Blockchain – A Hands-on Introduction

Author: Zigtur

Twitter: Zigtur
GitHub: Zigtur

Agenda

01

INTRODUCTION

Blockchain

- Peer-to-peer decentralized payment system
  2009 : Bitcoin (whitepaper here)
  2014 : Ethereum (whitepaper here)

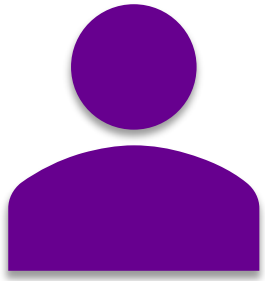
- Use cases have then evolved!
  - Voting systems
  - On-chain DNS (e.g. ENS)
  - File transfer services (e.g. IPFS)
  - Traceability
  - Games

02

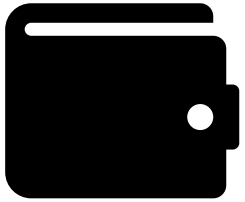
END-USERS

An end-user only needs:

- An internet connection



- A wallet application (optional), such as Metamask



- A cryptographic keypair, for digital signatures

Demo time !

- Presentation of a software wallet : Metamask

METAMASK

More wallets…

Keeping crypto on a centralized exchange

Keeping your digital assets secure with a Ledger Nano

- There are a lot of software wallets…

phantom

MyEtherWallet

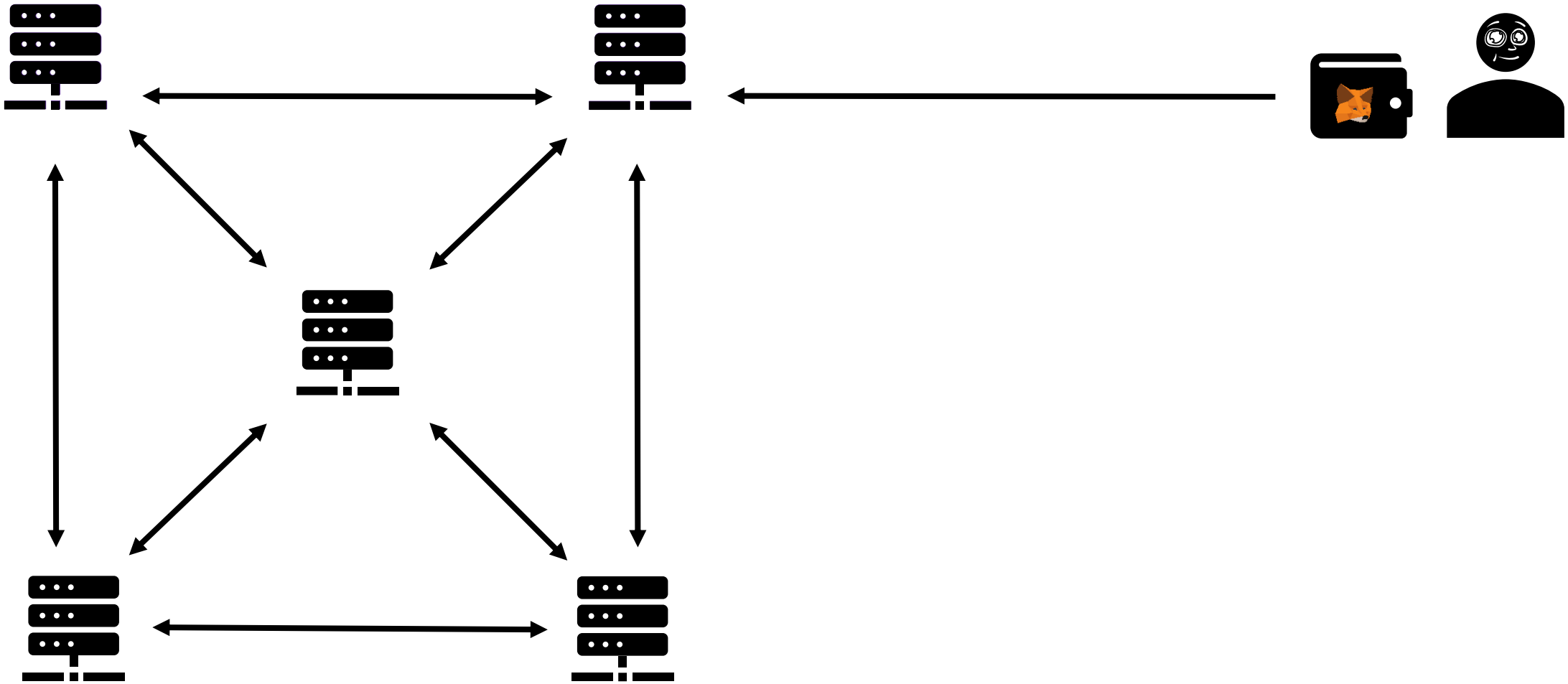
- But hardware wallet should be used !

LEDGER

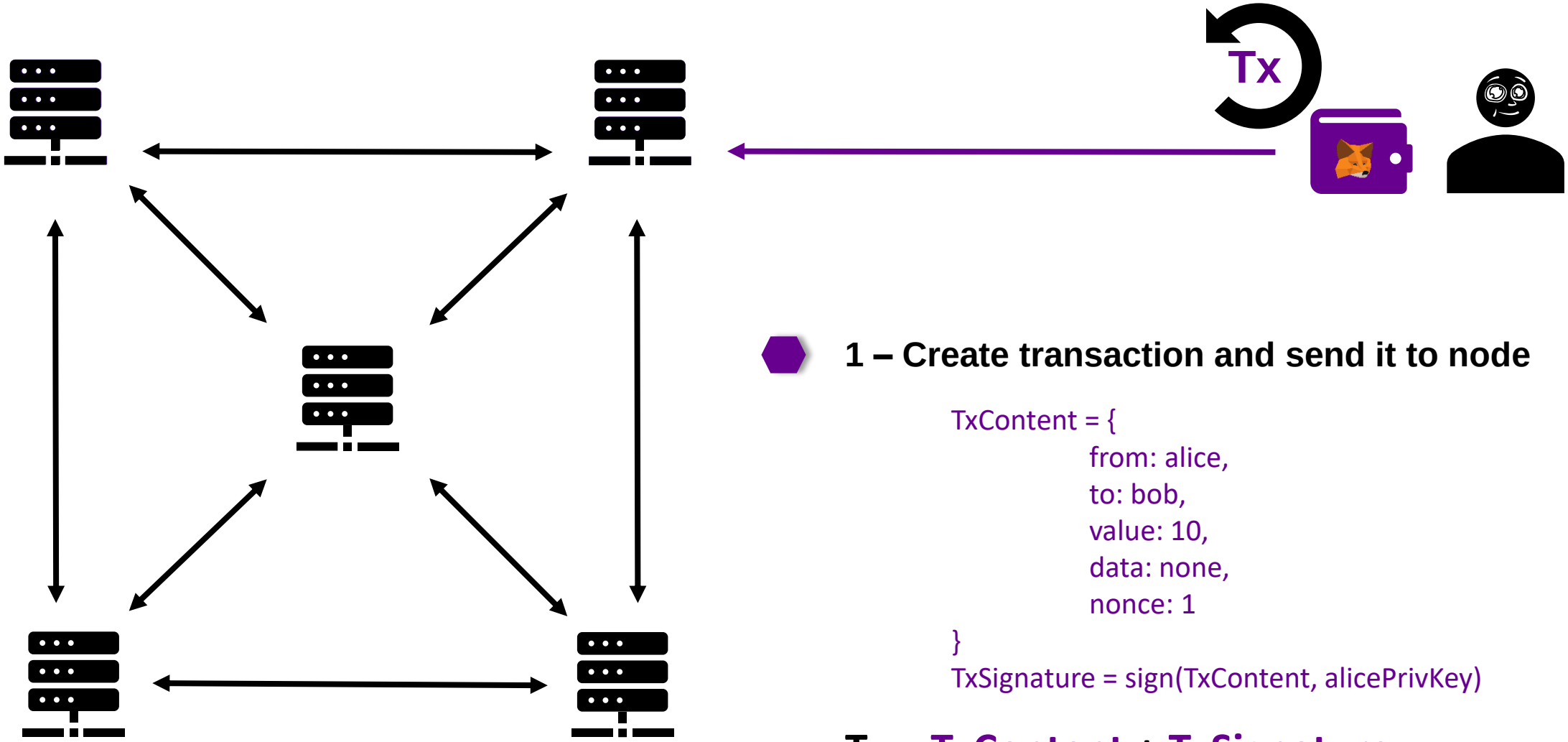
03

TRANSACTIONS, NODES, BLOCKS AND CONSENSUS

NODES



NODES

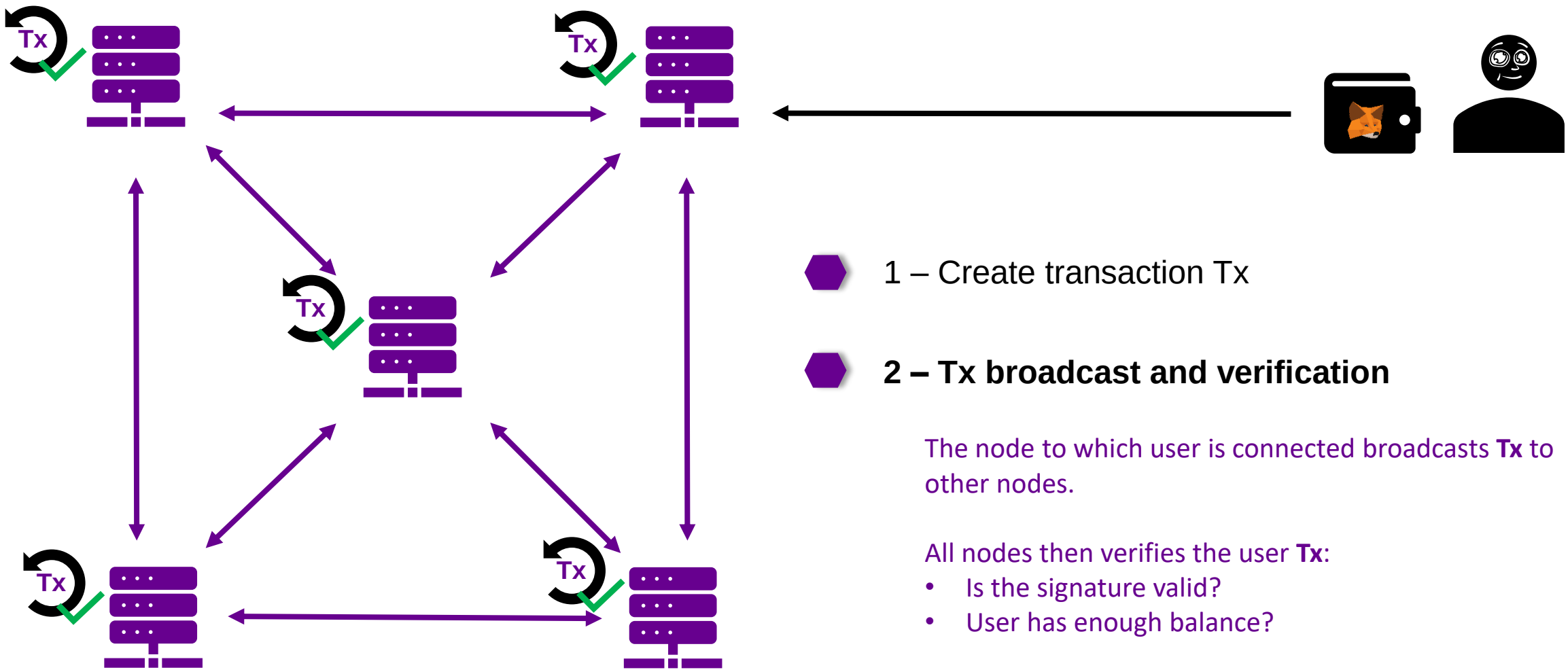


- **1 – Create transaction and send it to node**

```
TxContent = {
        from: alice,
        to: bob,
        value: 10,
        data: none,
        nonce: 1
}
TxSignature = sign(TxContent, alicePrivKey)
```

Tx = TxContent + TxSignature

NODES



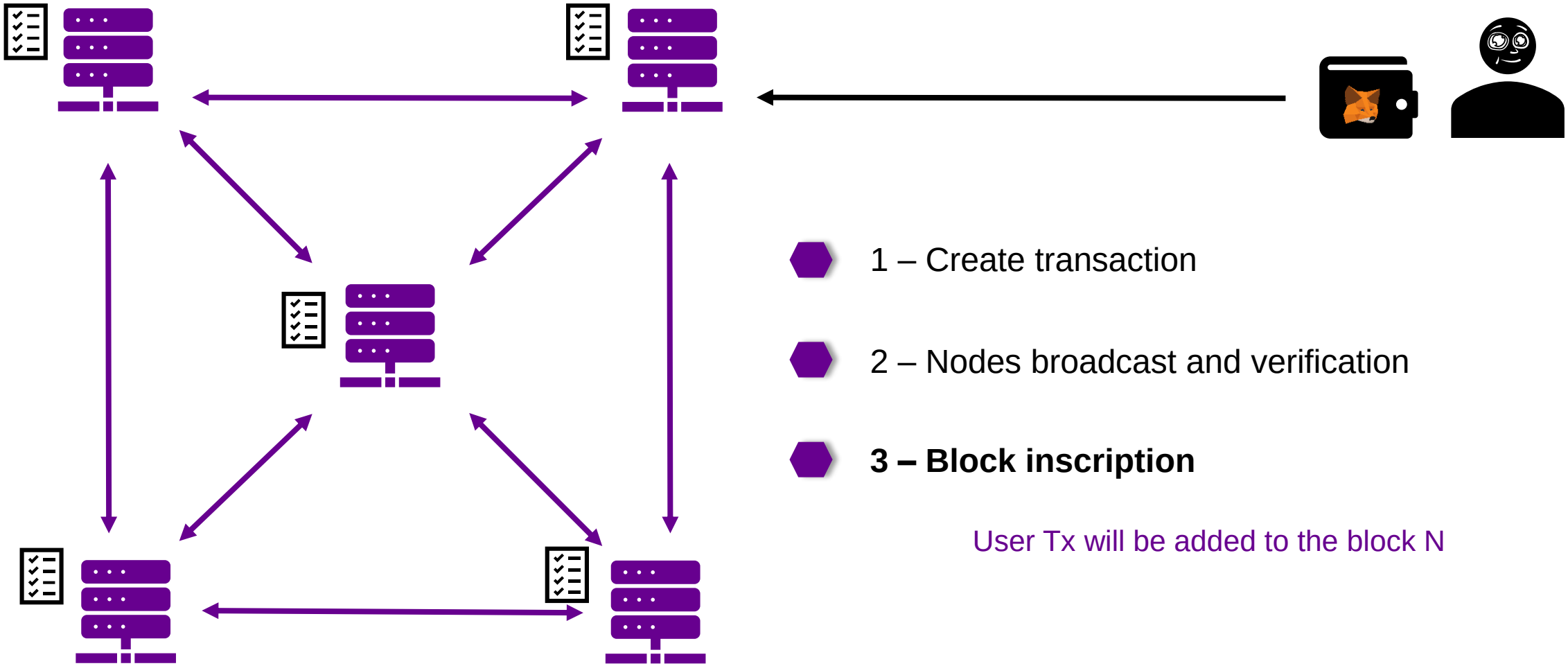
- 1 – Create transaction Tx
- **2 – Tx broadcast and verification**

The node to which user is connected broadcasts **Tx** to other nodes.

All nodes then verifies the user **Tx**:

- Is the signature valid?
- User has enough balance?

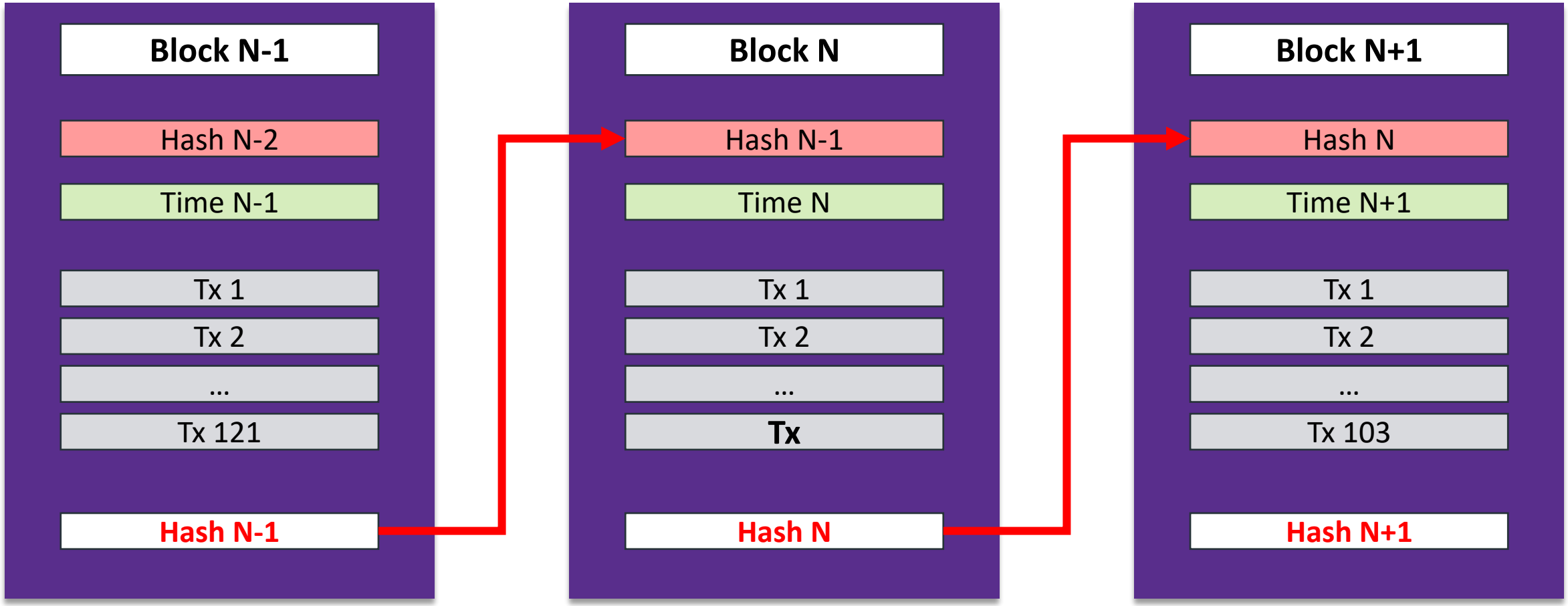
BLOCKS



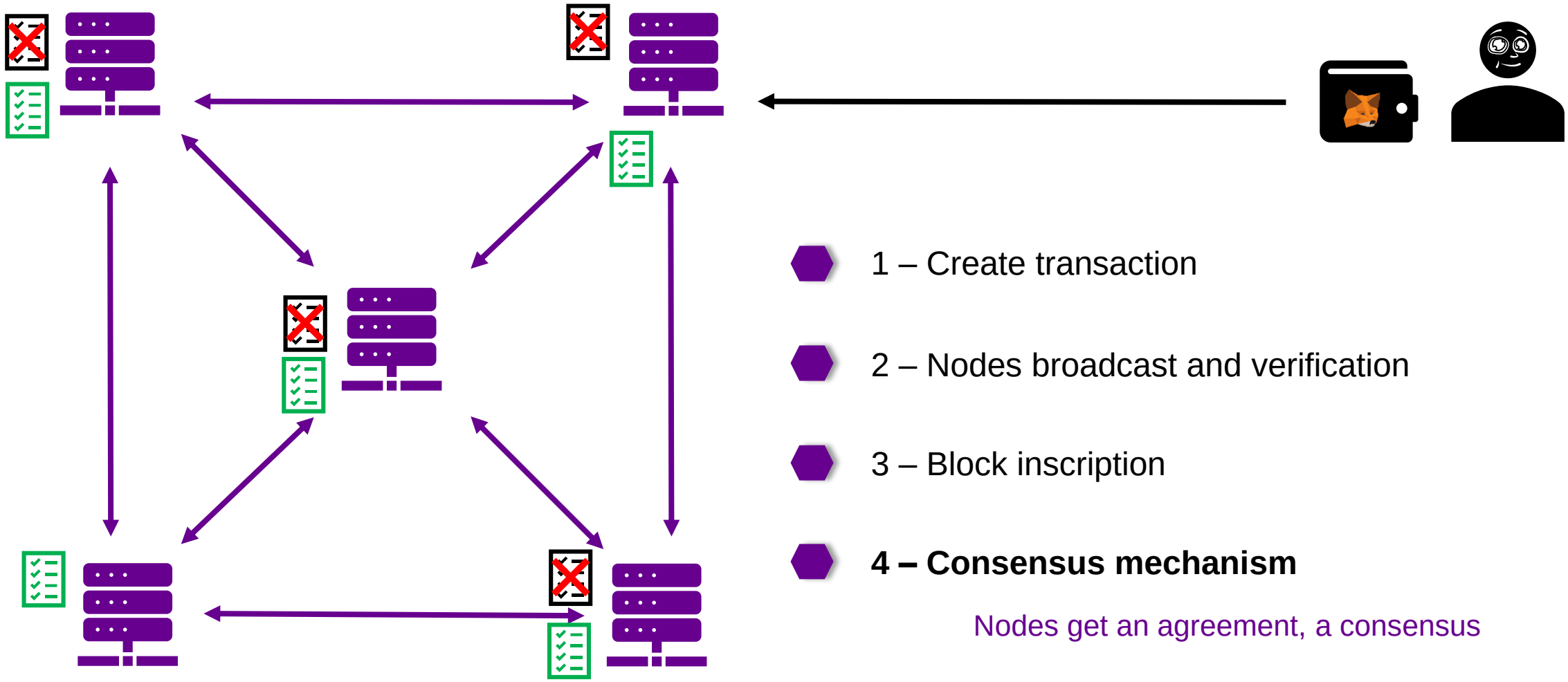
- 1 – Create transaction
- 2 – Nodes broadcast and verification
- **3 – Block inscription**

User Tx will be added to the block N

BLOCKS


Block N-1
Hash N-2
Time N-1
Tx 1
Tx 2
...
Tx 121
Hash N-1
Block N
Hash N-1
Time N
Tx 1
Tx 2
...
Tx
Hash N
Block N+1
Hash N
Time N+1
Tx 1
Tx 2
...
Tx 103
Hash N+1

CONSENSUS



- 1 – Create transaction
- 2 – Nodes broadcast and verification
- 3 – Block inscription
- **4 – Consensus mechanism**

  Nodes get an agreement, a consensus

  Tx is confirmed!

Consensus mechanism :

- Used to achieve a distributed agreement between all nodes

- There are two main mechanisms :

  - **Proof-of-Work**
    - Block hash must respect defined rules (like **X** first bits must be value 0)
    - First node that finds the right hash will broadcast it
    - A reward transaction is added in the block for the miner (the node)
    - Other nodes verify the hash of the block, and accept it if it respects the rules

  - **Proof-of-Stake**
    - Nodes lock a big amount of cryptocurrency to prove they have an interest in the network
    - The more you lock, the more you will verify
    - Rewards and penalty mechanisms

Ethereum time !

- Demonstration of a transaction
- Details of a block
- Consensus informations

04

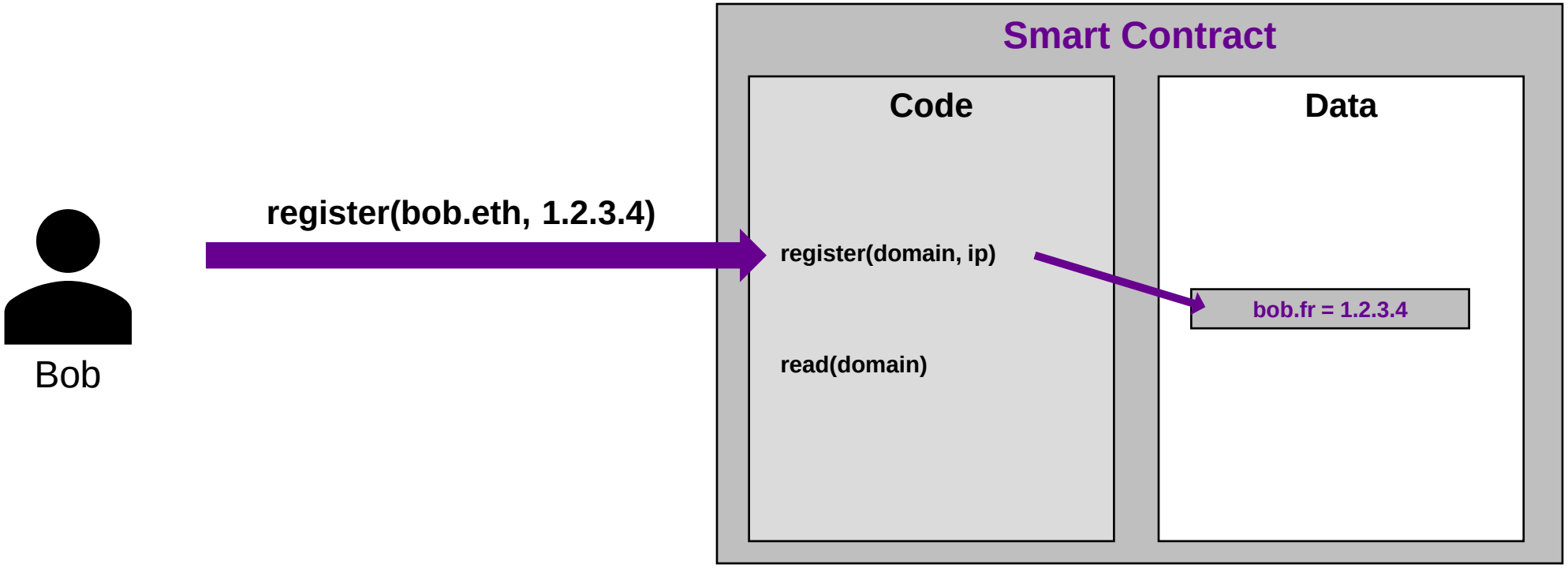
SMART CONTRACTS

Smart Contracts

- A contract is a program, on-chain
- Defines its own rules to use the blockchain storage
- A lot of use-cases !
  - Decentralized Finance - DeFi
  - Games
  - Voting
  - And more…

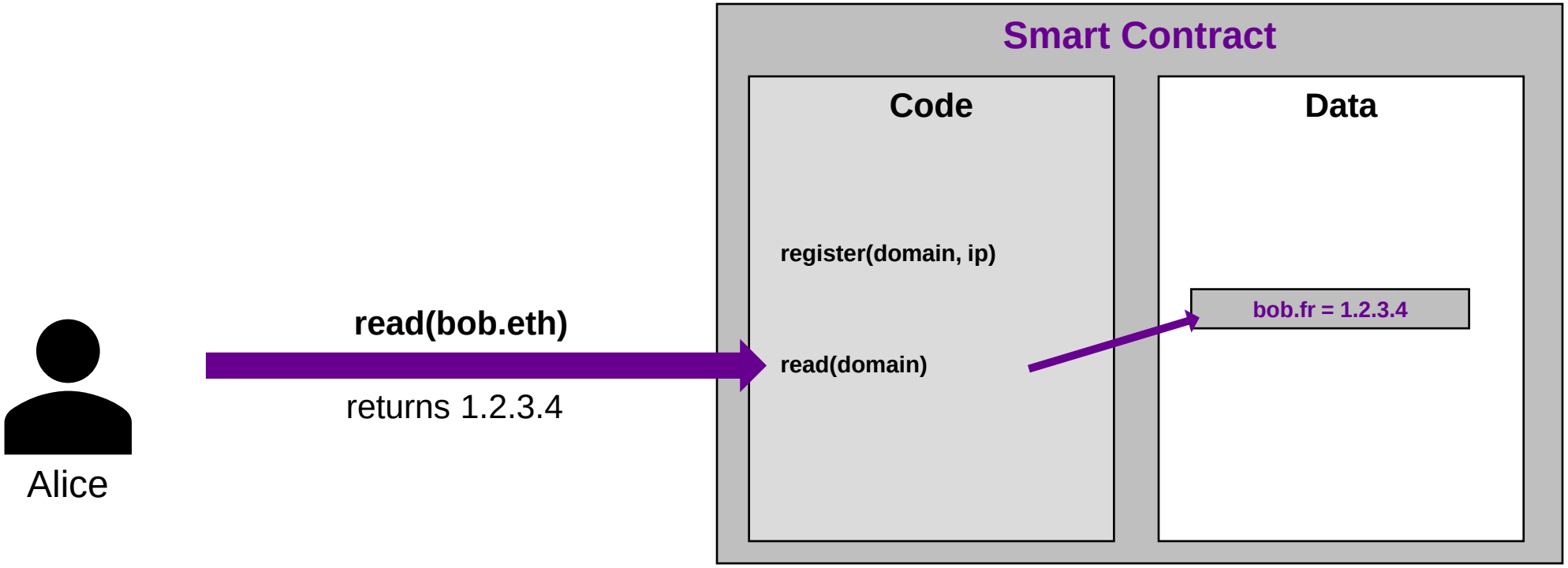
DNS use-case

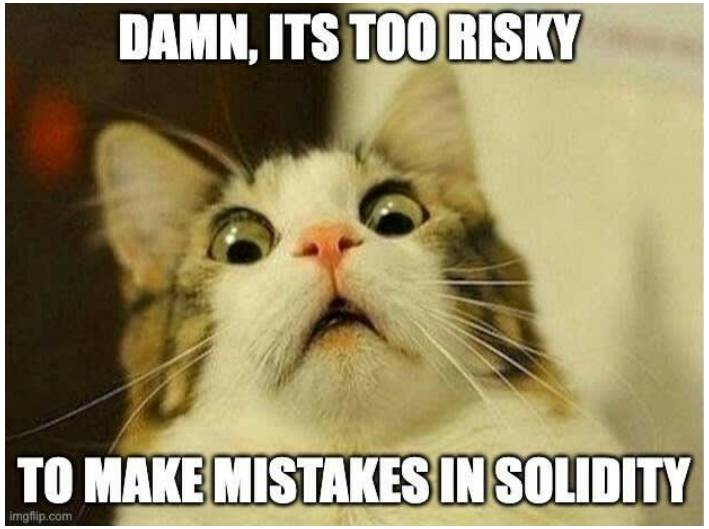
- Register a domain name



DNS use-case

- Read a domain name





Ethereum time !

- Introduction to Solidity
- DNS Example: Ethereum Name Service
- NFT Example: Bored Ape Yacht Club
- Game Example: Axie Infinity

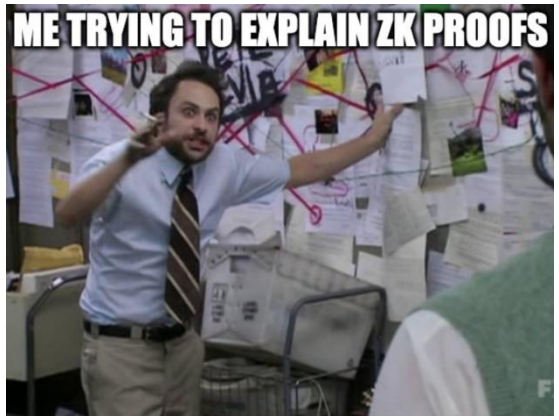
More details…

- Build a 100% decentralized website with IPFS and ENS
- Web3 hacks explained
- Solidity programming course: CryptoZombies
- Ethereum Layer 2

05

ZERO KNOWLEDGE

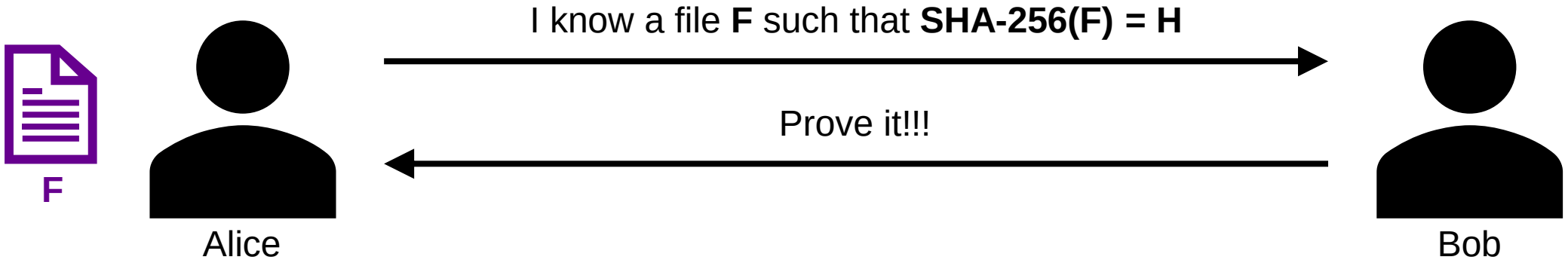
Zero Knowledge Proofs

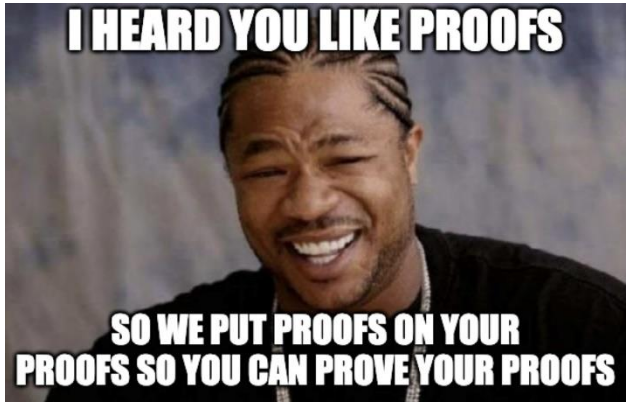


- Prove a statement, without telling how it is true
- Prover can’t create fake proofs
- Verifier can’t retrieve additional informations than the fact that it is true

Zero Knowledge Proofs

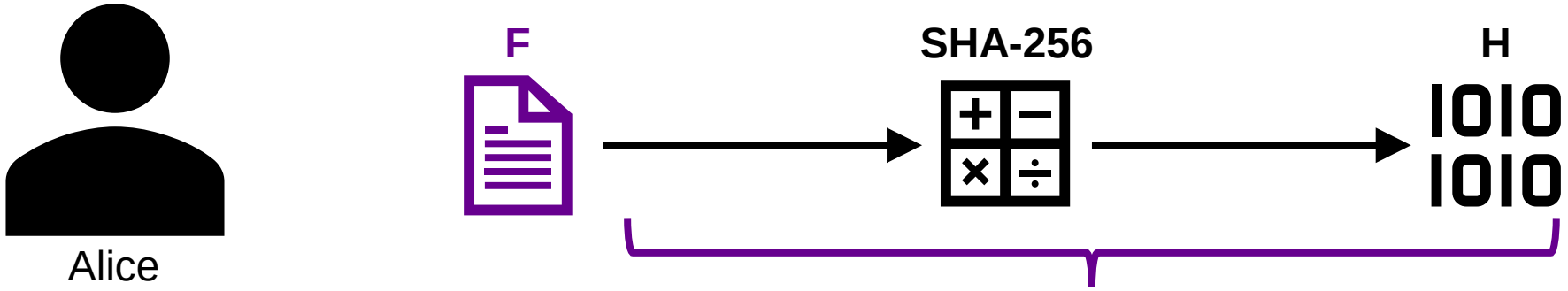
- **How does it work** - Hash function example





Zero Knowledge Proofs

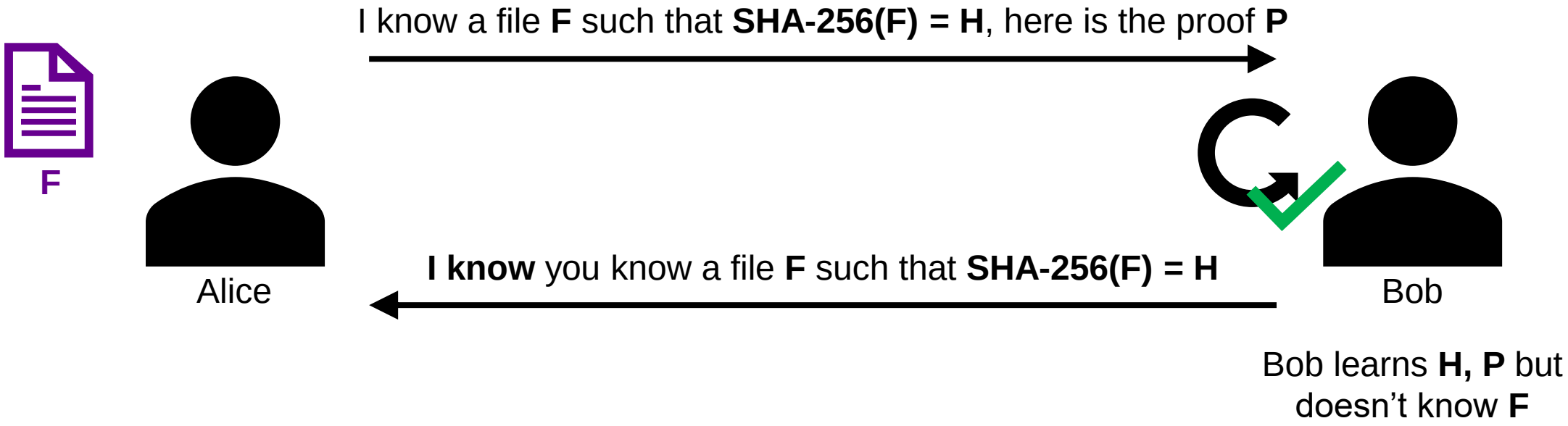
- **How does it work** - Hash function example
  - Proof generation



Create proof of computation with R1CS

Zero Knowledge Proofs

- **How does it work** - Hash function example



Zero Knowledge Proofs

- Use-cases
  - Private transactions for blockchains and banks
  - Proof of Identity
  - Proof of Passwords
  - And more…

END

THE END