eth warsaw

Introduction to Zero Knowledge

eth warsaw

How it all began

- Fall 2017 - a small after-hours project
- Became spiral of extremely positive events

eth warsaw

How it all began

- EDCON 2017 in Paris
- Founded ETHWORKS
- Unilogin (founded and failed)
- Waffle and useDApp
- Grow Ethworks from 20+ to 40+ in 7 months
- Organised 0xHack with 1000+ participants
- ETHWORKS got acquired
- CTO at TrustToken
- Watch next generation creating EthWarsaw

LinkTime
EDCON
European Ethereum Development Conference
A new world awaits us
17-18 February 2017 Paris, France

I was **lucky**. I **bet boldly**. And I achieved far more than what I **dreamed of**.

So many dreams come true…

- work with great people
- do great engineering work
- take part in impactful projects
- build popular opensource
- And many others…

but most importantly…

- I was **at the frontier**

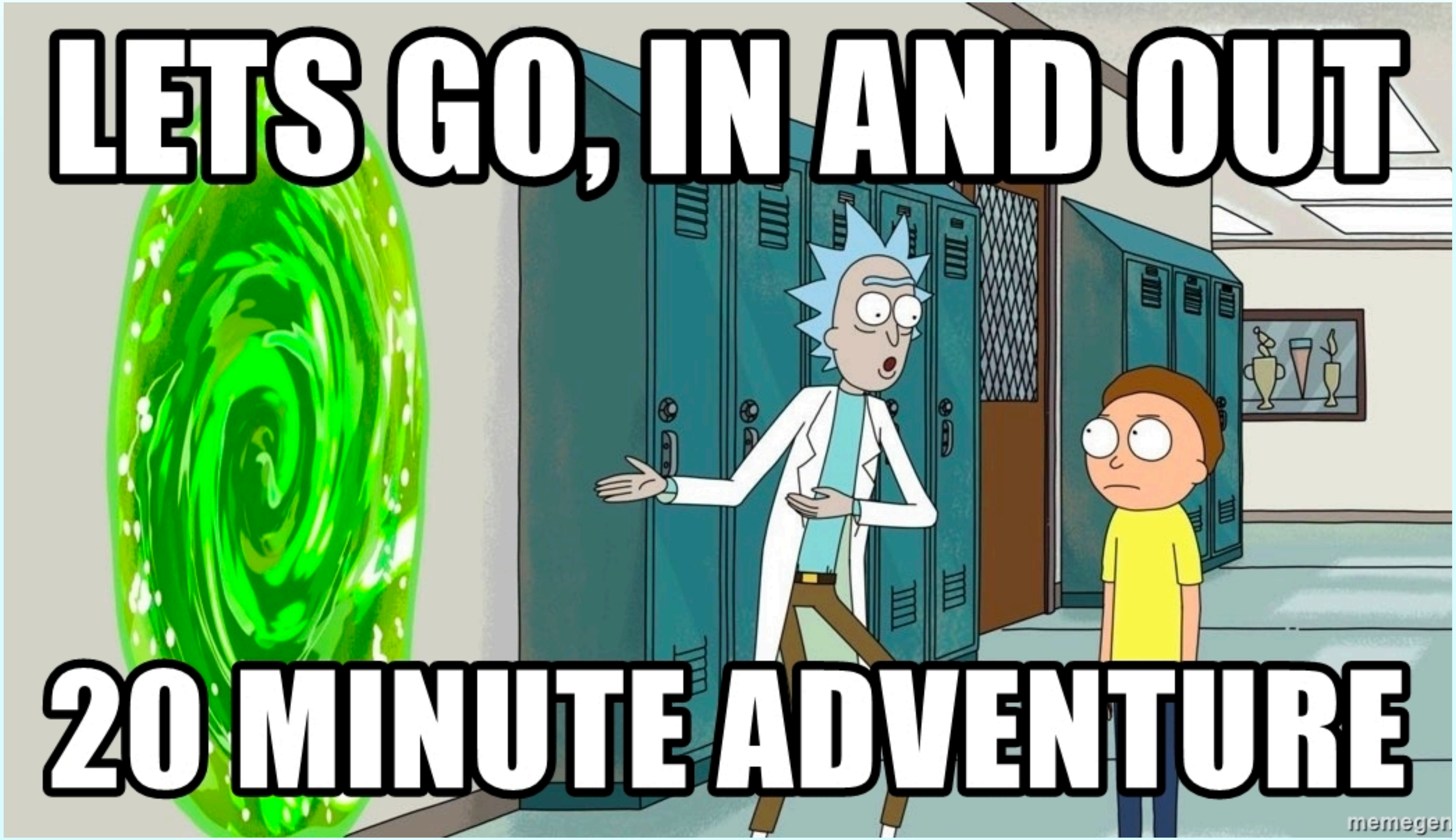
I keep looking for **bold bets** ever since. I am looking to be **at the frontier**.

Call to adventure...

I would like to invite you to **bet big**, to get to **frontiers**.
And today I am sharing with you my **best bet by far**.

eth warsaw

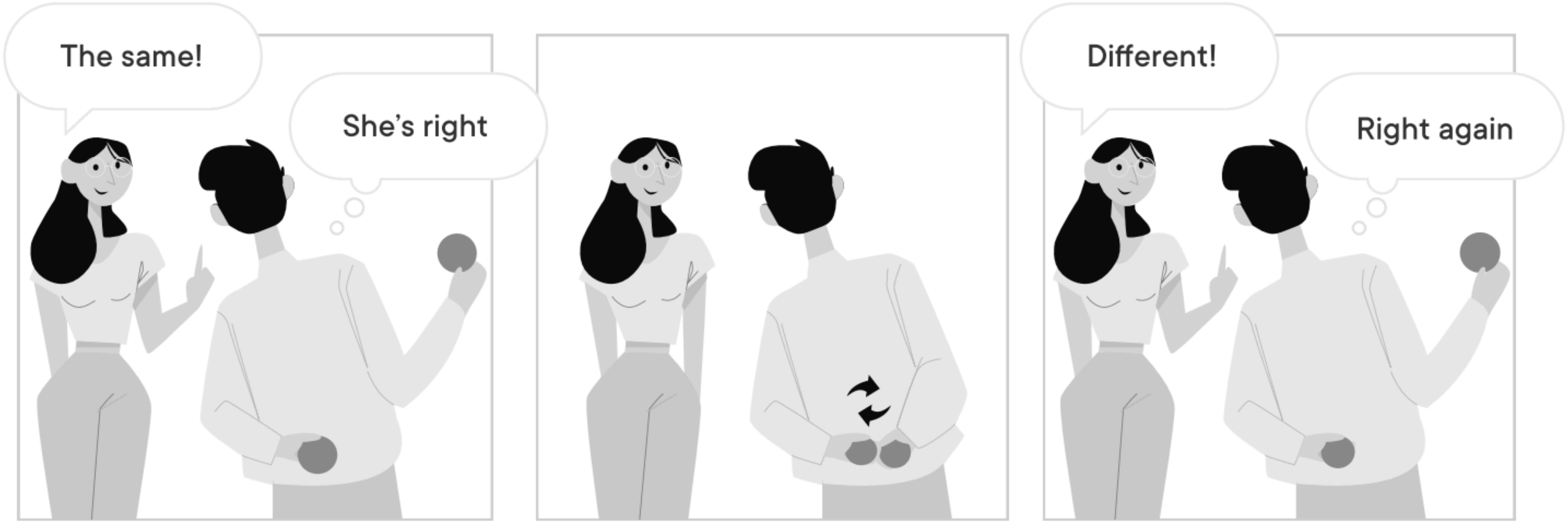
Call to adventure...



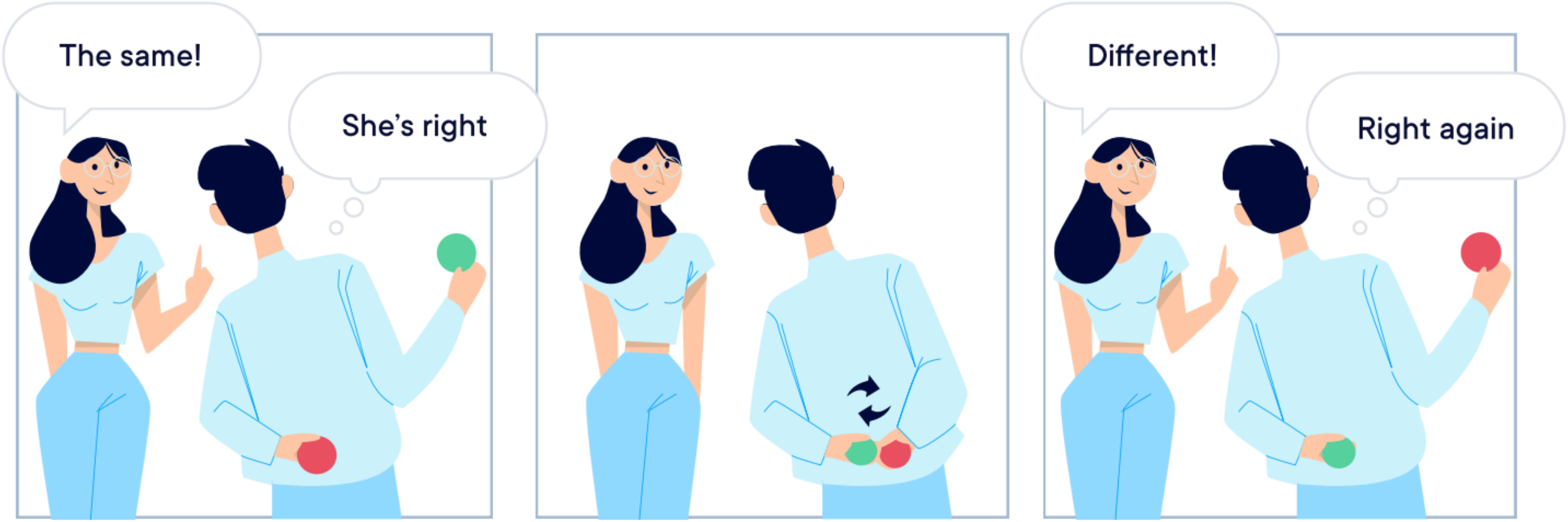
Introduction to Zero Knowledge

Color blind example

What Victor (verifier) sees



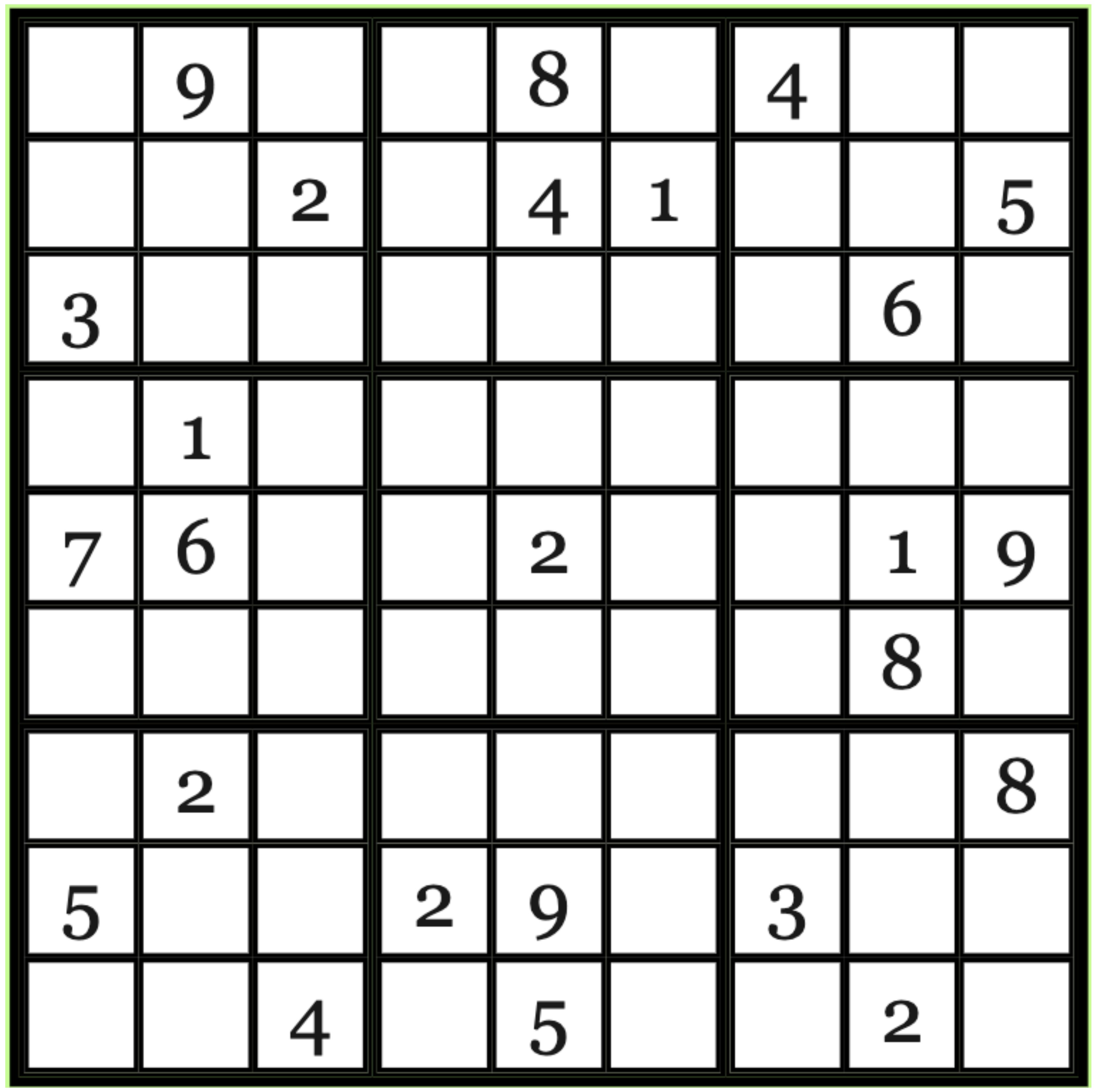
What Peggy (prover) sees



Sudoku example

eth warsaw

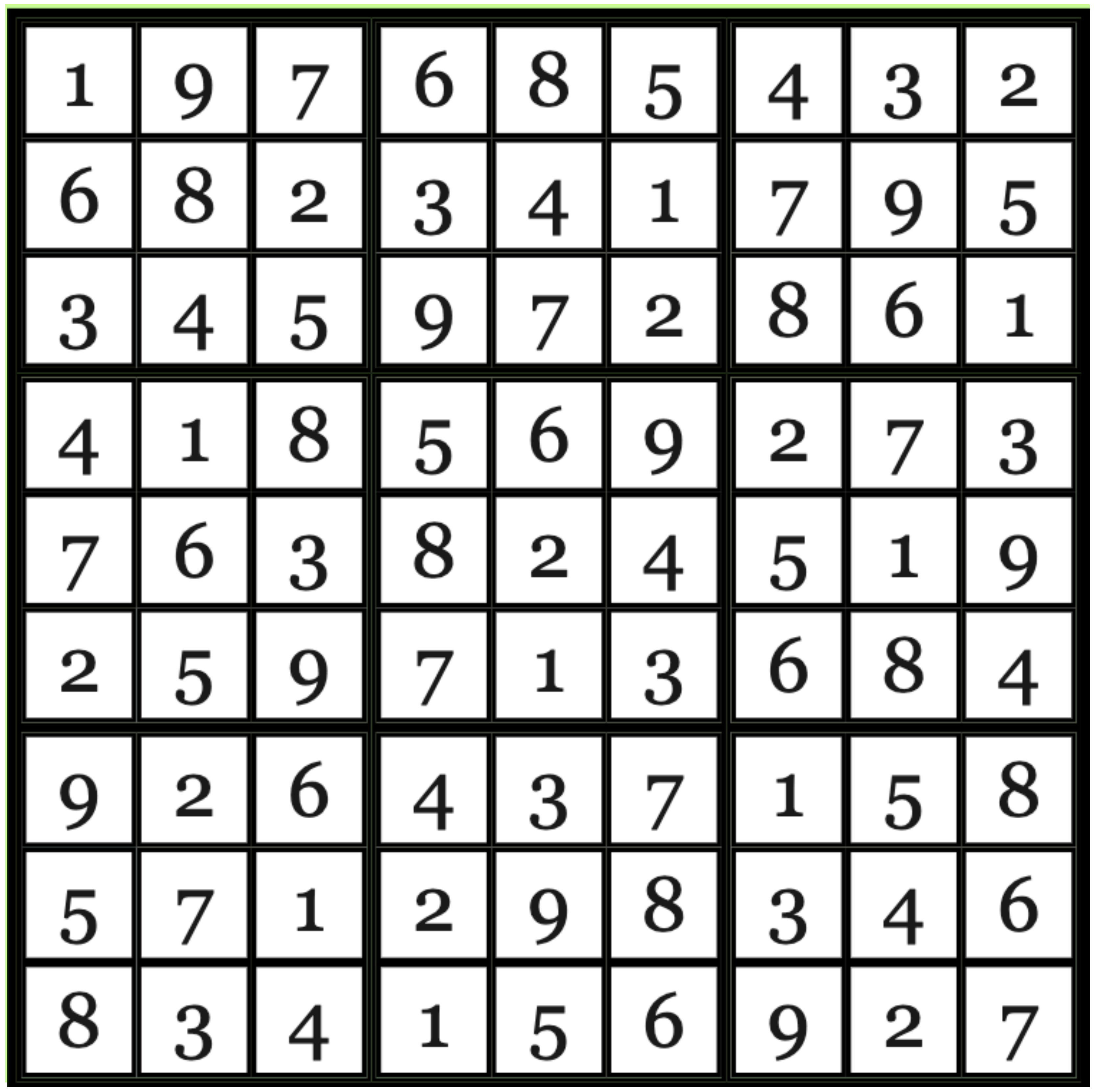
Sudoku



	9			8		4		
		2		4	1			5
3							6	
	1							
7	6			2			1	9
							8	
	2							8
5			2	9		3		
		4		5			2	

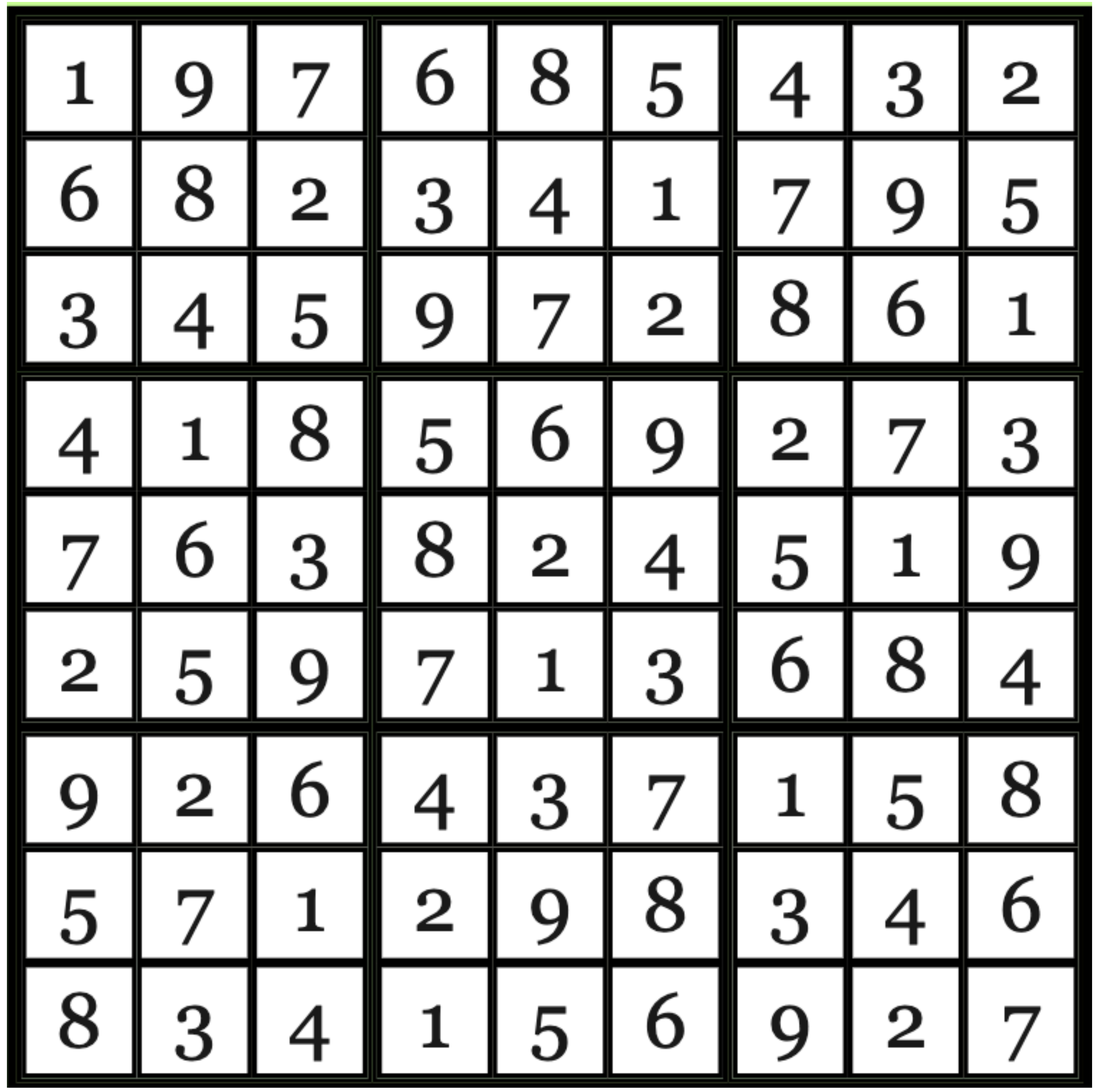
eth warsaw

Sudoku

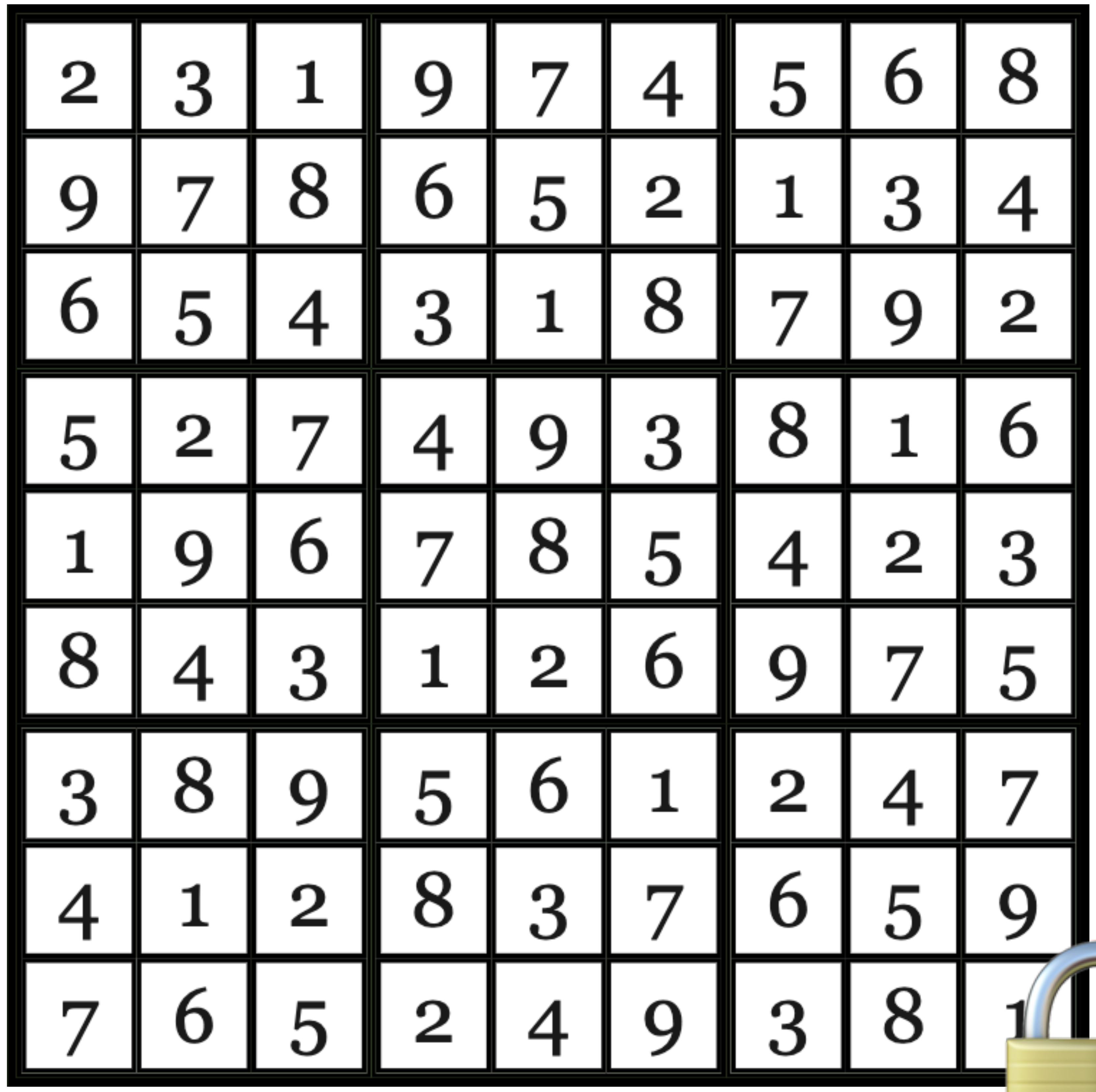


1	9	7	6	8	5	4	3	2
6	8	2	3	4	1	7	9	5
3	4	5	9	7	2	8	6	1
4	1	8	5	6	9	2	7	3
7	6	3	8	2	4	5	1	9
2	5	9	7	1	3	6	8	4
9	2	6	4	3	7	1	5	8
5	7	1	2	9	8	3	4	6
8	3	4	1	5	6	9	2	7

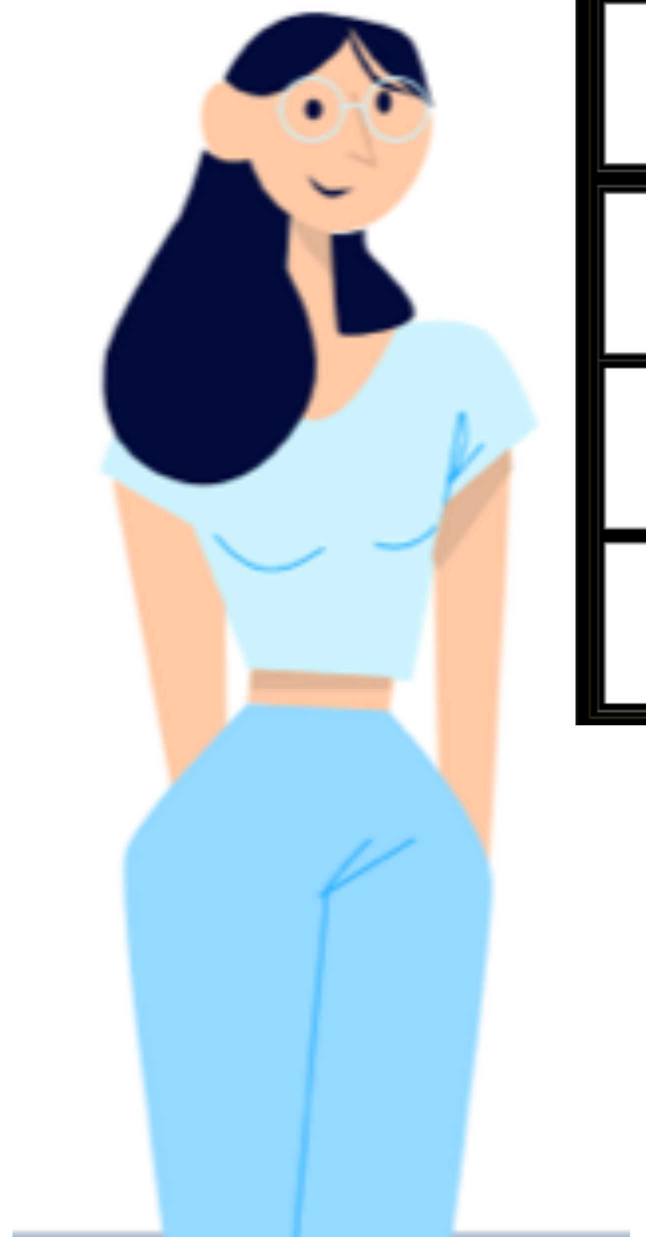
Random permutation



1	9	7	6	8	5	4	3	2
6	8	2	3	4	1	7	9	5
3	4	5	9	7	2	8	6	1
4	1	8	5	6	9	2	7	3
7	6	3	8	2	4	5	1	9
2	5	9	7	1	3	6	8	4
9	2	6	4	3	7	1	5	8
5	7	1	2	9	8	3	4	6
8	3	4	1	5	6	9	2	7



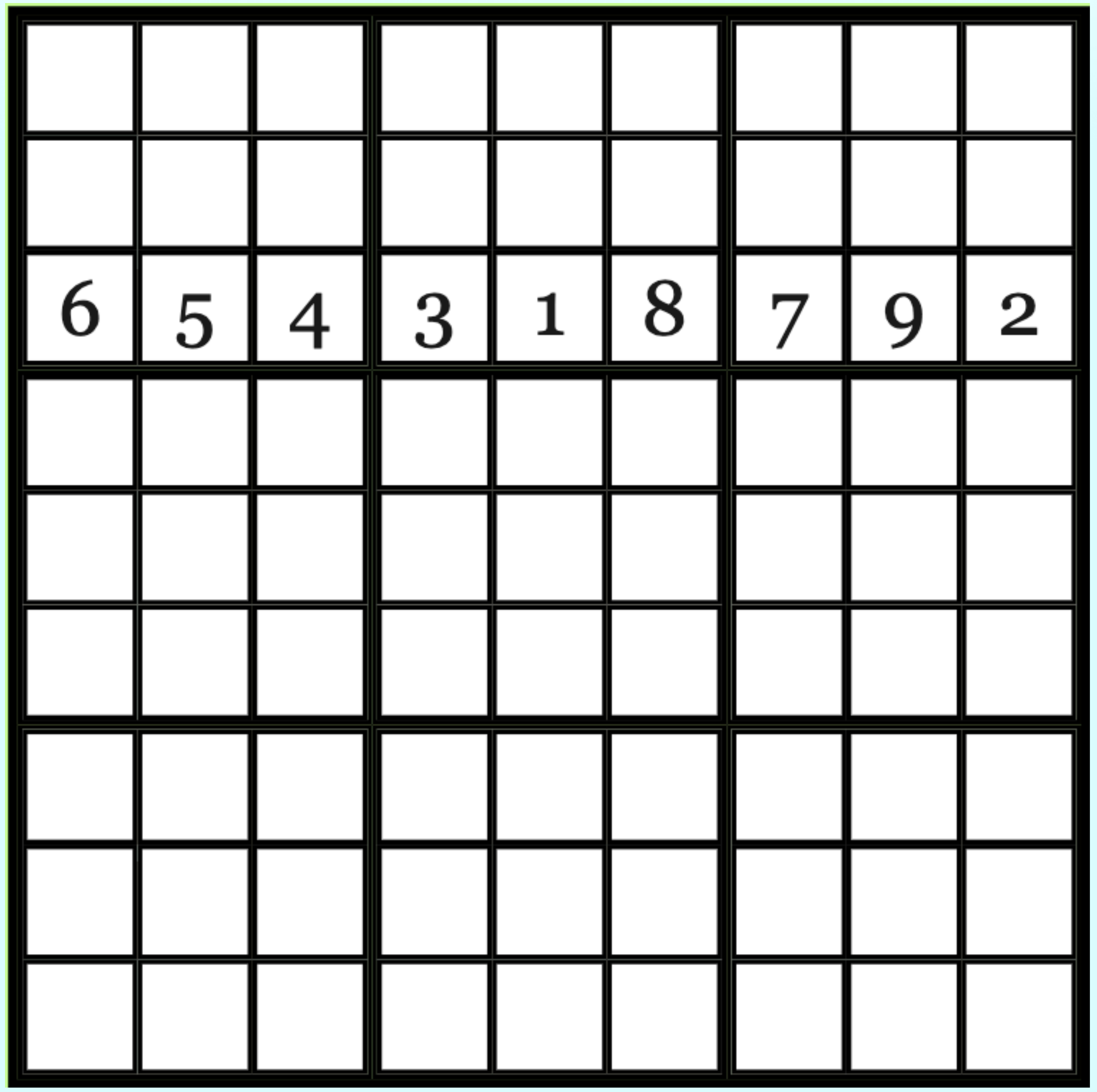
2	3	1	9	7	4	5	6	8
9	7	8	6	5	2	1	3	4
6	5	4	3	1	8	7	9	2
5	2	7	4	9	3	8	1	6
1	9	6	7	8	5	4	2	3
8	4	3	1	2	6	9	7	5
3	8	9	5	6	1	2	4	7
4	1	2	8	3	7	6	5	9
7	6	5	2	4	9	3	8	1



The proof

Victor can make one of 28 choices:

- **Choose one of the rows**
- Choose one of the columns
- Choose one of the sub-boxes
- See the permuted version of the original puzzle



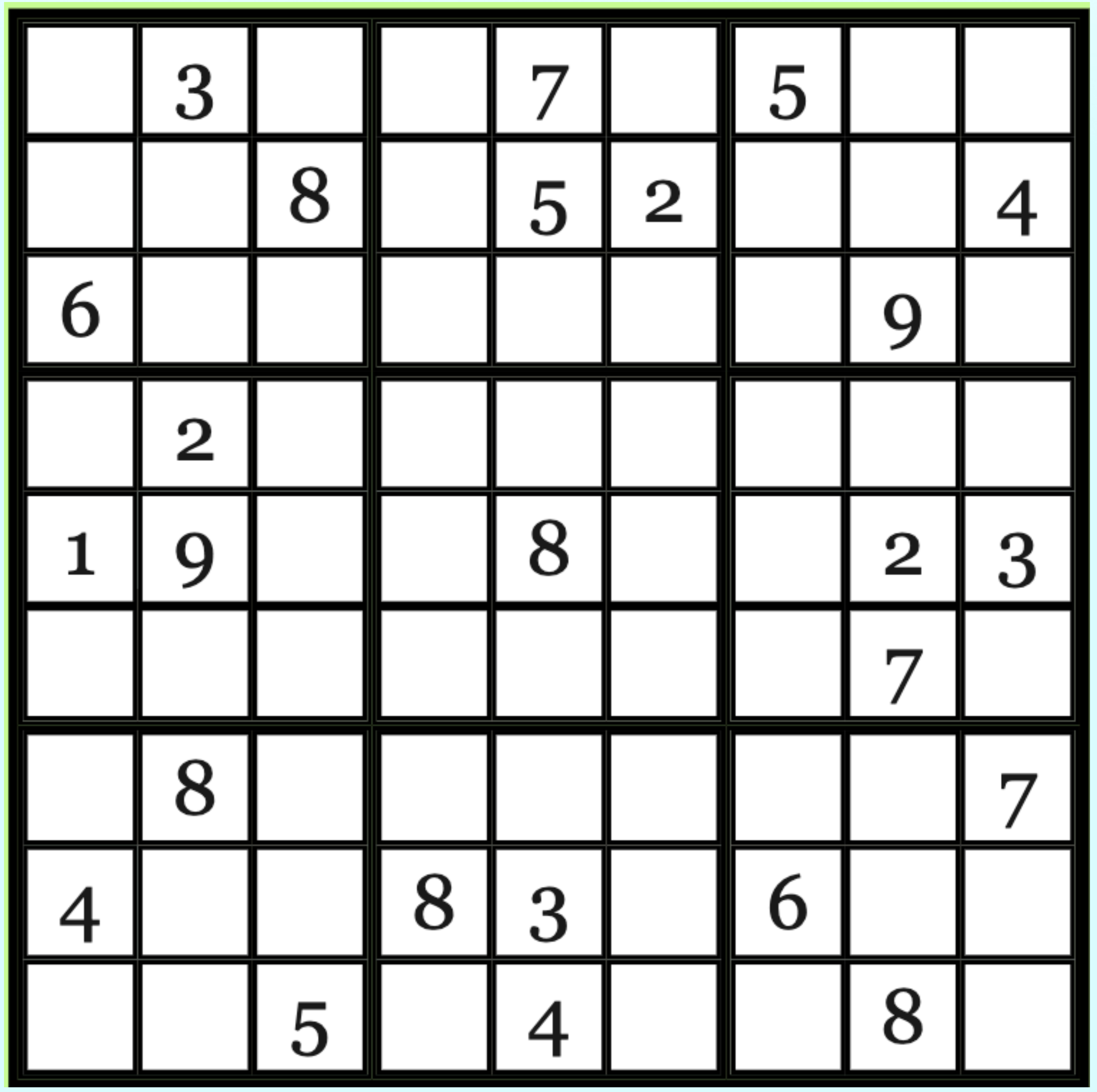
6	5	4	3	1	8	7	9	2

eth warsaw

The proof

Victor can make one of 28 choices:

- Choose one of the rows
- Choose one of the columns
- Choose one of the sub-boxes
- **See the permuted version of the original puzzle**



	3			7		5		
		8		5	2			4
6							9	
	2							
1	9			8			2	3
							7	
	8							7
4			8	3		6		
		5		4			8	

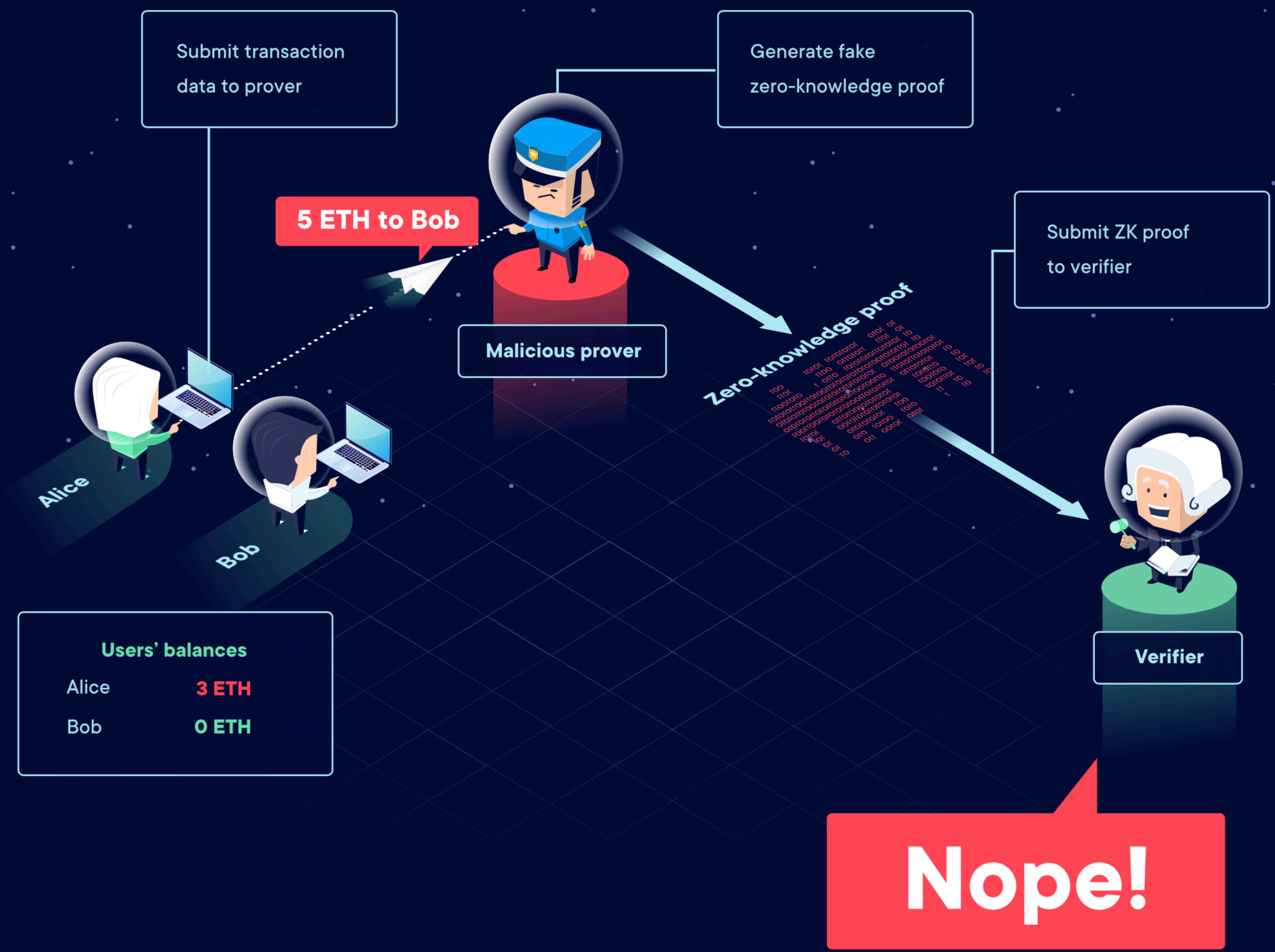
Source: https://blog.computationalcomplexity.org/2006/08/zero-knowledge-sudoku.html

Production

Rules
account balance ≥ 0
tx.in = tx.out
tx signature is valid
Use rules to create ZK circuit
Generate prover and verifier
Rules
Prover
Verifier

Submit transaction data to prover
Generate zero-knowledge proof
2 ETH to Bob
Submit ZK proof to verifier
Ordinary prover
Zero-knowledge proof
Alice
Bob
Verifier
Ok!

Users' balances	
Alice	3 ETH
Bob	0 ETH


Submit transaction data to prover
Generate fake zero-knowledge proof
5 ETH to Bob
Malicious prover
Submit ZK proof to verifier
Zero-knowledge proof
Alice
Bob
Verifier
Users' balances
Alice 3 ETH
Bob 0 ETH
Nope!

Oh right, what we can do with it?

- Verifiable random function
- Verifiable delay function (VDF)
- On-chain mixers
- Privacy oriented cryptocurrencies
- Layer 2
- Compress blockchain: Mina

Verifiable random function (PoS)

RANDOM SERIES: 42, 42, 42, 42

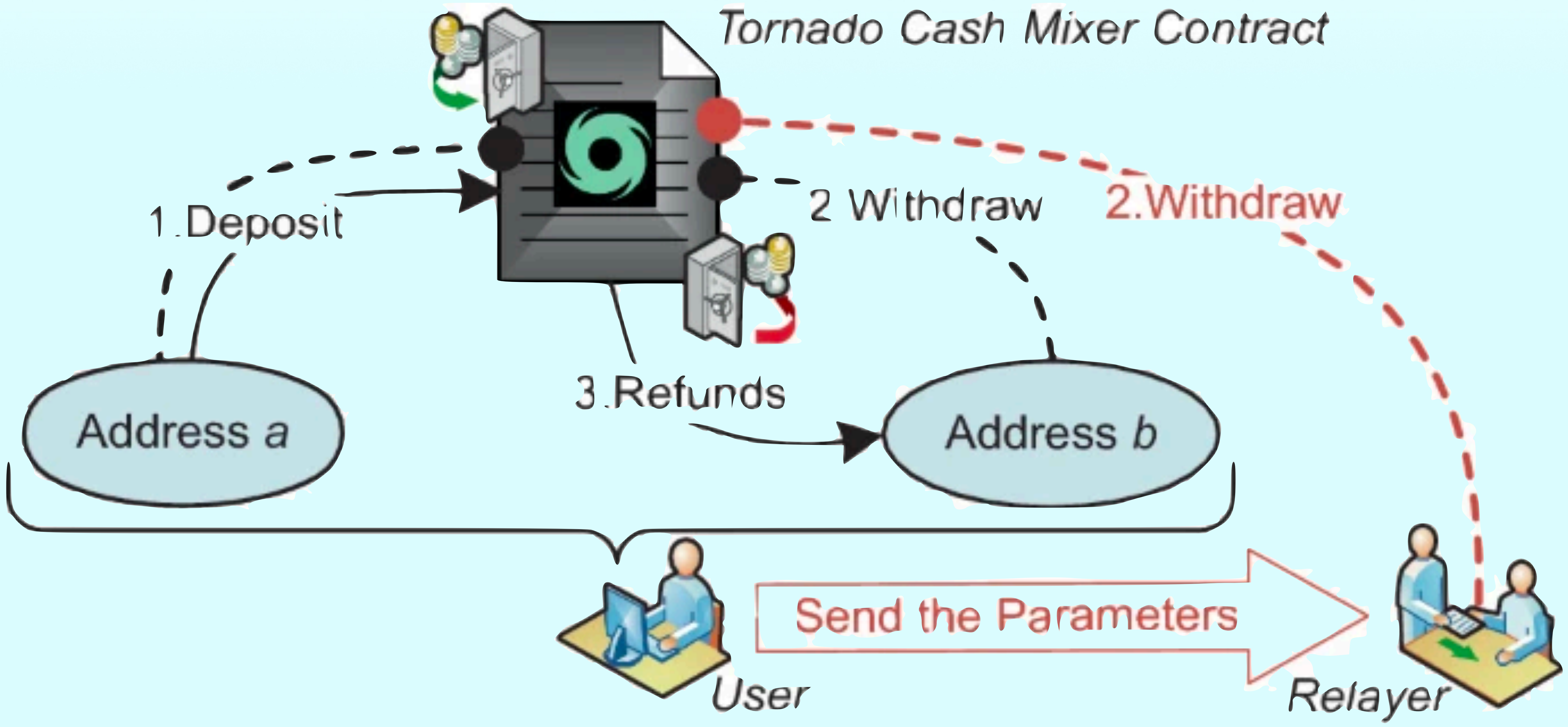
RANDOM(X) = HASH(x)

Verifiable delay function

$$\text{DEALED RANDOM}(x) = \text{HASH}^{\text{BIG}}(x)$$

Tornado Cash

- You can input the deposit from the addr. A
  0.1ETH, 1ETH, 10ETH or 100ETH
- Get a note in return (zk-proof)
- Can forward the not to Relayer who will execute the transaction
- A note will be added to Nullifiers



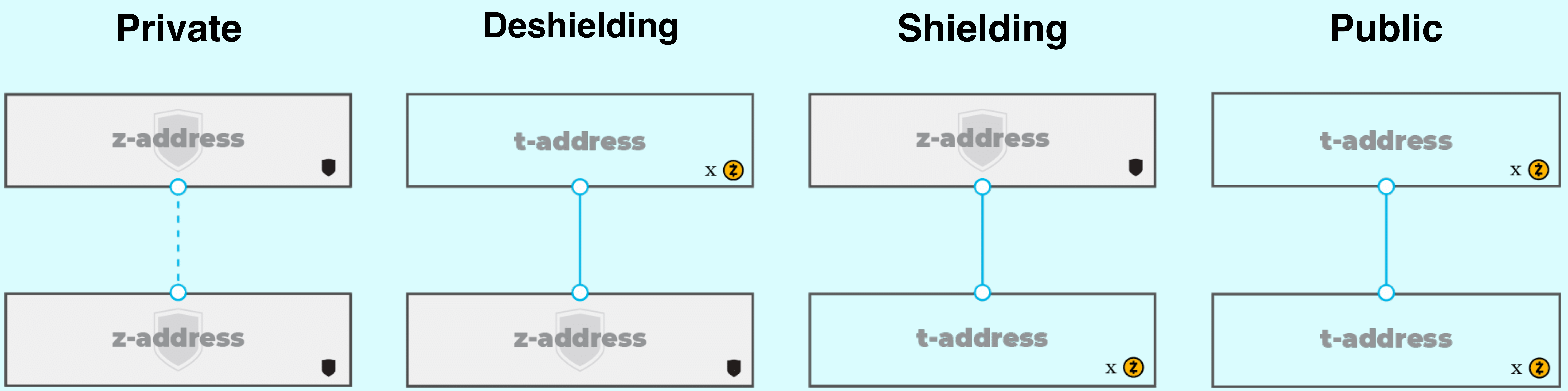
Source: https://blog.computationalcomplexity.org/2006/08/zero-knowledge-sudoku.html

Tornado Cash Nova

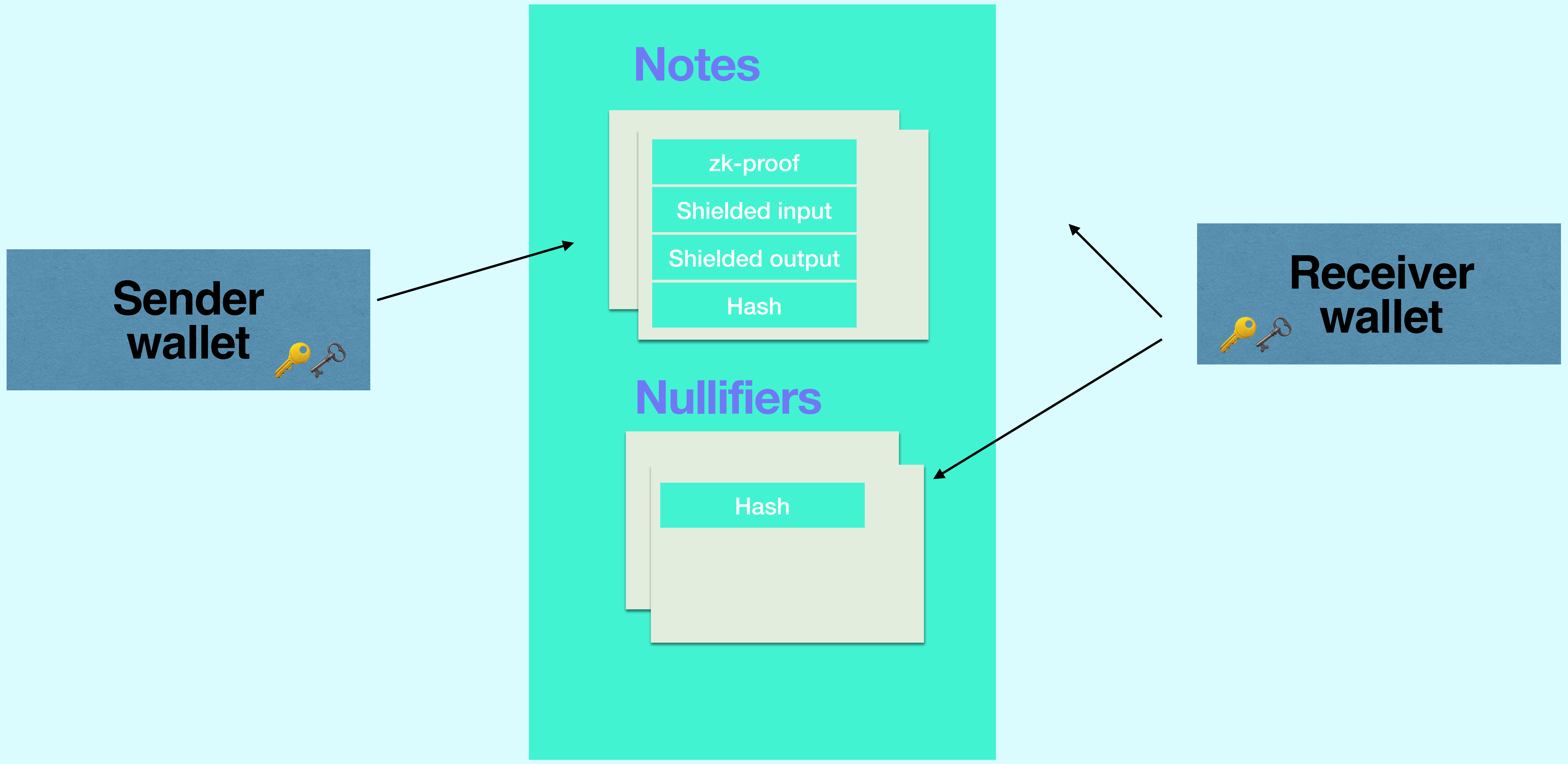
- Support notes splitting
- Briding to Gnosis Chain
- Released in December
- Also coming: Railgun

Source: https://blog.computationalcomplexity.org/2006/08/zero-knowledge-sudoku.html

Zcash


Private
Deshielding
Shielding
Public
z-address
z-address
t-address
z-address
z-address
t-address
t-address
t-address
x
x
x
x
x
x

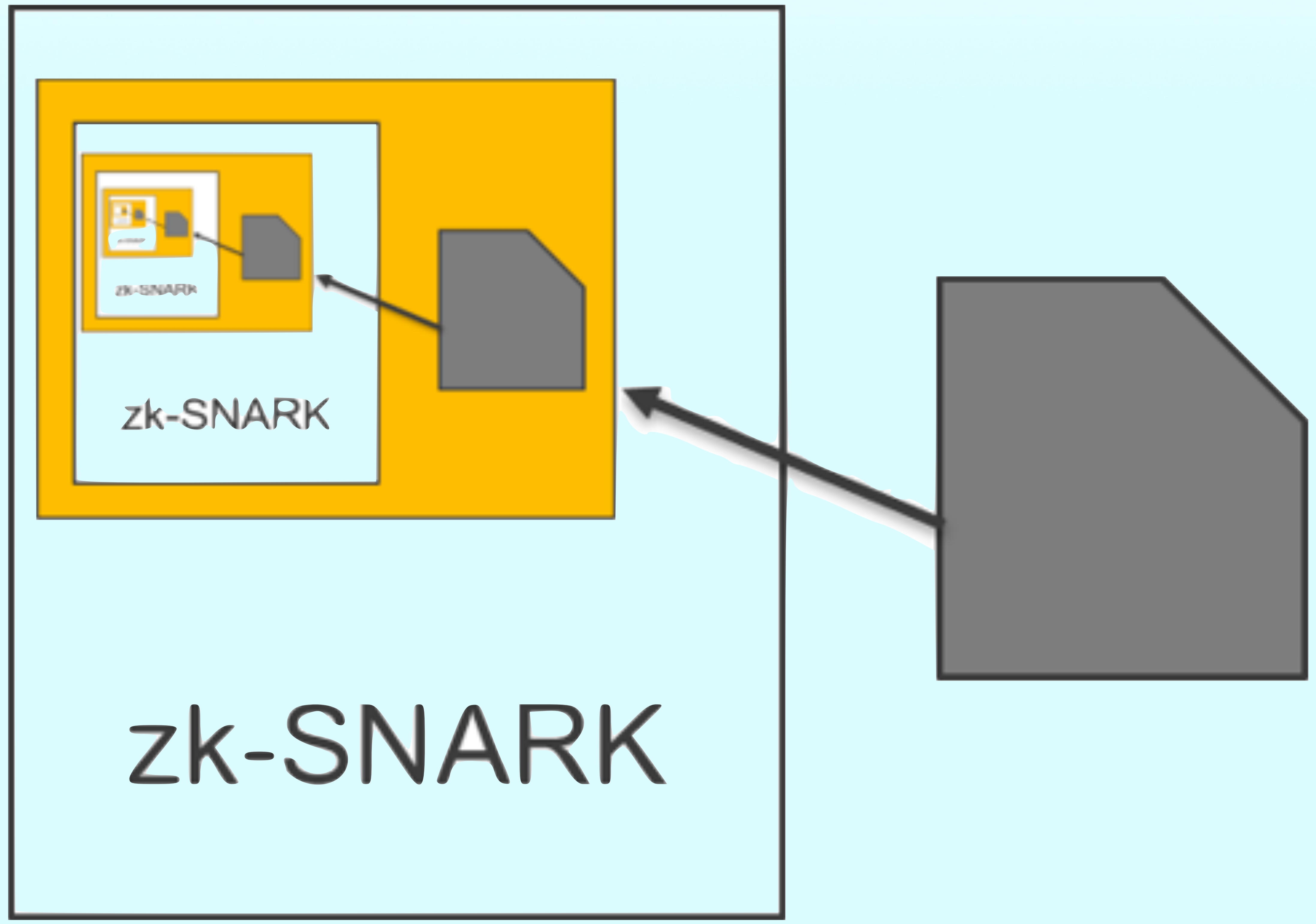
Zcash private transactions



Mina Blockchain

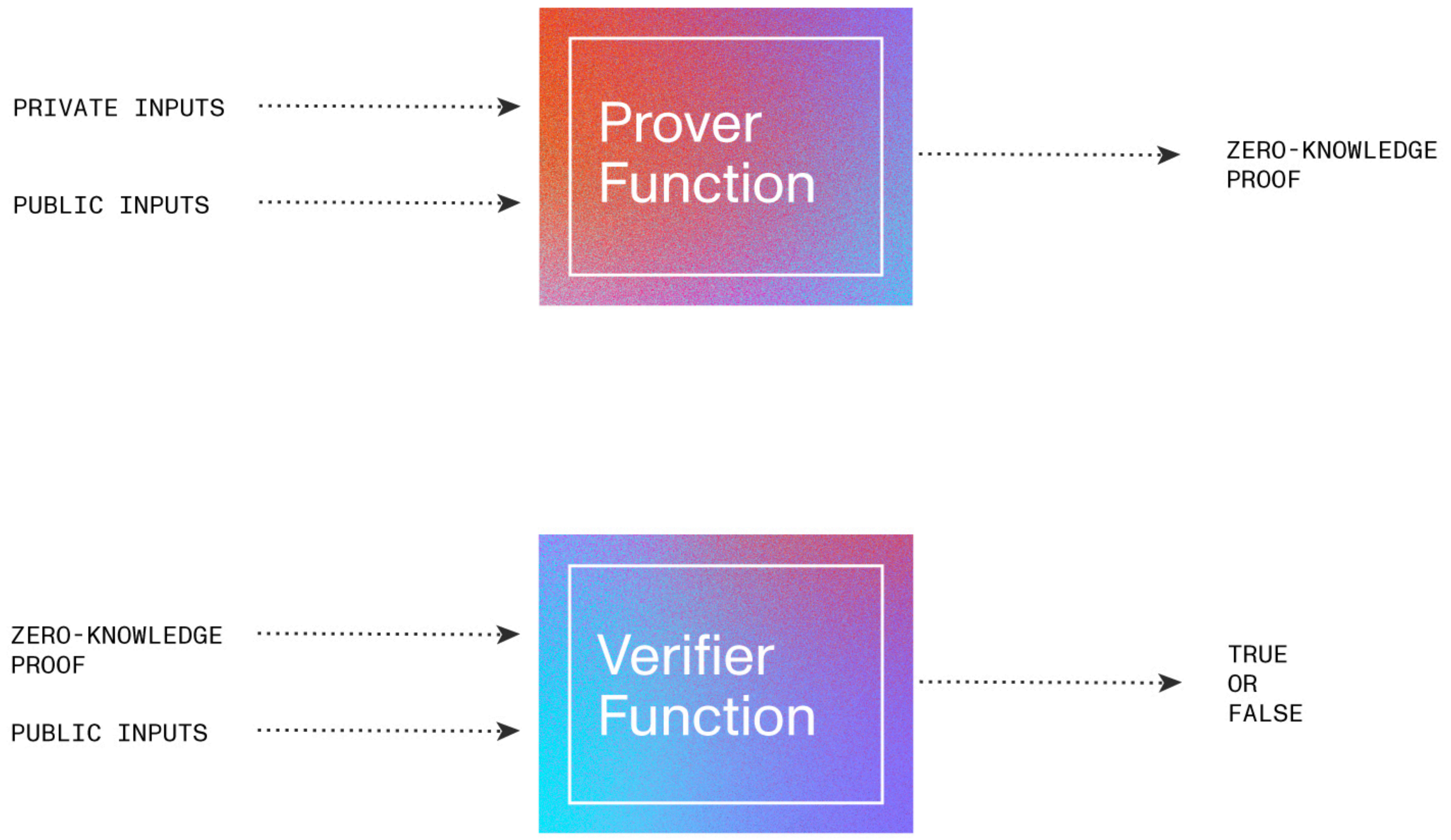
22KB[1]

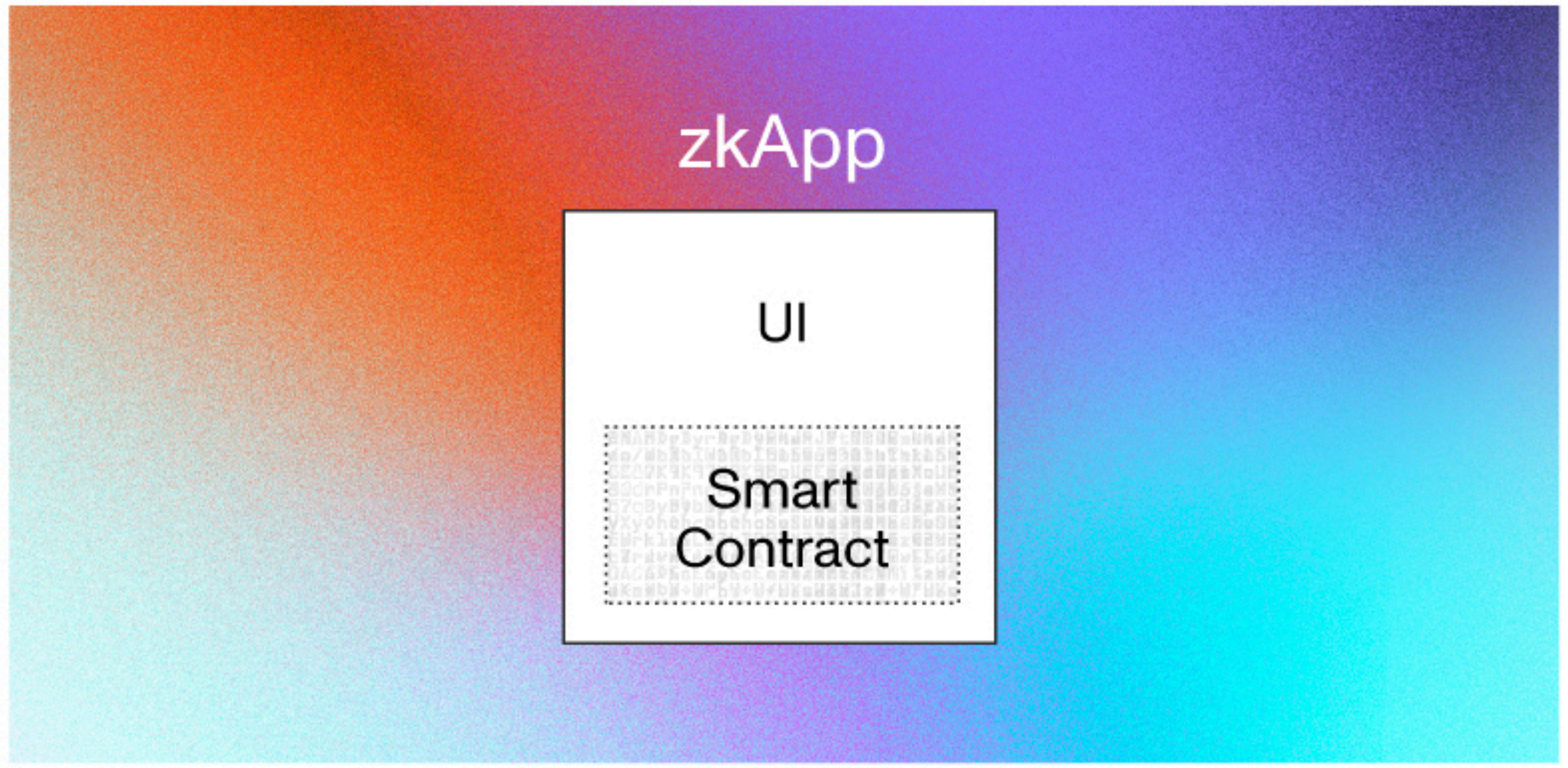
FIXED SIZE

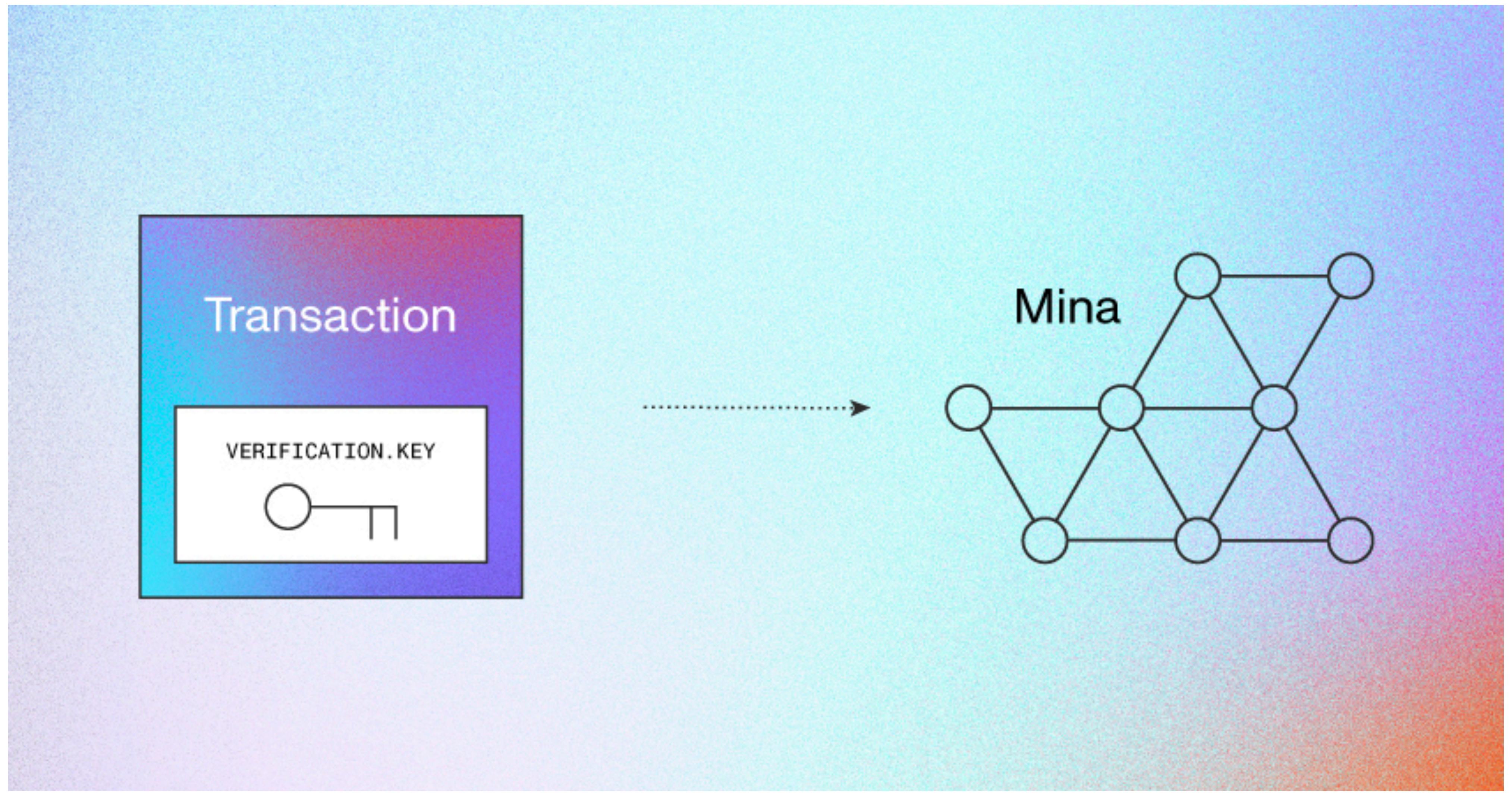


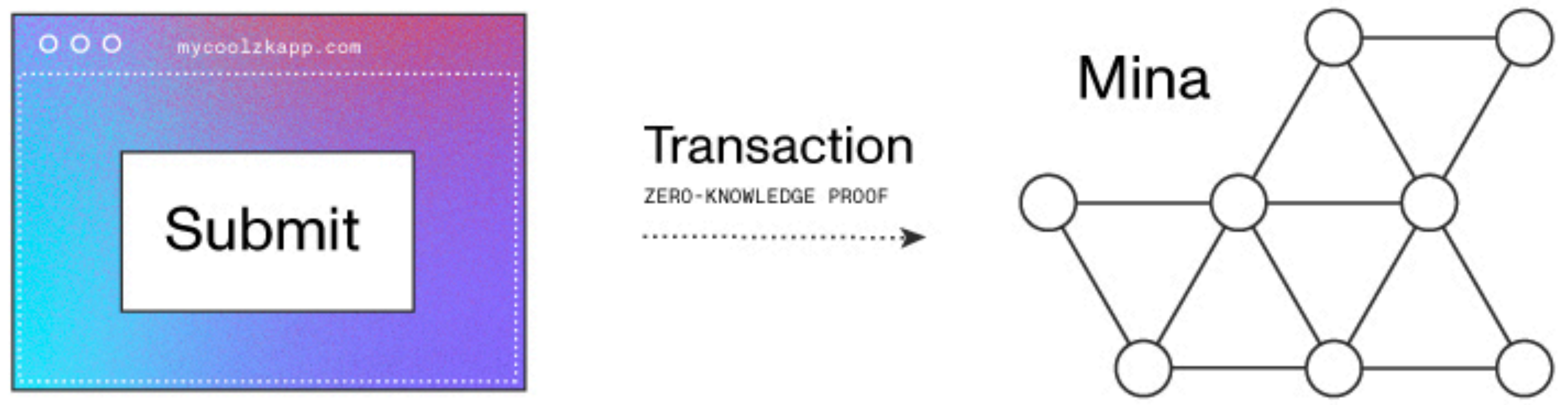
Example zkApp:

https://github.com/o1-labs/zkapp-cli/blob/main/examples/tictactoe/ts/src/index.ts


PRIVATE INPUTS
PUBLIC INPUTS
Prover Function
ZERO-KNOWLEDGE PROOF
ZERO-KNOWLEDGE PROOF
PUBLIC INPUTS
Verifier Function
TRUE OR FALSE


zkApp
UI
Smart Contract

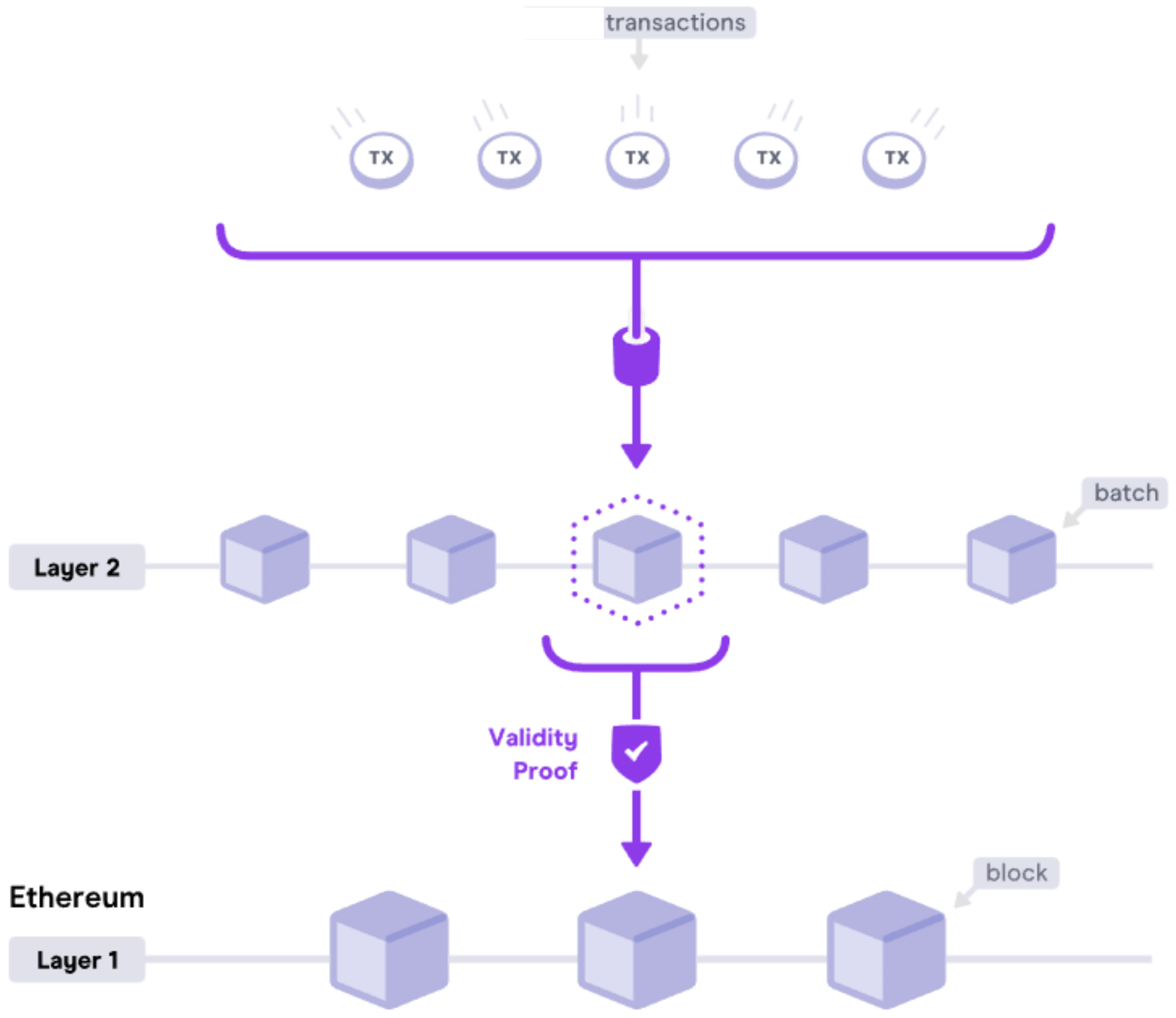

Transaction
VERIFICATION.KEY
Mina


mycoolzkapp.com
Submit
Transaction
ZERO-KNOWLEDGE PROOF
Mina

```
class TicTacToe extends SmartContract {
  // The board is serialized as a single field element
  @state(Field) board: State<Field>;
  // false -> player 1 | true -> player 2
  @state(Bool) nextPlayer: State<Bool>;
  // defaults to false, set to true when a player wins
  @state(Bool) gameDone: State<Bool>;

  // player 1's public key
  player1: PublicKey;
  // player 2's public key
  player2: PublicKey;

  // initialization
  constructor(
    initialBalance: UInt64,
    address: PublicKey,
    player1: PublicKey,
    player2: PublicKey
  ) {
    super(address);
    this.balance.addInPlace(initialBalance);
    this.board = State.init(Field.zero);
    this.nextPlayer = State.init(new Bool(false)); // player 1 starts
    this.gameDone = State.init(new Bool(false));

    // set the public key of the players
    this.player1 = player1;
    this.player2 = player2;
  }
}
```

```
// get player token
const player = Circuit.if(
    pubkey.equals(this.player1),
    new Bool(false),
    new Bool(true)
);
```

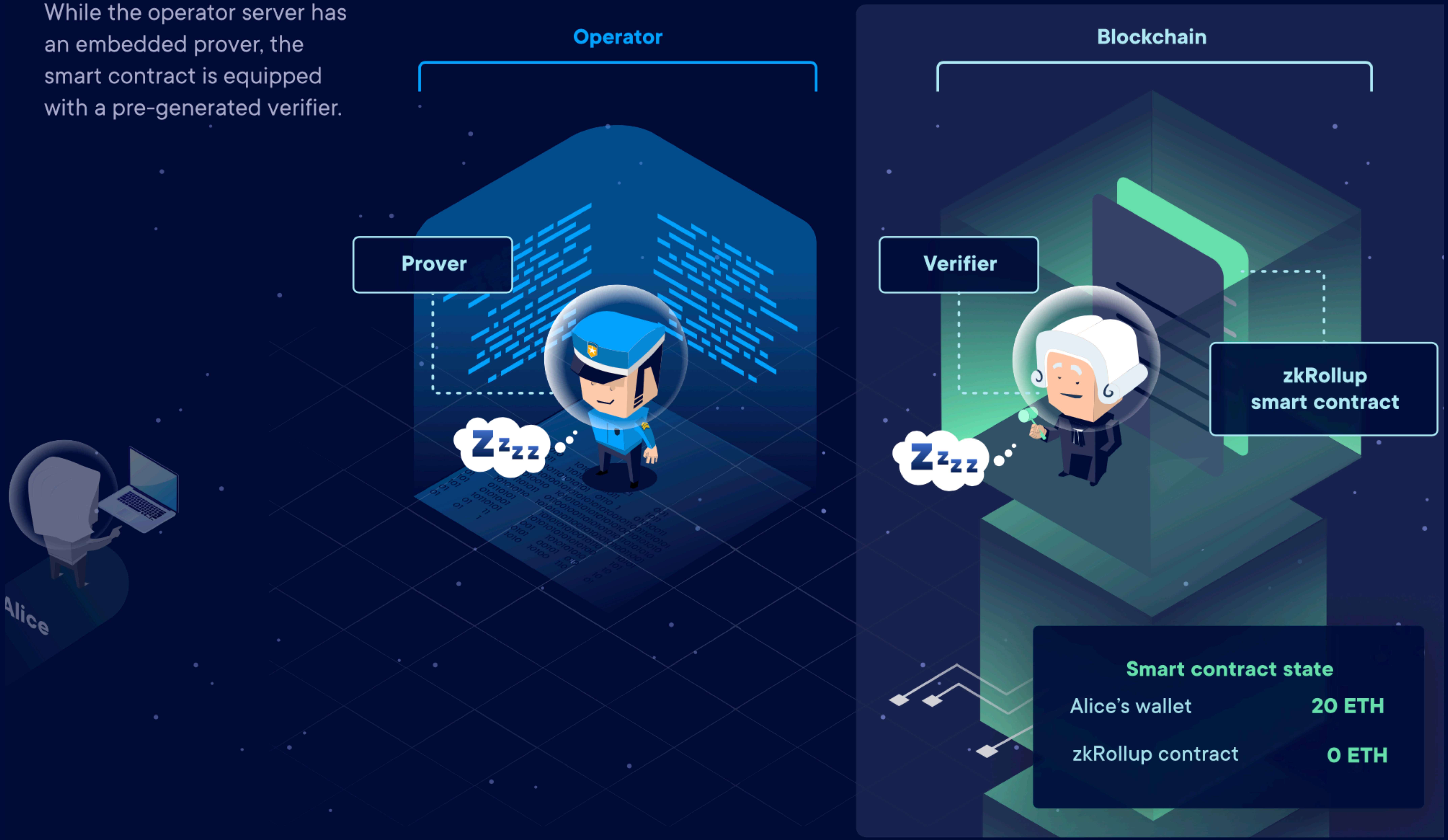
zkRollups



Many validity proofs
can be included in one layer 1 block

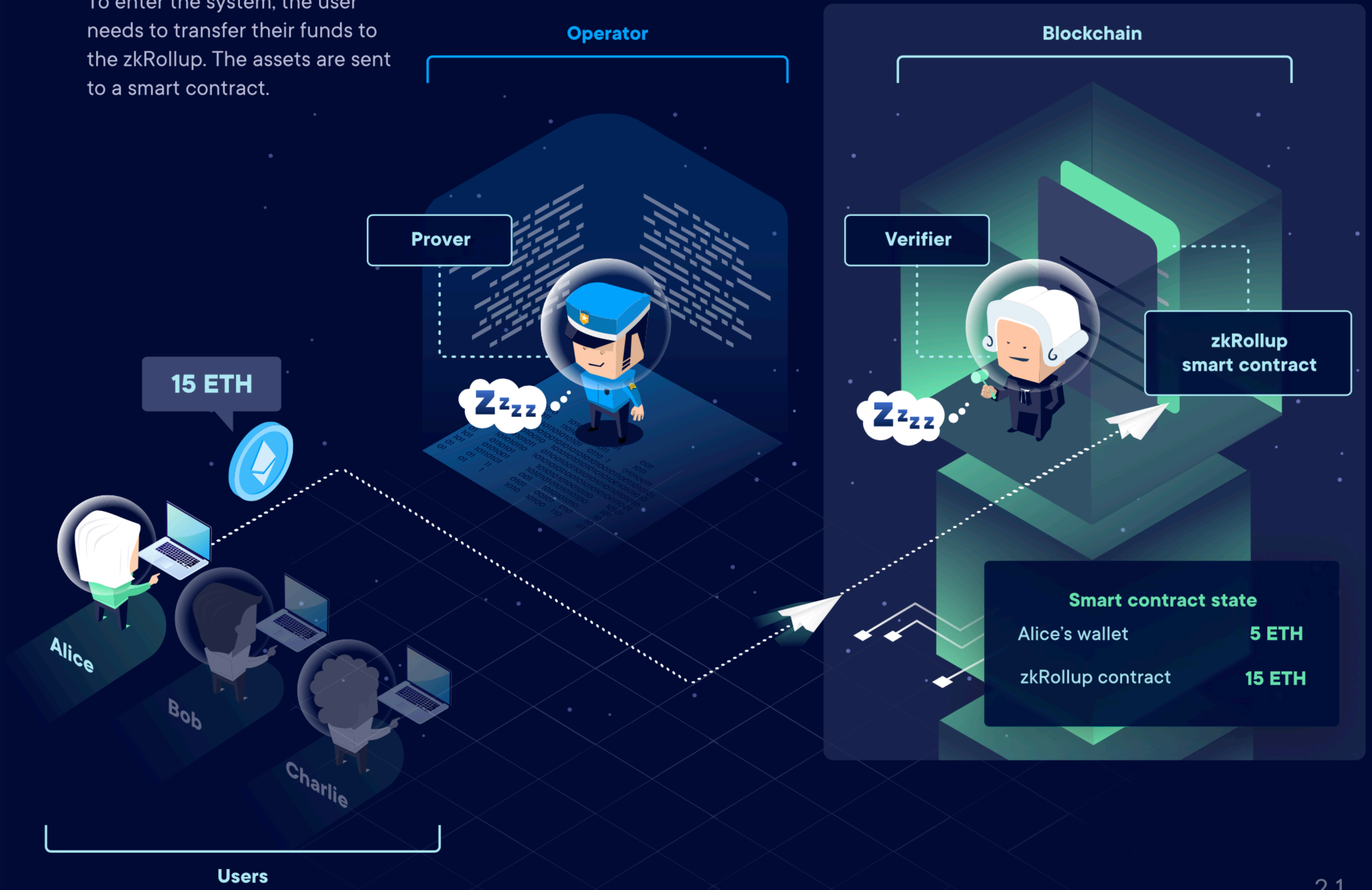
01. Start

While the operator server has an embedded prover, the smart contract is equipped with a pre-generated verifier.



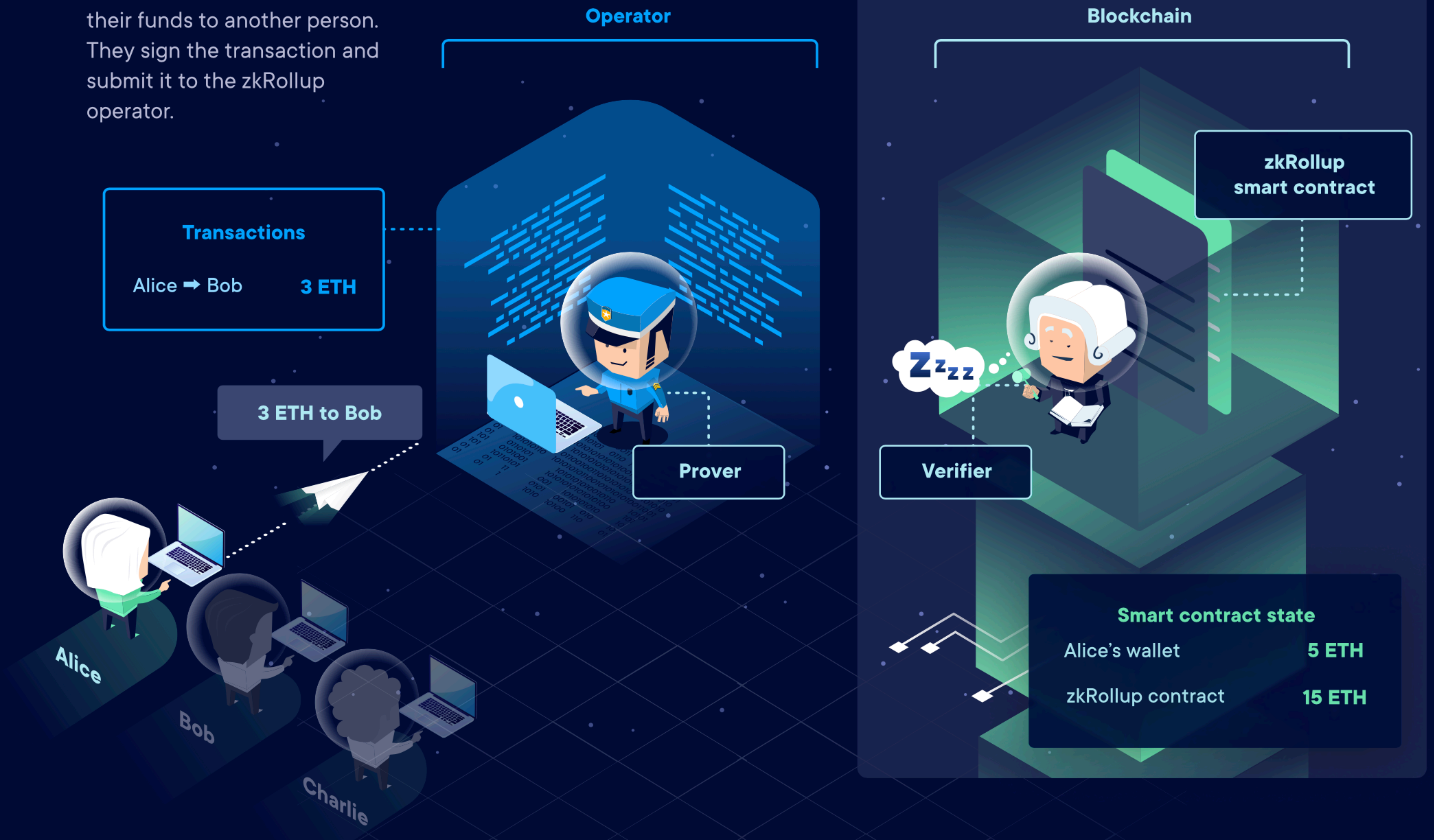
02. Alice's Enter

To enter the system, the user needs to transfer their funds to the zkRollup. The assets are sent to a smart contract.

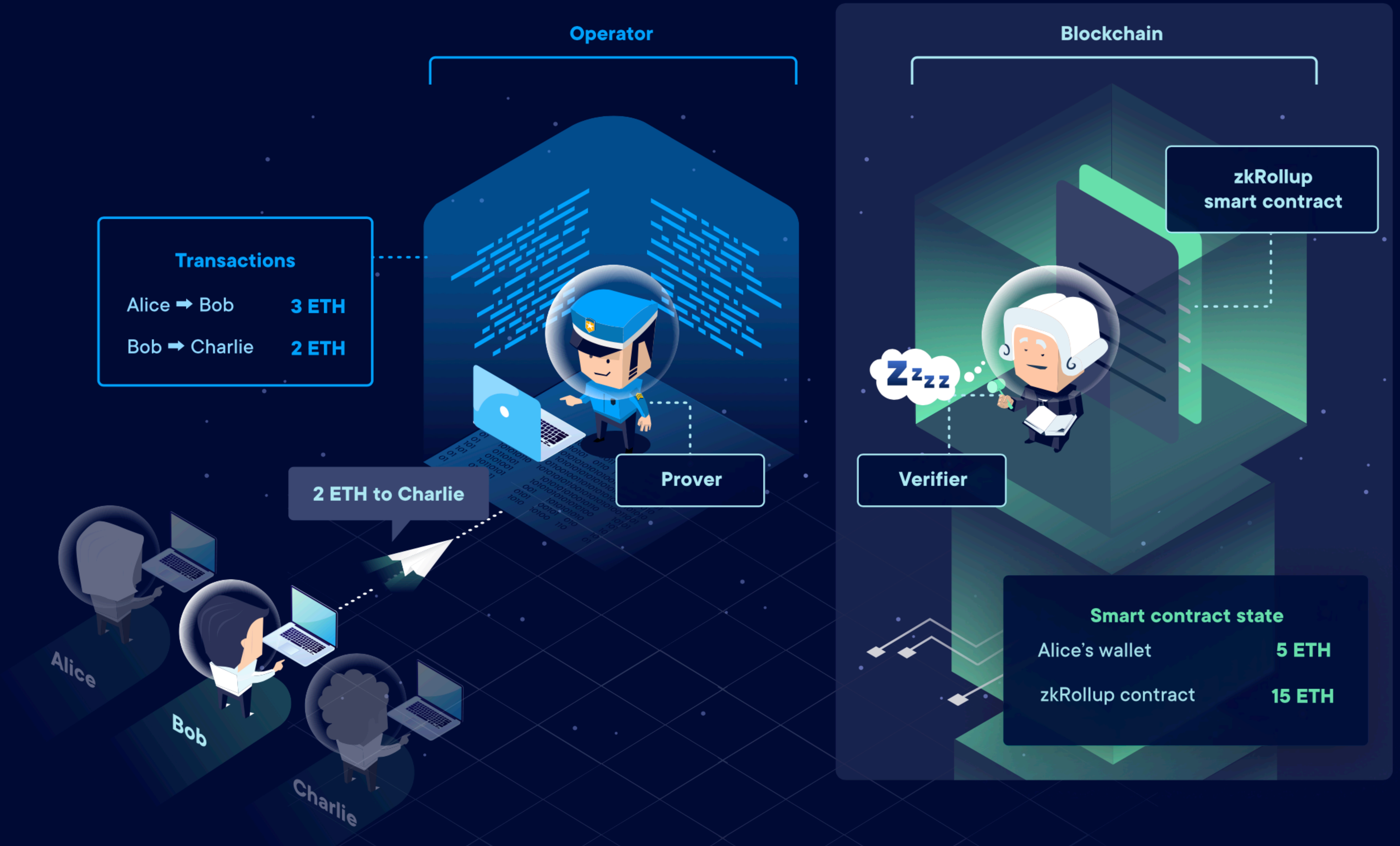


03. Alice's Transfer

The user can now transfer their funds to another person. They sign the transaction and submit it to the zkRollup operator.



04. Bob's Transfer



Transactions

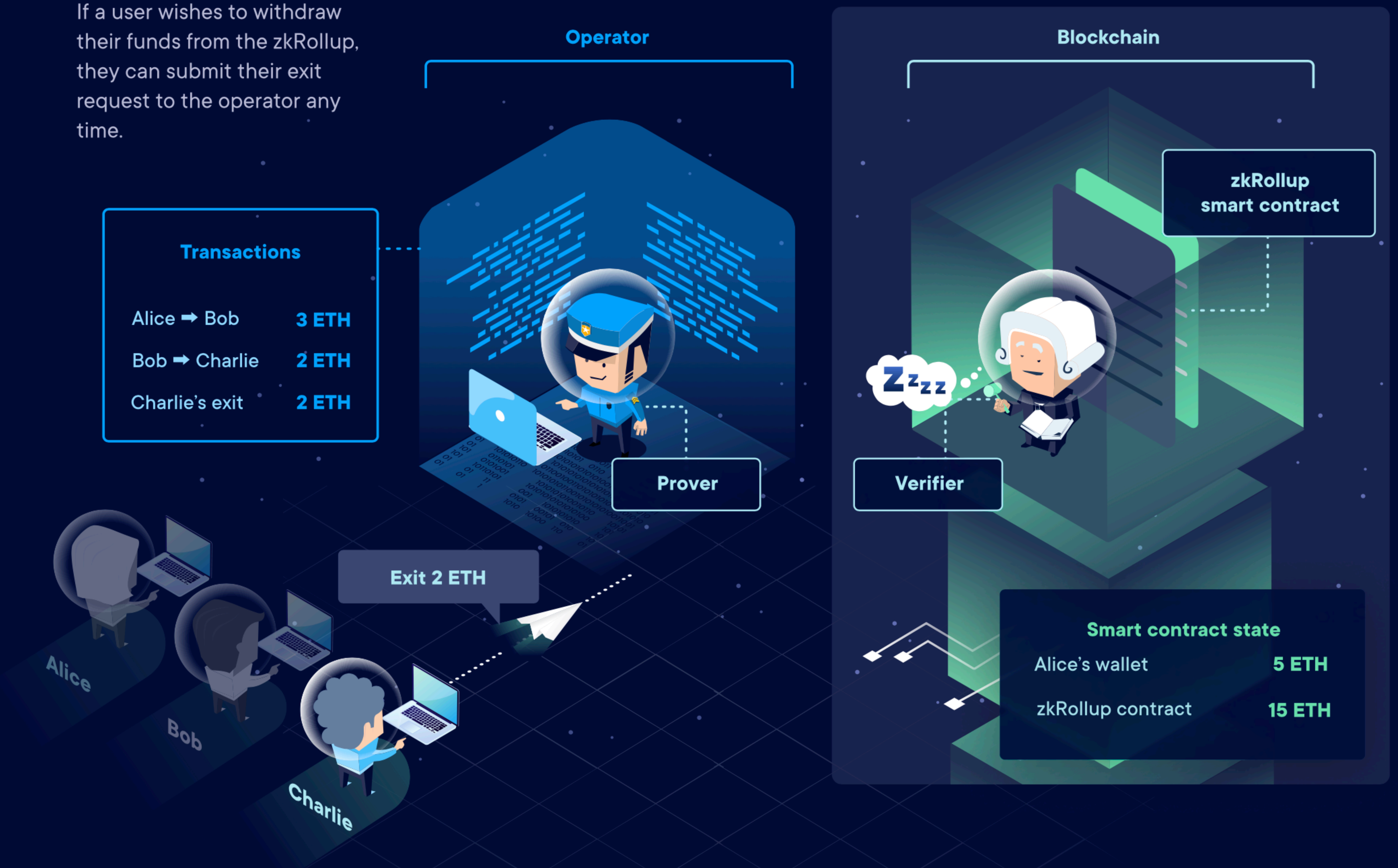
Alice ➡ Bob	3 ETH
Bob ➡ Charlie	2 ETH

Smart contract state

Alice's wallet	5 ETH
zkRollup contract	15 ETH

05. Charlie's Exit

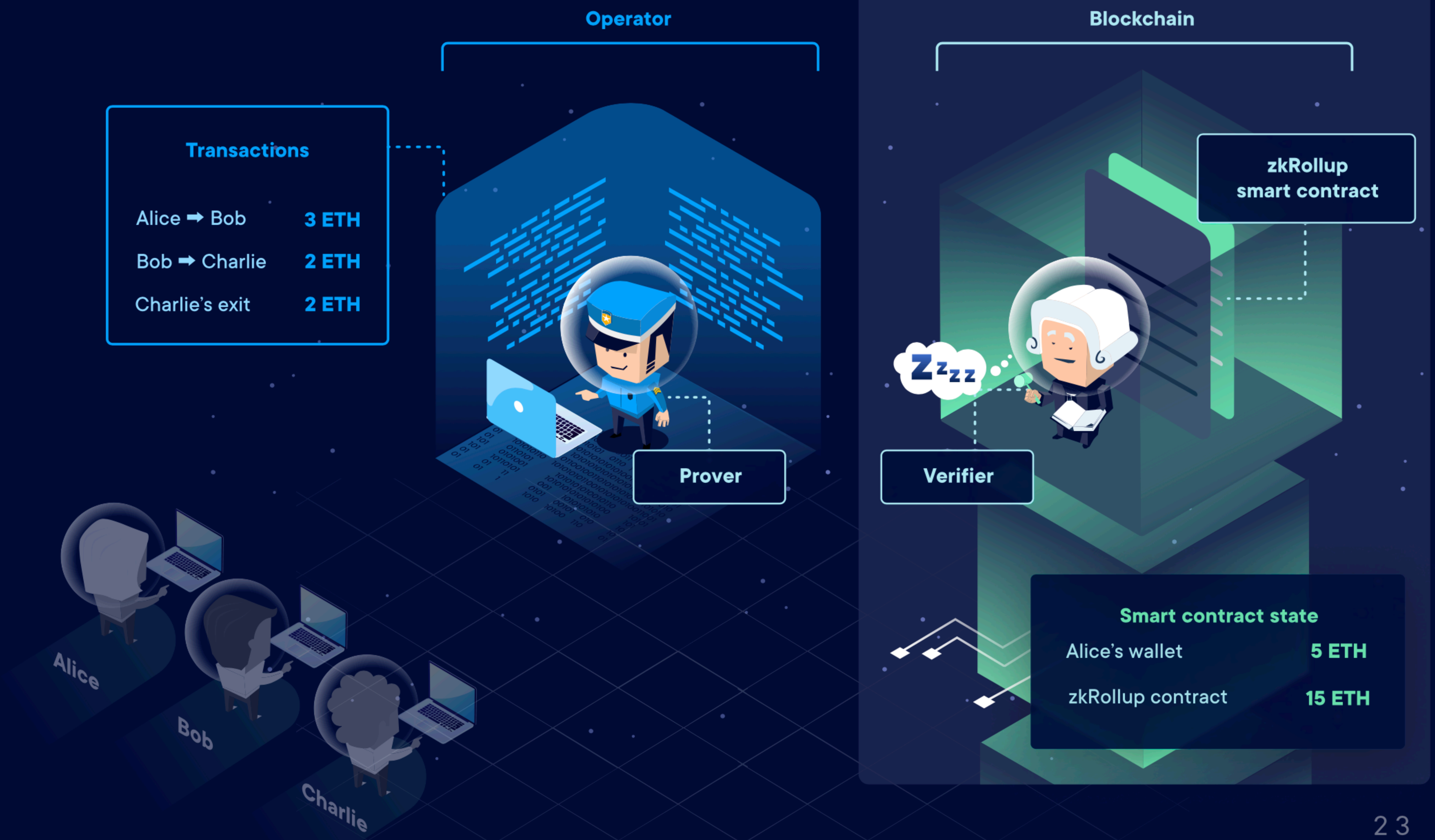
If a user wishes to withdraw their funds from the zkRollup, they can submit their exit request to the operator any time.



06. Collecting Transactions

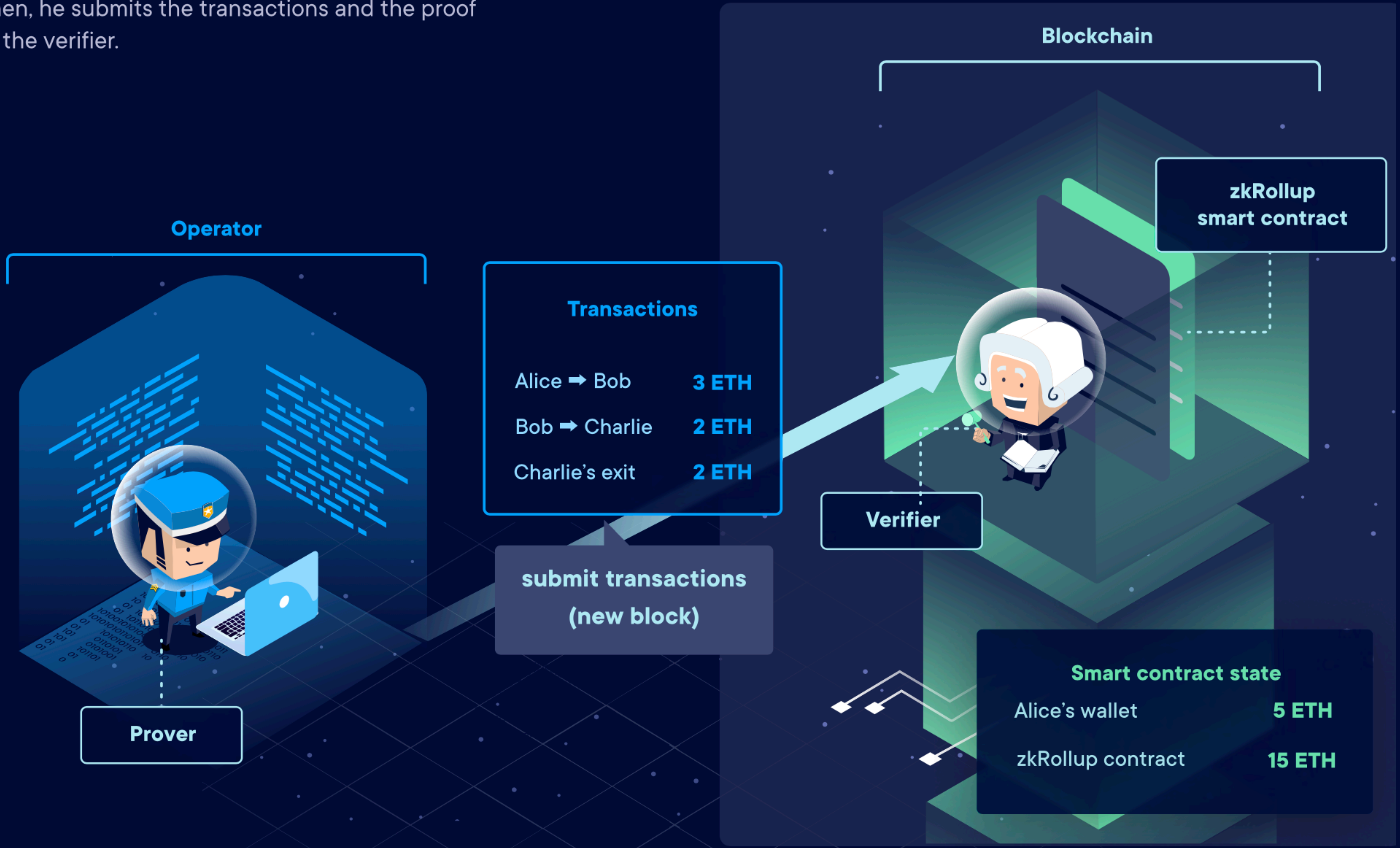
In the meantime, the operator collects transactions and exit requests from many users.

* Note that even if Bob and Charlie didn't have any funds on the zkRollup, they could still receive transfers from other users.



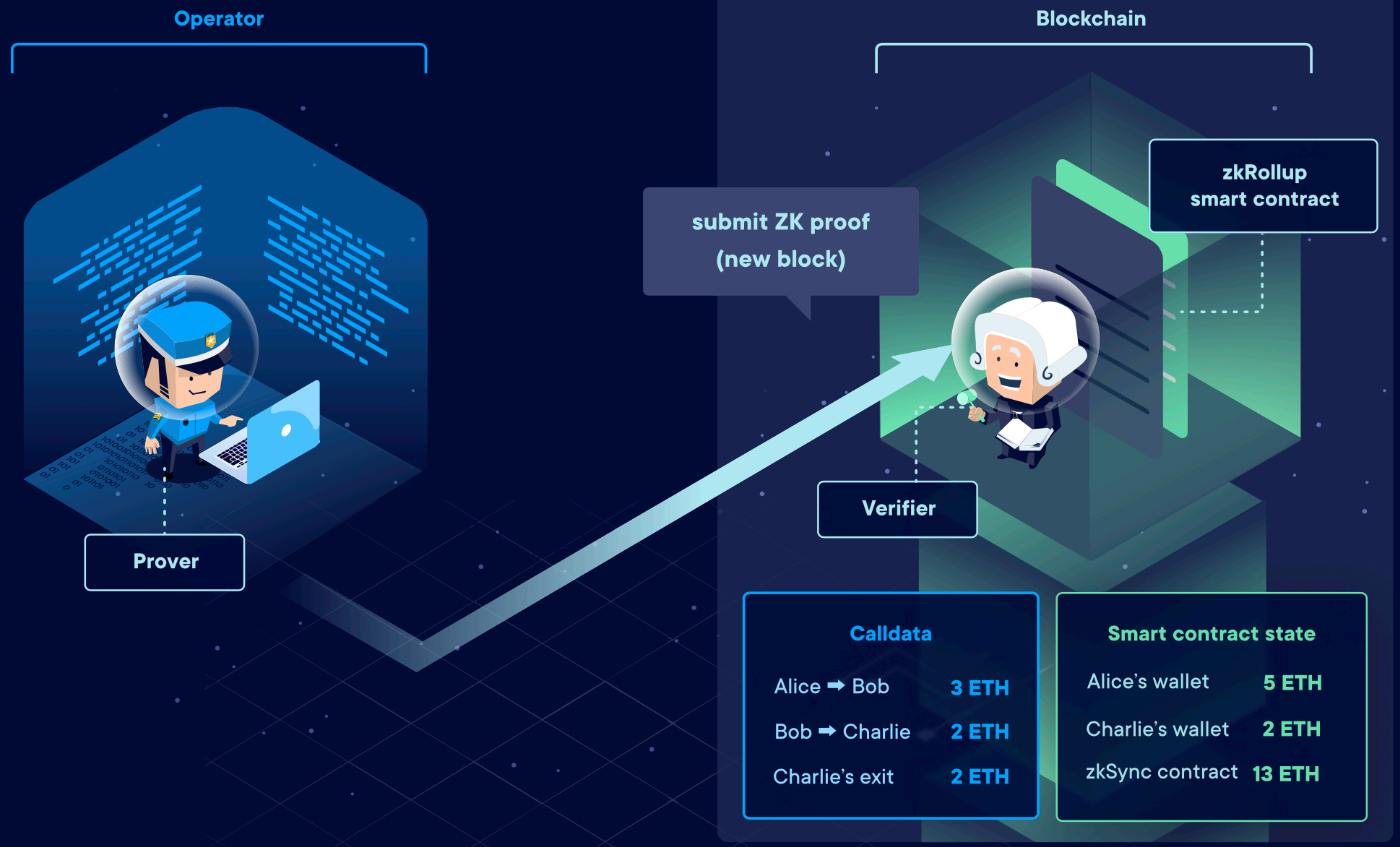
07. Submitting Transactions

Once in a while, the operator bundles the collected transactions together and generates a ZK proof. Then, he submits the transactions and the proof to the verifier.



08. Submitting ZK Proof

The smart contract verifies the transactions and the proof. Once it's done, the transactions are finalized.



Calldata

Alice ➡ Bob	3 ETH
Bob ➡ Charlie	2 ETH
Charlie's exit	2 ETH

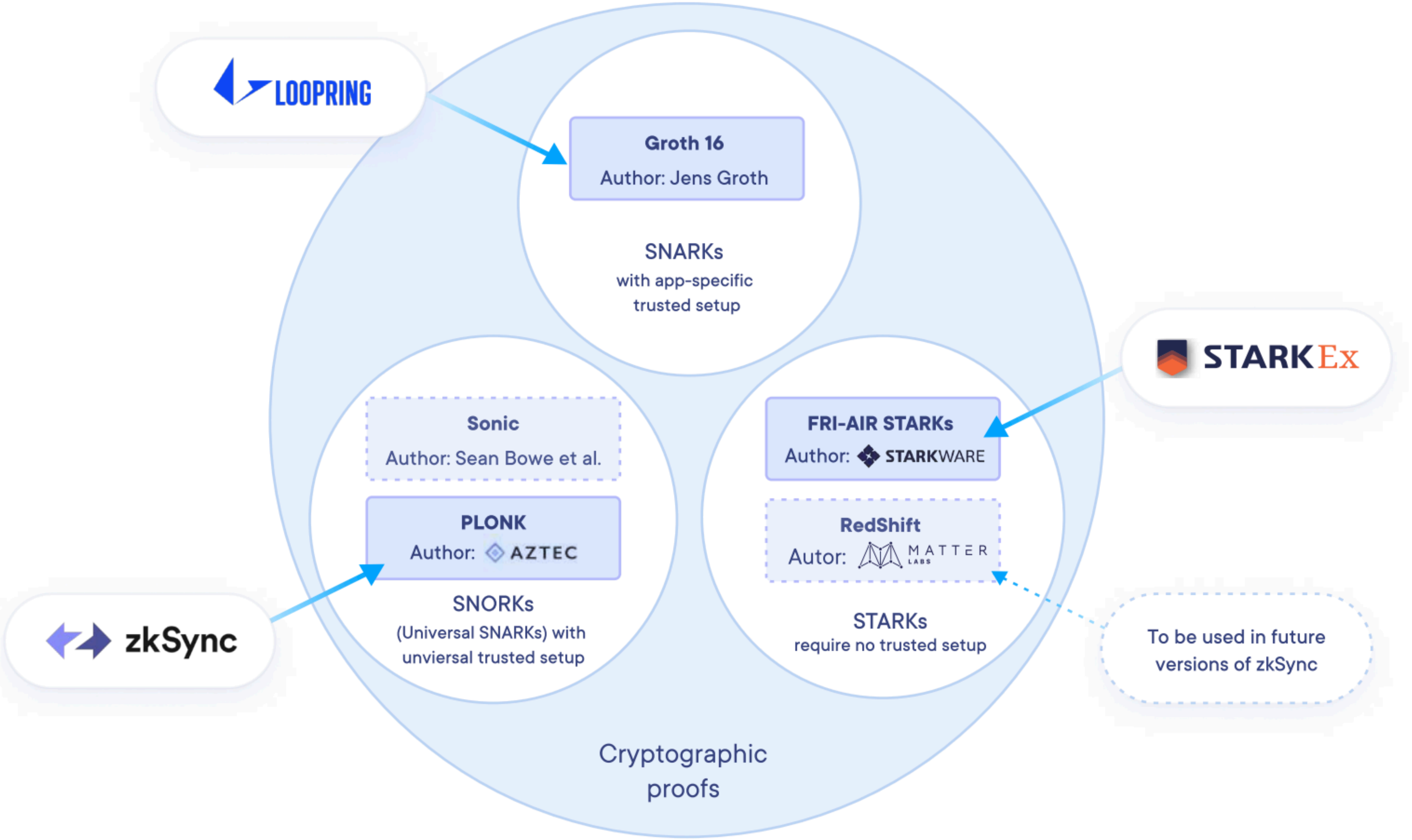
Smart contract state

Alice's wallet	5 ETH
Charlie's wallet	2 ETH
zkSync contract	13 ETH

SNARKs vs. STARKs

SNARK stands for:

- **s**uccinct: the proof is significantly smaller than the data it represents and can be verified quickly,
- **n**on-interactive: only one set of information is sent by the prover to the verifier, thus there's no back-and-forth interaction between them,
- **ar**gument of **k**nowledge: the proof is considered computationally sound—a malicious prover isn't likely to cheat the system without possessing the knowledge to support its statement.


LOOPRING
Groth 16
Author: Jens Groth
SNARKs
with app-specific trusted setup
STARKEx
FRI-AIR STARKs
Author: STARKWARE
RedShift
Autor: MATTER LABS
STARKs
require no trusted setup
To be used in future versions of zkSync
Sonic
Author: Sean Bowe et al.
PLONK
Author: AZTEC
SNORKs
(Universal SNARKs) with unviersal trusted setup
zkSync
Cryptographic proofs

Solutions based on the early SNARK technology (i.e. Groth16) require conducting the ceremony for every new version of the product. That's why Loopring described in our report later on needed to conduct one before launching the latest version of their protocol last year.

Another variant called Universal SNARKs or SNORKs (e.g. PLONK and SONIC), leverages **universal trusted setup**. For example, zkSync creators didn't have to conduct their own ceremony while launching the product: they re-used the ignition multi-party computing performed last year with approx. 200 reputable figures such as Vitalik Buterin. Universal trusted setup also allows them to extend and upgrade the zero-knowledge part of the protocol without conducting another ceremony.

Technology

- Circom + zksnarks.js
- SnarkyJS - Mina
- Cairos - Starkware
- Zokrates

How does it work?

Computation → Arithmetic Circuit → R1CS → QAP → zk-SNARK

- Homomorphic Hiding
- Blind Evaluation of Polynomials
- The Knowledge of Coefficient Test and Assumption
- How to make Blind Evaluation of Polynomials Verifiable
- From Computations to Polynomials
- The Pinocchio Protocol
- Pairings of Elliptic Curves

Thank you!

Marek Kirejczyk

@ethmarek