Week 14: Cryptography Applications (P2)

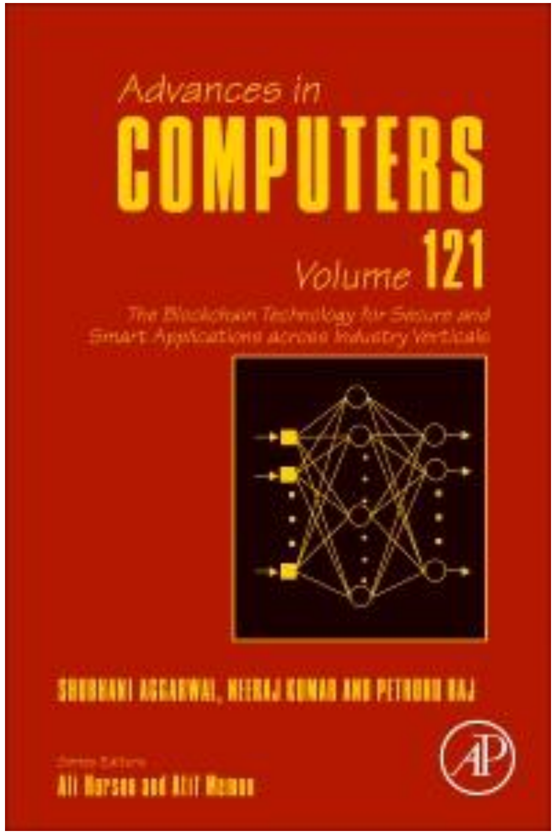
PhD. Ngoc-Tu Nguyen
tunn@uit.edu.vn

Outline

- Network secure protocols
  - Authentication;
  - Key agreemen;
  - IPSec; SSL/TLS; SSH; Kerberos
- Blockchain network security
  - Motivation
  - Database structure;
  - Secure Transactions and consensus mechanism;
  - Network architecture;
  - Applications

Textbooks and References

- Referent book



Kumar, N., Aggarwal, S., & Raj, P. (2021). The blockchain technology for secure and smart applications across industry verticals. Academic Press.

Outline (P2)

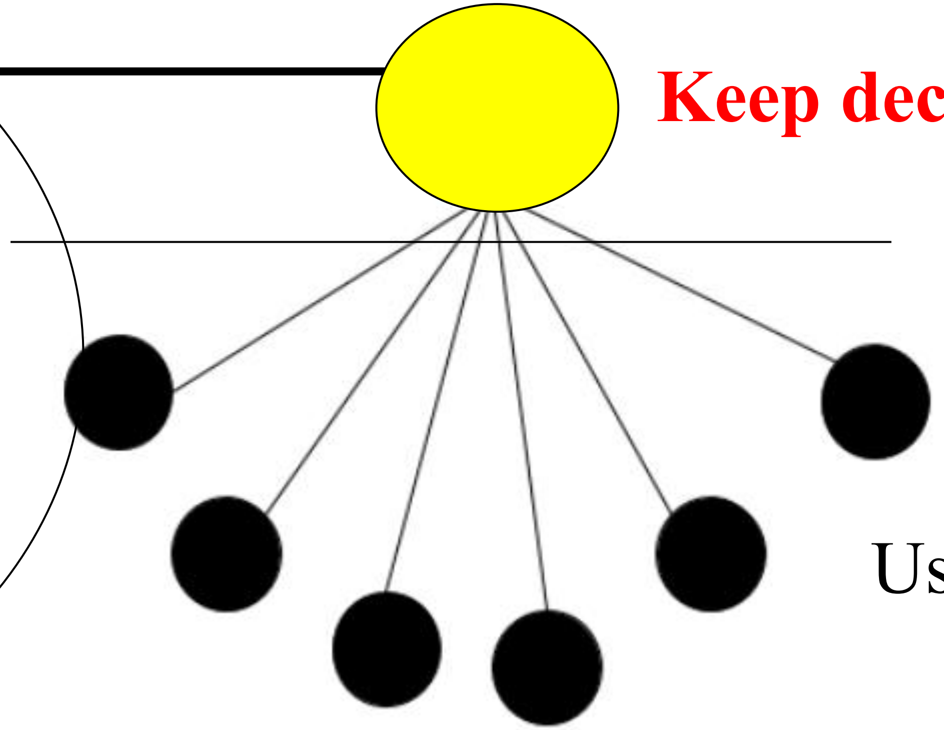
- Blockchain network and applications
  - Motivations
  - Hash-based secure storage (immutable database)
  - Signature-based authentication and verification (users/events):
  - Consensus machanism and coin mining (majority-rule security)
  - Blockchain architecture: a bitcoin case study;
  - Next generation: Transaction protocol (smart contract) and distributed applications
  - Implementation and application sectors

Motivations

Centralized systems

Who and how?

- Create data
- Store data
- Update/maintain database(add/modify/delete)
- User access control



Keep decision rights

User nodes

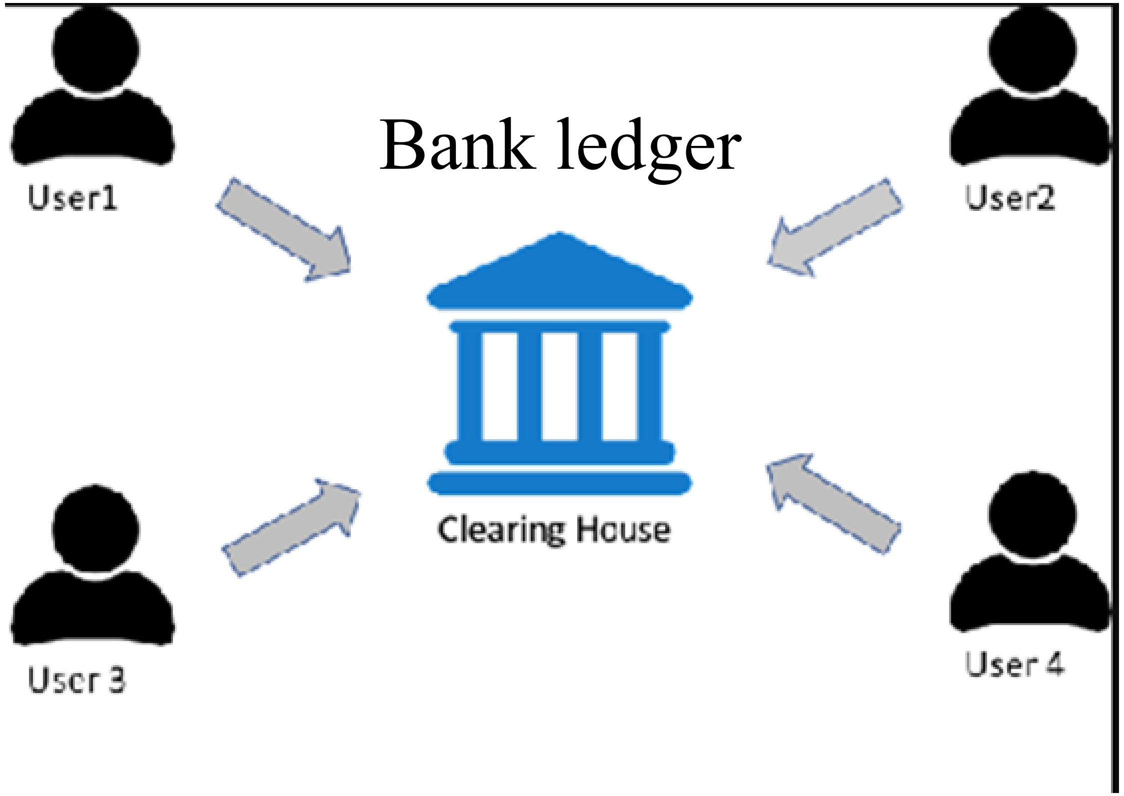
Bank systems (for example)

Only one party have decision rights!

Motivation

User's records: transaction history, account balance,…

Bank ledger



The bank controls everything!

single point of risk

(dishonest, compromised)

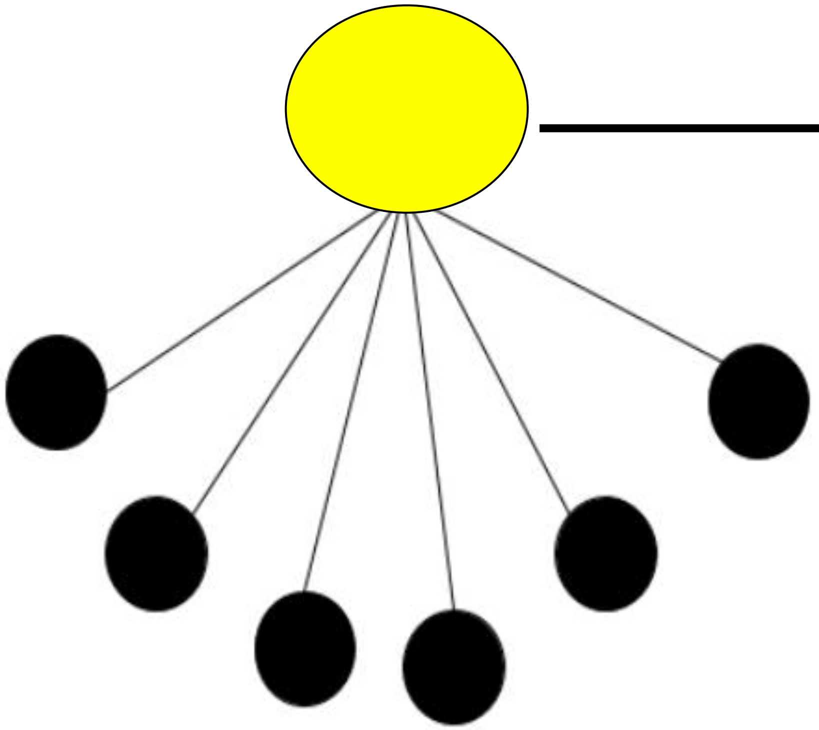
Motivations

Centralized authorities

Only one party have decision rights!

- We don't know what they do but have to trust!
- Single point of risk (dishonest, compromised)

Centralized systems



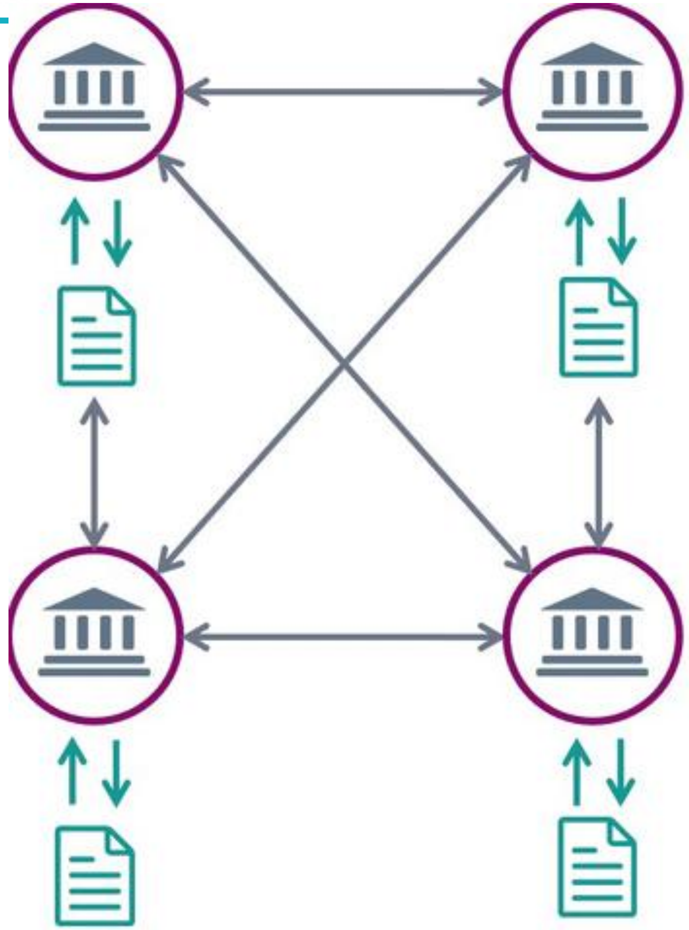
- Create data
- Store data
- Update/maintain database(add,modify,delete)
- User access control

Trust or distrust (spoofing)?

Secure? (integrity, privacy transparency, …)

Motivations

Remove the single point of control



Distributed rights systems

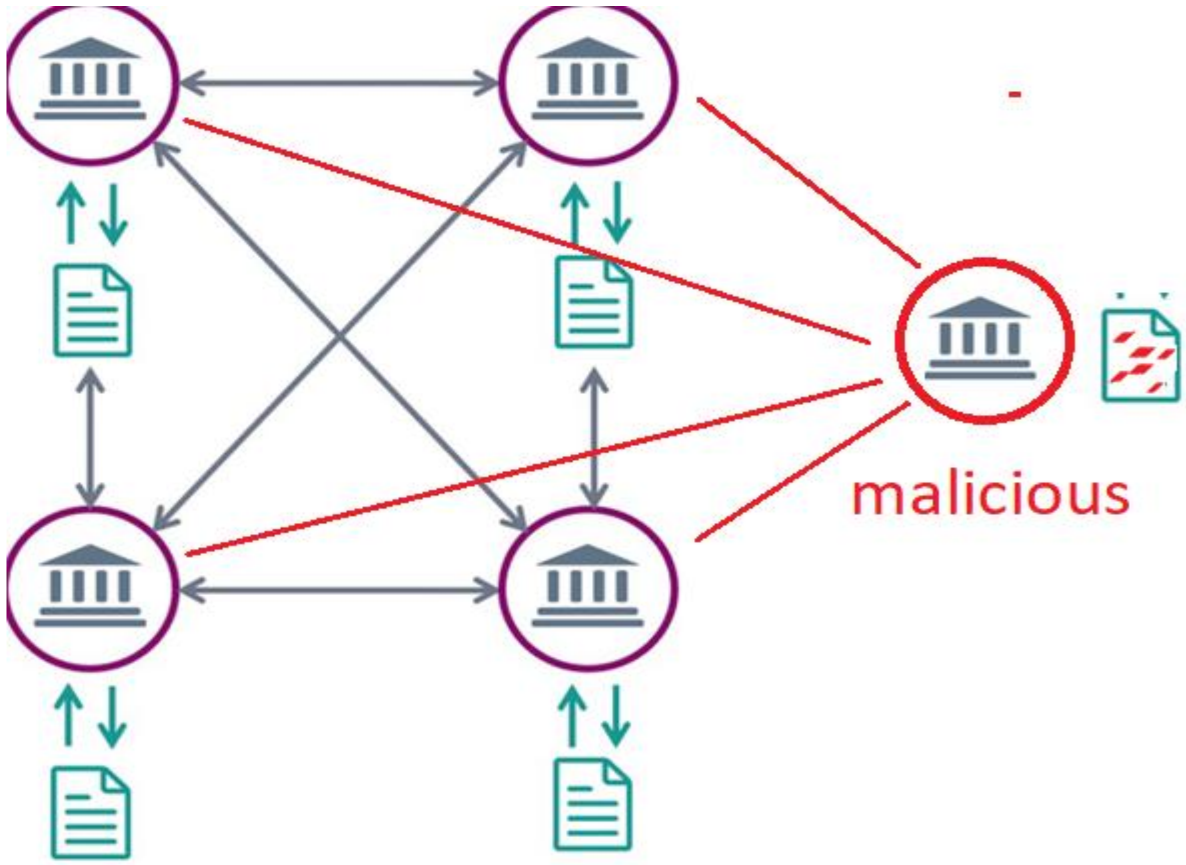
- **Decision rights:** multiple nodes cooperate to make decisions for new events (~ votes in parliament)
- **Secure verfication and storage:** multiple locations (all full nodes)

How to verify data, securely synchronize or record new events with error-tolerant (appear fake nodes)

- **Verifying data/transations (security features)?**
- **Securely synchronize data (consensus machanism)**
  - ✓ **Create, verify, synchronize new events?**

Ex. User's record: transaction history, account balance,…

Fault-tolerant mechanism



- **How to ?**
  - ✓ Verifying new events/transations (data authentication, other security features)
  - ✓ Securely storage, verify, synchonize new events (all nodes): immutability, transability,

Fault-tolerant mechanism

Byzantine Fault Tolerance

- "Crash Fault Tolerance" (CFT) aims to guarantee the functioning of a distributed system in the presence of machines crashing or disconnecting and reconnecting at random
- "Byzantine Fault Tolerance" (BFT) aims to guarantee the ability of a group of nodes to achieve consensus in the presence of malicious actors who are trying to subvert the consensus process in their favour, or prevent consensus from being achieved.
- BFT is a much harder goal than CFT, and hasn't been solved mathematically at scale; BFT generally makes use of cryptographic signatures, etc.
- Approximations of BFT exist, but they all come with significant constraints and limitations attached

Outline

- Network secure protocols
  - Authentication;
  - Key agreemen;
  - IPSec; SSL/TLS; SSH; Kerberos
- Blockchain network security
  - Motivation
  - Database structure;
  - Secure Transactions and consensus mechanism;
  - Network architecture;
  - Applications

Immutable database (Integrity)

User's record:

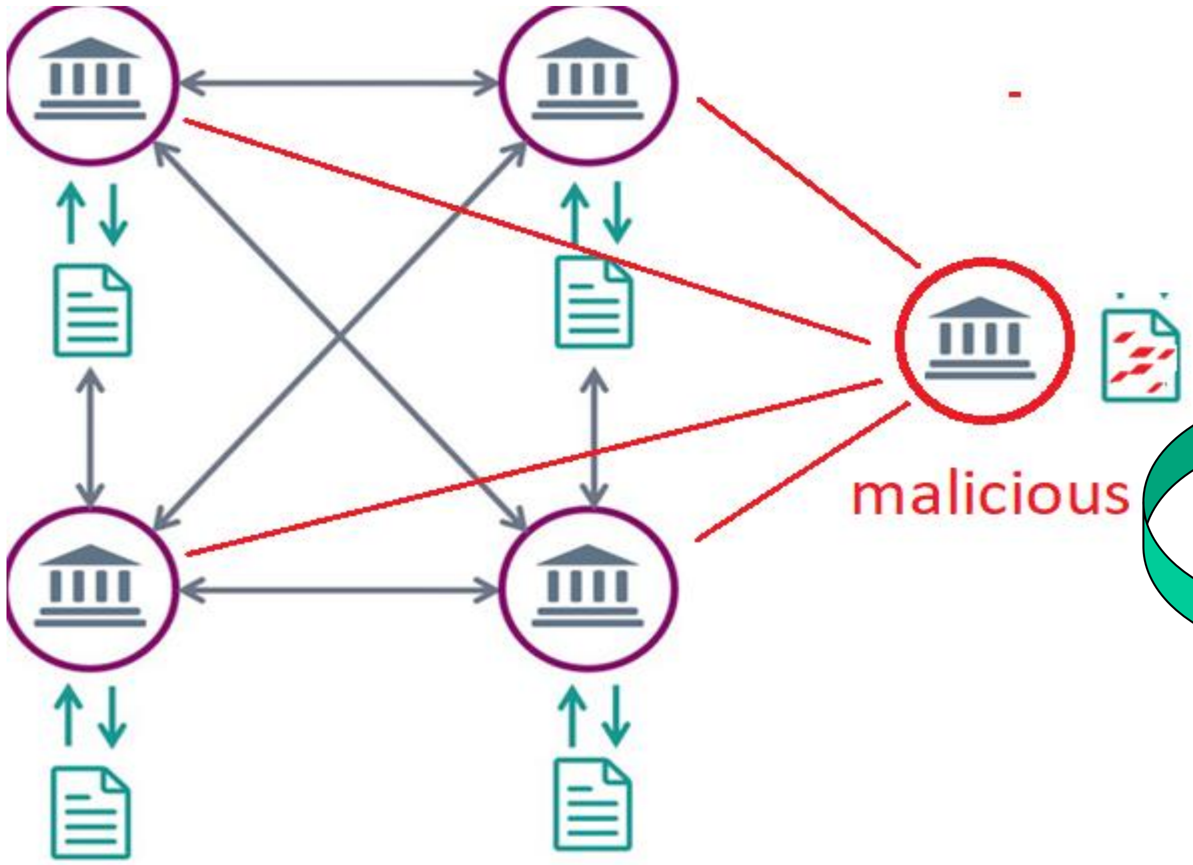
- Owner
- account balance,
- **transaction records**,
- …

→ need authenticate and secure storage (integrity, immutability)

✓ Common methods

- Signature-based: authenticate the related parties, and verify the integrity (original);
- Hash-based: Immutable storage
  - Hash trees
  - Hash chains

Immutable database (Integrity)

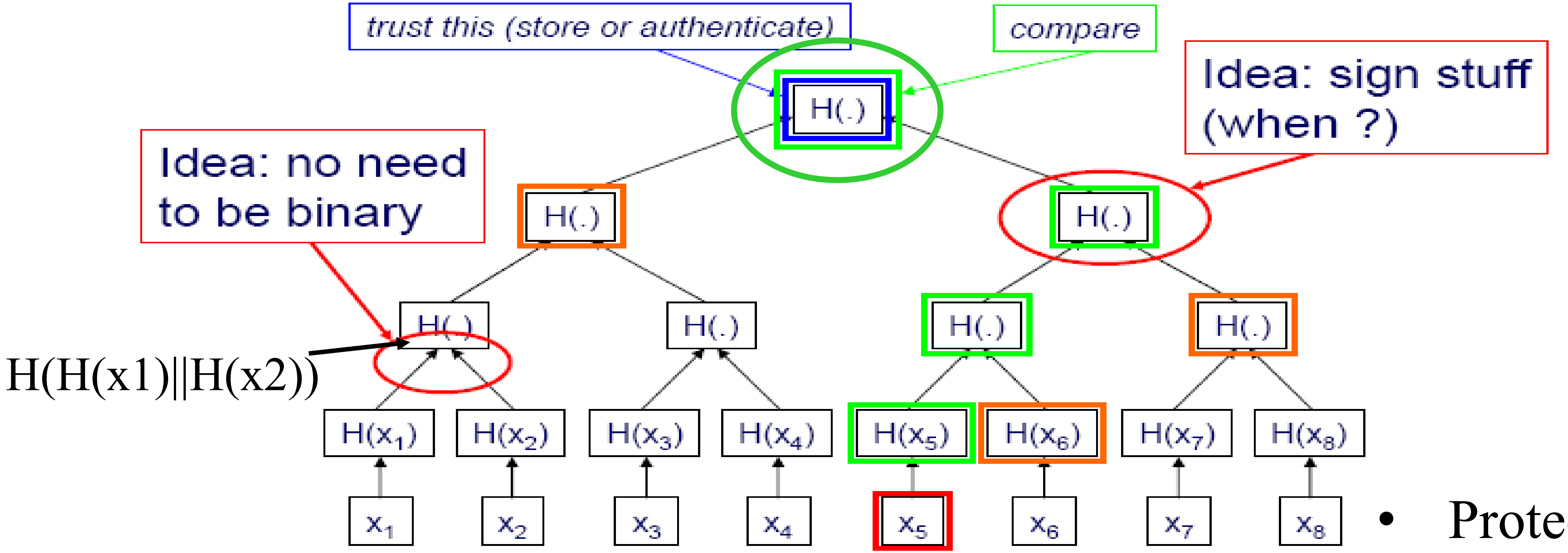


- **How to ?**
  - ✓ Verifying new events/transations (data authentication, other security features)
  - ✓ Securely storage, verify, synchonize new events (all nodes): immutability, transability,

Hashchain + Merkle hash tree = blockchain

Immutable database (Integrity)

- Merkle hash tree
  - → Protect one root

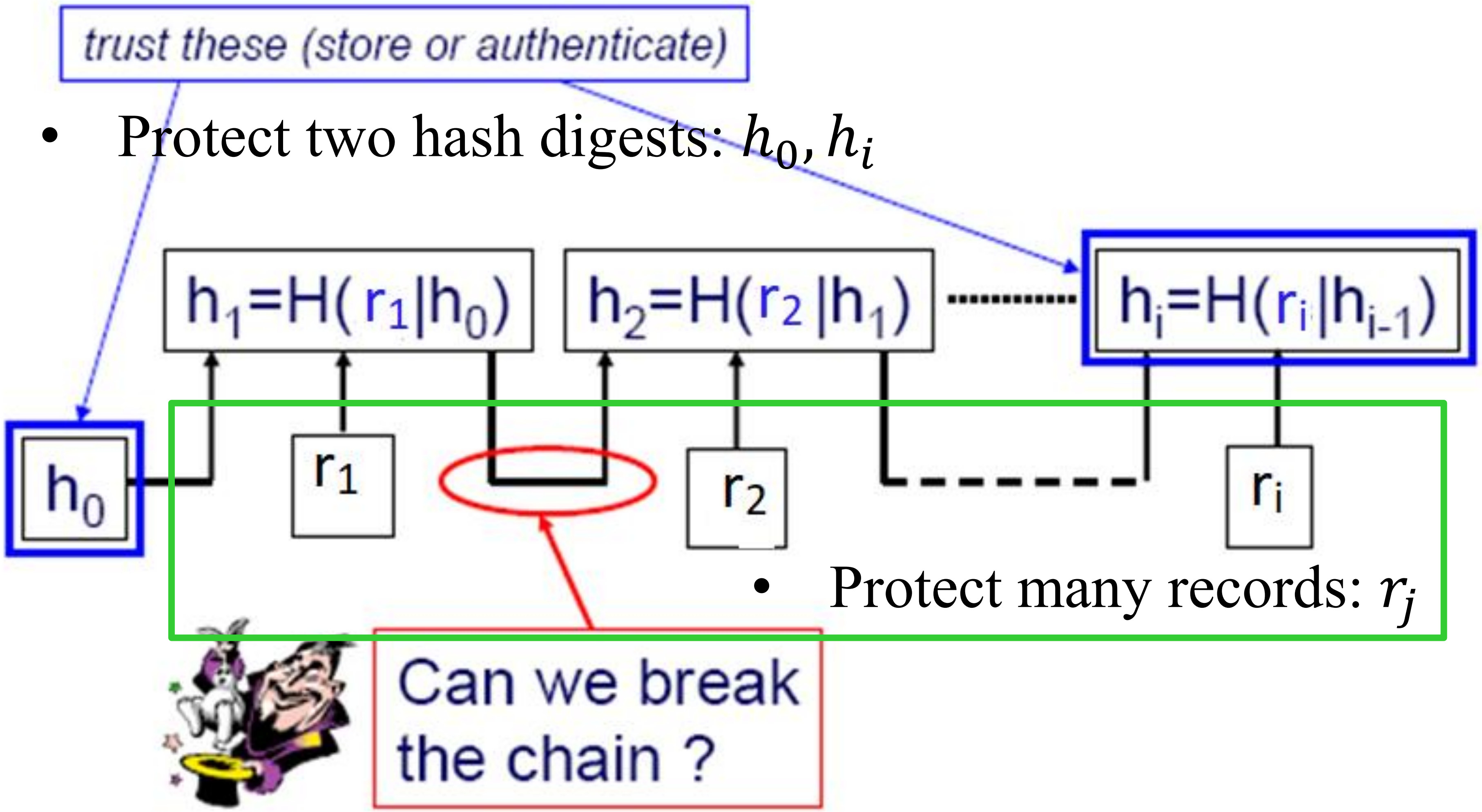


- Protect many leafs

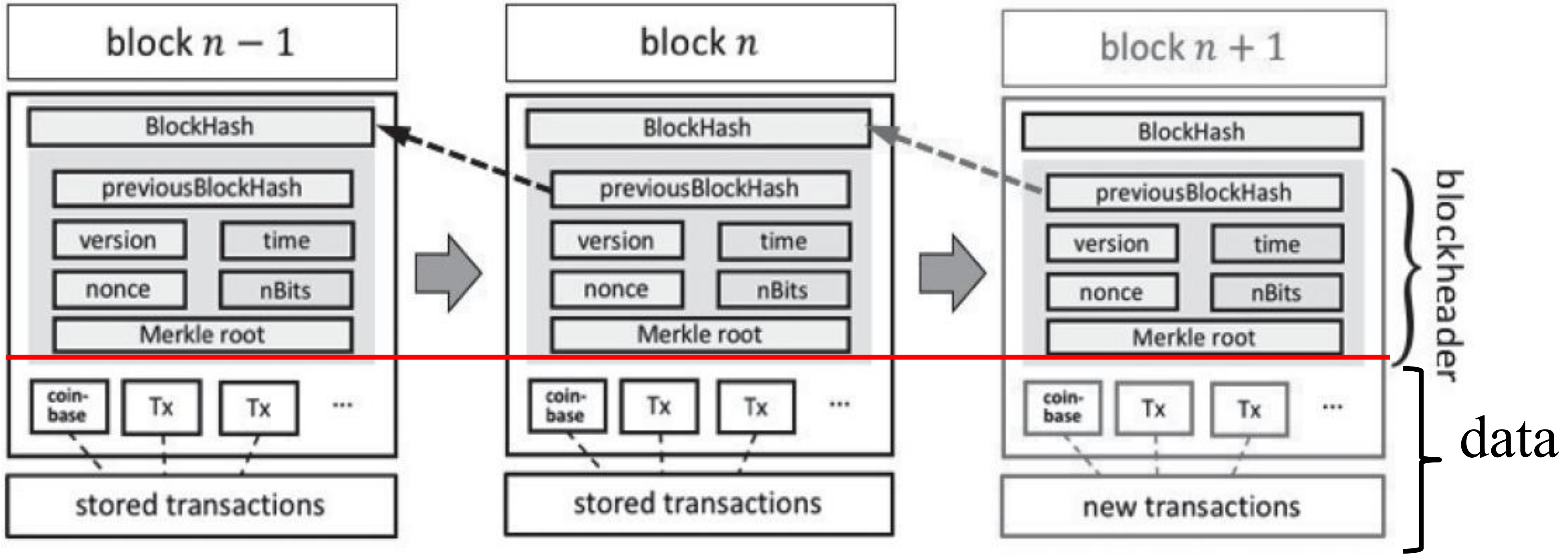
- "||" denote the string concatenation;
- Can be stored with tree like structure : index, xml

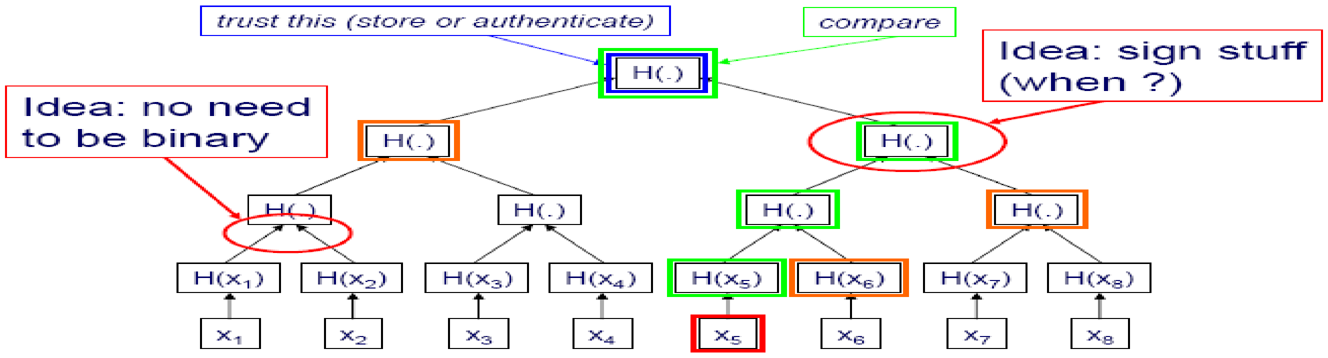
Immutable database (Integrity)

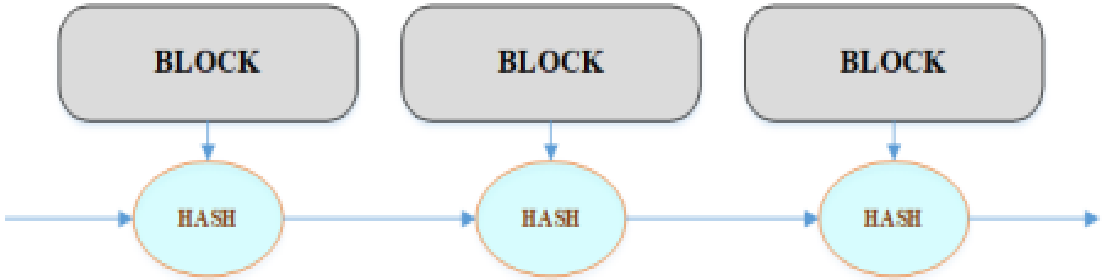
- Hash chains



Bitcoin database: a case study

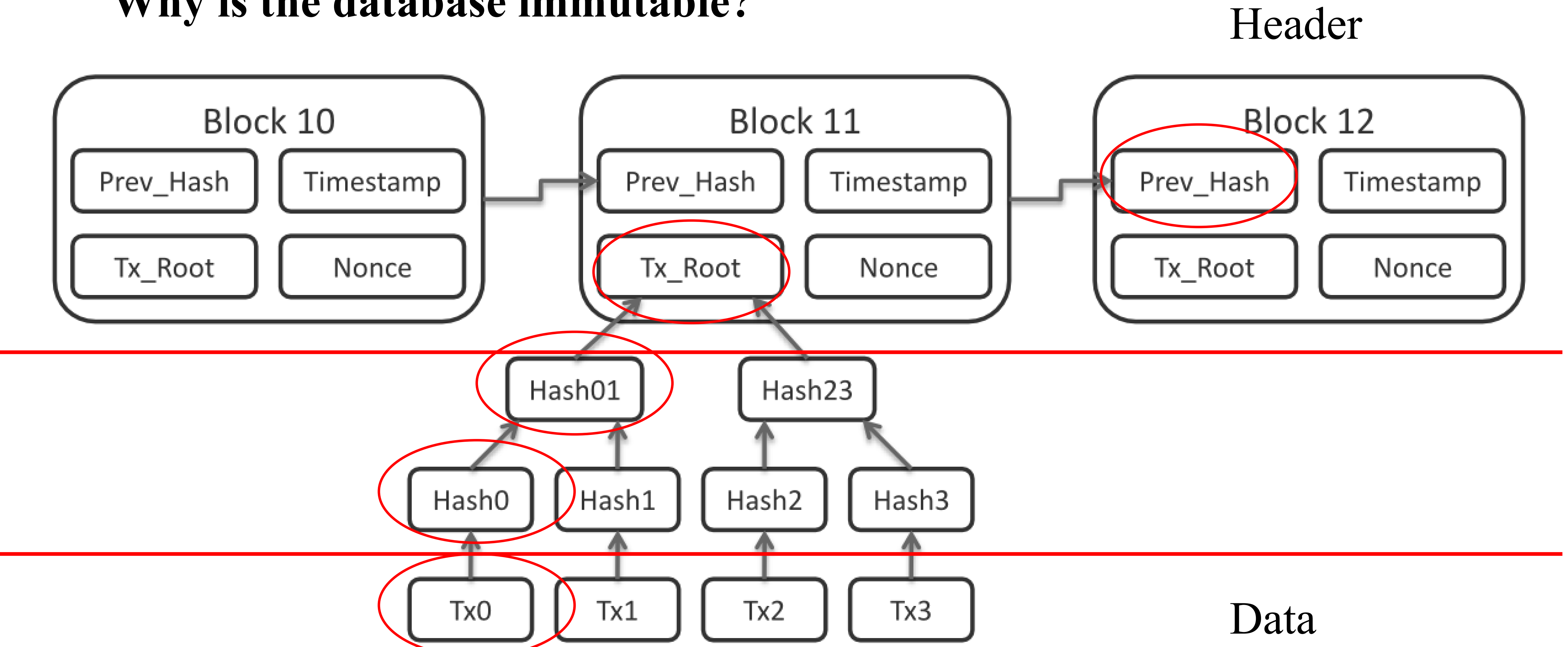






Bitcoin database: a case study

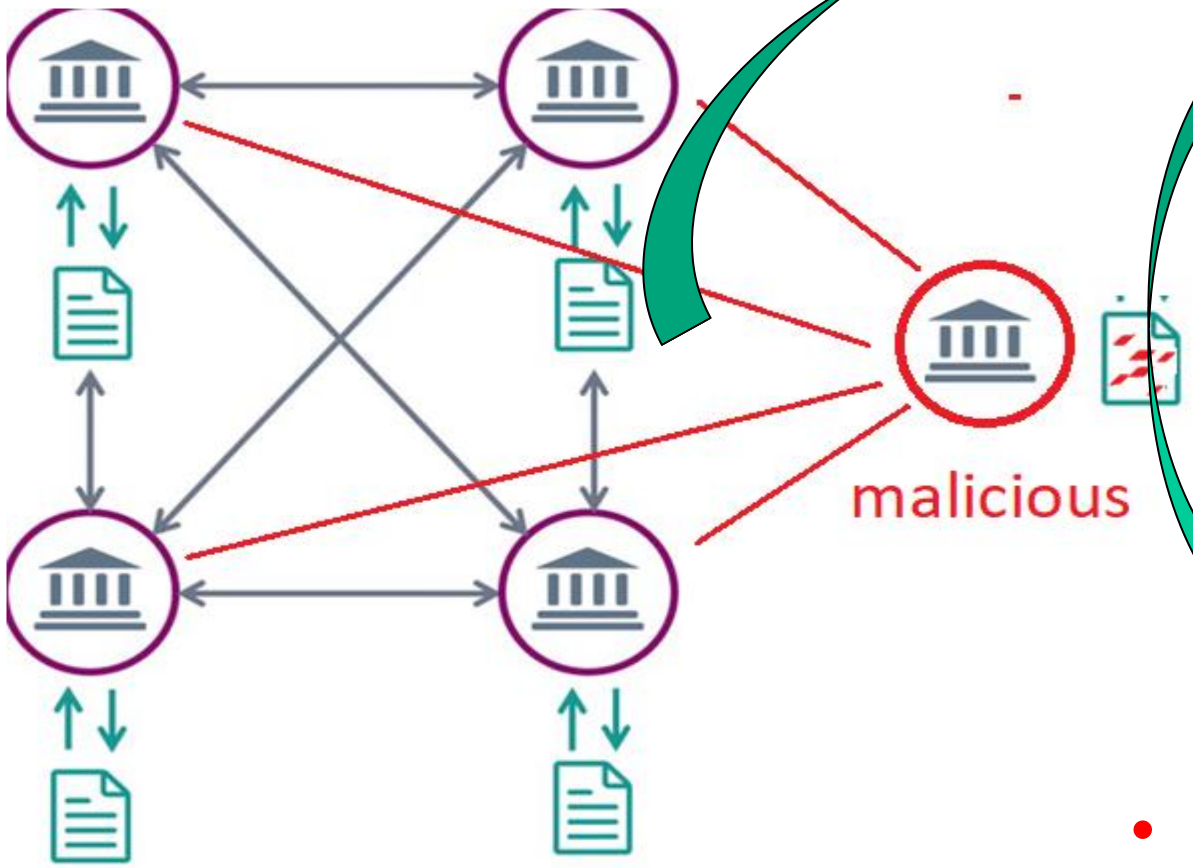
Why is the database immutable?



Outline

- Network secure protocols
  - Authentication;
  - Key agreemen;
  - IPSec; SSL/TLS; SSH; Kerberos
- Blockchain network security
  - Motivation
  - Database structure;
  - Secure Transactions and consensus mechanism;
  - Network architecture;
  - Applications

Verifying transation data



- **How to ?**
  - ✓ Verifying new events/transations (data authentication, other security features)
  - ✓ Securely storage, verify, synchonize new events (all nodes): immutability, transability,

Signature-based: authenticate the related parties and data (sources, content, integrity);

- Transaction contents (input/output)?
- Node (user)/ Public keys?
- Sources/realated parites/oringials?

Verifying transation data: Transaction contents

- **Verify input-output (looking from database - historical transaction)**

Transaction 7957a35fe64f80d234d76d83a2a8f1a0d8149a41d81de548f0a65a8a999f6f18

INPUTS From		OUTPUTS To	
From (previous transactions Joe has received):			
Joe	0.1005 BTC	Output #0 Alice's Address	0.1000 BTC (spent)
		Transaction Fees:	0.0005 BTC

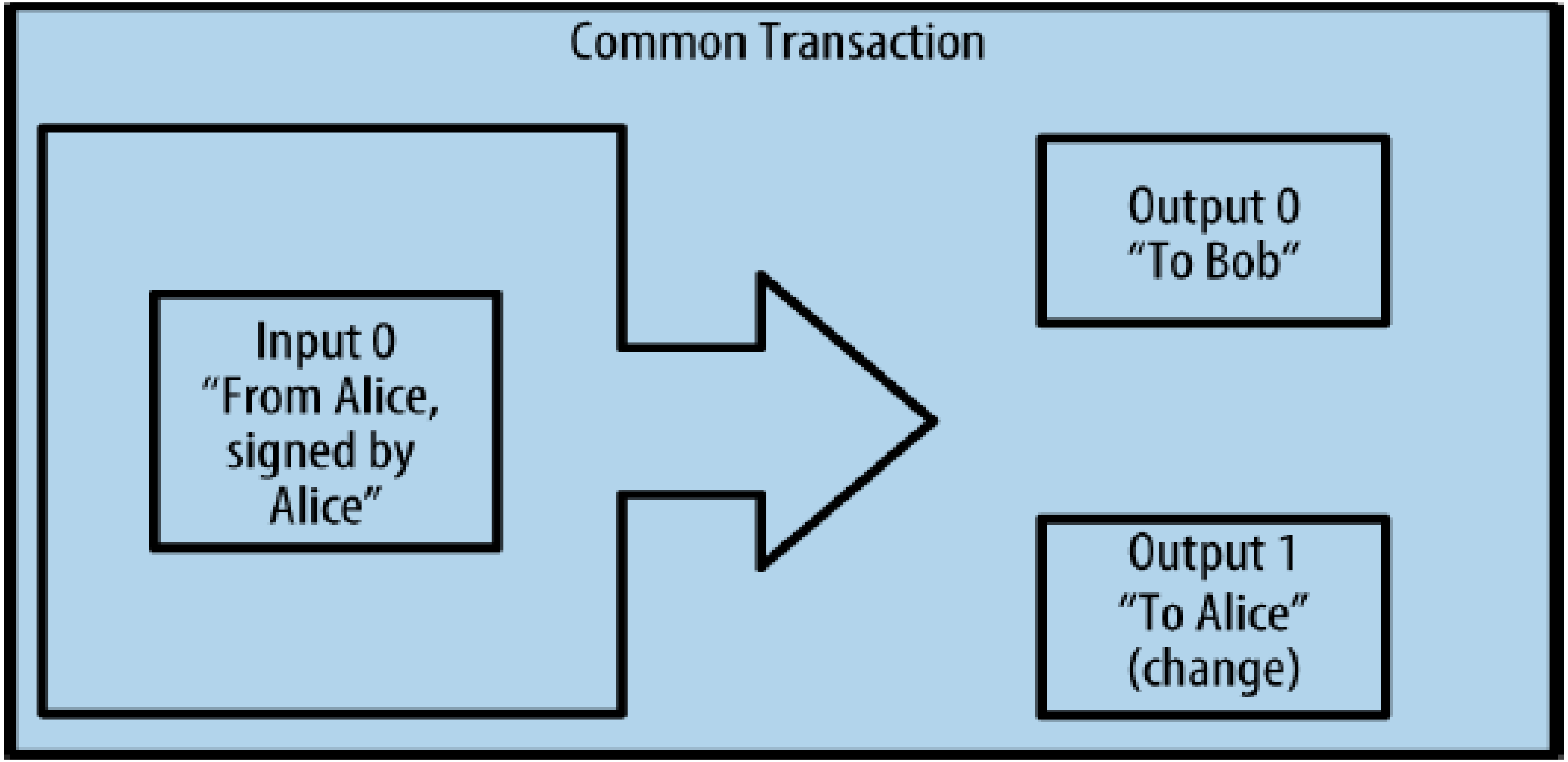
Transaction 0627052b6f28912f2703066a912ea577f2ce4da4caa5a5fbd8a57286c345c2f2

INPUTS From		OUTPUTS To	
7957a35fe64f80d234d76d83a2a8f1a0d8149a41d81de548f0a65a8a999f6f18 : 0			
Alice	0.1000 BTC	Output #0 Bob's Address	0.0150 BTC (spent)
		Output #1 Alice's Address (change)	0.0845 BTC (unspent)
		Transaction Fees:	0.0005 BTC

Transaction 2bbac8bb3a57a2363407ac8c16a67015ed2e88a4388af58cf90299e0744d3de4

INPUTS From		OUTPUTS To	
0627052b6f28912f2703066a912ea577f2ce4da4caa5a5fbd8a57286c345c2f2 : 0			
Bob	0.0150 BTC	Output #0 Gopesh's Address	0.0100 BTC (unspent)
		Output #1 Bob's Address (change)	0.0045 BTC (unspent)
		Transaction Fees:	0.0005 BTC

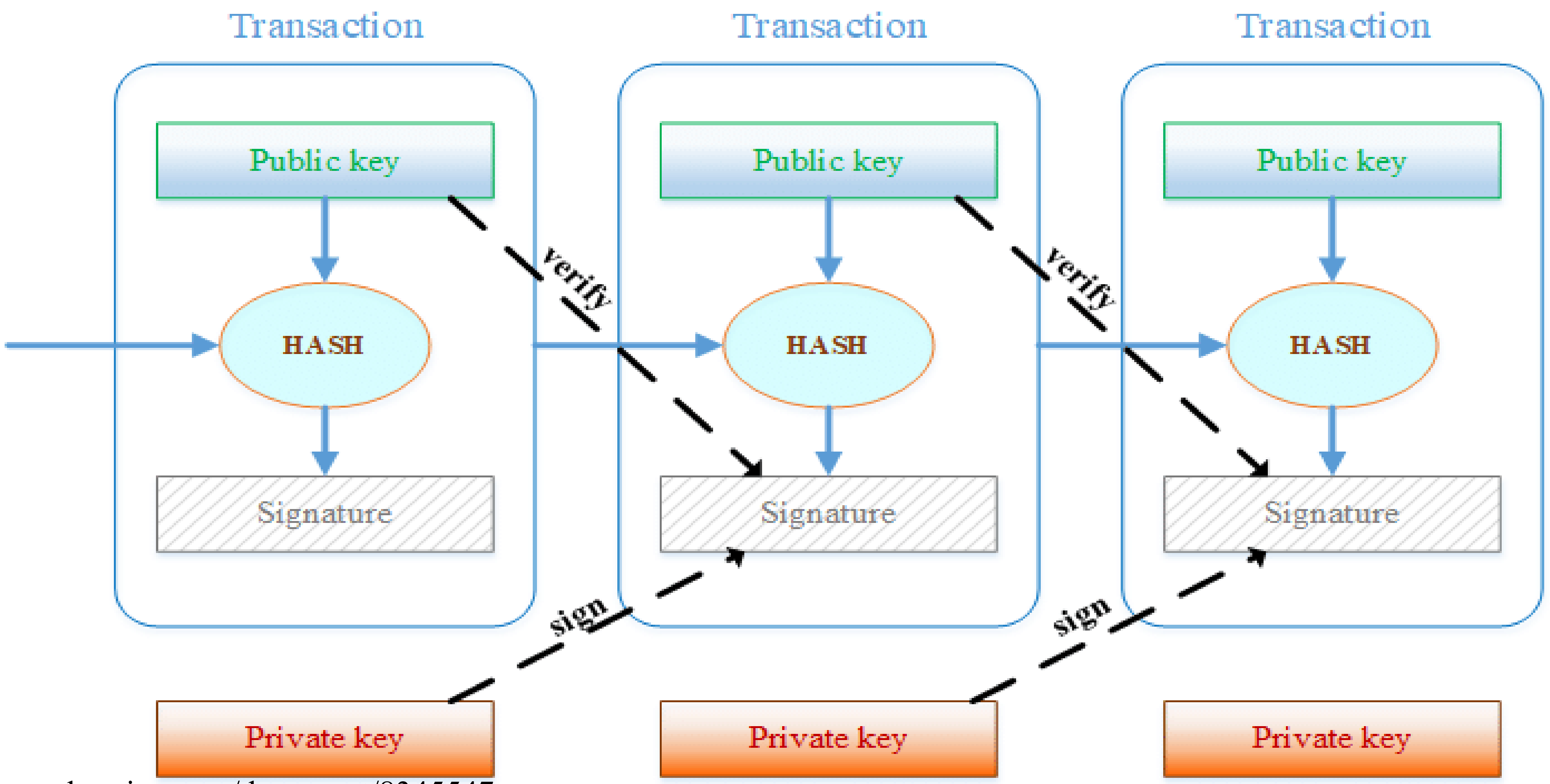
Verifying transation data: Transaction contents



The structure of transaction in a Bitcoin blockchain

Verifying transation data: Transaction contents

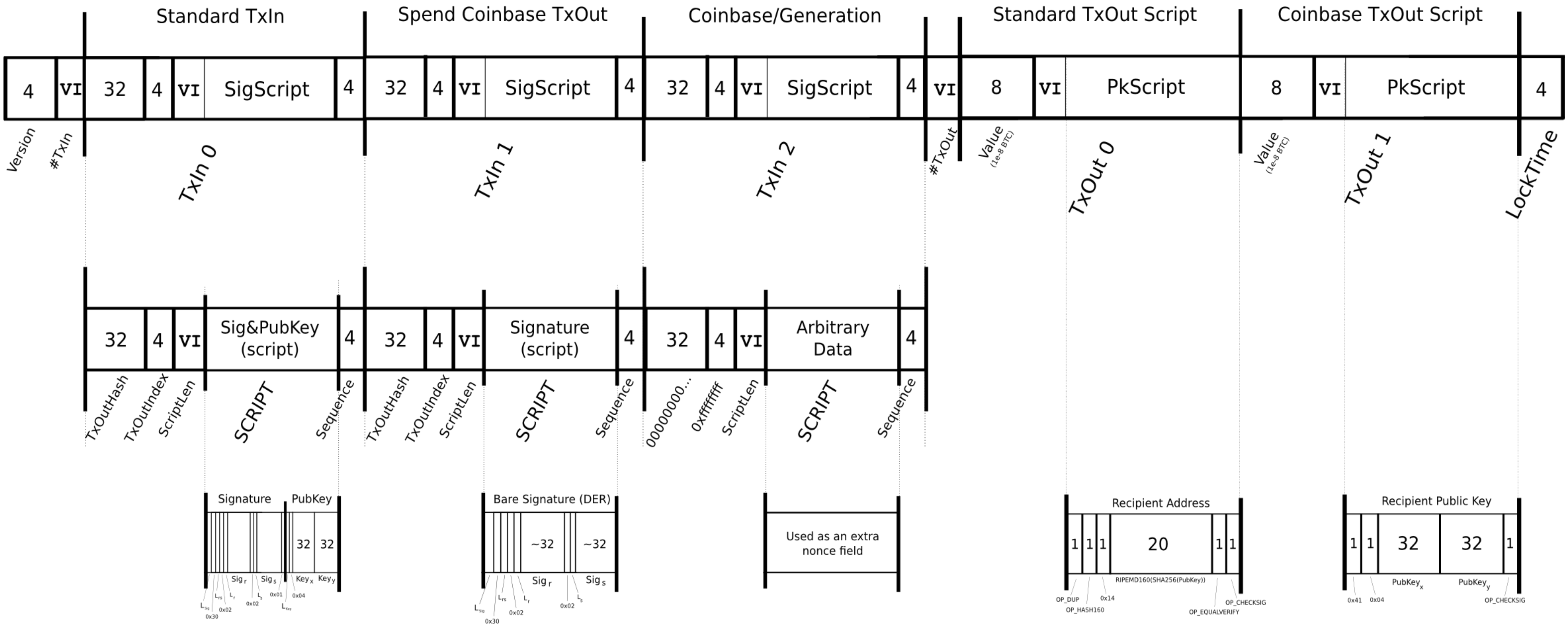
The structure of transaction in a Bitcoin blockchain



https://ieeexplore.ieee.org/document/8345547

Verifying transation data: Transaction contents

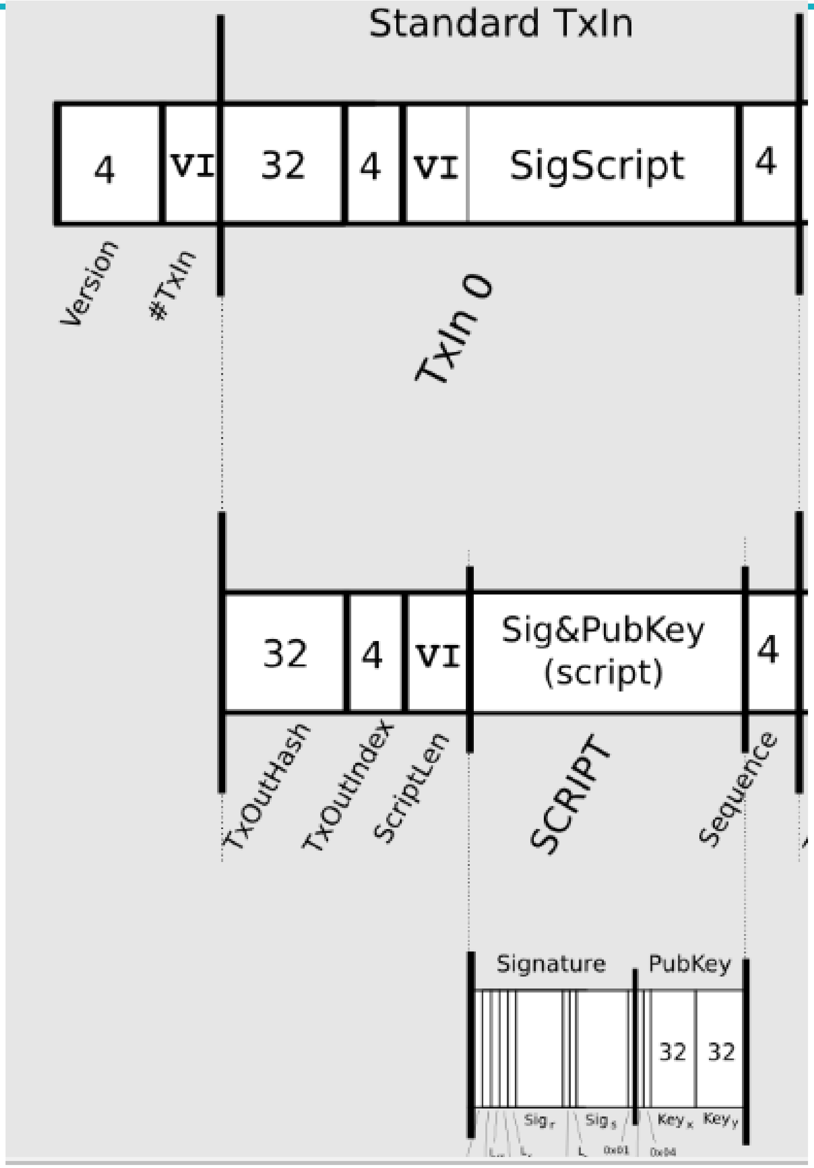
Transaction https://en.bitcoin.it/w/images/en/e/e1/TxBinaryMap.png

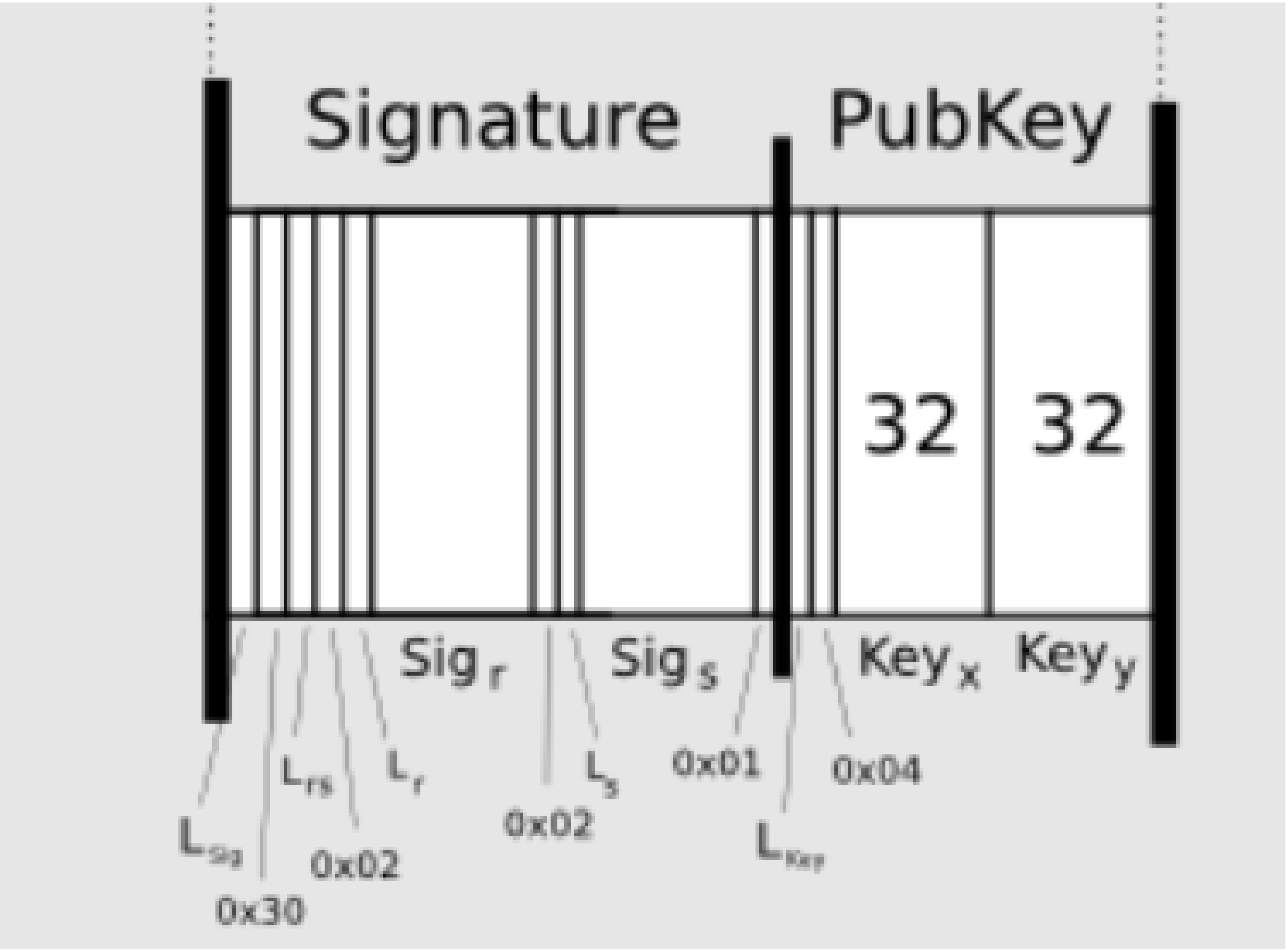


Scripts and DER encoding both use big-endian values, all other serializations use little-endian

etotheipi@gmail.com / 1Gffm7LKXcNFPrtxy6yF4JBoe5rVka4sn1

Verifying transation data: Transaction contents





Elliptic Curve Digital Signature Algorithm (ECDSA)

ECDSA parameters

- Prime number: p $(or\ f(x))$
- Curve coefficients: $a, b \in \mathbb{Z}_p$
- Base points: $G \in E(\mathbb{Z}_p)$
- The number $n = ord(\langle G \rangle)$
- The number $h = \dfrac{ord\ (E(\mathbb{Z}_p))}{n}$
- $H\colon \{0,1\}^* \to \{0,1\}^l,\ l = l(n)$

Bit length of n	Maximum Cofactor (h)	Comparable Security Strength
224 - 255	2^{14}	approximately $n/2$; at least 112 bits
256 - 383	2^{16}	approximately $n/2$; at least 128 bits
384 - 511	2^{24}	approximately $n/2$; at least 192 bits
≥ 512	2^{32}	approximately $n/2$; at least 256 bits

NIST.FIPS.186-5

Key generation (for signer)

- Secret key: $d \in_R [1, n-1]$
- Public key: Q= $d.G \in E(\mathbb{Z}_p) = (Key_x, Key_y)$

Key distribution: Curve, Q

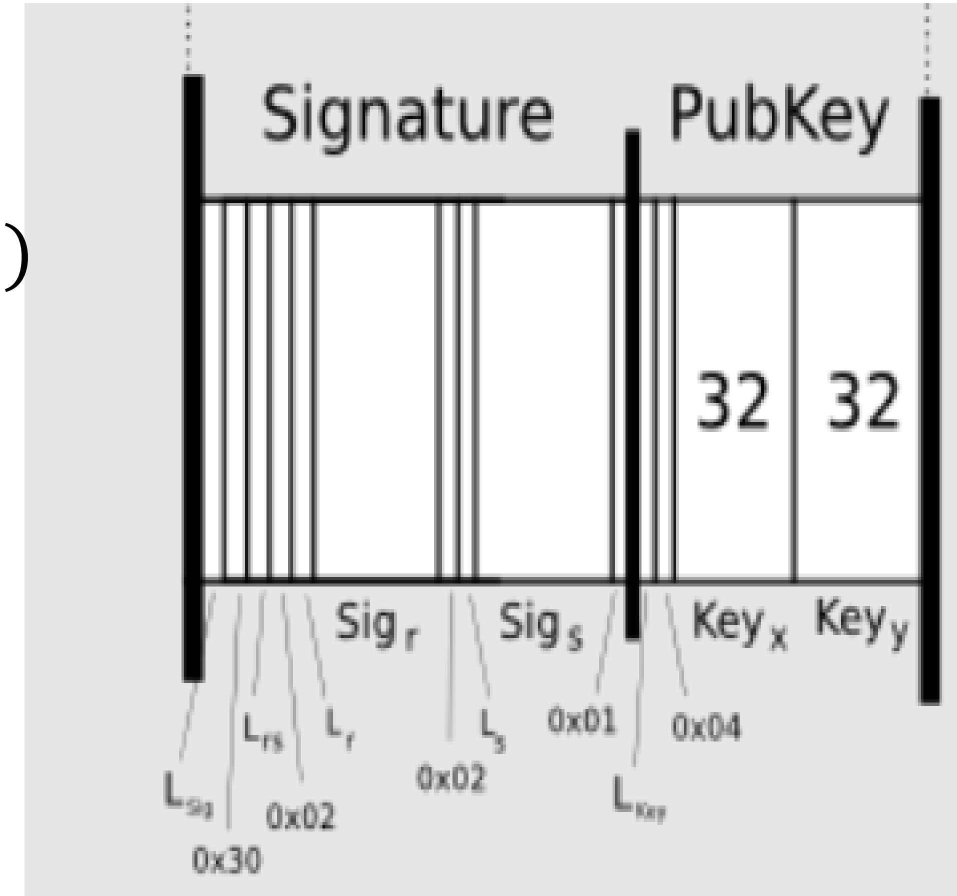
Elliptic Curve Digital Signature Algorithm (ECDSA)

Key generation (for signer)

- Secret key: $d \in_R [1, n-1]$
- Public key: Q= $d.G \in E(\mathbb{Z}_p) = (Key_x, Key_y)$

Signing (the message *m*)

- Choose secret for each message: $k \in_R [1, n-1]$
- Compute $R = k.G = (x_1, y_1), r = x_1$
- Compute $s = k^{-1}(H(m) + d.r) \bmod n$
- Output signature (r, s)



L_{sig}	0x 03	L_{rs}	0x 02	L_r	sig_r	0x 02	L_s	sig_s	0x 01	L_{key}	0x 04	key_x	key_y

Verifying transation data: Node (user)/ Public keys?

- **Bitcoin wallet address:** Pay-to-Public-Key-Hash (P2PKH)

Verifying transation data: Node (user)/ Public keys?

- **Bitcoin wallet address:** Pay-to-Public-Key-Hash (P2PKH)

$$A = RIPEMD160(SHA256(K))$$

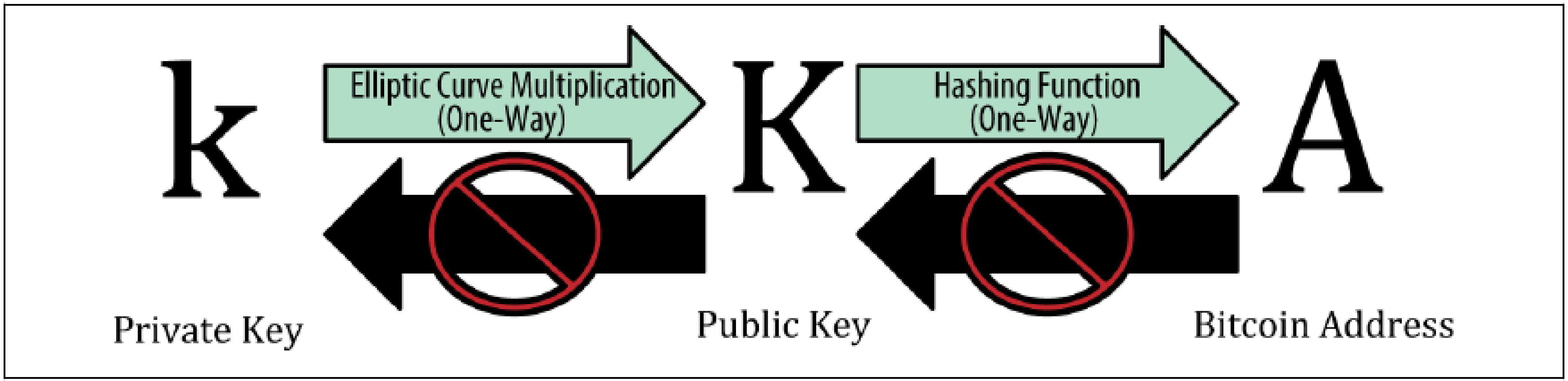
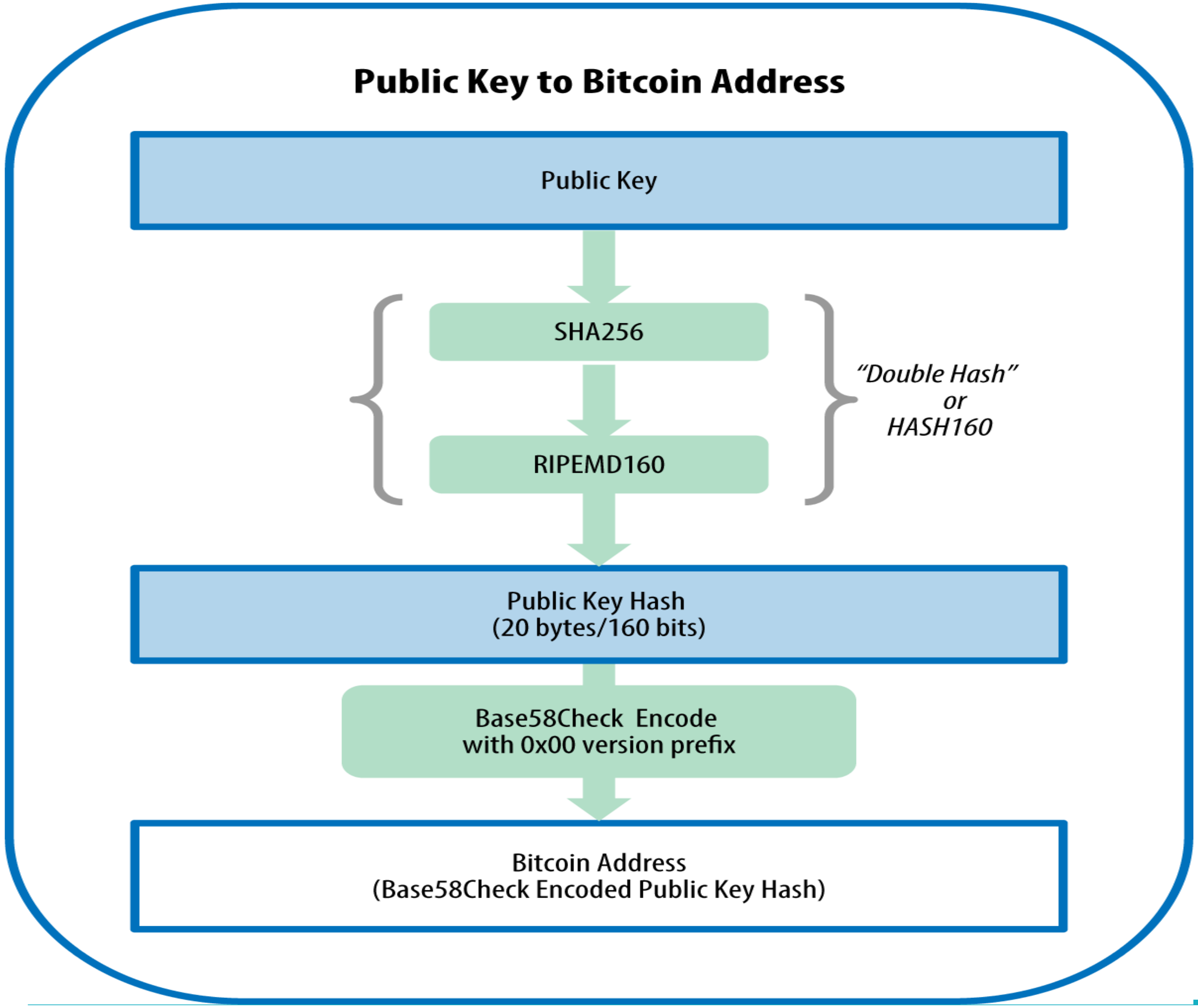


Figure 4-1. Private key, public key, and bitcoin address

hash functions	BLAKE2b, BLAKE2s, Keccack (F1600), SHA-1, SHA-2, SHA-3, SHAKE (128/256), SipHash, LSH (128/256), Tiger, RIPEMD (128/160/256/320), SM3, WHIRLPOOL

Verifying transation data: Node (user)/ Public keys?

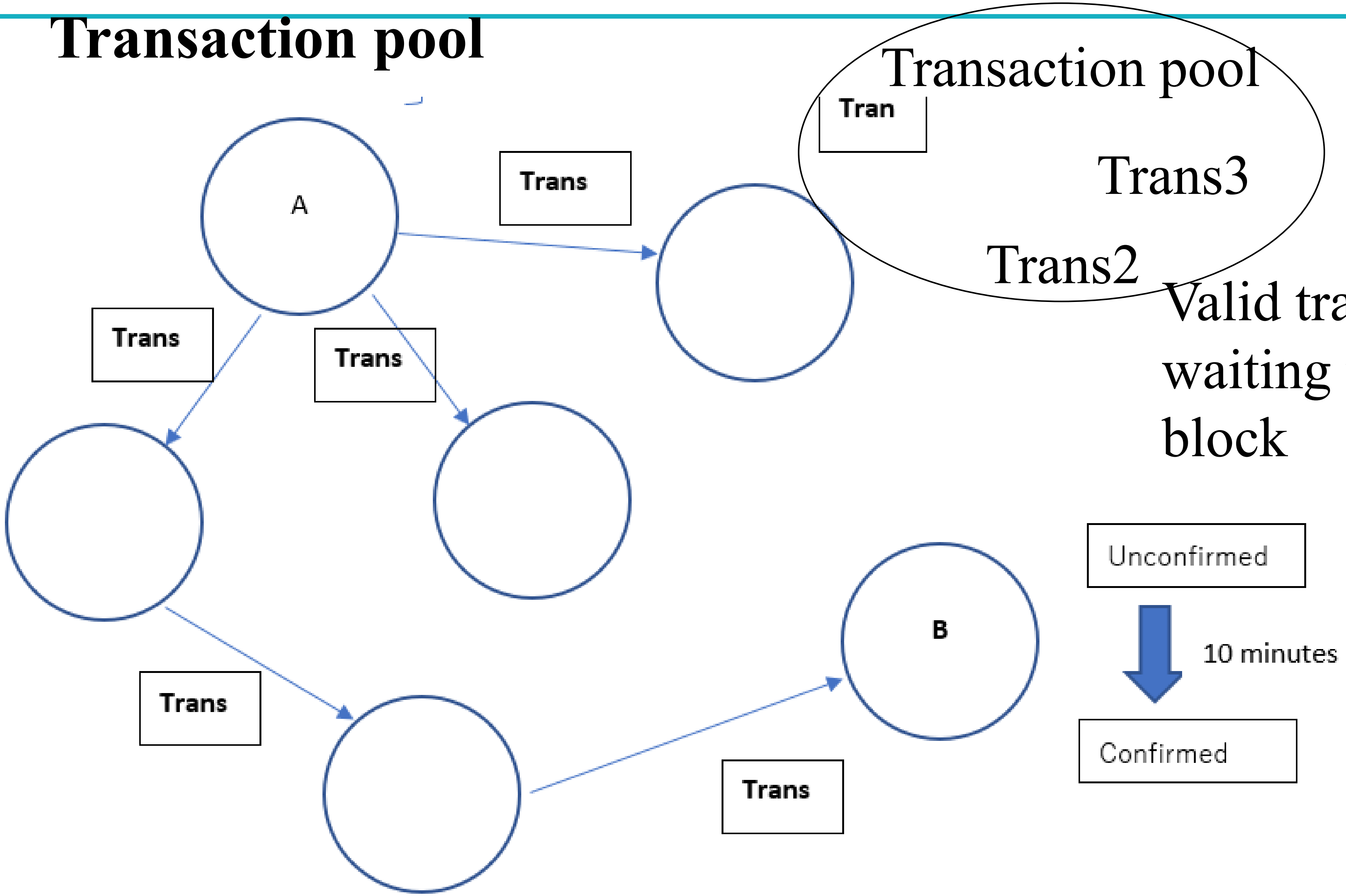
- **Bitcoin wallet address:** Pay-to-Public-Key-Hash (P2PKH)



$$A = RIPEMD160(SHA256(K))$$

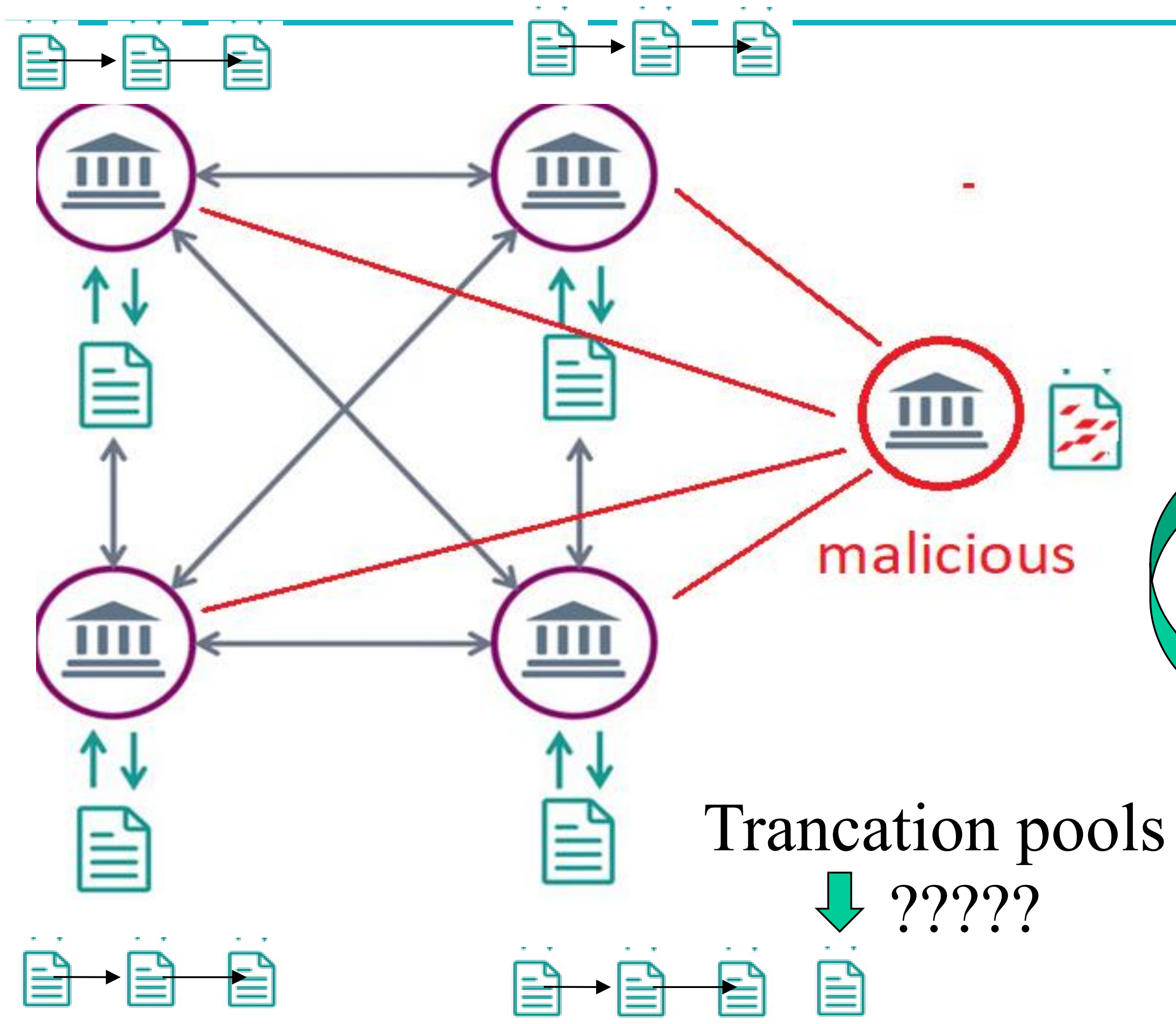
Verifying transation data: Transaction pools

- **Transaction pool**



Valid transatons waiting to record to block

Securely synchronizing



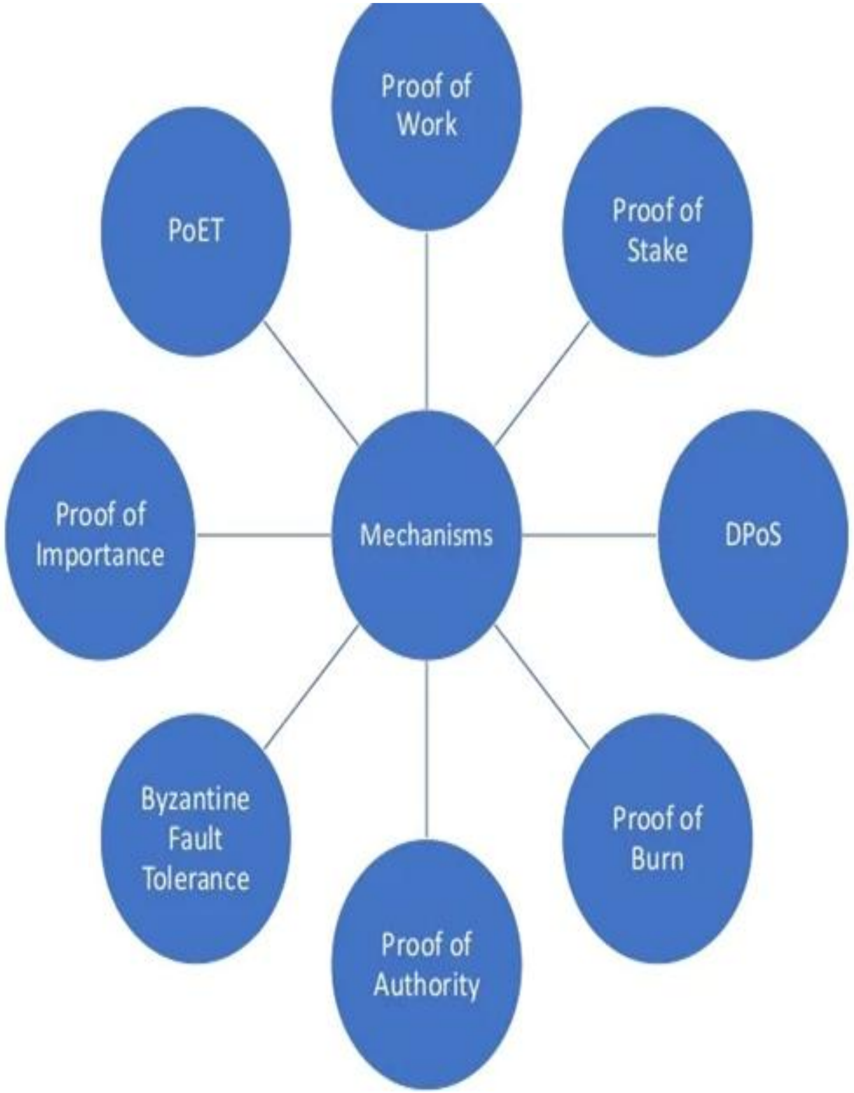
- **How to ?**
  - ✓ Verifying new events/transations (data authentication, other security features)
  - ✓ Securely storage, create, verify, synchonize a new block (all nodes): immutability, transability,

Consensus machanism

Fault-tolerant mechanism

Consensus machanism:

- Supporting nodes (create new space to record events (blocks), distributing data);
- Securly synchronizing events;



Consensus machanism

See chapter Eleven

A **consensus mechanism** is a fault-tolerant mechanism to **reach an agreement** on a single state of the network among distributed nodes.

Contents

Consensus machanism

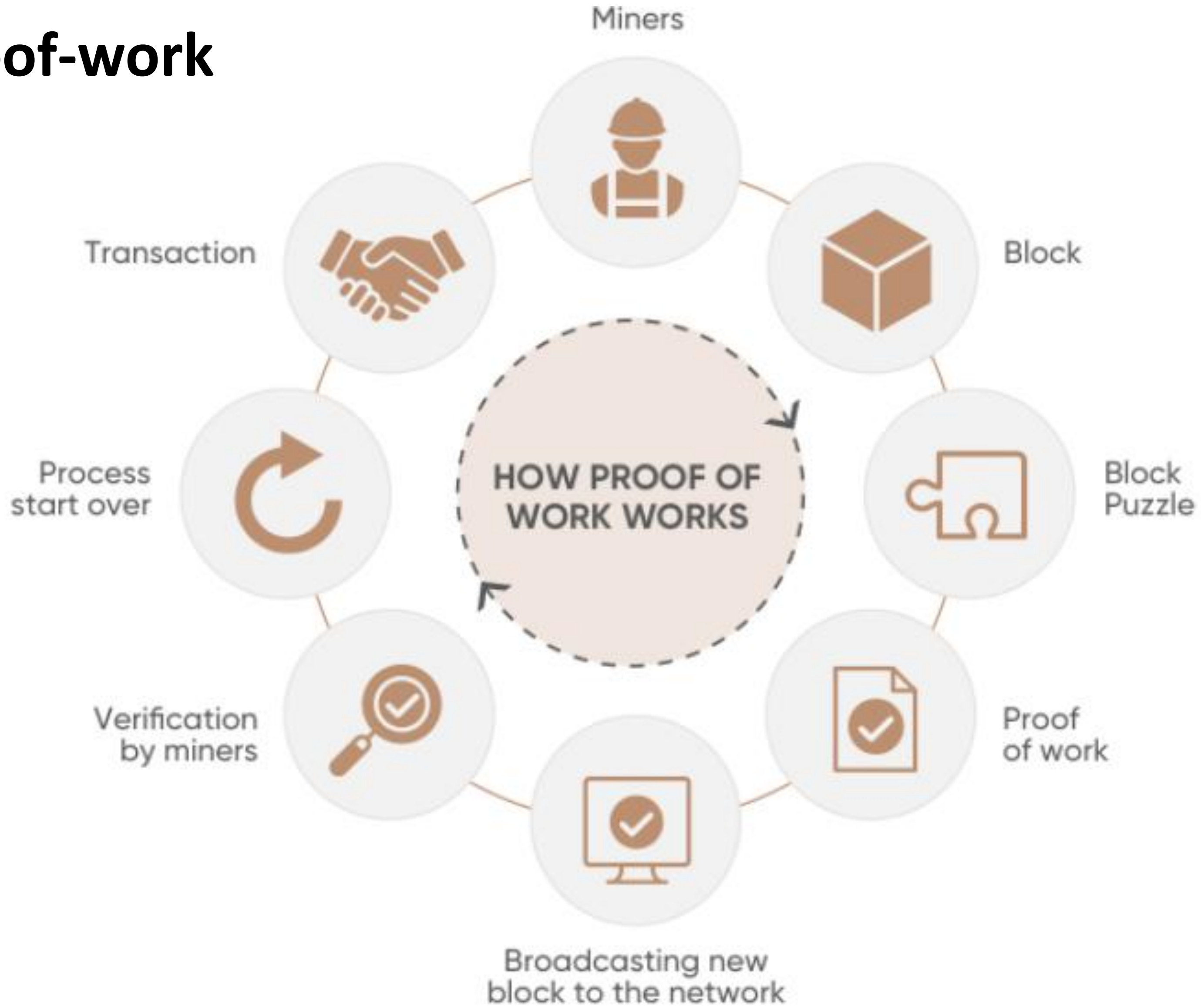
Proof-of-work

To add each block to the chain, miners must compete to solve a difficult puzzle using their computers processing power.

- Prevent multiple fake requests
- Trustless and distributed consensus

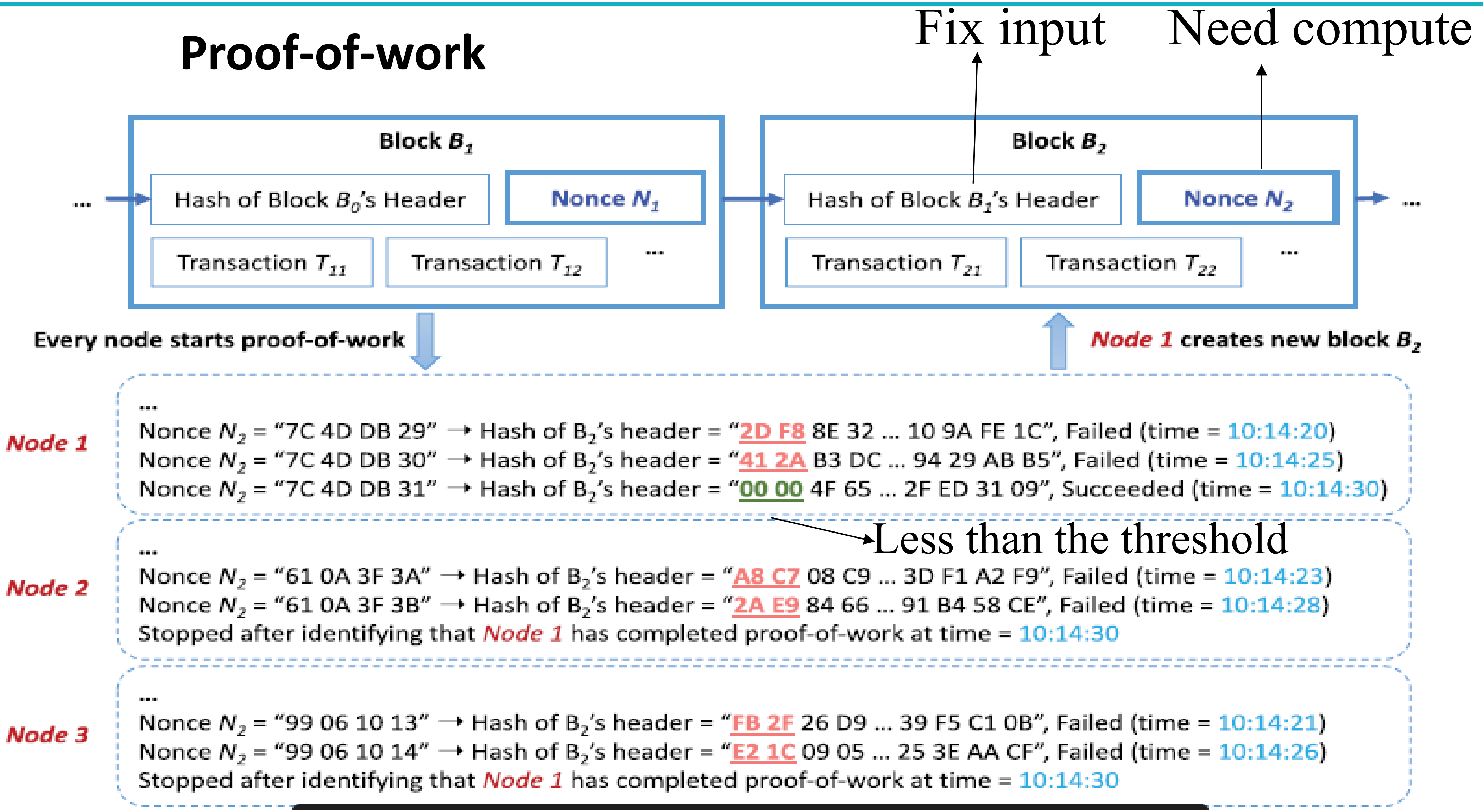
Consensus machanism

Proof-of-work



Consensus machanism

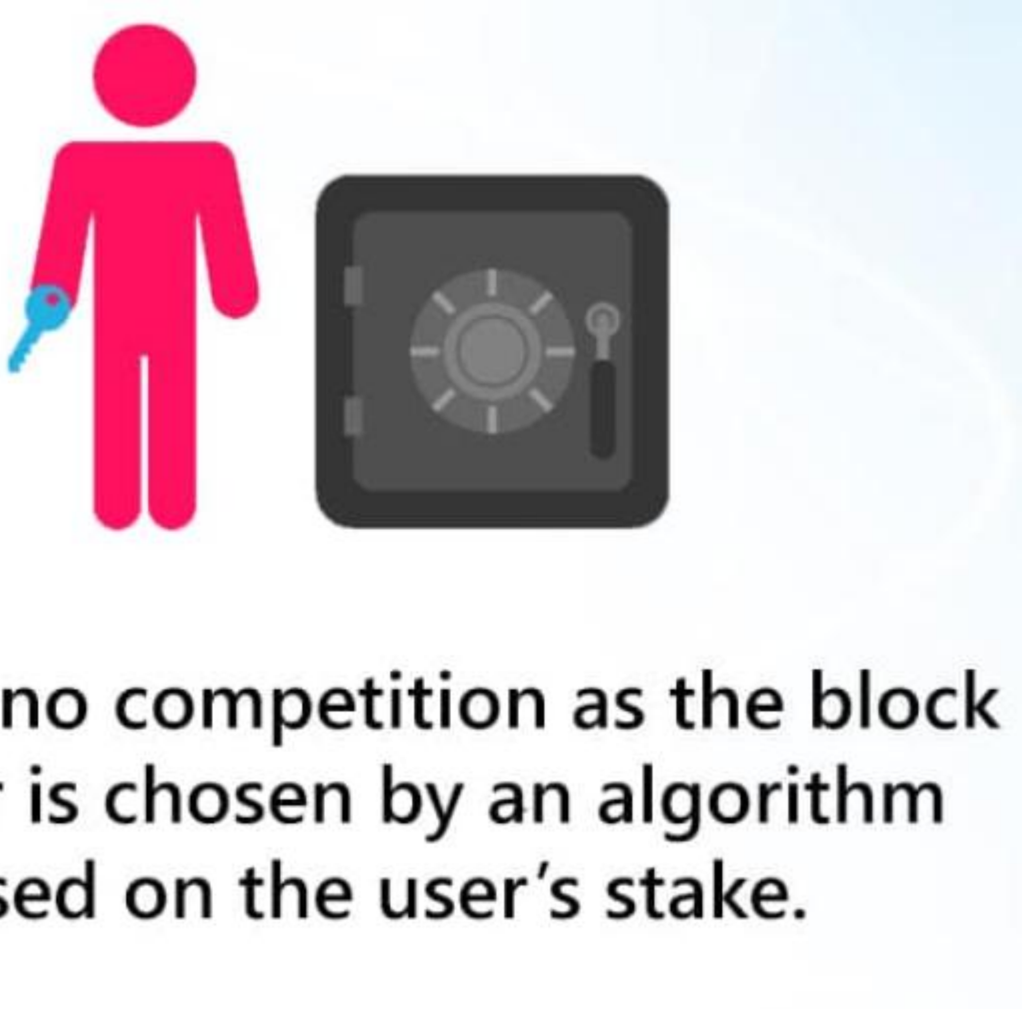
Proof-of-work



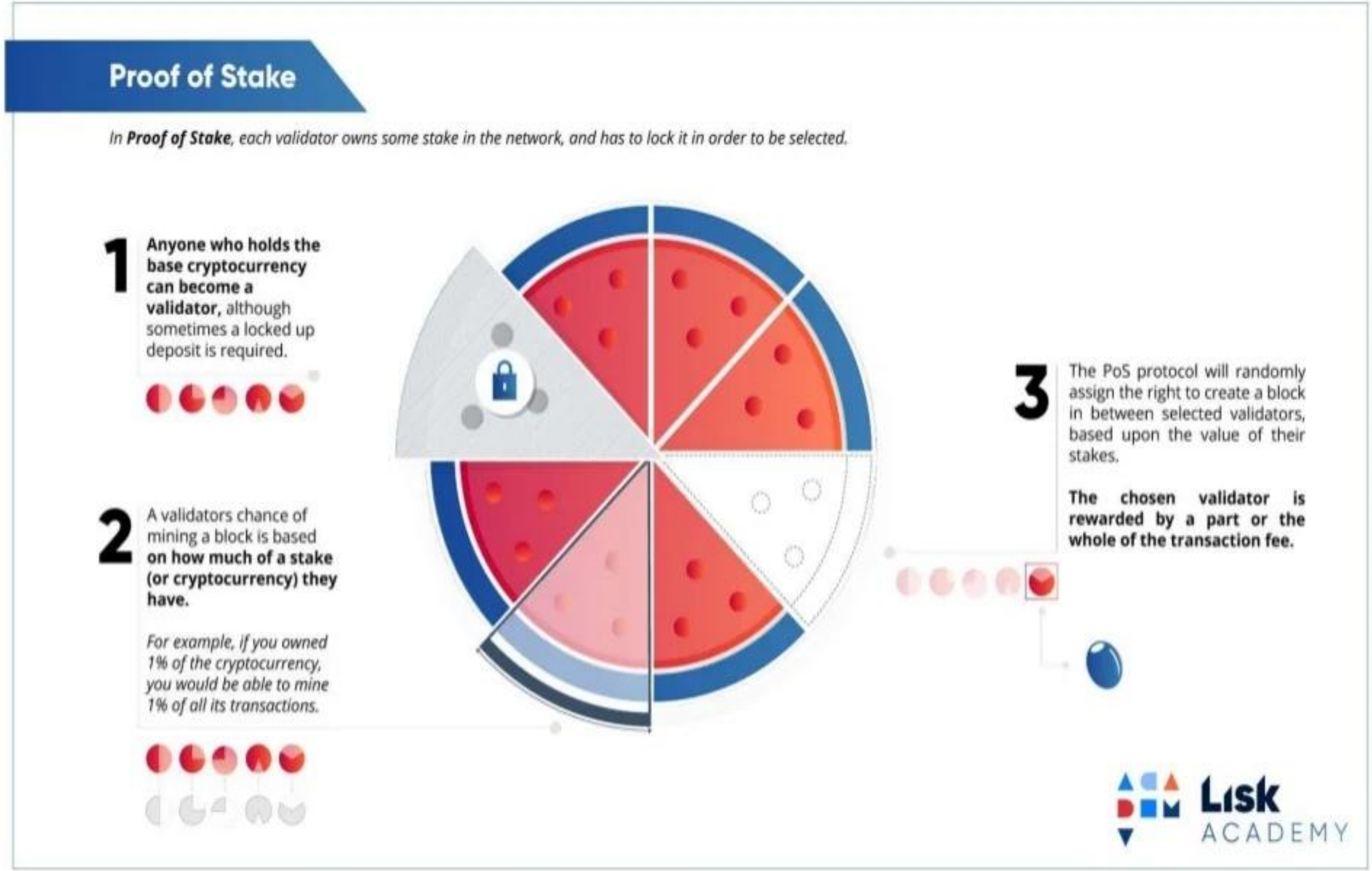
Consensus machanism

Proof-of-Stake

miners ⇒ validators



Consensus machanism

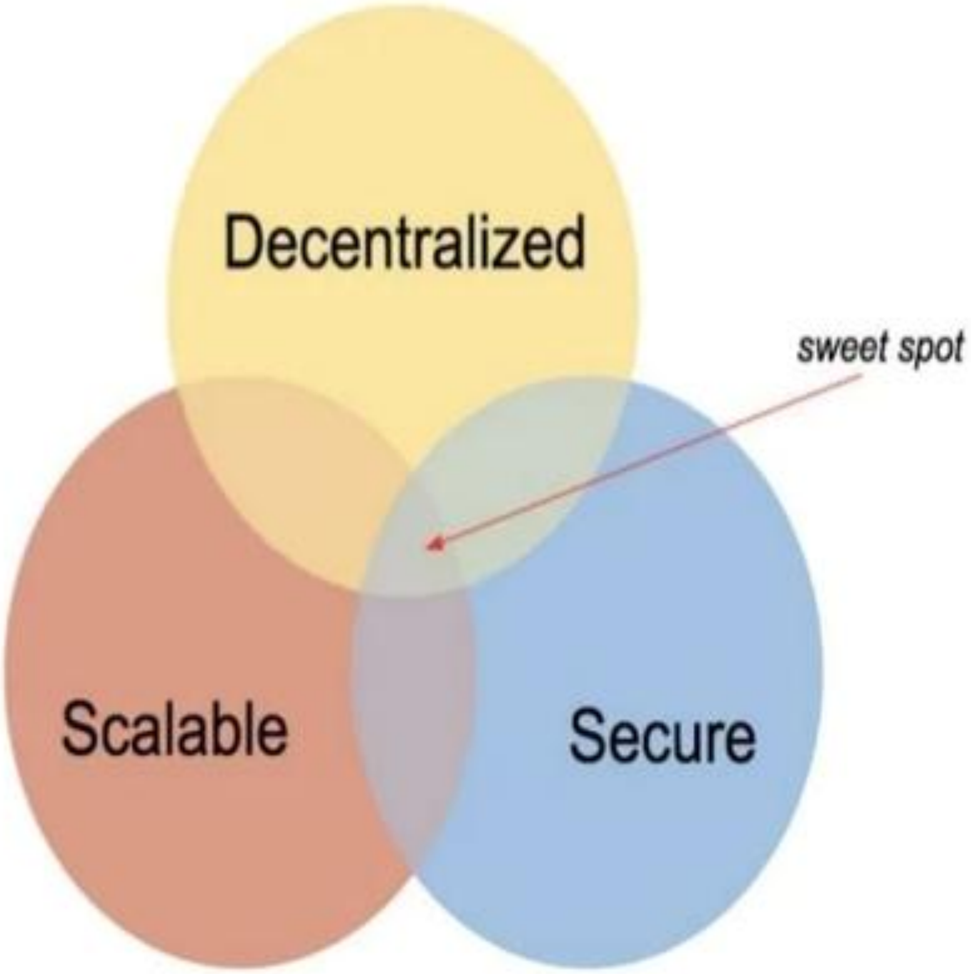


https://lisk.io/content/5-academy/2-blockchain-basics/4-how-does-blockchain-work/8-proof-of-stake/8-pos-infographic.jpg

Consensus machanism

The Scalability Trilemma

- Each of the three goals by itself is "easy" to achieve
- But you have to sacrifice one of Decentralization, Scalability or Security to achieve a high level in the other two
- Bitcoin sacrifices scale (to some degree) for decentralization and security
- Ripple, Stellar, EOS sacrifice decentralization (to some degree) for scale and security
- Related to CAP Theorem for Distributed Data Stores:
  - Consistency
  - Availability
  - Partition Tolerance



Bitcoin: a case study

Decentralization

+ no central point of control

+ consensus protocols

  - participants on the network all must agree unanimously add a data
  - do it while ensuring its integrity;

Immutability (impossible to change)

+ data remains unchangeable once it's been recorded and processed on the blockchain (protected from any modifications or attacks);

+ makes the system more secure (eliminates trust required by traditional centralized authorities)

Bitcoin: a case study

Transparency (data open/transparent)

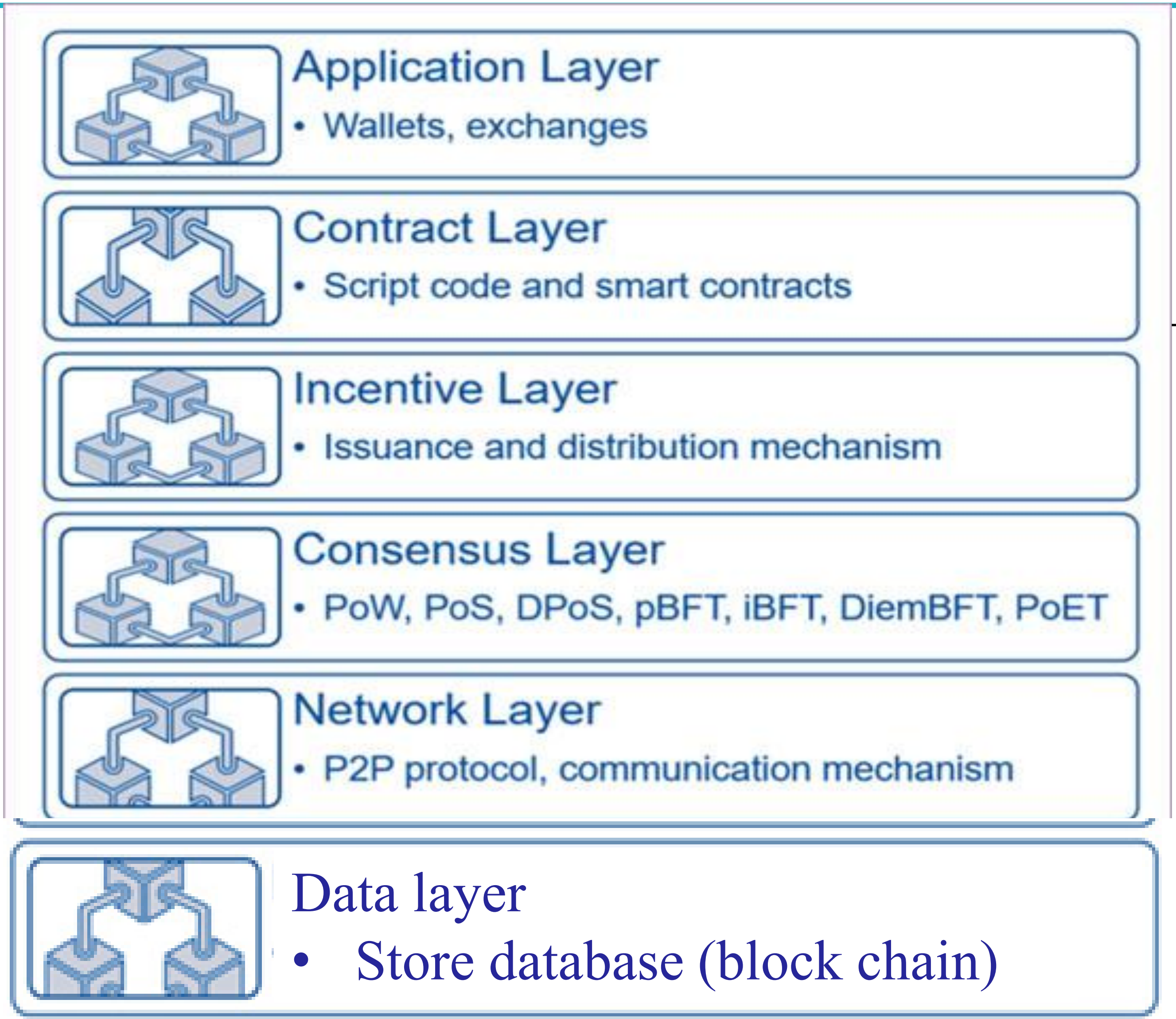
+ block explorer: access all block datas;

+ search the blocks of a blockchain: access their contents and their relevant details;

Outline

- Network secure protocols
  - Authentication;
  - Key agreemen;
  - IPSec; SSL/TLS; SSH; Kerberos
- Blockchain network security
  - Motivation
  - Database structure;
  - Secure Transactions and consensus mechanism;
  - Network architecture;
  - Applications

Network Architecture



Programmable distributed applications

- **How to ?**
  - ✓ Verifying new events/transations (data authentication, other security features)
  - ✓ Securely storage, create, verify, synchonize a new block (all nodes): immutability, transability,

Next generation of blockchain network

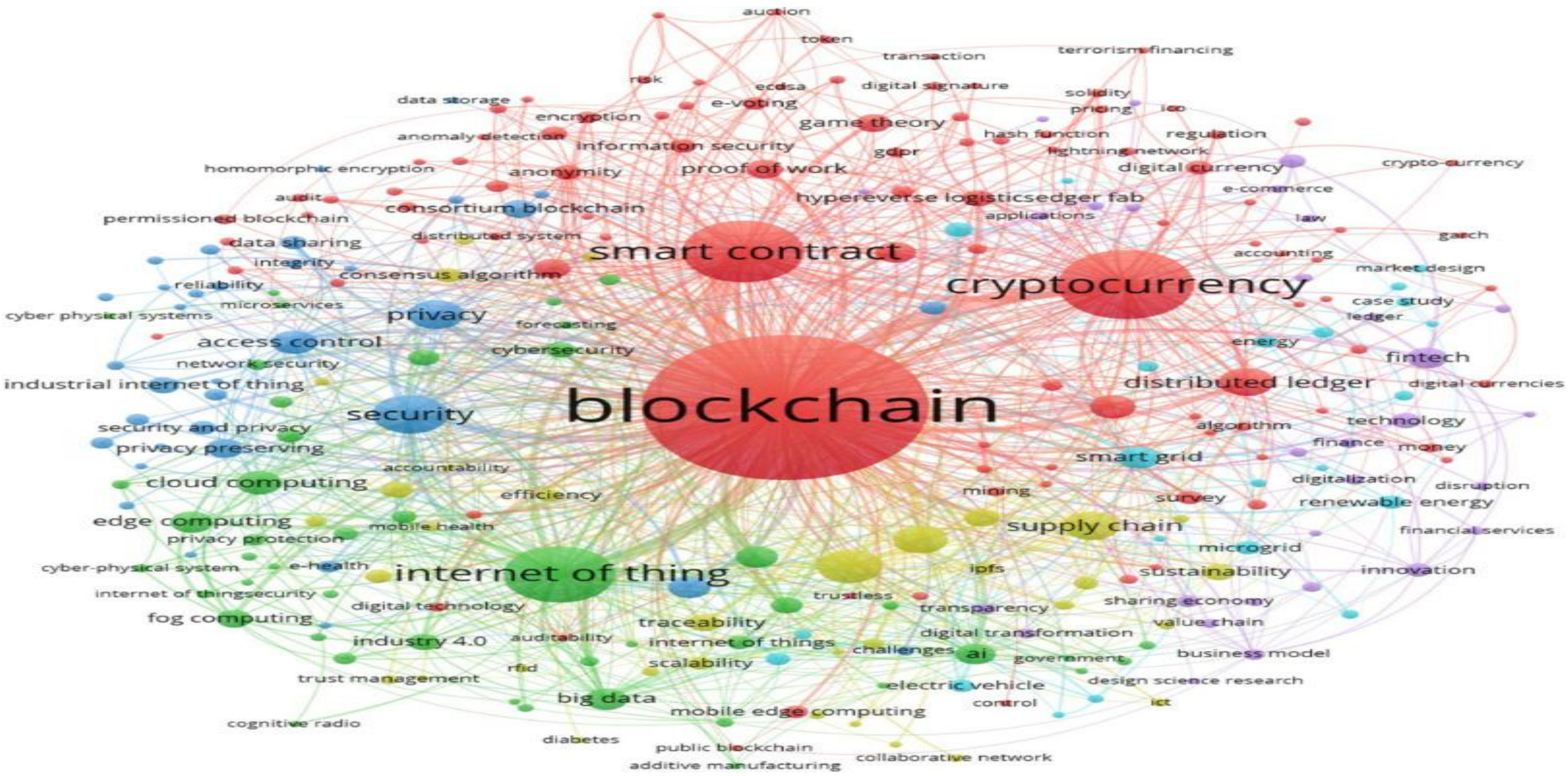
- A smart contract is a computer program or a transaction protocol
  - ✓ automatically execute
  - ✓ control or document legally relevant events (**compiled code**)
  - ✓ actions according to the terms of a contract or an agreement (**sending a transaction from a wallet to the blockchain**)

Smart contract

```
#![no_std]

elrond_wasm::imports!();

#[elrond_wasm::derive::contract]
pub trait Adder {
    #[view(getSum)]
    #[storage_mapper("sum")]
    fn sum(&self) -> SingleValueMapper<BigInt>;

    #[init]
    fn init(&self, initial_value: BigInt) {
        self.sum().set(&initial_value);
    }

    #[endpoint]
    fn add(&self, value: BigInt) -> SCResult<()> {
        self.sum().update(|sum| *sum += value);

        Ok(())
    }
}
```

https://docs.near.org/docs/develop/contracts/rust/intro

Application domains



https://ars.els-cdn.com/content/image/1-s2.0-S0040162520312890-gr9.jpg