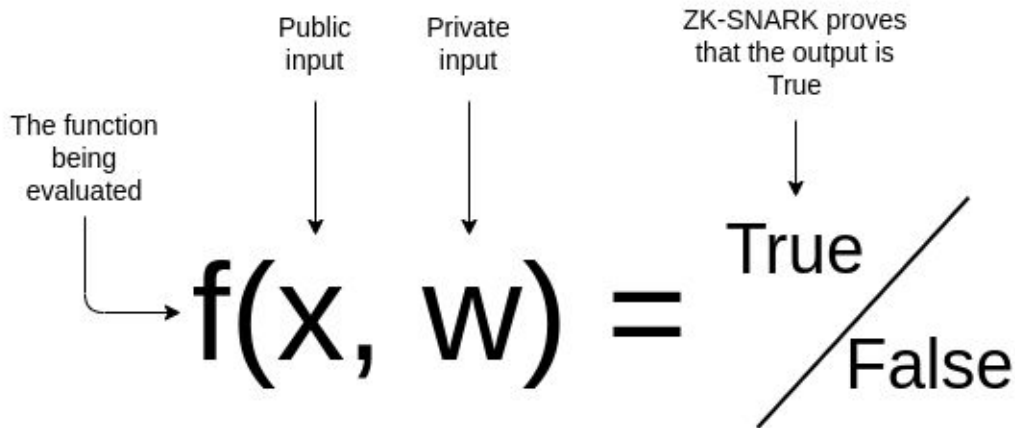
(zk) SNARK

Succinct: short, concise proof

Non-Interactive: there is no interaction, the prover produces it and sends it to the verifier.

Argument of Knowledge: Some information that the prover knows.

Zero-Knowledge: None of the private information reaches the validator.



ZK programming - engineering flow

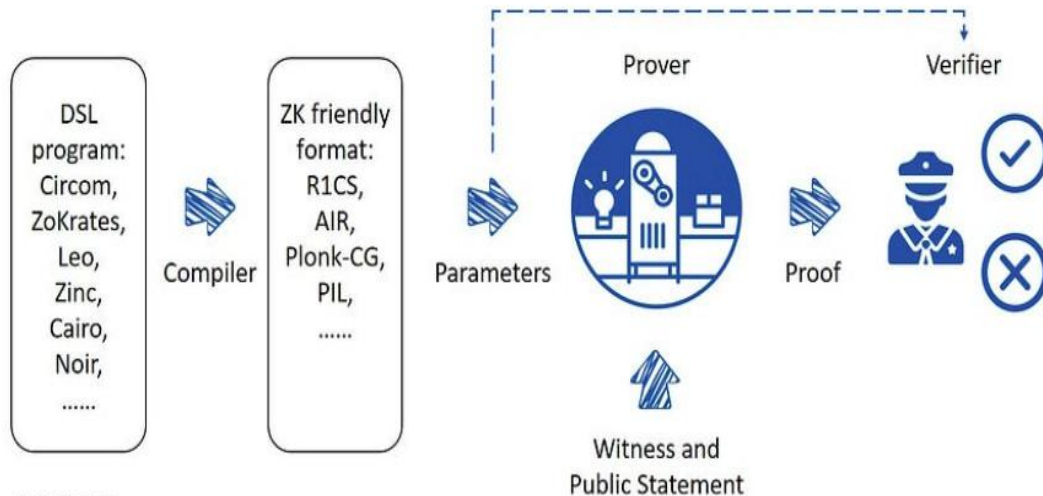
Domain specific languages for SNARK
or zkSNARK programming

Abstracting away some of the
mathematical and theoretical
complexity

ZK and SNARK programming without
cryptographic knowledge ? Not yet :)

Compilation to mathematical
representation, R1CS

Complex development frameworks,
compile, test, prover, verifier module
integrations.



Core algorithm consideration

Under very active development

Proof size

Verification time

Setup :

- per circuit
- universal
- transparent

Post quantum readiness

	size of proof π	verifier time	setup	post-quantum?
Groth'16	≈ 200 Bytes $O_\lambda(1)$	≈ 1.5 ms $O_\lambda(1)$	trusted per circuit	no
Plonk / Marlin	≈ 400 Bytes $O_\lambda(1)$	≈ 3 ms $O_\lambda(1)$	universal trusted setup	no
Bulletproofs	≈ 1.5 KB $O_\lambda(\log C)$	≈ 3 sec $O_\lambda(C)$	transparent	no
STARK	≈ 100 KB $O_\lambda(\log^2 C)$	≈ 10 ms $O_\lambda(\log^2 C)$	transparent	yes

using "Replace Image" to show your own photo.

DSL language and tool selection



Core algorithm consideration






Imperative / description / circuit languages

Different base programming language

Different programming language and framework integration modules

Technological life cycle: all are early stage, but:

- Successful productive usage
- Stable releases

Language	Team
Noir	 Aztec
SnarkyJS	O(1) Labs
Leo	 Aleo
Circom	 iden3
Cairo	 STARKWARE
Lurk	

Circom

DLS / circuit programming language
and development environment for
arithmetic circuits and constraints

Used e.g. in tornado cash

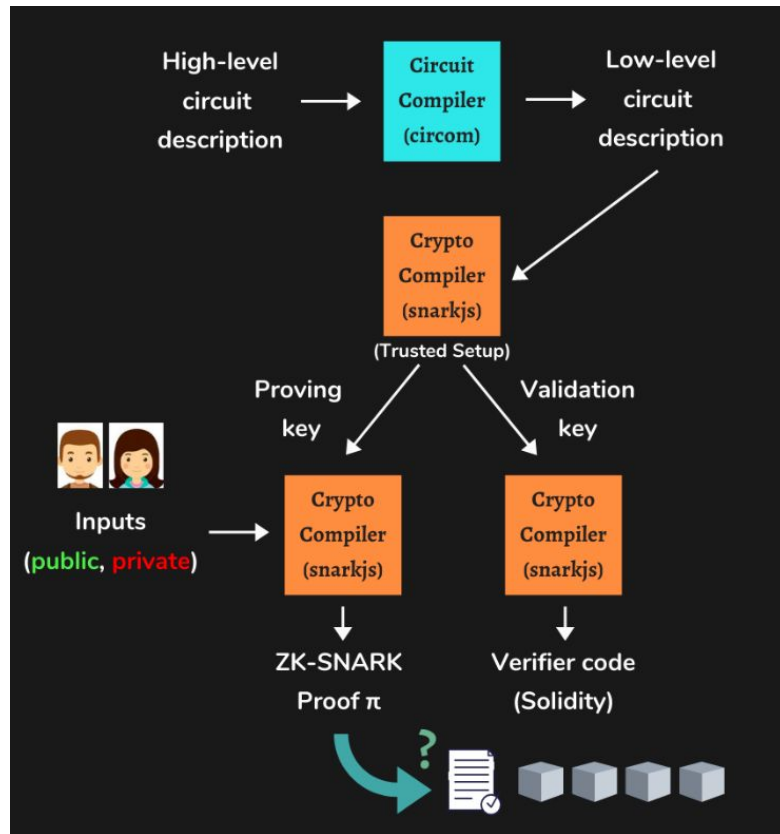
Supports:

- Groth16
- Plonk

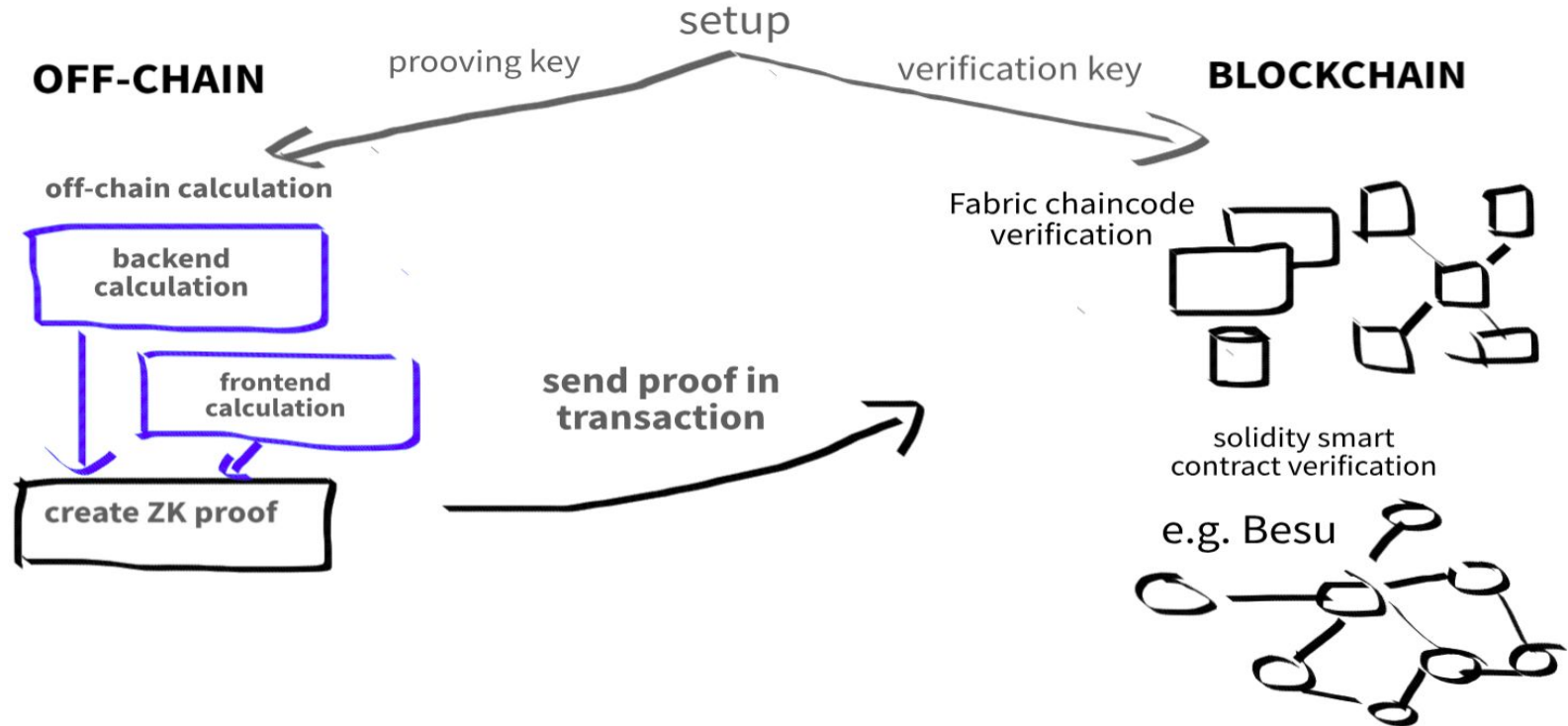
Well established (exist 3 years :)

Supported integration:

- javascript (snarkjs)
- cpp
- solidity verifier



Architectures - demo setups



Power of Tau



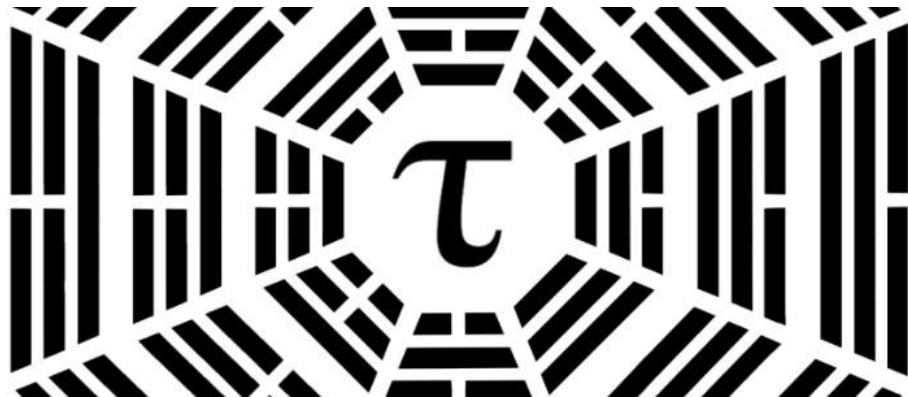
Decentralized setup ceremony

For trusted setup zk protocols

Making trusted setup “trustless” and more “decentralized”

Independent contributors creating a core secret.

If one contributor is “honest”, the result will be true random.



Try right clicking on a photo and using "Replace Image" to show your own photo.

Circom language

Rank 1 constraint / circuit programming language:

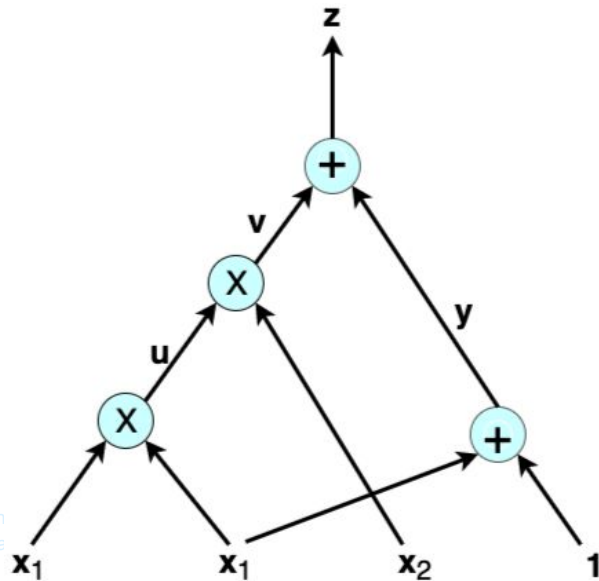
- arithmetic circuits and rank 1 constraints
- $\alpha x + \beta = \gamma$: affine combination of variables

Circuit / Hardware description language (HDL):

- Wires
- Signals
- Gates
- Sub-circuits
- Connecting, wiring circuits, gates

Templates and template programming for subcircuits:

- Programming elements for creating modules, signals
- Variables, control structure over templates and signals
- Templates for higher abstraction
- The output must be always a fix number of circuits, modules



Circom language

Rank 1 constraint / circuit programming language:

$z \leftarrow x * y$: assigning variable to a signal

Assignment is more general - any expression

$z == x * y$: rank 1 constraint among the signals

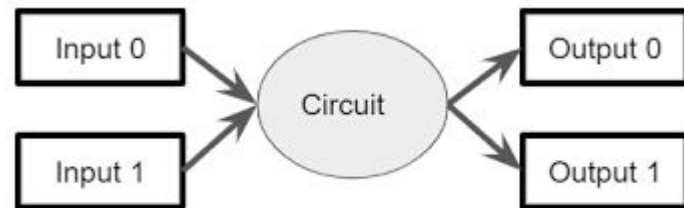
$z \leq x * y$: both assignments and constraints

Circuit / Hardware description language (HDL):

- component
- component main {} : public vs private signals

Metaprogramming:

- Template argument : fix at compiler time !
- Array of signals (fix)
- For loops over array of signals
- fix iteration evaluated at compiler time
- variables : used only at compilation to generate the circuit
- Embedding into another circuit



Try right clicking on a photo and using "Replace Image" to show your own photo.

Circomlib



Predefined and tested circuits:

<https://github.com/iden3/circomlib>

Basic circuits:

- **And, Or, Not, Xor**
- **Multiplexer**
- **Comparison / comparators**

Cryptographic primitives:

- **EdDSA signature**
- **Pedersen commitment**
- **MiMC hash function**
- **Poseidon hash function**
-

iden3/circomlib

Library of basic circuits for circom



 15

Contributors

 3k

Used by

 563

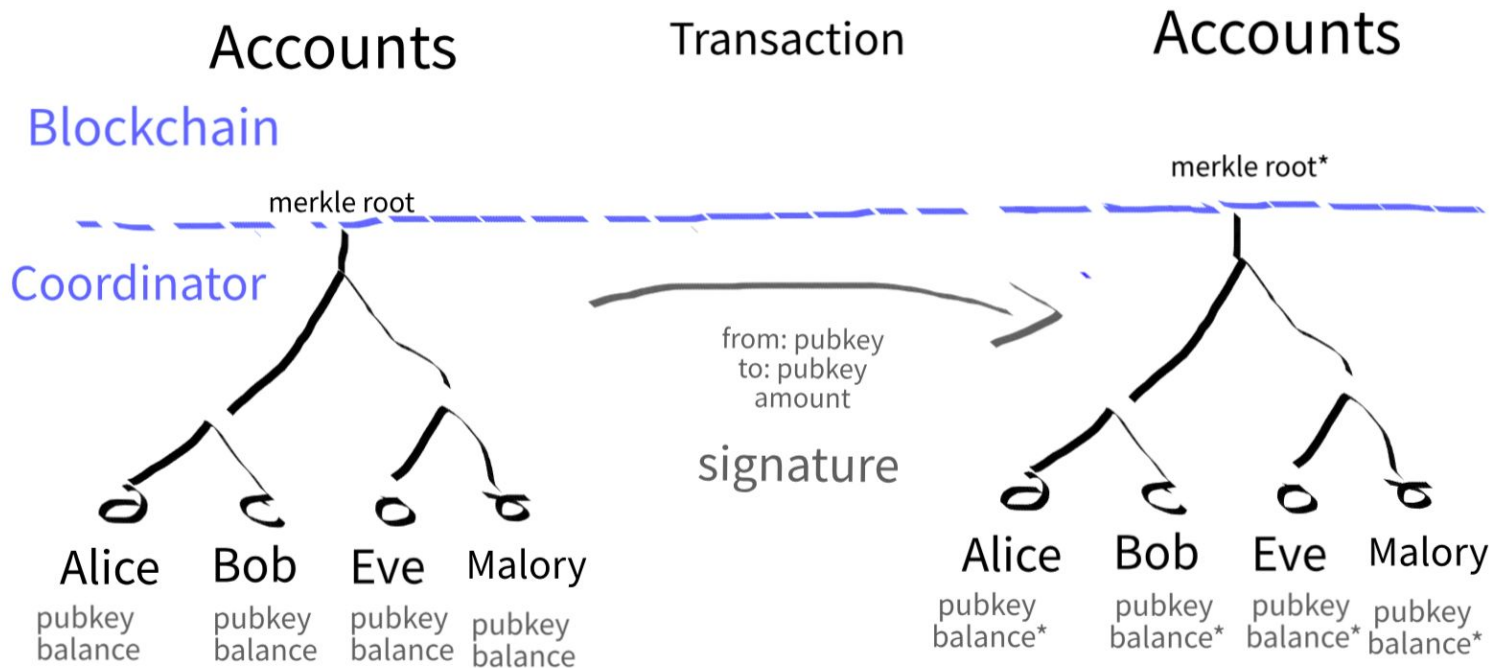
Stars

 203

Forks



Off-chain transaction execution

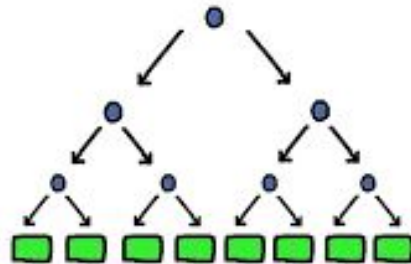


ZK Proof for off-chain execution

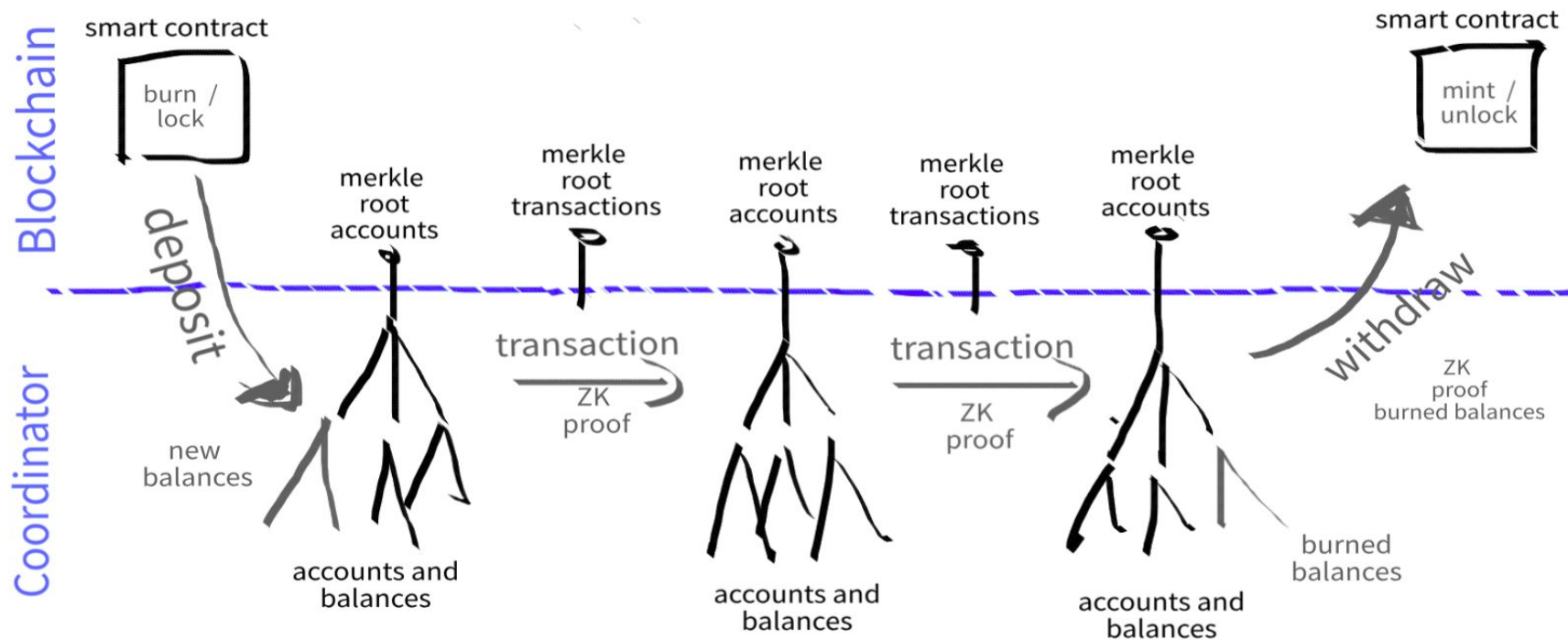
Accounts are represented as leafs in the merkle tree.

Transaction validation steps (simplified):

- Transaction : send *<from>* , *<to>*, *<amount>*
- check if *<from>* and *<to>* are on the account tree, which merkle root is already in the blockchain
- check if *<from>* has enough balance
- debit *<from>* with *<amount>*
- credit *<to>* with the *<amount>*
- calculate new account tree and merkle root for it
- create zk proof about the calculation
- publish new root + zkProof to the ledger
- if the proof is valid, record new root to in the ledger



ZK rollup flow





Q & A

Daniel Szego
András Szabolcsi

