Introduction to Zero-knowledge Proofs

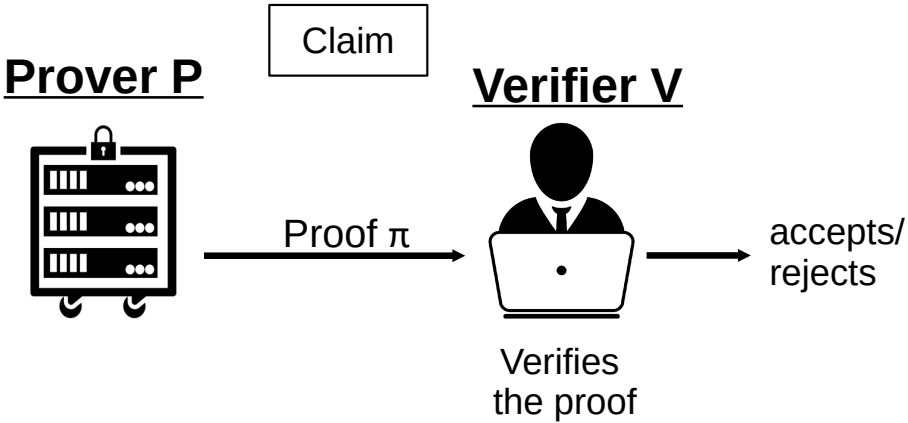
Martin Perešíni, Samuel Olekšák

Brno University of Technology, Faculty of Information Technology
Božetěchova 1/2, 612 00 Brno-Královo Pole
{iperesini, ioleksak}@fit.vutbr.cz

July 31, 2024

Outline

1. What are zero-knowledge proofs (ZKPs)
2. Simple examples of ZKPs
3. ZKP properties
4. ZKP demo with ZoKrates
5. ZKP applications in blockchains


Claim
Prover P
Verifier V
Proof π
accepts/
rejects
Verifies
the proof

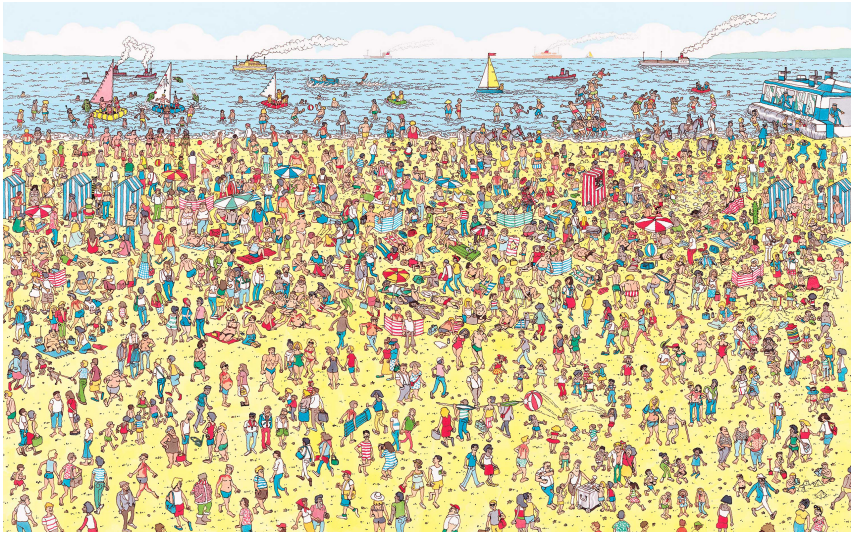
Zero-knowledge Proofs (ZKPs)

- Prover can convince the verifier about a truthfullness of a statement without revealing any aditional information about the statement

ZKP Motivation

- Blockchain enables ZKPs by storing proofs on-chain
- Use-cases:
  - Verifiable Computations
  - Private Transactions
  - Scalable and Secure Layer-2s
  - Decentralized Identity and Authentication

Example of a ZKP – *Where's Waldo?*



Example of a ZKP – *Where's Waldo?*

Peggy wants to convince **Victor** that she has a solution to the puzzle without revealing it.

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

Peggy draws a random permutation on $\{1, \ldots, 9\}$ and replaces the numbers accordingly.

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

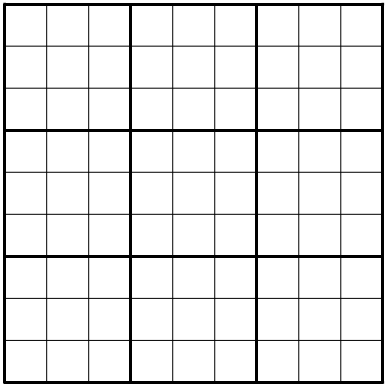
$\rightarrow$

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

Peggy does not reveal the whole solution to **Victor**.

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

→

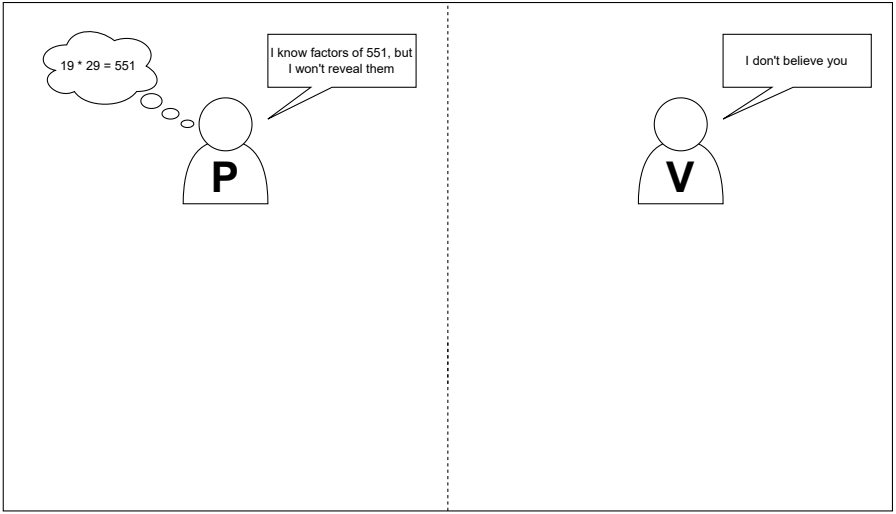


Victor chooses either a row, column, or box.[1] **Peggy** reveals it. **Verifier** checks for consistency. Repeat each time, **Peggy** draws a new permutation. Notice "INTERACTIVITY", "PROBABILISTIC TRUST".

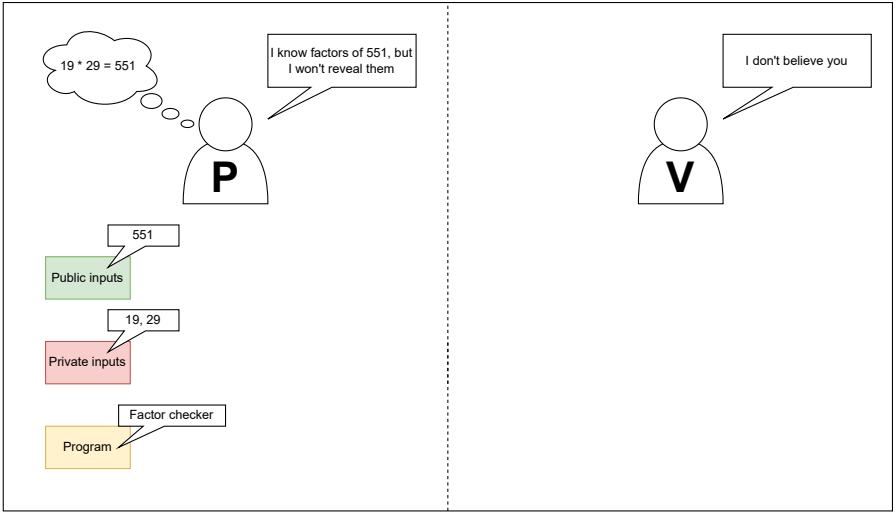
					3			
					2			
					1			
					5			
					9			
					4			
					8			
					7			
					6			

[1] **Victor** can also ask to see the original puzzle 'numbers'.

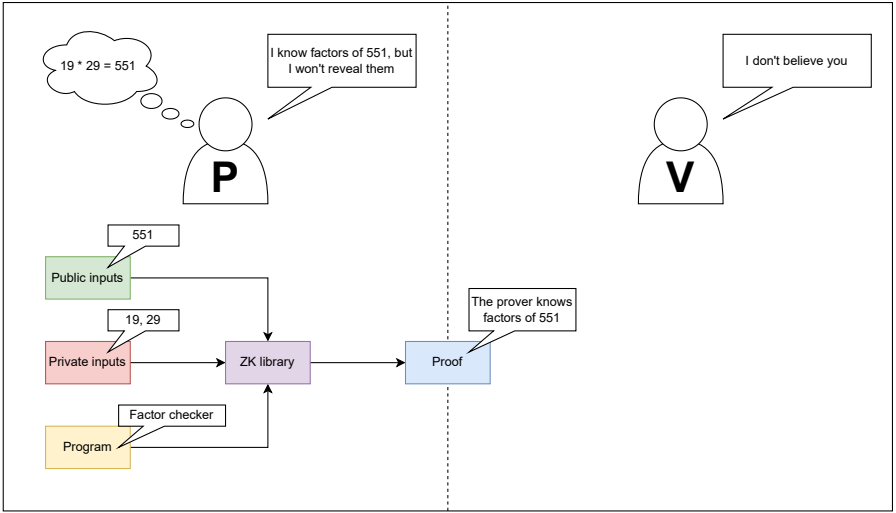
Privacy Preservation with ZKPs



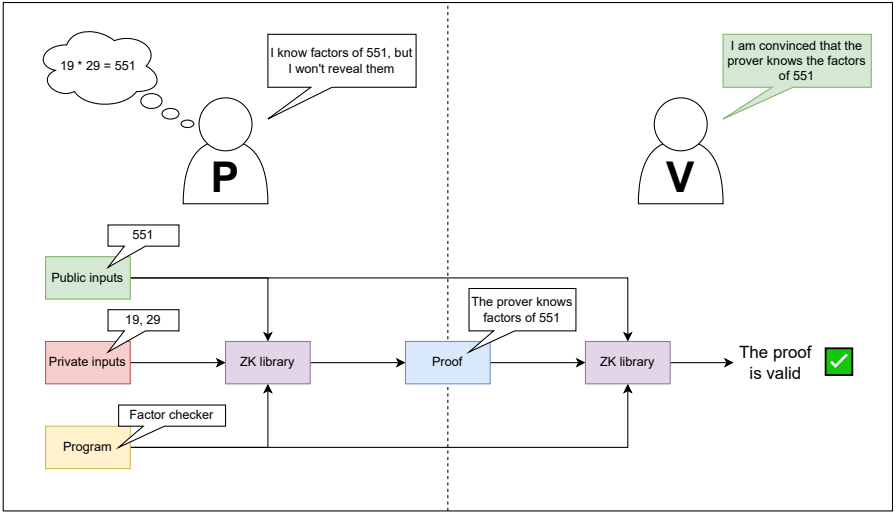
Privacy Preservation with ZKPs



Privacy Preservation with ZKPs



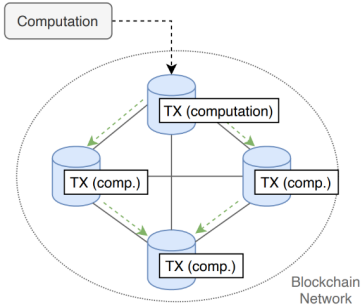
Privacy Preservation with ZKPs



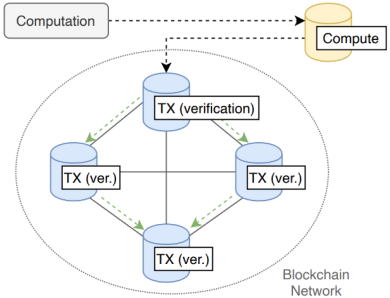
Succinctness

- Proofs are often smaller in size compared to the original statement
- Proofs are often faster to verify compared to the original statement
- The size and verification time depend on the ZK protocol and the circuit (program)

Default on Blockchains:
Replicated Computation



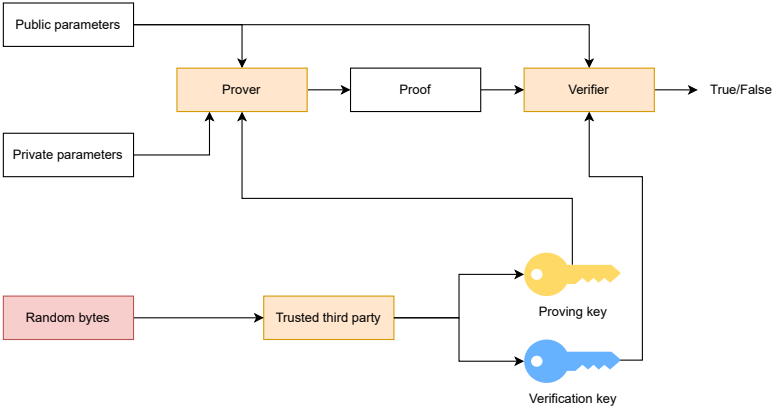
Verifiable Off-chain Computation (VOC):
Replicated Verification



	zk-SNARKs	zk-STARKs
Primary cryptographic mechanism	Elliptic curves	Hashing
Interactive	No	Yes
Require initial setup (ceremony)	Yes	No
Believed to be post-quantum secure	No	Yes
Proof size	Smaller	Larger

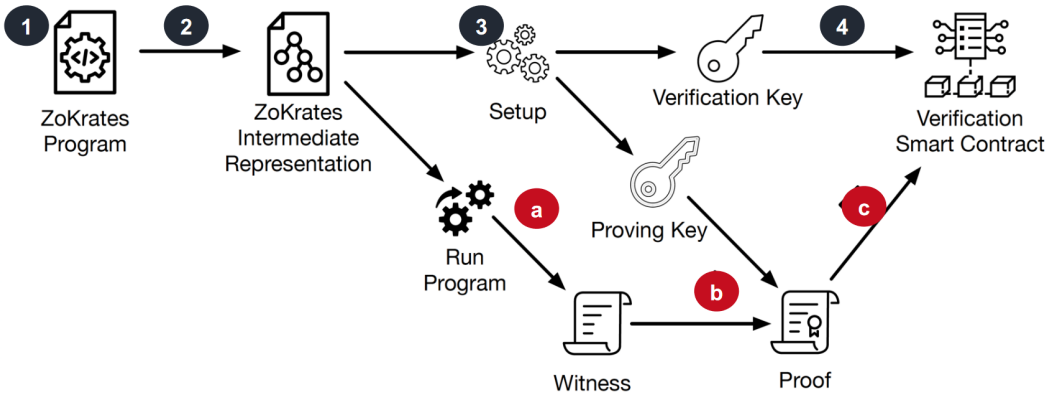
Comparison of properties of two popular zero-knowledge protocols.

Trusted Setup



zk-SNARK requires performing trusted setup, where both parties have to trust a third party to generate proving and verification key and delete the used random byte. Knowledge of random bytes allows a malicious party to generate fake proofs.

- Open-source zk-SNARK library written in Rust
- Enables proof generation and verification
- Verification can be performed directly with the library or on-chain using the auto-generated smart contract
- Provides a domain-specific high-level language for describing arithmetic circuits


1
ZoKrates
Program
2
ZoKrates
Intermediate
Representation
3
Setup
Verification Key
4
Verification
Smart Contract
Proving Key
a
Run
Program
b
c
Witness
Proof

ZoKrates – Example

```
def main(
    private field factor1,
    private field factor2,
    public field product
) -> bool {
    assert (factor1 * factor2 == product);

    return true;
}
```

Prove the knowledge of two secret factors which make up public product.

ZoKrates – Example

```
def main(private u32 a) -> bool {
    u32 mut b = a;

    for u32 i in 0..5 {
        b = if b % 2 == 0 { b / 2 } else { b * 3 + 1 };

        assert (b != 1);
    }

    return true;
}
```

Prove knowledge of a number that would take at least 5 steps to converge to the number 1 given the rules of Collatz conjecture without revealing that number to the verifier.

ZoKrates Demo – Rewarding Integer Factorization

- Let's say we need a list of semiprime numbers[2] for RSA key generation
- For the RSA to be secure, it should be really hard to find the factors of a chosen semiprime
- Let's create a blockchain service which will publicly announce the semiprimes
- If a semiprime is factored by some node in the network, they can announce it and receive a token reward
- The corresponding semiprime is removed from the list since it was probably not secure and should not be used in RSA

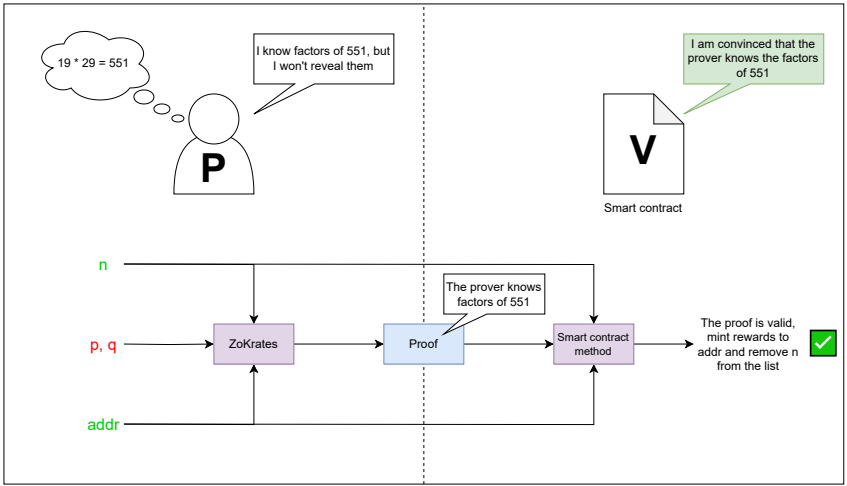
[2] Number is a semiprime if it is a product of exactly two primes

- The simplest solution
  - Create a smart contract which will have a function to check if the provided factors are correct and will mint a reward
  - Prone to front-running attacks
    - User A will publicly announce a transaction with the factors to be included in the mempool
    - User B will copy the transaction with themselves as the reward recipient
    - User B can increase the gas fee to increase chances of their transaction being confirmed first

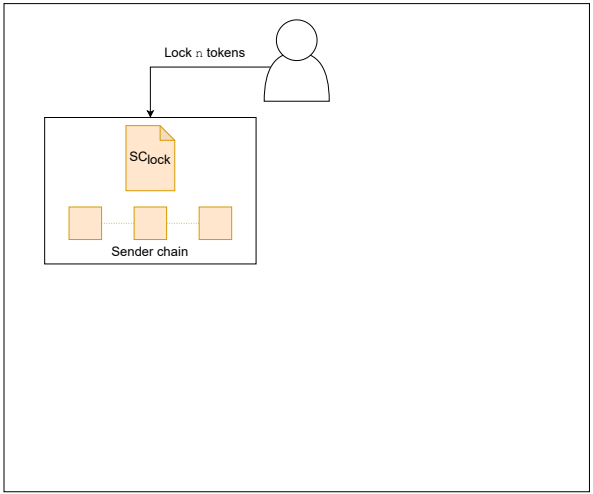
ZoKrates Demo – Rewarding Integer Factorization

- We can prevent the front-running attack using ZKPs
- A node in the network can submit a ZKP containing the factors p, q as the private parameters and their address *addr* along with the semiprime n
- Upon successful verification the contract would send the reward to *addr*
- If an attacker tried to change *addr* to their address then the proof verification would fail
- Attacker cannot recreate the transaction since p, q are only known by the prover (neither does the smart contract know)

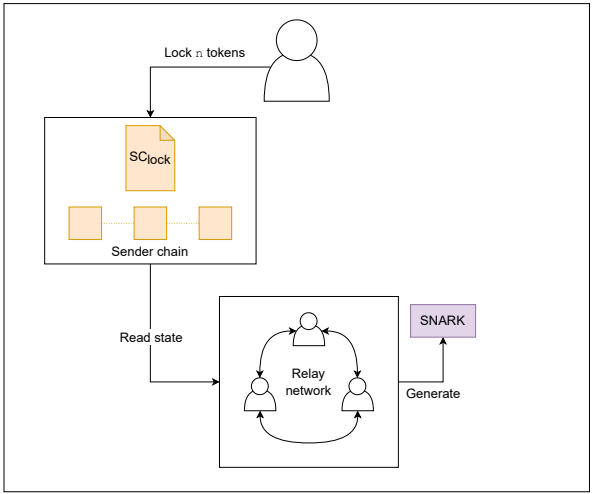
ZoKrates Demo – Rewarding Integer Factorization



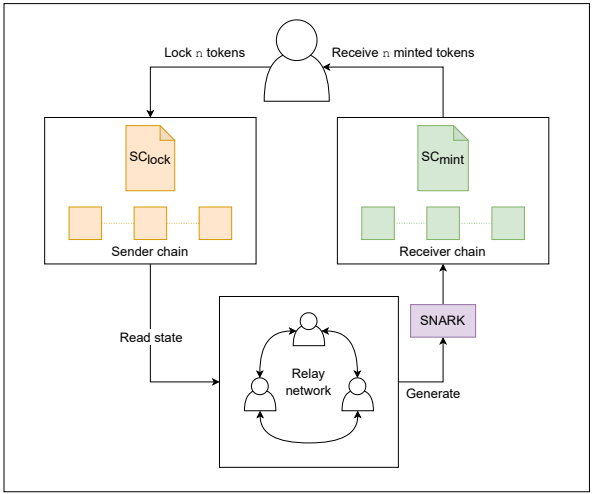
Applications – Cross-chain Interoperability (1)



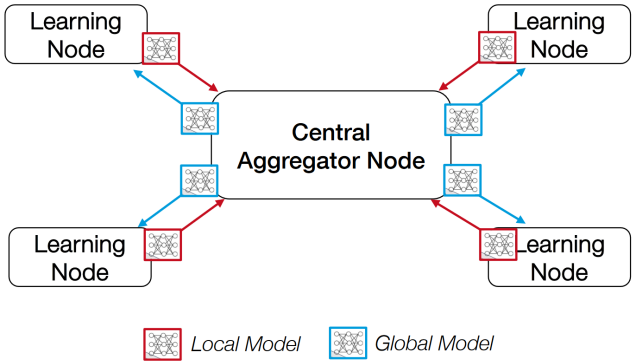
Applications – Cross-chain Interoperability (1)



Applications – Cross-chain Interoperability (1)

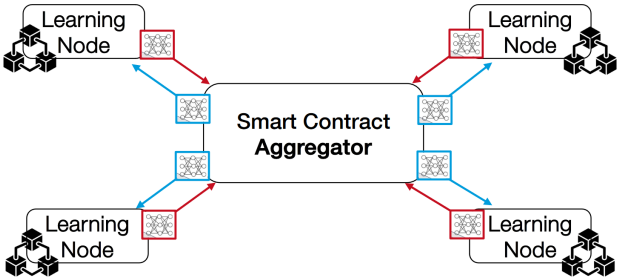


Applications – Federated Learning (2)



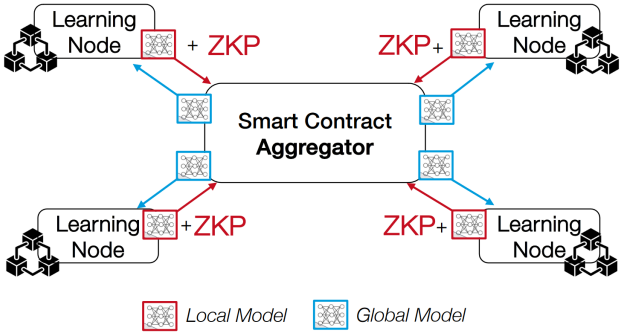
Learning nodes train and submit local models to the aggregator and synchronize with the global model.

Applications – Federated Learning (2)



Each node uses a blockchain client to submit local model to a smart contract. The aggregator has to trust each node not to poison the model.

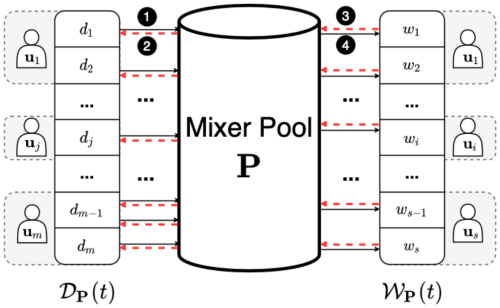
Applications – Federated Learning (2)



ZKP attests to the correctness of the performed computation and prevents malicious nodes from poisoning the model. The aggregator verifies the proof before creating and returning the global model.

Applications – Transaction Mixing (3)

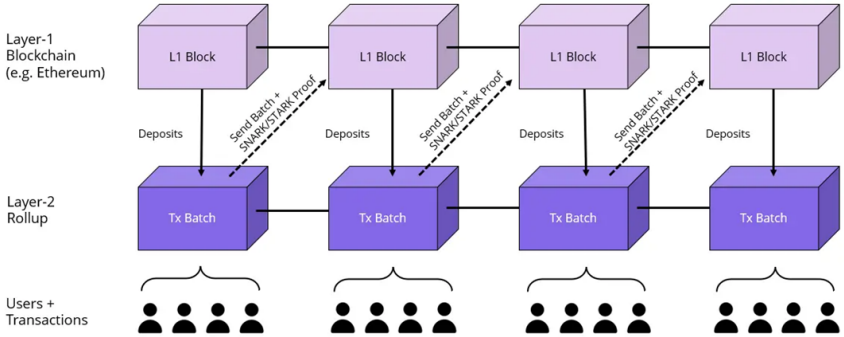
- Achieve higher account privacy by pooling together funds from multiple users.
  1. Deposit tokens in mixer pool
  2. Receive deposit key
  3. Prove that you are the holder of the deposit key with zk (from another address)
  4. Receive withdraw
- *Tornado Cash*



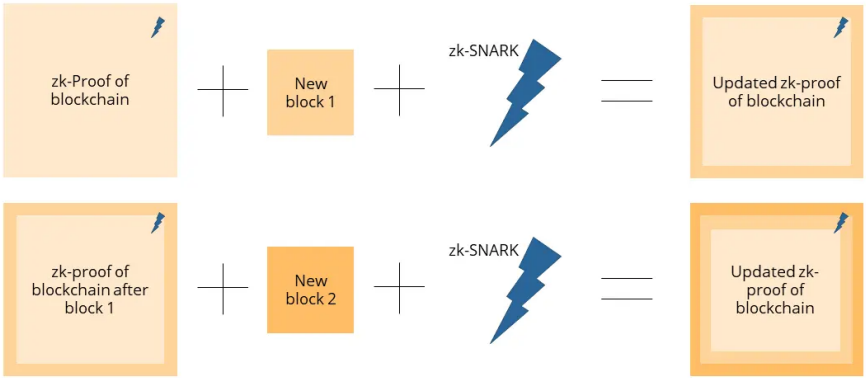
MESSARI

ZK Rollup Transaction Process

Transactions are batched on the rollup network and sent back to the mainnet with a SNARK proof to verify transactions



Applications – Tiny Blockchain with Recursive ZKPs (5)



References

(1) T. Xie, J. Zhang, Z. Cheng, F. Zhang., Y. Zhang et al.
ZkBridge: Trustless Cross-chain Bridges Made Practical

(2) J. Heiss, E. Grünewald, N. Haimerl, S. Schulte, S. Tai
Advancing Blockchain-based Federated Learning through Verifiable Off-chain Computations

(3) A. Pertsev, R. Semenov, R. Storm
Tornado Cash Privacy Solution

(4) Ethereum Foundation
Zero-knowledge rollups

(5) J. Bonneau, I. Meckler, V. Rao, E. Shapiro
Mina: Decentralized Cryptocurrency at Scale

Further Reading – ZKPs in General

A. Berentsen, J. Lenzi, and R. Nyffenegger
An Introduction to Zero-Knowledge Proofs in Blockchains and Economics
Very short introduction to zk-STARKs

A. Berentsen, J. Lenzi, and R. Nyffenegger
A Walk-Through of a Simple zk-STARK Proof
Complete overview of zk-STARK proof generation and verification

Least Authority
The MoonMath Manual
Deep dive into zk-SNARKs for non-cryptographers

A. Alonso, J. Sedlmeir, J. Heiss
Compute Offchain, Verify Onchain: How to Build zk-DApps with Circom and ZoKrates
Introduction to ZKP and tutorial on ZoKrates and Circom

Further Reading – Uses of ZoKrates

- M. Westerkamp and J. Eberhardt
  zkRelay: Facilitating Sidechains using zkSNARK-based Chain-Relays
  Using ZoKrates to facilitate cross-chain communication
- S. Slávka
  Mobile Cryptocurrency Wallet Based on zk-SNARKs and Smart Contracts
  Using zk-SNARKs with ZoKrates to offload blockchain verification from mobile devices
- S. Olekšák
  The Analysis of Cryptographic Techniques for Offloading Computations and Storage in Blockchains
  Analyze existing use-cases of ZKPs for storage and computation offloading from blockchain and implement SNARK marketplace