Blockchain and Distributed Ledgers

Tom Van Cutsem
DistriNet KU Leuven

COSIC Course July 1-4, 2024

tvcutsem.github.io
be.linkedin.com/in/tomvc
github.com/tvcutsem
x.com/tvcutsem
@tvcutsem@techhub.social

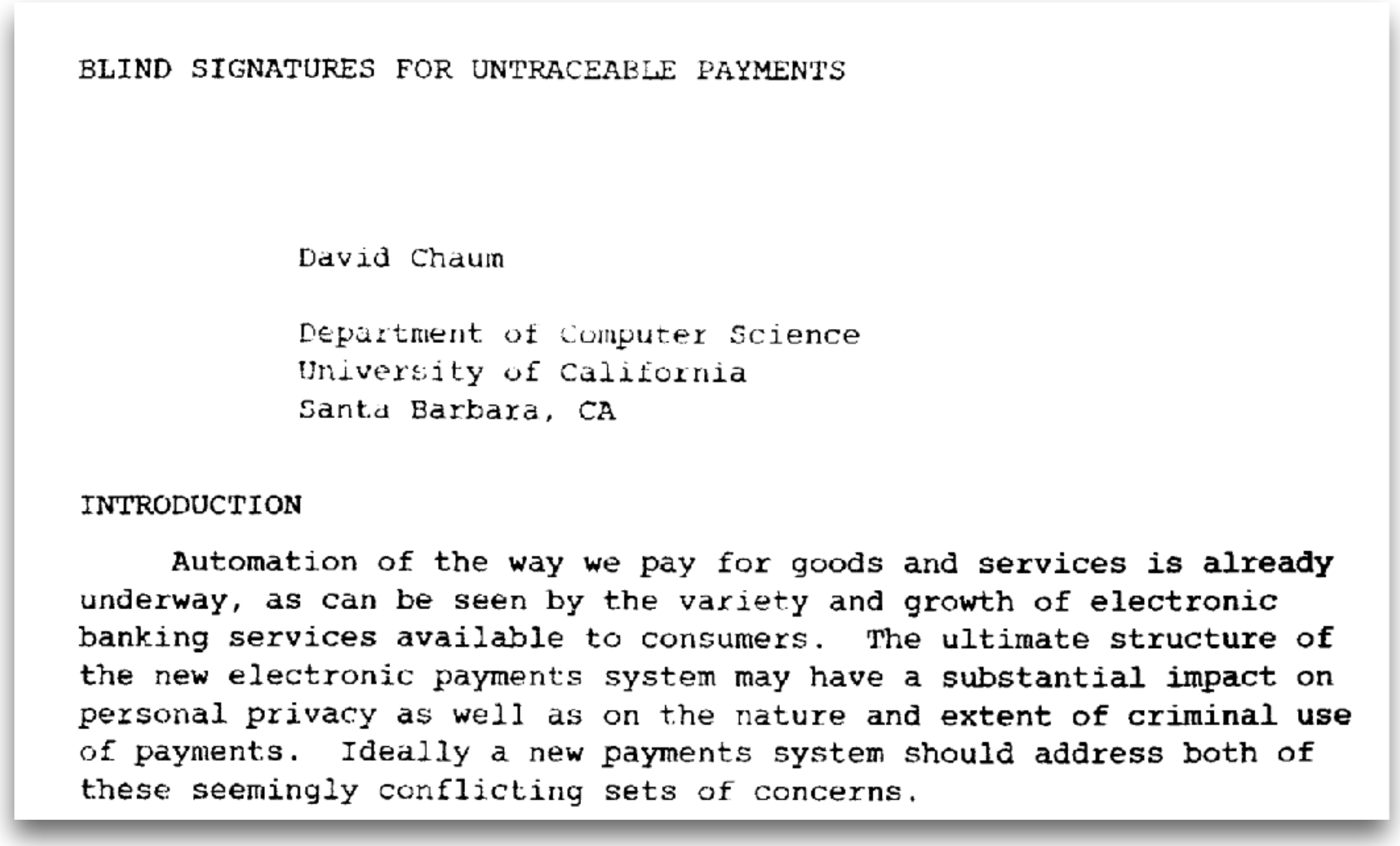
Course topics

- The **origins** of Blockchain
- What are the **cryptographic building blocks** of a blockchain?
- How does a blockchain process transactions? **Life of a blockchain transaction**.
- **Consensus** in blockchain networks: Proof-of-Work, Proof-of-Stake, BFT Consensus
- **Permissioned** versus **Permissionless** blockchain networks
- Bonus: Blockchains as trusted computers: **smart contracts** and **Ethereum**
- **Conclusion** and latest trends

Blockchain: origins

Electronic **cash**

- Since the dawn of the Internet, cryptographers have tried to create digital currencies that are more similar to physical cash or coins
- Money as a “bearer instrument” token: whoever holds the token can spend it
- Payments are anonymous and untraceable
- Example: e-cash (Digicash)



BLIND SIGNATURES FOR UNTRACEABLE PAYMENTS

David Chaum

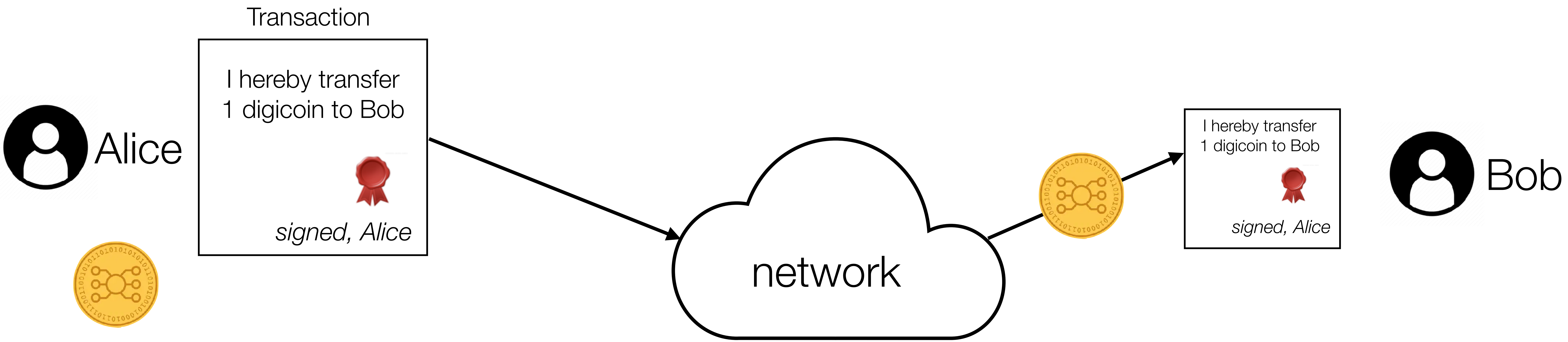
Department of Computer Science
University of California
Santa Barbara, CA

INTRODUCTION

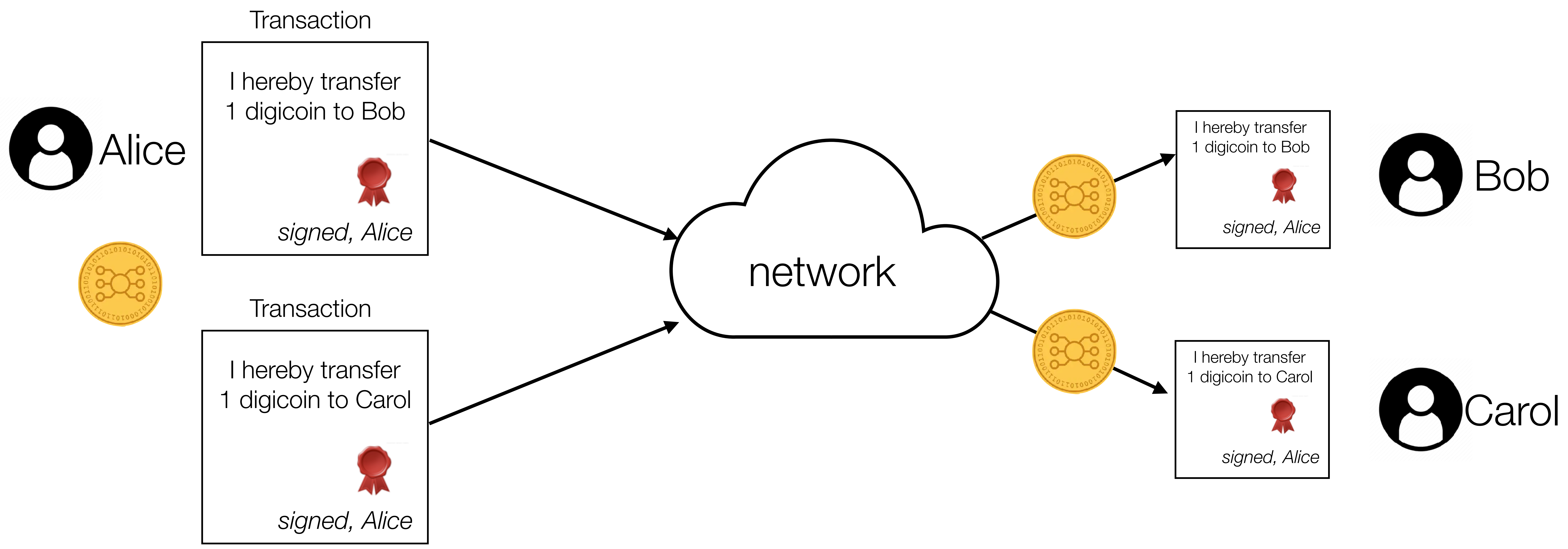
Automation of the way we pay for goods and services is already underway, as can be seen by the variety and growth of electronic banking services available to consumers. The ultimate structure of the new electronic payments system may have a substantial impact on personal privacy as well as on the nature and extent of criminal use of payments. Ideally a new payments system should address both of these seemingly conflicting sets of concerns.

David Chaum
Electronic cash (1982)

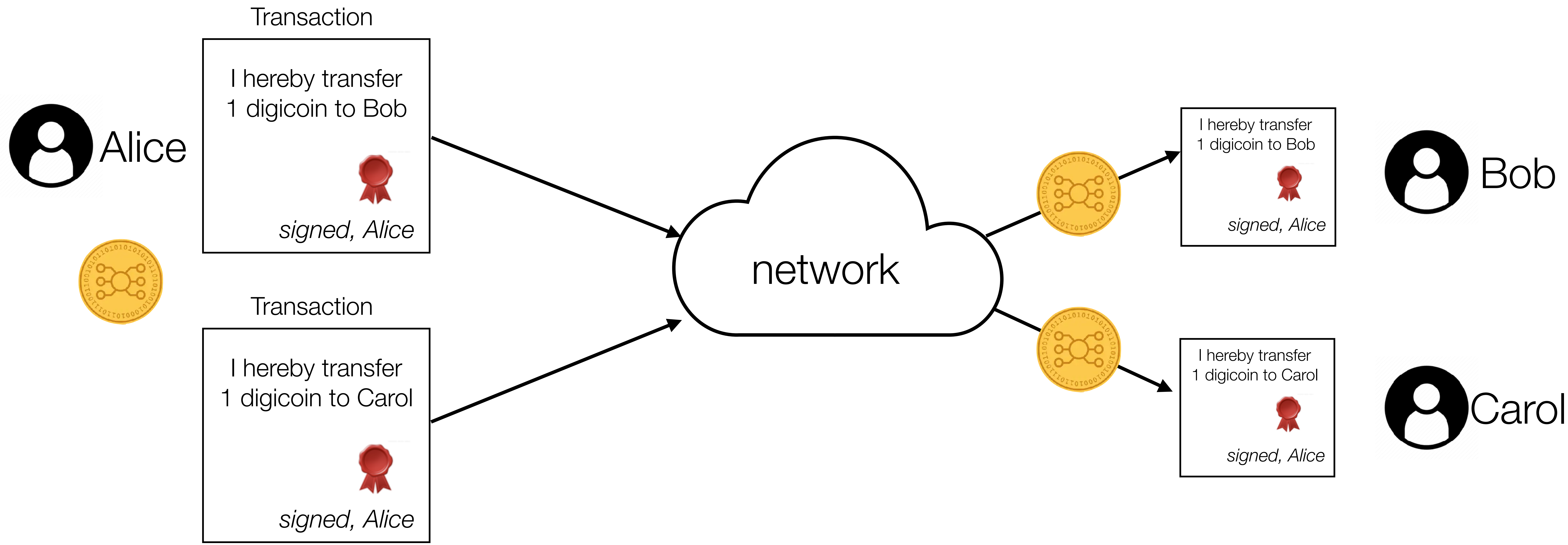
The problem with electronic cash: the Double Spending Problem



The problem with electronic cash: the Double Spending Problem



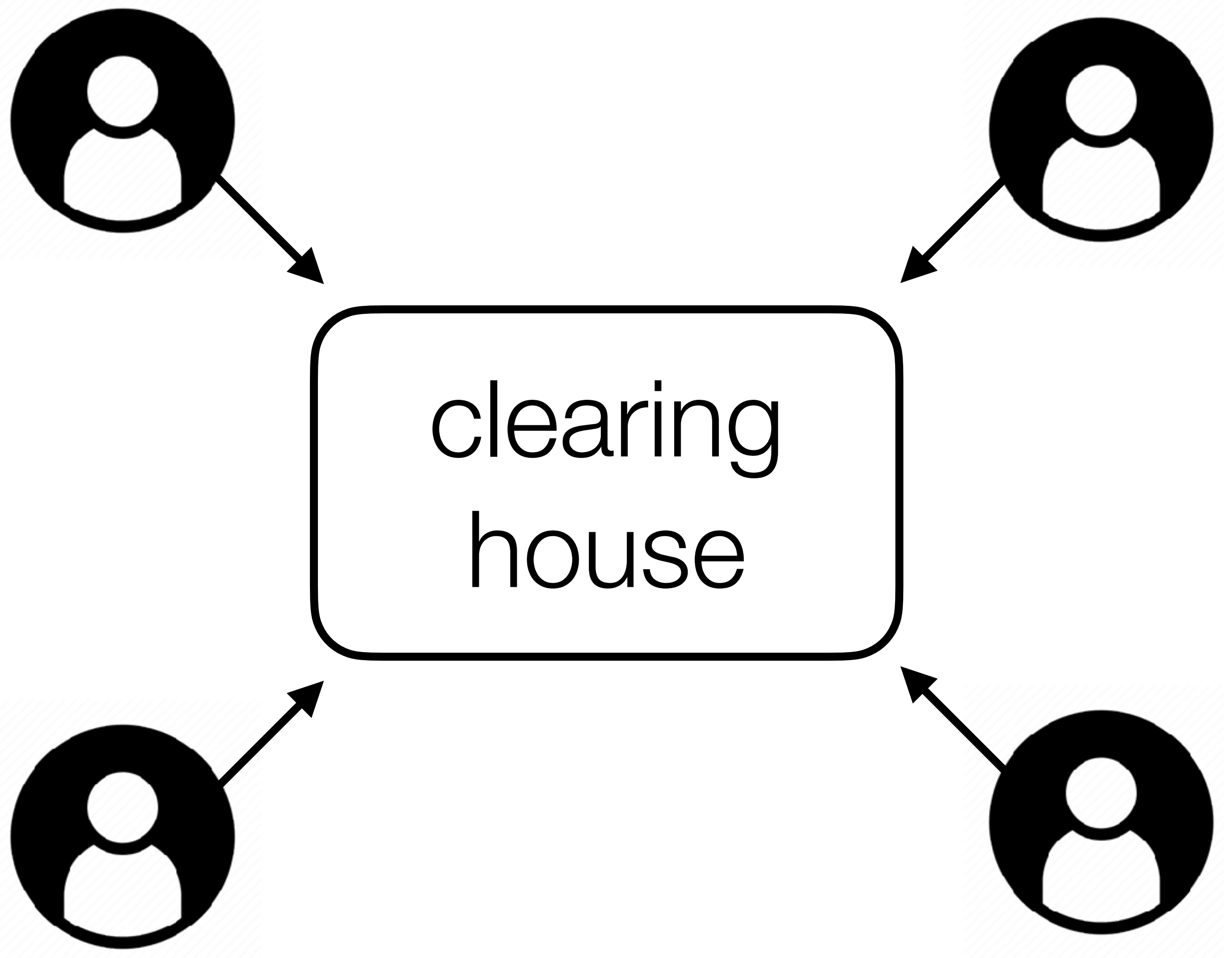
The problem with electronic cash: the Double Spending Problem



How can Bob and Carol be sure they are now the sole owner of Alice's coin?

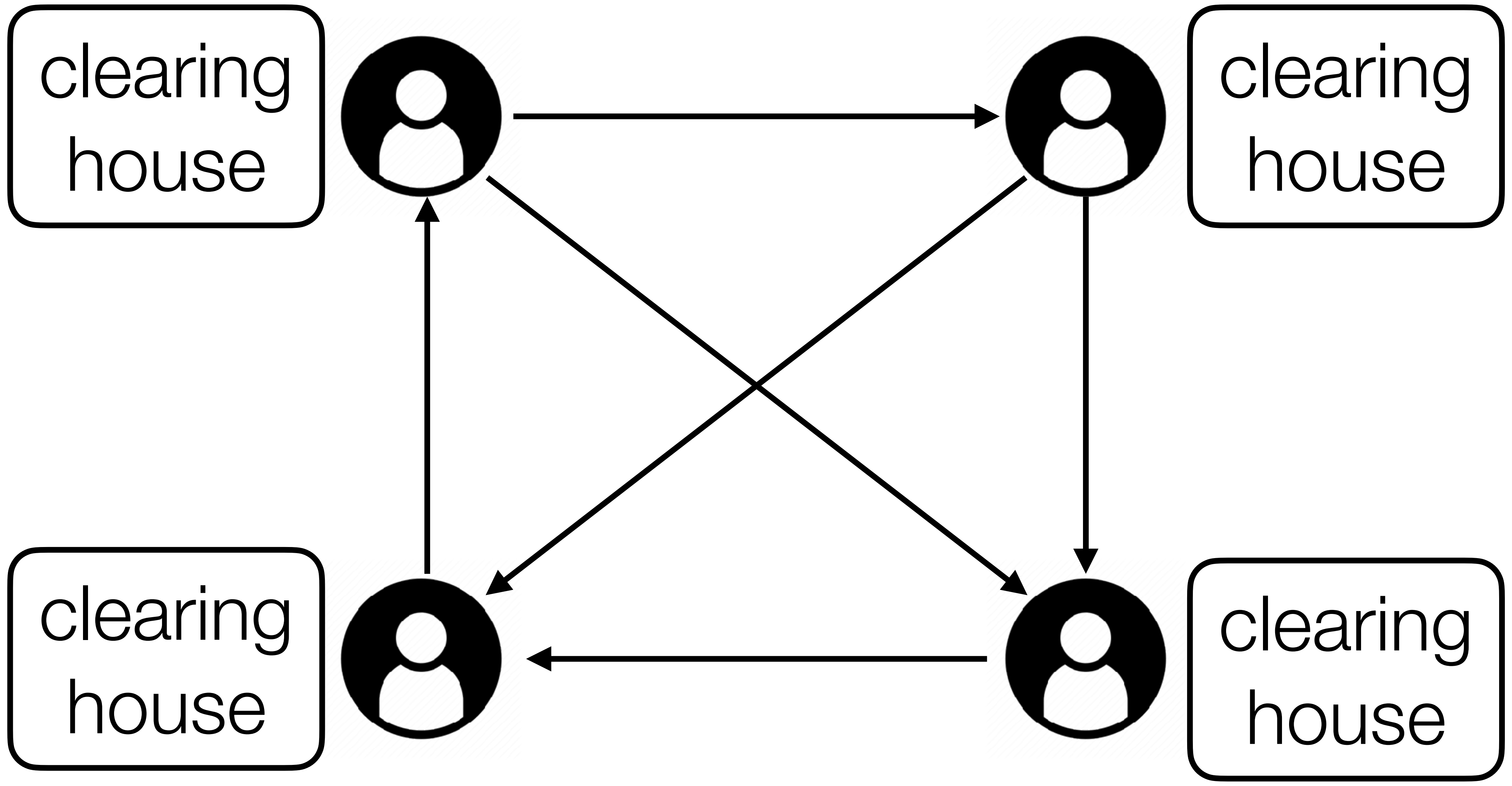
Straightforward solution: use a central clearing house

- The clearing house does the accounting of what tokens have already been spent. This **avoids "double spending"** the same token.
- The payments themselves can still be **anonymous**! We just need to record spent tokens. Privacy risks partially mitigated using blind signatures.
- Problem: everyone depends on the clearing house. **Risks**:
  - **Technical** risks: availability (what if the clearing house is unavailable?) and security (what if the clearing house gets attacked? This may include insider threats!)
  - **Economic** and **political** risks: what if the company running the clearing house goes bankrupt or is threatened in court? (E.g. Digicash actually went bankrupt in 1998)

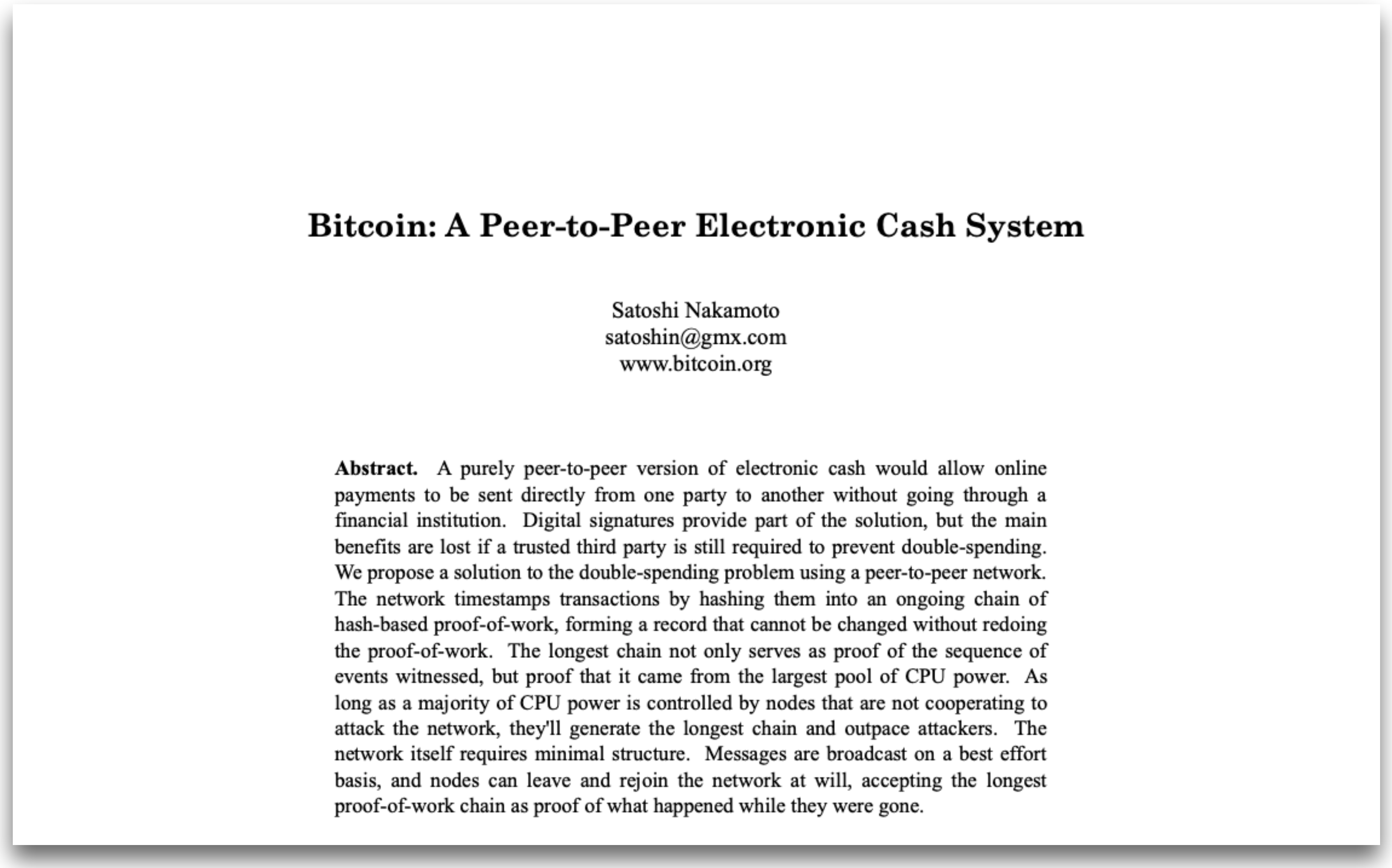


Blockchain networks

- Bitcoin's breakthrough idea: rather than having a single party record who owns what, let *everyone* collectively do the accounting of who owns what
- Store account balances in an append-only **replicated database** called a **blockchain**



Fully decentralised payment network



Bitcoin: A Peer-to-Peer Electronic Cash System

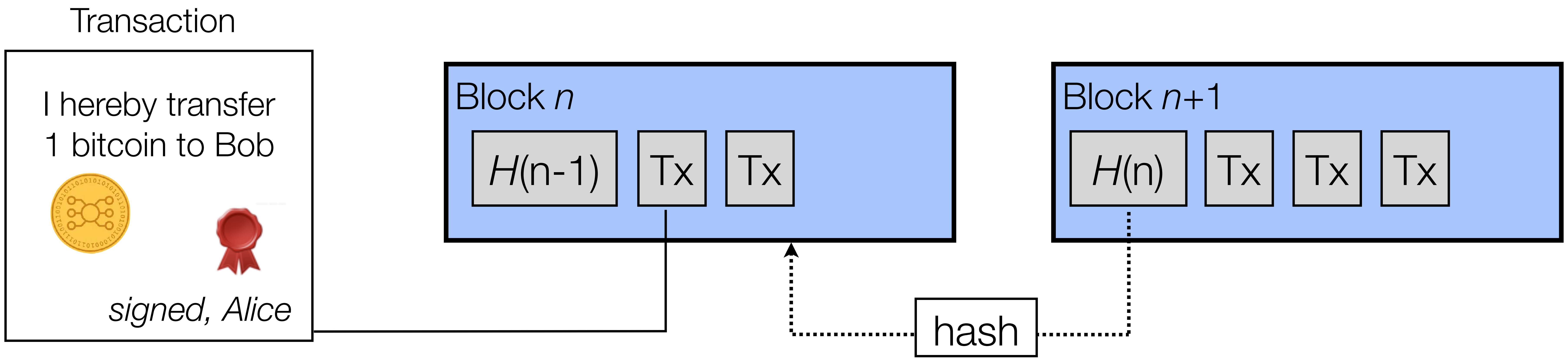
Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

Abstract. A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.

The "Bitcoin whitepaper" by "Satoshi Nakamoto", 2009

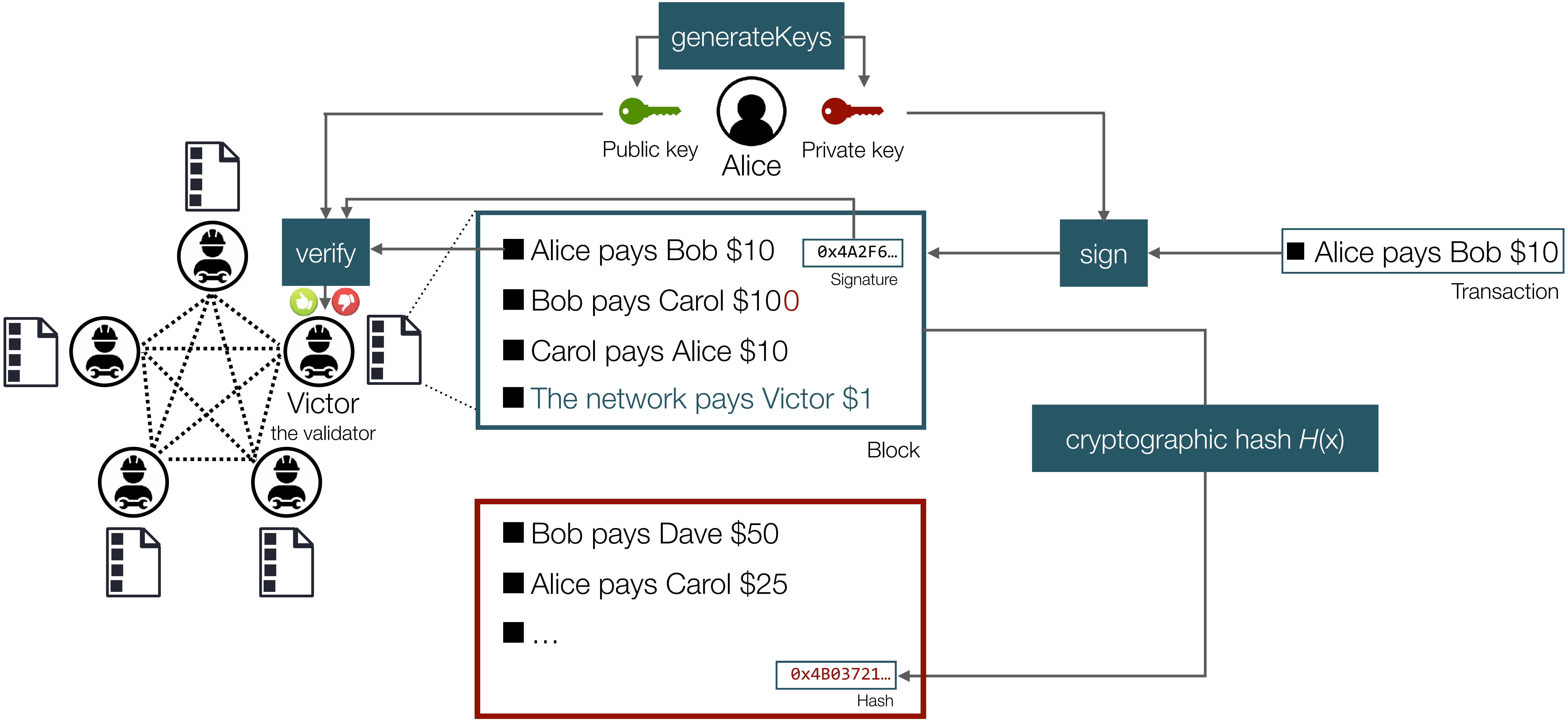
Bitcoin's blockchain

- Replace central clearing house by a **public, replicated, append-only, tamper-resistant ledger**
- Validator nodes group transactions in "blocks", "chained" together into **a linear sequence** using cryptographic hashes, secured using "Proof of Work"



What are the cryptographic building blocks of a blockchain?

How cryptography is used to securely record transactions on a blockchain

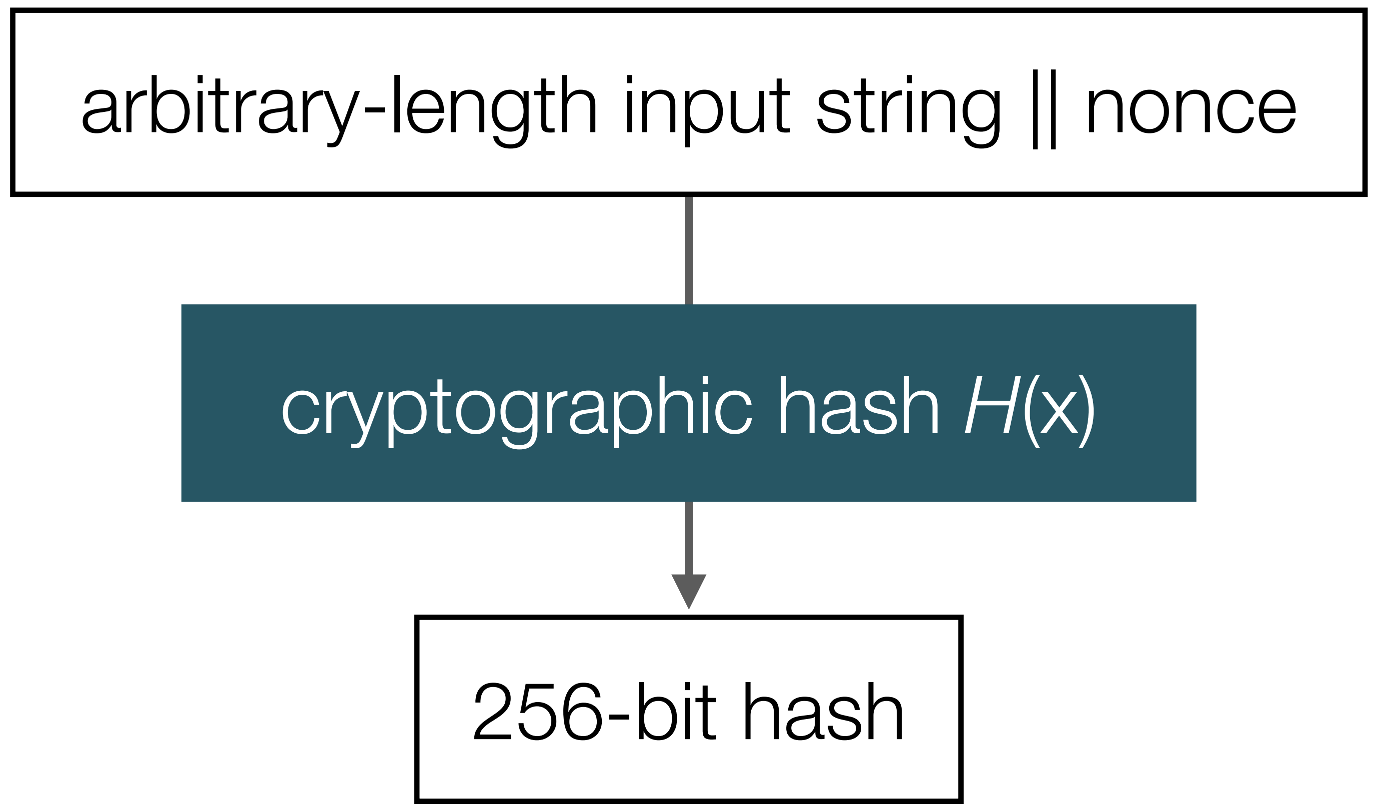


Common cryptographic algorithms used in blockchain systems

Cryptographic hashes

SHA-256 or KECCAK-256

Secure hash algorithm



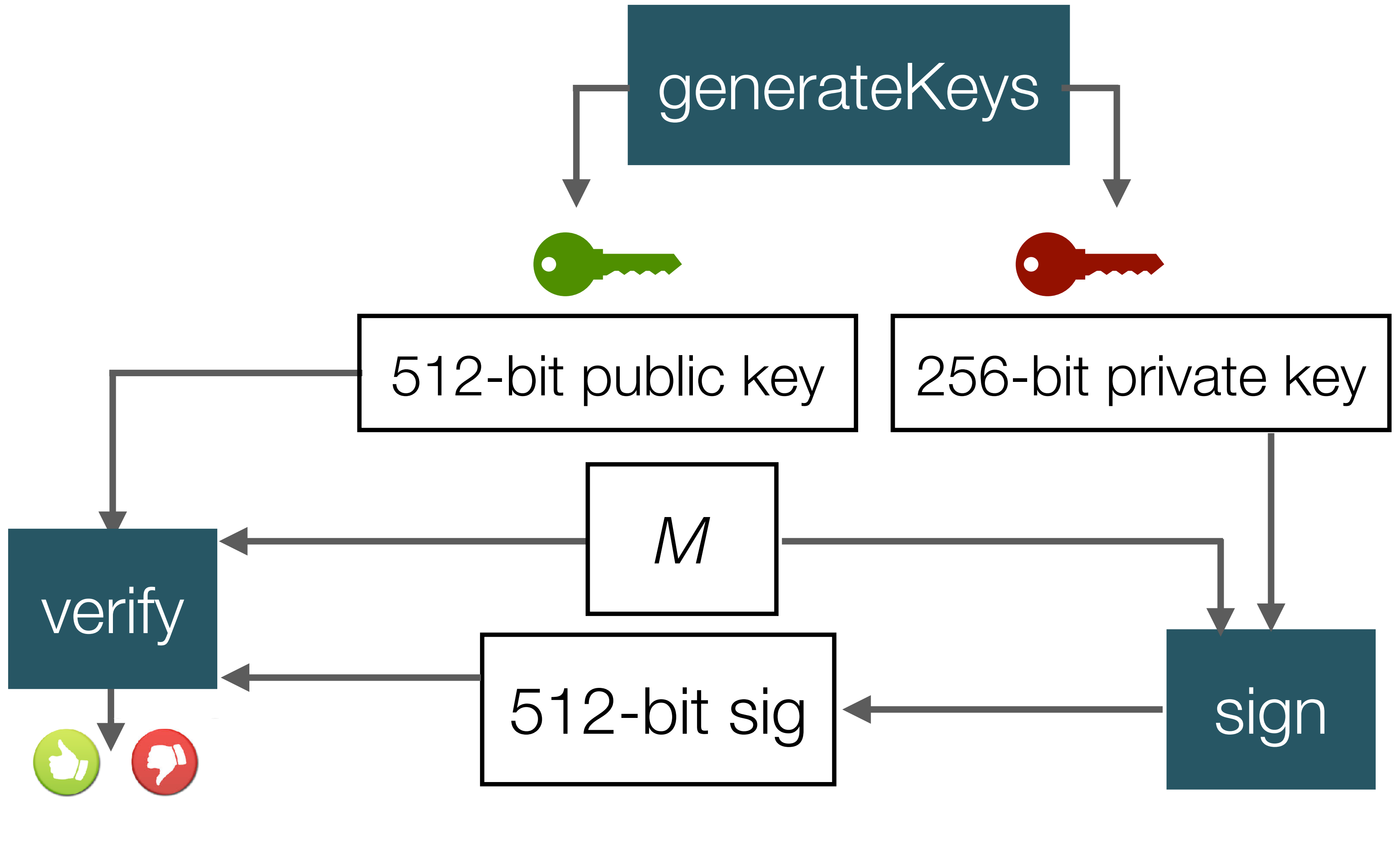
Desirable properties:

- *H* is collision-resistant
- *H* hides its input x
- *H* is “puzzle-friendly”

Digital signatures

ECDSA (secp256k1 curve)

Elliptic curve digital signature algorithm

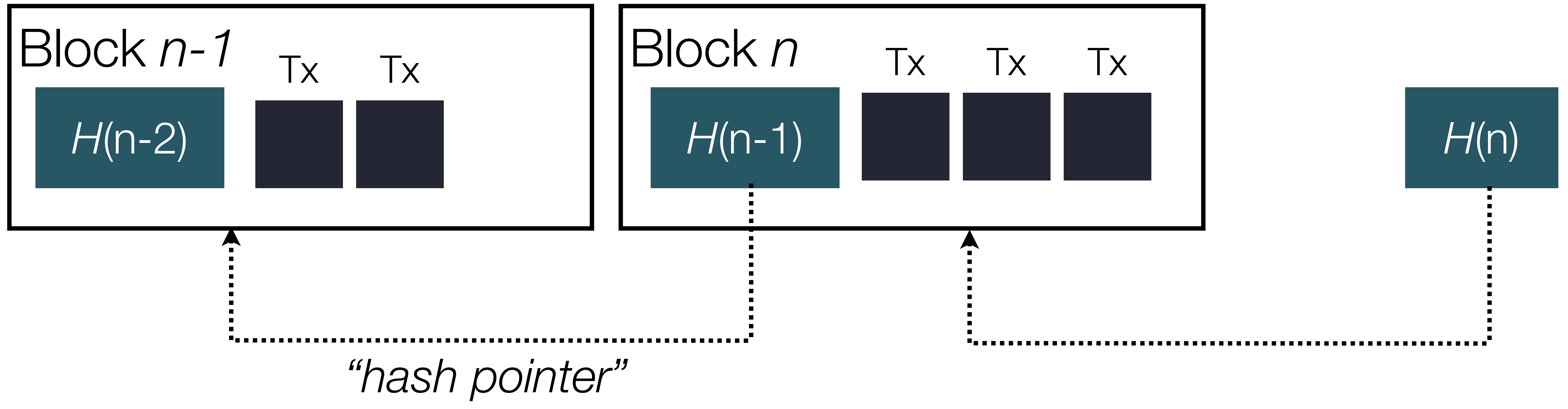


Desirable properties:

- Valid signatures must verify
- Signatures are unforgeable
- Signature is unique to *M*

Common cryptographic algorithms used in blockchain systems

Hash pointers



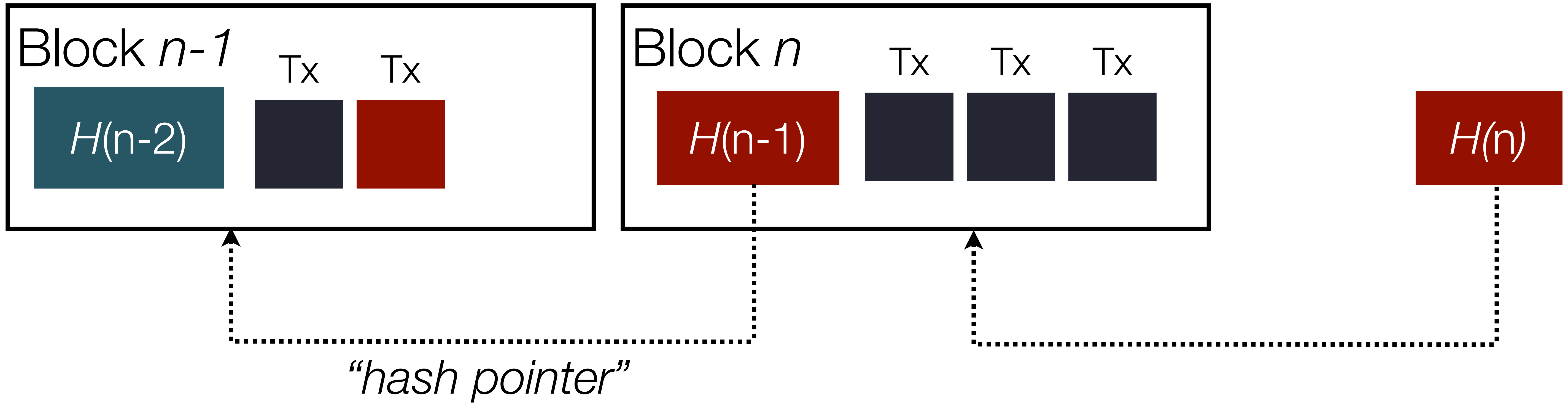
The hash is used both as:

- a unique identifier (to identify and lookup the data)
- a digest (to verify that the data has not been tampered with)

Any non-cyclical data structure can be built from hash pointers

Common cryptographic algorithms used in blockchain systems

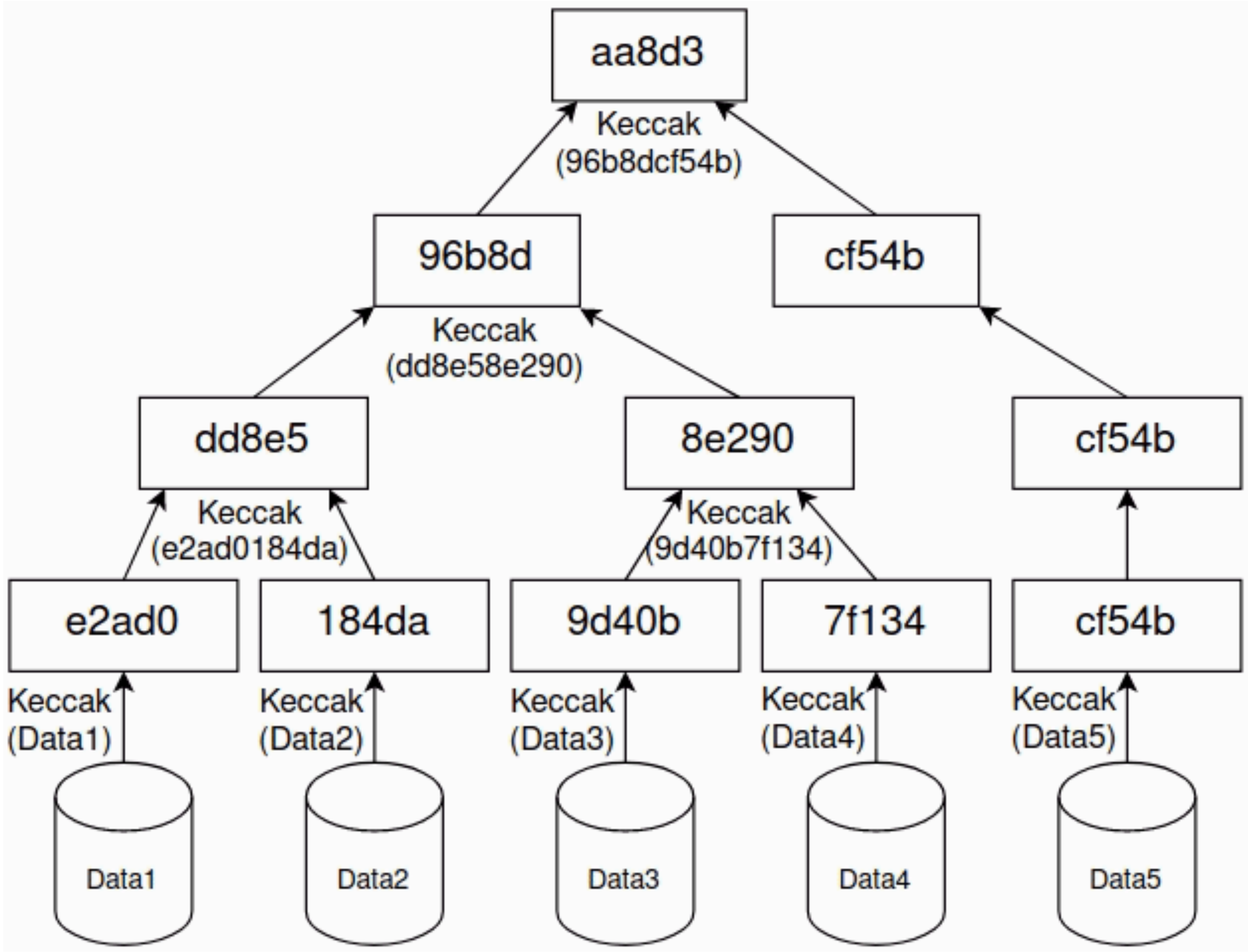
Why use hash pointers?



- We want the transaction log history to be immutable (i.e. only *append* new transaction, not *edit* past transactions).
- By using hash pointers, we ensure that modifying *any* data in *any* past block would invalidate the hash pointers of *all* the following blocks.
- This makes it immediately clear to anyone with a historical copy of the blockchain that data has been tampered with.
- This makes the transaction log “tamper-evident”.

Merkle Trees (a.k.a. Binary hash trees)

- Invented by cryptographer Ralph Merkle in 1979
- Goal: efficiently verify that a piece of data is included in a list of data blocks
- Leaf nodes are labelled with cryptographic hash of a single data element
- Branch nodes are labelled with cryptographic hash of the *concatenation* of the labels of its children

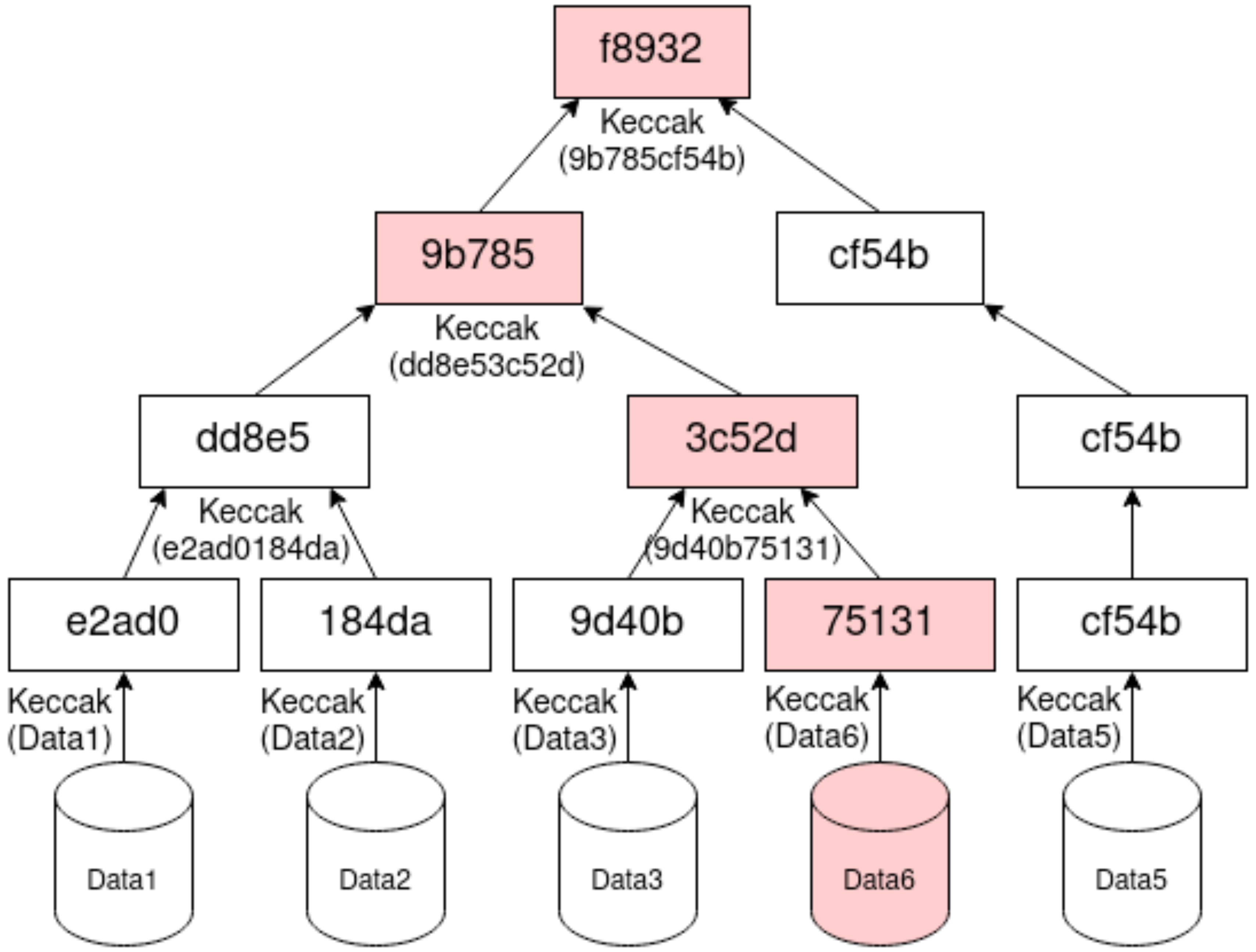


(Image credit: T. Kanstrén, Merkle Trees: Concepts and Use Cases, medium.com)

Merkle Trees: cryptographic commitment

- Changing a single data item would change the leaf hash, and consequently all intermediate hash values up to the root hash

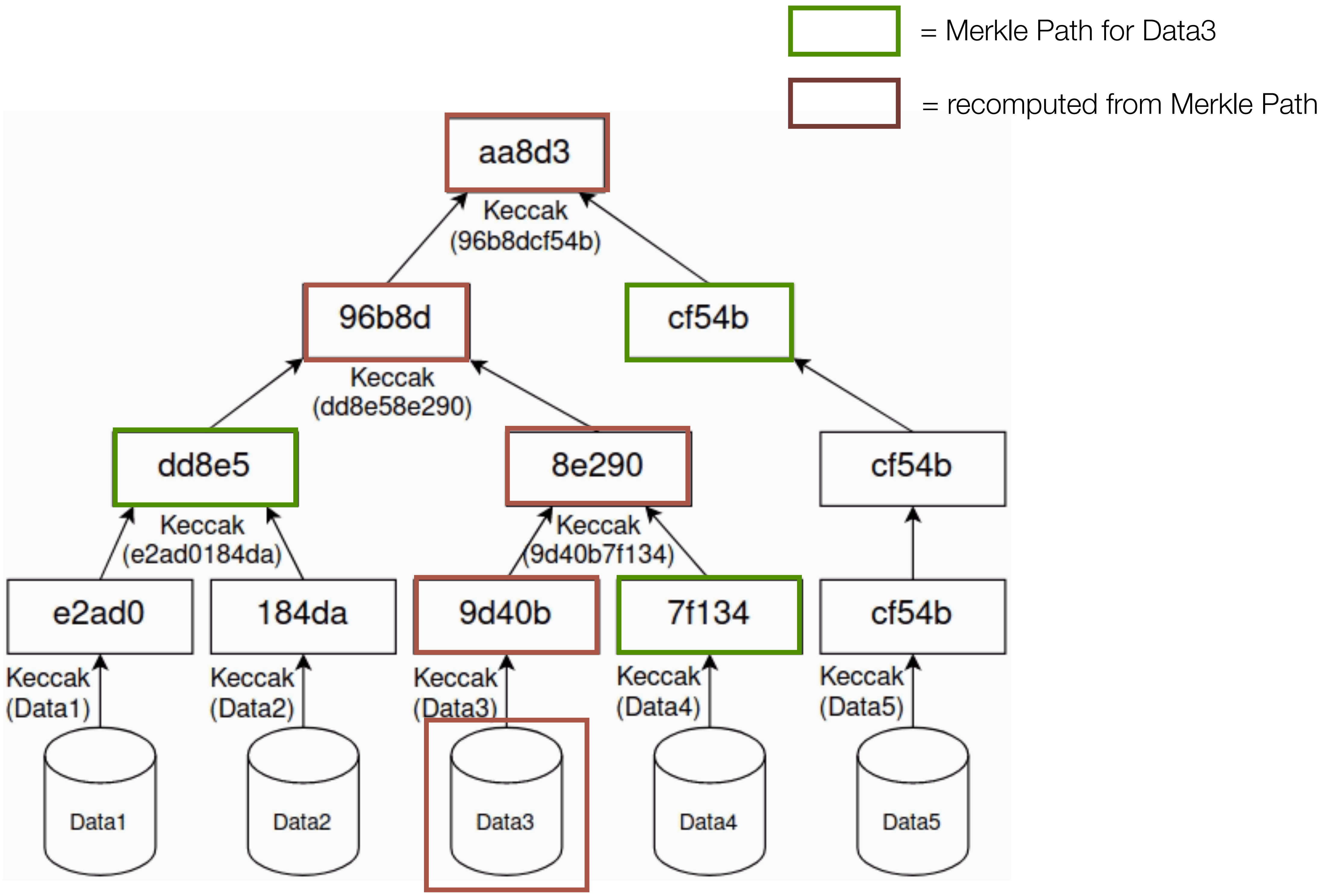
- The root node hash thus represents a *cryptographic commitment* to the entire list of data items



(Image credit: T. Kanstrén, Merkle Trees: Concepts and Use Cases, medium.com)

Merkle Trees support efficient inclusion proofs

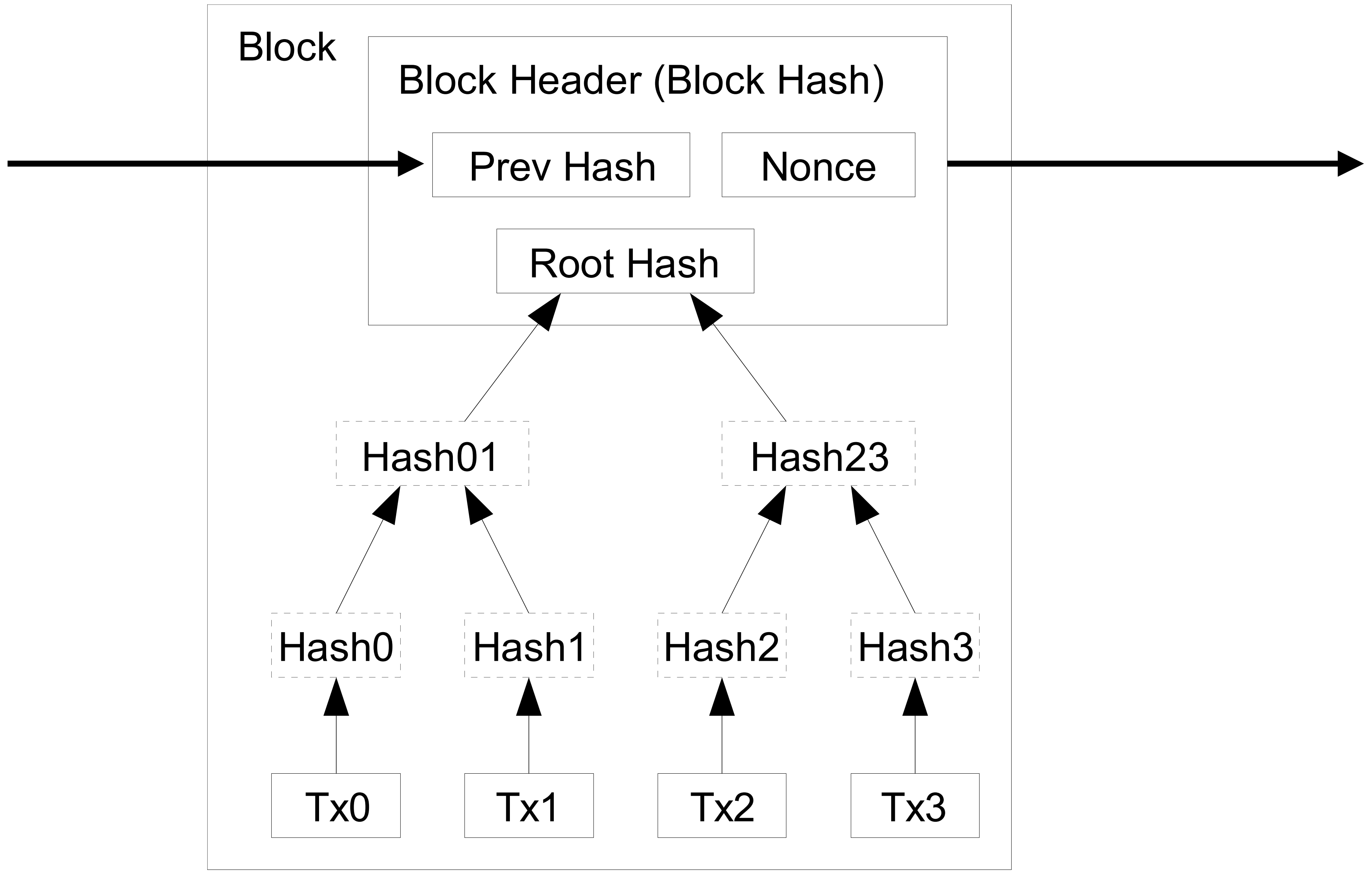
- Goal: prove that a data item is part of the original list (e.g. Data3)
- Only need the hash values of the branches along the data item's path
- $O(\log(n))$ steps, where n is the number of data items (leaves)
- How many steps would have been needed if we would have just stored the hash of the list of data items?



(Image credit: T. Kanstrén, Merkle Trees: Concepts and Use Cases, medium.com)

Merkle trees in the Bitcoin blockchain

- An actual Bitcoin block consists of a Block Header and a transaction list stored separately as a Merkle Tree
- The block header contains the root hash of the Merkle Tree
- This enables clients to efficiently verify that a transaction was included in a block without downloading the full transaction information in each block (“**SPV**” or “Simplified Payment Verification”):
    - Assume client has information on a transaction to verify, including its associated Merkle Path
    - 1. Client queries the Bitcoin network for block headers included in the longest chain
    - 2. Client can recompute Merkle root hash from transaction information and Merkle Path
    - 3. Client can verify that its computed root hash is part of a block header in the longest chain



Transactions Hashed in a Merkle Tree

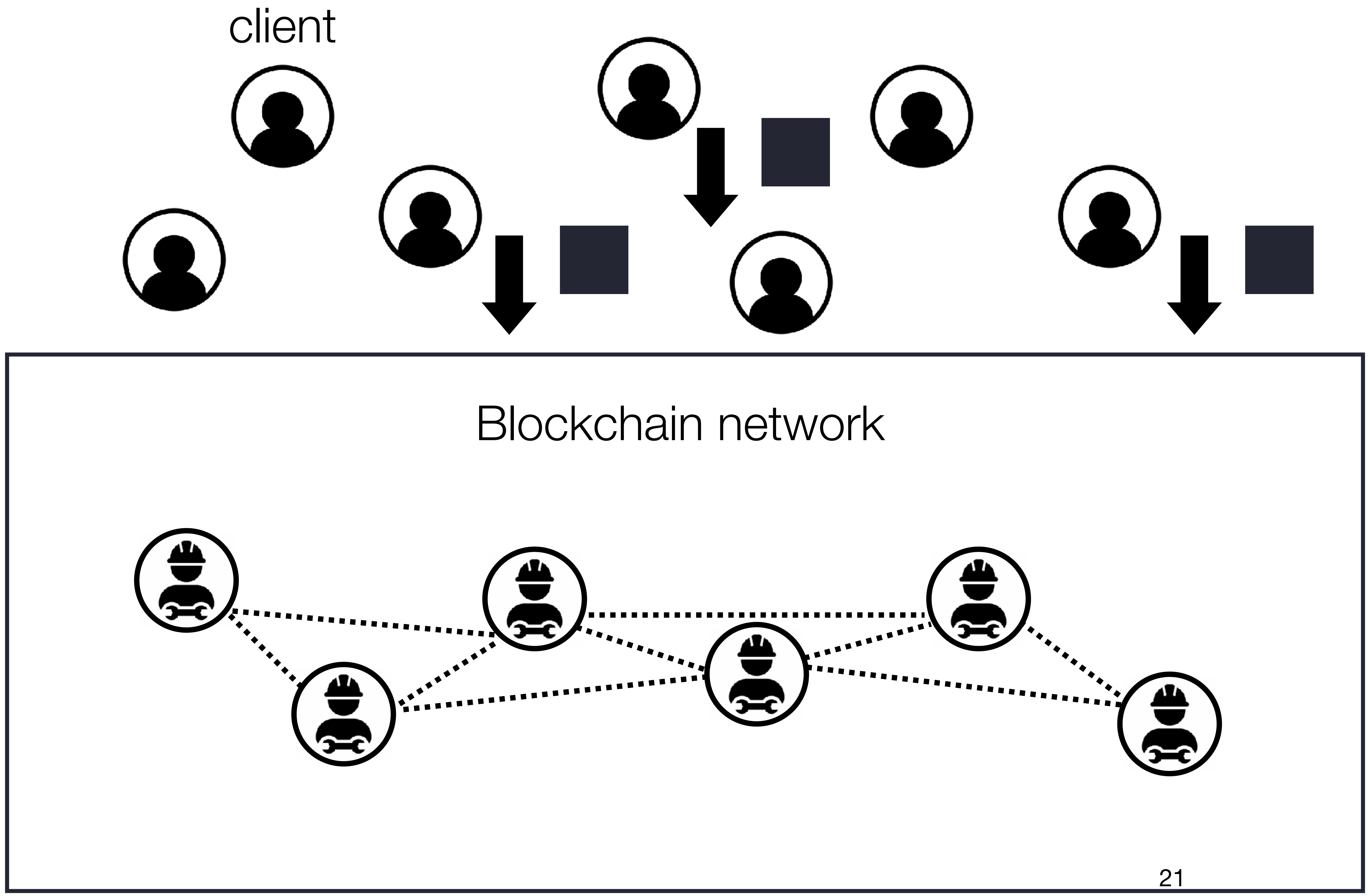
(Source: S. Nakamoto, 2008, “Bitcoin: A Peer-to-Peer Electronic Cash System”)

How does a blockchain network process transactions?

A.k.a. the “life of a blockchain transaction”

Step 1: clients submit signed transactions

- Clients **concurrently** submit signed transactions to one or more validators.



Example transactions…

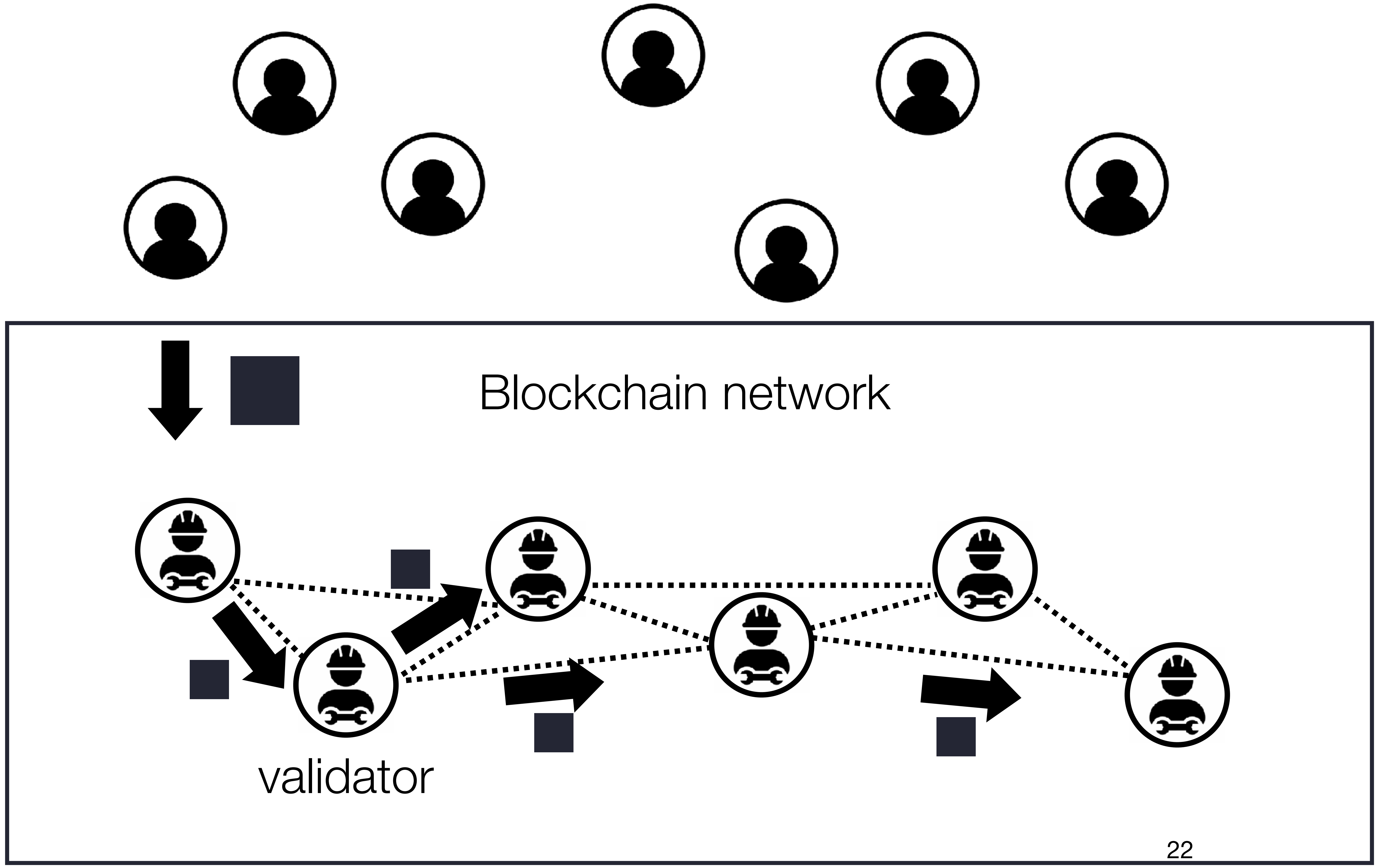
= “send x bitcoin to address *a*”

= “call function *f* on contract *a* with input *x*”

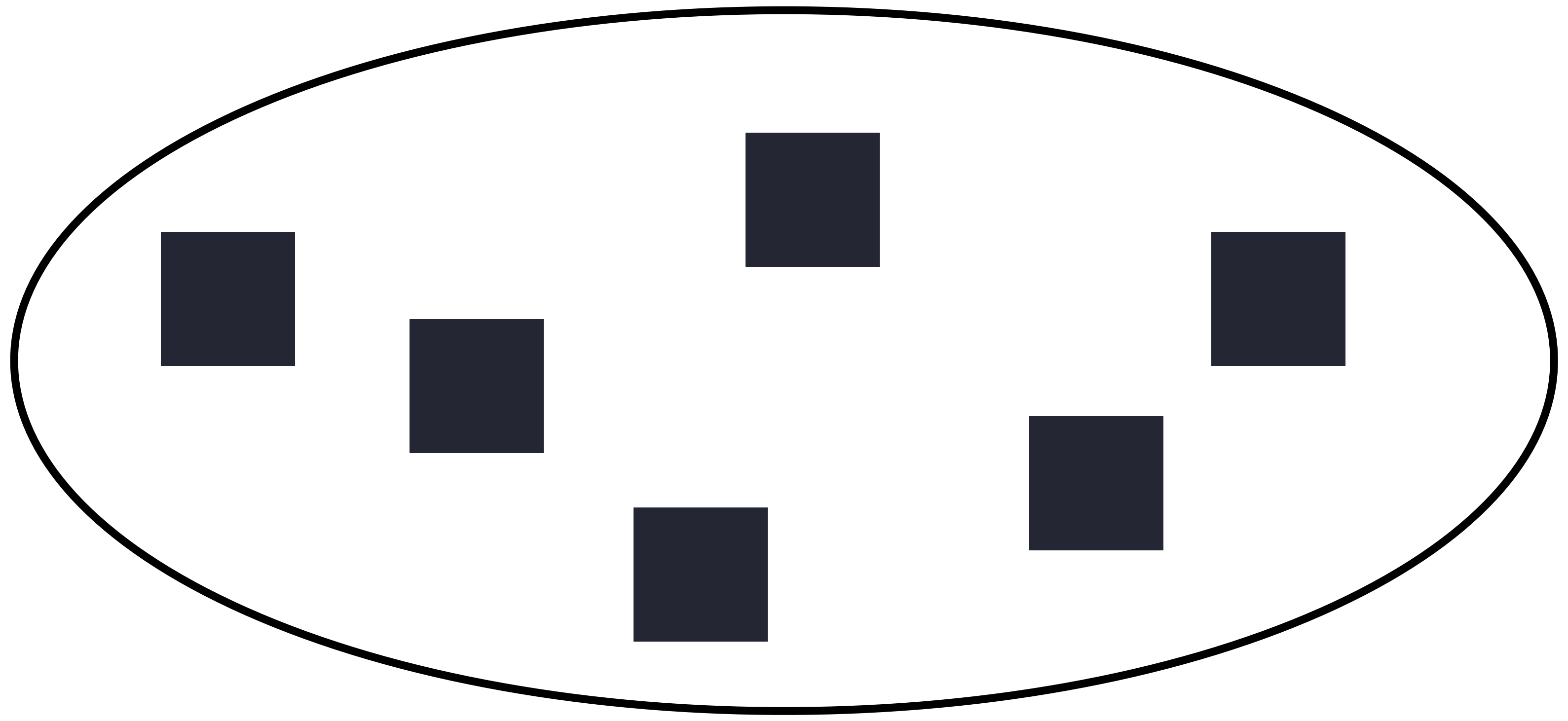
= “please store these bytes”

Step 2: validators validate and gossip transactions

- A **validator** is a network node that maintains an **unordered set** (“mempool”) of incoming transactions. It collects, validates and broadcasts transactions to other peers (using a **gossip** broadcast protocol)

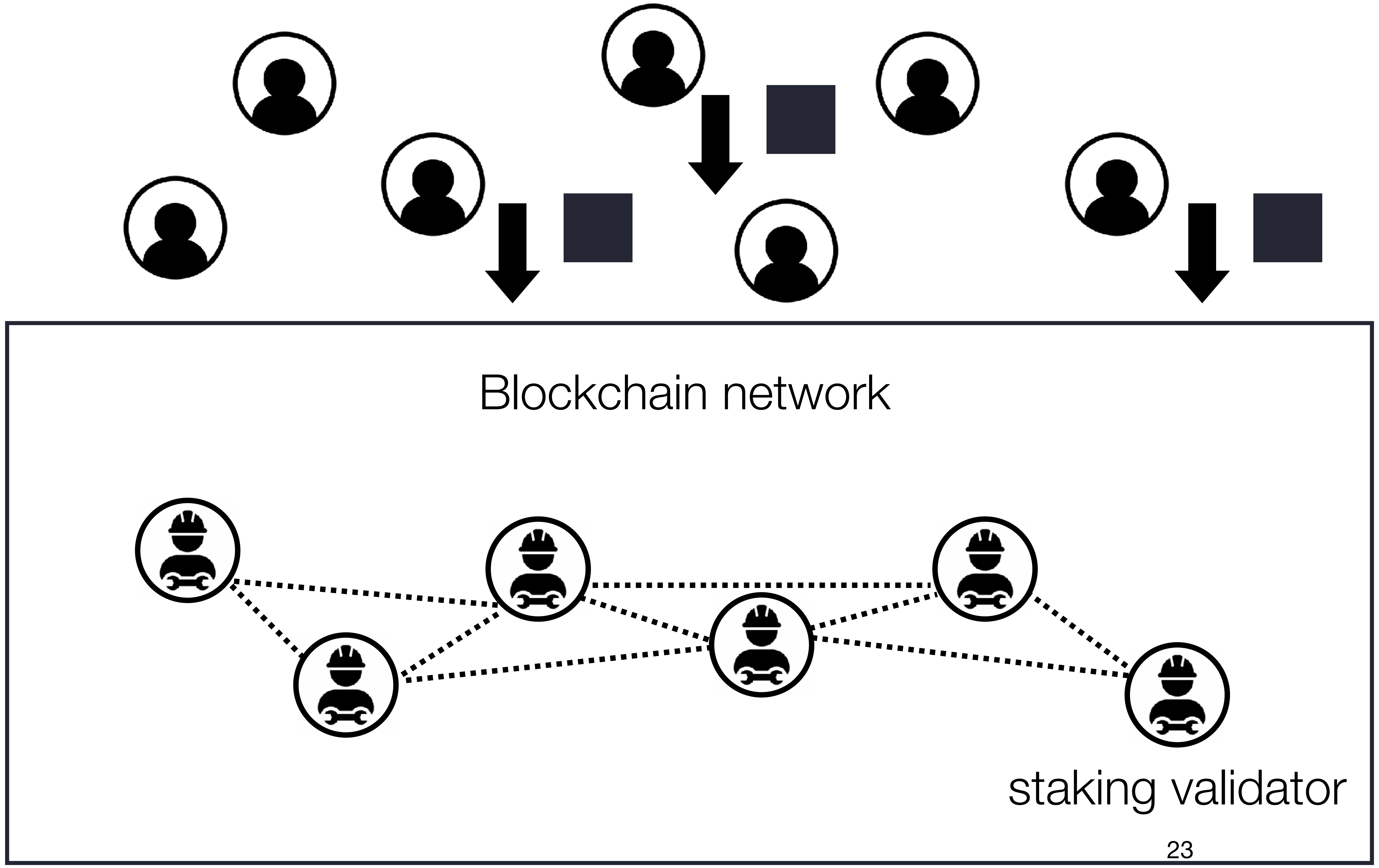


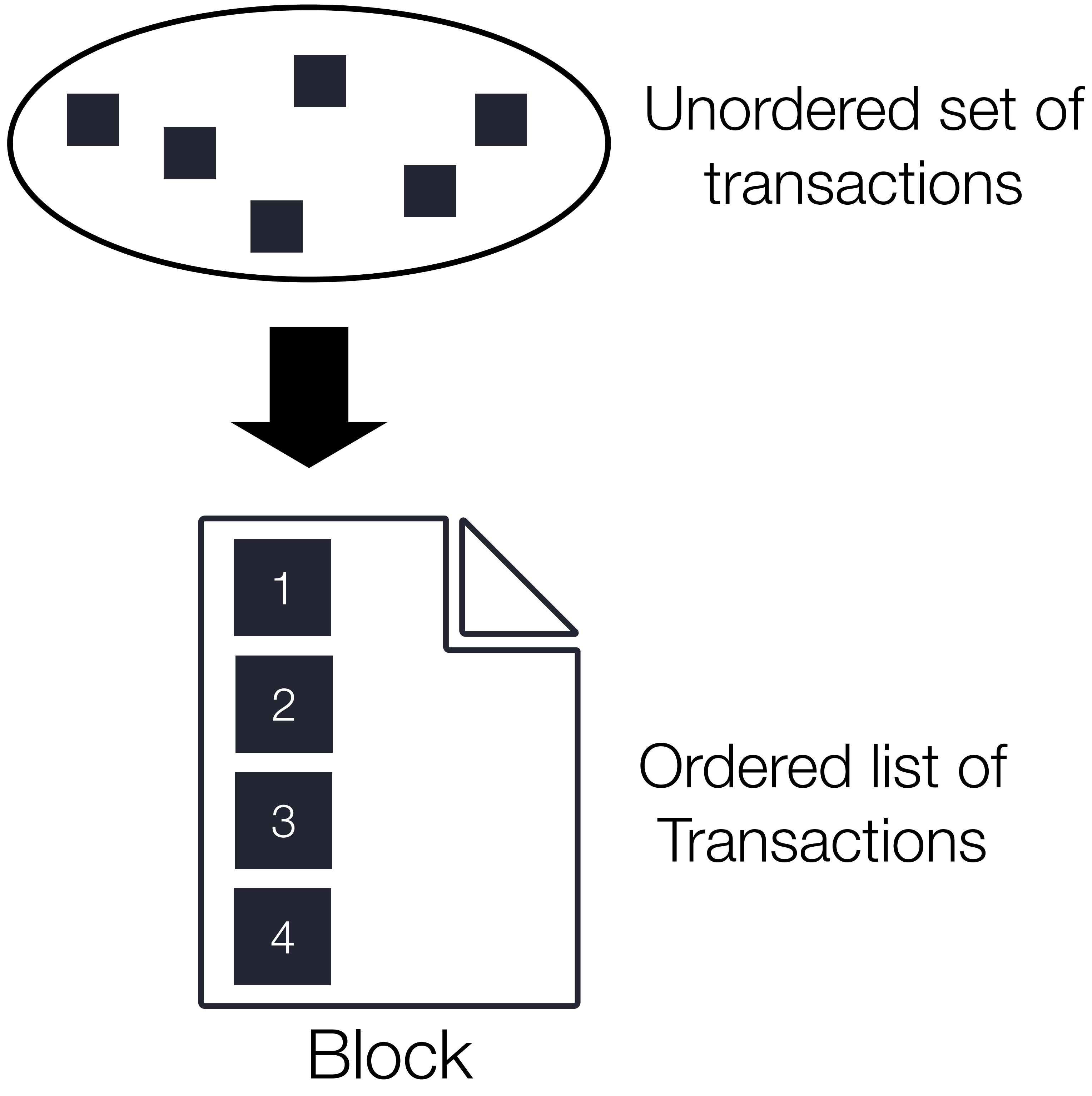
Unordered set of transactions



Step 3: a validator produces a block of transactions

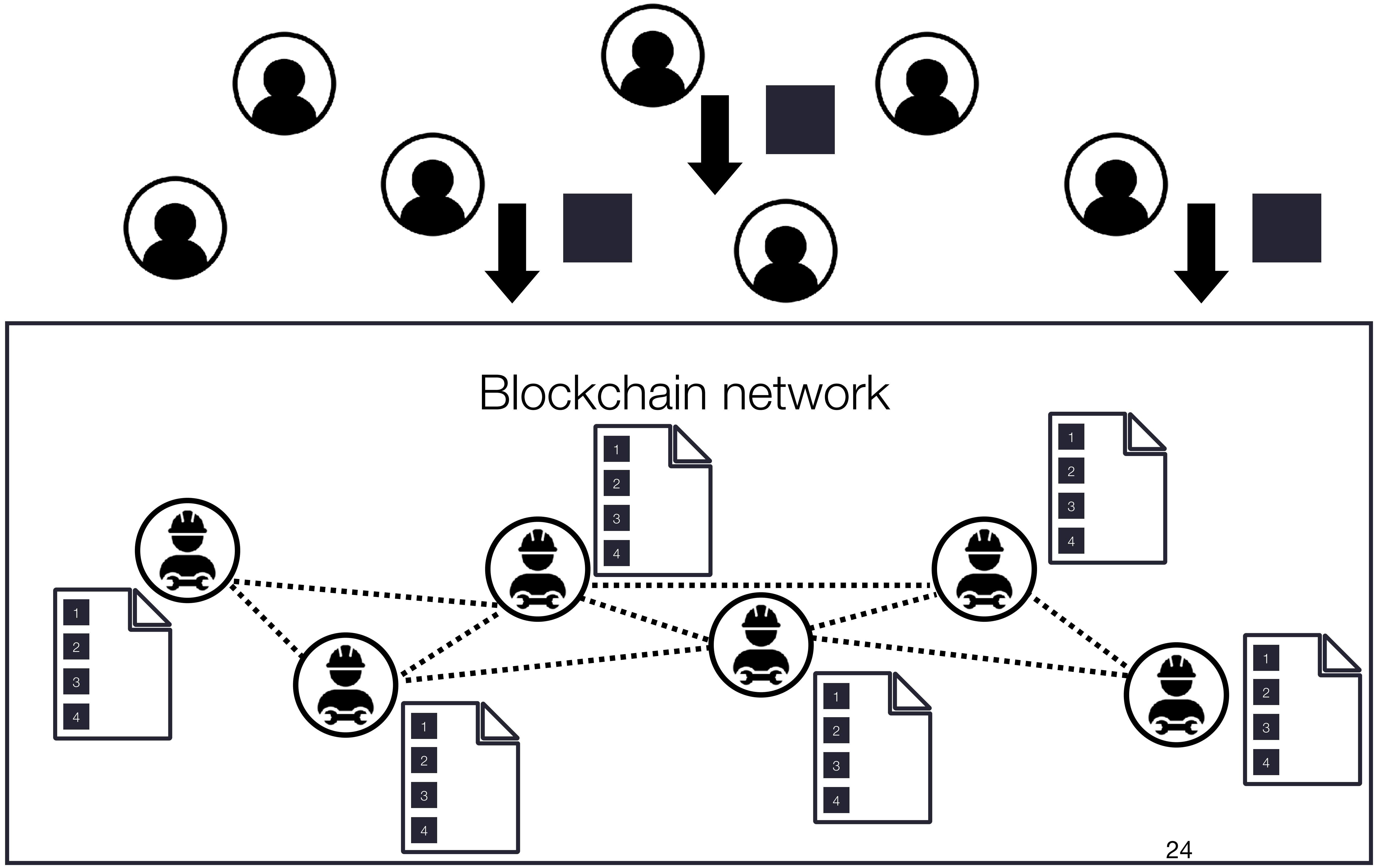
- At regular intervals, a subset of validators pick a subset of transactions from the pool and *sequence* them, thus producing an *ordered* list of transactions. These validators are sometimes called "**miners**" or "**staking validators**". The transaction list is called a "block"

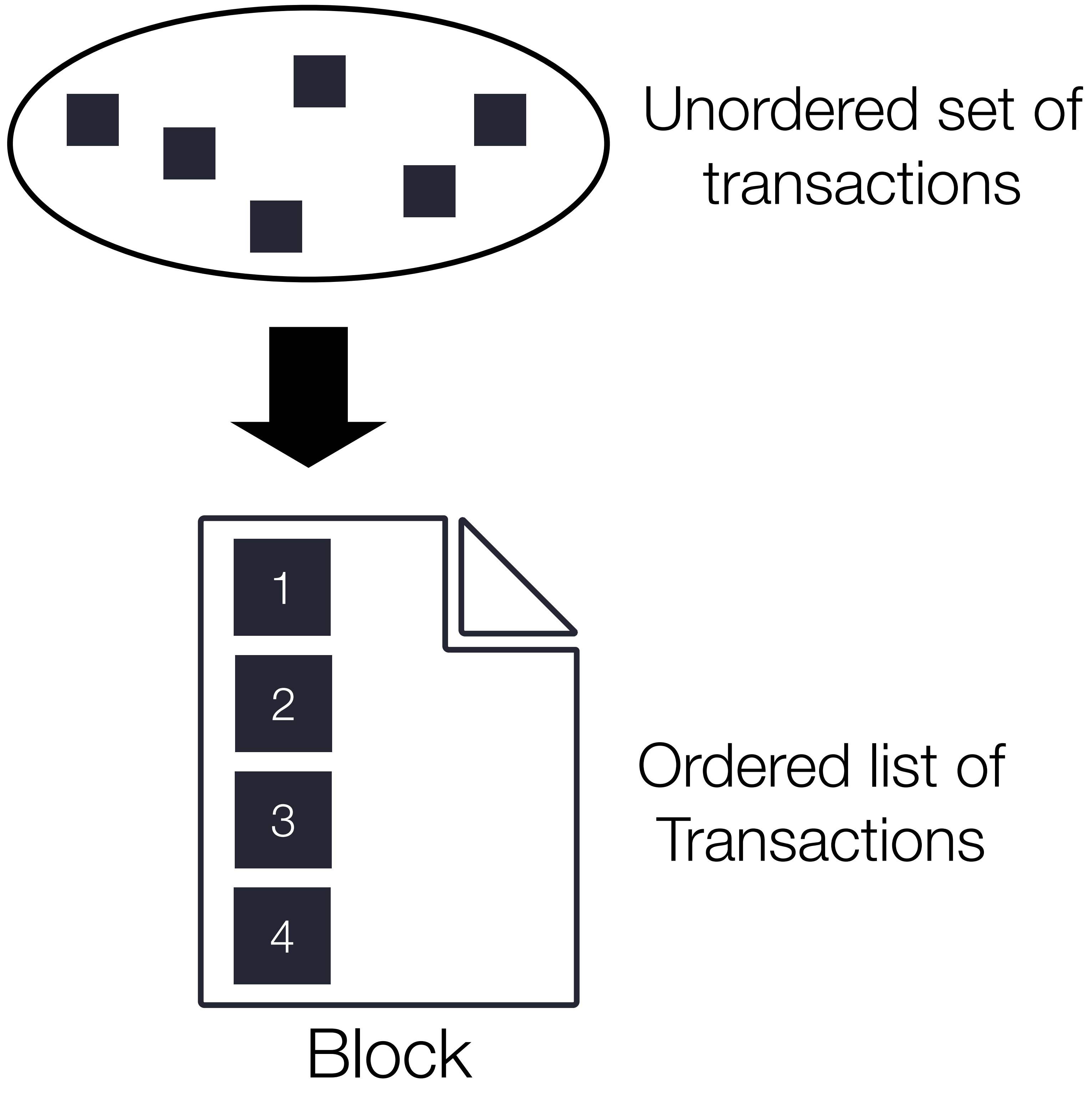




Step 4: validators gossip block and append to the blockchain

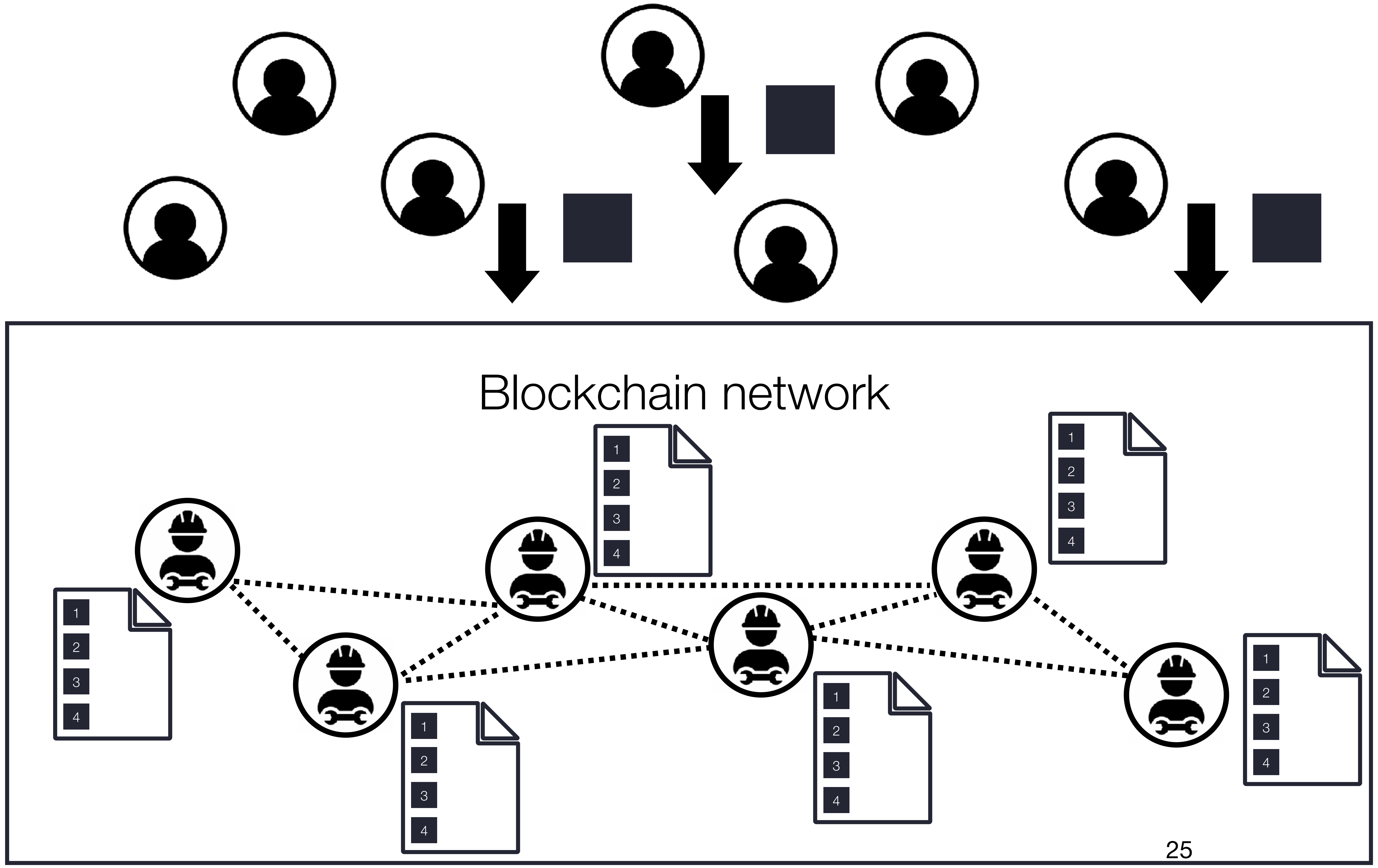
- The **block is broadcast** to all validators (again using gossip). Each validator **checks again** if all transactions in the block are valid. If yes, they **append** the block to their local transaction log (aka the blockchain).

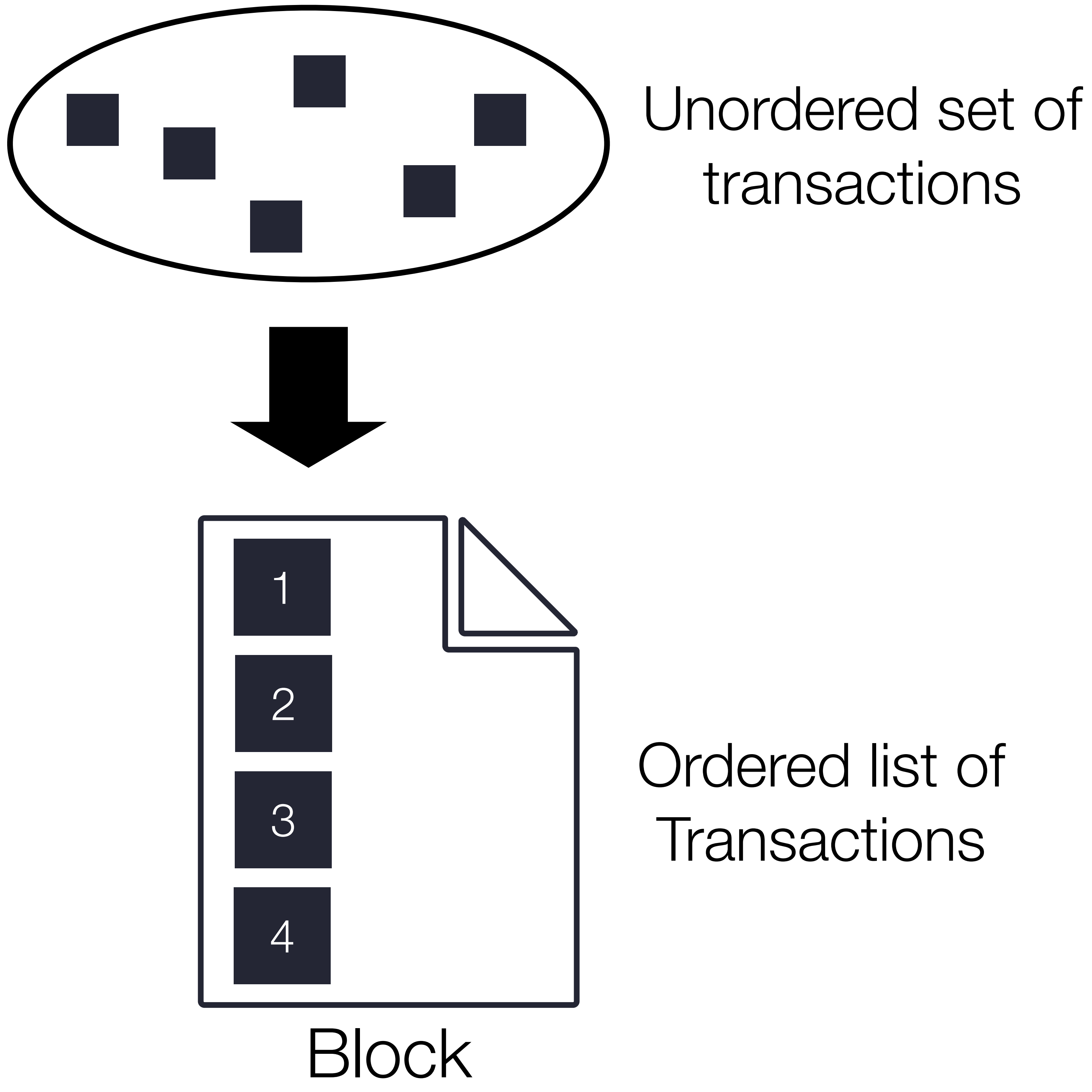




Consensus

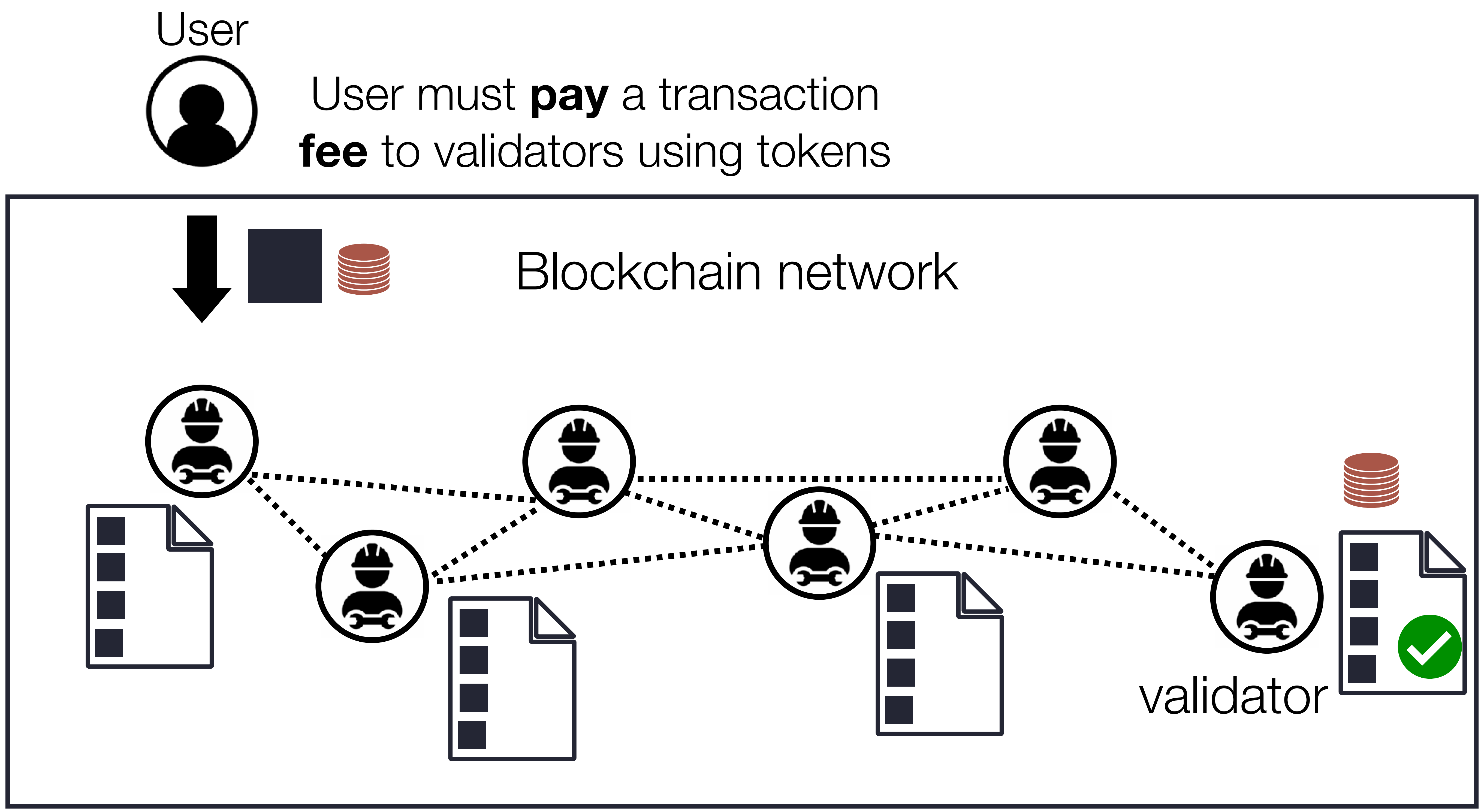
- All validators must reach **consensus** on the exact same transaction history!
- Need to make sure that blocks get appended everywhere *in the same order*





Blockchain networks: tokens, transaction fees and mining rewards

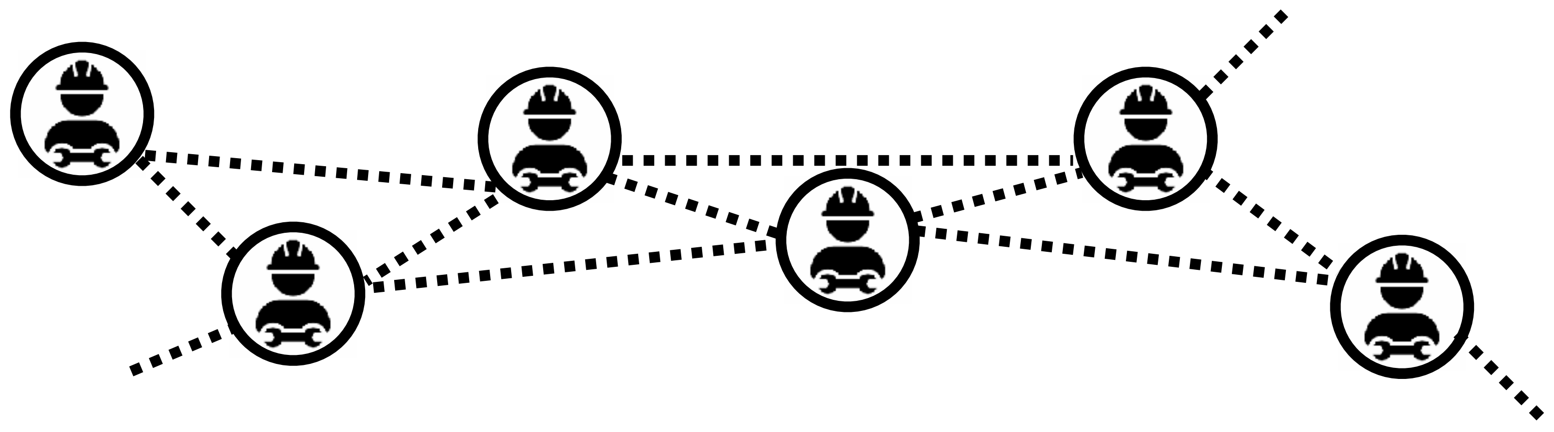
- **Tokens** are used to a) pay for transaction processing (transaction **fee**) and b) to **reward** validators for contributing hardware resources (compute, bandwidth, storage) to validate transactions. They act as an **incentive mechanism** to keep validators honest.



Validators can **earn** additional tokens
by producing valid blocks
(a process called “mining” or “staking”)

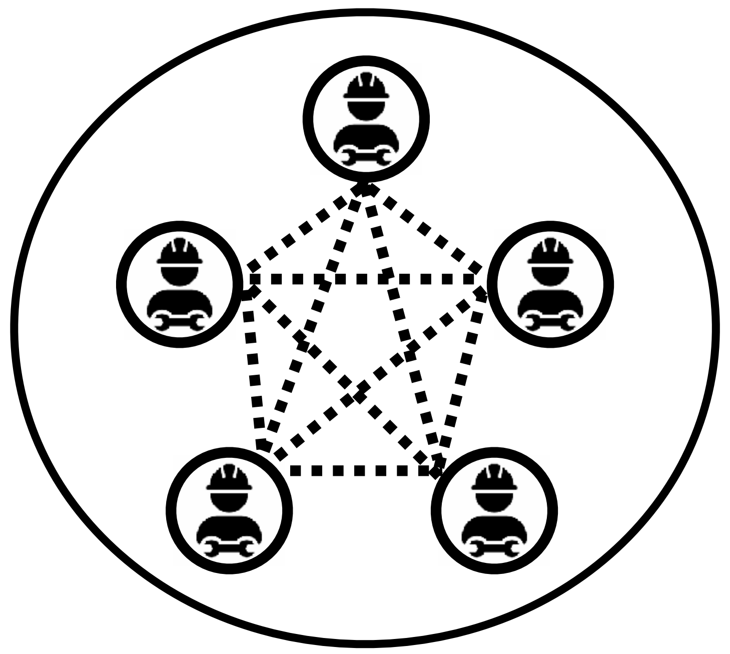
Who can be a validator?

- In **permissionless** blockchains: anyone can join the network to become a transaction validator. No need to ask for permission to anyone. Group membership is ***open***.
- In **permissioned** blockchains: must receive *permission* from a coordinator or from existing validators in order to become a transaction validator. Group membership is ***closed***.



Open

vs



Closed

KU LEUVEN DistriNet

Permissioned vs Permissionless networks: examples

- Examples of **permissionless** blockchain networks:
  - Bitcoin (decentralized payments)
  - Ethereum (decentralized computation)
  - Filecoin (decentralized storage)
  - Helium (decentralized wireless networks)
- Examples of **permissioned** blockchain networks:
  - Hyperledger Fabric
  - Corda
  - Private Ethereum networks (“Enterprise Ethereum”)
  - Hyperledger Sawtooth

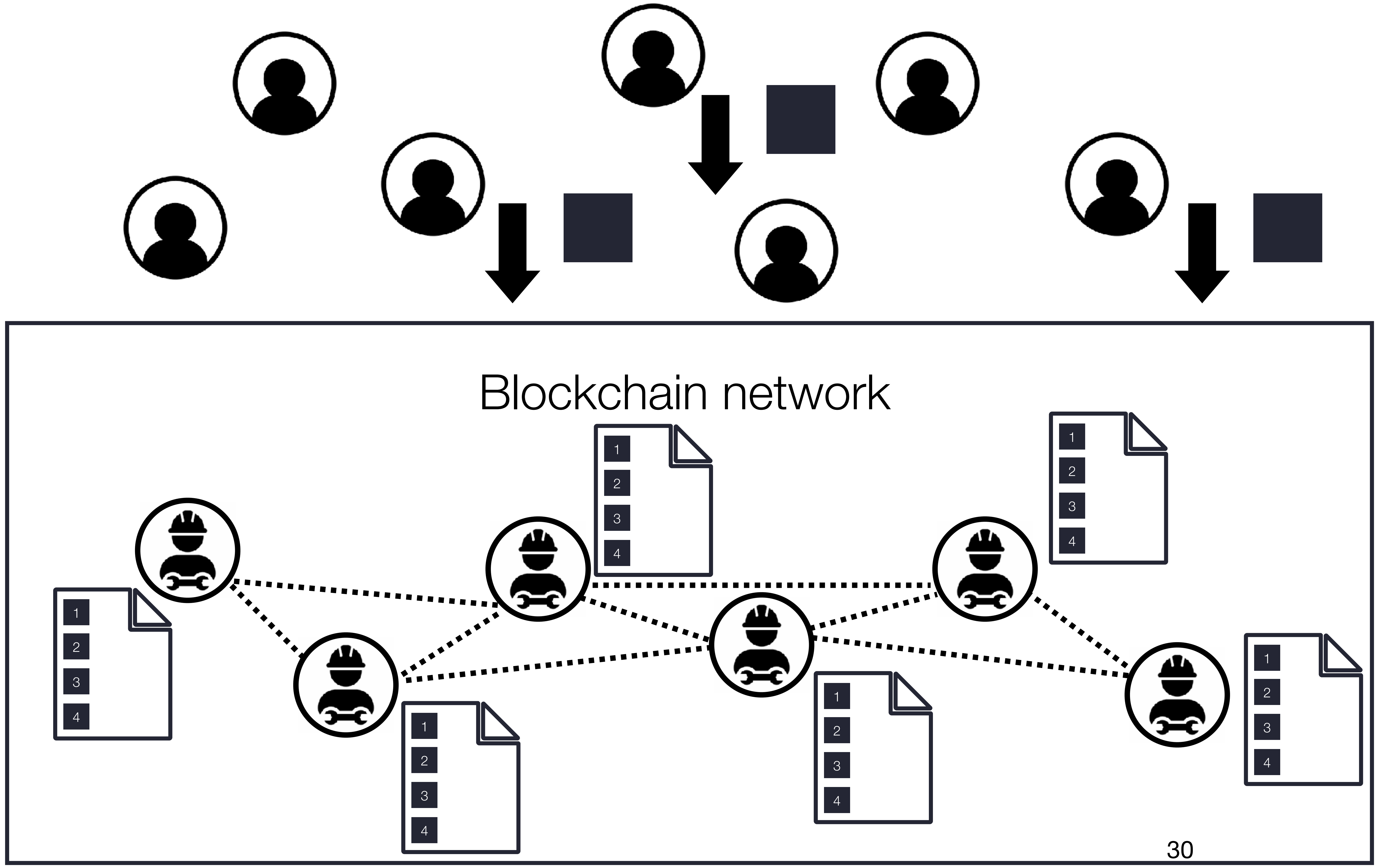
Bitcoin Ethereum

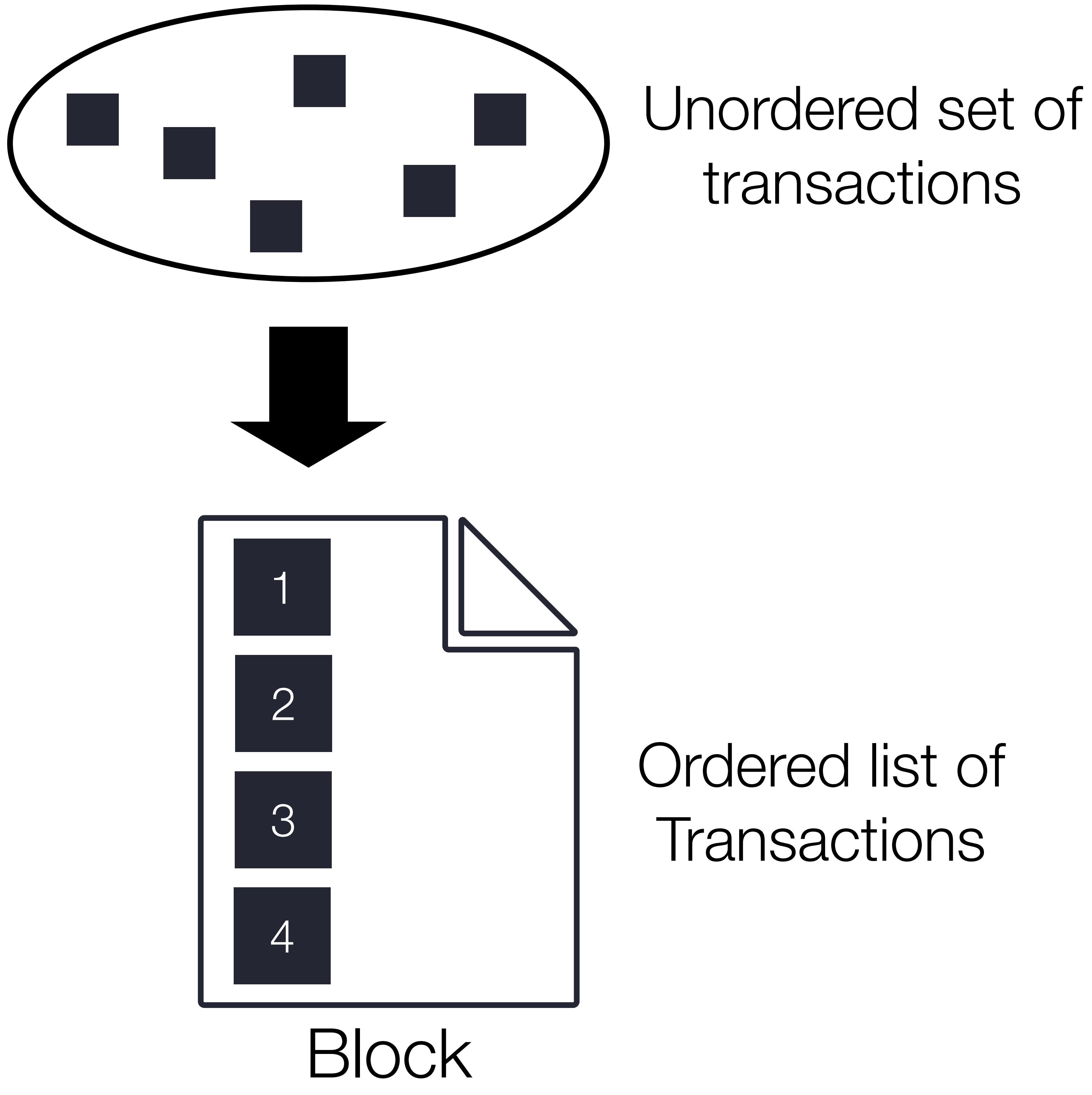
Filecoin Helium

Consensus in Blockchain networks

Consensus in Blockchain networks: recap

- All validators must reach **consensus** on the exact same transaction history!
- Need to make sure that blocks get appended everywhere *in the same order*

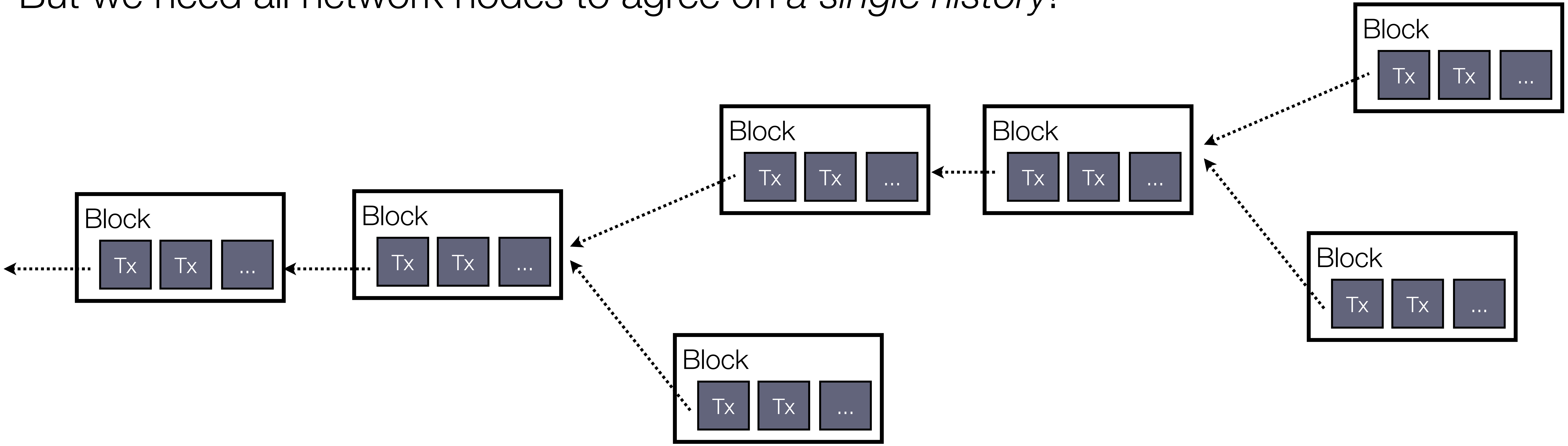




KU LEUVEN DistriNet

Problem: diverging histories

- In an open system, if *anyone* can easily produce a valid block and add it directly to the ledger, there is little hope that the network will end up agreeing on a *single* ledger
- More likely, we would end up with a quickly growing *tree* of blocks
- But we need all network nodes to agree on *a single history*!



How to get consensus: organize a vote?

- We can let the network **vote** to **elect a single validator node** to propose the next block
- Ideally the proposer node is chosen **randomly** to avoid any bias in the election process
- But how to organize a vote in an open and permissionless network?
  - 1. We don't even have a fixed list of nodes to organize a voting poll
  - 2. Even if we would have a list of nodes, how to assign **voting rights** to each one?
- One IP address = one vote? Problem: attacker may control multiple IP addresses
- This is known as a **sybil attack**. The same problem holds for any other type of "identity" that is cheap to create (e.g. public keys)

How to get consensus: organize a lottery!

- To elect a node from an open group of participants, organize a **lottery**: each node “buys” tickets, whoever can “prove” they have the lucky ticket is the winner (and so gets to propose the next block)

- The lottery should have the following properties:

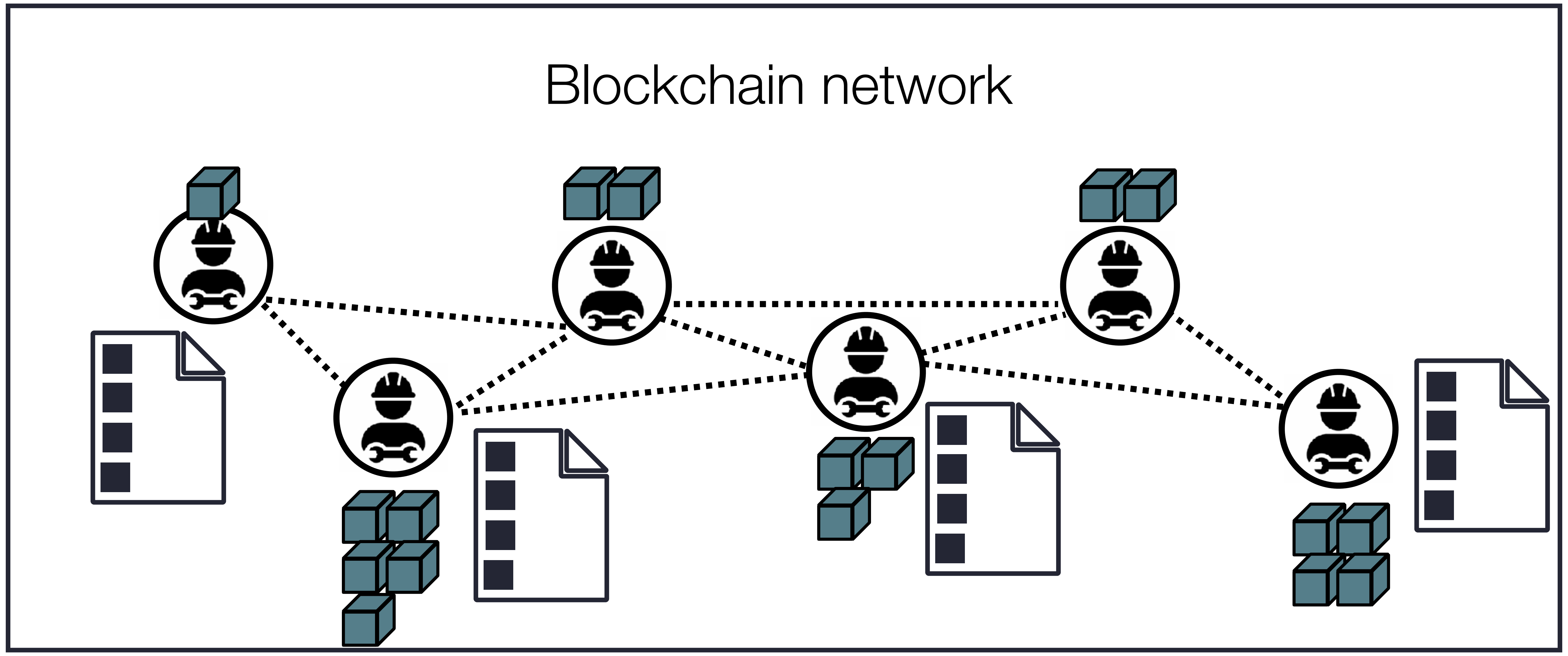
  - **Fair** - node election should be distributed across the broadest possible population of participants (i.e. “everyone can buy a ticket”)

  - **Proportional** - The cost of controlling the election process should be proportional to the value gained from it (i.e. “the more tickets bought, the higher the chance of winning”)

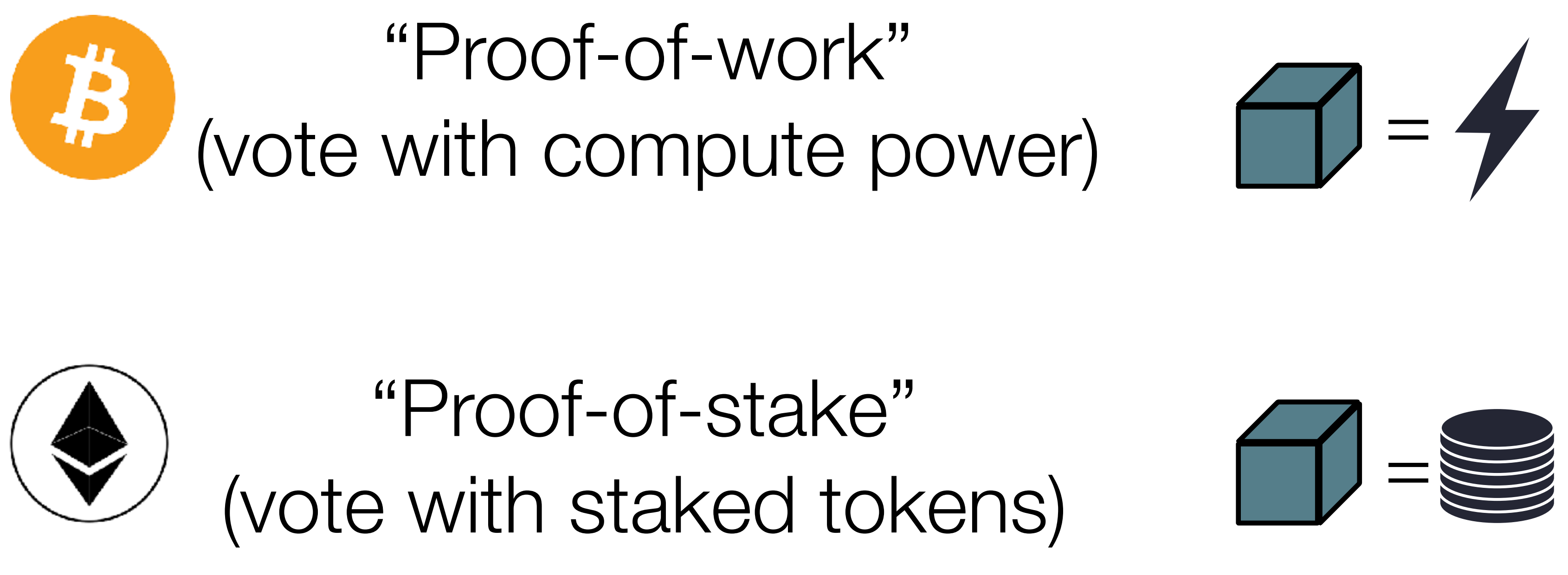
  - **Verifiable** - It should be relatively simple for all participants to verify that the winning node was legitimately selected (i.e. “everyone can verify whether the winning ticket is indeed a valid ticket”)

Lottery-based consensus in permissionless blockchains (“proof-of-X”)

- Validators enter the lottery by proving ownership of a digital or physically **scarce resource**
- Different blockchain networks may use different kinds of resources

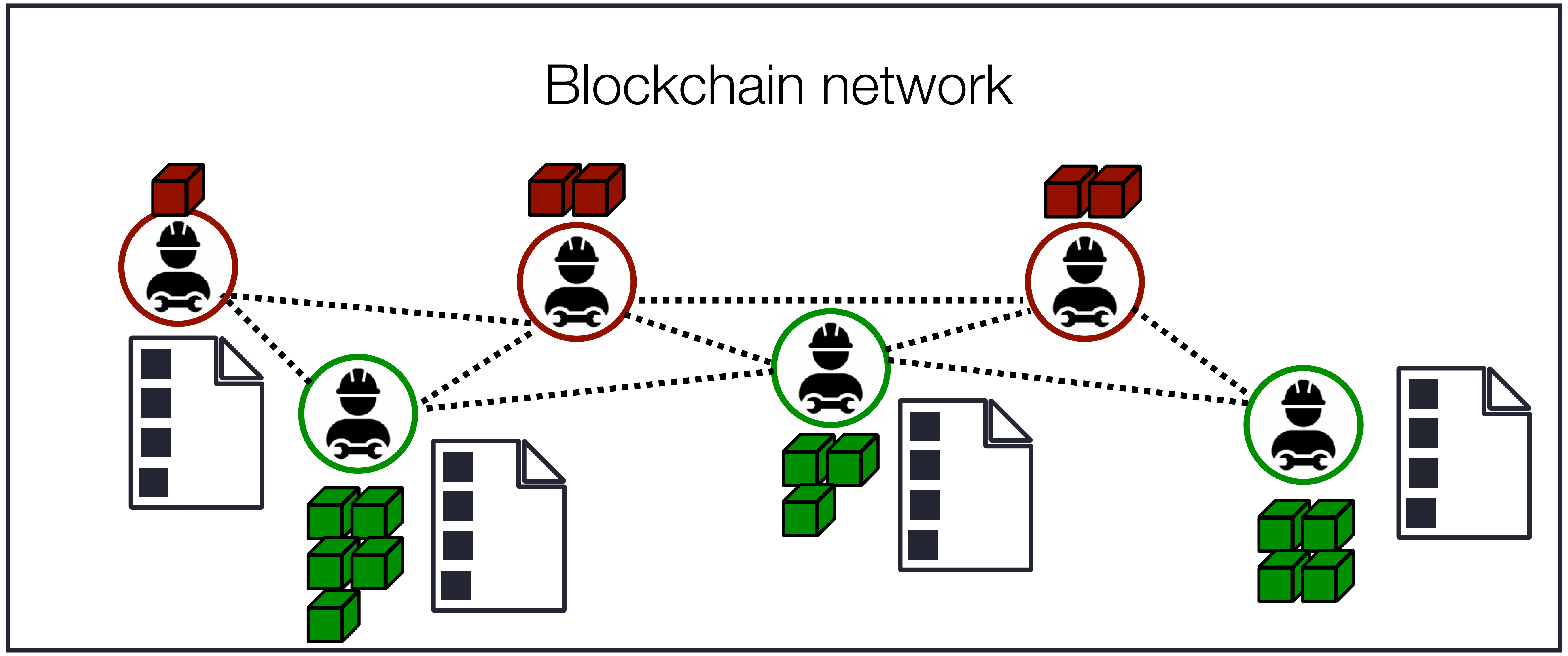


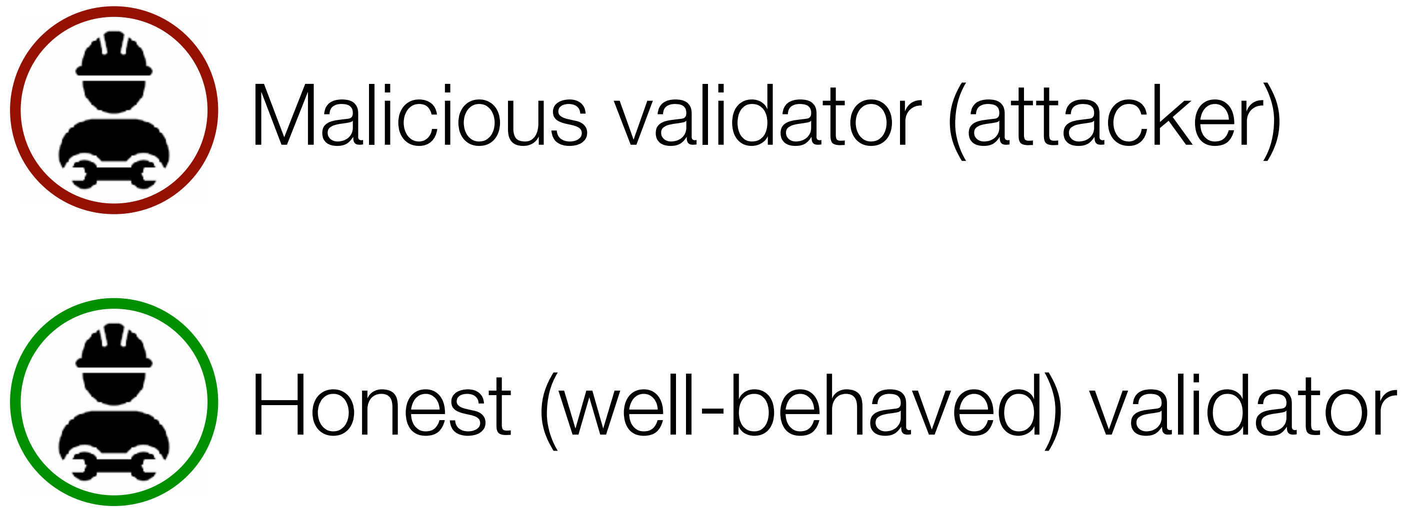
Example lottery-based consensus protocols:



Lottery-based consensus in permissionless blockchains (“proof-of-X”)

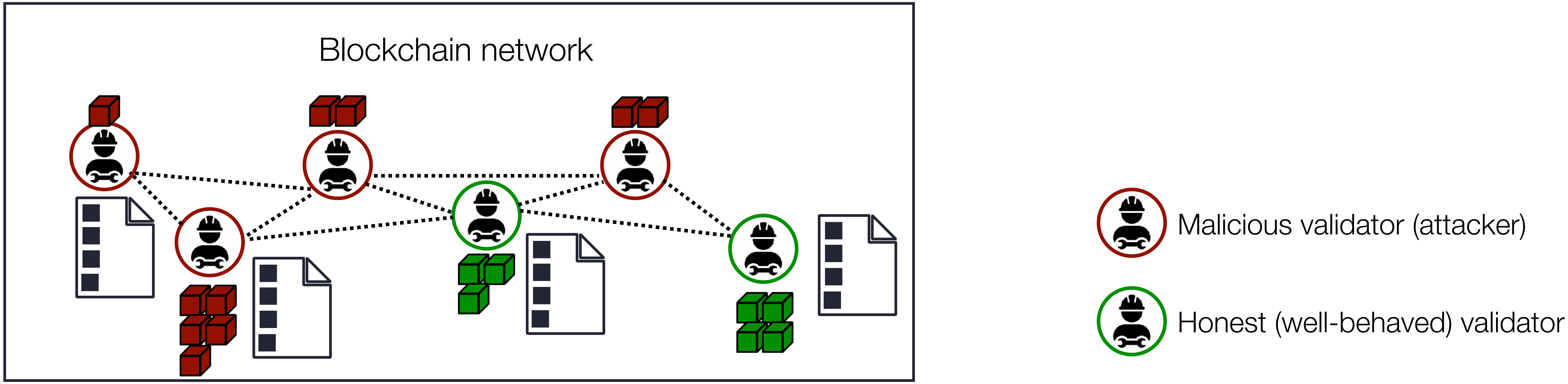
- The integrity of the blockchain is guaranteed as long as a **majority** of the network, **weighted** by their resource ownership, is controlled by well-behaved validators





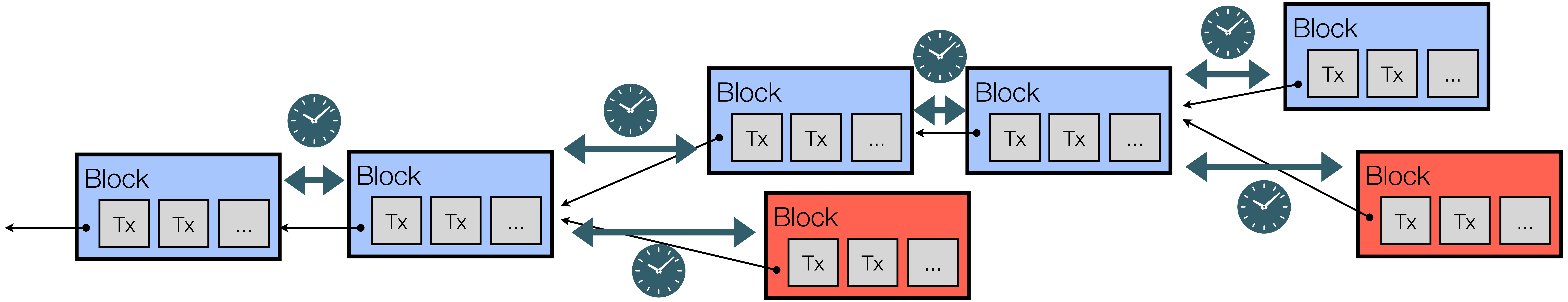
Attacking a permissionless blockchain network: “51% attack”

- If an attacker (or group of attackers) **controls >50% of the scarce resources**, they effectively control the production of new blocks.
- While such an attacker cannot create “fake” signed transactions (i.e. steal tokens), they can reject (**censor**) any number of transactions and can approve transactions that **double-spend** their own tokens by “forking” the blockchain and “rewriting” block history.



Proof-of-Work consensus

- Require blocks to contain a "proof-of-work": a proof that significant (computational) work was done to find the solution to a puzzle, where the solution - once known - is easy to verify
- The purpose is to **slow down** block production
  - so that only **one node at a time** can propose a new block
  - so that there is time to **propagate the new block** across the entire P2P network to avoid disagreement on what is the latest valid block
- In Bitcoin: propose a new block on average every 10 minutes

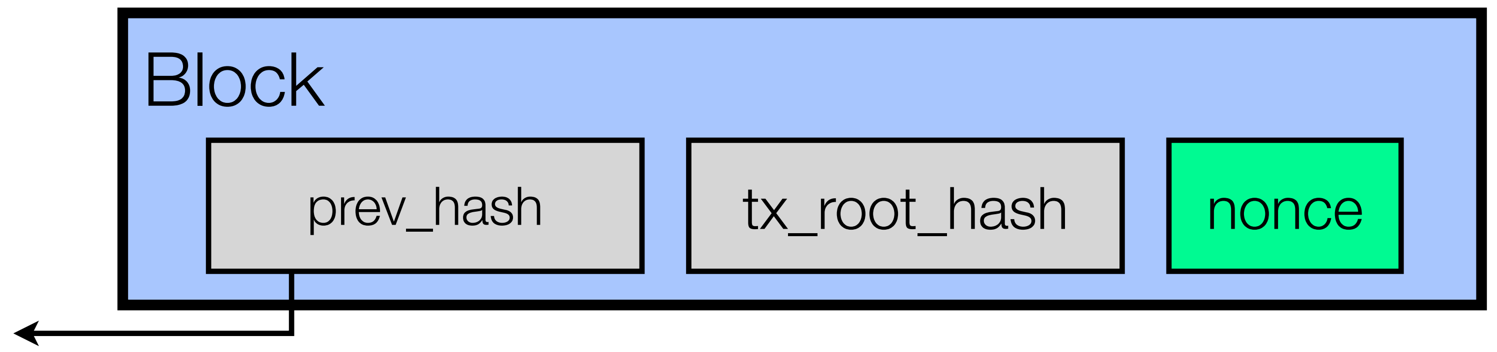


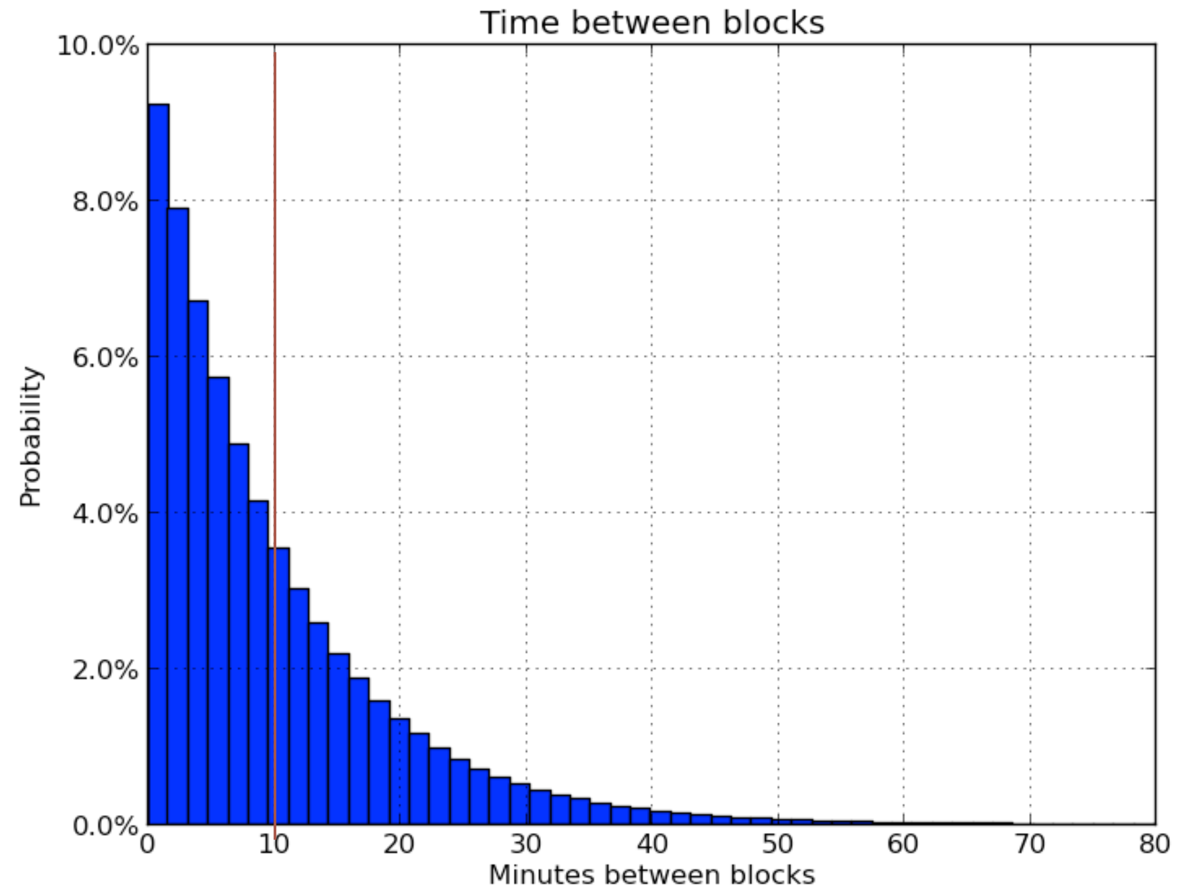
Proof-of-Work in Bitcoin

- The proof-of-work involves searching for a value *v* such that hash(*v*) is smaller than a given target threshold value (known as the **difficulty** parameter)
- Because the output of a cryptographic hash cannot be predicted, there is no known strategy better than a **brute force** search
- The search is done by incrementing a number in the block (the **nonce**) until a value is found such that the block's hash satisfies the target difficulty
- The difficulty parameter is **adjusted** every 2016 blocks such that the **average time** between blocks remains 10 minutes.

Find a nonce (a number) such that:

$$H\text{ (nonce || prev\_hash || tx\_root\_hash) < target}$$



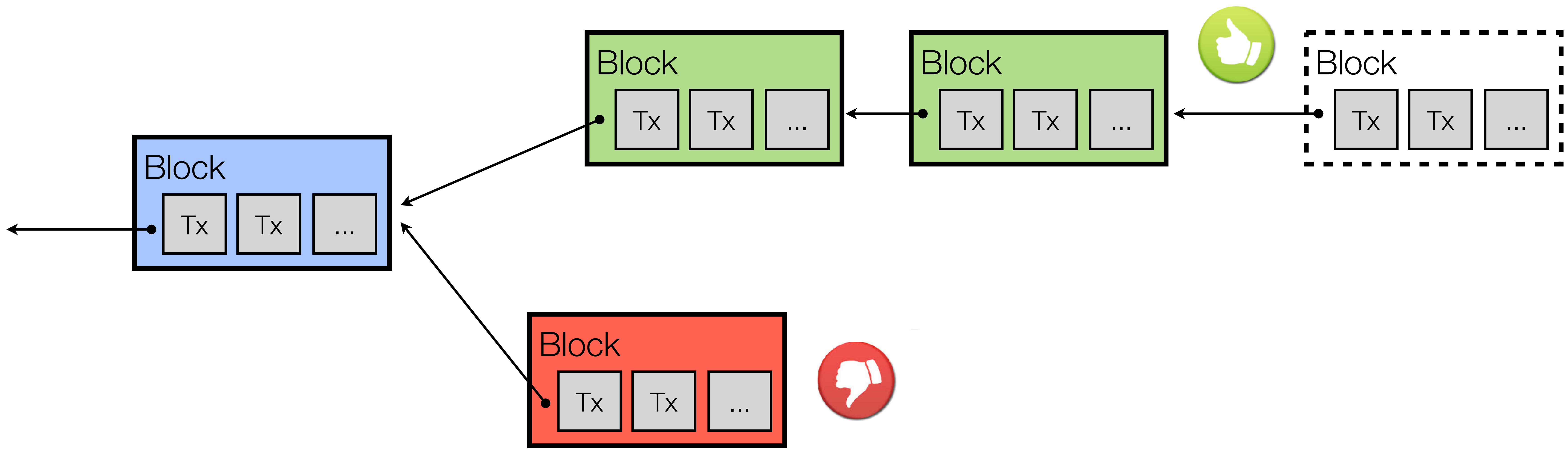


(source: Bitcoin wiki, retrieved November 2022)

KU LEUVEN DistriNet

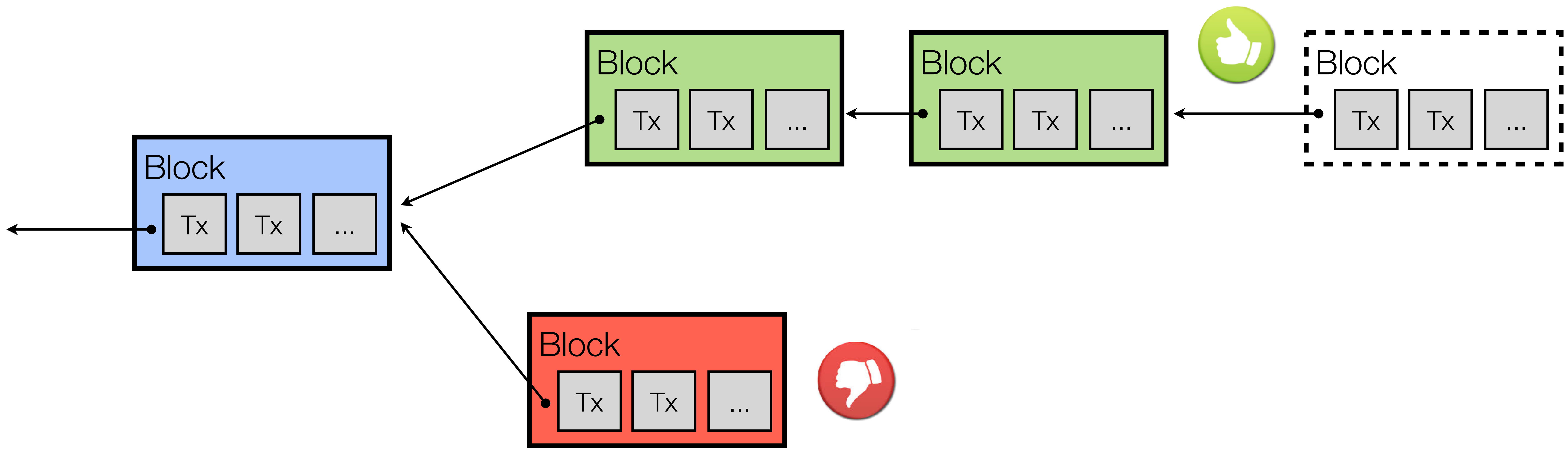
How Proof-of-Work solves the consensus problem

- Nodes implicitly “vote” with their computational power
- Nodes silently accept a block by working on extending the block (= mining)
- Nodes silently reject a block by refusing to work on it
- The majority decision is represented by **the longest chain**, which has the greatest proof-of-work effort invested in it.



How Proof-of-Work solves the consensus problem

- If a majority of compute power is controlled by honest nodes, the honest chain will grow the fastest and outpace any competing chains.
- Put differently: an attacker must control more compute power than **all the honest nodes combined** in order to outpace the "honest chain"

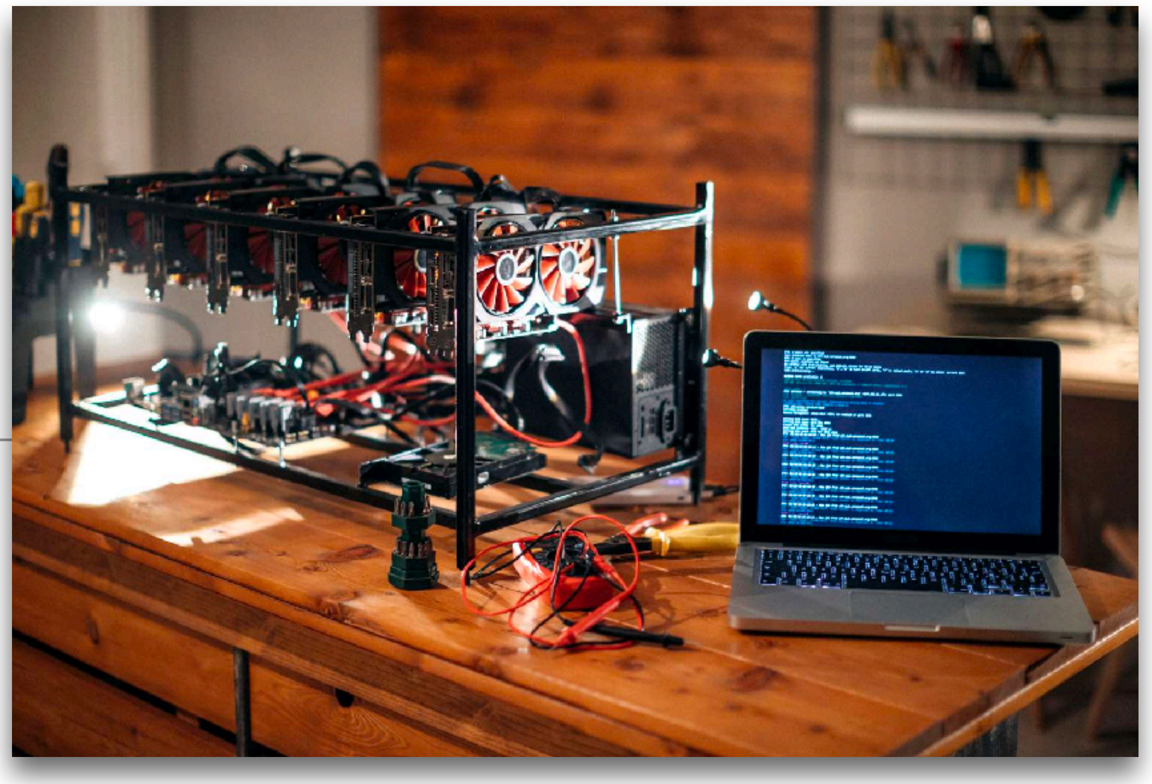


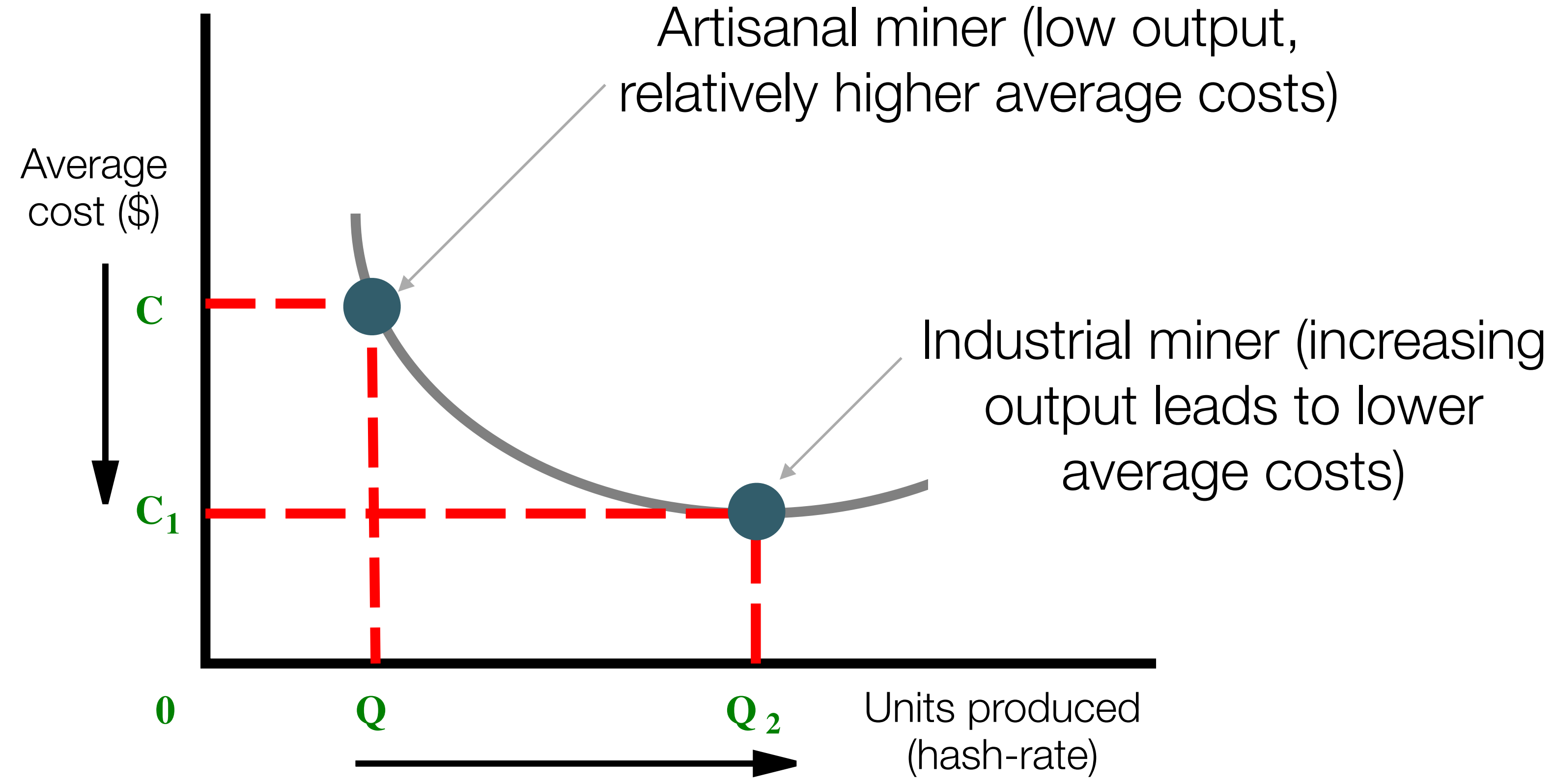
KU LEUVEN DistriNet

Proof-of-Work

- Miners “race” each other to find the next block. The more computational power a miner has, the higher the chance of winning the race.
- But Proof-of-Work is:
  - Slow (by design)
  - Energy-inefficient (by design)
  - Subject to centralizing economies of scale (mining pools, large-scale mining facilities)

A small-scale GPU-based Bitcoin “mining rig”
(Image credit: investopedia.com)





A large-scale ASIC-based Bitcoin “mining farm”
(Image credit: stockhouse.com)

Proof-of-Stake

- Proof-of-Stake (PoS): the chance of proposing the next block is **proportional to the economic stake** in the system.
    - The more tokens “staked” (= locked in escrow), the higher the chance of becoming the next block proposer.
- Many **variations** of PoS **exist**. Two large families include:
    - **Lottery-based** Proof-of-Stake. Similar to Proof-of-Work. Also called **chain-based** Proof-of-Stake.
    - **Voting-based** Proof-of-Stake. Uses a BFT algorithm like PBFT or similar. Also called **BFT-based** Proof-of-Stake.

Two families of voting-based consensus algorithms

- **Crash fault-tolerant (CFT) consensus**: assume participants may fail due to crashes or network failures, but also assume all participants execute the consensus algorithm correctly and strictly follow the same protocol.
  - Tolerate up to (but not including) 1/2 participants failing by crashing ("fail-stop")
  - Example: Paxos (Lamport, 1989)
- **Byzantine fault-tolerant (BFT) consensus**: assume participants may fail due to crashes or network failures, but make no additional assumptions. In particular, processes may incorrectly execute the consensus algorithm and may deviate from the protocol in arbitrary ways.
  - Tolerate up to (but not including) 1/3 processes failing in arbitrary ways
  - Example: PBFT (Castro and Liskov, 1999)

PBFT in Permissionless blockchains: Tendermint

- Tendermint is an **adaptation of PBFT** to function as a consensus algorithm for a **permissionless** blockchain *without requiring Proof-of-Work mining*

- It was one of the first attempts (~2014) to adapt classical (pre-blockchain) BFT algorithms such as PBFT to support consensus in the context of a blockchain:

  - Tendermint uses **"Proof-of-Stake"** to limit participation in the committee and to introduce token incentives (rewards and penalties).

  - Rather than seeking agreement on individual *operations*, peers take turns **proposing the next block** in the chain. If a +⅔ quorum agrees on the block, it is added.

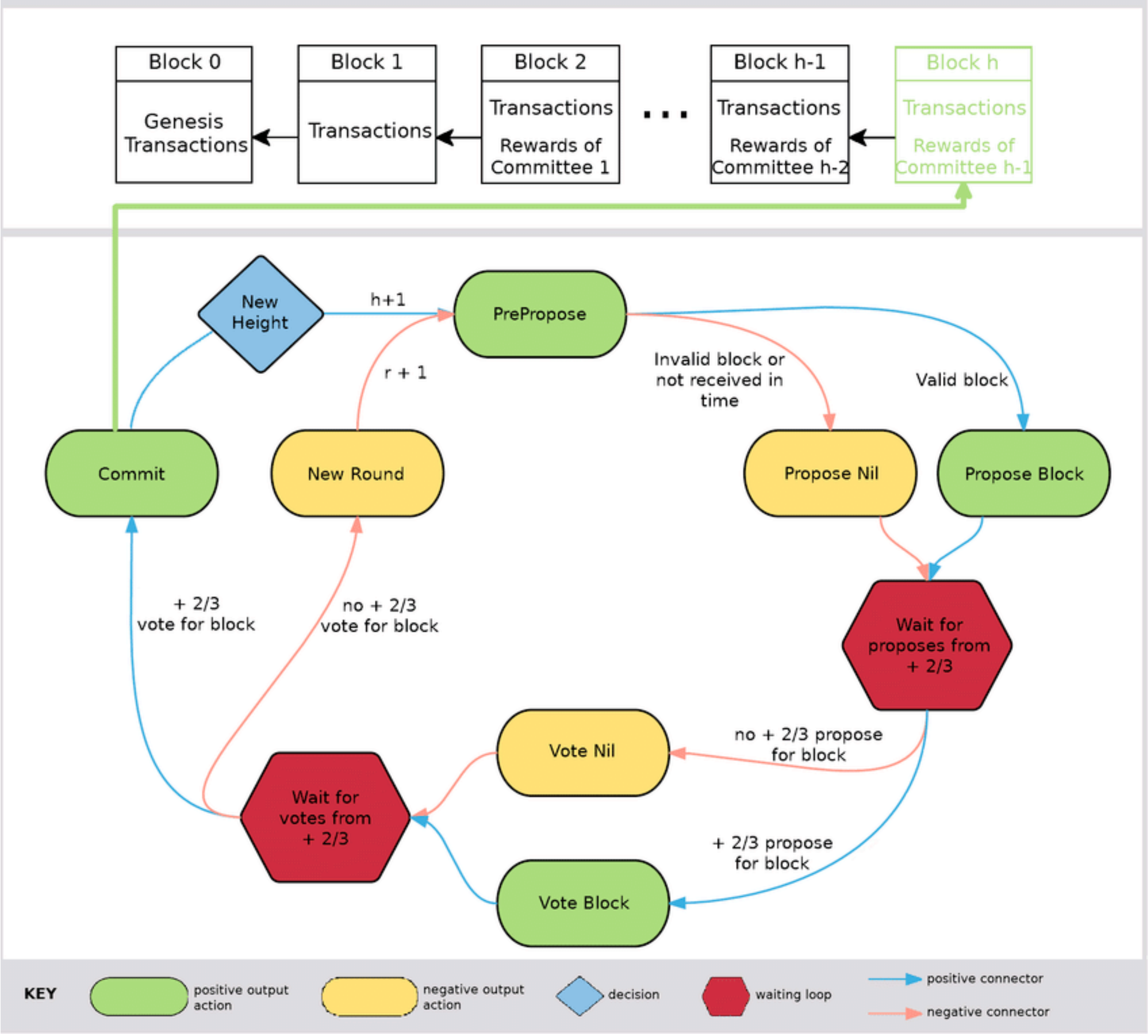
  - Tendermint assumes a large, slow, wide-area network rather than a small, fast, local-area network. Therefore, as in Bitcoin and Ethereum, peers use **gossip** communication.

  - Tendermint supports **dynamic group membership** safely by requiring a +⅔ quorum of validators to approve of membership changes. It is commonly used along with Proof-of-Stake so that only peers that can prove ownership of staked tokens can participate.

  - Tendermint supports **slashing** of staked tokens when validators are observed to deviate from the protocol.

- Tendermint is used as the consensus algorithm in the **Cosmos** project and the Cosmos Hub permissionless blockchain. The latest version is now known as "CometBFT".

Tendermint



(Source: Lagaillardie, N.; Djari, M.A.; Gürcan, Ö. A Computational Study on Fairness of the Tendermint Blockchain Protocol. *Information* 2019, 10, p. 378)

Proof-of-Stake consensus in Ethereum: Beacon Chain protocol

- Nodes have to **stake minimum 32 ether** into a deposit contract as collateral to become a **validator**. Some or all of a validator's stake can be destroyed if it is found to be dishonest. Nodes thus have strong **economic incentives** to remain honest.

- Ethereum works with fixed time slots. In every time slot (spaced 12 seconds apart) a validator is **pseudorandomly selected** to be the block **proposer** and another group of nodes is pseudorandomly selected to form a **committee** (with a minimum size of 128).

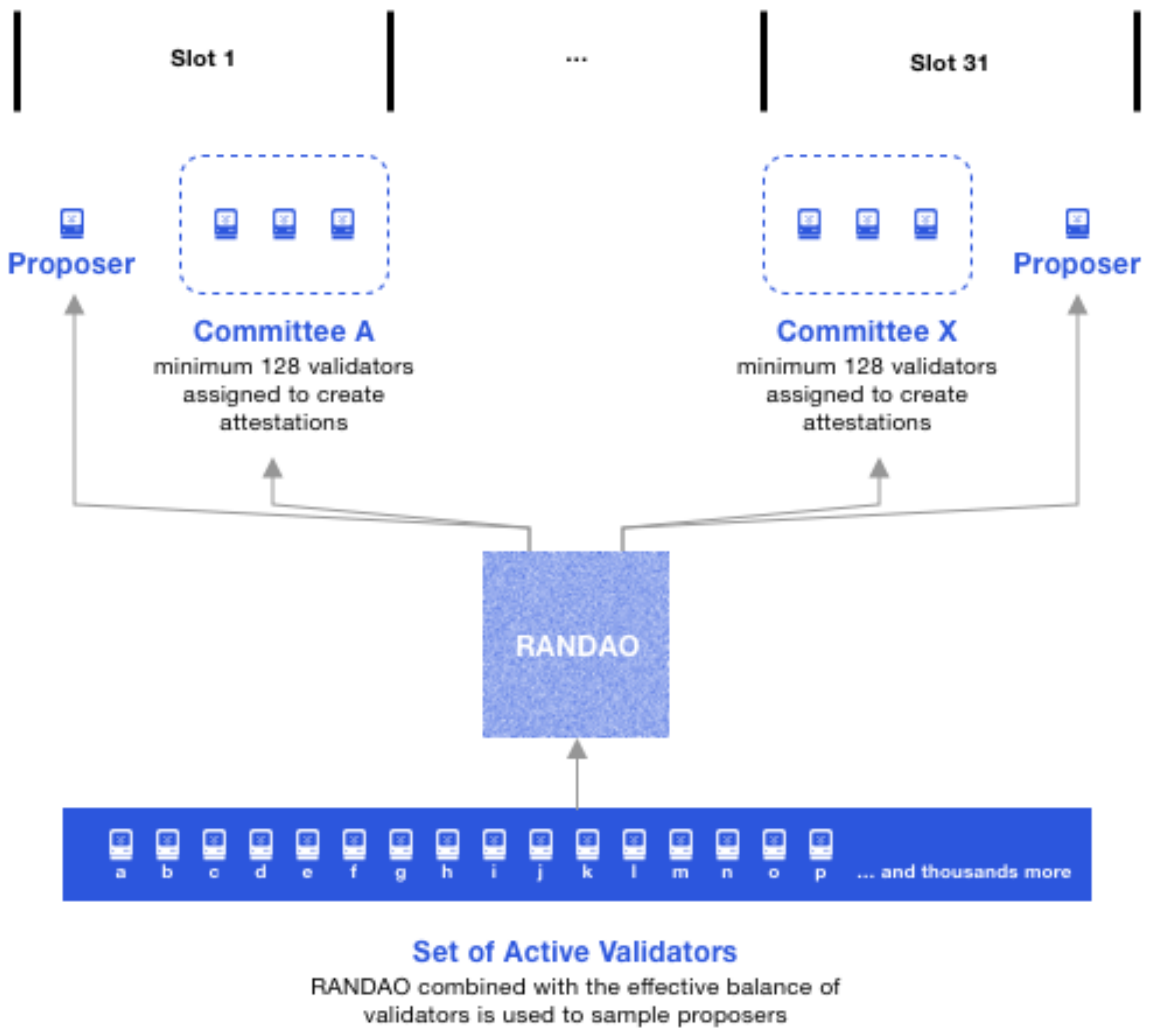
- The chance to get elected as the block proposer is **proportional to a validator's staked funds**.

- The **proposer** bundles transactions, executes them and determines the new system state. They wrap this information into a block and broadcast it to the committee.

- When validators in the **committee** receive the block, they re-execute the transactions to ensure they reach the identical new system state. If they agree, they **attest** to the validity of the block by signing it. **A 2/3 majority weighted by stake is needed** to finalize the block.

- Validators sign blocks using **BLS** signatures (for compact **signature aggregation**: only 96 bytes per aggregate signature)

- If a validator sees two conflicting blocks for the same slot they pick the one with the **greatest economic weight** of block attestations (LMD-GHOST fork-choice rule).



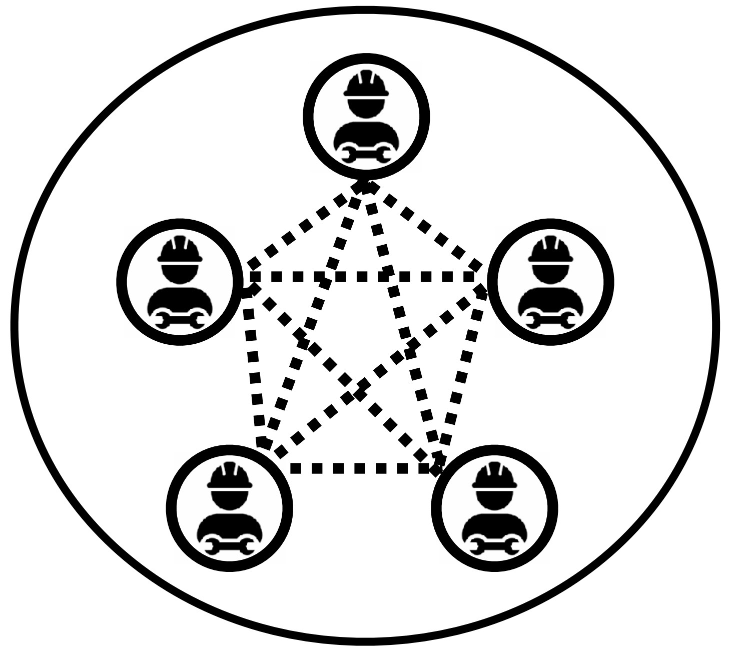
(Image credit: Consensys. Source: consensys.net, February 2020)

Permissioned Blockchains

- Avoid the privacy and scalability challenges of permissionless blockchains by **limiting** readers and/or writers **to** a set of **authorised parties only**

- This **avoids** the **sybil attack** problem and the need to use “**lottery**”-based consensus (Proof-of-Work, Proof-of-Stake, …)

- Instead, use standard “**voting**”-based consensus algorithms such as PBFT



HYPERLEDGER FABRIC

corda

ENTERPRISE ETHEREUM ALLIANCE

HYPERLEDGER SAWTOOTH

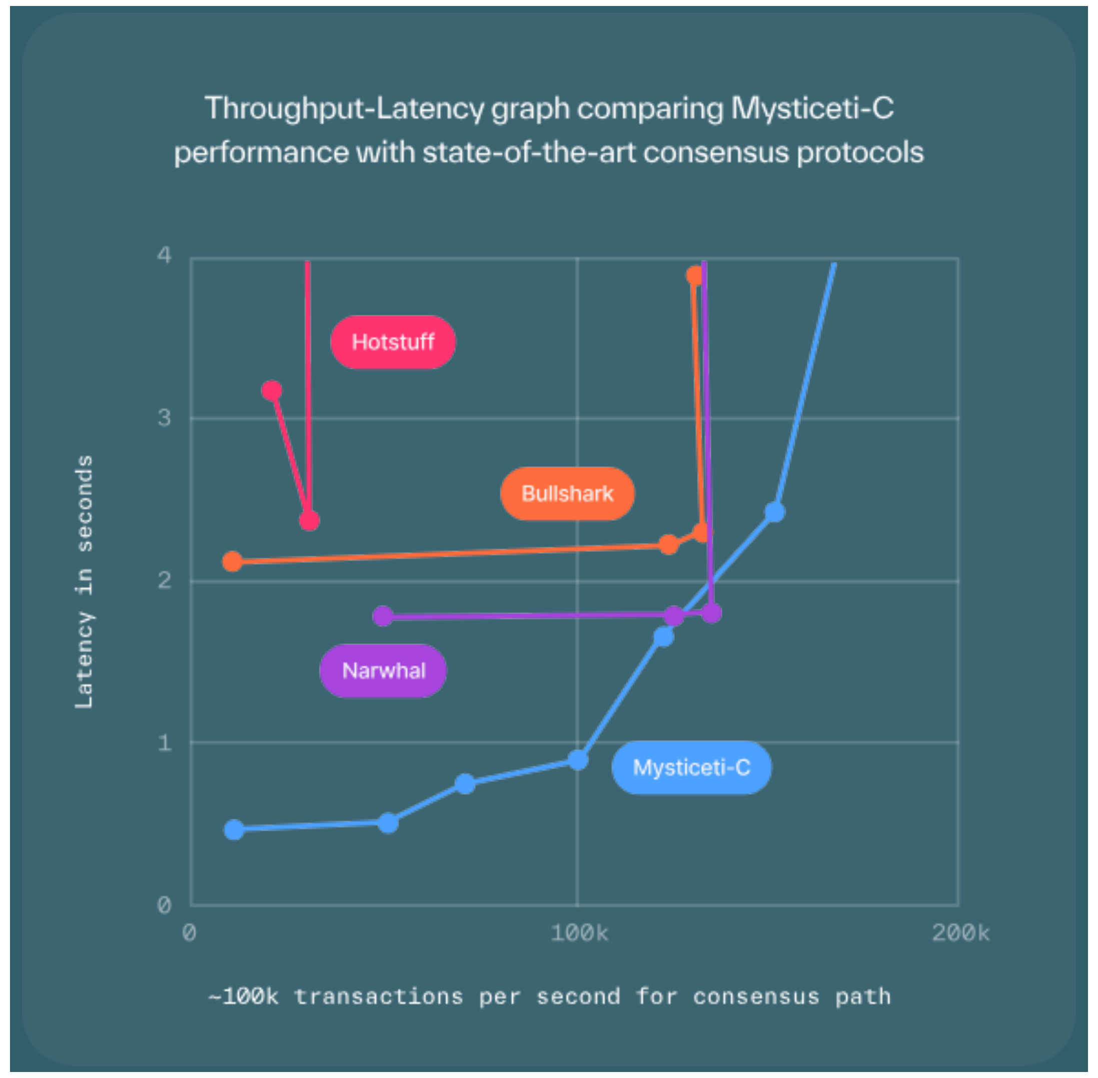
Permissioned vs Permissionless Blockchains: summary

	Permissionless	Permissioned
Network peers	Validator nodes are **pseudonymous**.	Validator node **identity** is usually **revealed**.
Peer membership	**Open** (anyone can join, no need to ask "permission" to join)	**Closed** (an administrator authorizes membership, or pre-existing members vote to update the membership list)
Network size	Scales to **large** number of peers (>1000 nodes)	Usually **small** (< 100 nodes)
Network connectivity	**Low** (not all peers may be able to connect to all other peers)	**High** (often fully connected - all nodes can reach all other nodes)
Consensus achieved via	**Lottery**-based algorithms, based on proof of owning some scarce resource (e.g. Proof-of-Work, Proof-of-Stake)	**Voting**-based algorithms, such as Byzantine Fault-tolerant (BFT) consensus algorithms (e.g. PBFT)
Transaction throughput	**Low** (~10 TPS for Bitcoin, Ethereum). Generally: the larger the network, the lower the TPS.	**High** (10,000 or more TPS) (TPS = transactions per second)
Transaction & Block finality	**Probabilistic & slow** (blocks are considered final only after being extended by enough newer blocks, which can take 10s of minutes)	**Deterministic & fast** (blocks are considered final as soon as 2/3 of validators accepted it, which may take < 1 second)
Safety threshold	At least **50% of a scarce resource** (compute power, staked tokens) under the control of honest (correct) peers.	At least $2f + 1$ honest (correct) peers for every f byzantine peers ($N - f = 2f+1$). In other words, >2/3 or **67%** honest.
Energy-efficiency	Very **low** for Proof-of-Work. **High** for Proof-of-Stake.	**High** (similar to standard replicated databases)

Final words about consensus algorithms

- Consensus protocols, like cryptographic protocols, are rife with implementation subtleties. Just like it is not wise to invent your own cryptography protocol, it is usually not wise to invent your own consensus algorithm.
- Before Blockchain (pre-2009) the focus of the academic community was almost exclusively on crash fault-tolerant (CFT) consensus among a *closed* group of processes.
- The PBFT algorithm (1999) was a milestone in achieving byzantine fault-tolerant (BFT) consensus in real-world networks, but also still assumed a *closed* group of processes.
- With the advent of Blockchain (post-2009), the focus has shifted to study BFT consensus in an *open* and *adversarial* environment (there is no a-priori closed group of processes).
- Today the **state-of-the-art** are **DAG-based consensus** algorithms
  - **DAG-Rider**: All You Need Is DAG (PODC 2021). [I. Keidar, E. Kokoris-Kogias, O. Naor, A. Spiegelman]
  - **Narwhal** & **Tusk** (EuroSys 2022) [G. Danezis, E. Kokoris-Kogias, A. Sonnino, A. Spiegelman]
  - **Bullshark** (CCS 2022) [A. Spiegelman, N. Giridharan, A. Sonnino, E. Kokoris-Kogias]

Sui Mysticeti: 100.000 tps at <1sec finality (paper)



(Source: Sui / Mysten Labs, 2024)

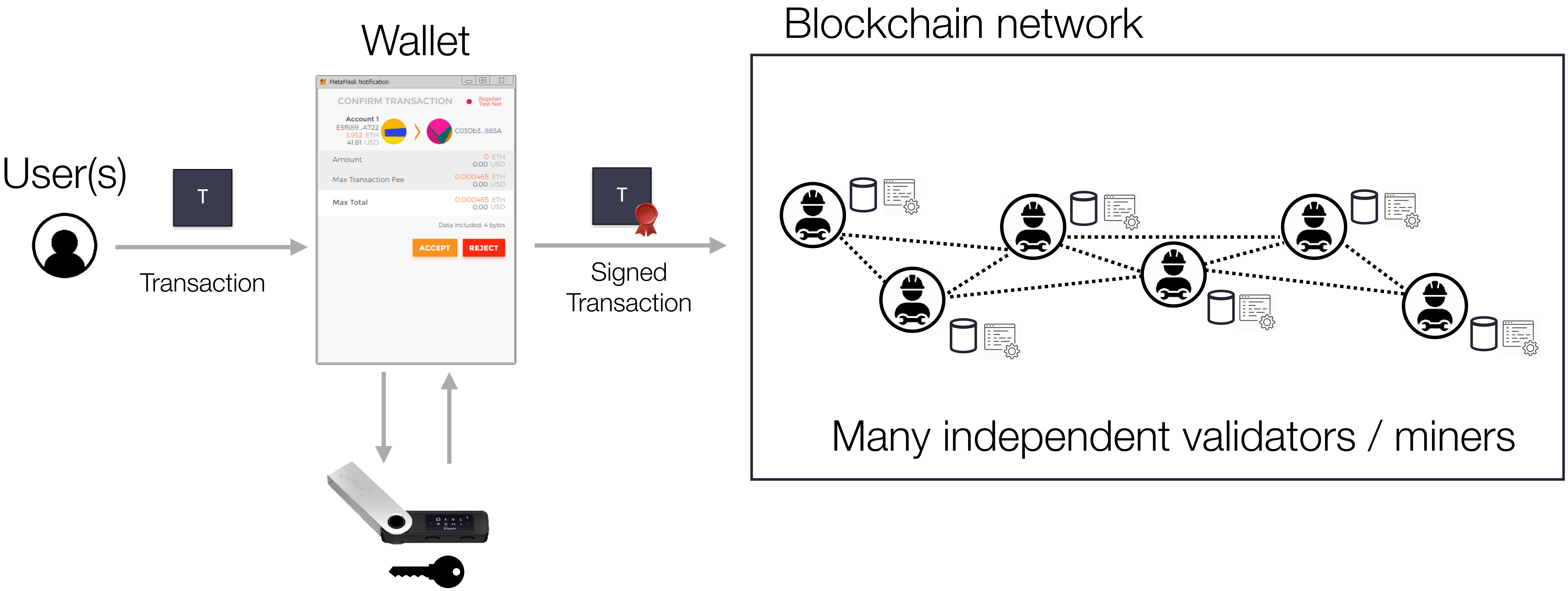
KU LEUVEN DistriNet

Blockchain beyond payment ledgers

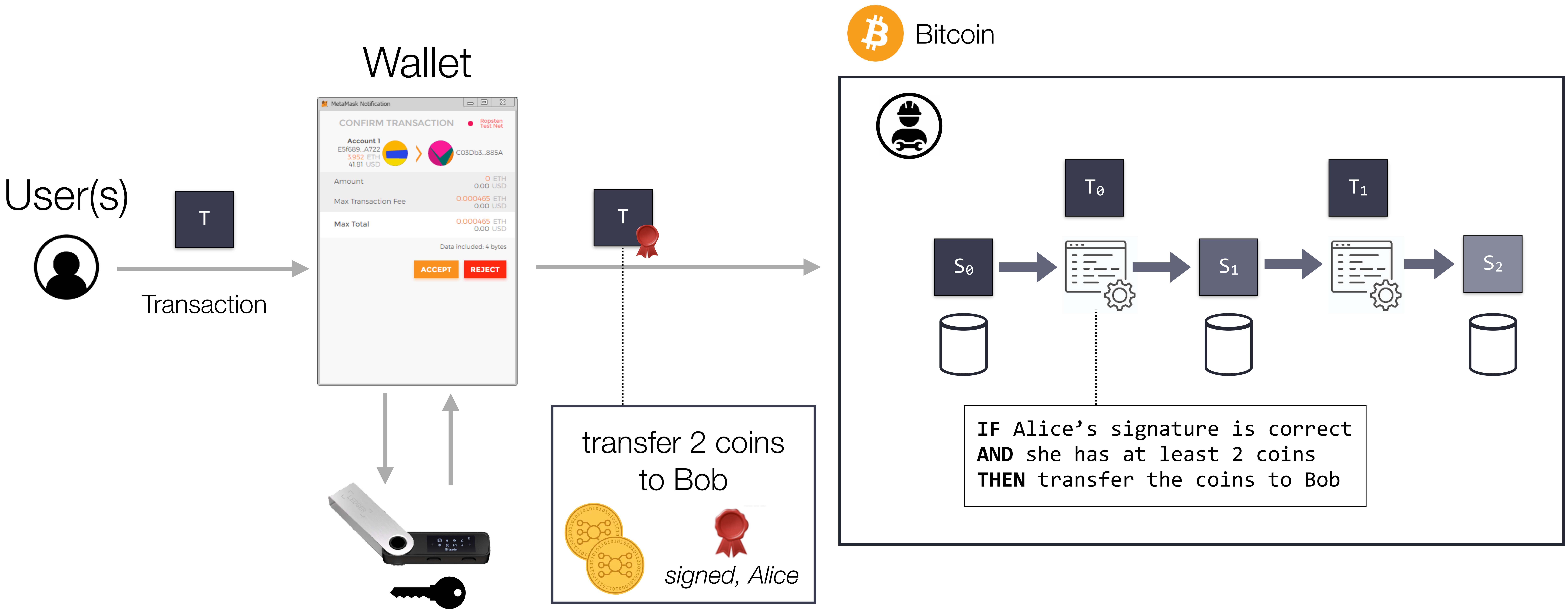
- Recall: a blockchain is an append-only, tamper-evident transaction log.
- Can such logs be used to record data other than payments? Yes!
- **Transparency Logs**: improve accountability by forcing a trusted party to record activities in a tamper-proof, verifiable log.
    - Example: Google's **Certificate Transparency** project for TLS Certificates issued by root CA's. Uses merkle trees (not full blockchains) to enable cryptographic proof of inclusion of new certificates in one or more transparency logs.
- Secure **timestamping / notarisation**: prove existence of a document or data item by storing its hash on-chain.
- Public **randomness beacons**: generate fresh random values that no single party can manipulate or predict (Rabin, 1983)
- Decentralized **PKI**: use a blockchain to associate meta-data with public keys (cfr. W3C Decentralized Identifiers - **DIDs**)
- Decentralized **name registries**: DNS-like name resolution services without "root servers". Example: Ethereum Name Service (ENS) and .eth domain names

Blockchains as trusted computers: **smart contracts** and **Ethereum**

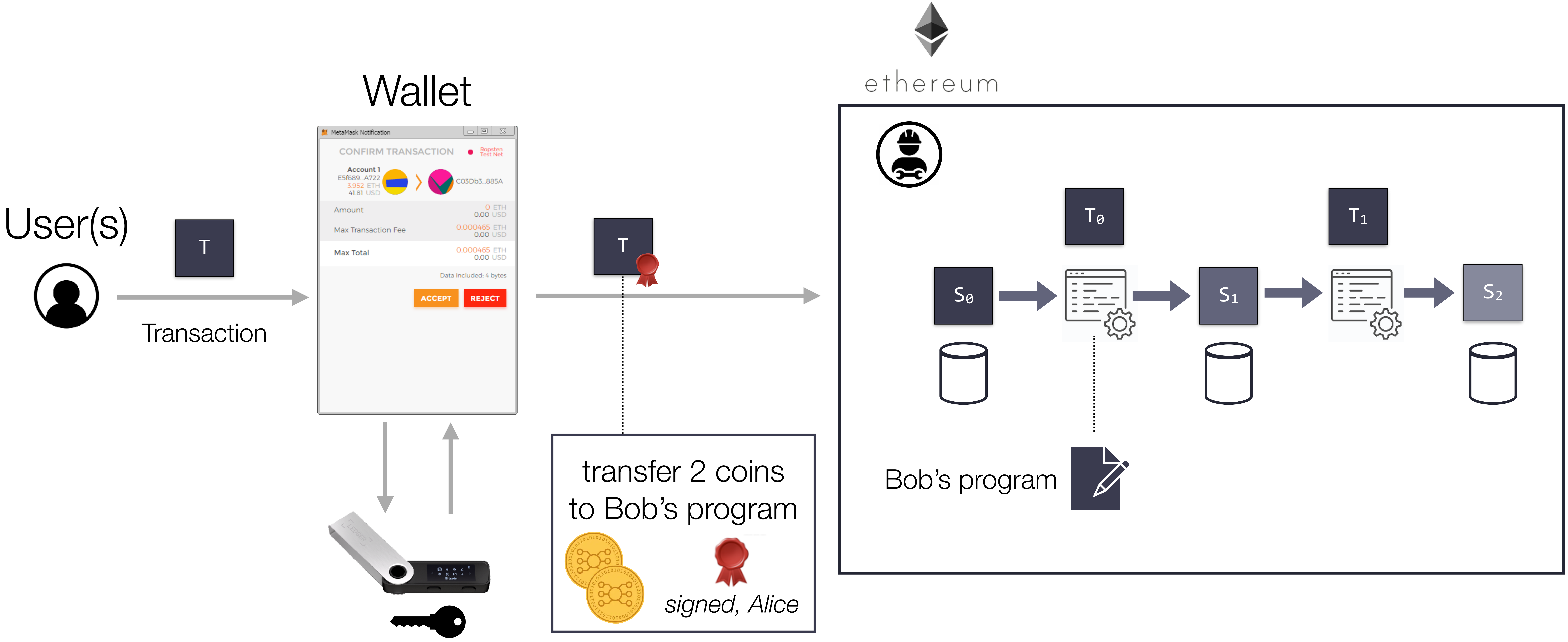
Physical view: a blockchain is a peer-to-peer network of computers



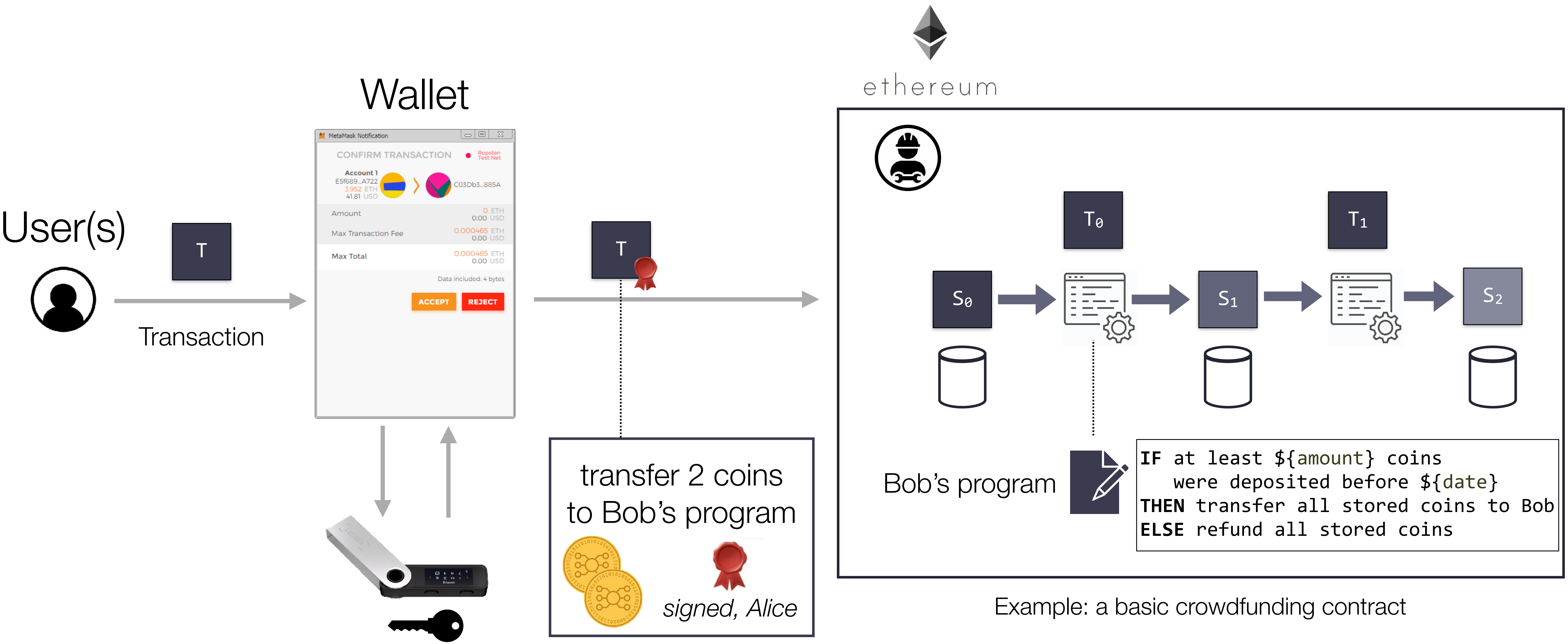
Logical view: a blockchain is a transaction processing machine



Ethereum's innovation: make the transactions programmable!



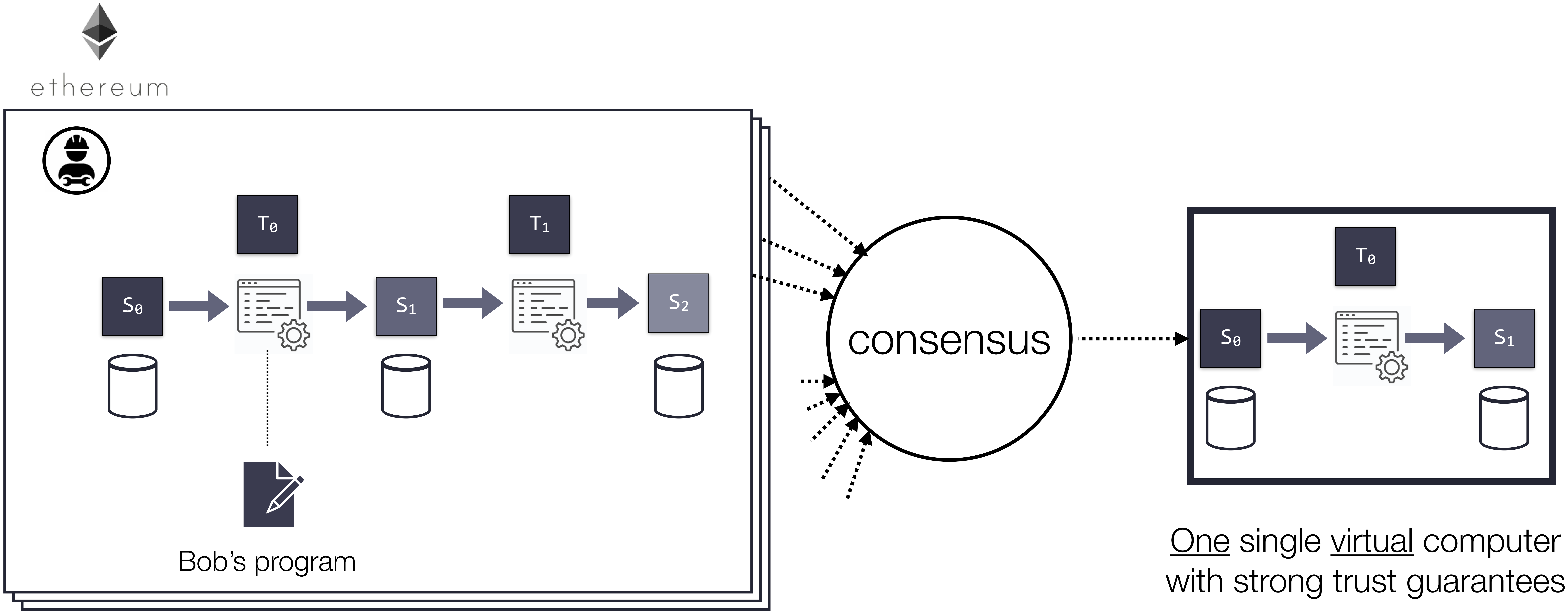
Ethereum's innovation: make the transactions programmable!




```
IF at least ${amount} coins
   were deposited before ${date}
THEN transfer all stored coins to Bob
ELSE refund all stored coins
```

Example: a basic crowdfunding contract

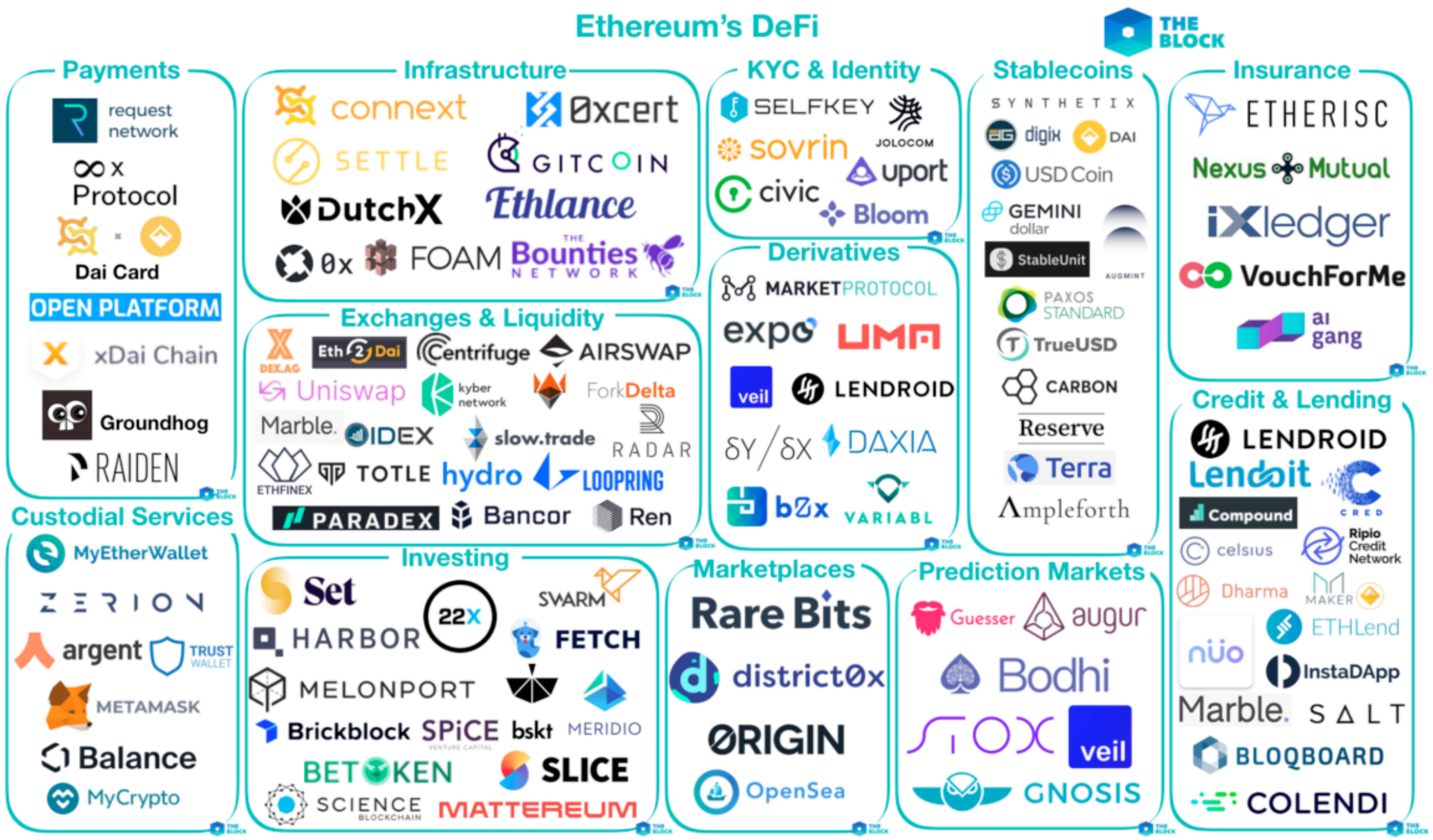
Blockchains as *trusted* virtual computers



One single virtual computer with strong trust guarantees

Many (1000s) untrustworthy physical computers

Applications? Ethereum's "Decentralized Finance"



(image credit: theblockcrypto.com)

Fungible tokens (ERC-20)

Non-fungibles (NFTs)

Stablecoins

SBTs

.eth names

…

New kinds of **electronic rights** collectively worth over **$100 Billion**

(source: coingecko.com, retrieved May 2024)

Ethereum has challenges

- Can be expensive to use (> $10 in transaction fees is not uncommon)
- Slow (~10-14 transactions per second)
- Bugs in contracts can be fatal

"On the layer 1 Ethereum blockchain, high demand leads to slower transactions and nonviable gas prices." [1]

1 https://ethereum.org/en/developers/docs/scaling

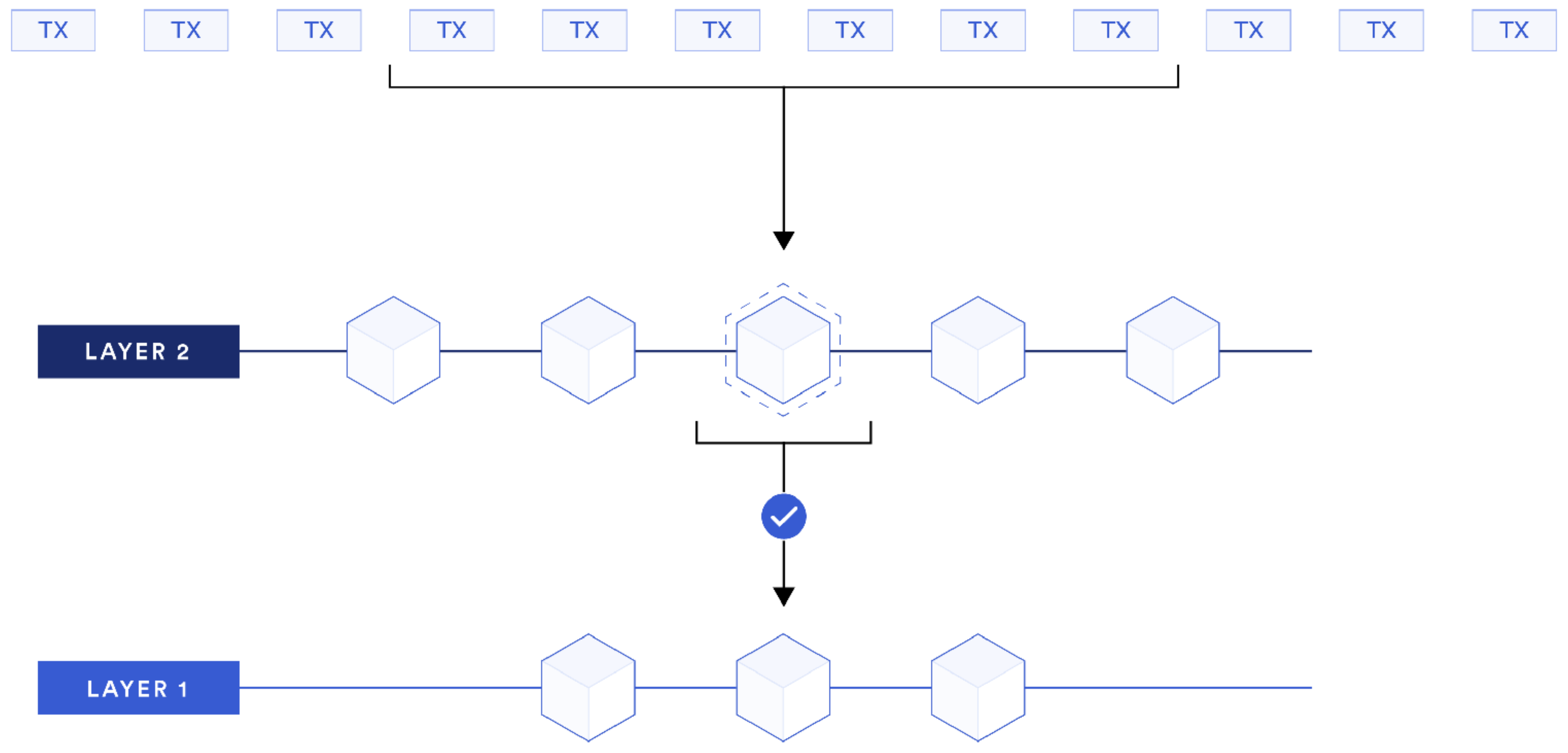
“Layer 2” scaling solutions (a.k.a. “rollups”)

- Key idea: batch many “Layer 2” (L2) transactions into a single combined transaction stored on “Layer 1” (L1)

- Offer a way for anyone to verify that the batch of L2 transactions was correctly executed

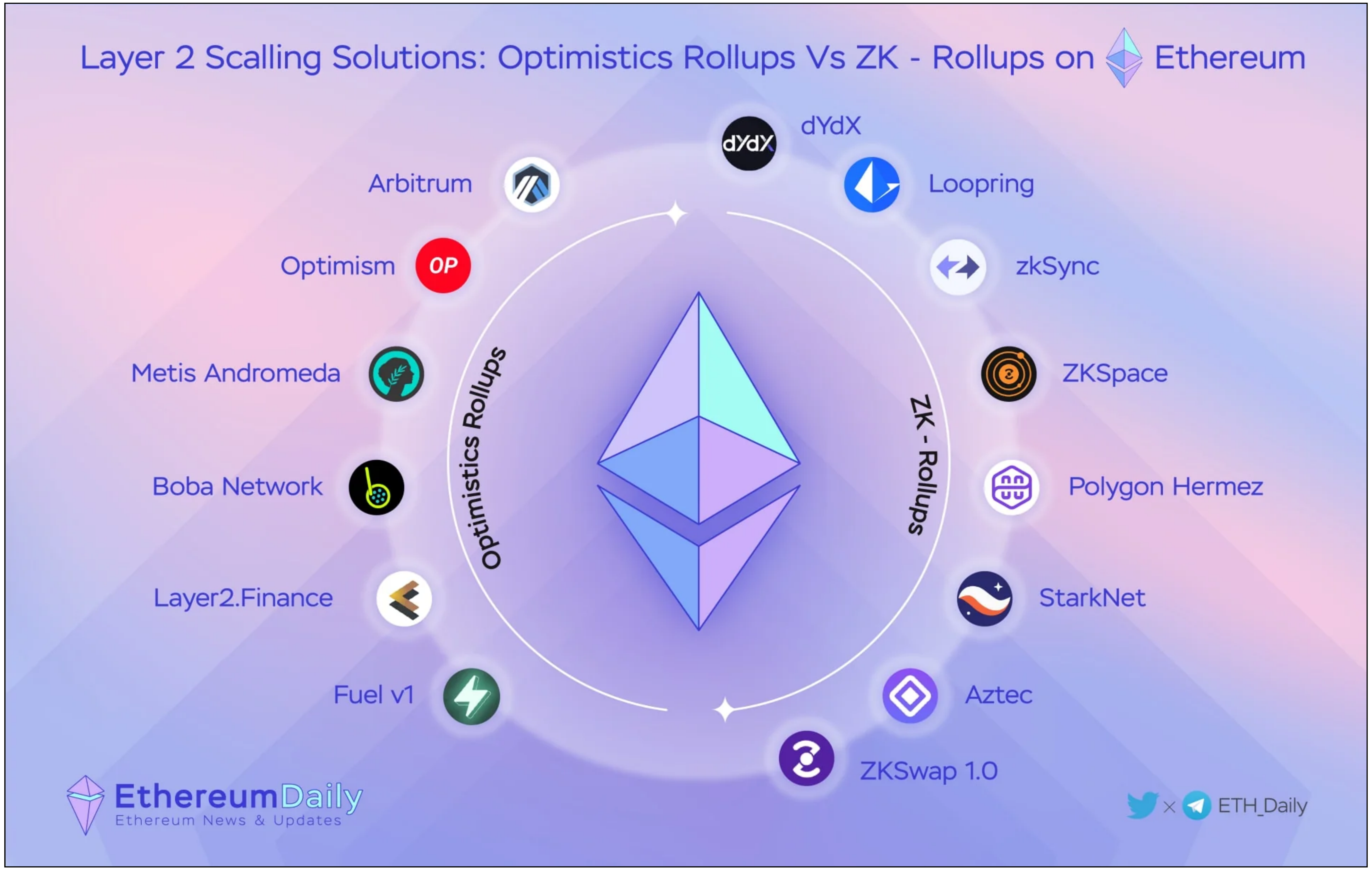
  - “fraud proofs” => optimistic rollups

  - “zero-knowledge proofs” => zk-rollups



(Source: Chainlink)

“Layer 2” scaling solutions: landscape



“Layer 2” scaling solutions: benefits

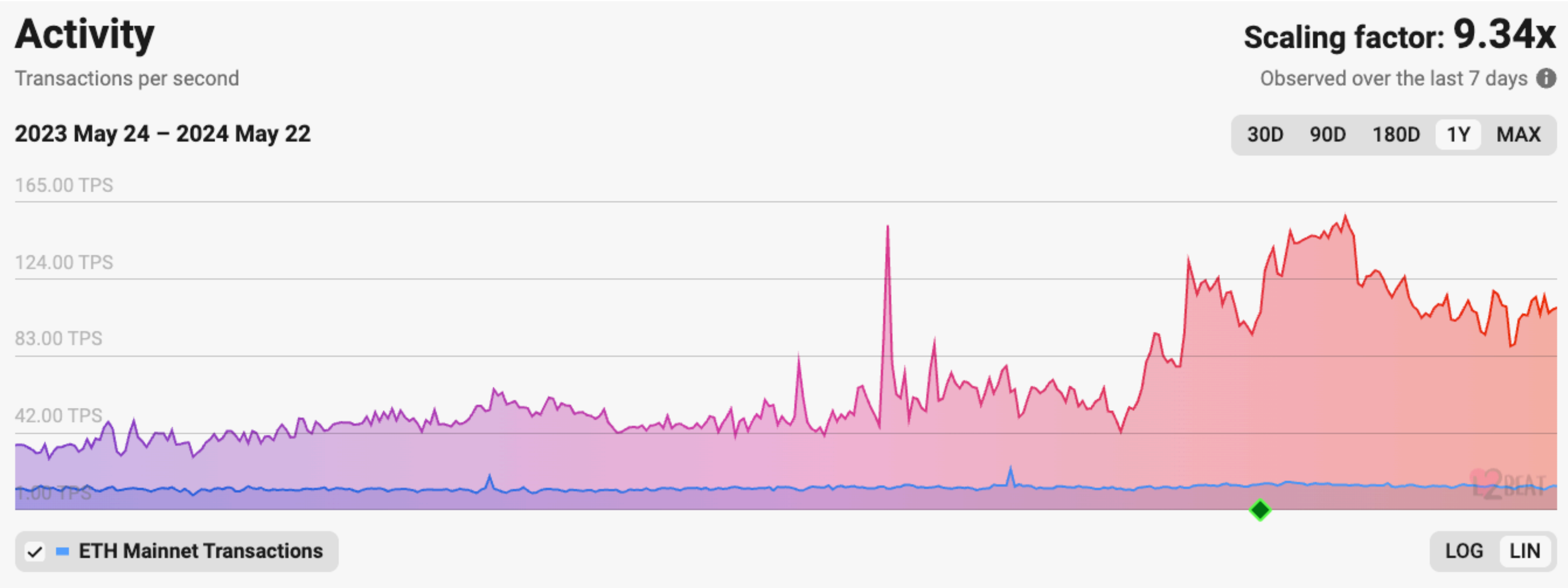
- **Lower** transaction **fees** (< $0.01 / tx)

All L2s | Full Rollups

Name	Send ETH	Swap tokens
StarkNet	< $0.01	< $0.01
Arbitrum One	< $0.01	$0.01
Optimism	< $0.01	$0.02
Polygon zkEVM	$0.02	$0.32
Metis Network ⚠	$0.03	$0.14
Loopring	$0.05	-
zkSync Lite	$0.06	$0.14
DeGate	$0.17	-

(Source: l2fees.info)

- **Higher** transaction **throughput** (100-1000 tps at ~13min finality)



(Source: L2Beat)

Conclusion

Blockchain: hot topics & open challenges

- **Privacy**! By default, all transaction data is public. Can't store secrets on the blockchain. Potential solution: prove possession of secrets using Zero-knowledge proofs (SNARKs and STARKs) (Zcash, Starknet, …)

- **Scalability** & the Blockchain Trilemma ("secure, scalable, decentralized: choose two"). Potential solutions: sharding and application-specific blockchains (Cosmos), L2 rollups (Arbitrum, Optimism), Payment channels (Bitcoin Lightning network)

- **Safe programming** of smart contracts (avoiding bugs): use Rust and WebAssembly instead of Ethereum's Solidity and EVM to write smart contracts (Internet Computer, Polkadot, Cosmwasm, NEAR, …) Also: Move (Sui, Aptos)

- **"Stateless" light clients** for secure blockchain access on Web/Mobile: ultra-compact inclusion proofs (Verkle trees), constant-size blockchain state (Mina protocol), data availability (Celestia network), decentralized RPC (Portal network)

- **Blockchain interoperability**: making assets on one chain available on another chain: inter-blockchain communication protocol (Cosmos IBC), cross-chain "bridges" (Wormhole), …

- **Oracles**: trustworthy, reliable access to off-chain data (Chainlink DONs - decentralized oracle networks)

Recap: course topics

- The **origins** of Blockchain
- What are the **cryptographic building blocks** of a blockchain?
- How does a blockchain process transactions? **Life of a blockchain transaction**.
- **Consensus** in blockchain networks: Proof-of-Work, Proof-of-Stake, BFT Consensus
- **Permissioned** versus **Permissionless** blockchain networks
- Bonus: Blockchains as trusted computers: **smart contracts** and **Ethereum**

Blockchain and Distributed Ledgers

Tom Van Cutsem
DistriNet KU Leuven

Questions?
tom.vancutsem@kuleuven.be

tvcutsem.github.io be.linkedin.com/in/tomvc github.com/tvcutsem x.com/tvcutsem @tvcutsem@techhub.social

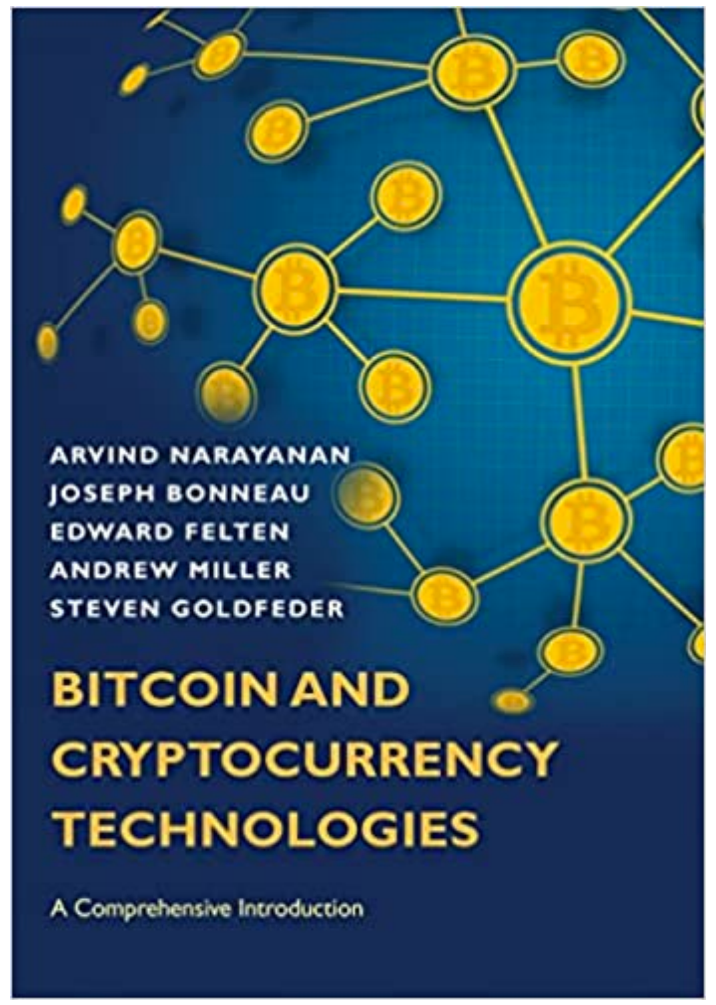
Further reading - good introductory resources on Blockchain

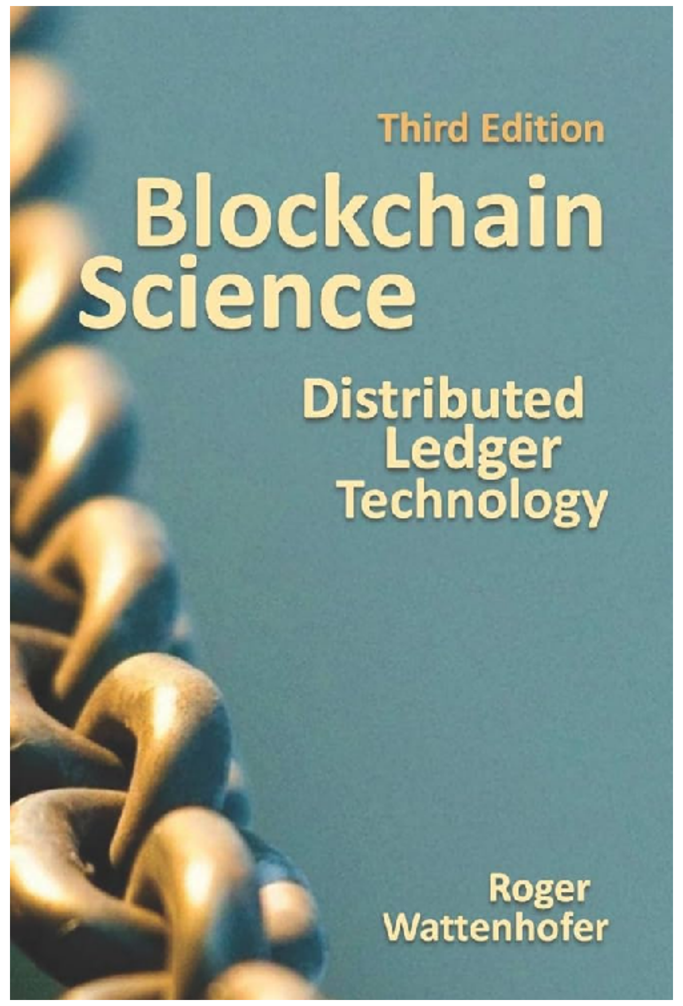
- Narayanan *et al.* “Bitcoin and Cryptocurrency Technologies” Princeton University Press, 2016 - preprint available for free online at: https://bitcoinbook.cs.princeton.edu/

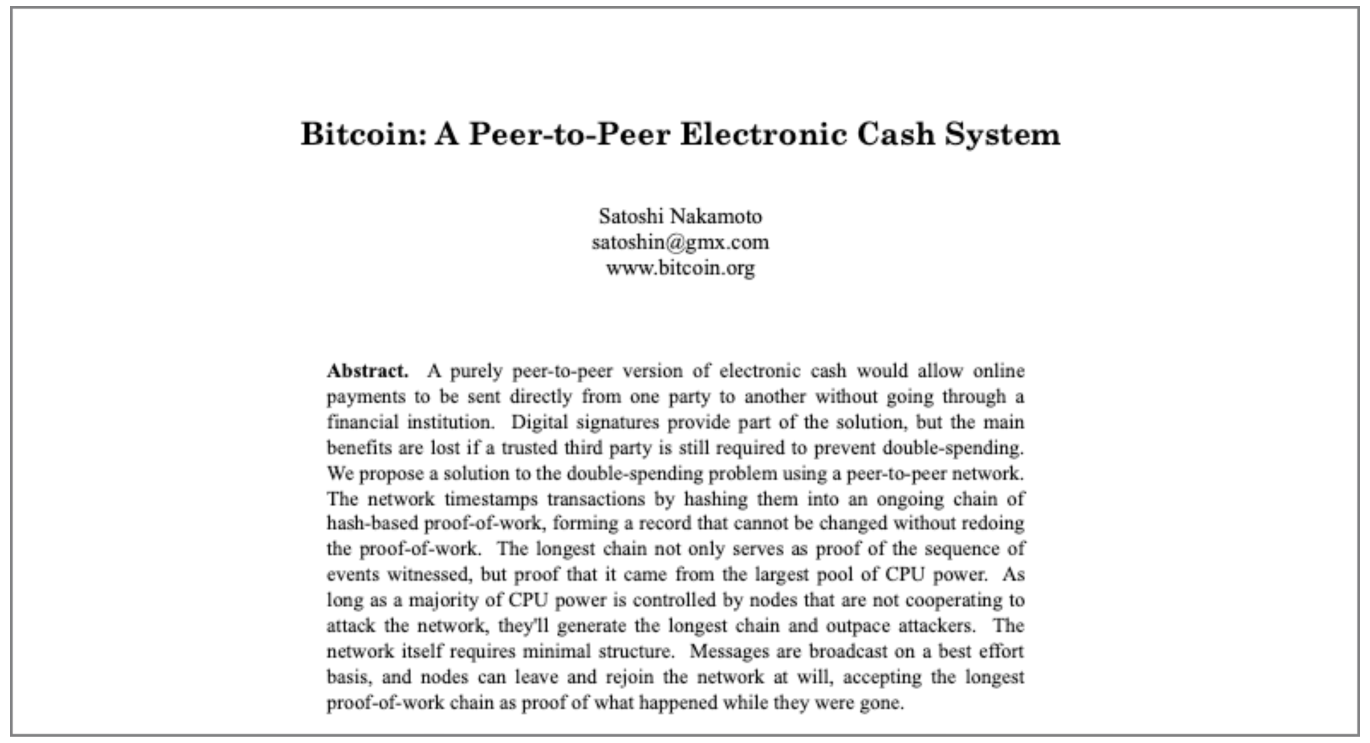
- Roger Wattenhofer (ETH Zurich), “Blockchain Science”, 2019

- Satoshi Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System” a.k.a. the “Bitcoin whitepaper” (2008)

    - Recommended: an annotated online version with helpful notes & clarifications (D. Hogg, 2021): https://blog.infocruncher.com/2021/10/31/bitcoin-whitepaper-annotated/







Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

Abstract. A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.