Advanced Web Security

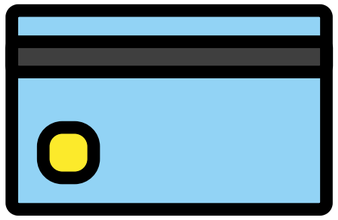
Secure Electronic Payments

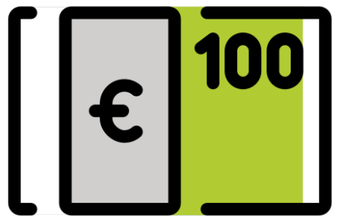
BLOCKCHAIN

Paul Stankovski Wagner - EITN41 LP2 2022 - introduction
Slides by Elena Pagnin

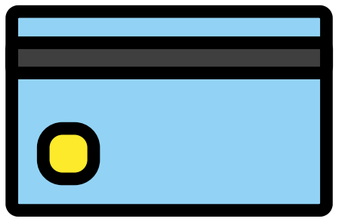
Quick History of (Digital) Money

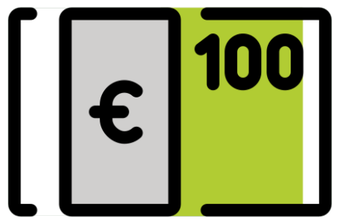
Quick History of (Digital) Money



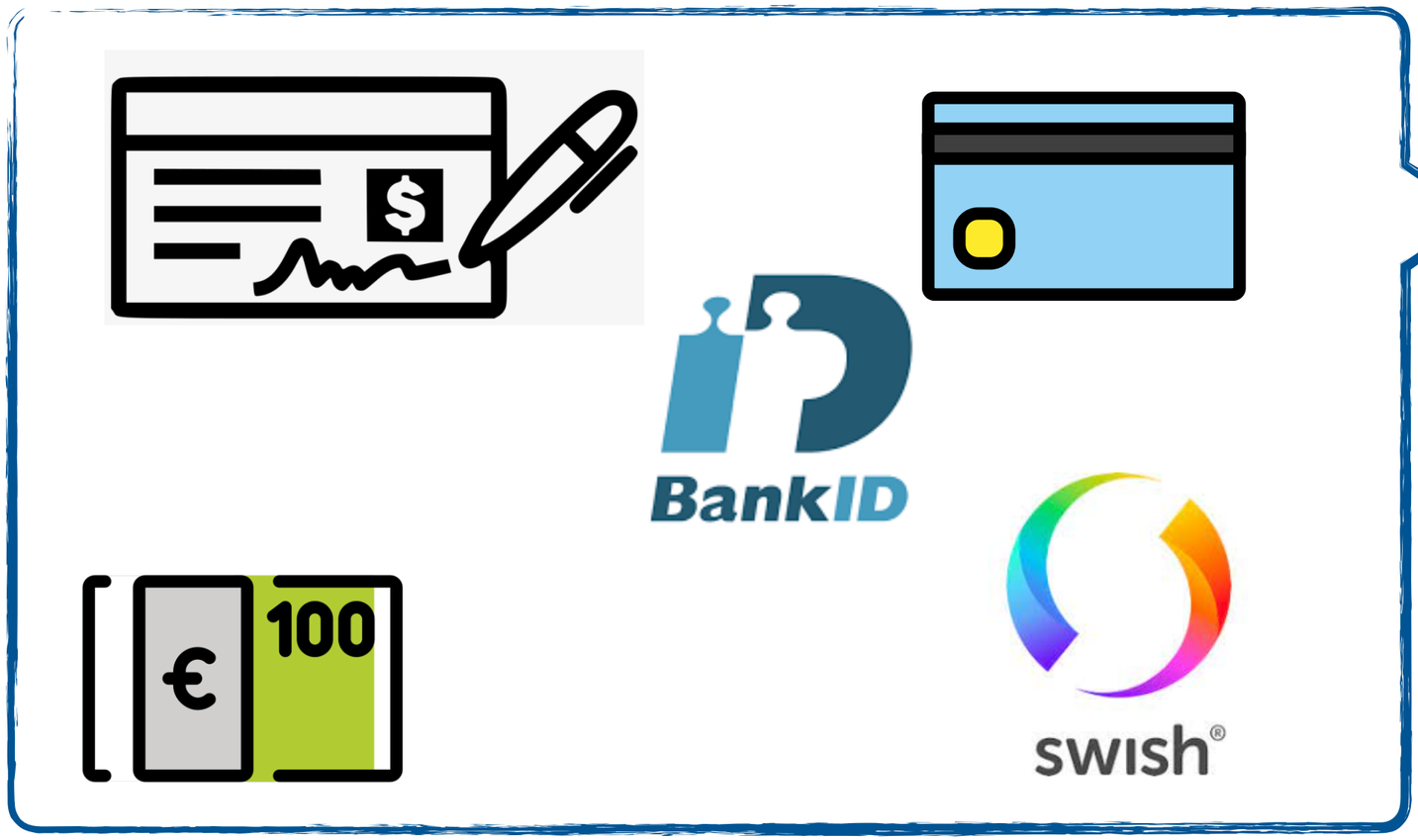


Quick History of (Digital) Money



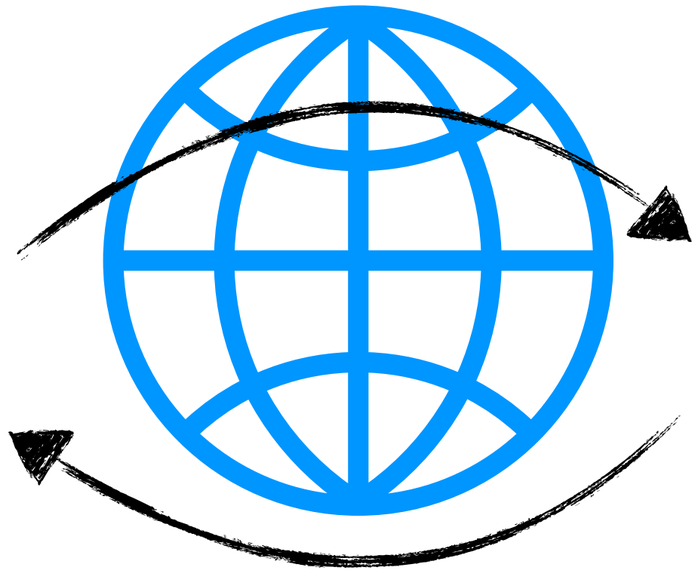


Quick History of (Digital) Money



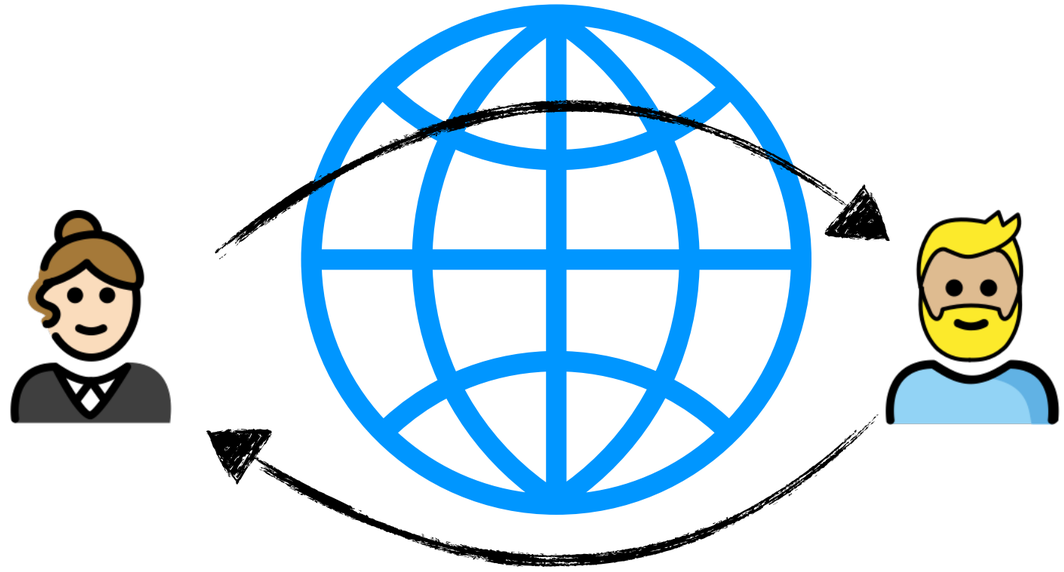
More details on card-based transactions in Lecture 2.

Why Cryptocurrencies?



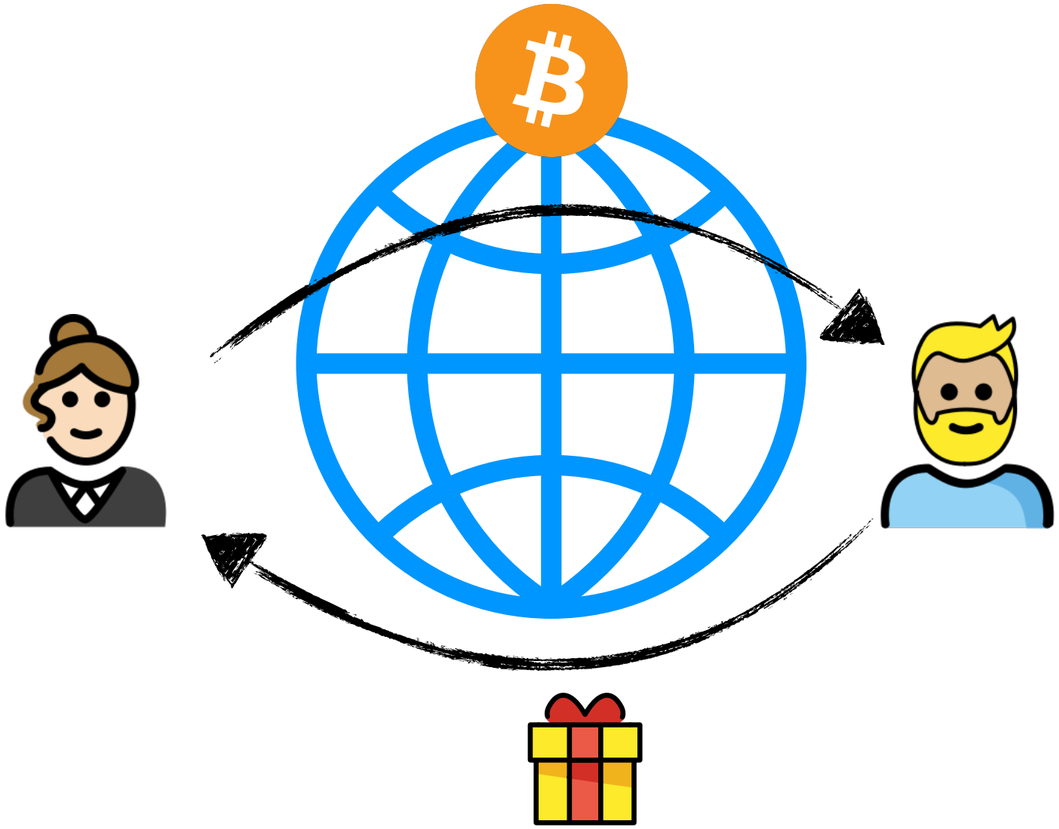
Bitcoin: A Peer-to-Peer Electronic Cash System by Satoshi Nakamoto (2008)

Why Cryptocurrencies?



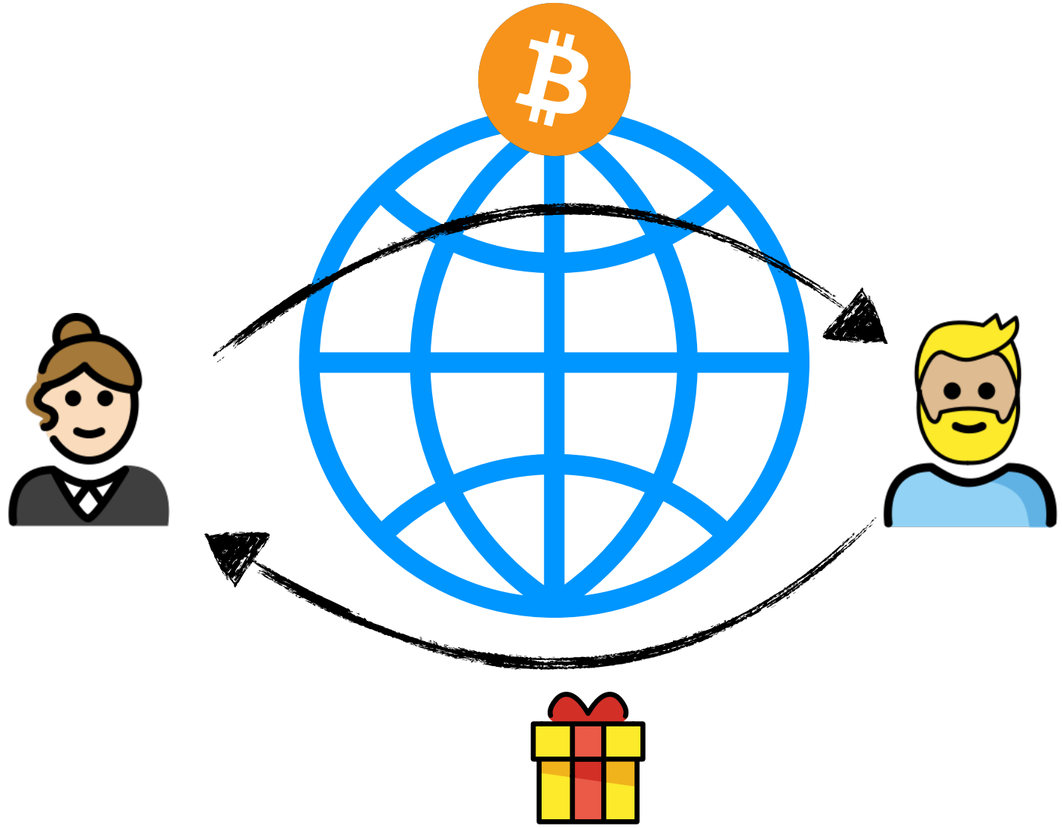
Bitcoin: A Peer-to-Peer Electronic Cash System by Satoshi Nakamoto (2008)

Why Cryptocurrencies?



Bitcoin: A Peer-to-Peer Electronic Cash System by Satoshi Nakamoto (2008)

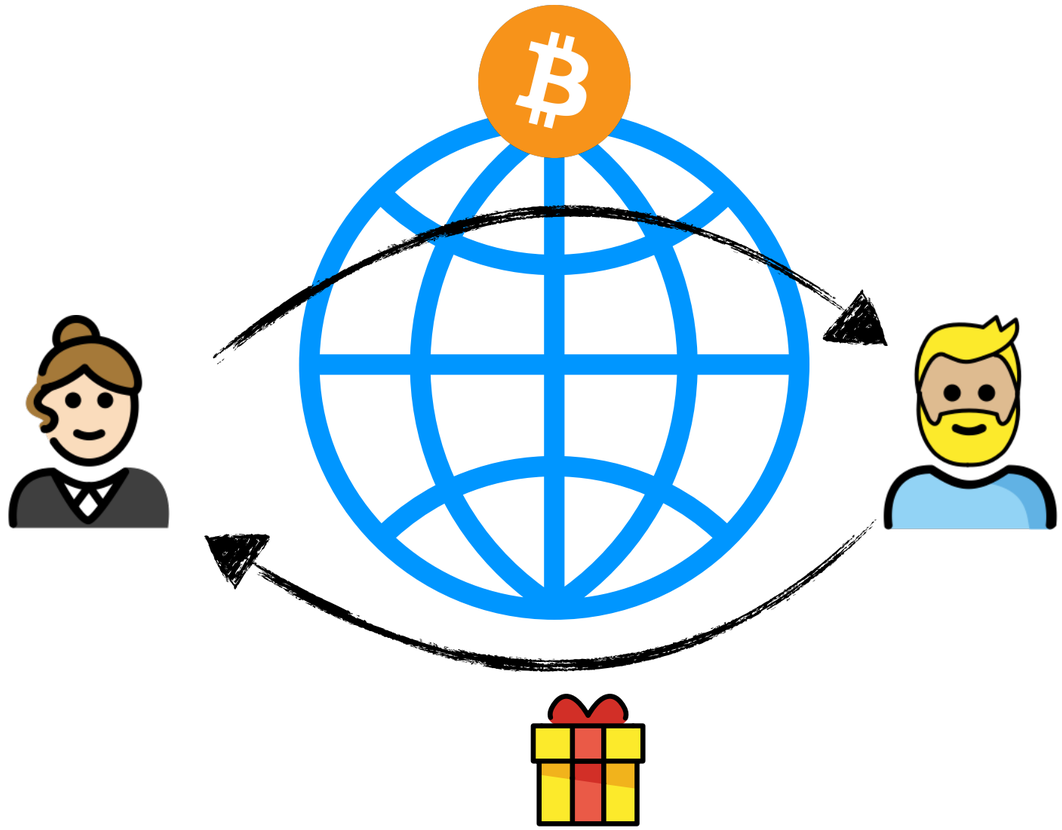
Why Cryptocurrencies?



Bitcoin: A Peer-to-Peer Electronic Cash System by Satoshi Nakamoto (2008)

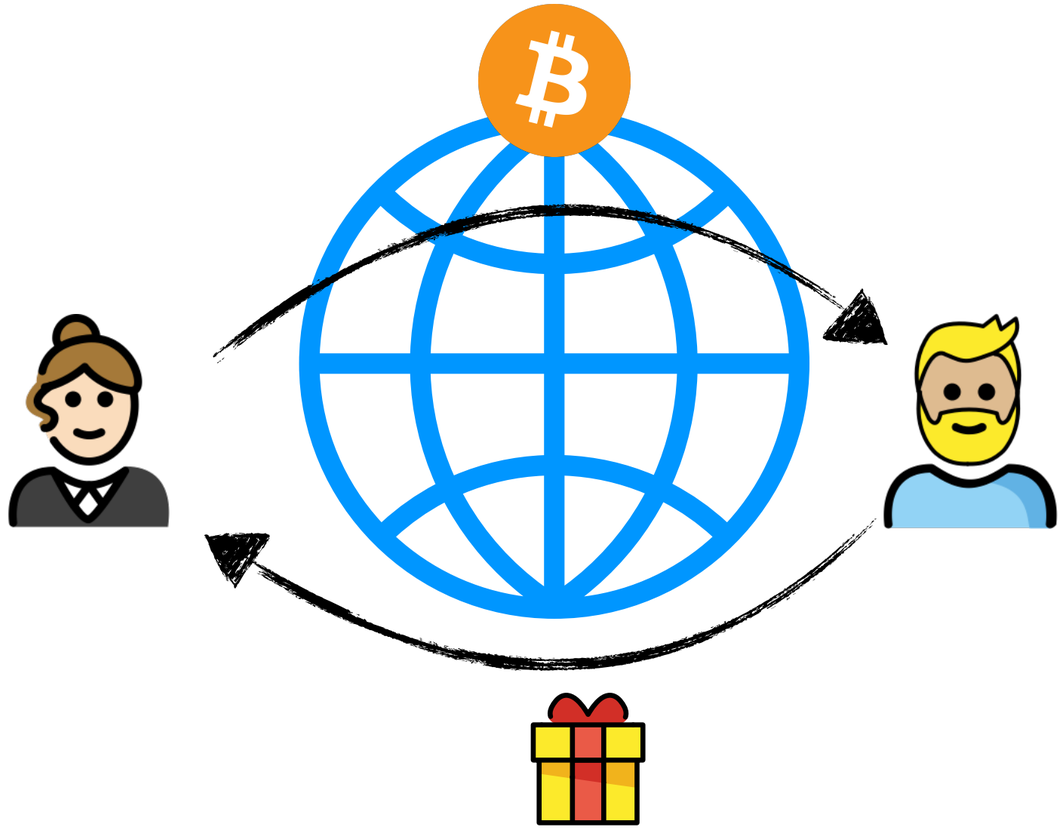
Why Cryptocurrencies?

Problems of earlier eCash systems



Bitcoin: A Peer-to-Peer Electronic Cash System by Satoshi Nakamoto (2008)

Why Cryptocurrencies?

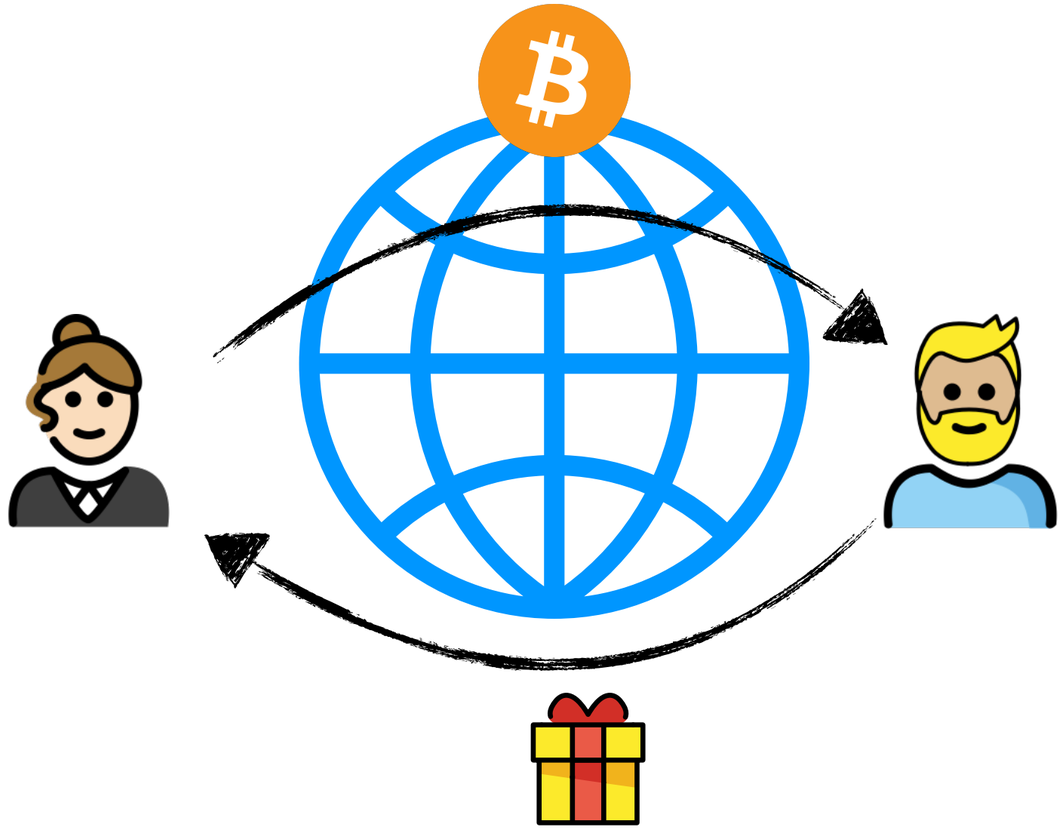


Problems of earlier eCash systems

– Need trusted center (bank, fonds)

Bitcoin: A Peer-to-Peer Electronic Cash System by Satoshi Nakamoto (2008)

Why Cryptocurrencies?



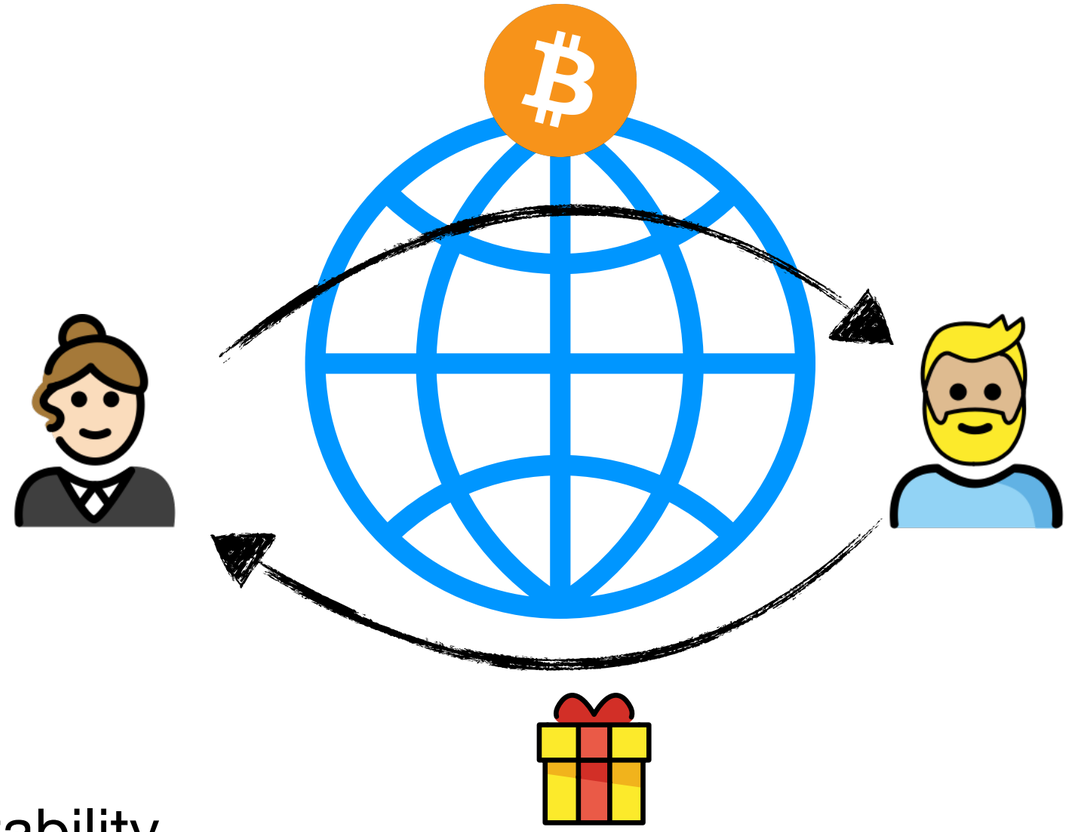
Problems of earlier eCash systems

– Need trusted center (bank, fonds)

– High transaction fees

Bitcoin: A Peer-to-Peer Electronic Cash System by Satoshi Nakamoto (2008)

Why Cryptocurrencies?



Problems of earlier eCash systems

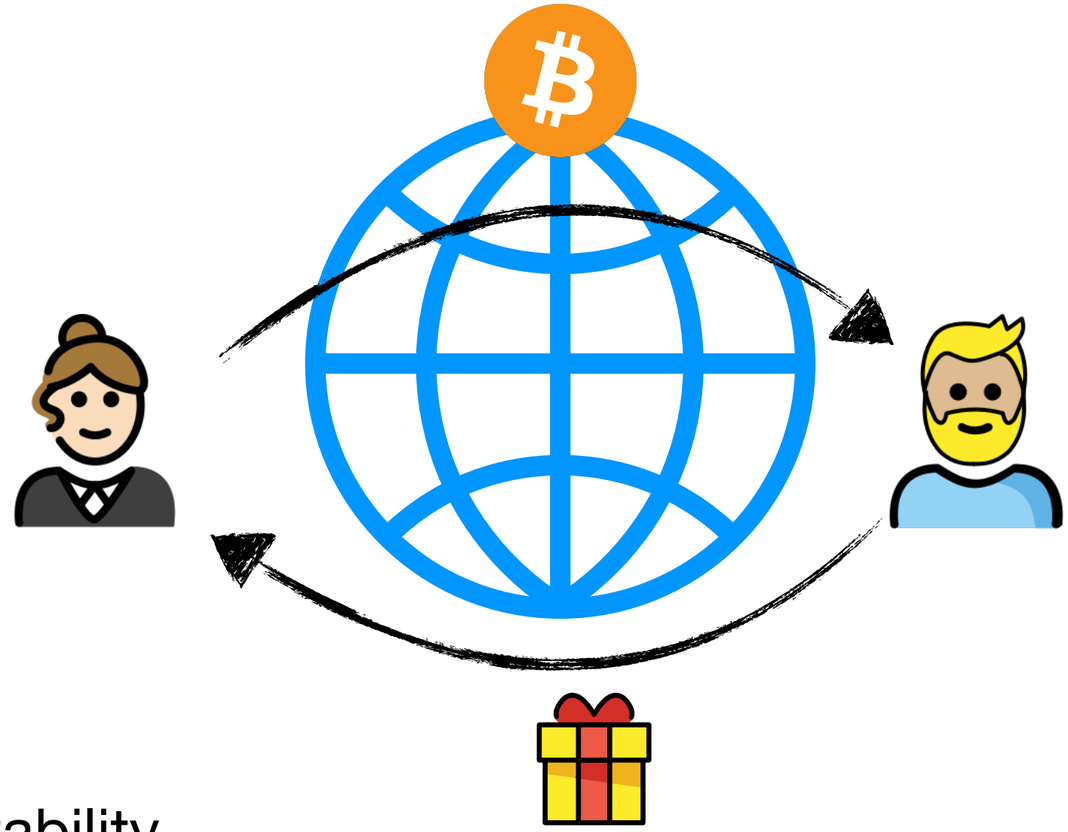
– Need trusted center (bank, fonds)

– High transaction fees

– Hard to achieve anonymity & accountability

Bitcoin: A Peer-to-Peer Electronic Cash System by Satoshi Nakamoto (2008)

Why Cryptocurrencies?



Problems of earlier eCash systems

– Need trusted center (bank, fonds)

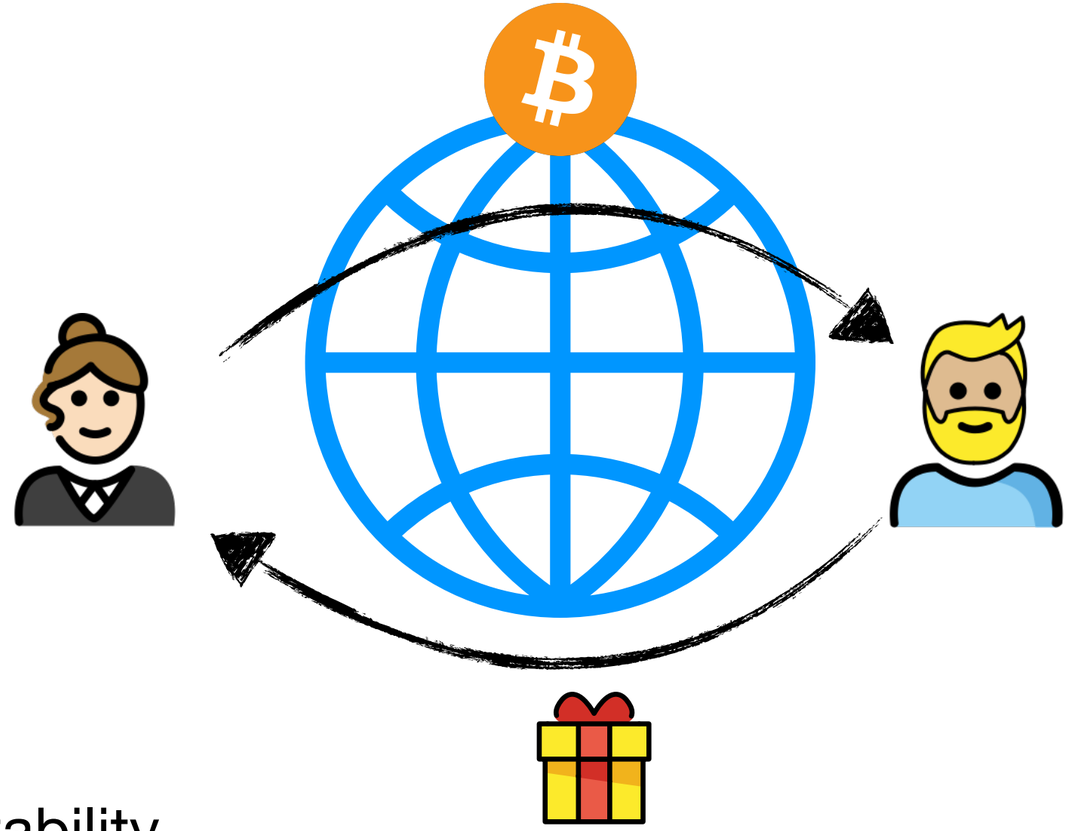
– High transaction fees

– Hard to achieve anonymity & accountability

Solutions in Bitcoin (and similar) ecosystem:

Bitcoin: A Peer-to-Peer Electronic Cash System by Satoshi Nakamoto (2008)

Why Cryptocurrencies?



Problems of earlier eCash systems

– Need trusted center (bank, fonds)

– High transaction fees

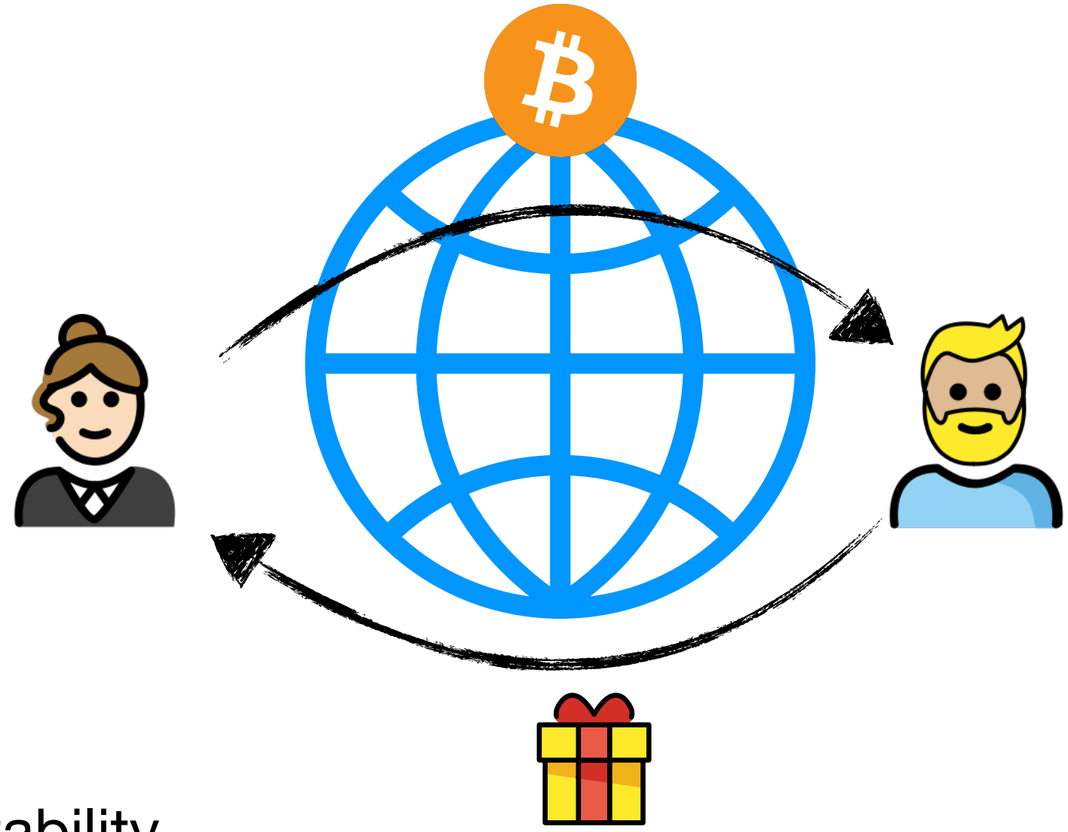
– Hard to achieve anonymity & accountability

Solutions in Bitcoin (and similar) ecosystem:

\+ Decentralized system (mining pools, remote wallets)

Bitcoin: A Peer-to-Peer Electronic Cash System by Satoshi Nakamoto (2008)

Why Cryptocurrencies?



Problems of earlier eCash systems

– Need trusted center (bank, fonds)

– High transaction fees

– Hard to achieve anonymity & accountability

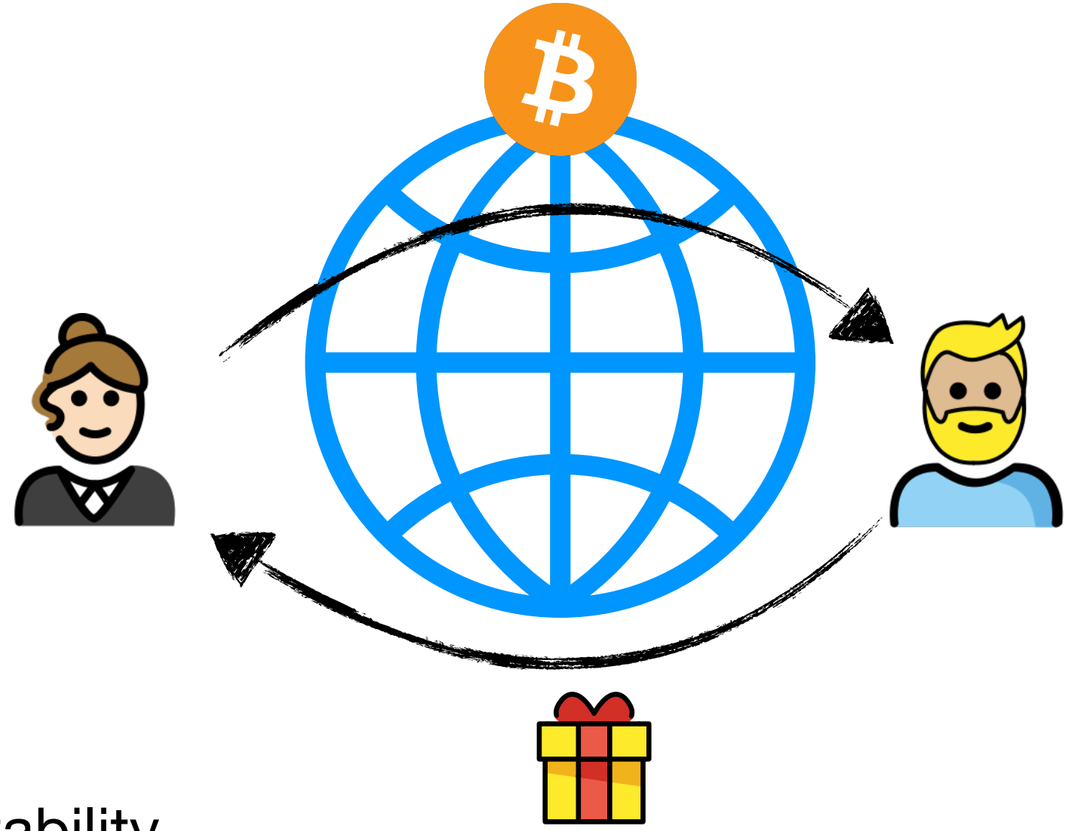
Solutions in Bitcoin (and similar) ecosystem:

+ Decentralized system (mining pools, remote wallets)

+ Variable transaction fees

Bitcoin: A Peer-to-Peer Electronic Cash System by Satoshi Nakamoto (2008)

Why Cryptocurrencies?



Problems of earlier eCash systems

– Need trusted center (bank, fonds)

– High transaction fees

– Hard to achieve anonymity & accountability

Solutions in Bitcoin (and similar) ecosystem:

+ Decentralized system (mining pools, remote wallets)

+ Variable transaction fees

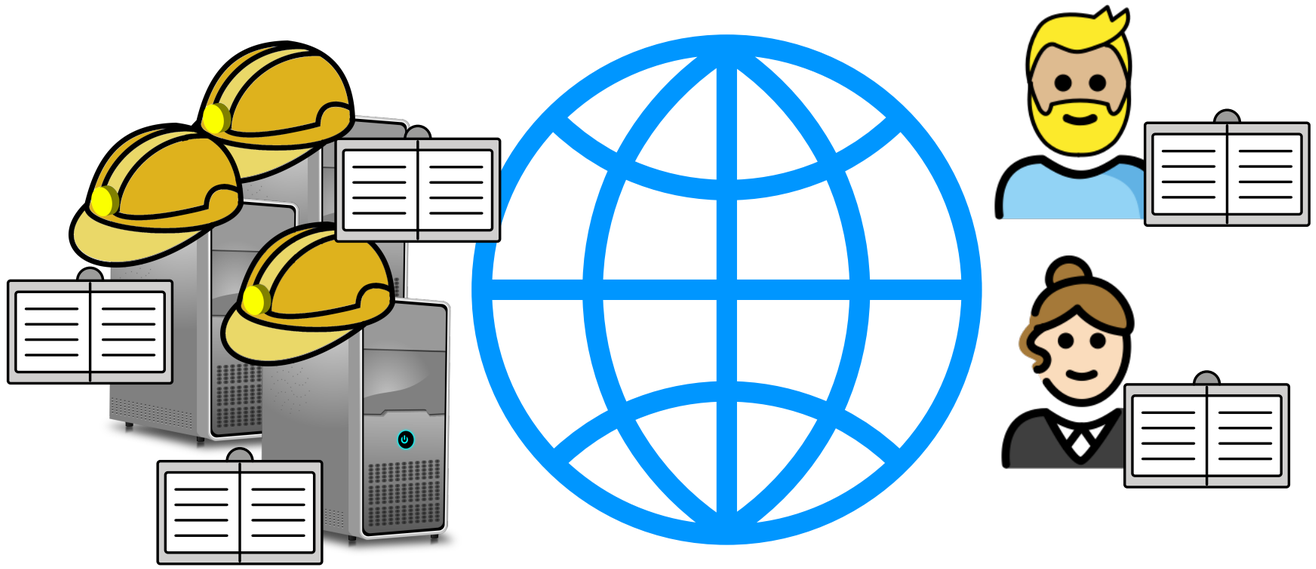
+ (Pseudo) Anonymity & no double-spending

Bitcoin: A Peer-to-Peer Electronic Cash System by Satoshi Nakamoto (2008)

Basics of Cryptocurrencies

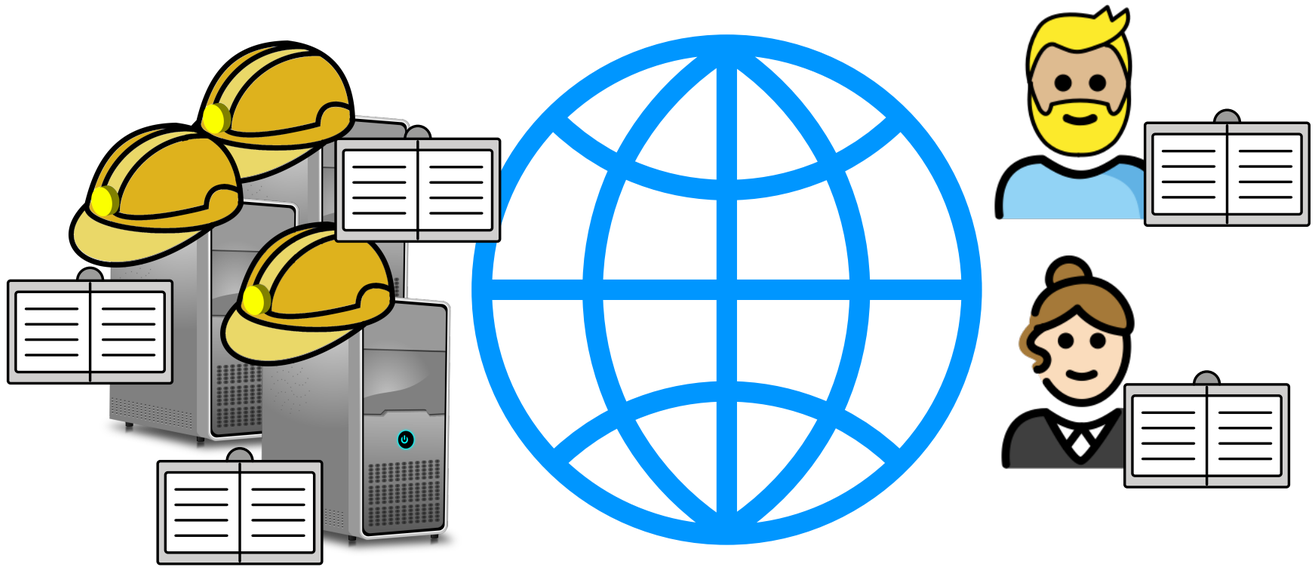
Basics of Cryptocurrencies

replace the bank with
miners and
a **bulletin board**



Basics of Cryptocurrencies

replace the bank with
miners and
a **bulletin board**



Main challenges

1) How to create a digital bulletin board (distributed ledger)?
2) How to agree on ONE ledger view (finality layer)?
3) How to create money?
4) How to prevent double-spending?

Bitcoin.org hack nets giveaway scammers $17,000 overnight

Adam Bannister 24 September 2021 at 13:28 UTC
Updated: 05 October 2021 at 08:36 UTC

Social Engineering | Cryptocurrency | Cybercrime

Open source project back online after fraudsters dangled double-your-money lure

Cryptocurrency funds removed from 6,000 Coinbase accounts due to flaw in SMS authentication

Jessica Haworth 04 October 2021 at 14:09 UTC
Updated: 06 October 2021 at 13:21 UTC

2FA | Cryptocurrency | Cybercrime

Lecture Agenda

How To Set Up a Bulletin Board

- Hash Functions
- Proof of Work (Cryptographic Puzzles)
- Example: Bitcoin

How To Generate Money

- Mining Blocks
- Merkle Trees
- Block Reward System

How To Circulate Money

- Transactions
- ECDSA

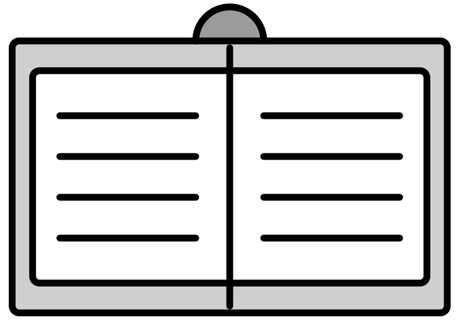
Some Problems With Blockchains

- Freshness of the Genesis Block
- 51% Attack
- Proof of Work VS Proof of Stake

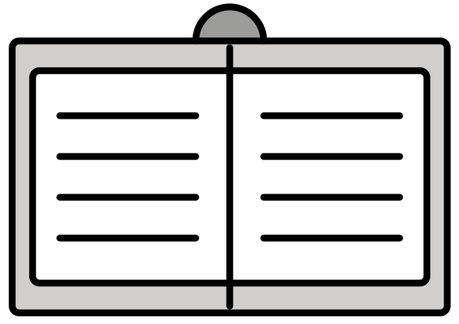
Advanced Tools

- Proofs of Knowledge
- zkSNARKs
- MPC

How To Set Up a Bulletin Board



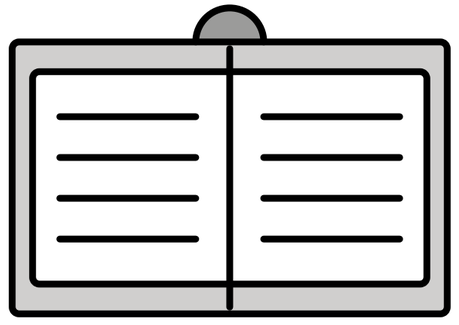
How To Set Up a Bulletin Board



Main property we want to implement = record keeping

1) the past is immutable
2) everyone agrees on the history

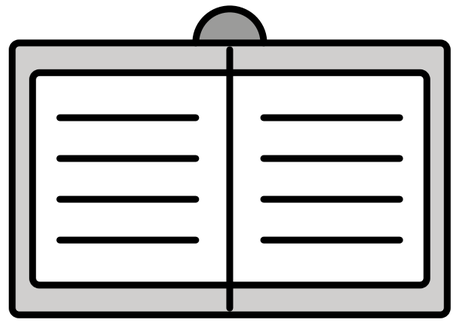
How To Set Up a Bulletin Board



Main property we want to implement = record keeping

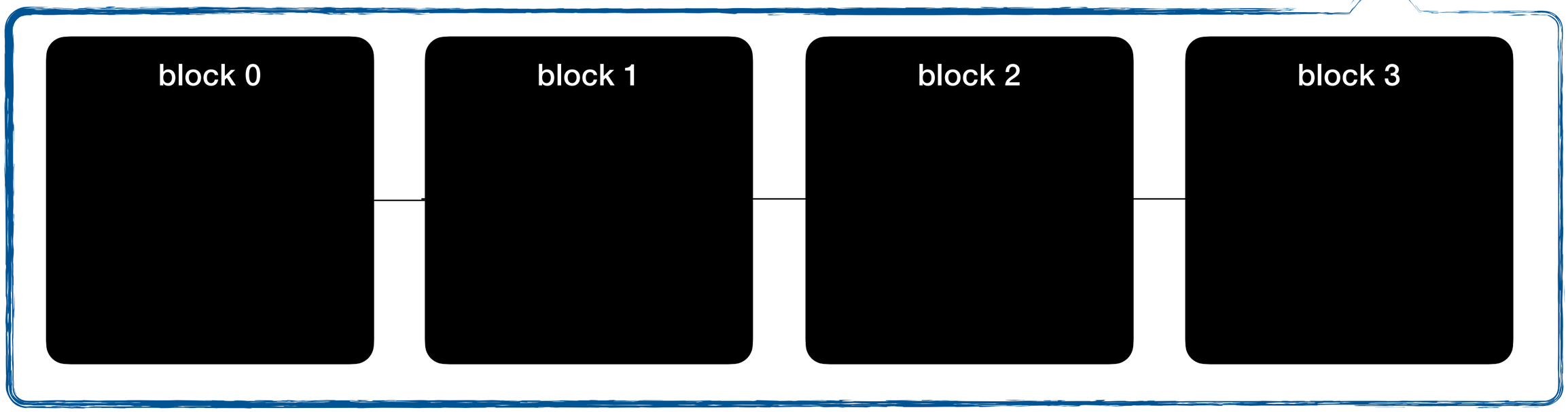
1) the past is immutable
2) everyone agrees on the history

How To Set Up a Bulletin Board



Main property we want to implement = record keeping

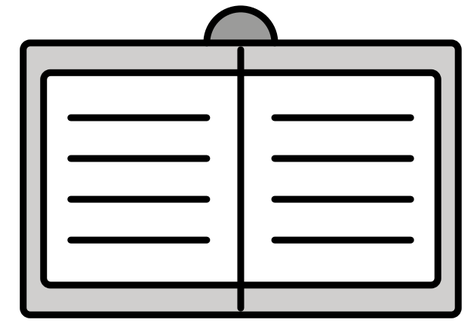
1) the past is immutable
2) everyone agrees on the history

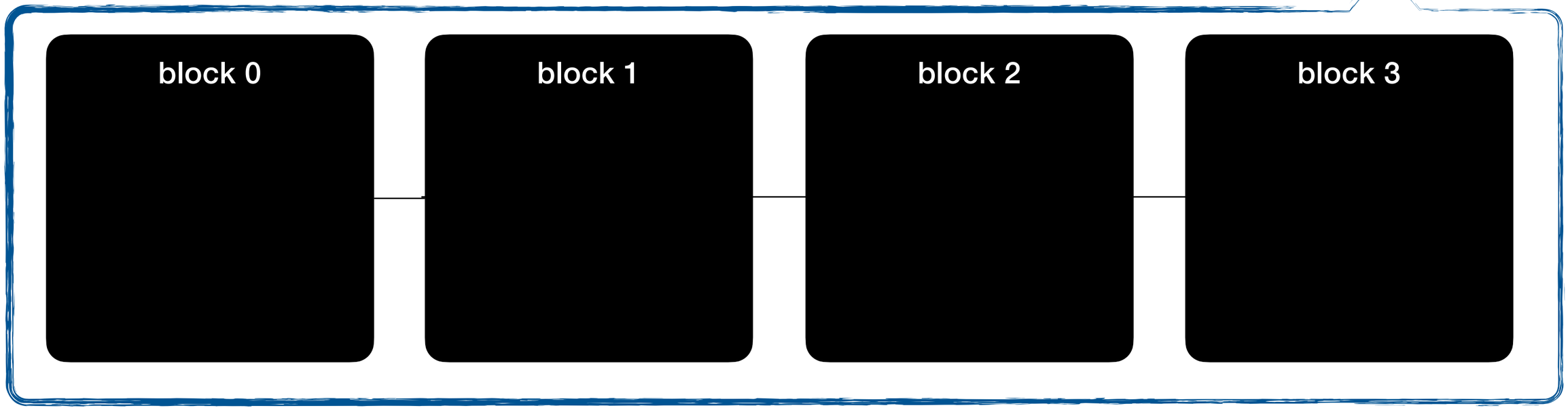


How To Set Up a Bulletin Board

Main property we want to implement = record keeping

1) the past is immutable
2) everyone agrees on the history



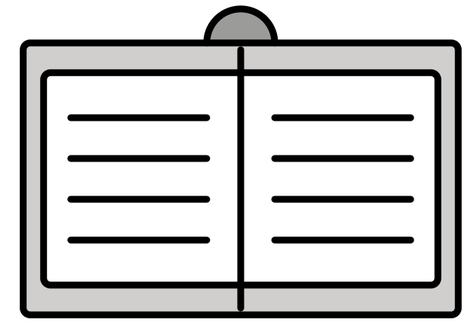


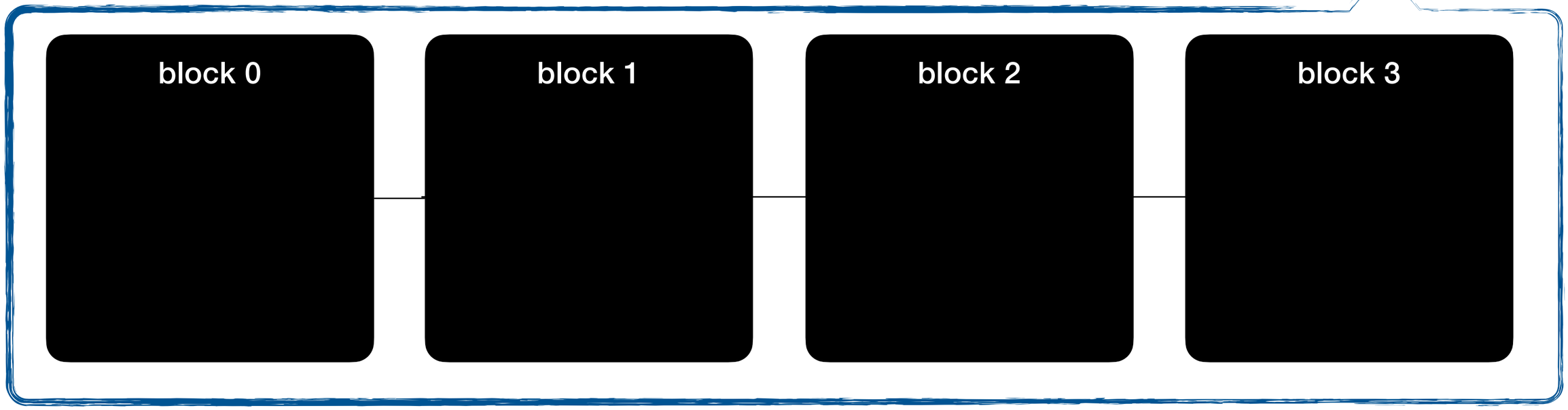
- Partition time into époques / periods / time windows

How To Set Up a Bulletin Board

Main property we want to implement = record keeping

1) the past is immutable
2) everyone agrees on the history



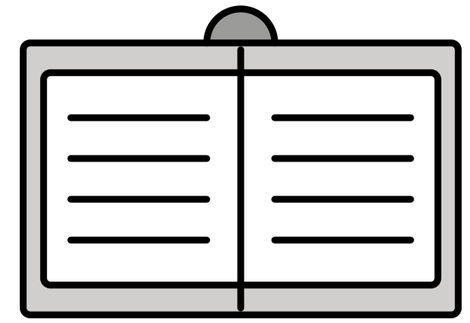


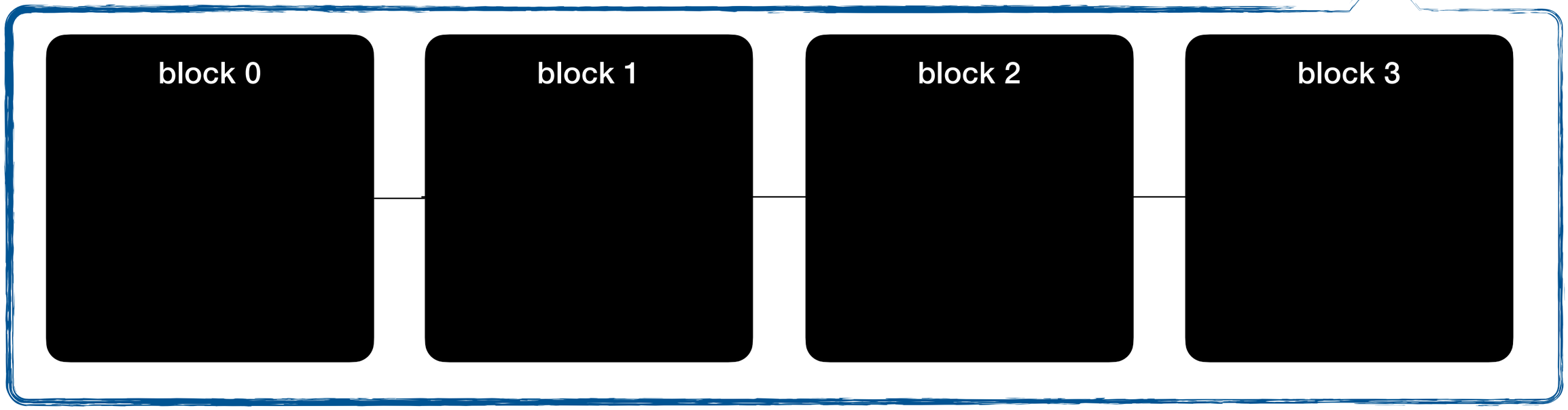
- Partition time into époques / periods / time windows
- Anything that happens in one time period is recorded into a block

How To Set Up a Bulletin Board

Main property we want to implement = record keeping

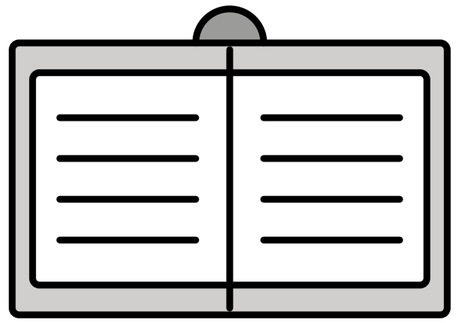
1) the past is immutable
2) everyone agrees on the history





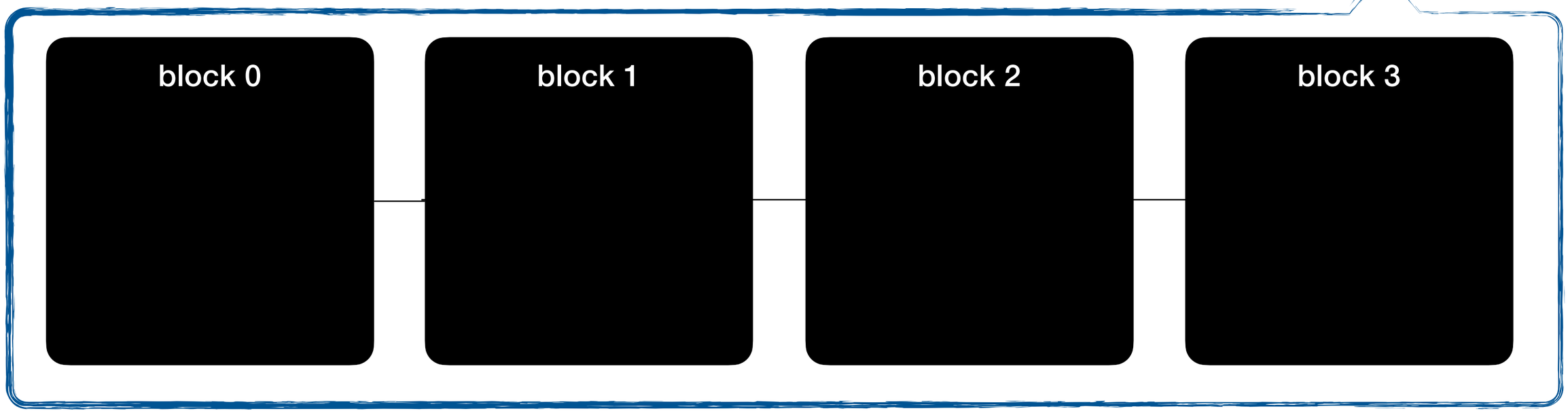
- Partition time into époques / periods / time windows
- Anything that happens in one time period is recorded into a block
- Any change to an ‘old’ block affects all following blocks

How To Set Up a Bulletin Board



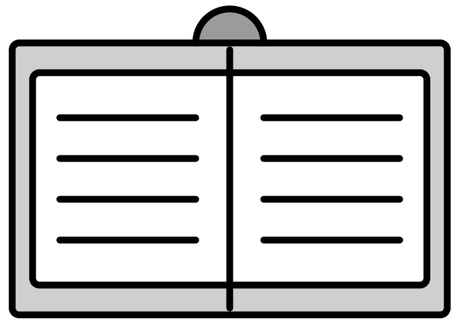
Main property we want to implement = record keeping

1) the past is immutable
2) everyone agrees on the history



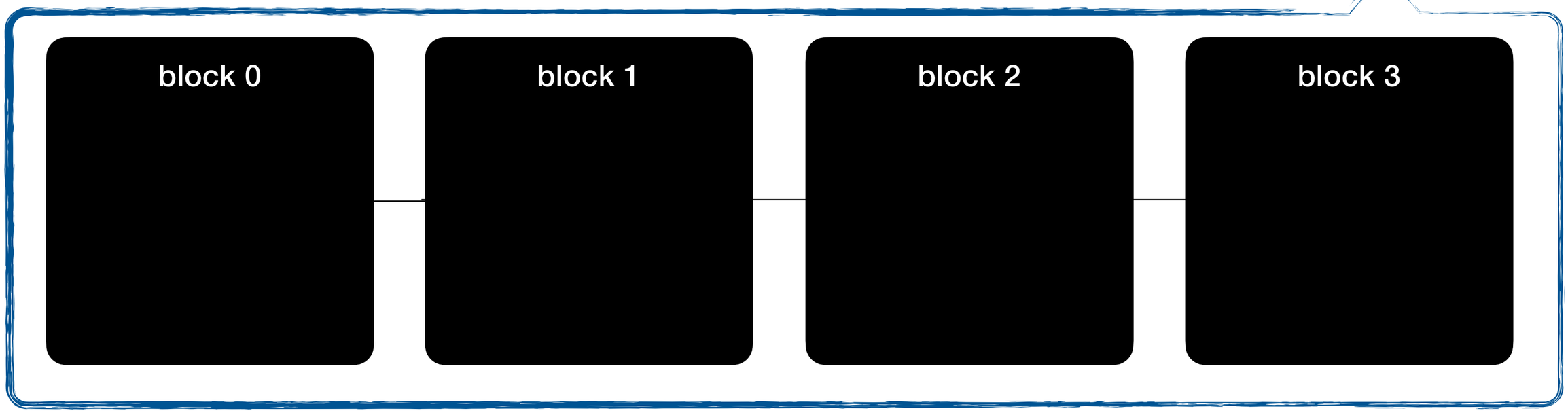
- Partition time into époques / periods / time windows
- Anything that happens in one time period is recorded into a block
- Any change to an ‘old’ block affects all following blocks

How To Set Up a Bulletin Board



Main property we want to implement = record keeping

1) the past is immutable
2) everyone agrees on the history



- Partition time into époques / periods / time windows
- Anything that happens in one time period is recorded into a block
- Any change to an ‘old’ block affects all following blocks

how can we implement this property using a cryptographic object?

Cryptographic Hash Functions

$$H : \{0,1\}^* \rightarrow \{0,1\}^N$$

- Any size input
- fixed-size output
- deterministic function
- yet the output (hash digest) looks “random”

Cryptographic Hash Functions

$$H : \{0,1\}^* \rightarrow \{0,1\}^N$$

- Any size input
- fixed-size output
- deterministic function
- yet the output (hash digest) looks “random”

```
sha256("Advanced Web Security”) =
f61d24aff60c141a0c0606cad04d7b4776db9bef9c33c0ea38766f48b5edf8ac


sha256(“EITN41") =
fa260b32967226899bdbf7b047ba6fc3bcdc23d3d536f32760fcc7d28160b70a
```

Cryptographic Hash Functions

$$H : \{0,1\}^* \rightarrow \{0,1\}^N$$

- Any size input
- fixed-size output
- deterministic function
- yet the output (hash digest) looks “random”

```
sha256("Advanced Web Security”) =
f61d24aff60c141a0c0606cad04d7b4776db9bef9c33c0ea38766f48b5edf8ac


sha256(“EITN41") =
fa260b32967226899bdbf7b047ba6fc3bcdc23d3d536f32760fcc7d28160b70a


sha256(“EITN42”) =
6cff517032f167f06b5ded14a61cfa237718e955341f8f85289319104b8897bf
```

Cryptographic Hash Functions

$$H : \{0,1\}^* \rightarrow \{0,1\}^N$$

- Any size input
- fixed-size output
- deterministic function
- yet the output (hash digest) looks “random”

```
sha256("Advanced Web Security”) =
f61d24aff60c141a0c0606cad04d7b4776db9bef9c33c0ea38766f48b5edf8ac


sha256(“EITN41") =
fa260b32967226899bdbf7b047ba6fc3bcdc23d3d536f32760fcc7d28160b70a


sha256(“EITN42”) =
6cff517032f167f06b5ded14a61cfa237718e955341f8f85289319104b8897bf
```

“avalanche effect”

Cryptographic Hash Functions

Cryptographic Hash Functions

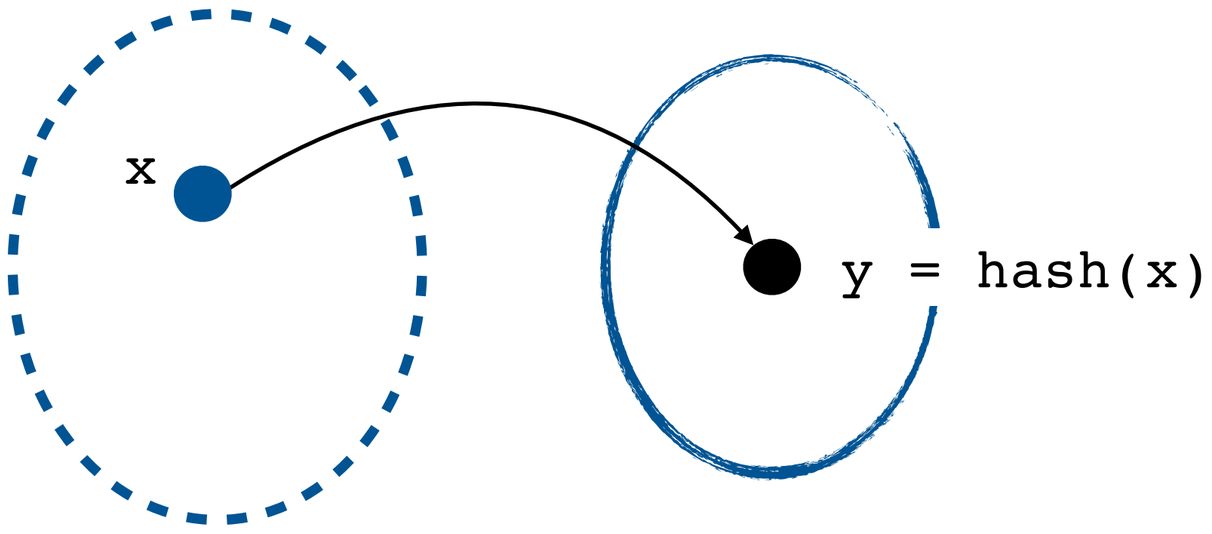
(First) preimage resistance

(you can find it eventually, 2^256 ~ 10^78 operations)

Cryptographic Hash Functions

(First) preimage resistance

(you can find it eventually, 2^256 ~ 10^78 operations)

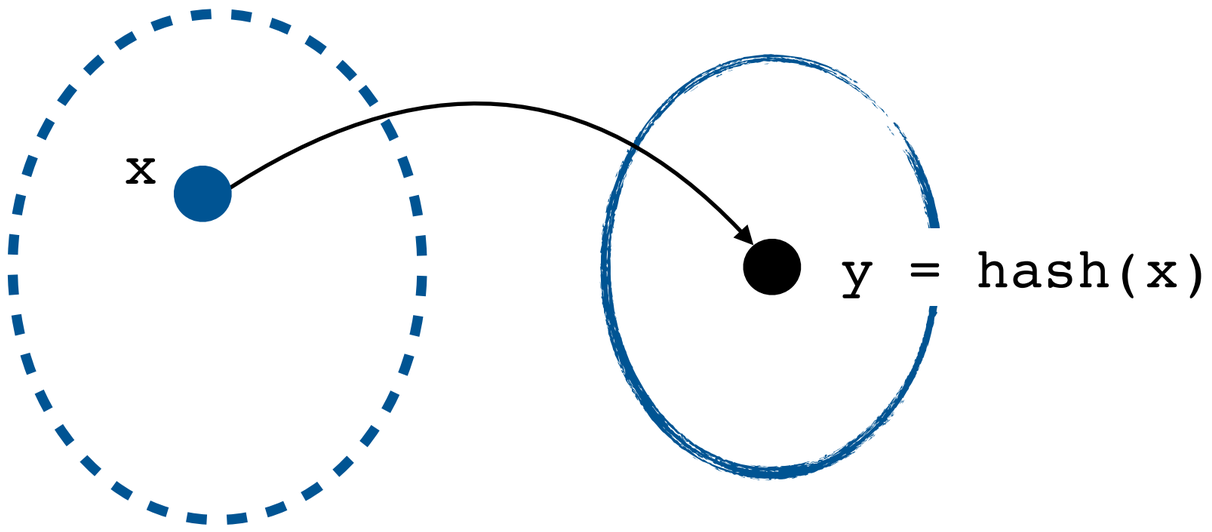


Cryptographic Hash Functions

(First) preimage resistance

(you can find it eventually, 2^256 ~ 10^78 operations)

Second preimage resistance

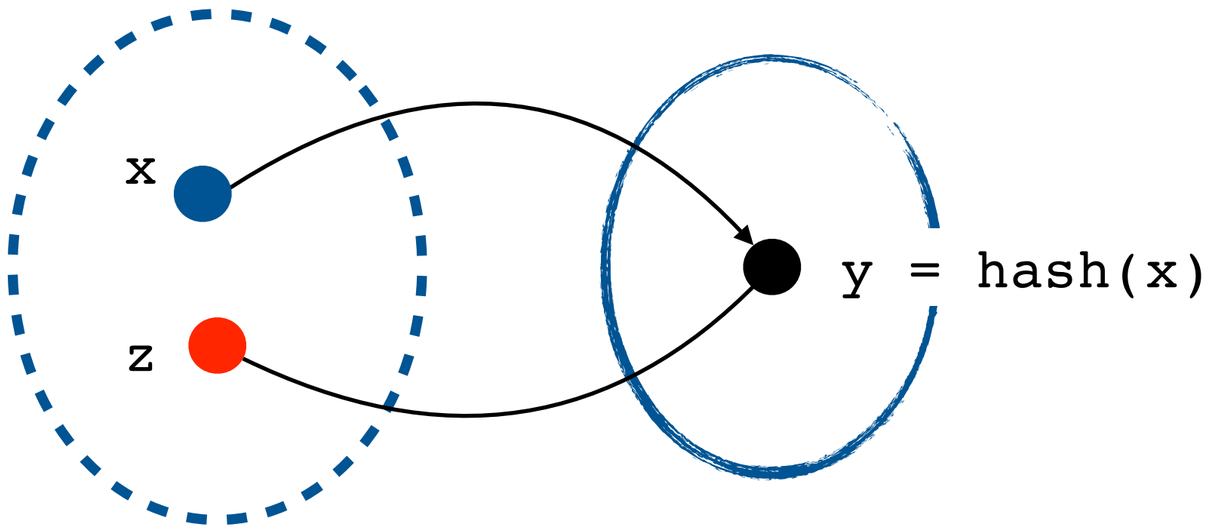


Cryptographic Hash Functions

(First) preimage resistance

(you can find it eventually, 2^256 ~ 10^78 operations)

Second preimage resistance



Cryptographic Hash Functions

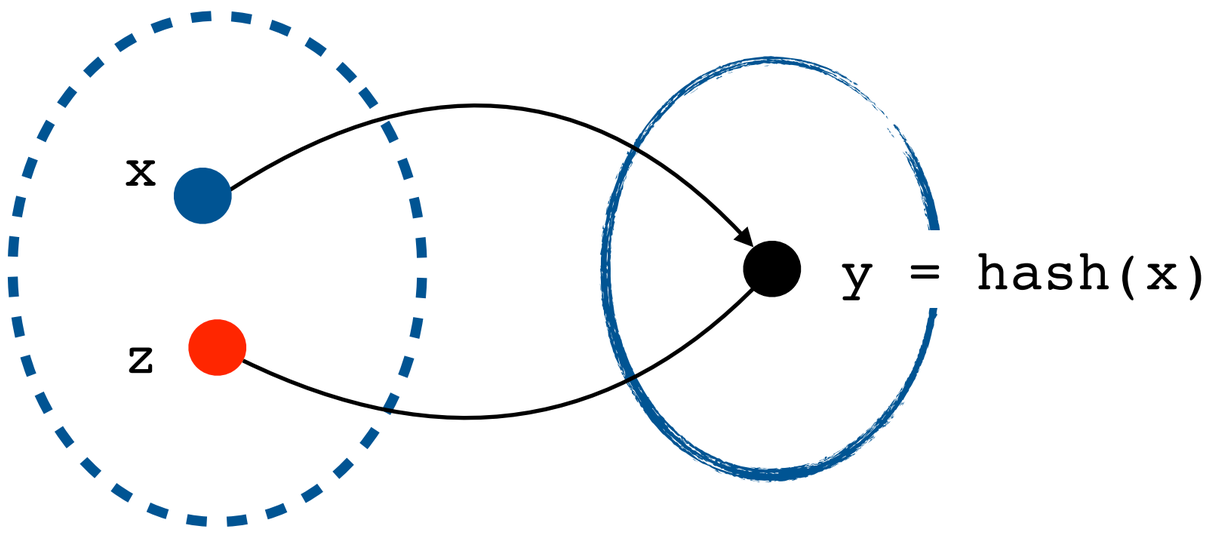
(First) preimage resistance

(you can find it eventually, 2^256 ~ 10^78 operations)

Second preimage resistance

Collision resistance

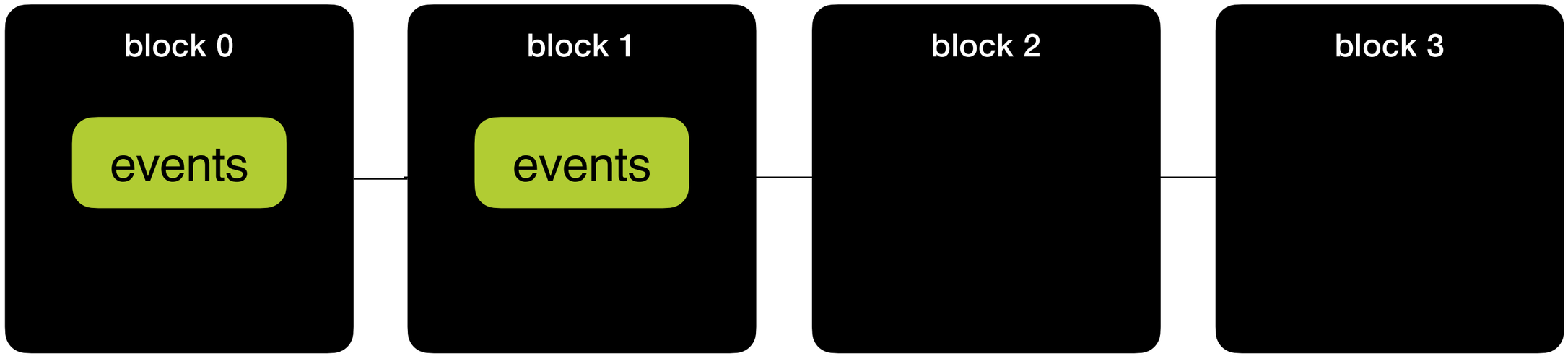
(you can find it eventually, in less than 2^256, expected ~ 2^128)



How To Set Up a Bulletin Board

Main property we want to implement = record keeping

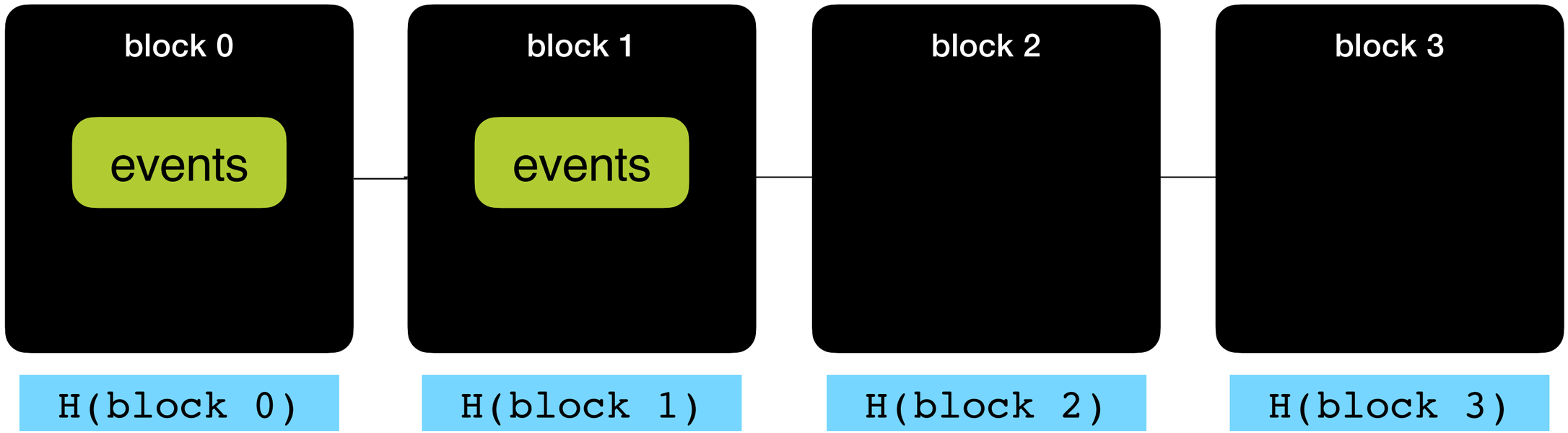
1) the past is immutable
2) everyone agrees on the history



How To Set Up a Bulletin Board

Main property we want to implement = record keeping

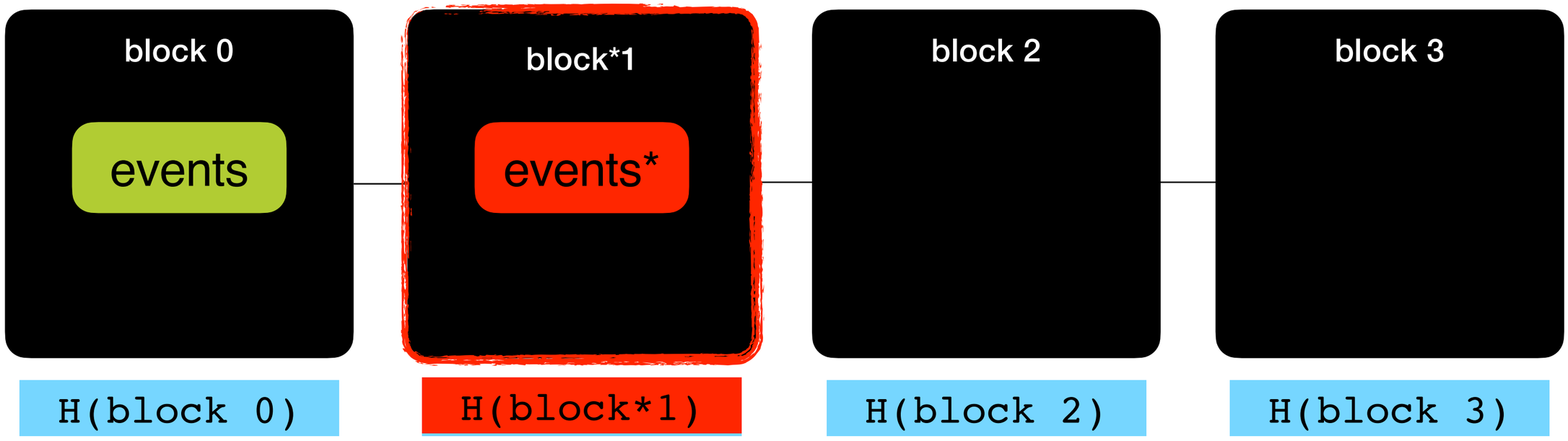
1) the past is immutable
2) everyone agrees on the history



How To Set Up a Bulletin Board

Main property we want to implement = record keeping

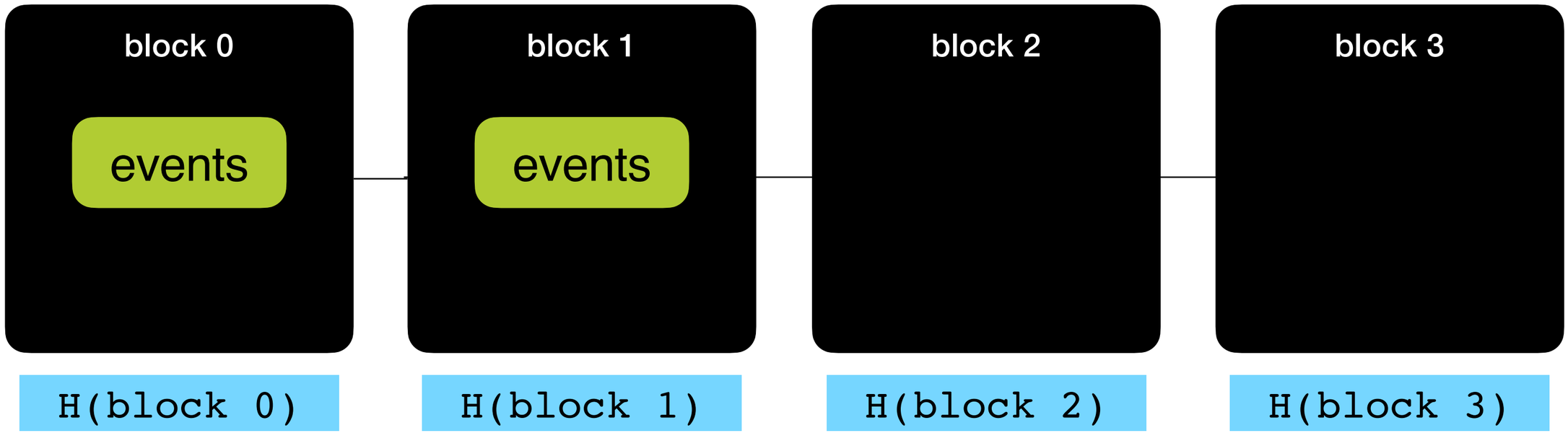
1) the past is immutable
2) everyone agrees on the history



How To Set Up a Bulletin Board

Main property we want to implement = record keeping

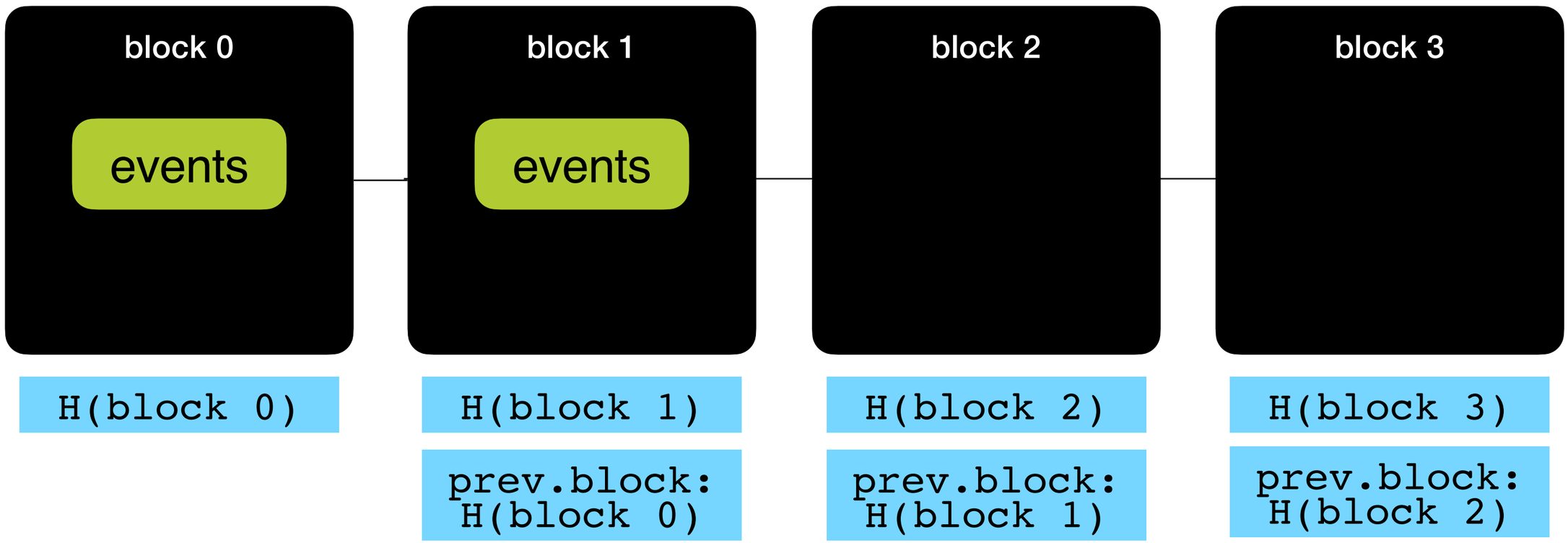
1) the past is immutable
2) everyone agrees on the history



How To Set Up a Bulletin Board

Main property we want to implement = record keeping

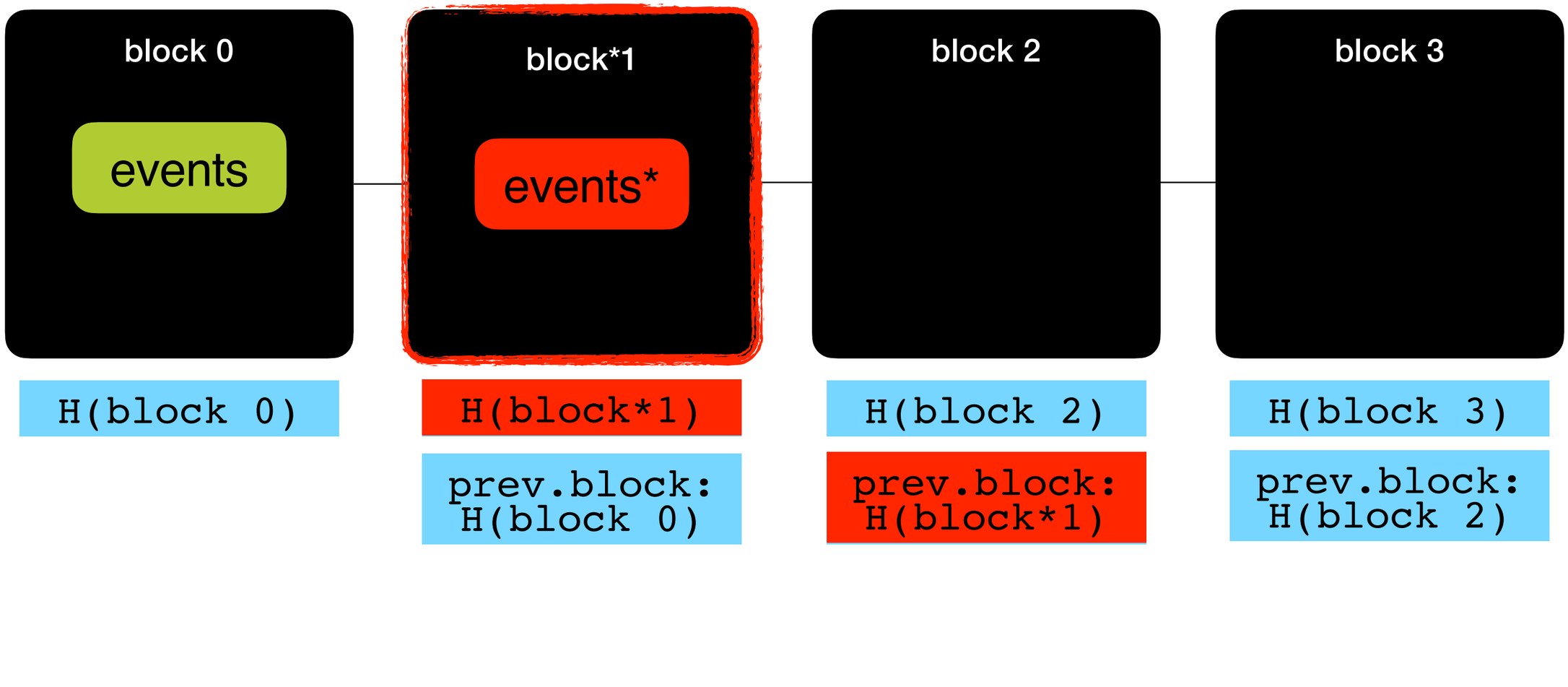
1) the past is immutable
2) everyone agrees on the history



How To Set Up a Bulletin Board

Main property we want to implement = record keeping

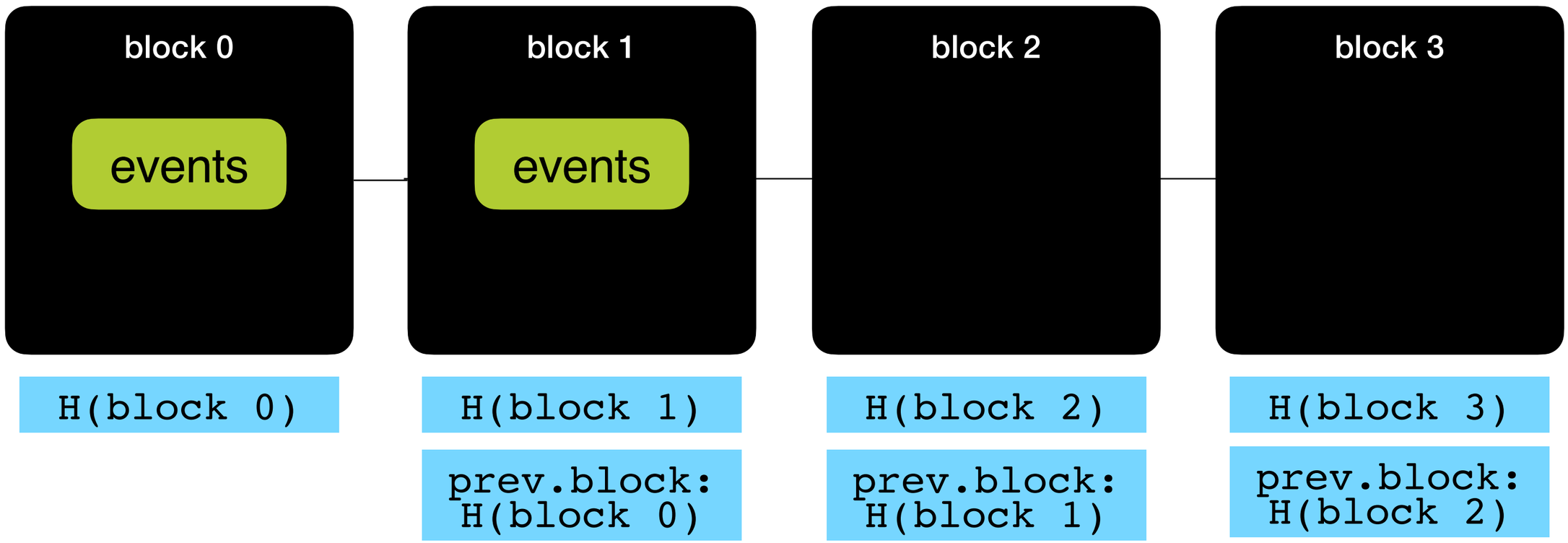
1) the past is immutable
2) everyone agrees on the history



How To Set Up a Bulletin Board

Main property we want to implement = record keeping

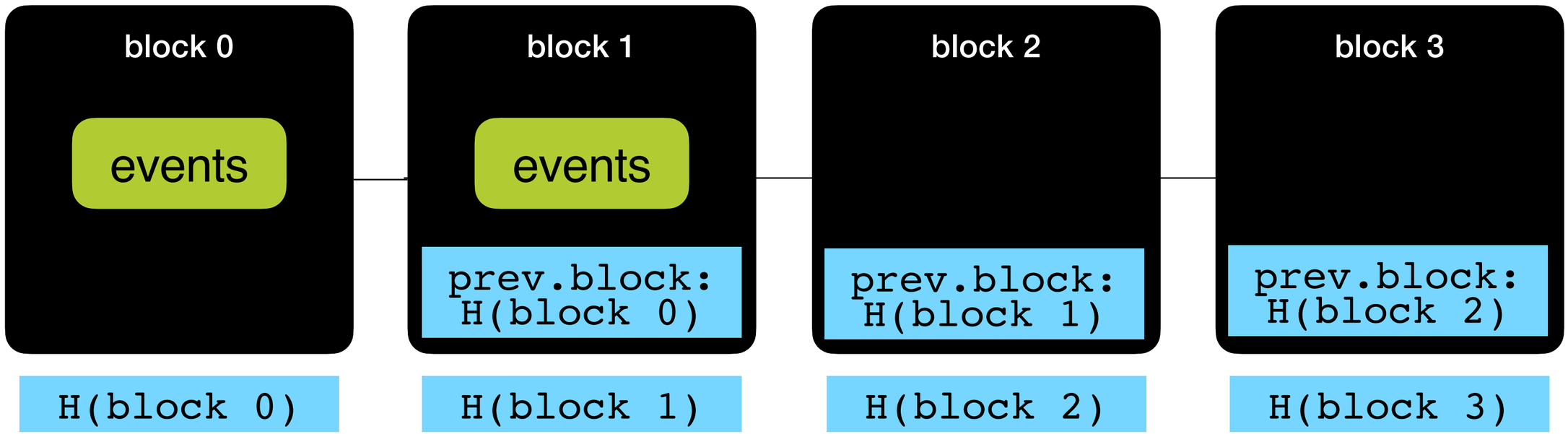
1) the past is immutable
2) everyone agrees on the history



How To Set Up a Bulletin Board

Main property we want to implement = record keeping

1) the past is immutable
2) everyone agrees on the history

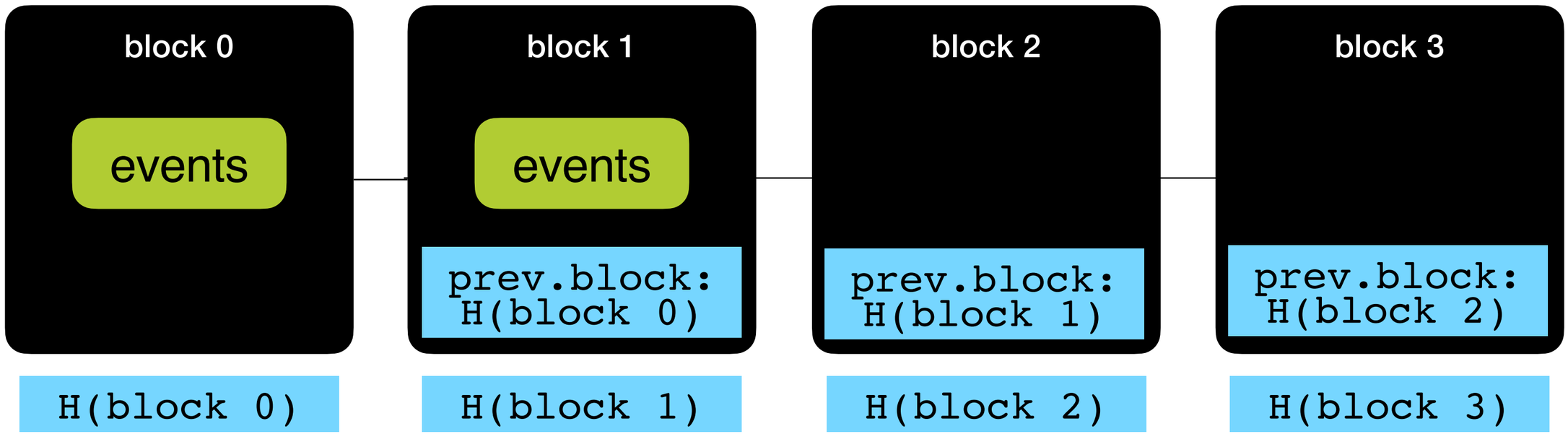


How To Set Up a Bulletin Board

Main property we want to implement = record keeping

1) the past is immutable
2) everyone agrees on the history

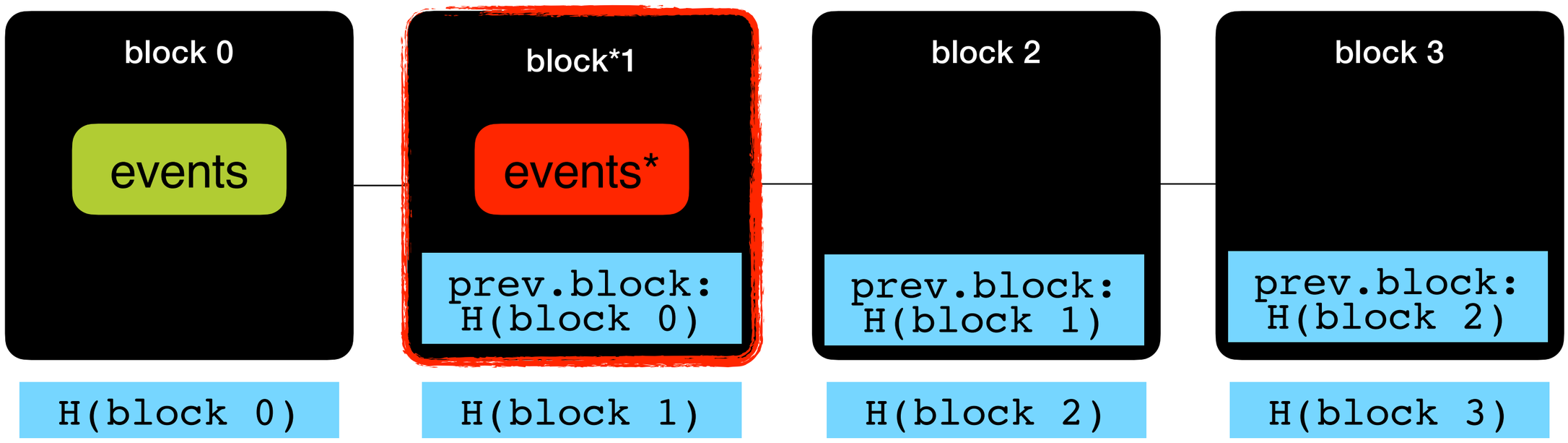
use the hash function to chain blocks



How To Set Up a Bulletin Board

Main property we want to implement = record keeping

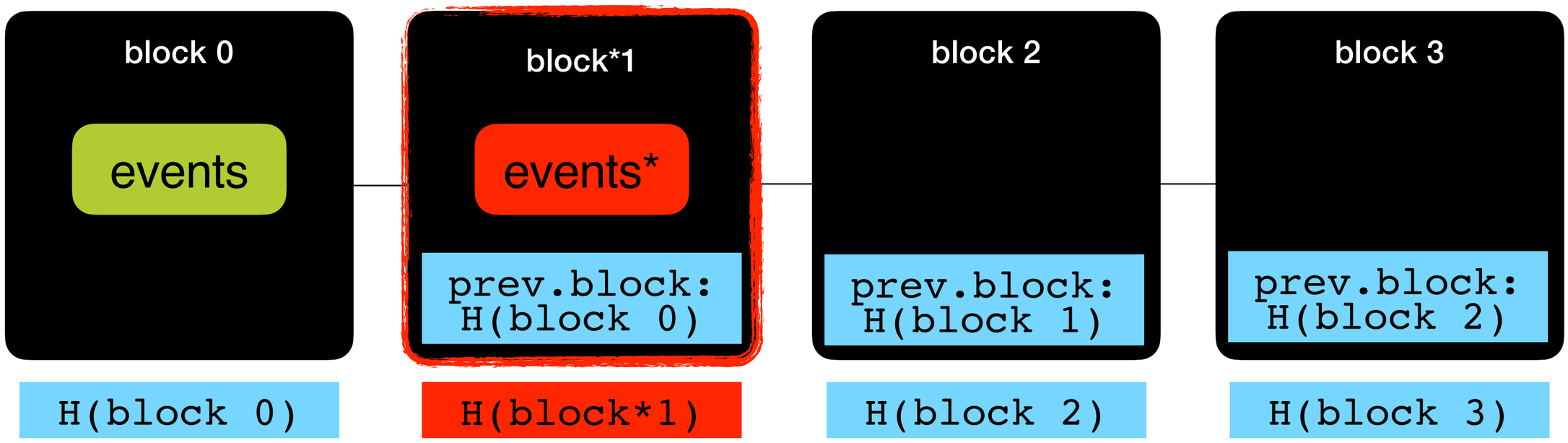
1) the past is immutable ⇦ use the hash function to chain blocks
2) everyone agrees on the history



How To Set Up a Bulletin Board

Main property we want to implement = record keeping

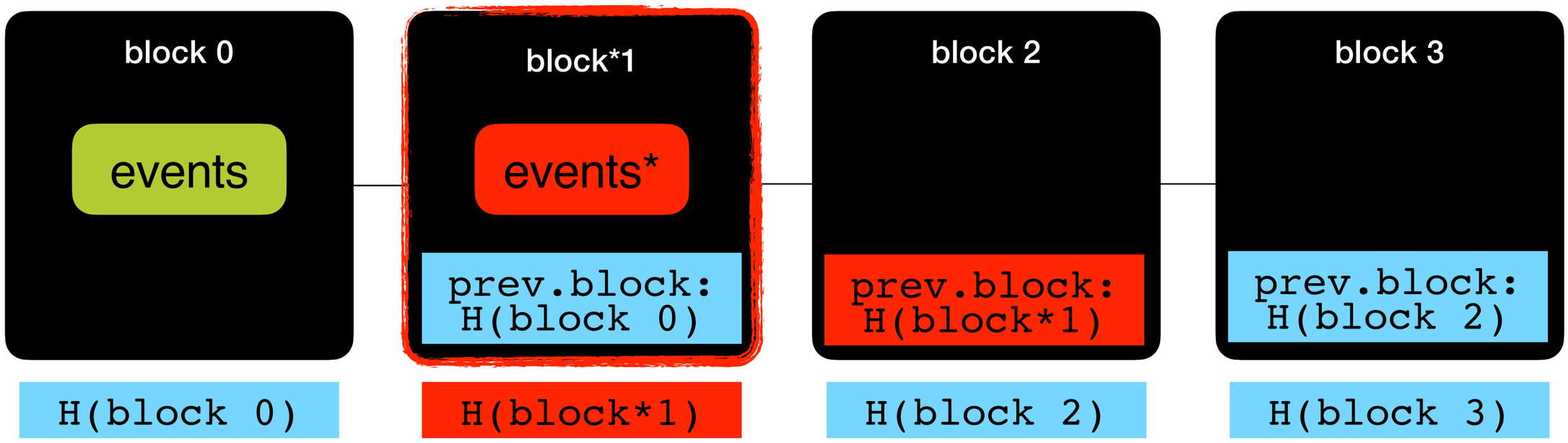
1) the past is immutable ⇦ use the hash function to chain blocks
2) everyone agrees on the history



How To Set Up a Bulletin Board

Main property we want to implement = record keeping

1) the past is immutable ⇦ use the hash function to <u>chain</u> blocks
2) everyone agrees on the history

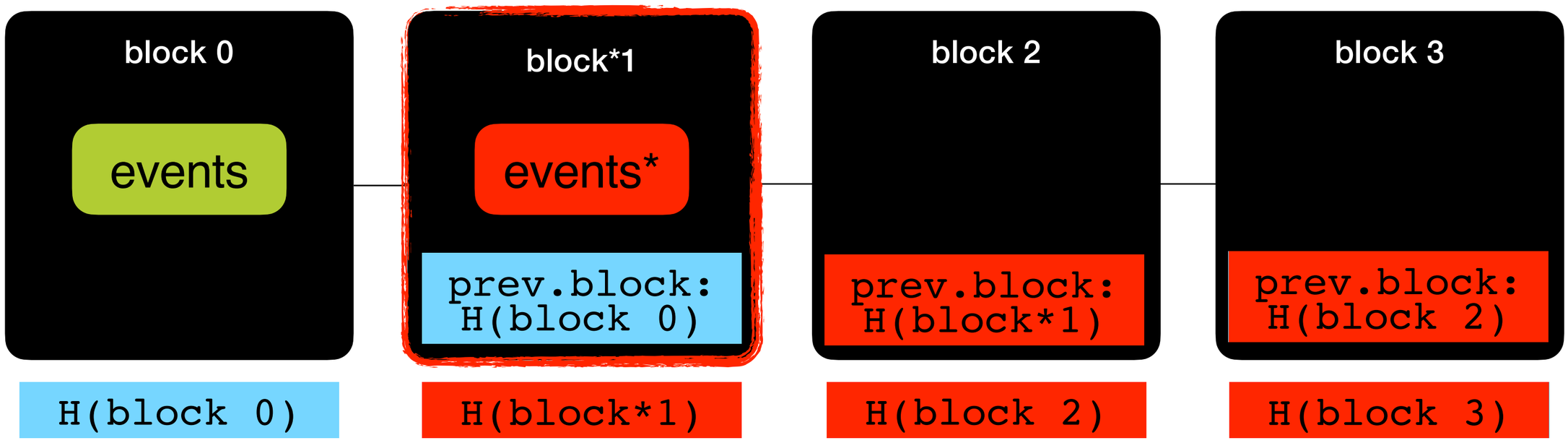


How To Set Up a Bulletin Board

Main property we want to implement = record keeping

1) the past is immutable
2) everyone agrees on the history

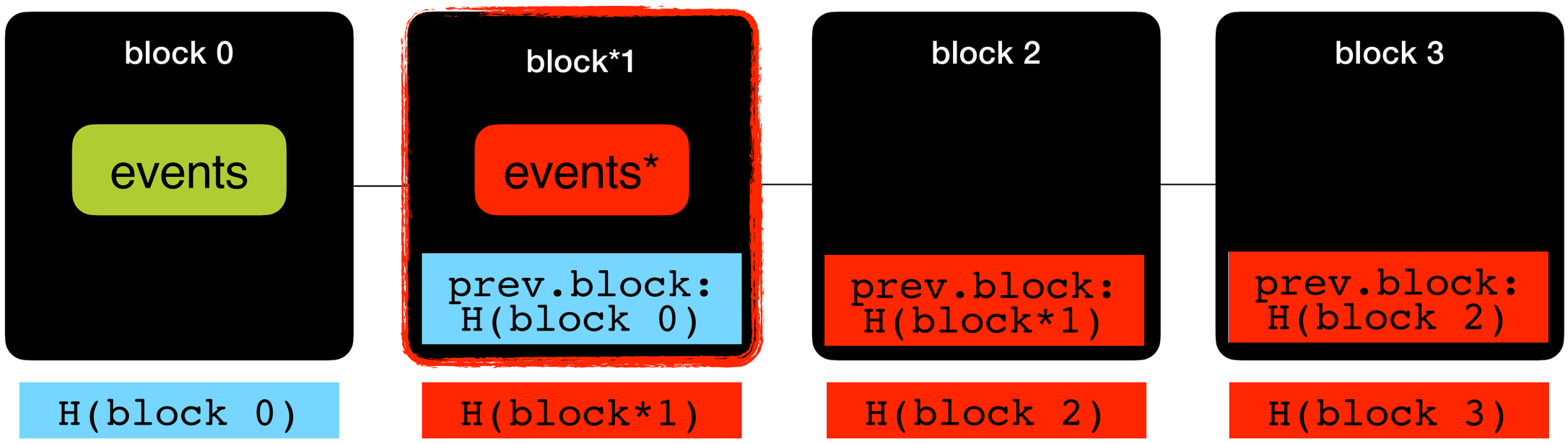
use the hash function to chain blocks



How To Set Up a Bulletin Board

Main property we want to implement = record keeping

1) the past is immutable ⇦ use the hash function to chain blocks
2) everyone agrees on the history

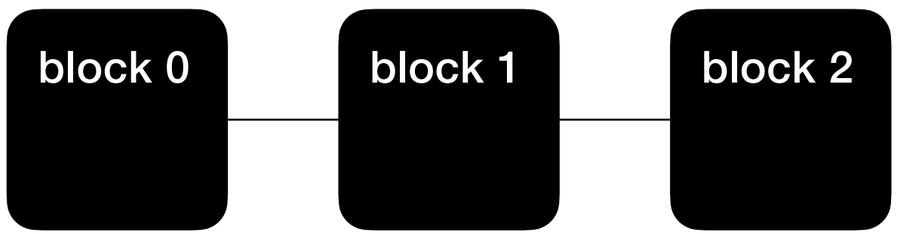


⇨ Any change to an ‘old’ block affects all following blocks

How To Set Up a Bulletin Board

Main property we want to implement = record keeping

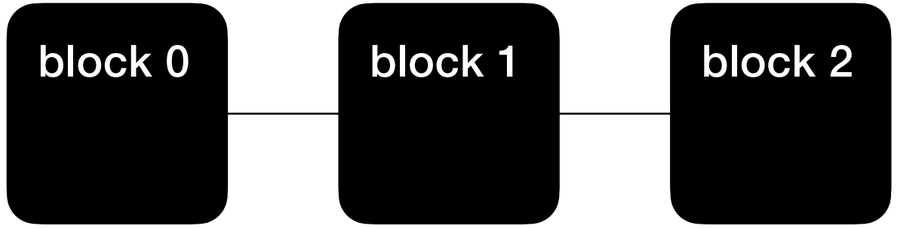
1) the past is immutable
2) everyone agrees on the history



How To Set Up a Bulletin Board

Main property we want to implement = record keeping

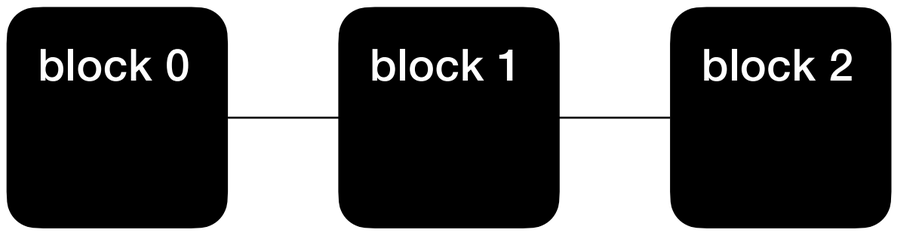
1) the past is immutable ✓
2) everyone agrees on the history ⇦



How To Set Up a Bulletin Board

Main property we want to implement = record keeping

1) the past is immutable ✓
2) everyone agrees on the history ⇦

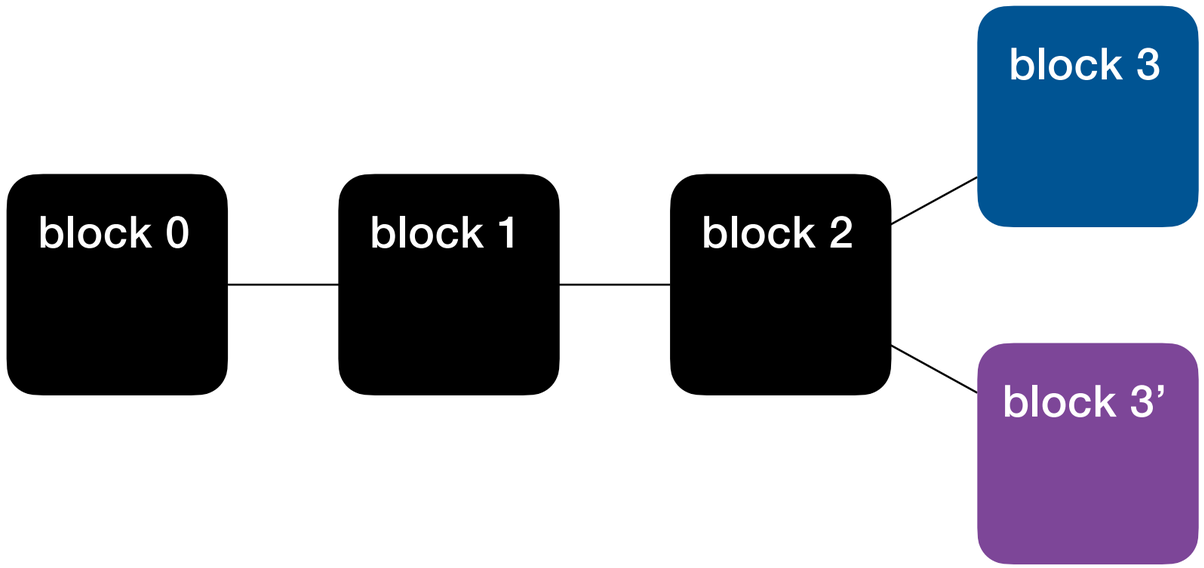


- Always build on the longest branch (longest chain rule)

How To Set Up a Bulletin Board

Main property we want to implement = record keeping

1) the past is immutable ✓
2) everyone agrees on the history ⇦

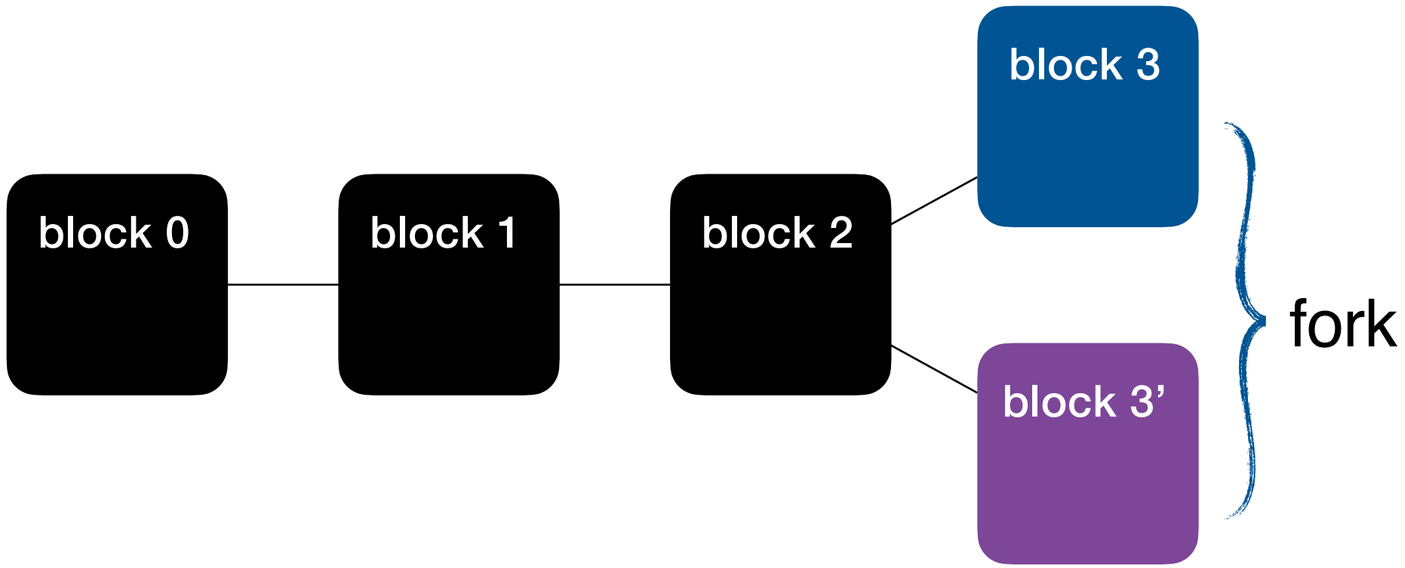


- Always build on the longest branch (longest chain rule)

How To Set Up a Bulletin Board

Main property we want to implement = record keeping

1) the past is immutable ✓
2) everyone agrees on the history ⇦

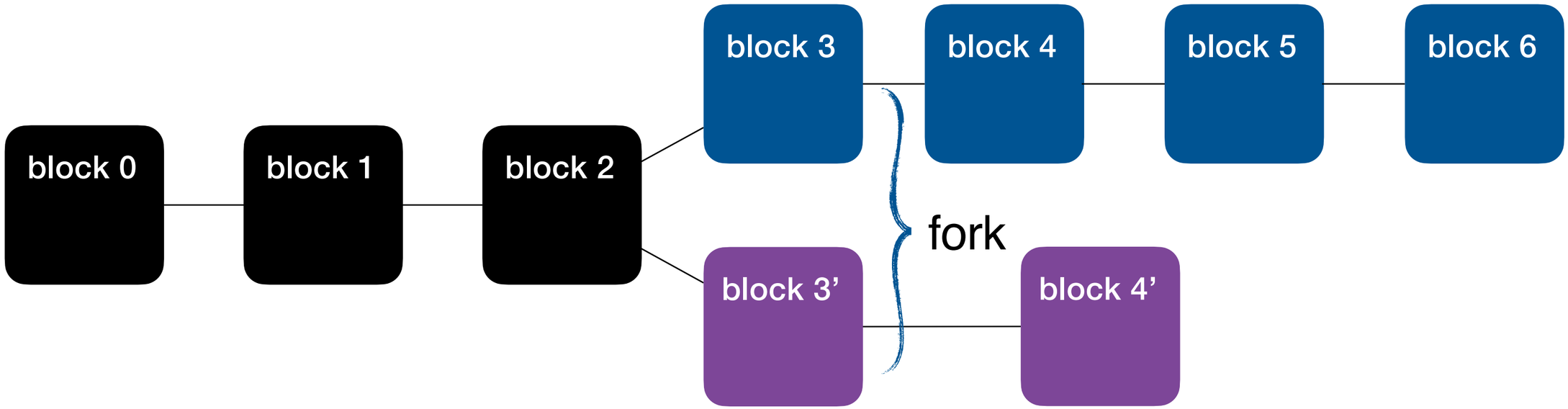


- Always build on the longest branch (longest chain rule)

How To Set Up a Bulletin Board

Main property we want to implement = record keeping

1) the past is immutable ✓
2) everyone agrees on the history ⇦

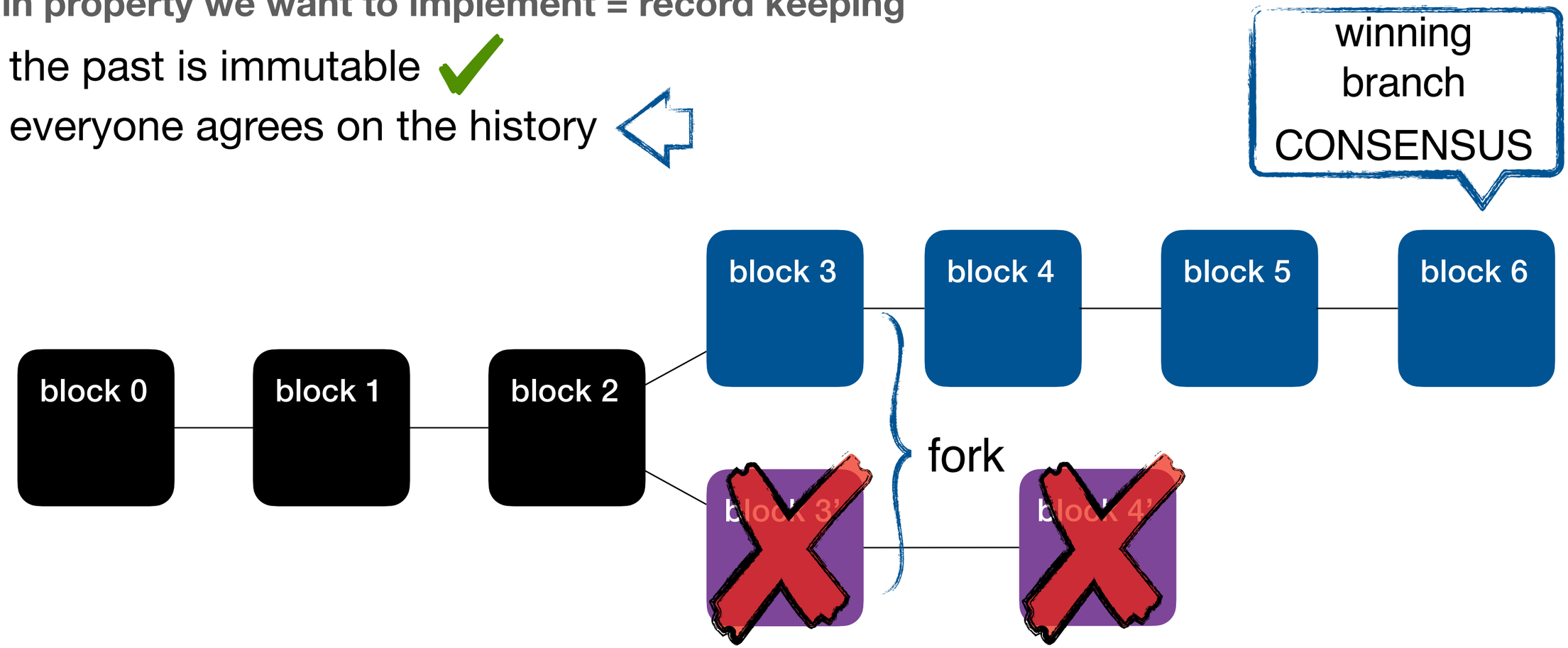


- Always build on the longest branch (longest chain rule)

How To Set Up a Bulletin Board

Main property we want to implement = record keeping

1) the past is immutable ✓
2) everyone agrees on the history ⇦

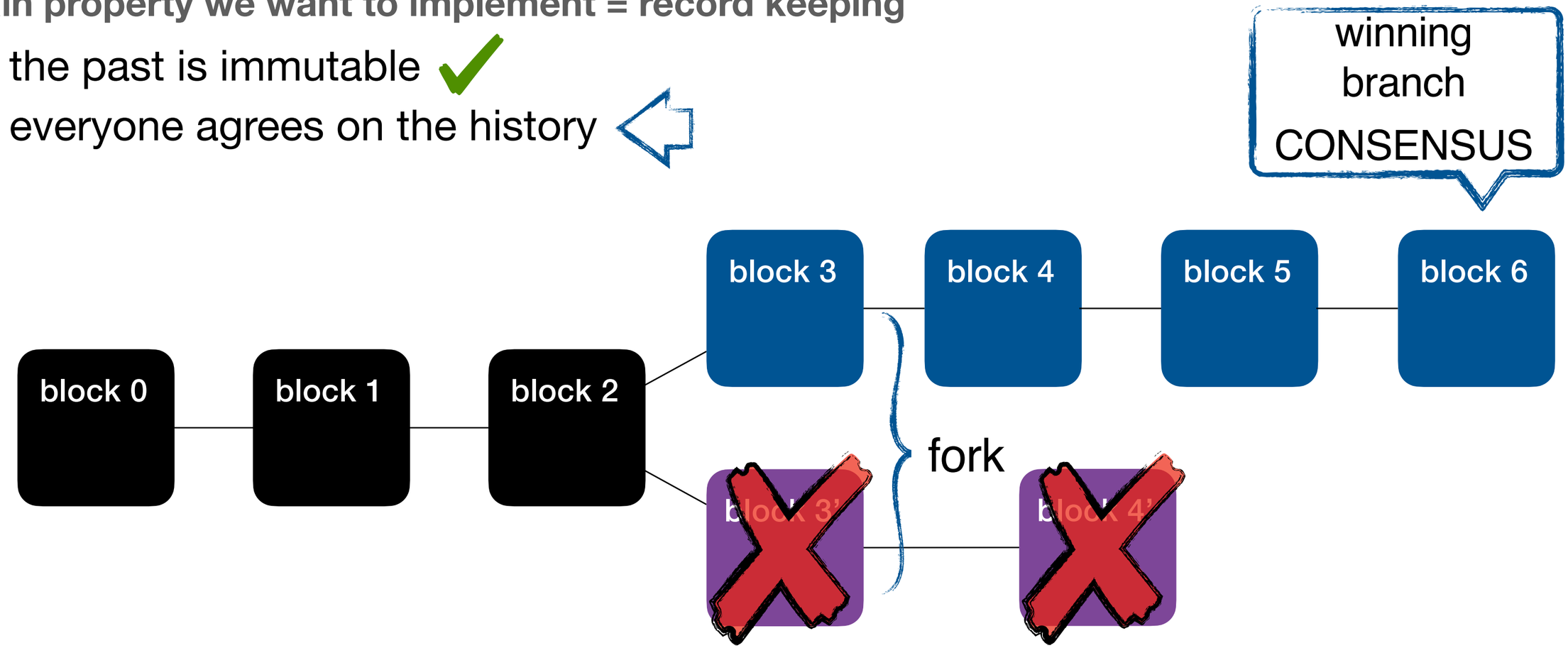


- Always build on the longest branch (longest chain rule)

How To Set Up a Bulletin Board

Main property we want to implement = record keeping

1) the past is immutable ✓
2) everyone agrees on the history ⇦

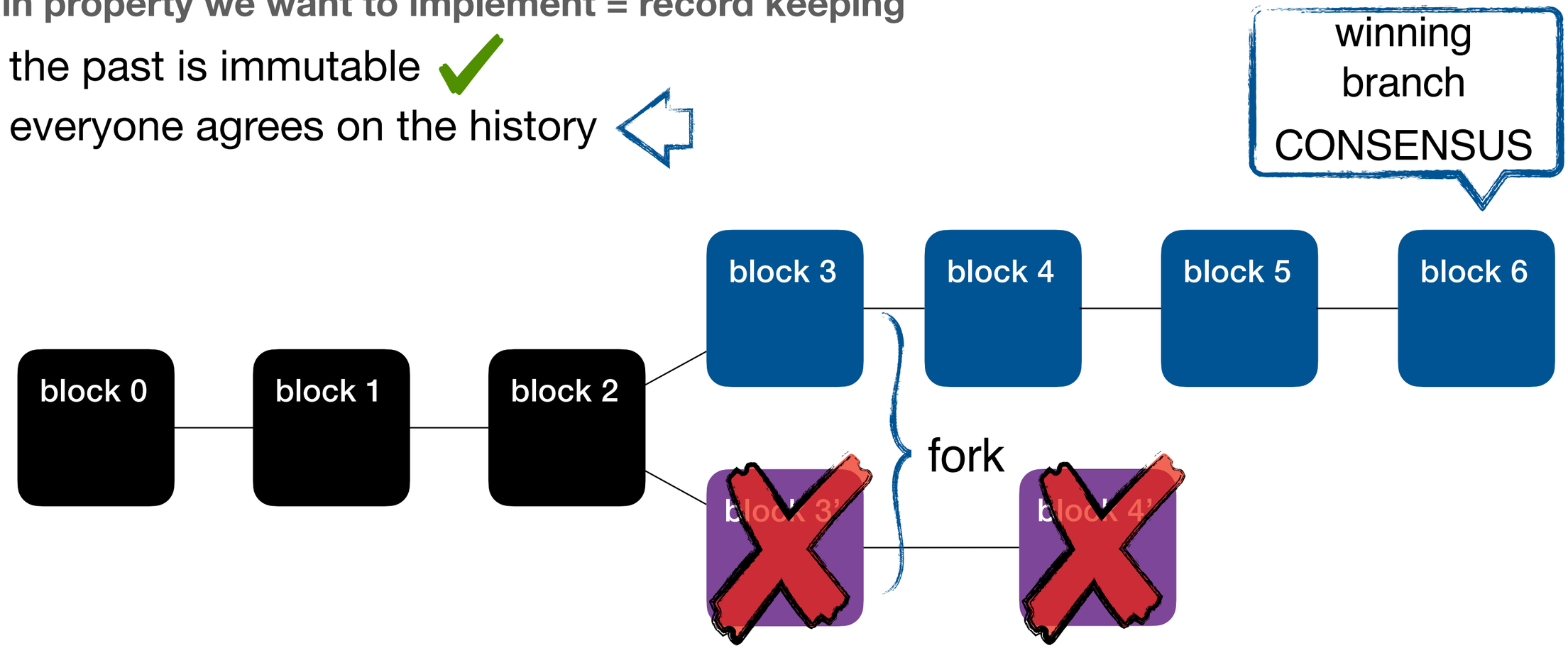


- Always build on the longest branch (longest chain rule)
- How to lower the chance that blocks appear at the same time?

How To Set Up a Bulletin Board

Main property we want to implement = record keeping

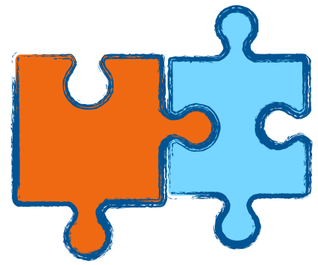
1) the past is immutable ✓
2) everyone agrees on the history ⇦



- Always build on the longest branch (longest chain rule)
- How to lower the chance that blocks appear at the same time?

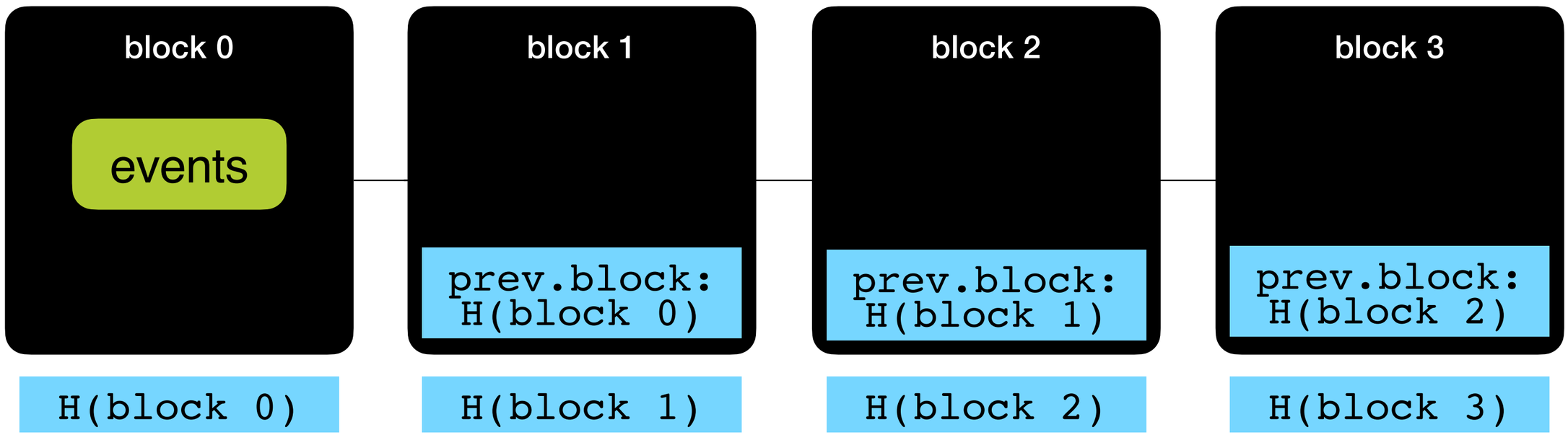
Proof of Work

Proof of Work (Cryptographic Hash Puzzles)

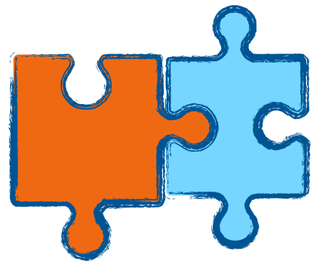


Main property we want to implement = record keeping

1) the past is immutable ✓
2) everyone agrees on the history ⇦

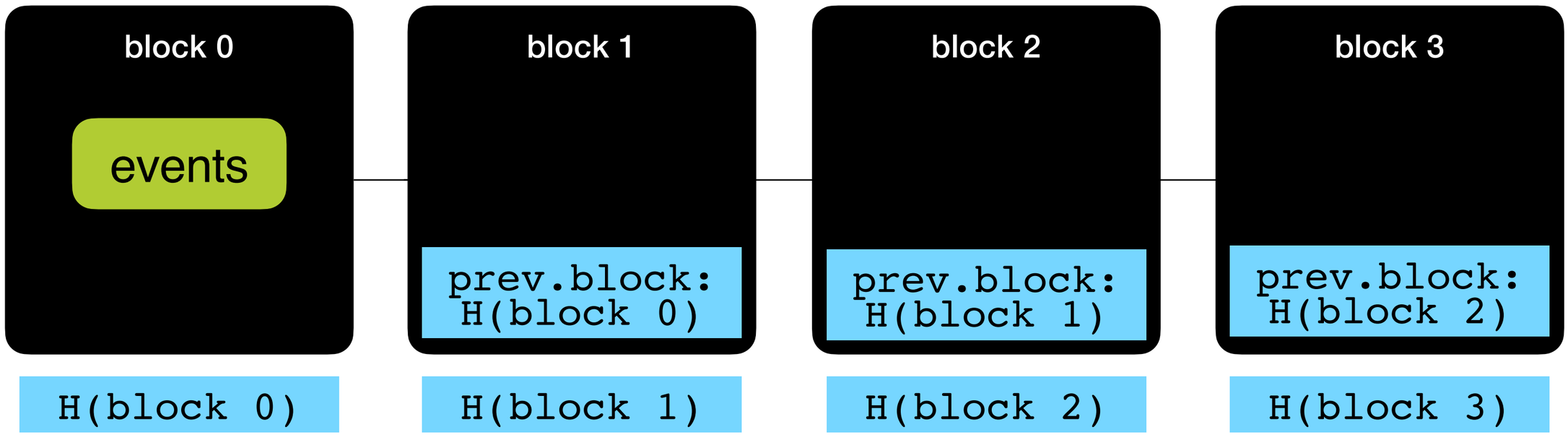


Proof of Work (Cryptographic Hash Puzzles)

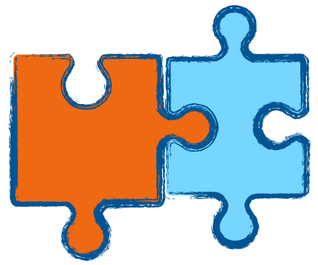


Main property we want to implement = record keeping

1) the past is immutable ✓
2) everyone agrees on the history ⇦ put a rule that makes it <u>hard</u> to compute a ‘good’ hash digest

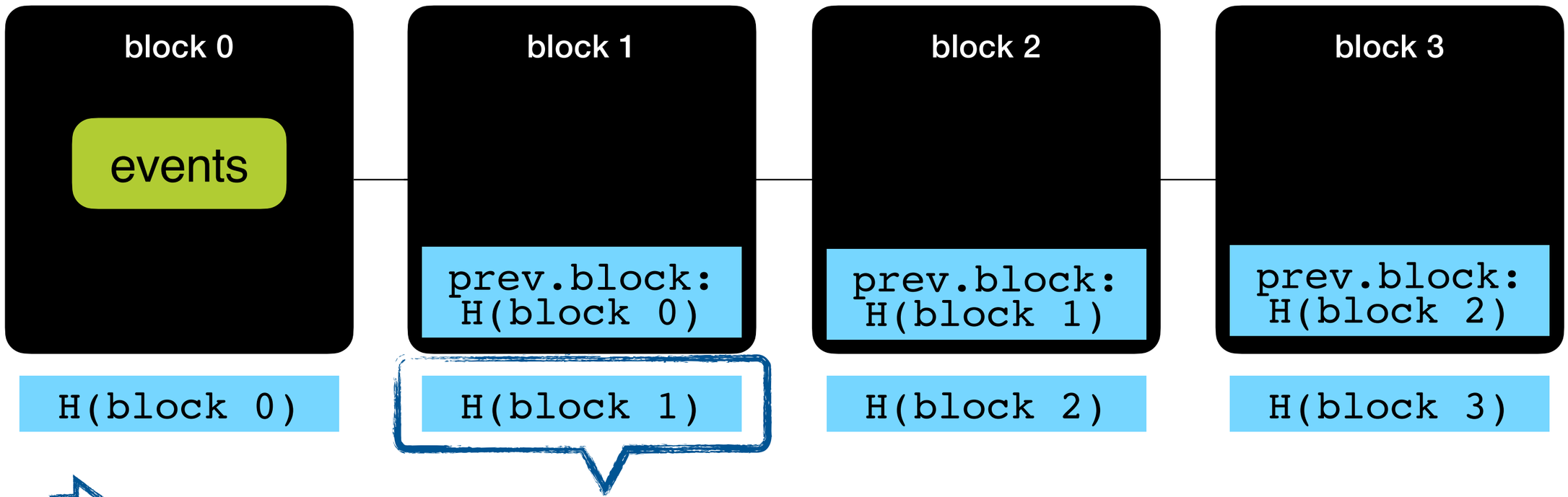


Proof of Work (Cryptographic Hash Puzzles)



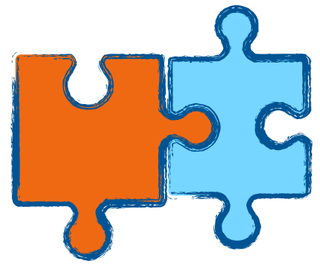
Main property we want to implement = record keeping

1) the past is immutable ✓
2) everyone agrees on the history ⇦ put a rule that makes it <u>hard</u> to compute a ‘good’ hash digest



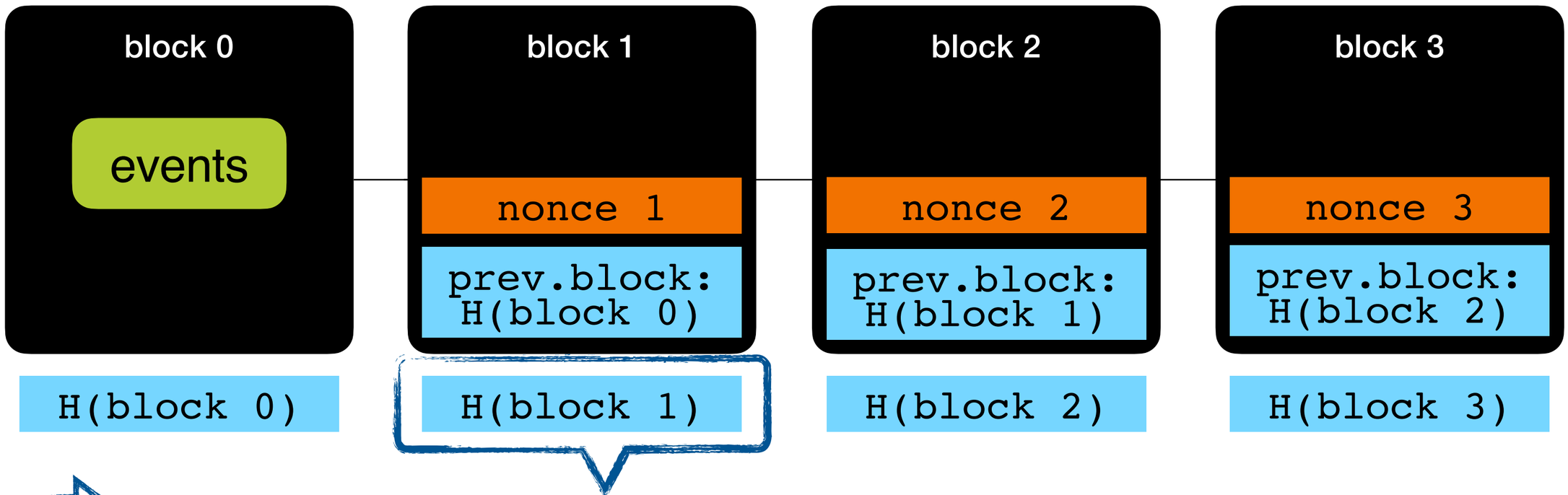
⇨ *RULE : EACH BLOCK HASH NEEDS TO START WITH A GIVEN* ***PREFIX***

Proof of Work (Cryptographic Hash Puzzles)



Main property we want to implement = record keeping

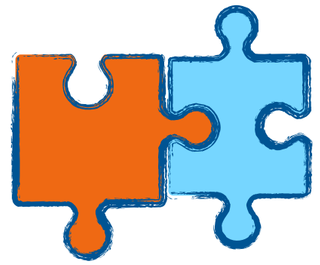
1) the past is immutable ✓
2) everyone agrees on the history ⇦ put a rule that makes it <u>hard</u> to compute a ‘good’ hash digest



⇨ *RULE : EACH BLOCK HASH NEEDS TO START WITH A GIVEN **PREFIX***

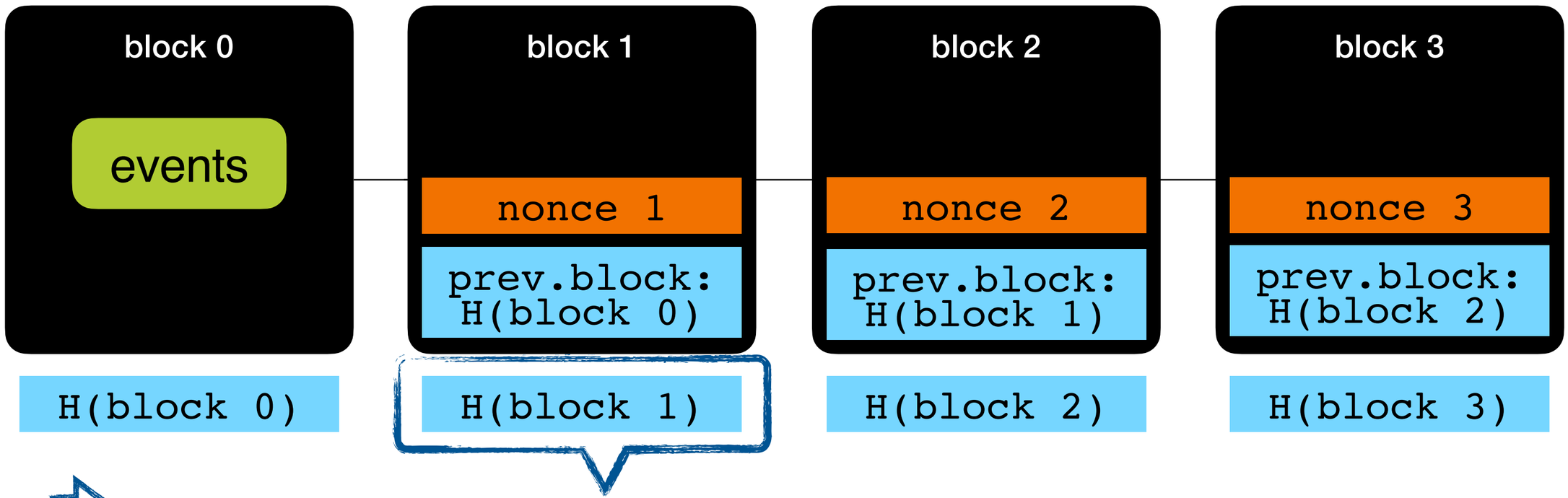
find a value **nonce** such that **H(block||nonce) = 00000******

Proof of Work (Cryptographic Hash Puzzles)



Main property we want to implement = record keeping

1) the past is immutable ✓
2) everyone agrees on the history ⇦ put a rule that makes it <u>hard</u> to compute a ‘good’ hash digest



⇨ *RULE : EACH BLOCK HASH NEEDS TO START WITH A GIVEN **PREFIX***

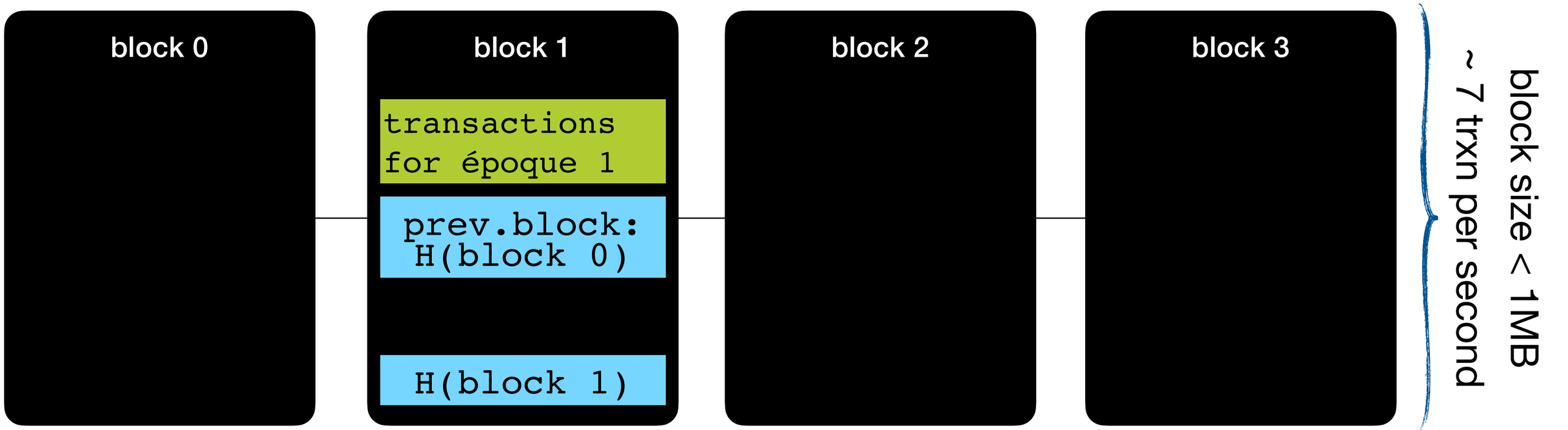
find a value **nonce** such that **H(block||nonce) = 00000******

```
sha256("Advanced Web Security")                = f61d24aff60c141a0c0606cad04d7b4776db9bef9c33c0ea38766f48b5edf8ac
sha256("Advanced Web Security"||878140116400) = 00000d5999cf256d9b8d29a28433223ef51051c956bb1e3d22fcbf7208cf34f
```

Example: Bitcoin

set by the puzzle difficulty
(currently 19 leading zeroes)

époque ~ 10 mins



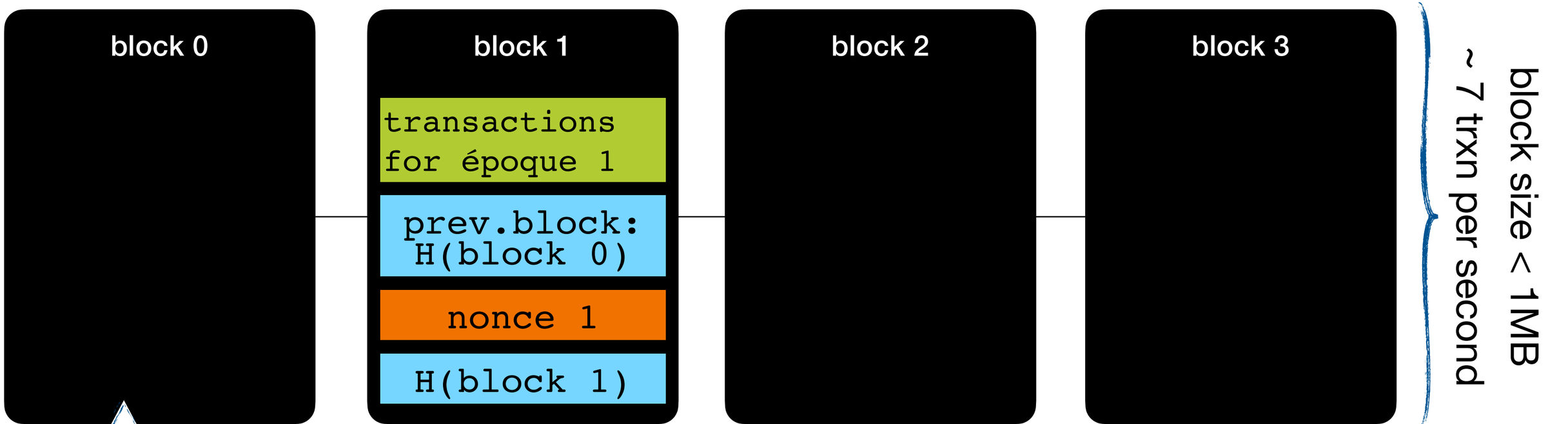
genesis block
created by Nakamoto
on 03.01.2009

how large is the bitcoin chain by NOW?

Example: Bitcoin

set by the puzzle difficulty
(currently 19 leading zeroes)

époque ~ 10 mins



genesis block
created by Nakamoto
on 03.01.2009

how large is the bitcoin chain by NOW?

Lecture Agenda

How To Set Up a Bulletin Board

- Hash Functions
- Proof of Work (Cryptographic Puzzles)
- Example: Bitcoin

How To Generate Money

- Mining Blocks
- Merkle Trees
- Block Reword System

How To Circulate Money

- Transactions
- ECDSA

Some Problems With Blockchains

- Freshness of the Genesis Block
- 51% Attack
- Proof of Work VS Proof of Stake

Advanced Tools

- Proofs of Knowledge
- zkSNARKs
- MPC

Lecture Agenda

How To Set Up a Bulletin Board

- Hash Functions
- Proof of Work (Cryptographic Puzzles)
- Example: Bitcoin

How To Generate Money

- Mining Blocks
- Merkle Trees
- Block Reword System

How To Circulate Money

- Transactions
- ECDSA

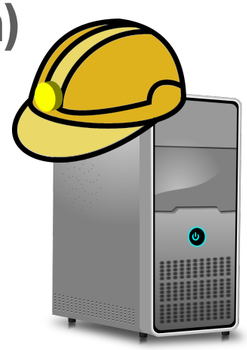
Some Problems With Blockchains

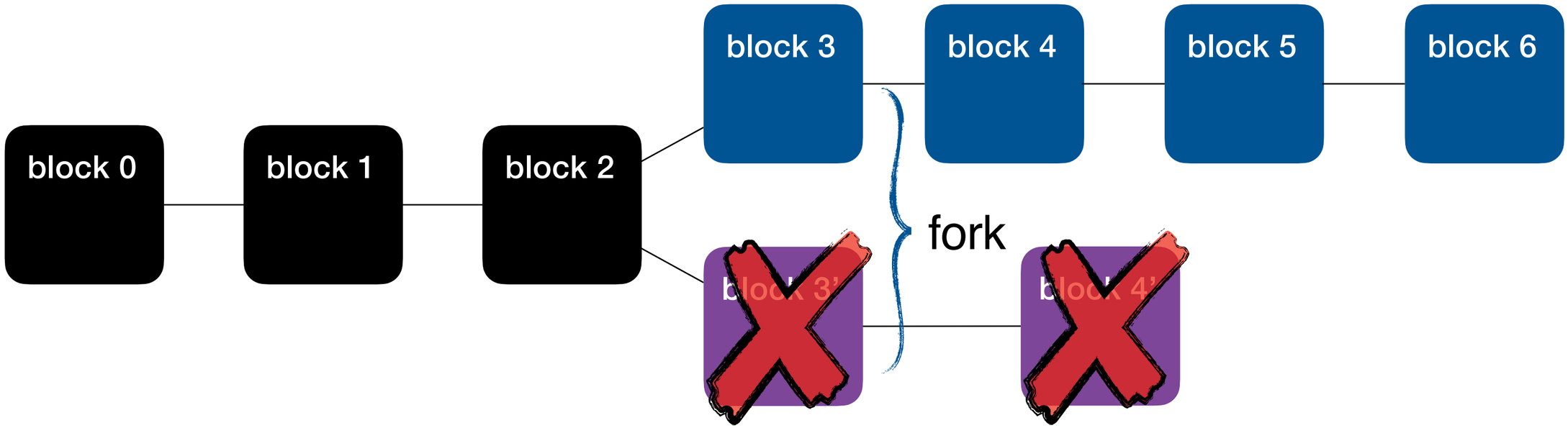
- Freshness of the Genesis Block
- 51% Attack
- Proof of Work VS Proof of Stake

Advanced Tools

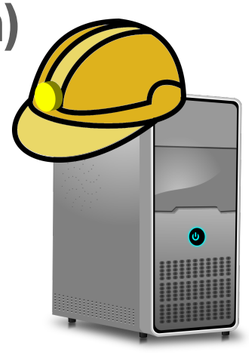
- Proofs of Knowledge
- zkSNARKs
- MPC

Mining Blocks (miner = first one to solve a hash puzzle on the longest chain)

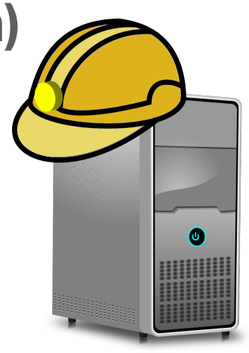




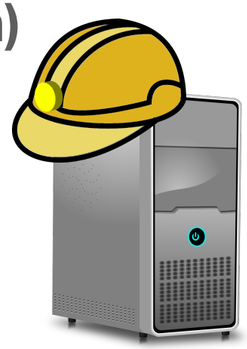
Mining Blocks (miner = first one to solve a hash puzzle on the longest chain)

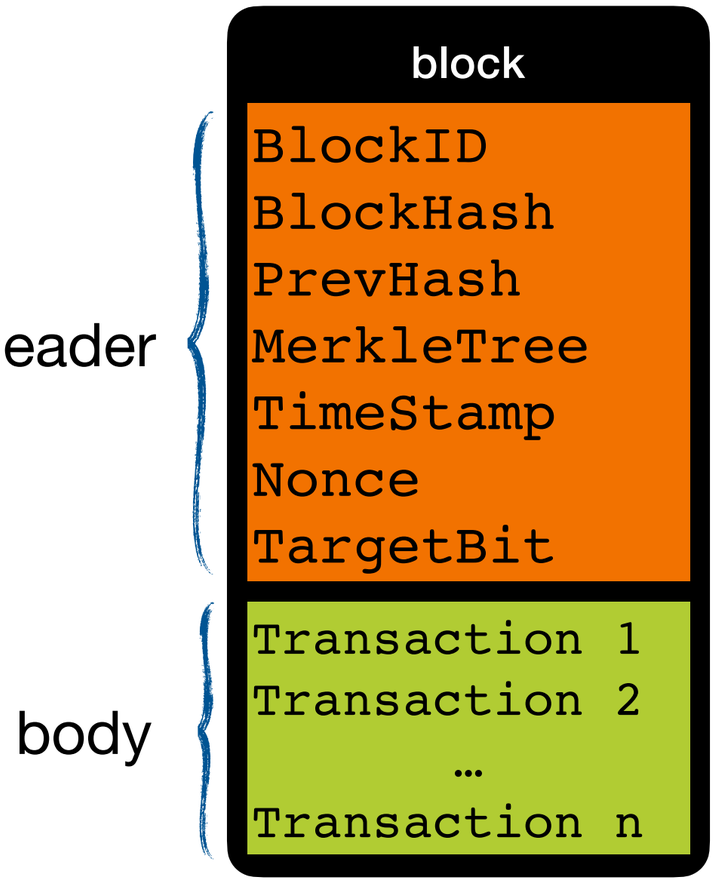


Mining Blocks (miner = first one to solve a hash puzzle on the longest chain)

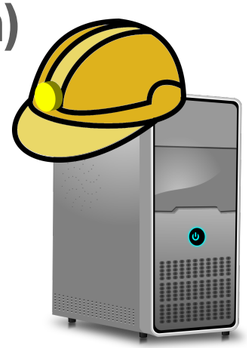


Mining Blocks (miner = first one to solve a hash puzzle on the longest chain)



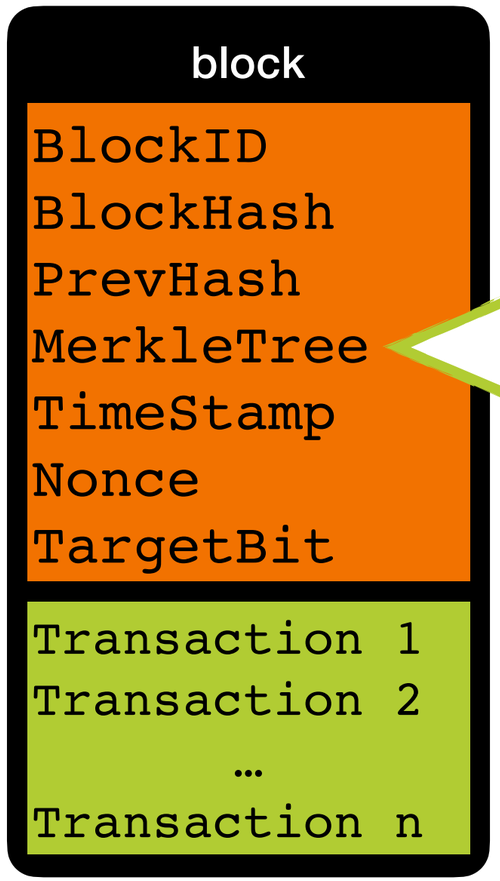


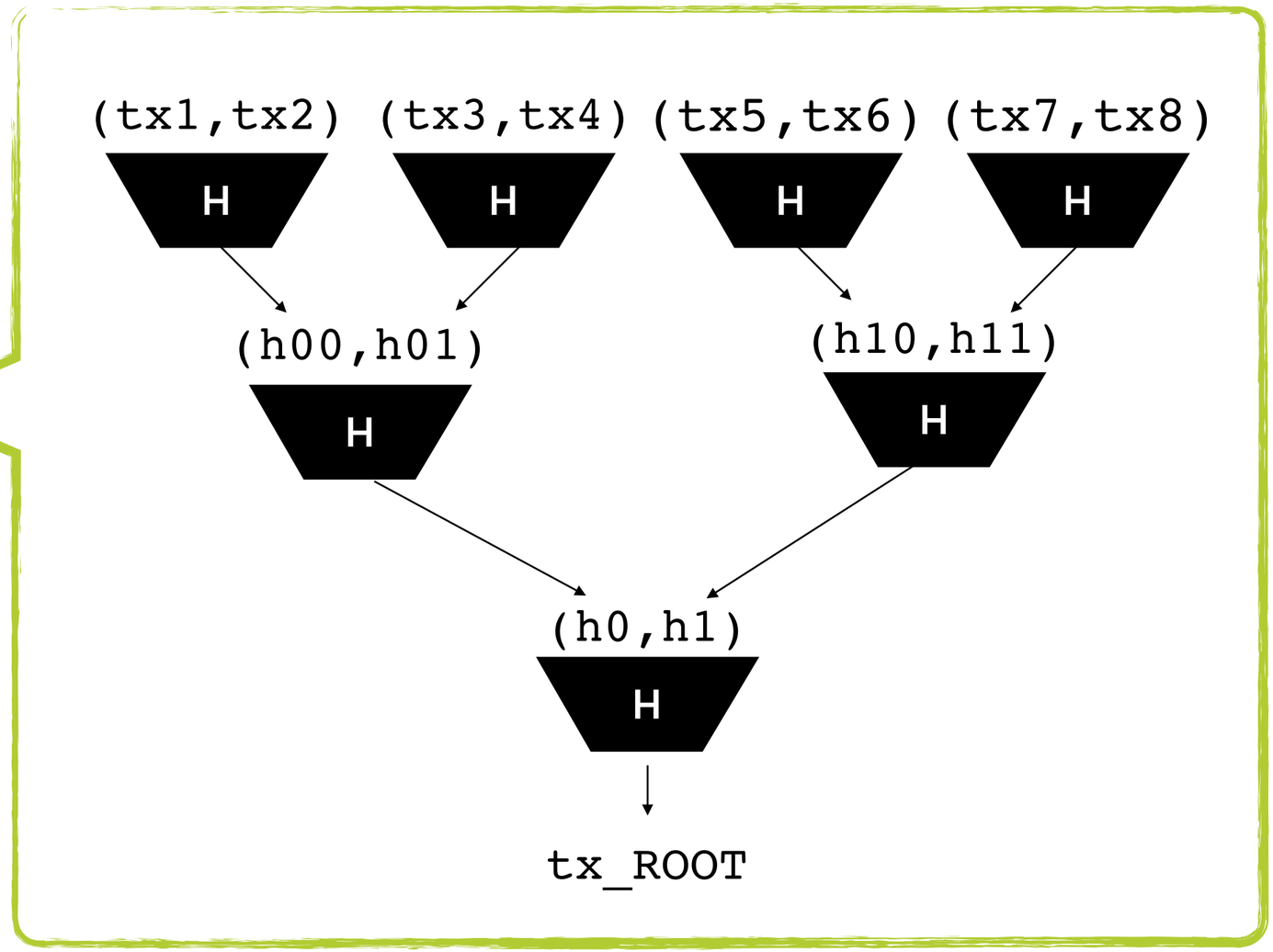
Mining Blocks (miner = first one to solve a hash puzzle on the longest chain)



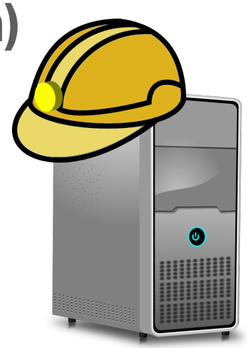
header

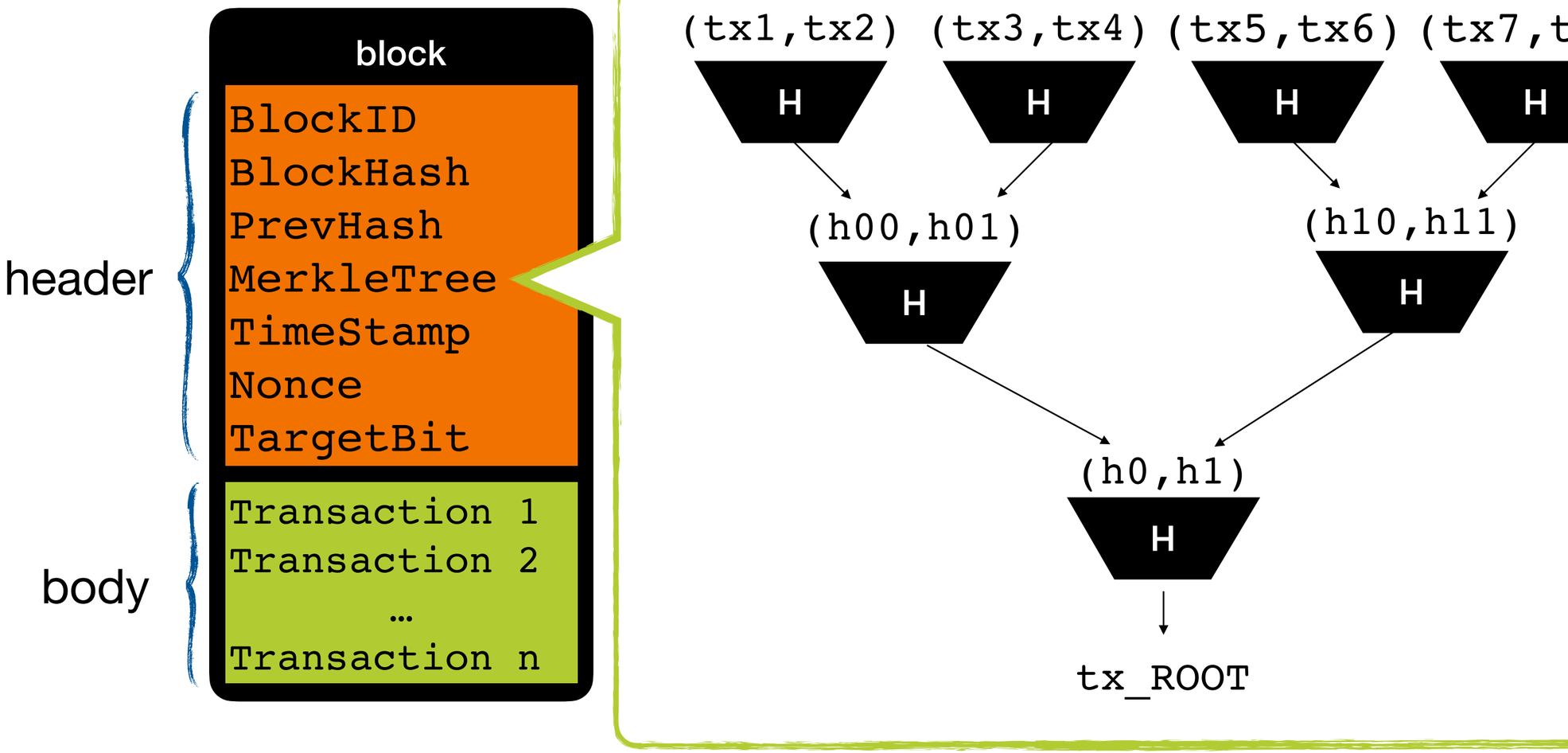
body





Mining Blocks (miner = first one to solve a hash puzzle on the longest chain)

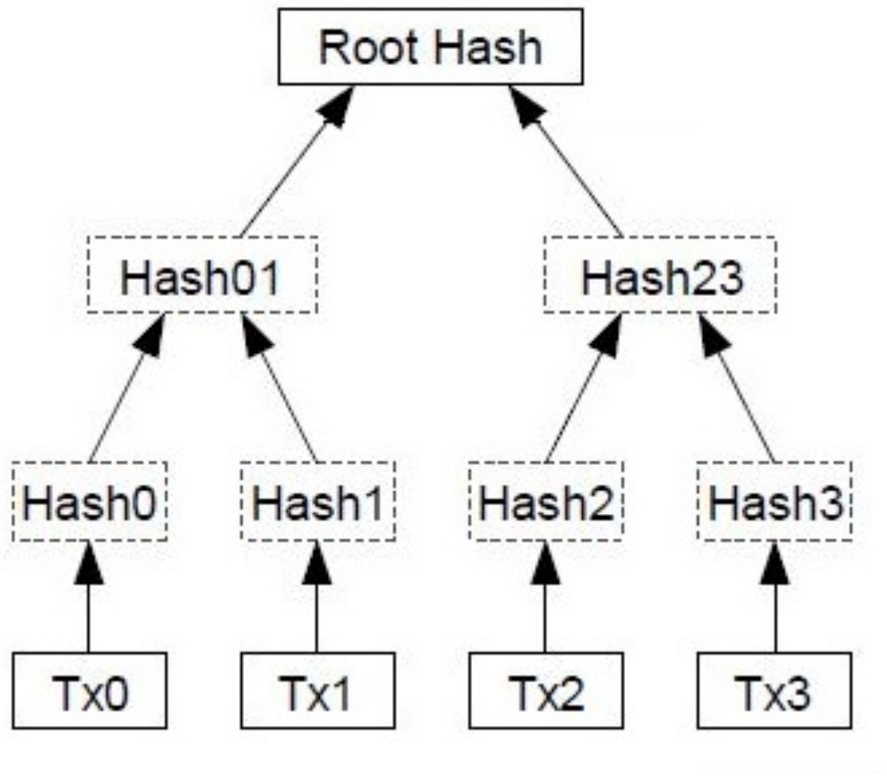




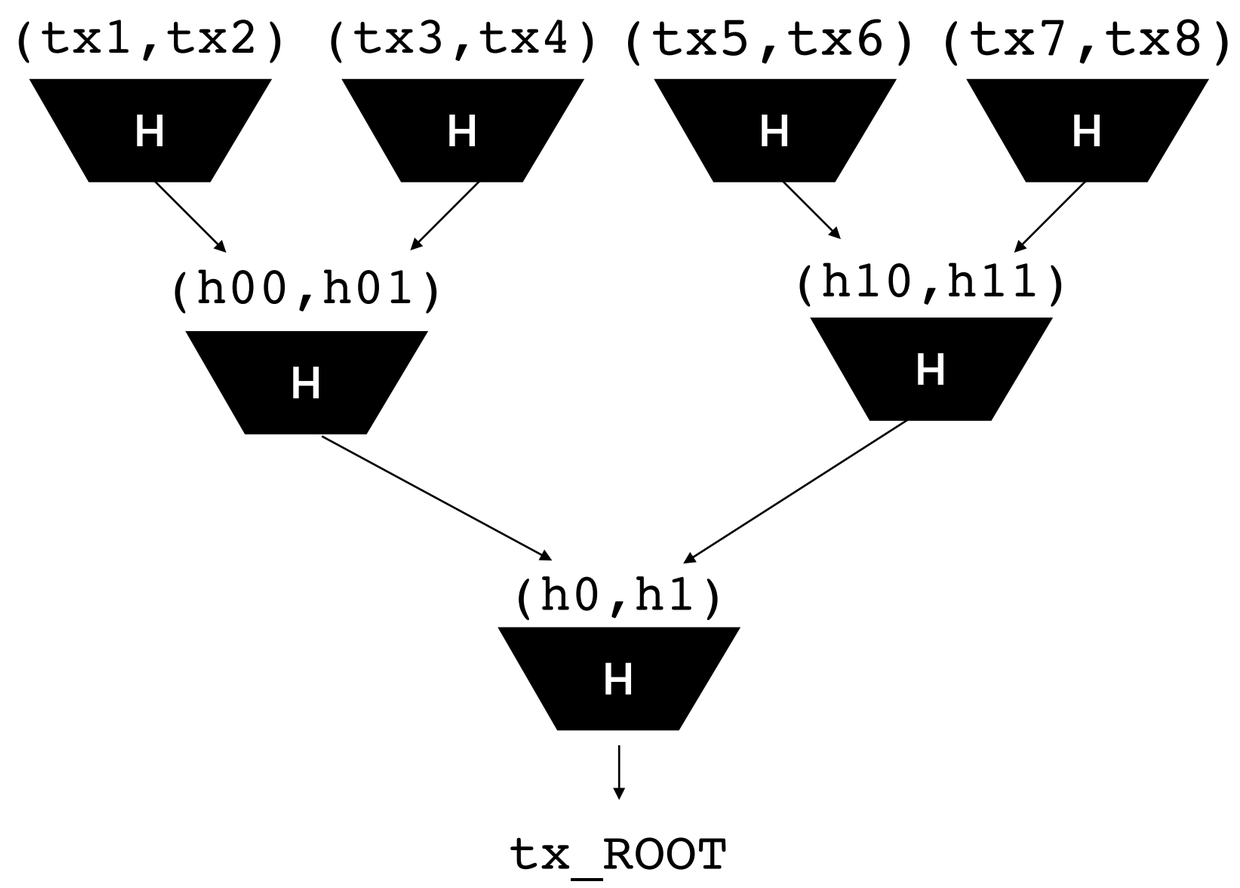
mine = “find `nonce` s.t. `H(PrevHash || tx_ROOT || nonce)` starts with 19 zeroes”

Merkel's Tree

Merkle Tree

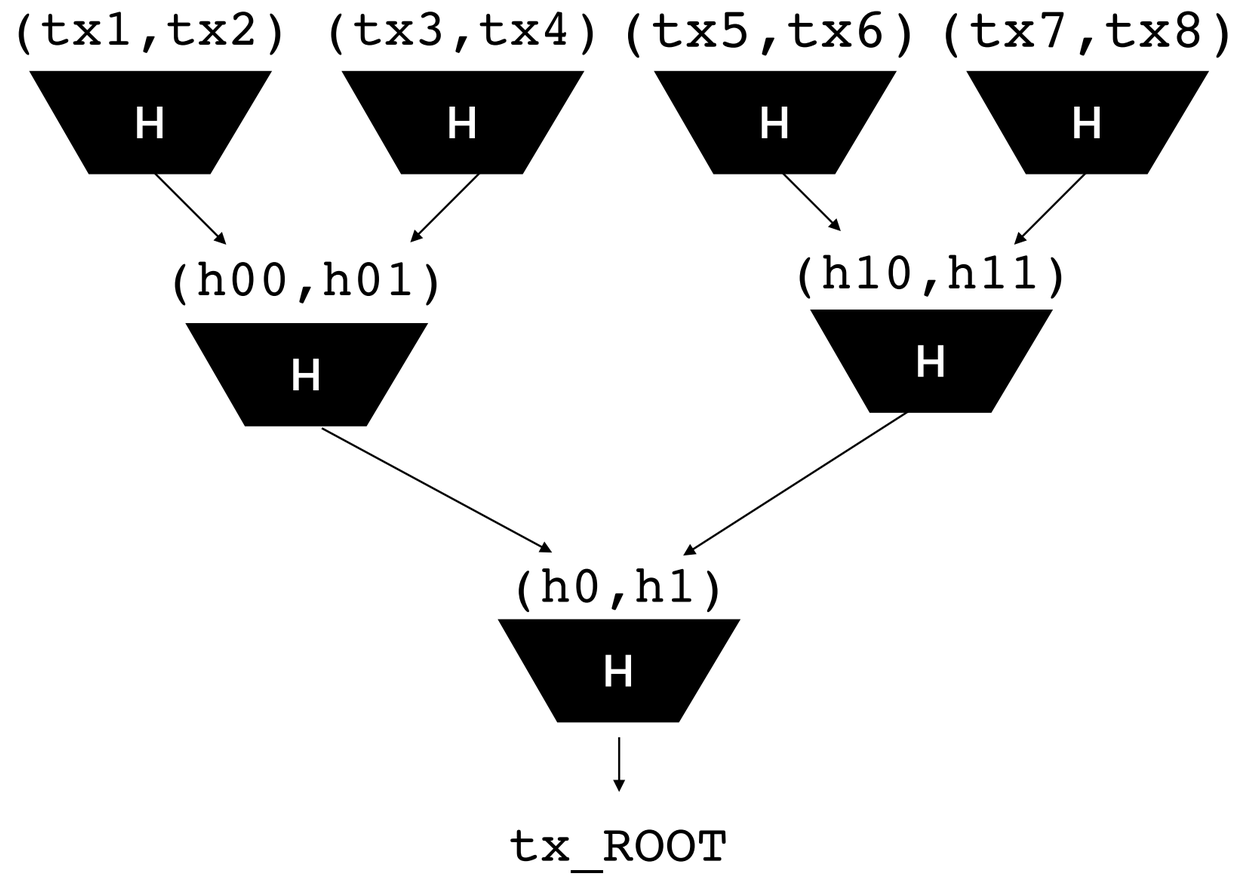


Why Merkle Trees?



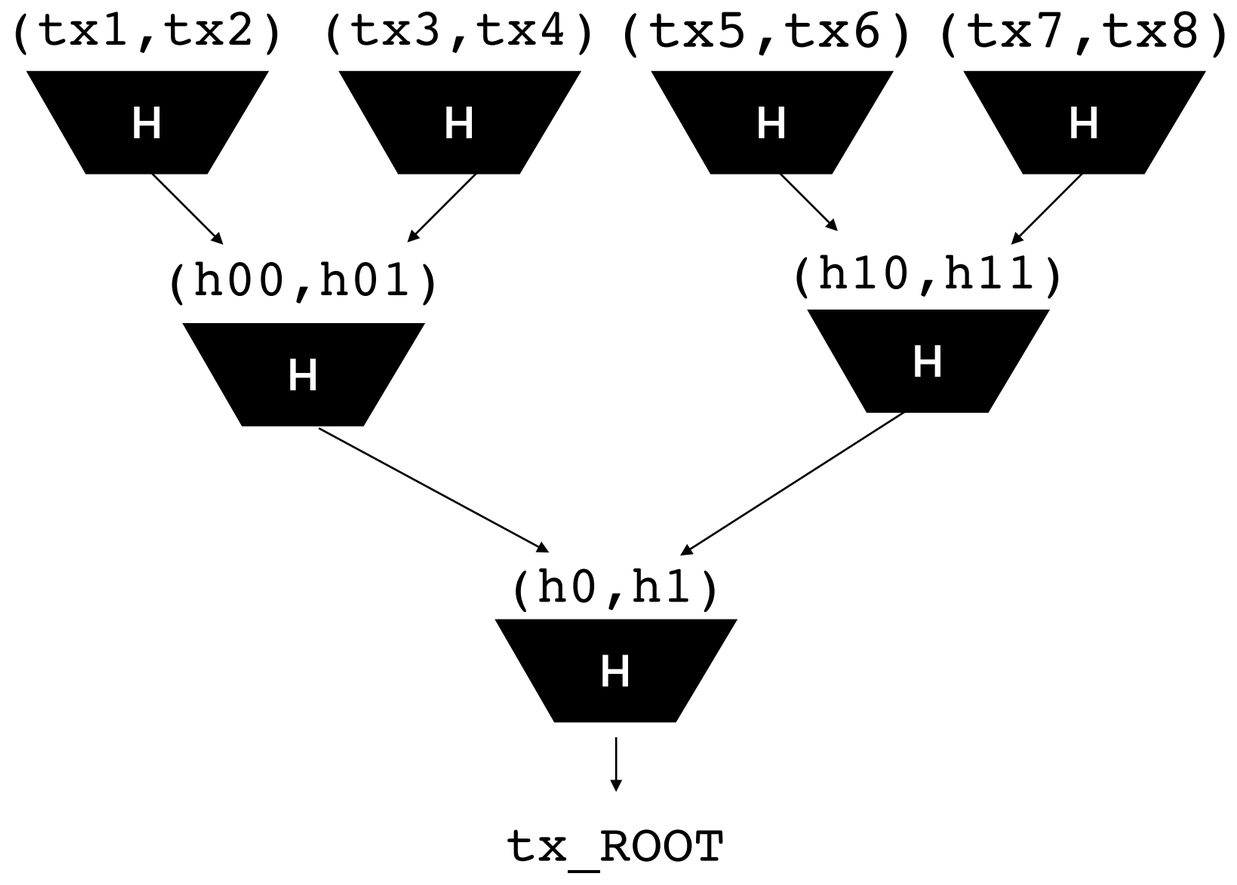
Why Merkle Trees?

1) The root is always of the same size



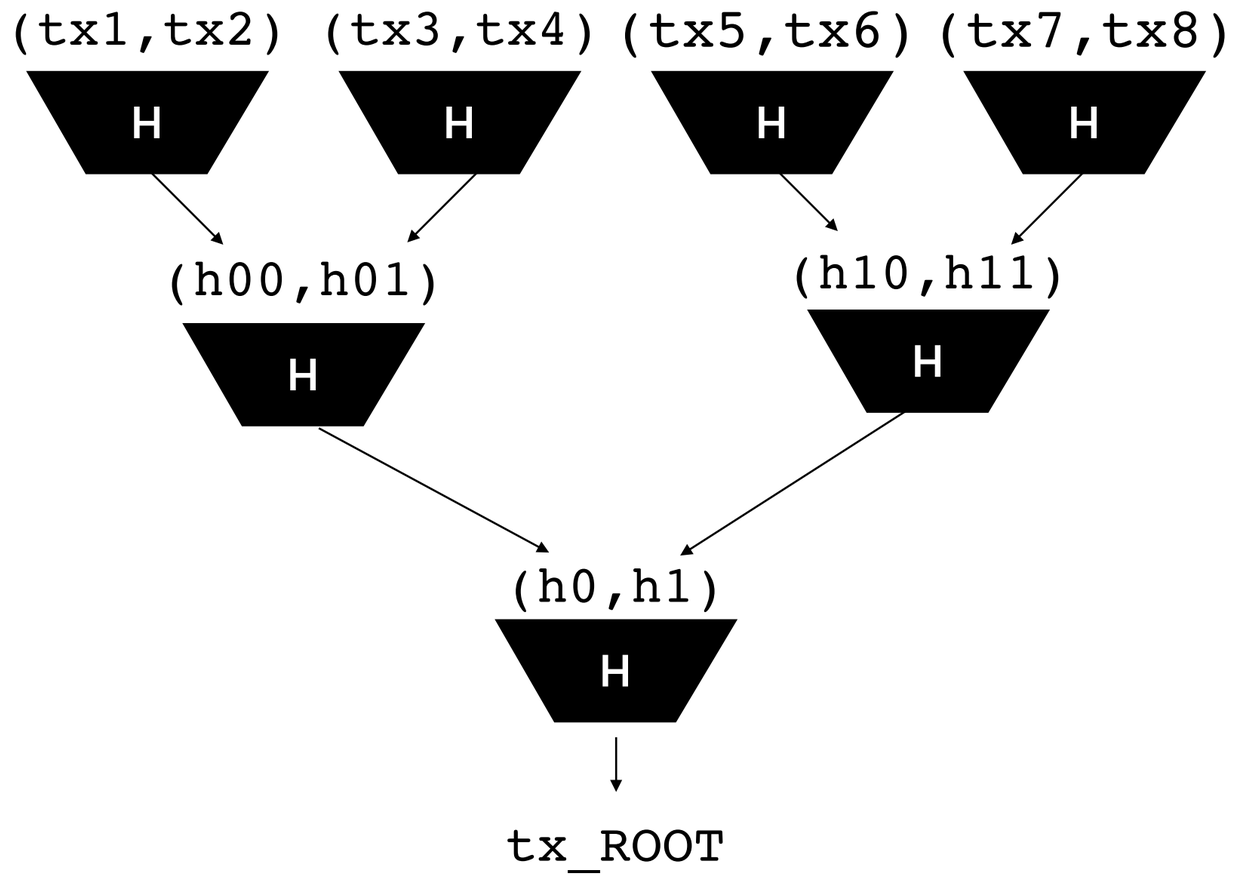
Why Merkle Trees?

1) The root is always of the same size
2) It is easy to transmit tx_ROOT for pooled mining



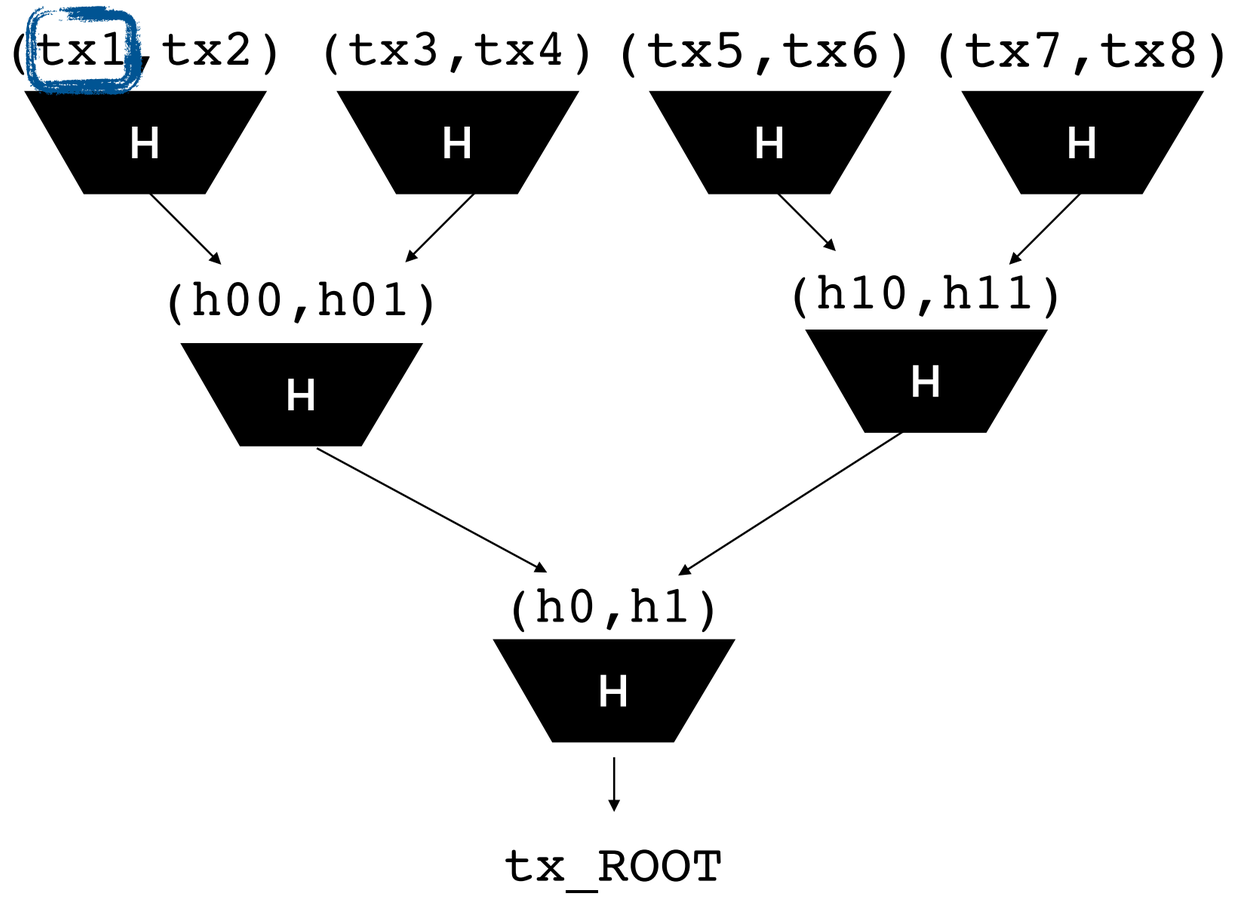
Why Merkle Trees?

1) The root is always of the same size
2) It is easy to transmit tx_ROOT for pooled mining
3) It is easy to prove that a transaction is in a block (short proofs)



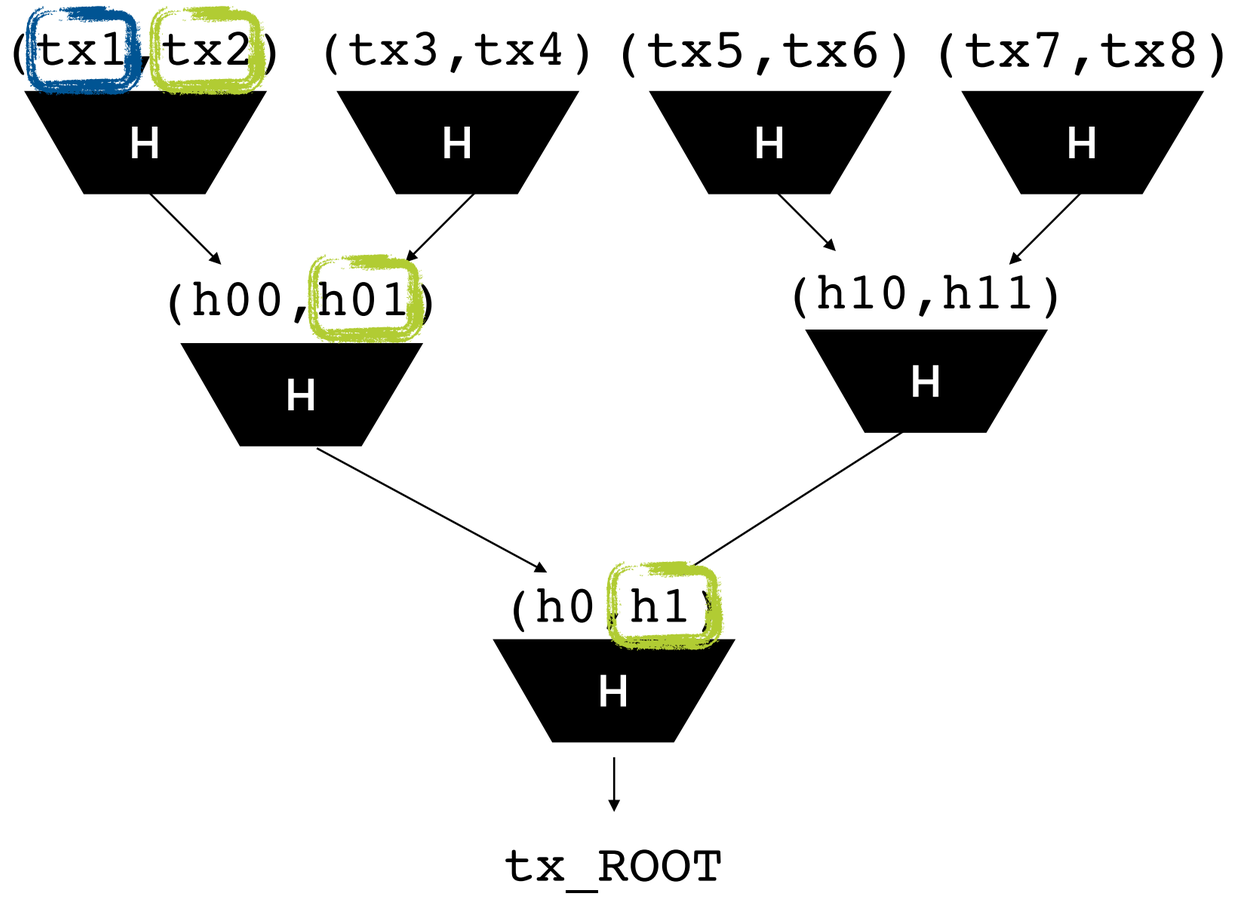
Why Merkle Trees?

1) The root is always of the same size
2) It is easy to transmit tx_ROOT for pooled mining
3) It is easy to prove that a transaction is in a block (short proofs)



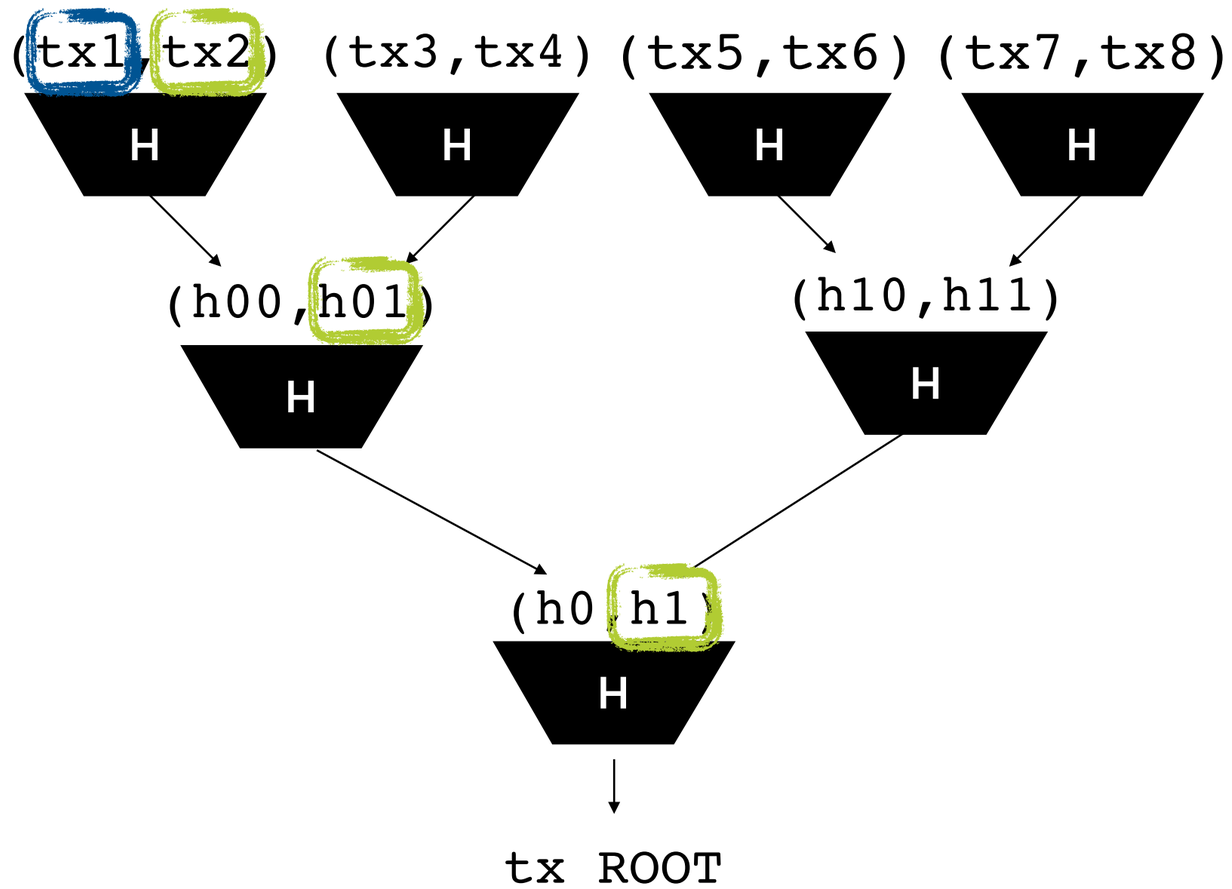
Why Merkle Trees?

1) The root is always of the same size
2) It is easy to transmit tx_ROOT for pooled mining
3) It is easy to prove that a transaction is in a block (short proofs)



Why Merkle Trees?

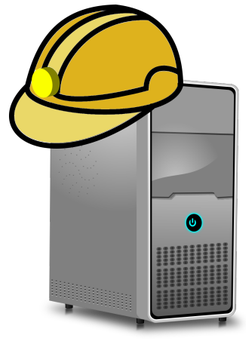
1) The root is always of the same size
2) It is easy to transmit tx_ROOT for pooled mining
3) It is easy to prove that a transaction is in a block (short proofs)



π = (tx1,tx2,h01,h1)

π proves that `tx1` is in the block this can be verified fast without the need to process the entire block

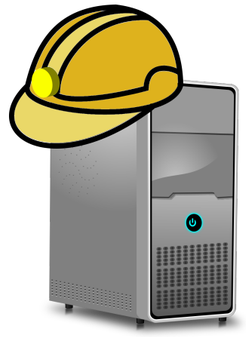
Block Reward System (in Bitcoin)



Main property we want to implement = record keeping

1) the past is immutable
2) everyone agrees on the history
3) all transactions on chain are valid

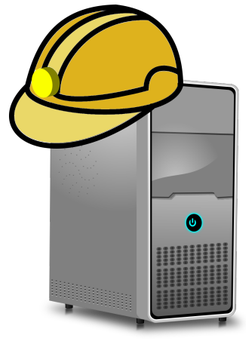
Block Reward System (in Bitcoin)



Main property we want to implement = record keeping

1) the past is immutable ✓
2) everyone agrees on the history ✓
3) all transactions on chain are valid

Block Reward System (in Bitcoin)



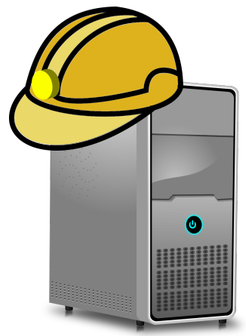
Main property we want to implement = record keeping

1) the past is immutable ✓
2) everyone agrees on the history ✓
3) all transactions on chain are valid ⇦

miners check validity of the chain,

block header and body (transactions)

Block Reward System (in Bitcoin)



Main property we want to implement = record keeping

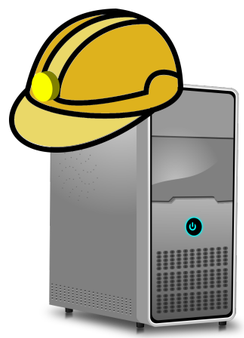
1) the past is immutable ✓
2) everyone agrees on the history ✓
3) all transactions on chain are valid ⇦

miners check validity of the chain, block header and body (transactions)

⇨ *INCENTIVE : the first transaction in a block is towards the miner!*

+ transaction fees

Block Reward System (in Bitcoin)



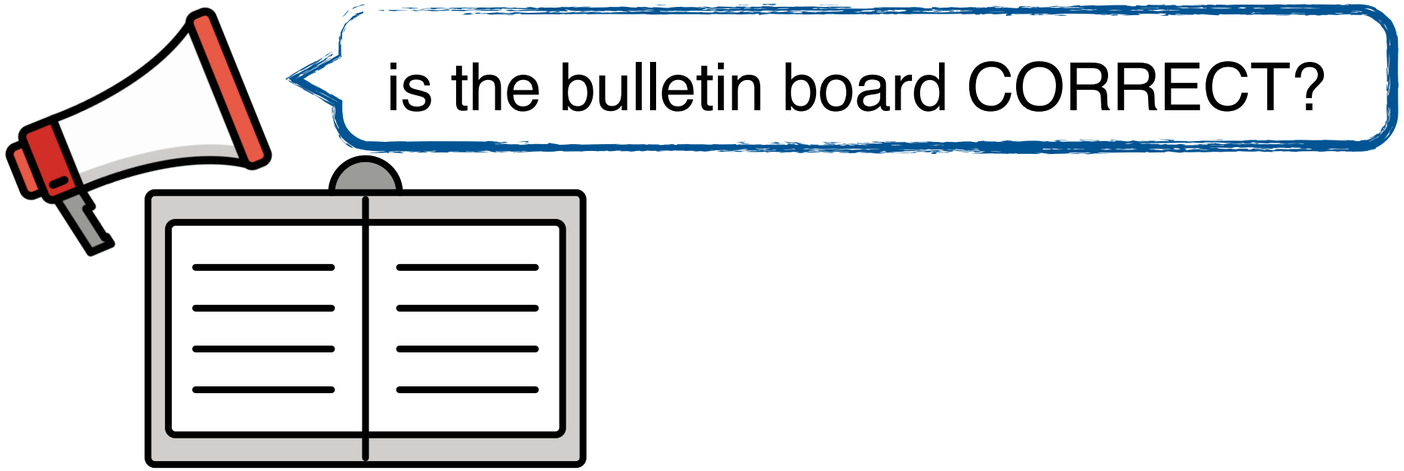
Main property we want to implement = record keeping

1) the past is immutable ✓
2) everyone agrees on the history ✓
3) all transactions on chain are valid ⇦

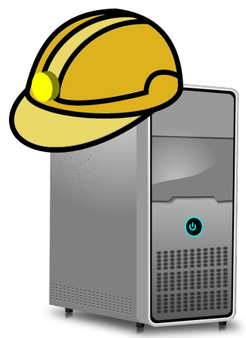
miners check validity of the chain, block header and body (transactions)

⇨ *INCENTIVE : the first transaction in a block is towards the miner!*

+ transaction fees



Block Reward System (in Bitcoin)



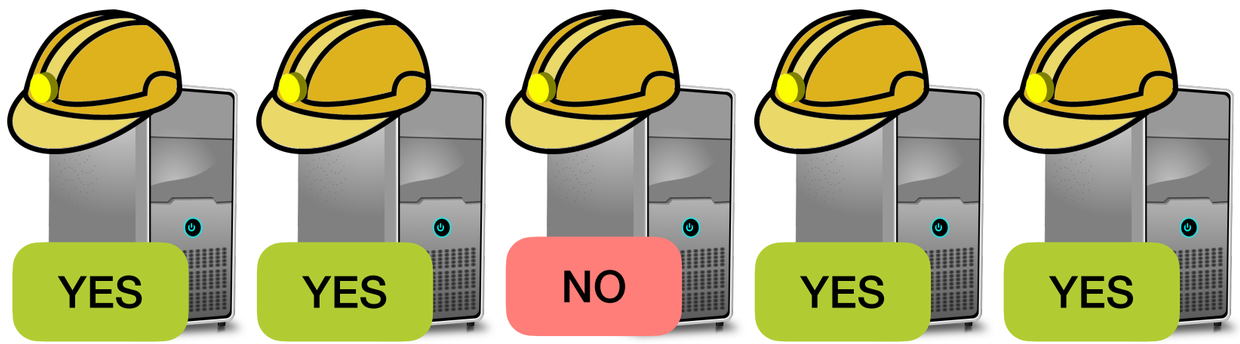
Main property we want to implement = record keeping

1) the past is immutable ✓
2) everyone agrees on the history ✓
3) all transactions on chain are valid ⇦

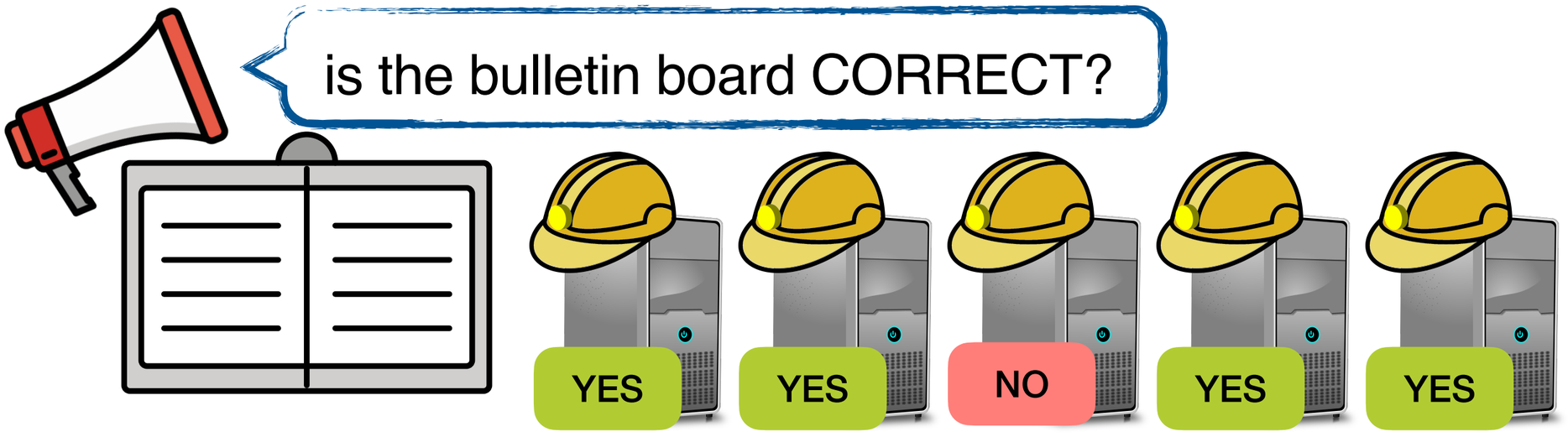
miners check validity of the chain, block header and body (transactions)

⇨ *INCENTIVE : the first transaction in a block is towards the miner!*

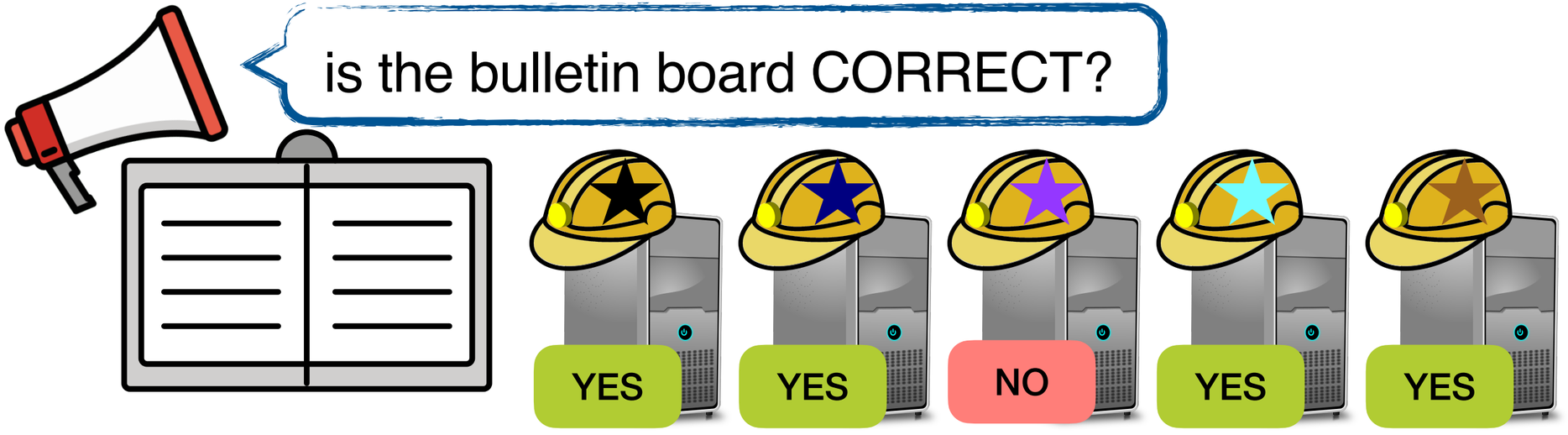
+ transaction fees



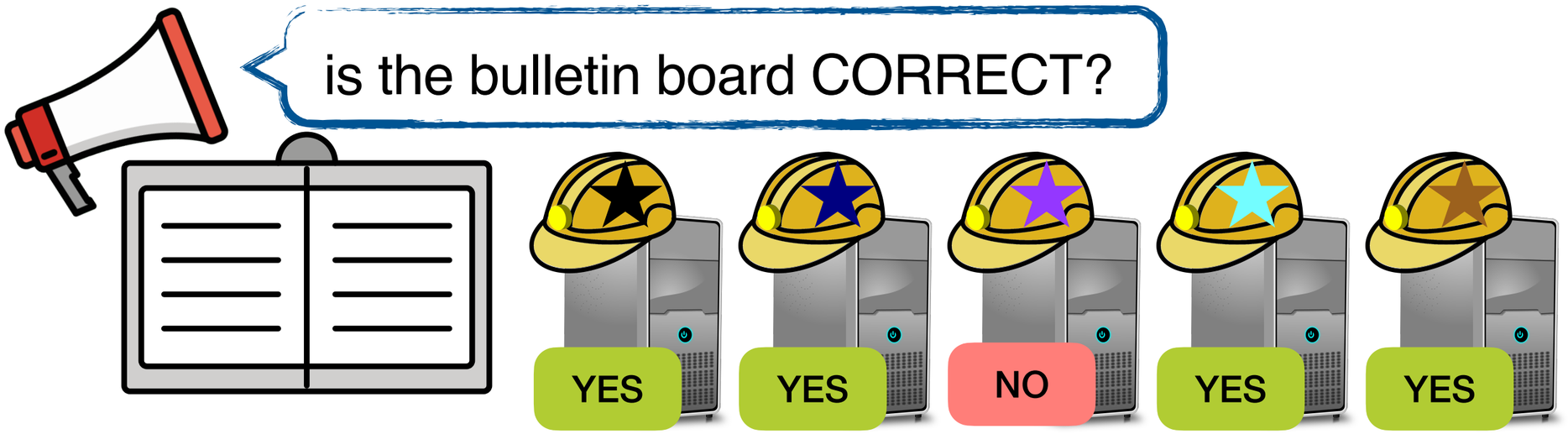
Consensus in Bitcoin



Consensus in Bitcoin



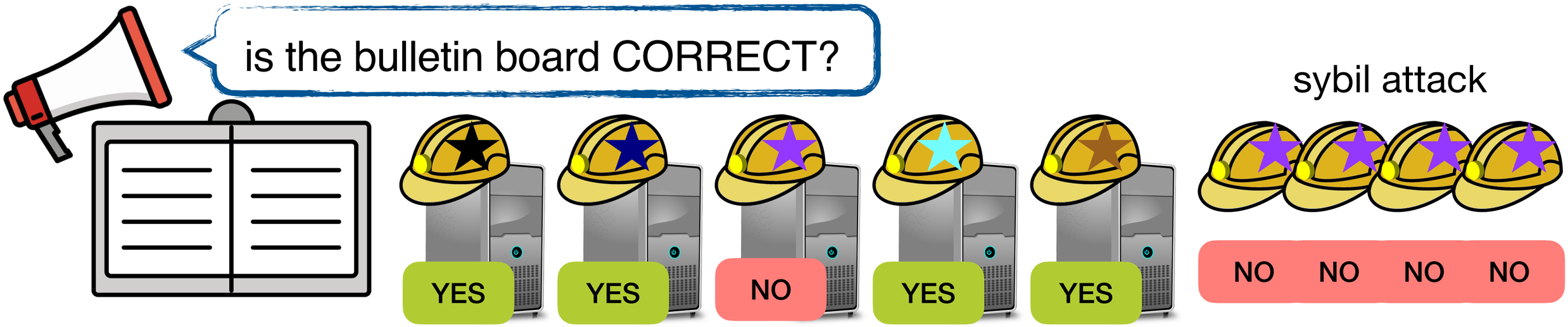
Consensus in Bitcoin



miners vote for the correct chain by mining along it

(longest chain ⇒ highest amount of votes = consensus)

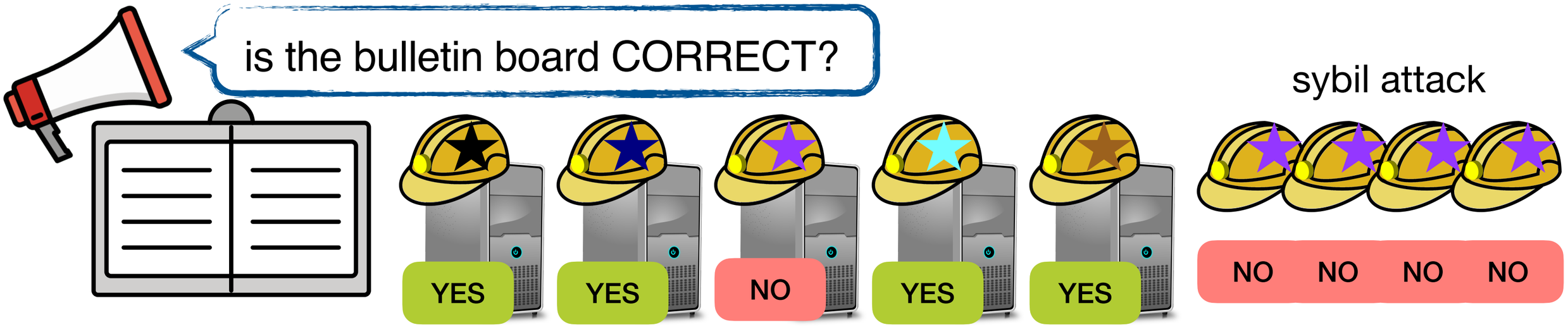
Consensus in Bitcoin



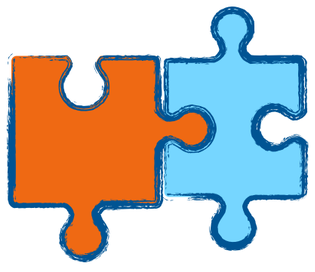
miners vote for the correct chain by mining along it

(longest chain ⇒ highest amount of votes = consensus)

Consensus in Bitcoin



miners vote for the correct chain by mining along it
(longest chain ⇒ highest amount of votes = consensus)



PoW prevents sybil attacks
(creating multiple entities does not create additional computational power)

Lecture Agenda

How To Set Up a Bulletin Board

- Hash Functions
- Proof of Work (Cryptographic Puzzles)
- Example: Bitcoin

How To Generate Money

- Mining Blocks
- Merkle Trees
- Block Reword System

How To Circulate Money

- Transactions
- ECDSA

Some Problems With Blockchains

- Freshness of the Genesis Block
- 51% Attack
- Proof of Work VS Proof of Stake

Advanced Tools

- Proofs of Knowledge
- zkSNARKs
- MPC

Lecture Agenda

How To Set Up a Bulletin Board

- Hash Functions
- Proof of Work (Cryptographic Puzzles)
- Example: Bitcoin

How To Generate Money

- Mining Blocks
- Merkle Trees
- Block Reword System

How To Circulate Money

- Transactions
- ECDSA

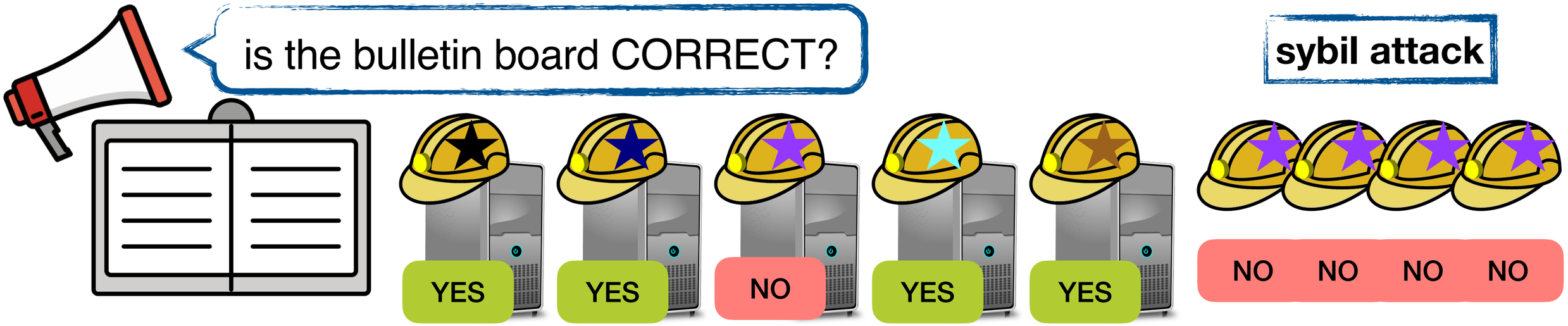
Some Problems With Blockchains

- Freshness of the Genesis Block
- 51% Attack
- Proof of Work VS Proof of Stake

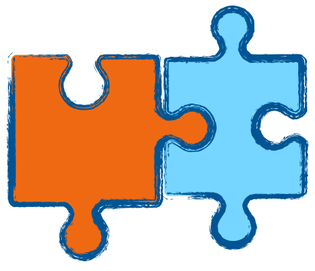
Advanced Tools

- Proofs of Knowledge
- zkSNARKs
- MPC

Consensus in Bitcoin

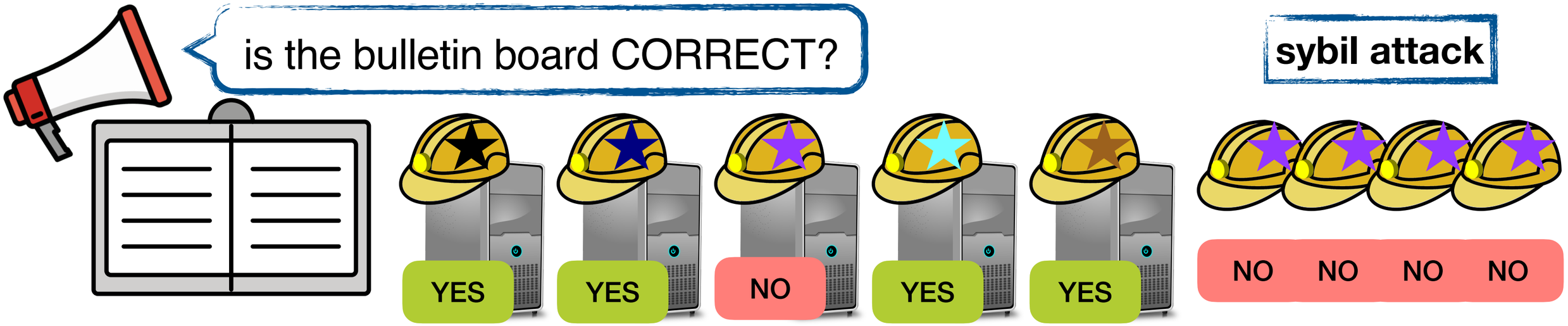


miners vote for the correct chain by mining along it
(longest chain ⇒ highest amount of votes = consensus)

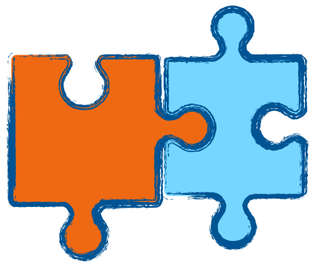


PoW prevents sybil attacks
(creating multiple entities does not create additional computational power)

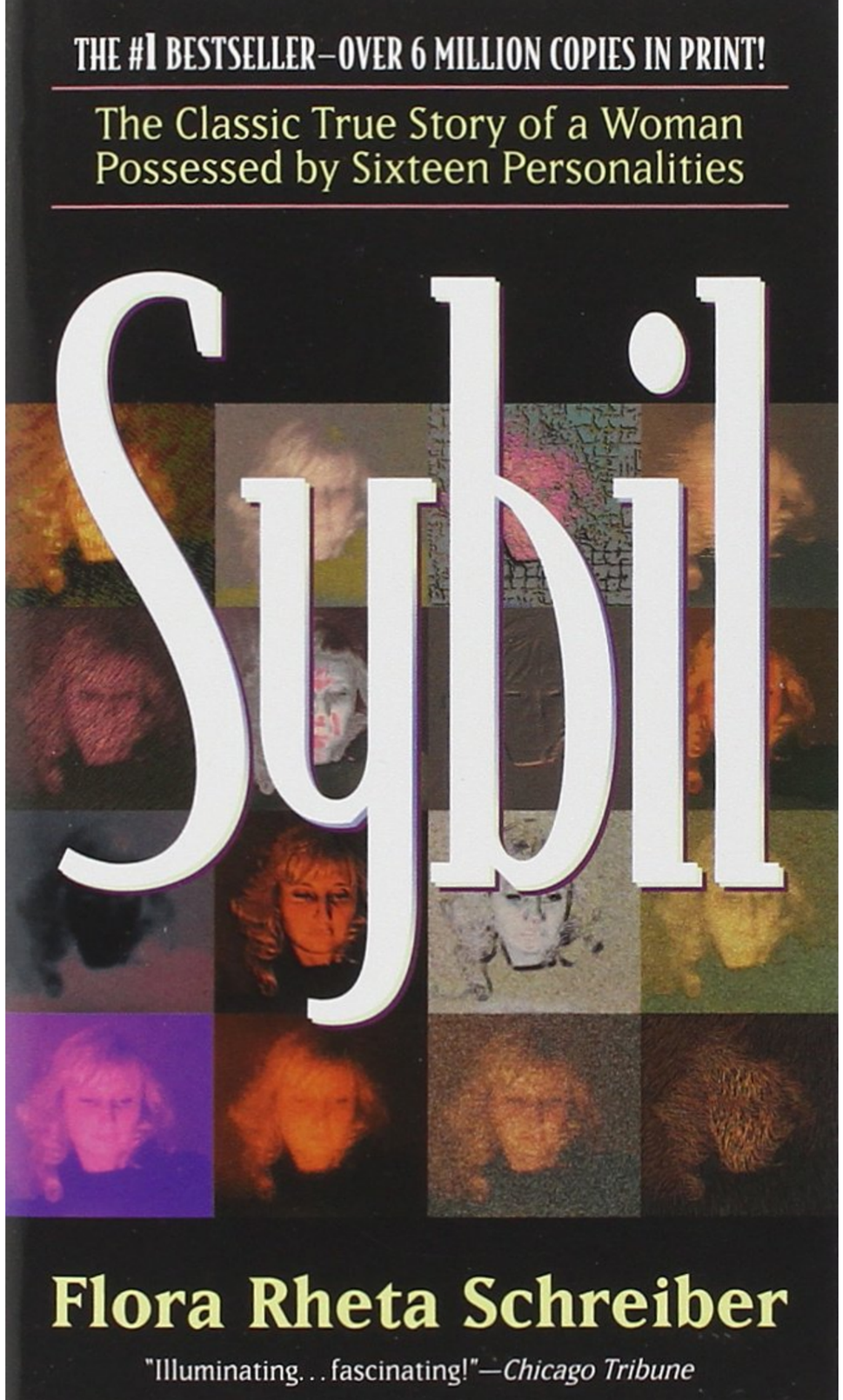
Consensus in Bitcoin



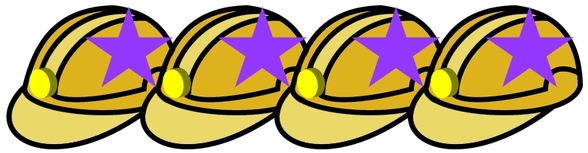
miners vote for the correct chain by mining along it
(longest chain ⇒ highest amount of votes = consensus)



PoW prevents sybil attacks
(creating multiple entities does not create additional computational power)



sybil attack



NO NO NO NO

Lecture Agenda

How To Set Up a Bulletin Board

- Hash Functions
- Proof of Work (Cryptographic Puzzles)
- Example: Bitcoin

How To Generate Money

- Mining Blocks
- Merkle Trees
- Block Reword System

How To Circulate Money

- Transactions
- ECDSA

Some Problems With Blockchains

- Freshness of the Genesis Block
- 51% Attack
- Proof of Work VS Proof of Stake

Advanced Tools

- Proofs of Knowledge
- zkSNARKs
- MPC

ECDSA - Algorithms

```
KeyGen (sec.par) ⇨ (sk, pk)
  d ←$— [0 ... n-1]
  sk = d
  pk = Q = d*G
```

```
Sign (sk, msg) ⇨ sgn
  k ←$— [0 ... n-1]
  R = k*G
  r = R_x mod n
  z = sha256(msg)
  s = inv(k)·(z + d·r) mod n
  sgn = (r, s)
```

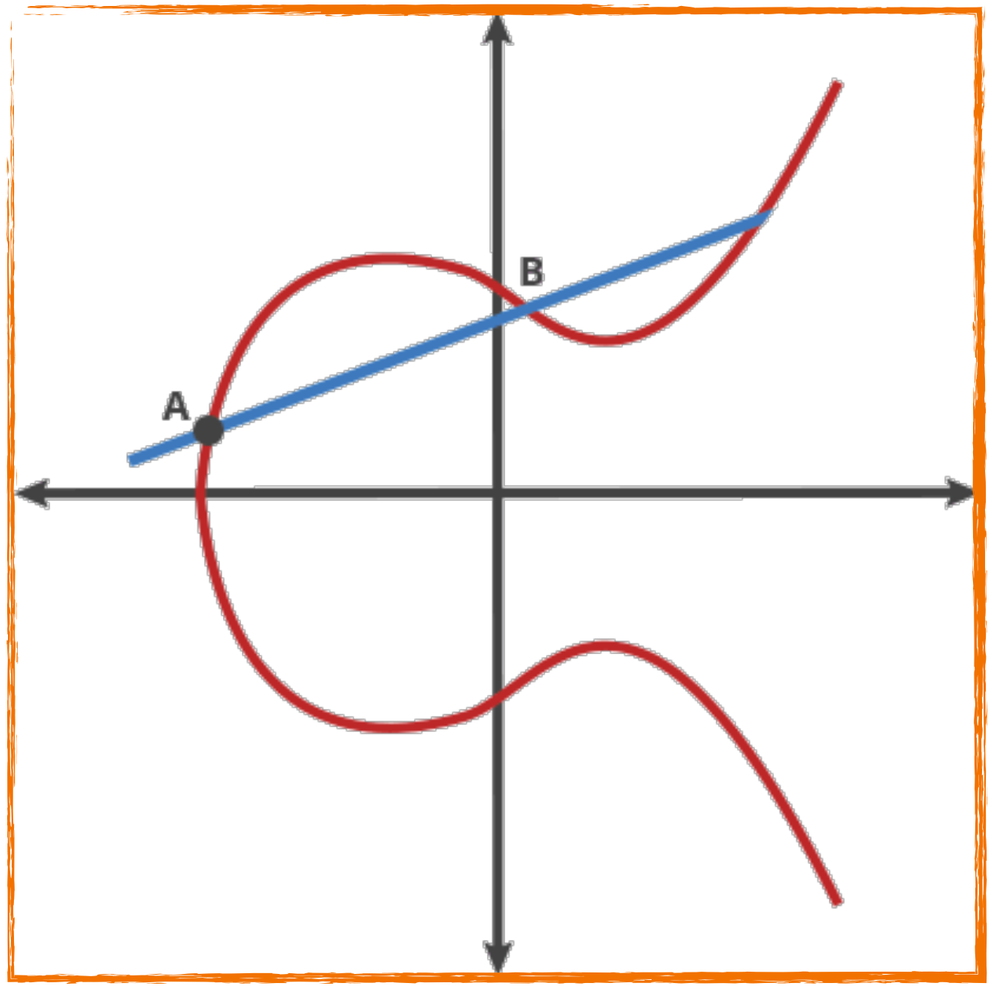
```
Verify(pk, msg, sgn) ⇨ {0, 1}
  z = sha256(msg)
  T = [z·inv(s) mod n]*G
  P = [inv(s)·r mod n]*Q
  if R == T+P return 1
  else return 0
```

ECDSA - Algorithms

```
KeyGen (sec.par) ⇨ (sk, pk)
  d ←$— [0 ... n-1]
  sk = d
  pk = Q = d*G
```

```
Sign (sk, msg) ⇨ sgn
  k ←$— [0 ... n-1]
  R = k*G
  r = R_x mod n
  z = sha256(msg)
  s = inv(k)·(z + d·r) mod n
  sgn = (r, s)
```

```
Verify(pk, msg, sgn) ⇨ {0, 1}
  z = sha256(msg)
  T = [z·inv(s) mod n]*G
  P = [inv(s)·r mod n]*Q
  if R == T+P return 1
  else return 0
```



ECDSA - the Good

- ★ Shorter keys and better security than the RSA signature scheme
- ★ Non-malleable
- ★ IoT friendly
- ★ In wide adoption (TLS, DigiCert (Symantec), Sectigo (Comodo) …)

ECDSA - the Bad

PS3 hacked through poor cryptography implementation

A group of hackers named fail0verflow revealed in a presentation how they ...

CASEY JOHNSTON - 12/30/2010, 6:25 PM

{* SECURITY *}

Android bug batters Bitcoin wallets

Old flaw, new problem

Richard Chirgwin Mon 12 Aug 2013 // 00:43 UTC

LadderLeak: Side-channel security flaws exploited to break ECDSA cryptography

Charlie Osborne 28 May 2020 at 14:07 UTC
Updated: 28 June 2021 at 09:05 UTC

ECDSA - the Bad

PS3 hacked through poor cryptography implementation

A group of hackers named fail0verflow revealed in a presentation how they ...

CASEY JOHNSTON - 12/30/2010, 6:25 PM

{* SECURITY *}

Android bug batters Bitcoin wallets

Old flaw, new problem

repeated nonce attack

Richard Chirgwin

Mon 12 Aug 2013 // 00:43 UTC

LadderLeak: Side-channel security flaws exploited to break ECDSA cryptography

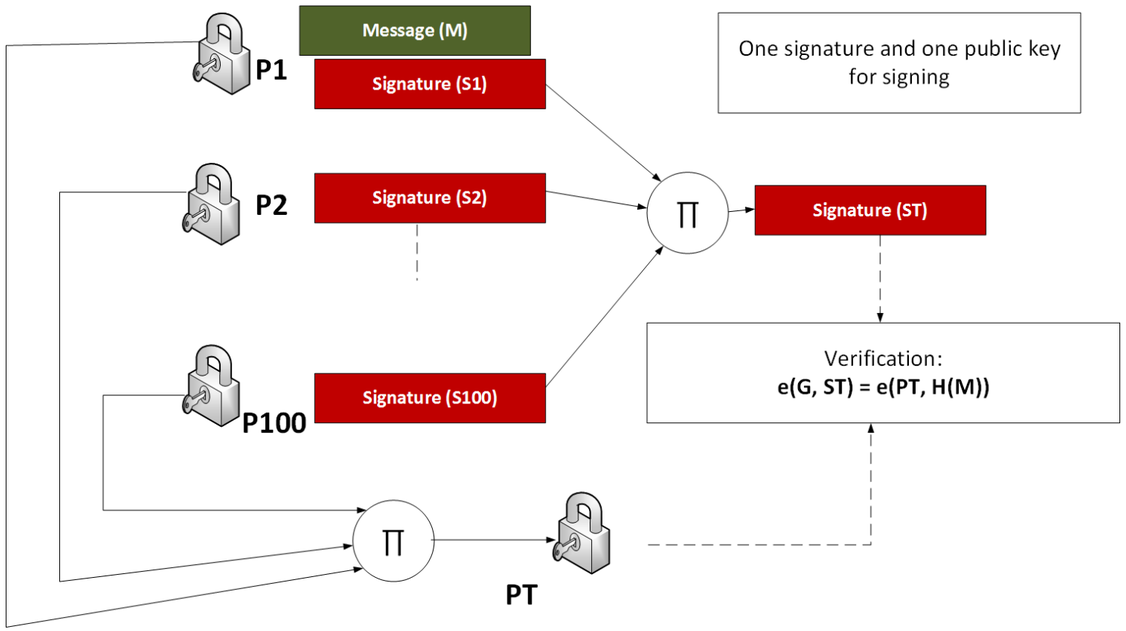
Charlie Osborne 28 May 2020 at 14:07 UTC

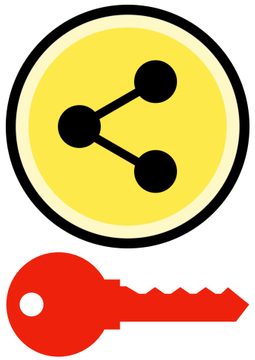
Updated: 28 June 2021 at 09:05 UTC

ECDSA - What's Next?

ECDSA - What's Next?

Threshold Signatures



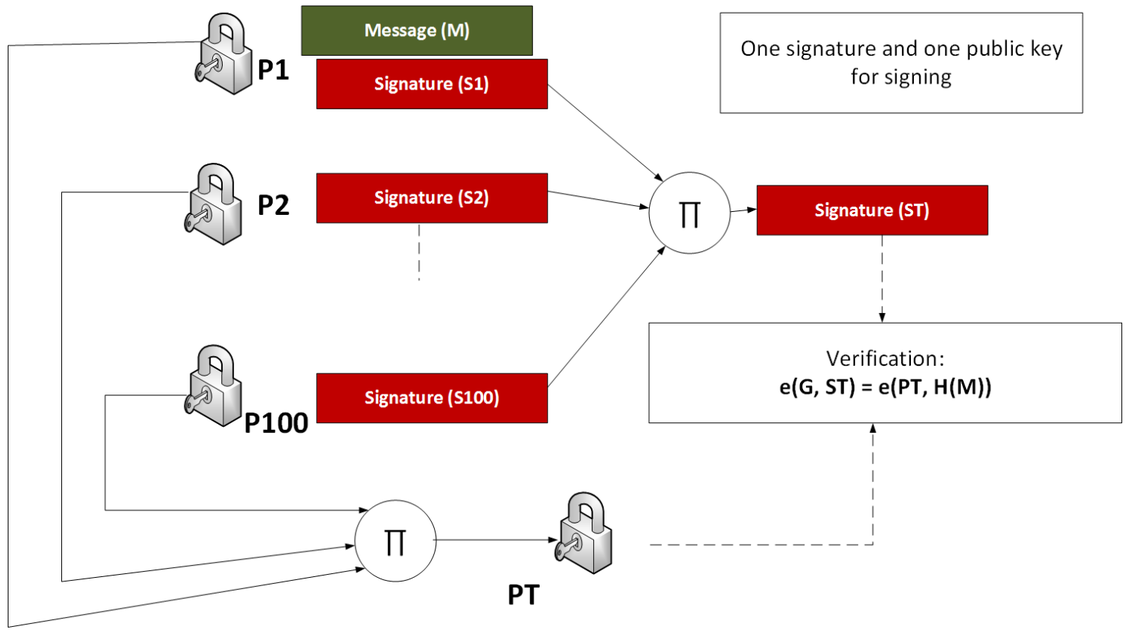


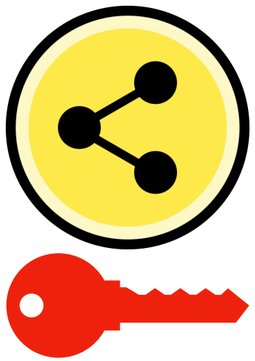
LBS signature

Schnorr signature

ECDSA - What's Next?

Threshold Signatures





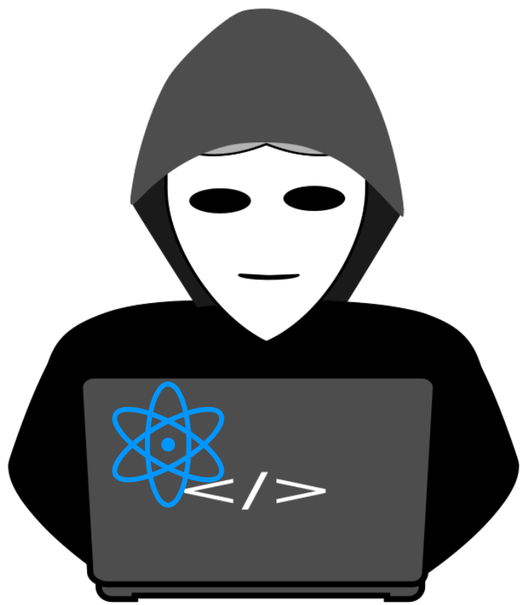
LBS signature

Schnorr signature

Post Quantum Secure Signatures

FALCON

Fast-Fourier Lattice-based Compact Signatures over NTRU



DILITHIUM

Lecture Agenda

How To Set Up a Bulletin Board

- Hash Functions
- Proof of Work (Cryptographic Puzzles)
- Example: Bitcoin

How To Generate Money

- Mining Blocks
- Merkle Trees
- Block Reword System

How To Circulate Money

- Transactions
- ECDSA

Some Problems With Blockchains

- Freshness of the Genesis Block
- 51% Attack
- PoW Vs PoS
- The Byzantine Fault Tolerance Problem

Advanced Tools

- Zero Knowledge Proofs
- Schnorr Interactive PoK
- Fiat-Shamir Heuristic (NIZK)

Lecture Agenda

How To Set Up a Bulletin Board

- Hash Functions
- Proof of Work (Cryptographic Puzzles)
- Example: Bitcoin

How To Generate Money

- Mining Blocks
- Merkle Trees
- Block Reword System

How To Circulate Money

- Transactions
- ECDSA

Some Problems With Blockchains

- Freshness of the Genesis Block
- 51% Attack
- PoW Vs PoS
- The Byzantine Fault Tolerance Problem

Advanced Tools

- Zero Knowledge Proofs
- Schnorr Interactive PoK
- Fiat-Shamir Heuristic (NIZK)

Longest vs Strongest Chain Rule

so far we talked about the longest chain rule but …

bitcoin actually uses the ***strongest*** chain rule

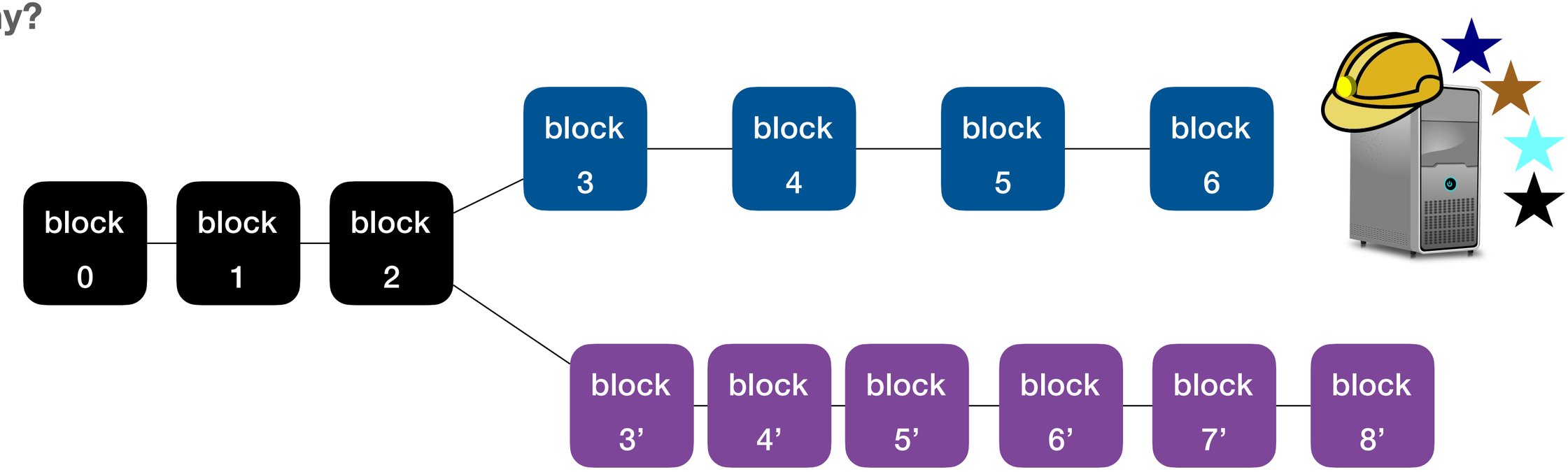
why?

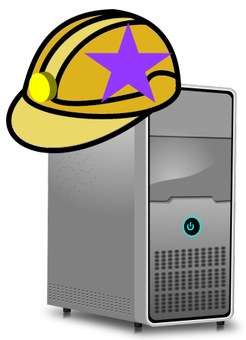
Longest vs Strongest Chain Rule

so far we talked about the longest chain rule but …

bitcoin actually uses the ***strongest*** chain rule

why?





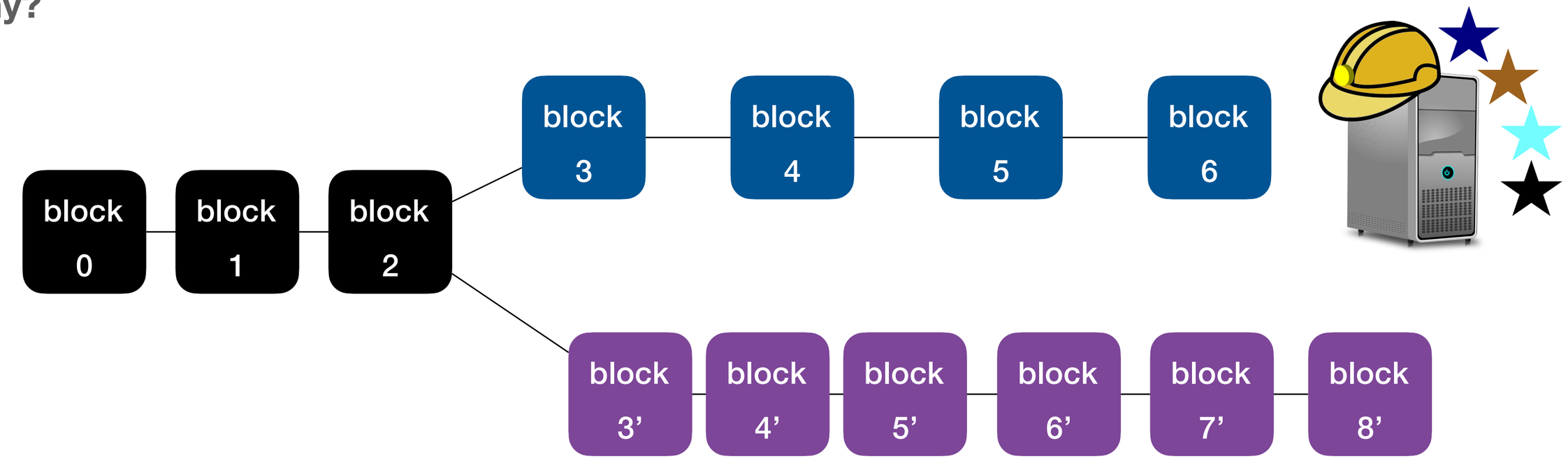
Secretly compute another chain with *fake timestamps* (indicating that it took a long time to produce it)

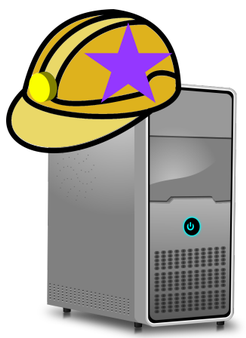
Longest vs Strongest Chain Rule

so far we talked about the longest chain rule but …

bitcoin actually uses the ***strongest*** chain rule

why?





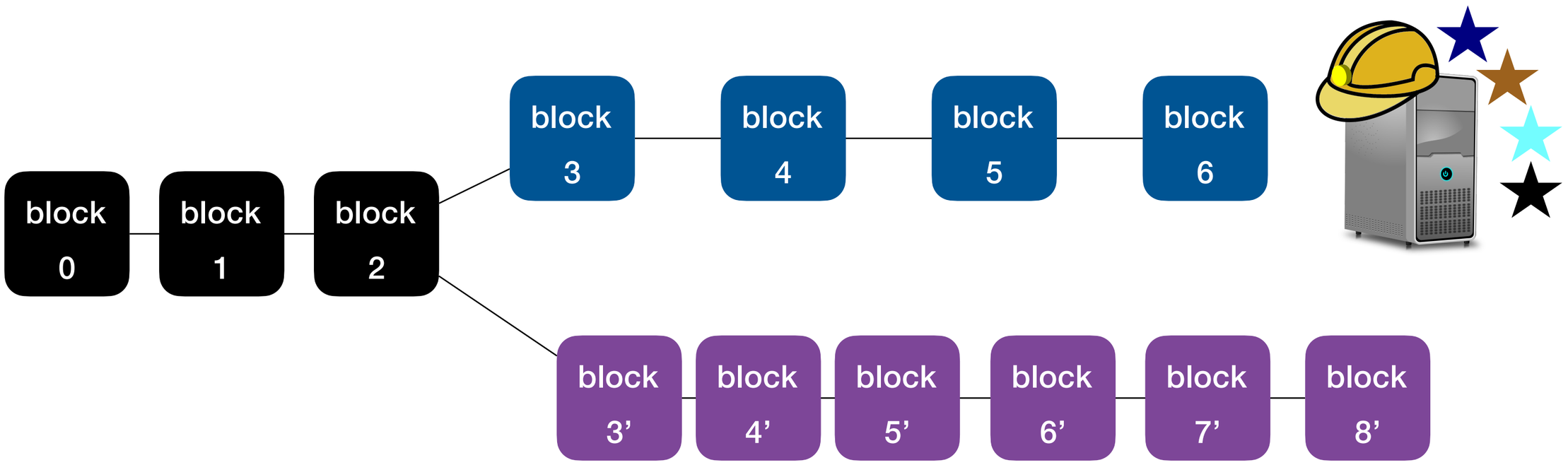
Secretly compute another chain with *fake timestamps* (indicating that it took a long time to produce it)

The difficulty **drops** dramatically, so can quickly produce a chain longer than the valid one and publish it

Longest vs Strongest Chain Rule

so far we talked about the longest chain rule but …

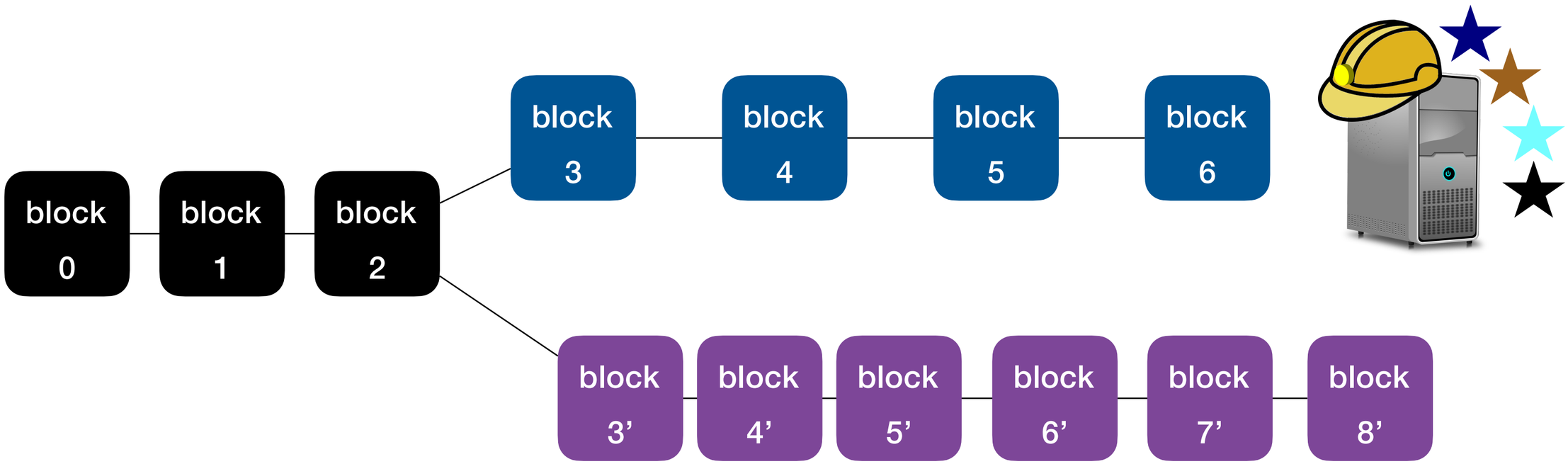
bitcoin actually uses the ***strongest*** chain rule



Longest vs Strongest Chain Rule

so far we talked about the longest chain rule but …

bitcoin actually uses the ***strongest*** chain rule

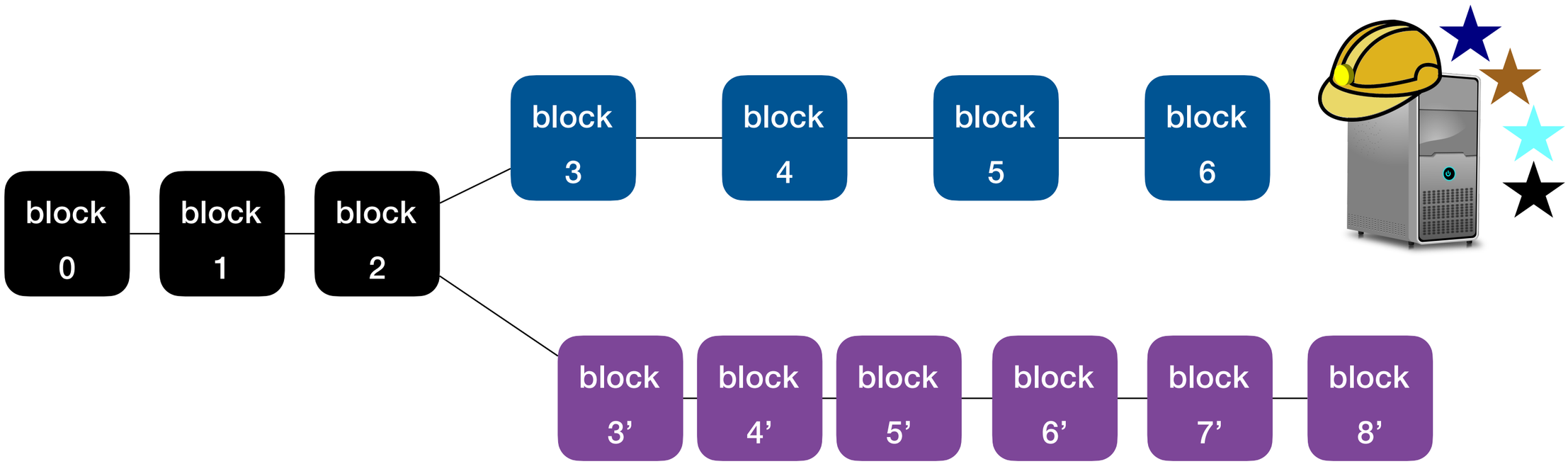


Mitigation:

Longest vs Strongest Chain Rule

so far we talked about the longest chain rule but …

bitcoin actually uses the ***strongest*** chain rule



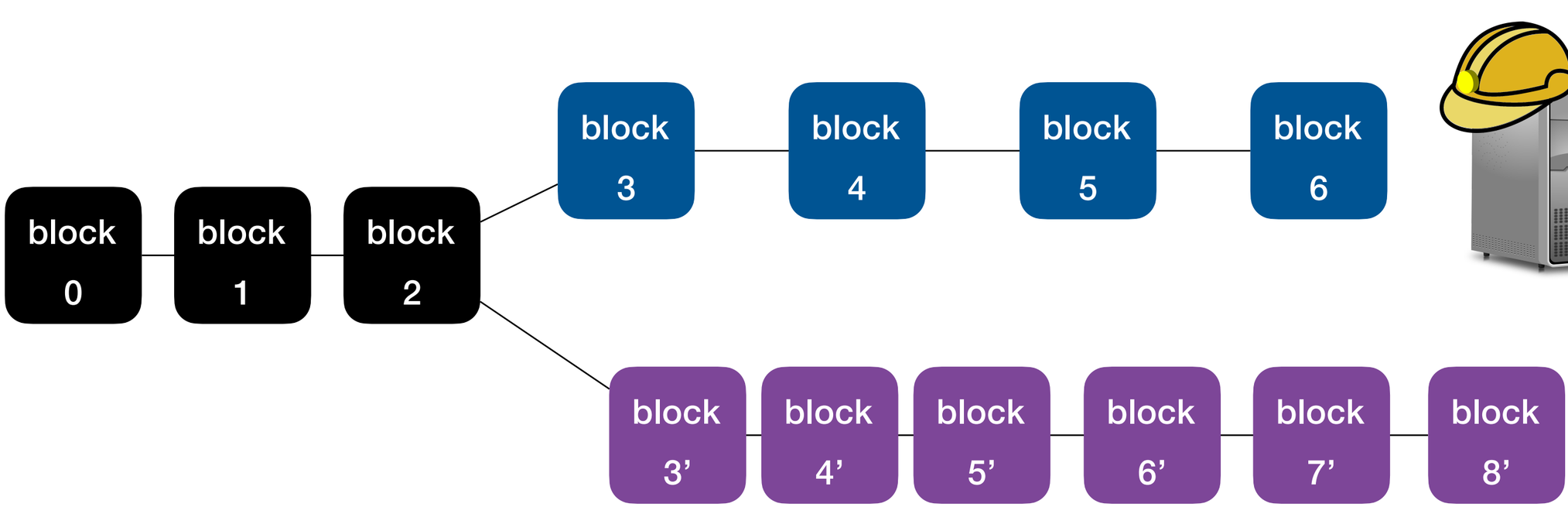
Mitigation:

add **weights** to blocks (proportional to the estimated number number of hashes needed to solve the block difficulty)

Longest vs Strongest Chain Rule

so far we talked about the longest chain rule but …

bitcoin actually uses the ***strongest*** chain rule



Mitigation:

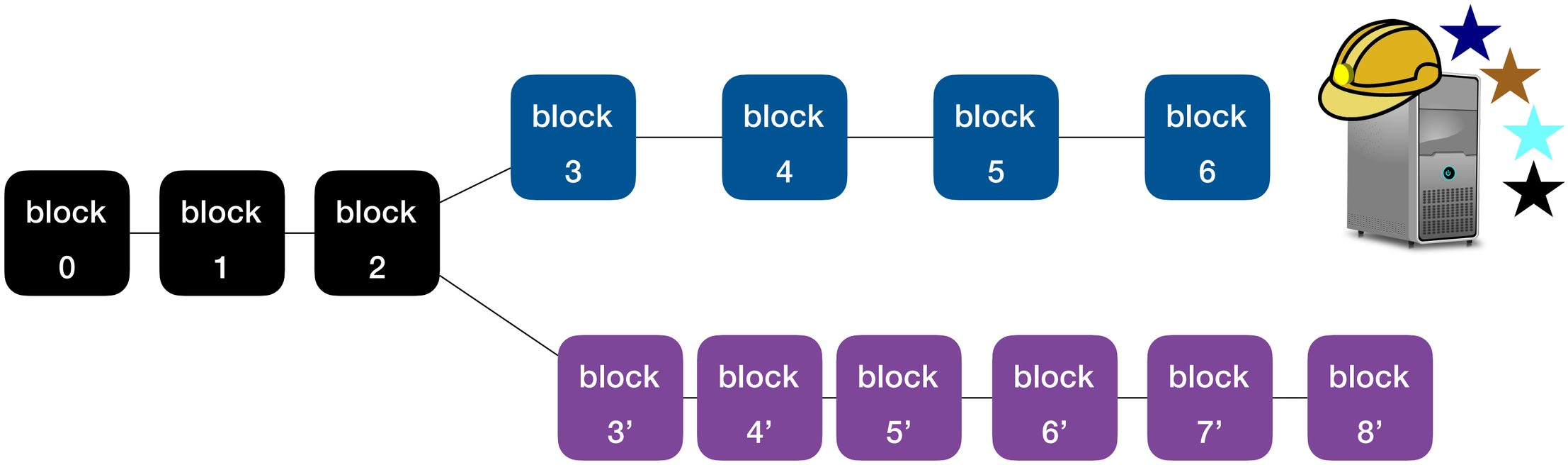
add **weights** to blocks (proportional to the estimated number number of hashes needed to solve the block difficulty)

weak blocks must be generated *quickly* to catch up with the honest, ever growing chain (quick ⇒ few hashes ⇒ light weight)

Longest vs Strongest Chain Rule

so far we talked about the longest chain rule but …

bitcoin actually uses the ***strongest*** chain rule



Mitigation:

add **weights** to blocks (proportional to the estimated number number of hashes needed to solve the block difficulty)

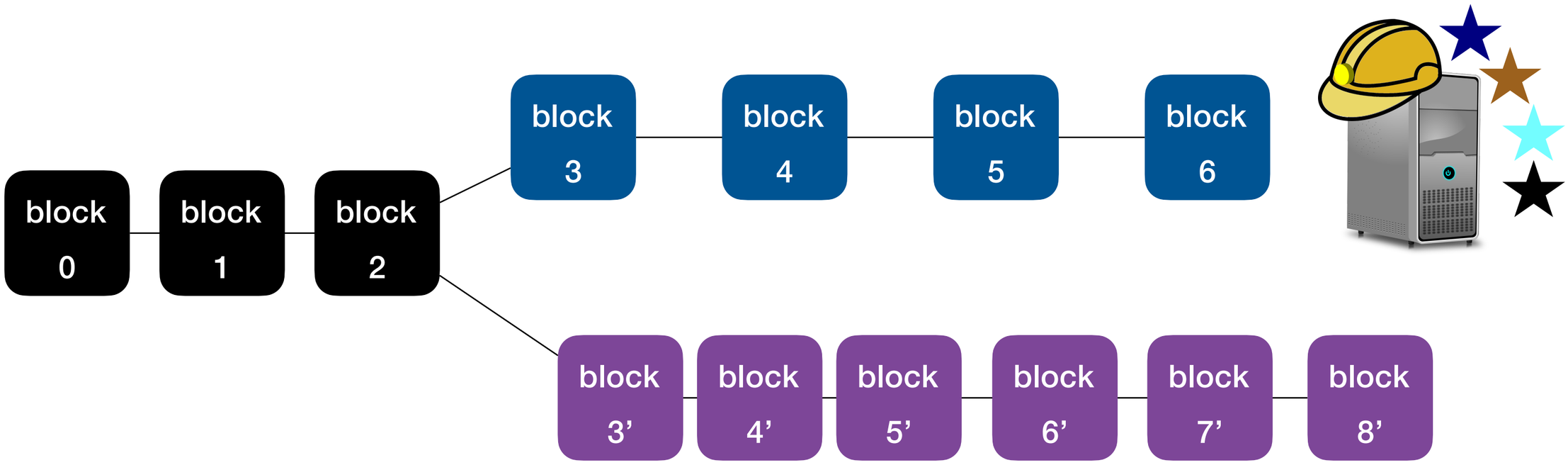
weak blocks must be generated *quickly* to catch up with the honest, ever growing chain (quick ⇒ few hashes ⇒ light weight)

counter attack: use fake timestamps for blocks in the future

Longest vs Strongest Chain Rule

so far we talked about the longest chain rule but …

bitcoin actually uses the ***strongest*** chain rule



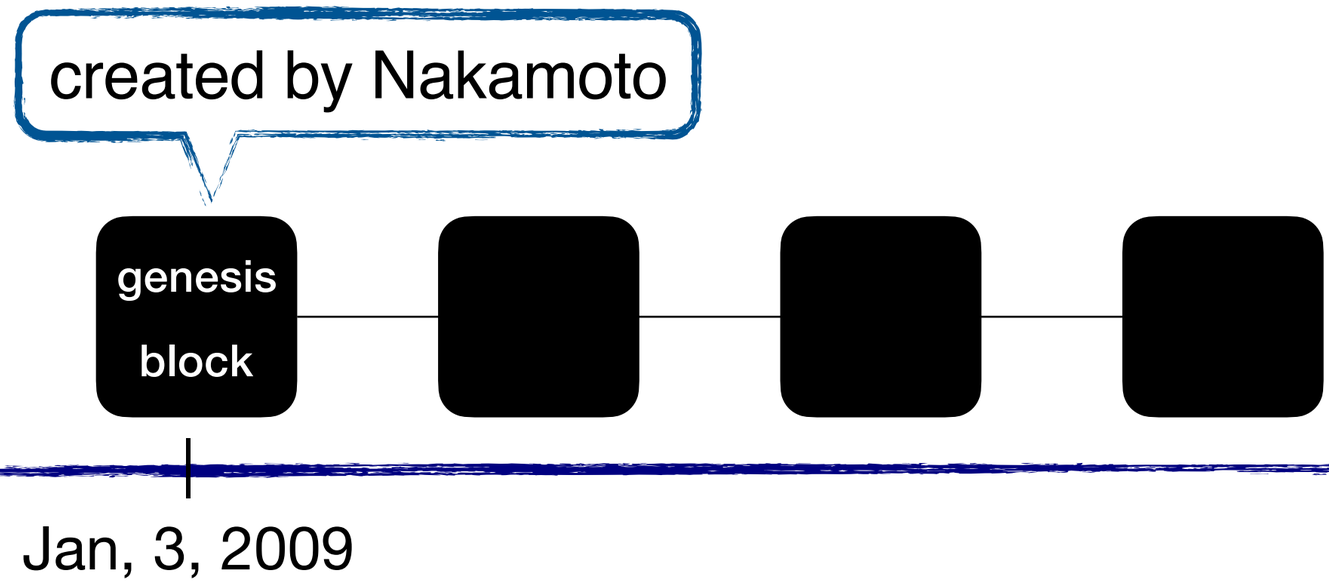
Mitigation:

add **weights** to blocks (proportional to the estimated number number of hashes needed to solve the block difficulty)

weak blocks must be generated *quickly* to catch up with the honest, ever growing chain (quick ⇒ few hashes ⇒ light weight)

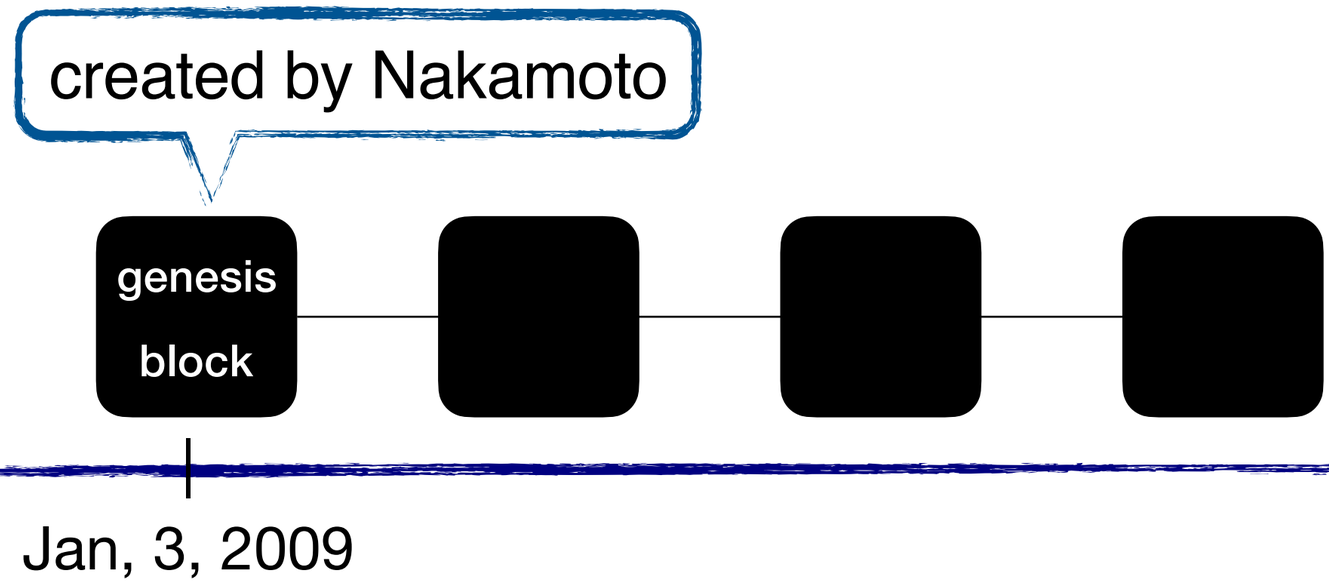
counter attack: use fake timestamps for blocks in the future — detectable!

Freshness of the Genesis Block



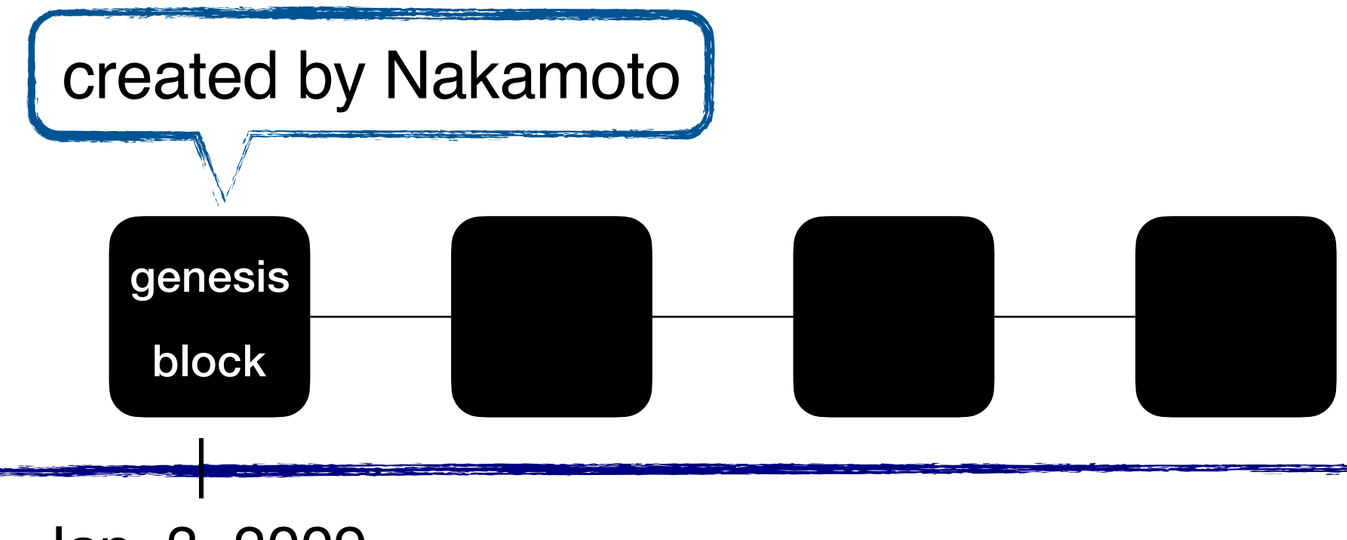
Freshness of the Genesis Block

implicit claim: “I did not know the genesis block before Bitcoin was launched”

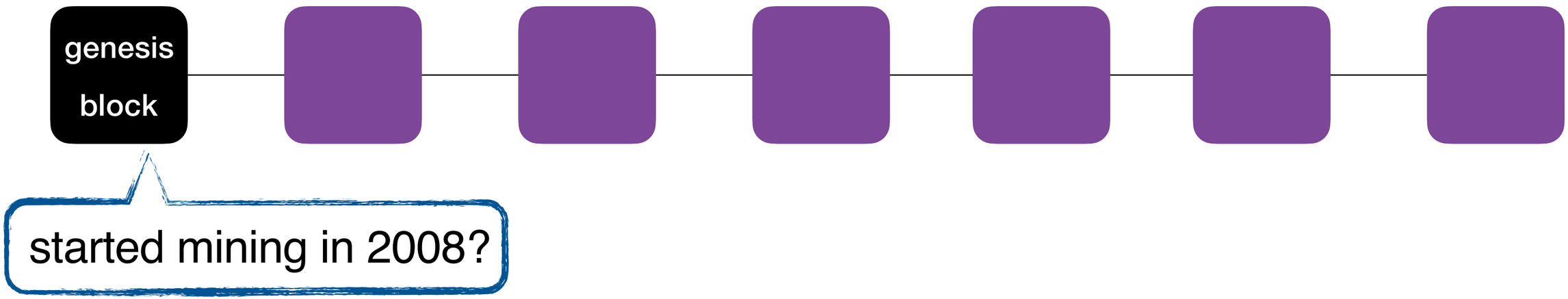


Freshness of the Genesis Block

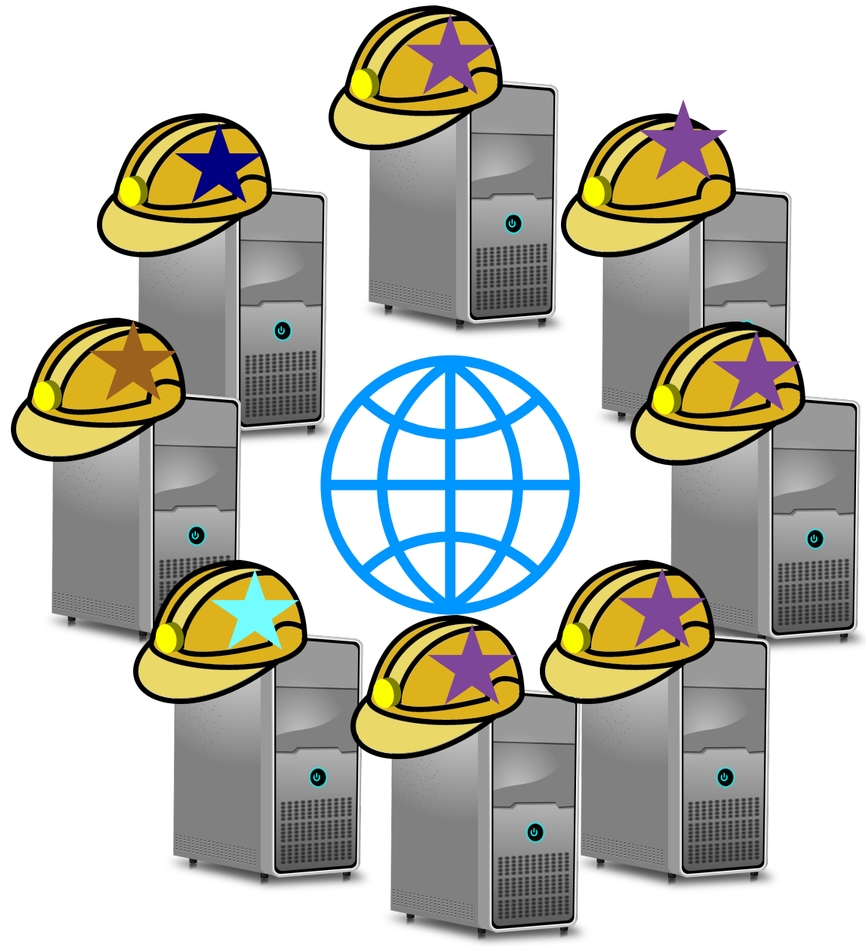
implicit claim: “I did not know the genesis block before Bitcoin was launched”



what if in reality

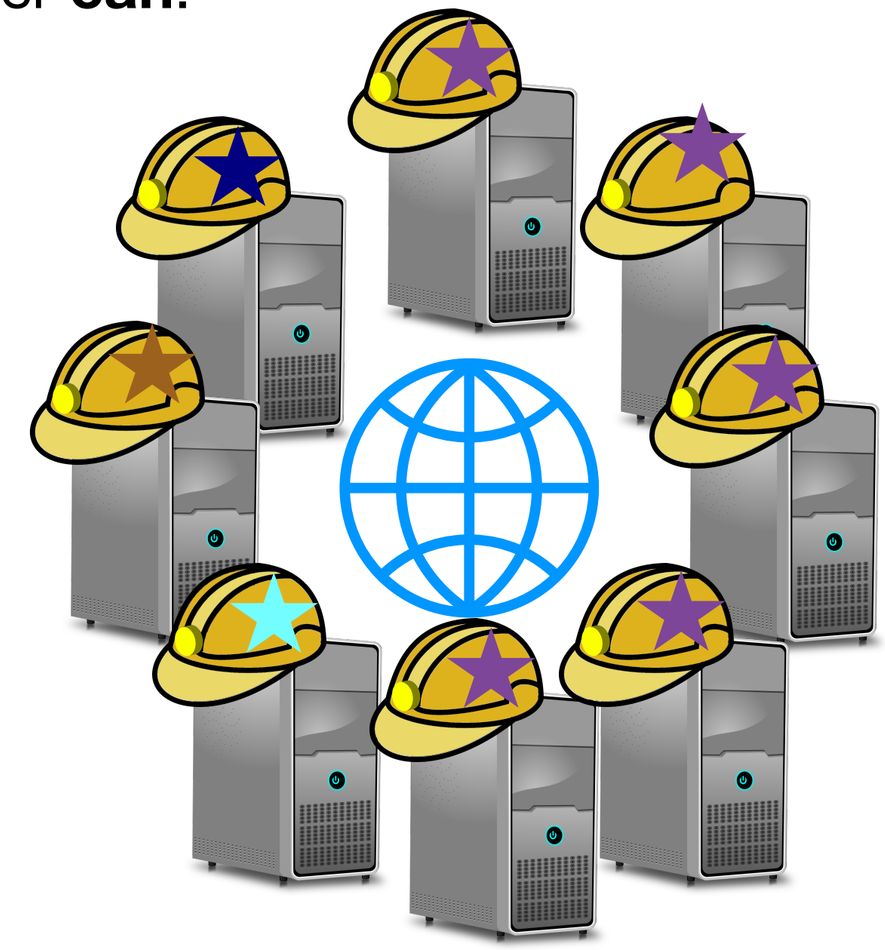


The 51% Attack



The 51% Attack

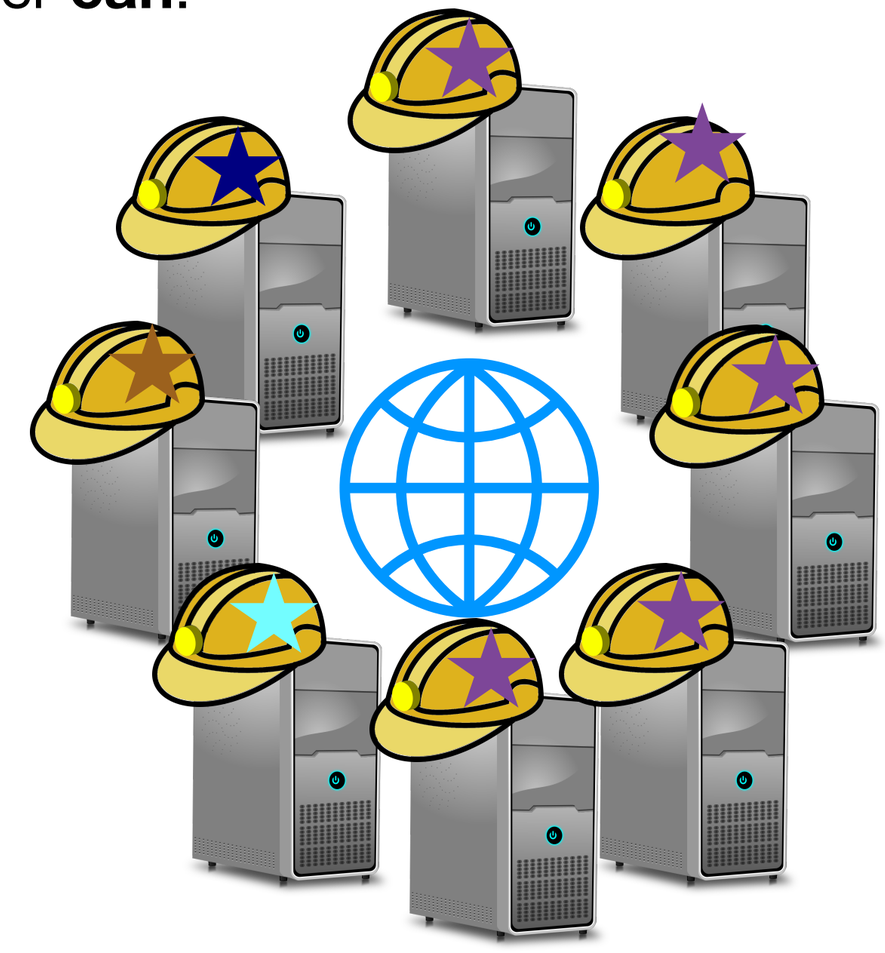
An adversary controlling majority of computational power **can**:



The 51% Attack

An adversary controlling majority of computational power **can**:

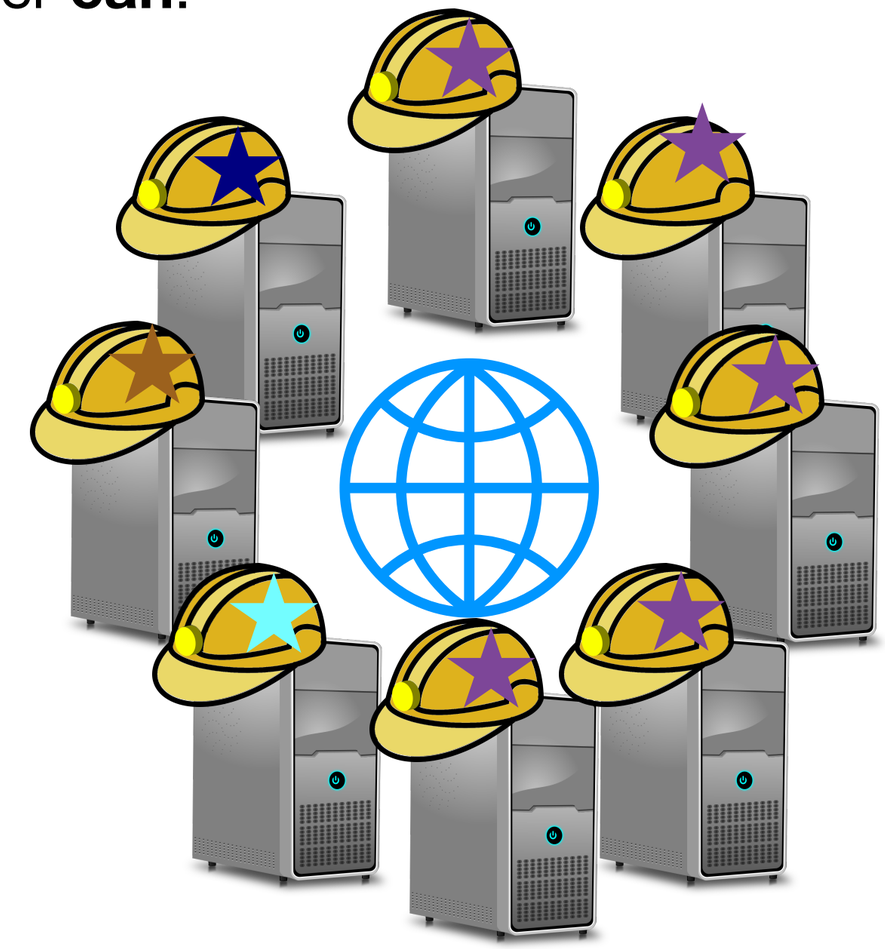
- Fork the chain and doublespend



The 51% Attack

An adversary controlling majority of computational power **can**:

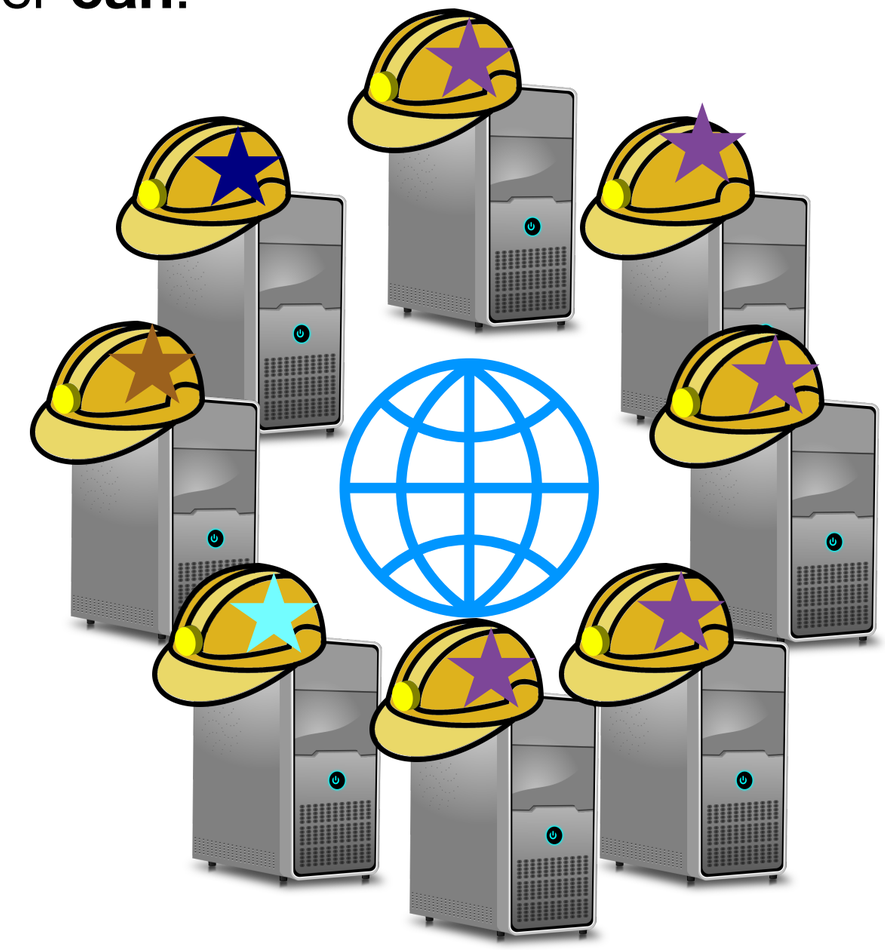
- Fork the chain and doublespend
- Reject all other miners' blocks



The 51% Attack

An adversary controlling majority of computational power **can**:

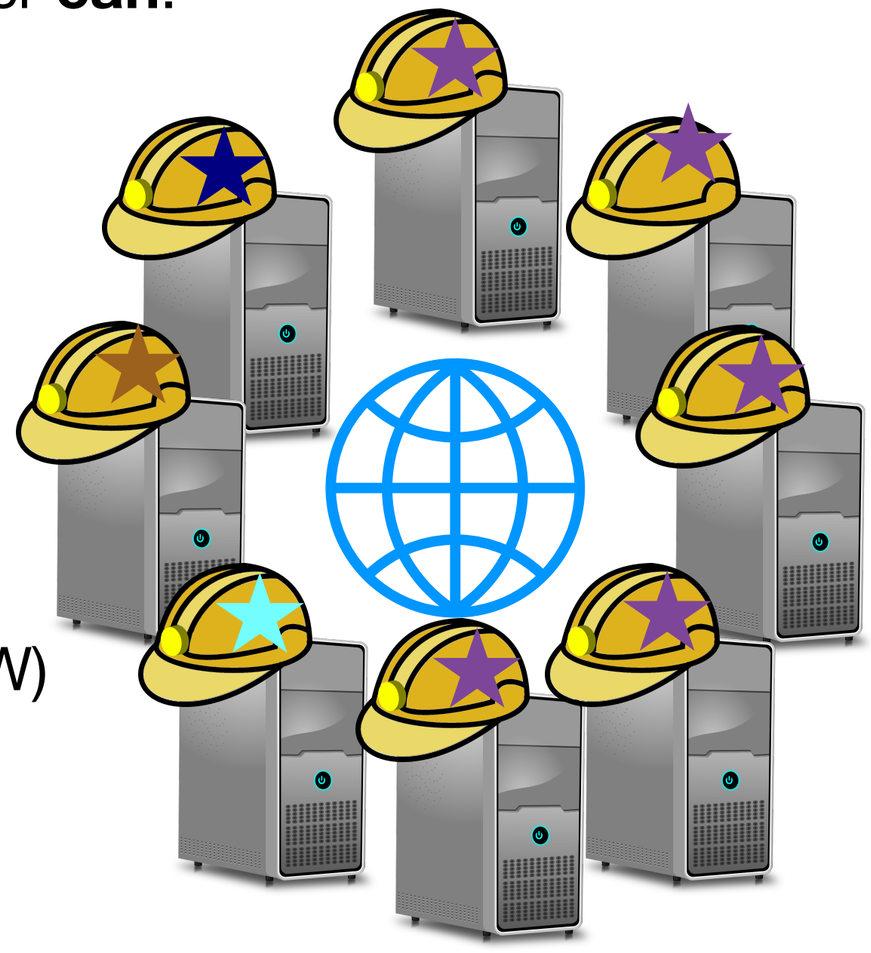
- Fork the chain and doublespend
- Reject all other miners' blocks
- Exclude certain transactions



The 51% Attack

An adversary controlling majority of computational power **can**:

- Fork the chain and doublespend
- Reject all other miners' blocks
- Exclude certain transactions



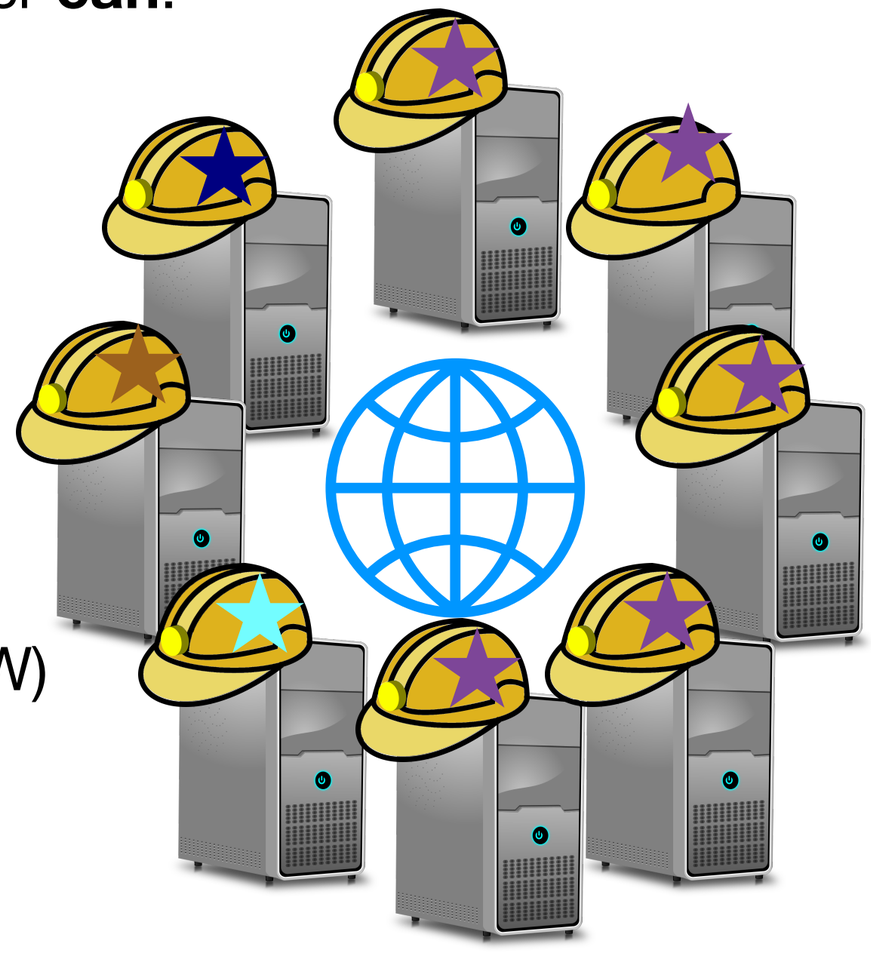
However such an adversary **cannot**:

- Steal money from earlier transactions (signatures)
- Generate money without effort (still needs to solve PoW)

The 51% Attack

An adversary controlling majority of computational power **can**:

- Fork the chain and doublespend
- Reject all other miners' blocks
- Exclude certain transactions



However such an adversary **cannot**:

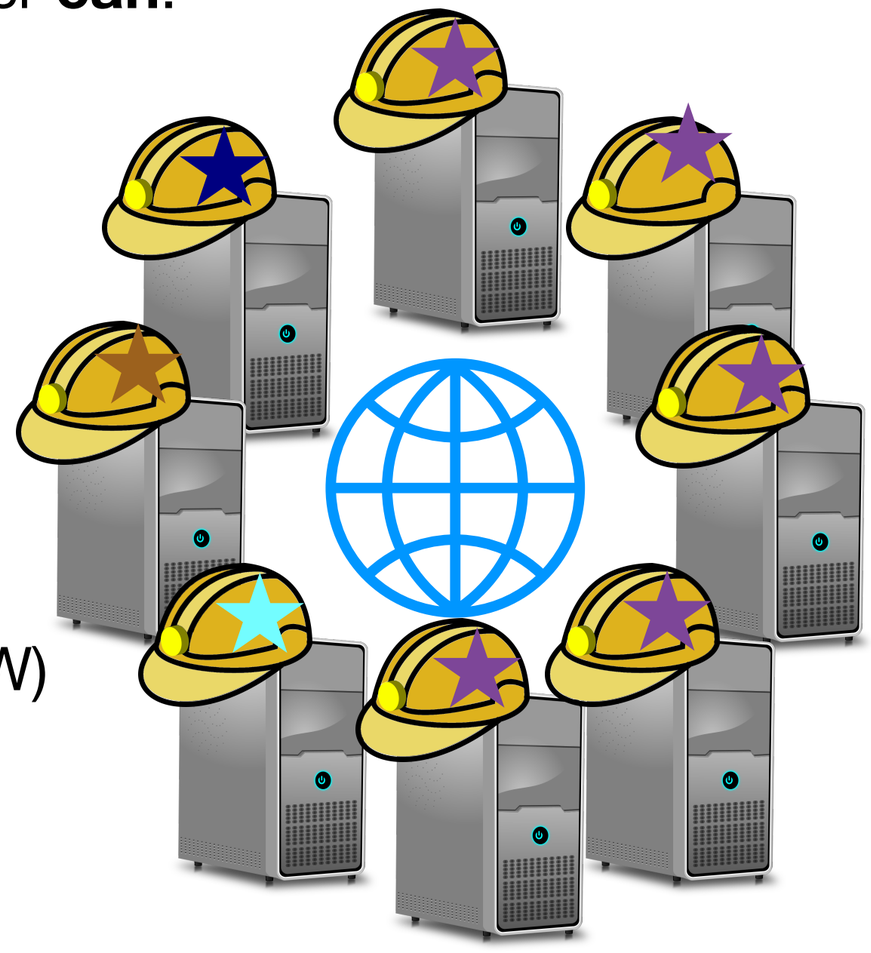
- Steal money from earlier transactions (signatures)
- Generate money without effort (still needs to solve PoW)

The Selfish Miner Attack

The 51% Attack

An adversary controlling majority of computational power **can**:

- Fork the chain and doublespend
- Reject all other miners' blocks
- Exclude certain transactions



However such an adversary **cannot**:

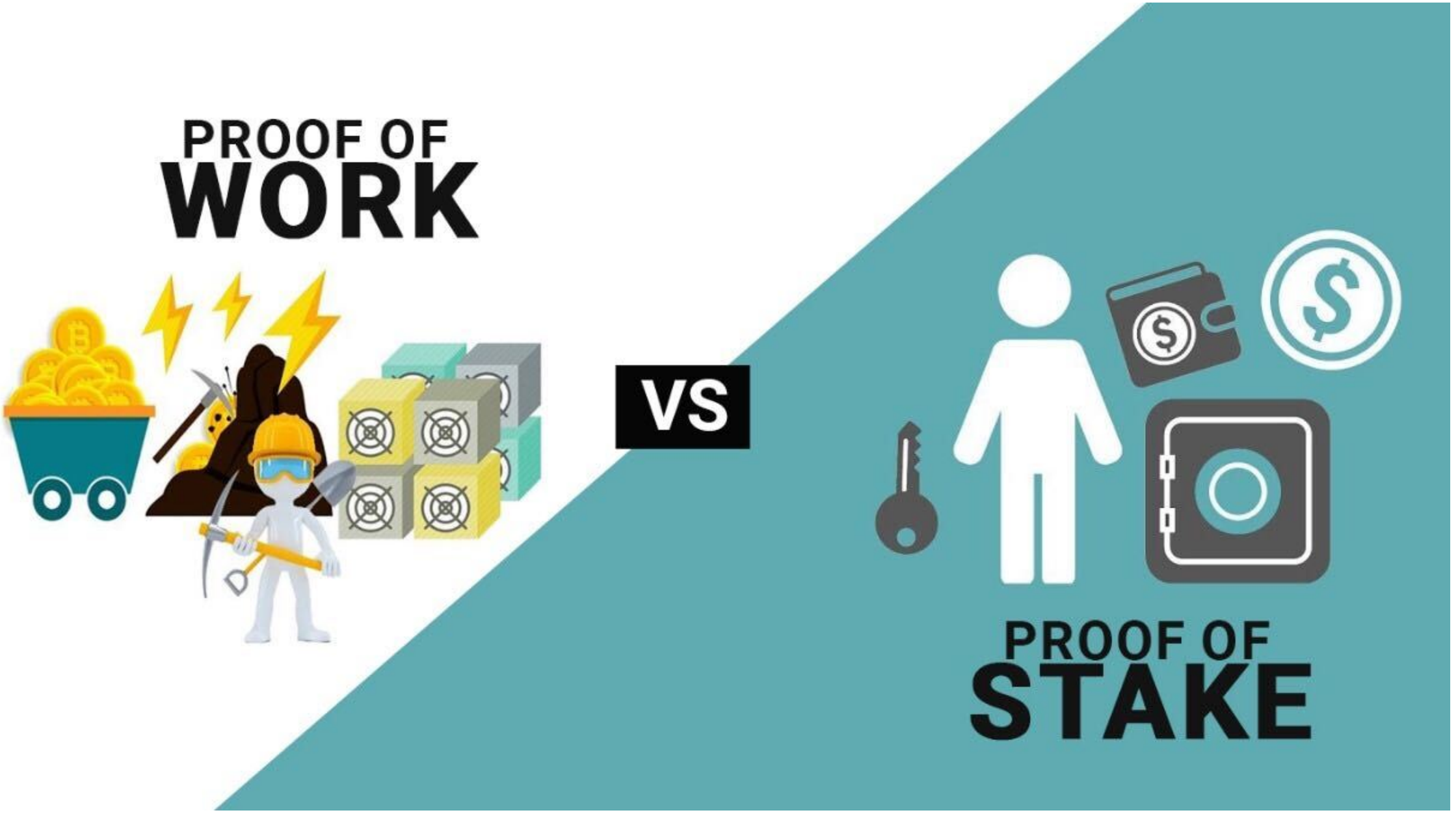
- Steal money from earlier transactions (signatures)
- Generate money without effort (still needs to solve PoW)

The Selfish Miner Attack

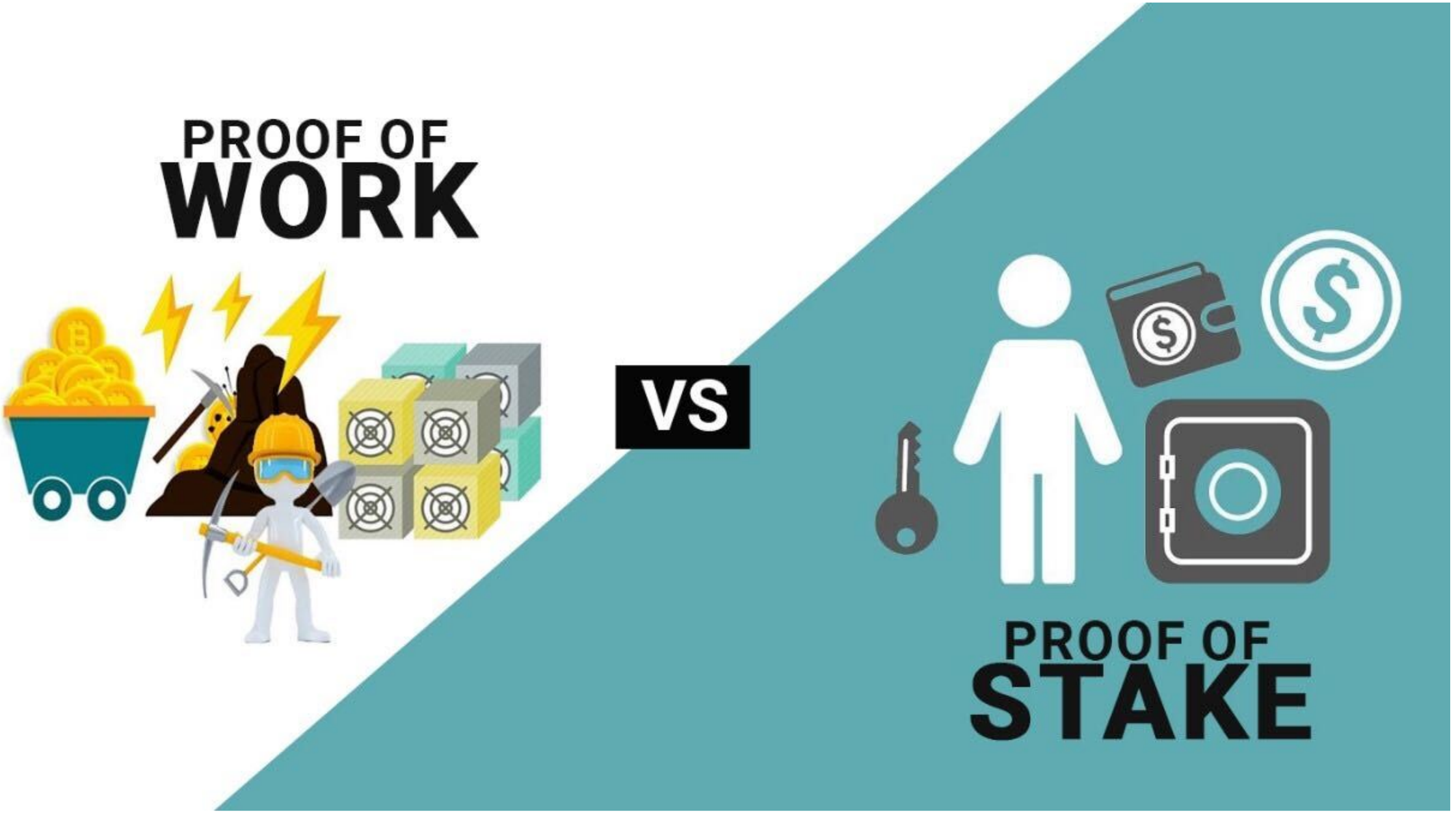
Produce a new block and keep it to oneself:

- Honest miners waste their effort to mine blocks that will never make it to the blockchain

Alternative Consensus Mechanisms

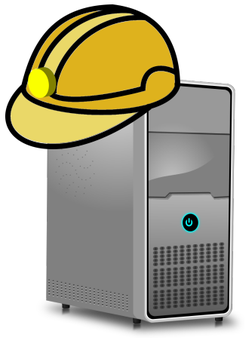


Alternative Consensus Mechanisms

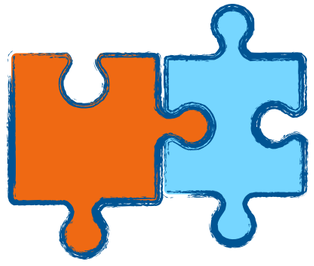


why do we need consensus?

Proof of Work (Recap)

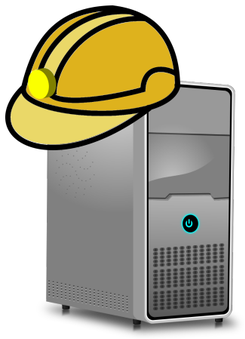


PoW

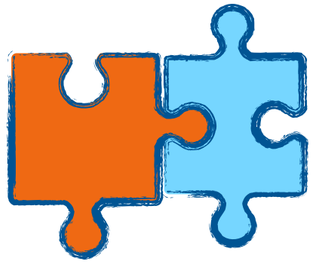


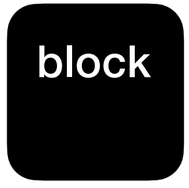
block

Proof of Work (Recap)



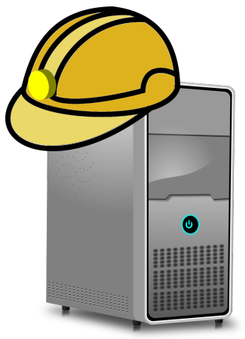
PoW



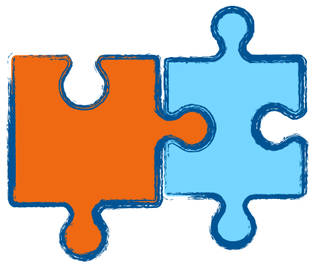


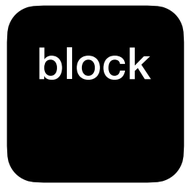
- ★ miners compete to solve a hard puzzles
- ★ the winner publishes the new block on the chain
- ★ all miners efficiently check the correctness of the new block
- ★ incentives for mining come from block rewards and transactions fees
- ★ implements a digital lottery system: the probability of being selected as the winner is exactly one's percentage mining power

Proof of Work (Recap)



PoW





- ★ miners compete to solve a hard puzzles
- ★ the winner publishes the new block on the chain
- ★ all miners efficiently check the correctness of the new block
- ★ incentives for mining come from block rewards and transactions fees
- ★ implements a digital lottery system: the probability of being selected as the winner is exactly one's percentage mining power

Single Bitcoin Transaction Footprints

Carbon Footprint	Electrical Energy	Electronic Waste
873.58 kgCO2	1839.11 kWh	262.70 grams
Equivalent to the carbon footprint of **1,936,150** VISA transactions or **145,596** hours of watching Youtube.	Equivalent to the power consumption of an average U.S. household over **63.04** days.	Equivalent to the weight of **1.60** iPhones 12 or **0.54** iPads. (Find more info on e-waste here.)

Mining One Bitcoin	One BTC Worth of Gold
521,675 kWh	39,560 kWh
of electrical energy is used to mine a single Bitcoin.	of electrical energy is used to mine a BTC worth of gold.

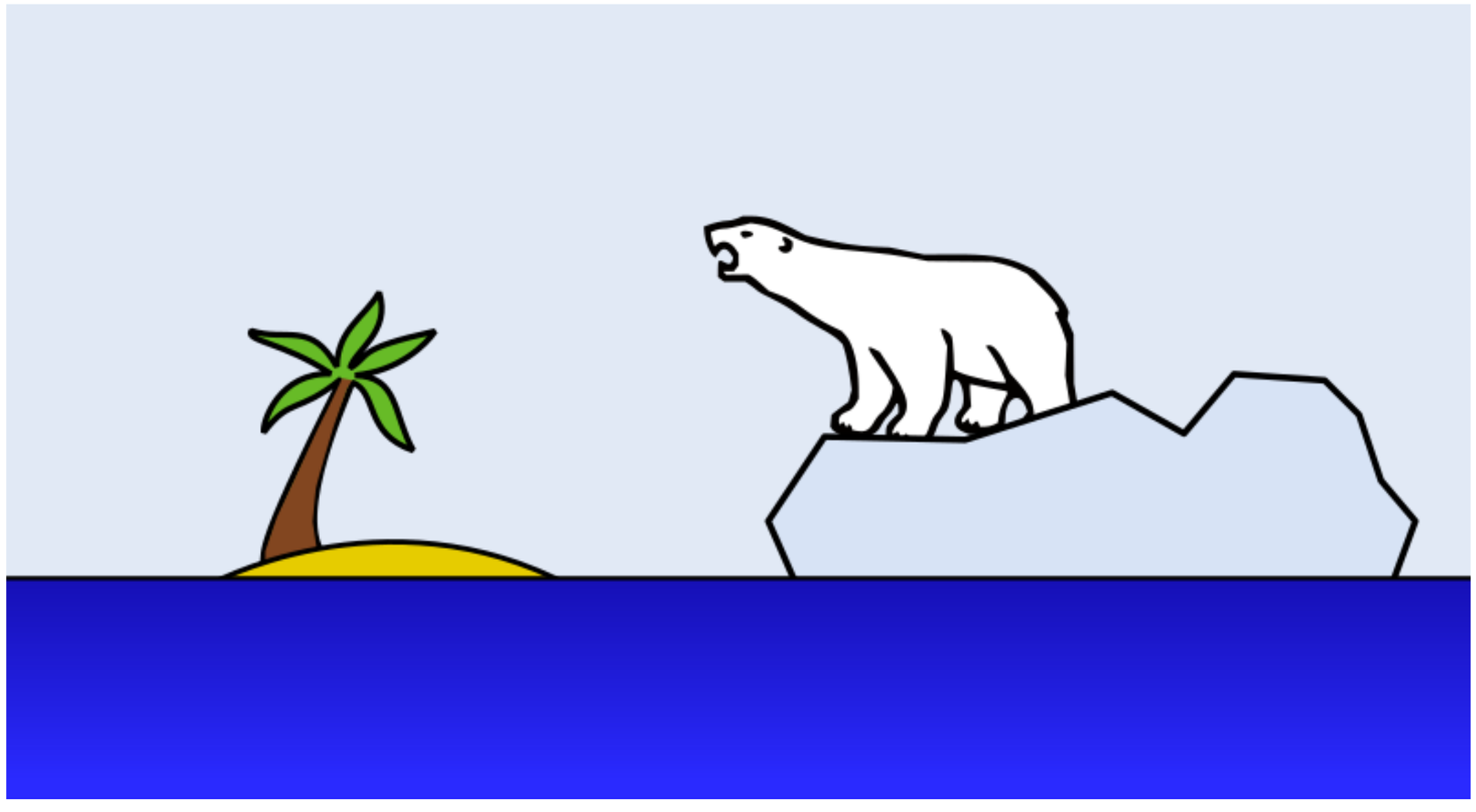
source: digiconomist.net

ENVIRONMENT

Rain Fell On The Peak Of Greenland's Ice Sheet For The First Time In Recorded History

August 20, 2021 · 11:00 AM ET

JOE HERNANDEZ



Proof of Stake

Idea: change the rule for selecting the lottery winner

Replace “work” with “stake” (computation power with digital cash)

The probability of being selected is directly proportional to one’s stake in the “bid”

Proof of Stake

Idea: change the rule for selecting the lottery winner

Replace “work” with “stake” (computation power with digital cash)

The probability of being selected is directly proportional to one’s stake in the “bid”

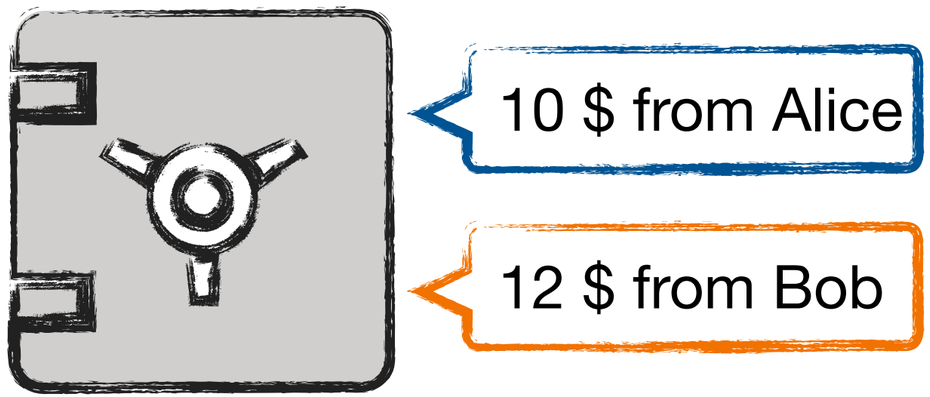
Proof of Stake

Idea: change the rule for selecting the lottery winner

Replace “work” with “stake” (computation power with digital cash)

The probability of being selected is directly proportional to one’s stake in the “bid”

temporary lock stakes to determine
who mints the next block



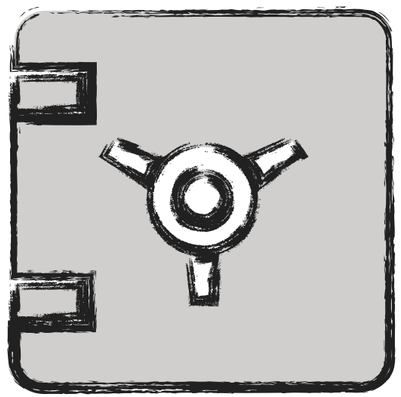
Proof of Stake

Idea: change the rule for selecting the lottery winner

Replace “work” with “stake” (computation power with digital cash)

The probability of being selected is directly proportional to one’s stake in the “bid”

temporary lock stakes to determine
who mints the next block



10 $ from Alice

12 $ from Bob

3 $ from Charlie

winner gets to check if all transactions in the block are valid, earns the transactions fees

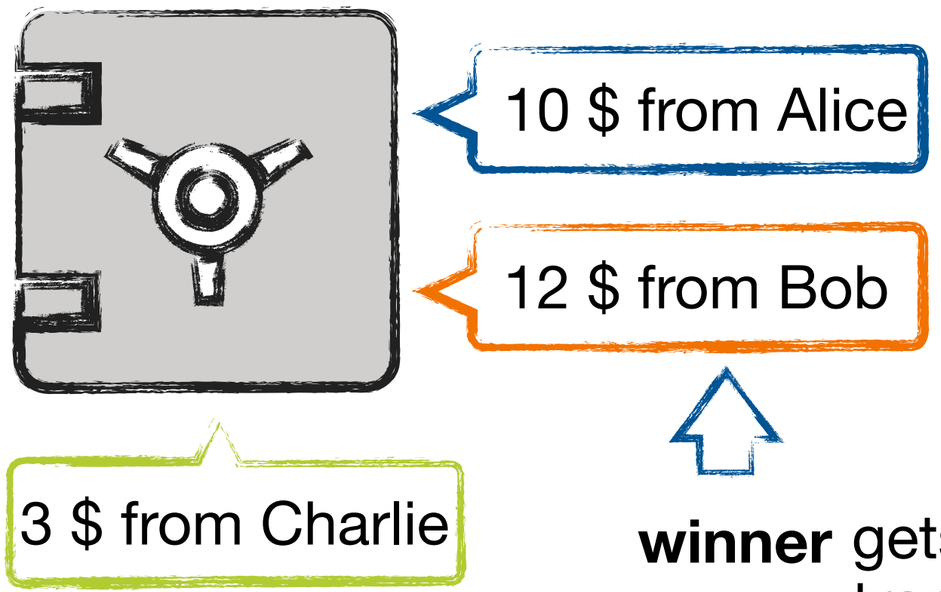
Proof of Stake

Idea: change the rule for selecting the lottery winner

Replace “work” with “stake” (computation power with digital cash)

The probability of being selected is directly proportional to one’s stake in the “bid”

temporary lock stakes to determine
who mints the next block



winner gets to check if all transactions in the block are valid, earns the transactions fees

How to enforce honest behaviours?

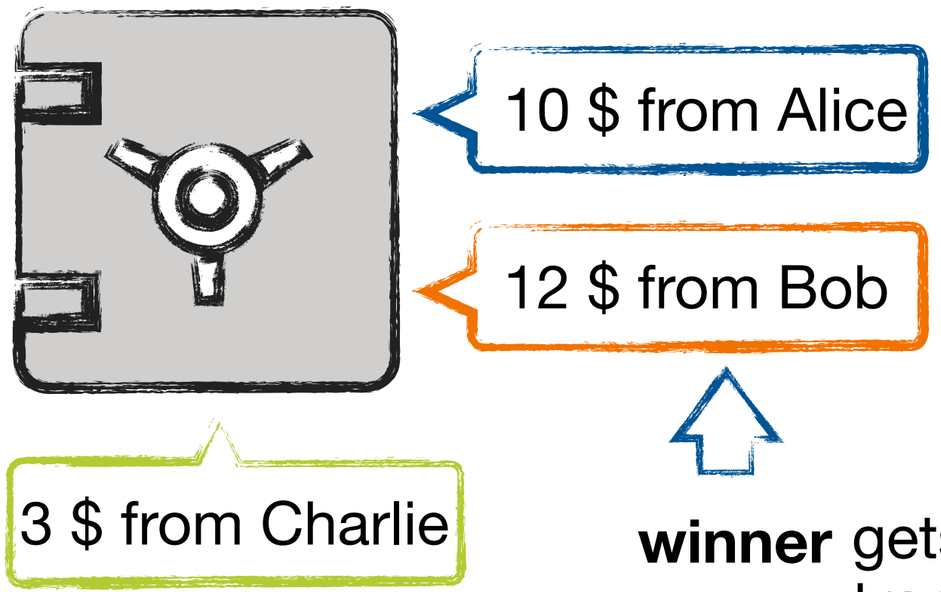
Proof of Stake

Idea: change the rule for selecting the lottery winner

Replace “work” with “stake” (computation power with digital cash)

The probability of being selected is directly proportional to one’s stake in the “bid”

temporary lock stakes to determine
who mints the next block



winner gets to check if all transactions in the block are valid, earns the transactions fees

How to enforce honest behaviours?

Validators lose part of their stake if they approve fraudulent transactions

Proof of Stake

EOS

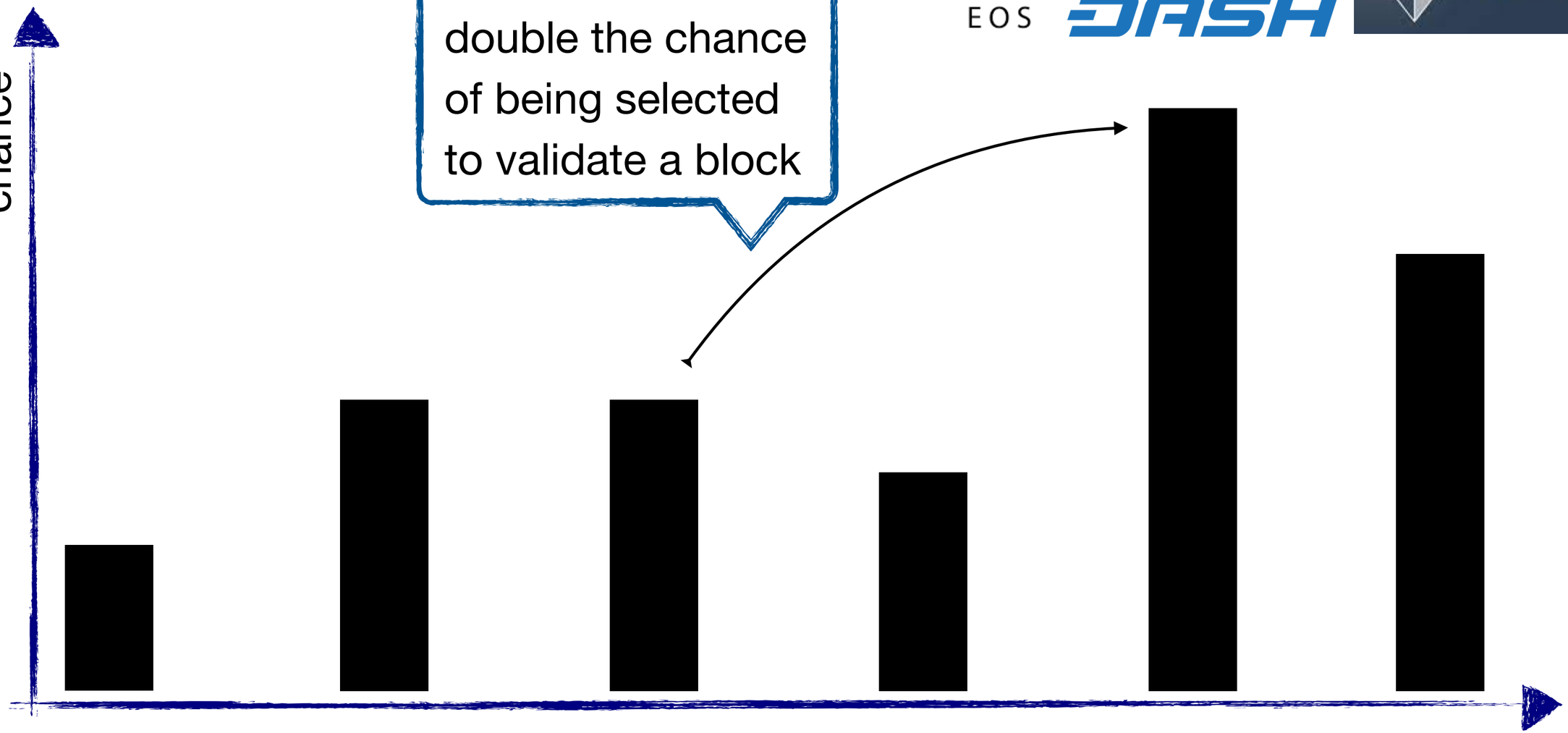
CARDANO

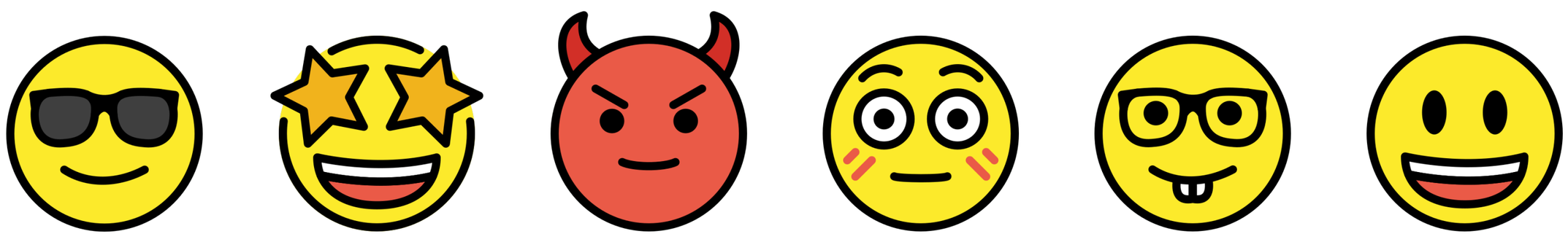
peercoin

DASH

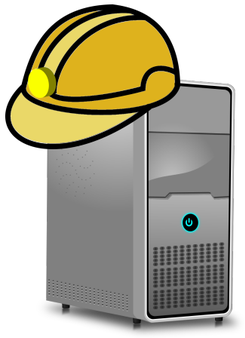
ETHEREUM v2.0

double the stake,
double the chance
of being selected
to validate a block

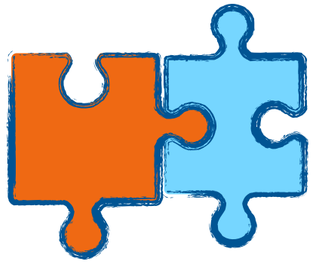


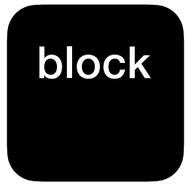


Proof of Work vs Proof of Stake

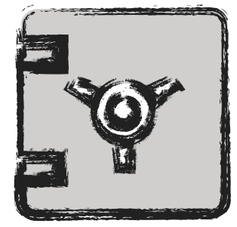


PoW

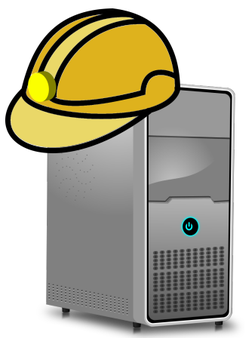




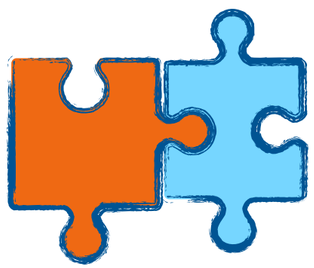
PoS

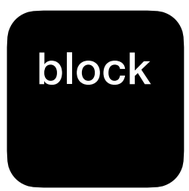


Proof of Work vs Proof of Stake

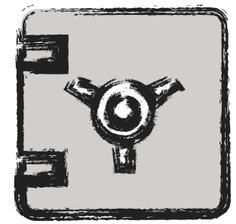


PoW





PoS



★ “miners” collect transaction fees (only)

★ efficient block “minting”

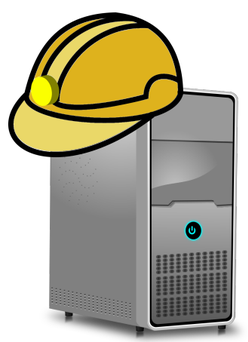
★ better scalability

★ supports for more transactions per second

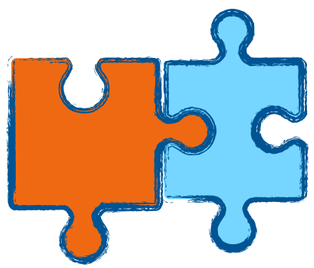
★ consensus is achieved **before** a block is constructed

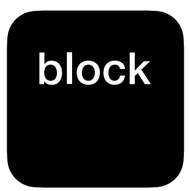
★ deterministic (s)election system: the one that input the highest stake gets to create the new block

Proof of Work vs Proof of Stake

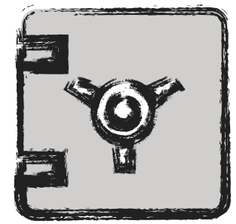


PoW





PoS



★ “miners” collect transaction fees (only)

★ efficient block “minting”

★ better scalability

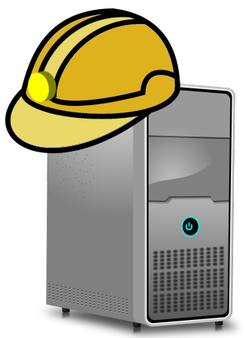
★ supports for more transactions per second

★ consensus is achieved **before** a block is constructed

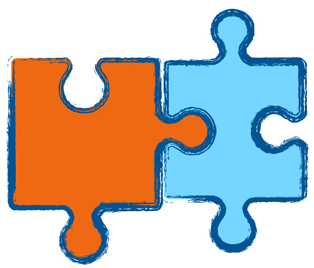
★ deterministic (s)election system: the one that input the highest stake gets to create the new block

Both selection mechanisms are *biased*

Proof of Work vs Proof of Stake



PoW



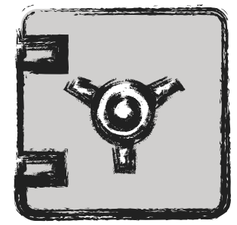
block

reward miners with best hardware (high computational power)

PoWs

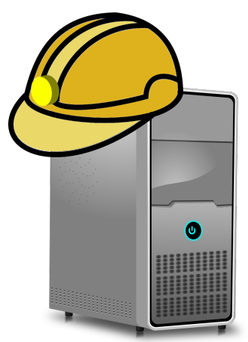
Both selection mechanisms are *biased*

PoS

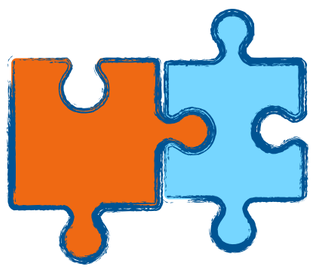


- ★ “miners” collect transaction fees (only)
- ★ efficient block “minting”
- ★ better scalability
- ★ supports for more transactions per second
- ★ consensus is achieved **before** a block is constructed
- ★ deterministic (s)election system: the one that input the highest stake gets to create the new block

Proof of Work vs Proof of Stake

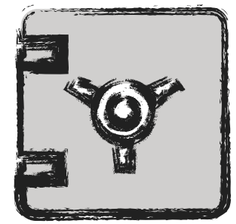


PoW



block

PoS



★ “miners” collect transaction fees (only)

★ efficient block “minting”

★ better scalability

★ supports for more transactions per second

★ consensus is achieved **before** a block is constructed

★ deterministic (s)election system: the one that input the highest stake gets to create the new block

reward miners with best hardware
(high computational power)

PoWs

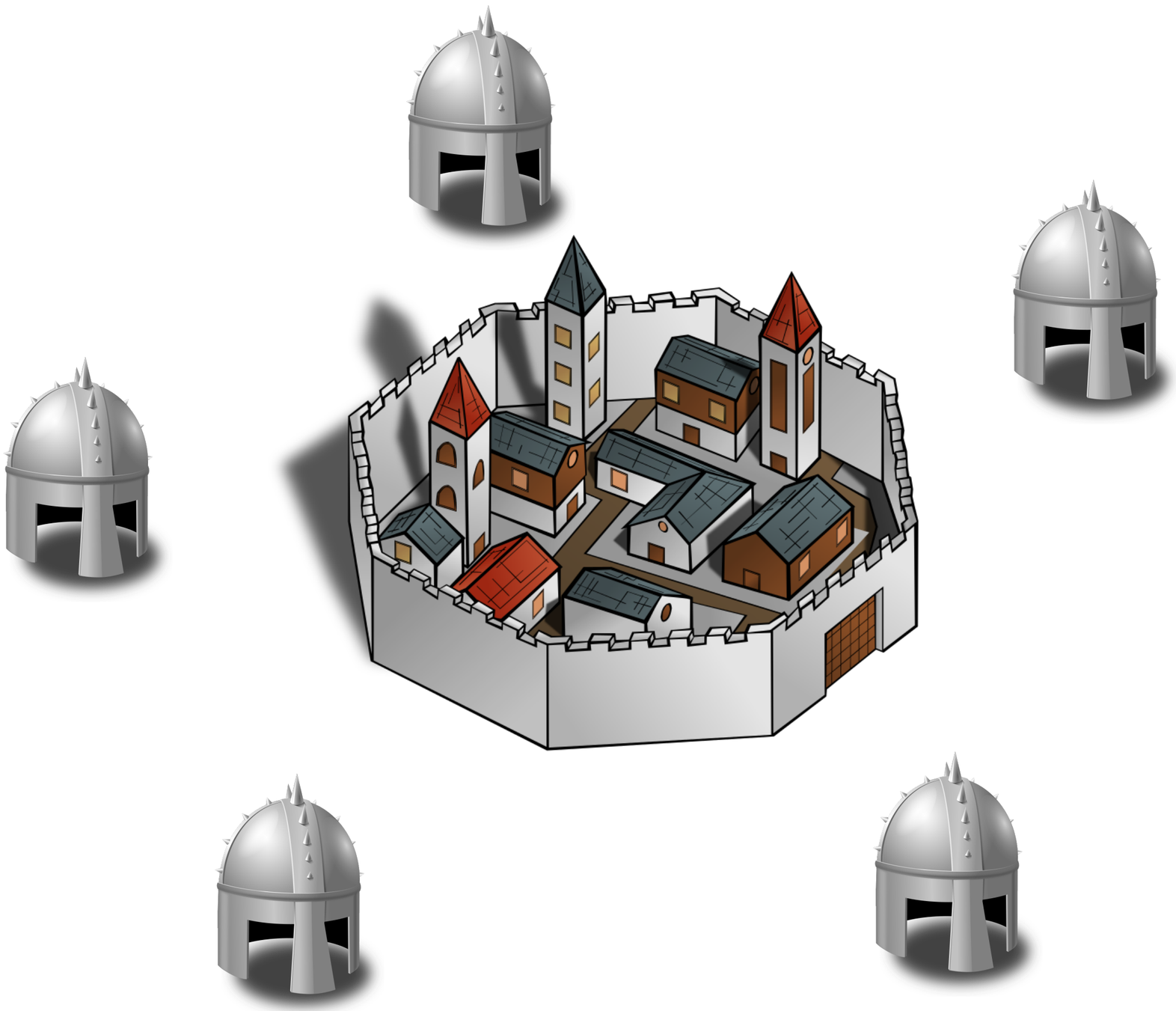
Both selection mechanisms are *biased*

PoS

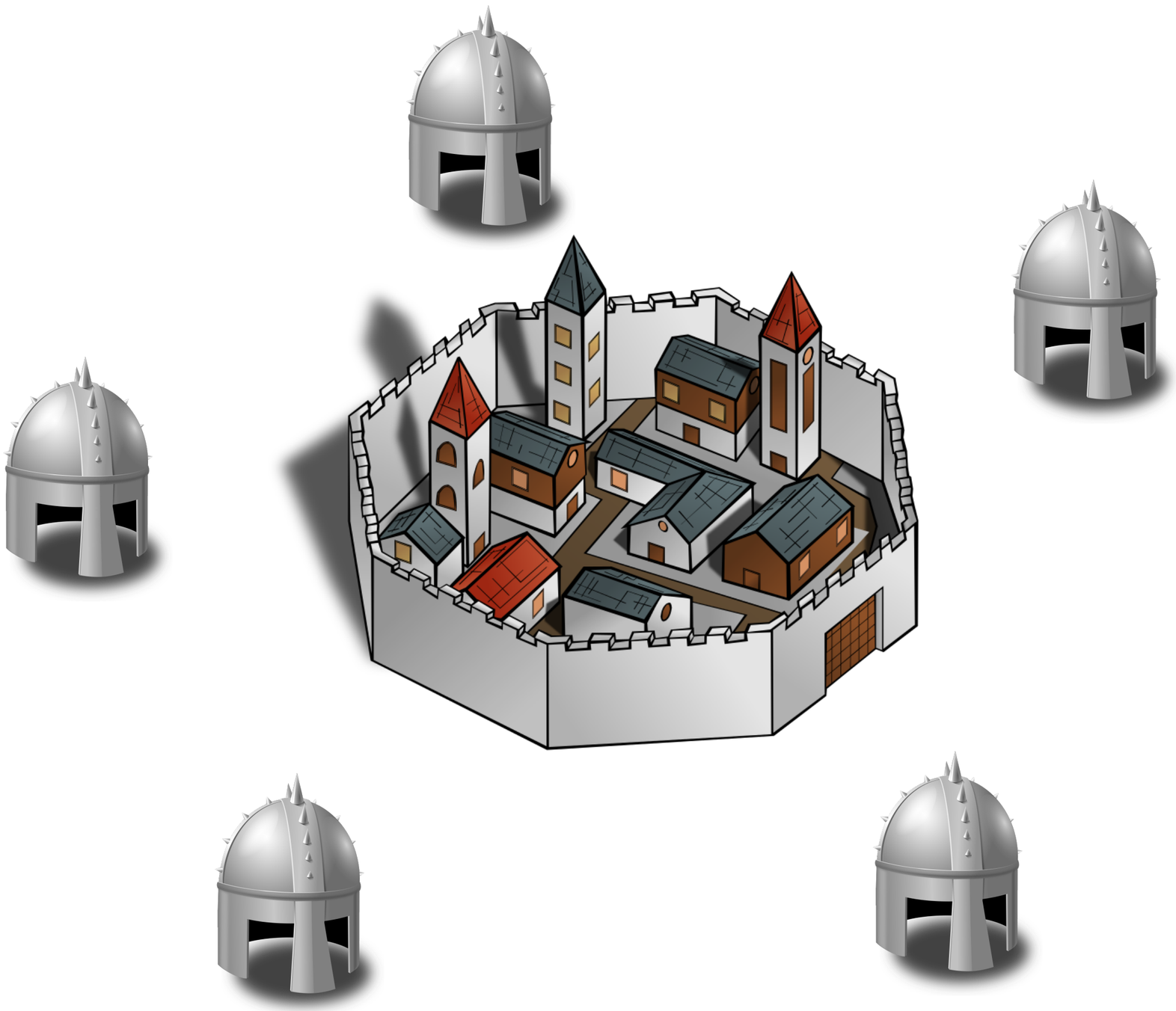
reward wealthier minters
(or who joined the coin first)

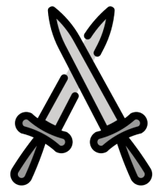
General View on the Problem

General View on the Problem



General View on the Problem

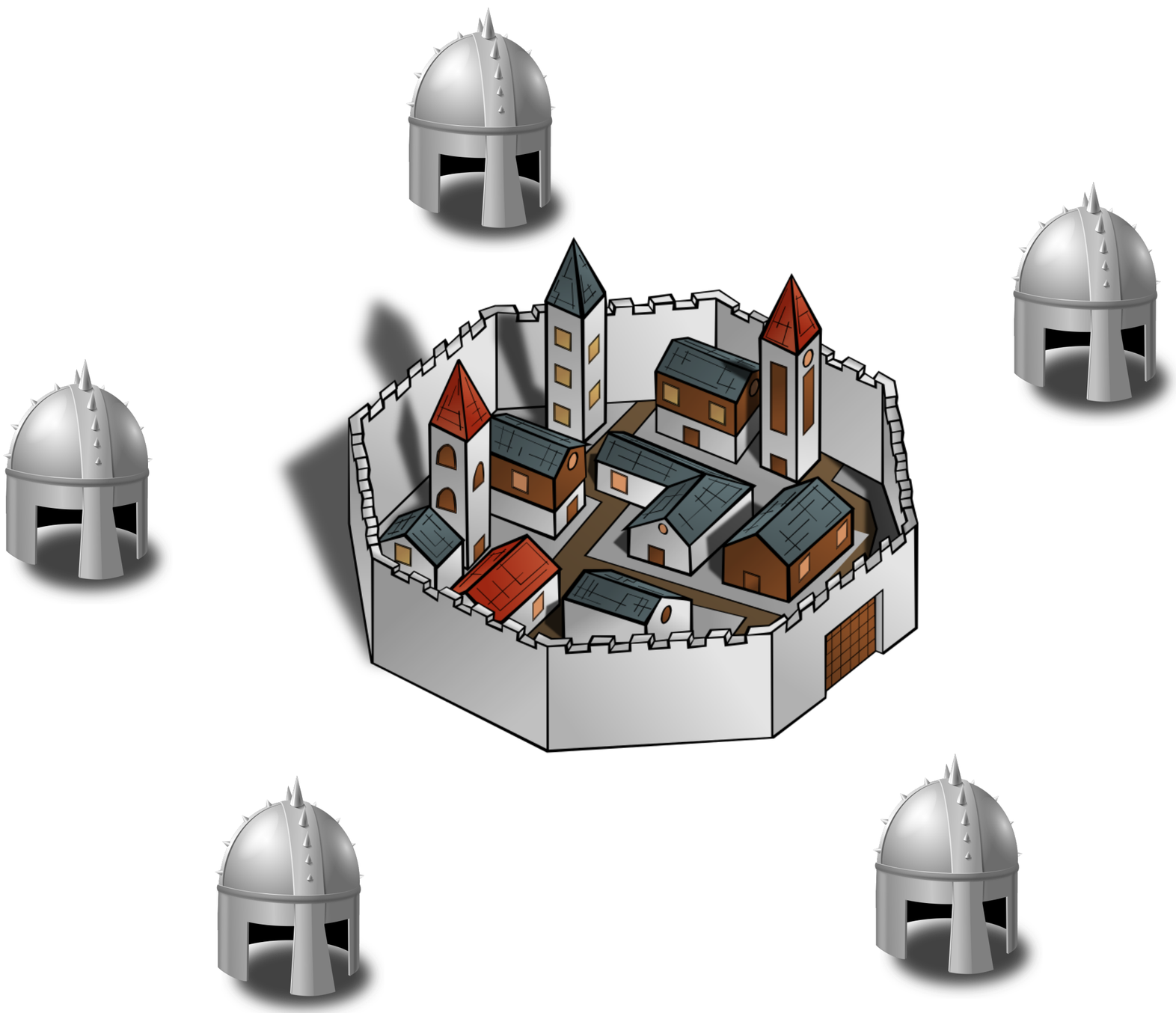


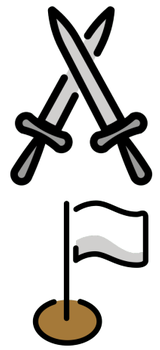


= attack

= retreat

General View on the Problem



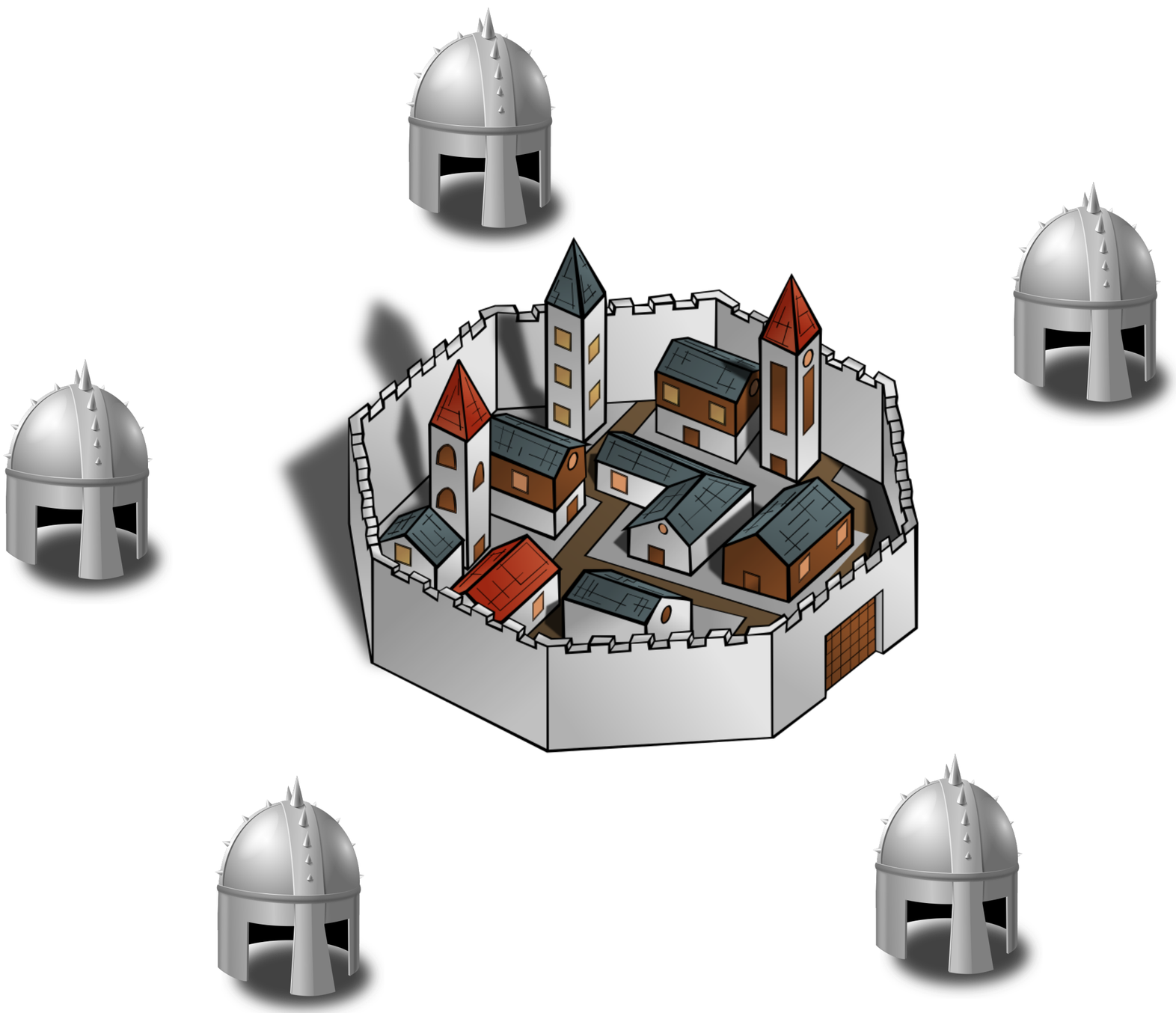


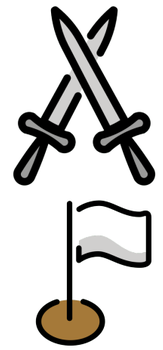
= attack

= retreat

= p2p channels

General View on the Problem



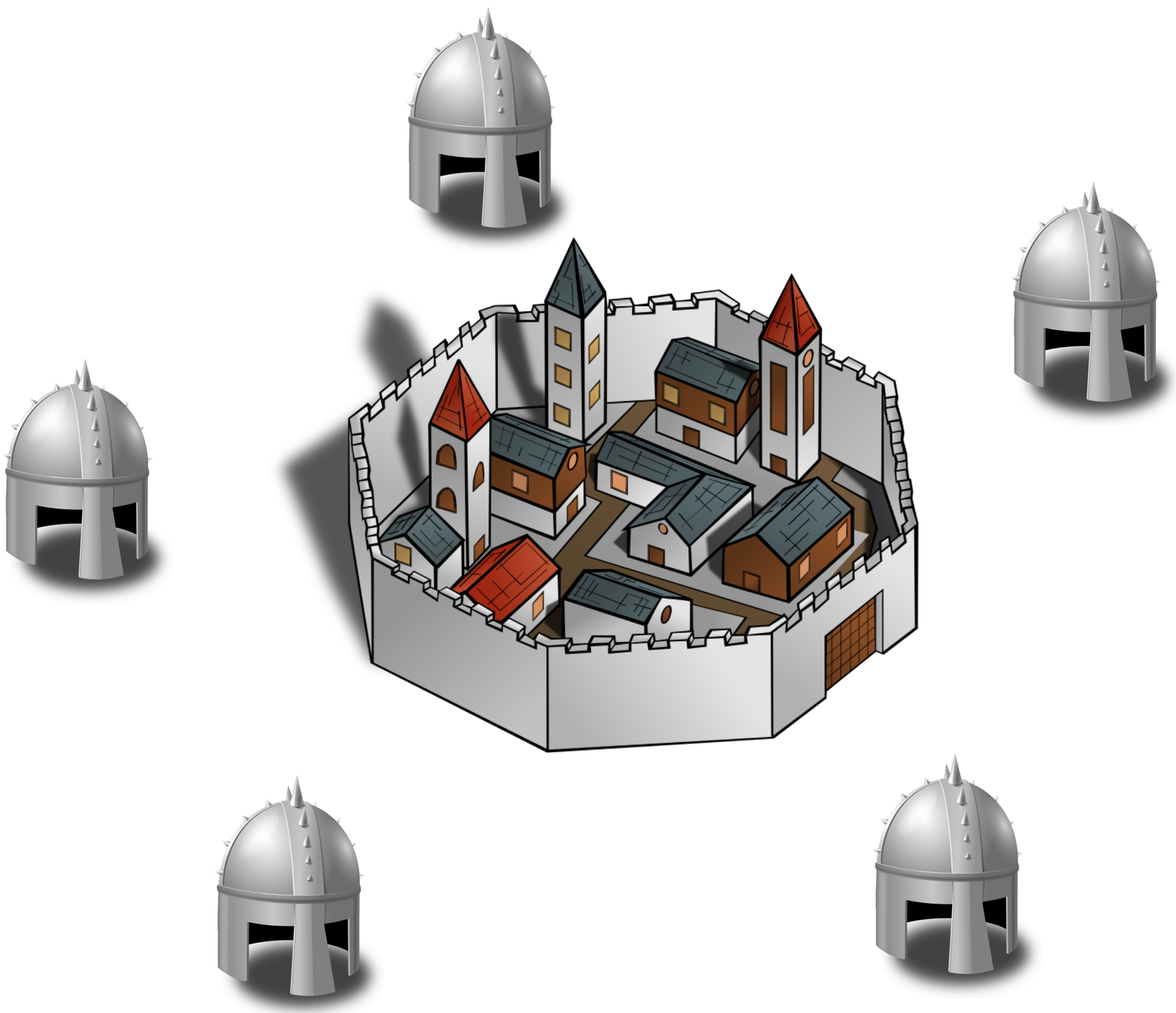


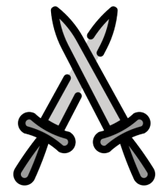
= attack

= retreat

= p2p channels
(not secured)

General View on the Problem





= attack

= retreat

= p2p channels
(not secured)

there are traitors,

Generals with inconsistent behaviours

The Byzantine Fault Tolerance Agreement

The Byzantine Fault Tolerance Agreement

Setting:

The Byzantine Fault Tolerance Agreement

Setting:

★ A system of *n* parties

The Byzantine Fault Tolerance Agreement

Setting:

★ A system of *n* parties

★ With *t* dishonest parties

The Byzantine Fault Tolerance Agreement

Setting:

- ★ A system of *n* parties
- ★ With *t* dishonest parties
- ★ Point-to-point channels

The Byzantine Fault Tolerance Agreement

Setting:

- ★ A system of *n* parties
- ★ With *t* dishonest parties
- ★ Point-to-point channels
- ★ Whenever one party **A** tries to broadcast a value **x**, the other parties are allowed to discuss with each other and verify the consistency of **A**'s broadcast values.

The Byzantine Fault Tolerance Agreement

Setting:

- ★ A system of *n* parties
- ★ With *t* dishonest parties
- ★ Point-to-point channels
- ★ Whenever one party **A** tries to broadcast a value **x**, the other parties are allowed to discuss with each other and verify the consistency of **A**'s broadcast values.

Properties:

The system is said to *resist Byzantine faults* if:

The Byzantine Fault Tolerance Agreement

Setting:

- ★ A system of *n* parties
- ★ With *t* dishonest parties
- ★ Point-to-point channels
- ★ Whenever one party **A** tries to broadcast a value **x**, the other parties are allowed to discuss with each other and verify the consistency of **A**'s broadcast values.

Properties:

The system is said to *resist Byzantine faults* if:

1. For all honest parties **H** sending a value **x_H**, all other honest parties in the system agree on the value **x_H** for **H**.

The Byzantine Fault Tolerance Agreement

Setting:

- ★ A system of *n* parties
- ★ With *t* dishonest parties
- ★ Point-to-point channels
- ★ Whenever one party **A** tries to broadcast a value **x**, the other parties are allowed to discuss with each other and verify the consistency of **A**'s broadcast values.

Properties:

The system is said to *resist Byzantine faults* if:

1. For all honest parties **H** sending a value **x_H**, all other honest parties in the system agree on the value **x_H** for **H**.
2. (No guarantees on dishonest parties)

The Byzantine Fault Tolerance Agreement

Setting:

- ★ A system of *n* parties
- ★ With *t* dishonest parties
- ★ Point-to-point channels
- ★ Whenever one party **A** tries to broadcast a value **x**, the other parties are allowed to discuss with each other and verify the consistency of **A**'s broadcast values.

Properties:

The system is said to *resist Byzantine faults* if:

1. For all honest parties **H** sending a value **x_H**, all other honest parties in the system agree on the value **x_H** for **H**.
2. (No guarantees on dishonest parties)
3. At the end of the protocol, all honest parties agree on the same value **y**.

BFT Consensus: History and Applications

BFT Consensus: History and Applications

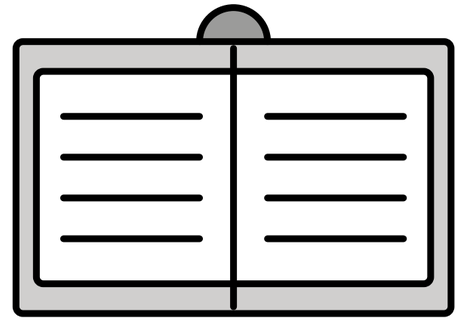
BFT Consensus: History and Applications

BFT Consensus: History and Applications

NASA

BFT Consensus: History and Applications

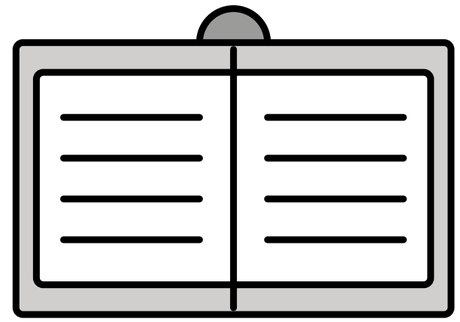
NASA



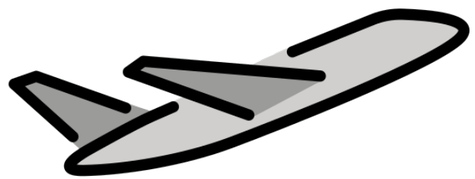
PoW / PoS ⇒ agree on a coherent global view of the ledger

BFT Consensus: History and Applications

NASA

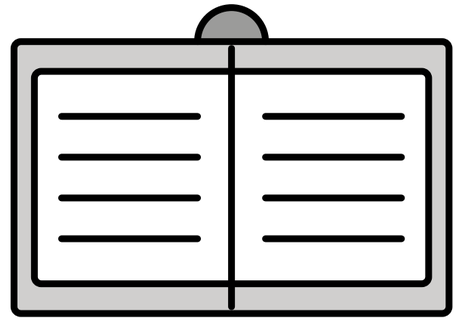


PoW / PoS ⇒ agree on a coherent global view of the ledger

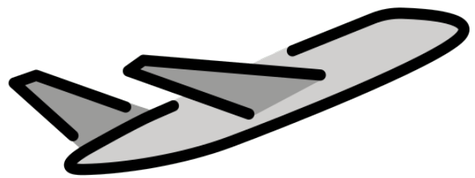


BFT Consensus: History and Applications

NASA



PoW / PoS ⇒ agree on a coherent global view of the ledger



Lecture Agenda

How To Set Up a Bulletin Board

- Hash Functions
- Proof of Work (Cryptographic Puzzles)
- Example: Bitcoin

How To Generate Money

- Mining Blocks
- Merkle Trees
- Block Reword System

How To Circulate Money

- Transactions
- ECDSA

Some Problems With Blockchains

- Freshness of the Genesis Block
- 51% Attack
- PoW Vs PoS
- The Byzantine Fault Tolerance Problem

Advanced Tools

- Zero Knowledge Proofs
- Schnorr Interactive PoK
- Fiat-Shamir Heuristic (NIZK)

Lecture Agenda

How To Set Up a Bulletin Board

- Hash Functions
- Proof of Work (Cryptographic Puzzles)
- Example: Bitcoin

How To Generate Money

- Mining Blocks
- Merkle Trees
- Block Reword System

How To Circulate Money

- Transactions
- ECDSA

Some Problems With Blockchains

- Freshness of the Genesis Block
- 51% Attack
- PoW Vs PoS
- The Byzantine Fault Tolerance Problem

Advanced Tools

- Zero Knowledge Proofs
- Schnorr Interactive PoK
- Fiat-Shamir Heuristic (NIZK)

Demo: Playing Around With Hash Functions

$$H : \{0,1\}^* \rightarrow \{0,1\}^N$$

Demo: Playing Around With Hash Functions

$$H : \{0,1\}^* \rightarrow \{0,1\}^N$$

```
H(H(x))   vs   H(x)
```

Demo: Playing Around With Hash Functions

$$H : \{0,1\}^* \rightarrow \{0,1\}^N$$

```
H(H(x))   vs   H(x)
```

```
H(x)||H(x)   vs   H(x)
```

Demo: Playing Around With Hash Functions

$$H : \{0,1\}^* \rightarrow \{0,1\}^N$$

```
H(H(x))   vs   H(x)

H(x)||H(x)   vs   H(x)

H_1(x)||H_2(x)   vs   H_1(x)
```

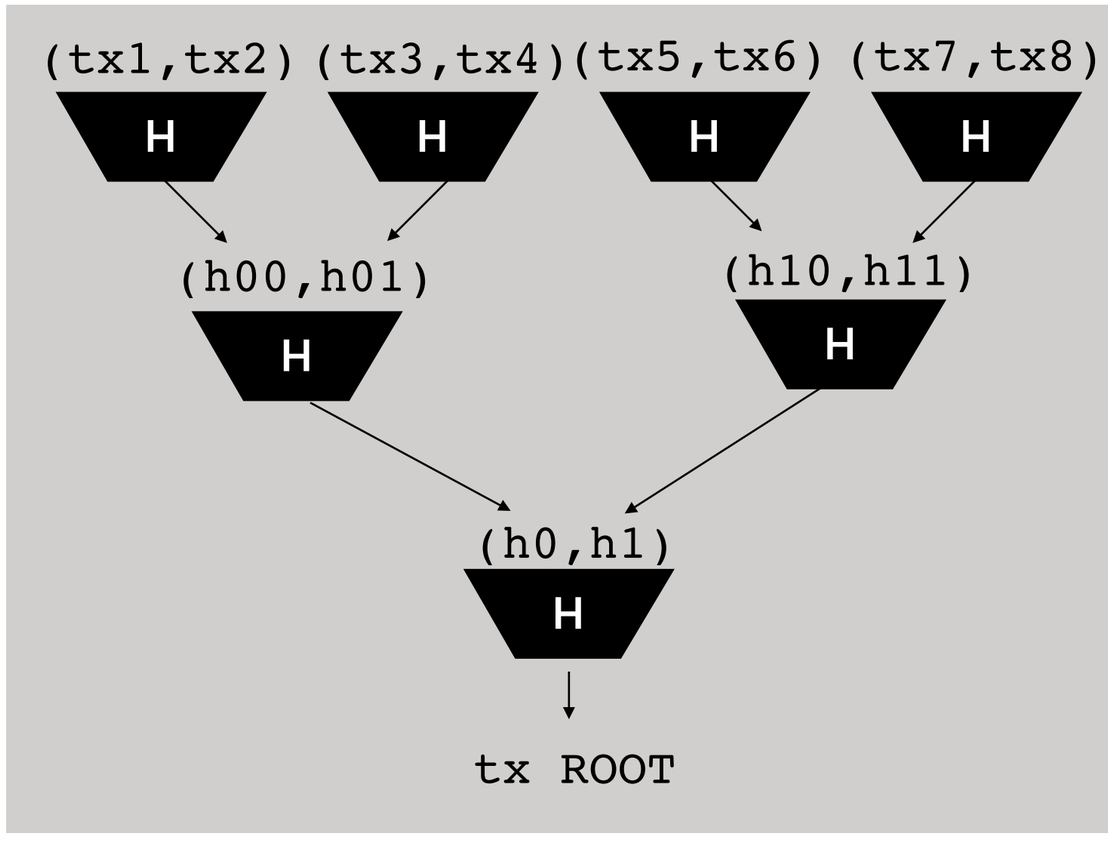
Demo: Playing Around With Hash Functions

$$H : \{0,1\}^* \rightarrow \{0,1\}^N$$

```
H(H(x))   vs   H(x)

H(x)||H(x)   vs   H(x)

H_1(x)||H_2(x)   vs   H_1(x)
```



Lecture Agenda

How To Set Up a Bulletin Board

- Hash Functions
- Proof of Work (Cryptographic Puzzles)
- Example: Bitcoin

How To Generate Money

- Mining Blocks
- Merkle Trees
- Block Reword System

How To Circulate Money

- Transactions
- ECDSA

Some Problems With Blockchains

- Freshness of the Genesis Block
- 51% Attack
- PoW Vs PoS
- The Byzantine Fault Tolerance Problem

Advanced Tools

- Zero Knowledge Proofs
- Schnorr Interactive PoK
- Fiat-Shamir Heuristic (NIZK)

Lecture Agenda

How To Set Up a Bulletin Board

- Hash Functions
- Proof of Work (Cryptographic Puzzles)
- Example: Bitcoin

How To Generate Money

- Mining Blocks
- Merkle Trees
- Block Reword System

How To Circulate Money

- Transactions
- ECDSA

Some Problems With Blockchains

- Freshness of the Genesis Block
- 51% Attack
- PoW Vs PoS
- The Byzantine Fault Tolerance Problem

Advanced Tools

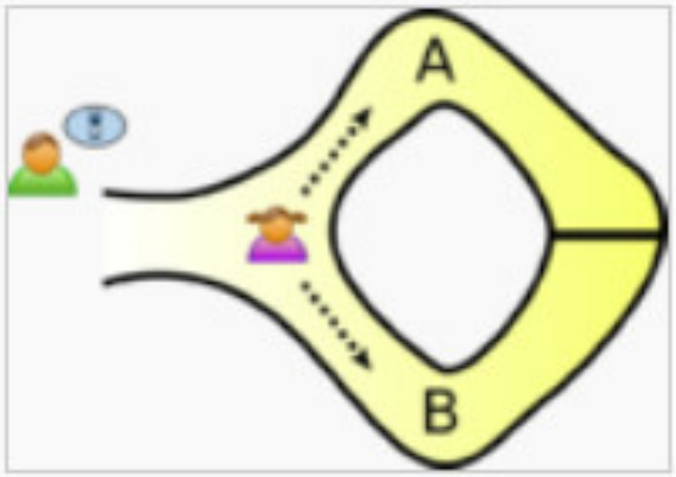
- Zero Knowledge Proofs
- Schnorr Interactive PoK
- Fiat-Shamir Heuristic (NIZK)

You can't have your cake and eat it too!

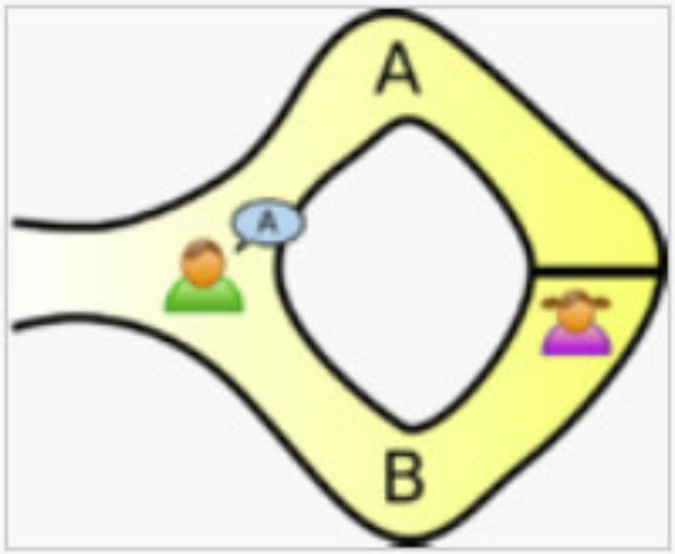
well, with crypto you do :)

zero knowledge proofs

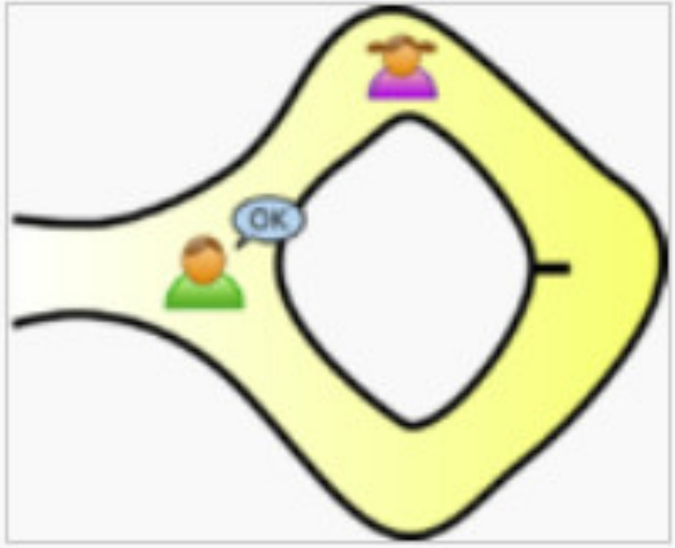
Zero Knowledge Proofs - a Metaphor



Peggy randomly takes either path A or B, while Victor waits outside



Victor chooses an exit path



Peggy reliably appears at the exit Victor names

How To Formalise This Into Math/Crypto?

How To Formalise This Into Math/Crypto?

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

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

How To Formalise This Into Math/Crypto?

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

x

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

w

The most general formalisation: $x \in L \text{ iff } \exists\, w \text{ s.t. } R(x,w) = 1$

How To Formalise This Into Math/Crypto?

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

x

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

w

set of all valid sudoku starters

The most general formalisation: $x \in L$ iff $\exists\ w$ s.t. $R(x,w) = 1$

How To Formalise This Into Math/Crypto?

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

x

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

w

set of all valid sudoku starters

solution satisfies sudoku rules

The most general formalisation: $x \in L \text{ iff } \exists\ w \text{ s.t. } R(x,w) = 1$

How To Formalise This Into Math/Crypto?

Proof System

A process in which a prover probabilistically convinces a verifier of the correctness of a mathematical proposition

How To Formalise This Into Math/Crypto?

Zero Knowledge **Proof System**

A process in which a prover probabilistically convinces a verifier of the correctness of a mathematical proposition, and the verifier learns nothing else.

How To Formalise This Into Math/Crypto?

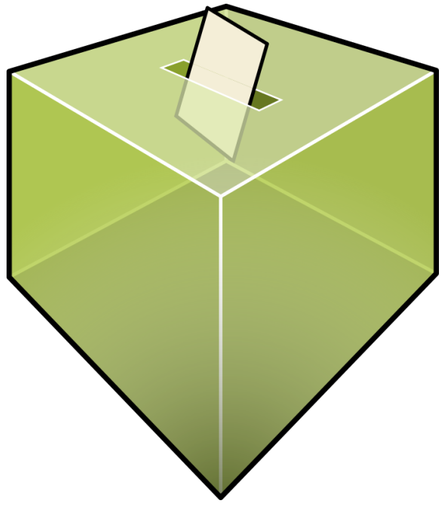
Zero Knowledge **Proof System**

A process in which a prover probabilistically convinces a verifier of the correctness of a mathematical proposition, and the verifier learns nothing else.

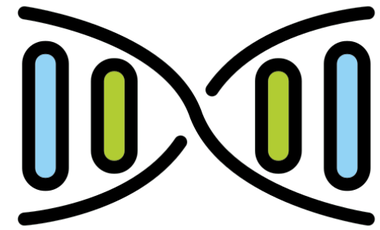
	statement	witness
factoring:	$\exists$ `p,q s.t. N = pq` $\wedge$ `p,q primes`	`(p,q)`
dLog:	$\exists$ `sk s.t. pk =` g^{sk}	`sk`
circuit satisfiability:	$\exists$ `w s.t.` $C_x(w)=1$	`w`

the language `L` is usually implicit in the application, what we make explicit is the relation `R`

Applications of Zero Knowledge Proofs

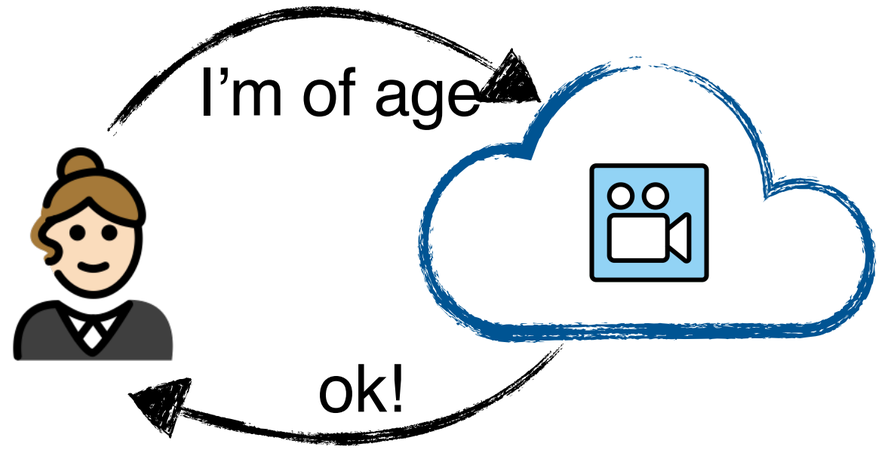


mixing and shufflers for electronic voting
& anonymous communication



DNA matches

anonymous transactions



anonymous credentials

Interactive Zero Knowledge Proofs (of Knowledge)

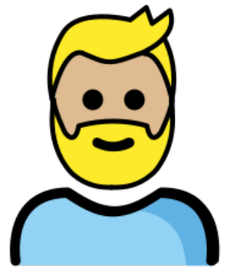
Interactive Zero Knowledge Proofs (of Knowledge)

Prover

unlimited computational power

may try to cheat

Verifier



bounded computational power

always honest

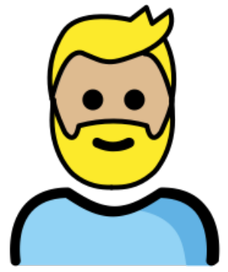
Interactive Zero Knowledge Proofs (of Knowledge)

Prover

unlimited computational power

may try to cheat

Verifier



bounded computational power

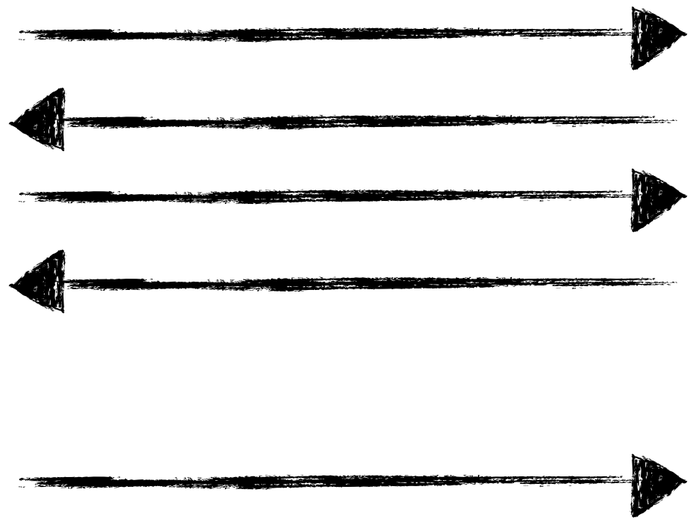
always honest

Interactive Zero Knowledge Proofs (of Knowledge)

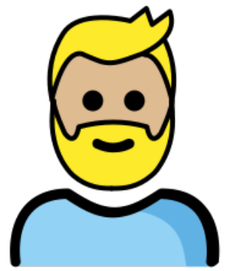
Prover

unlimited computational power

may try to cheat



Verifier



bounded computational power

always honest

I claim : `x ∈ L` because

I know a `w s.t. R(x,w) = 1`

Interactive Zero Knowledge Proofs (of Knowledge)

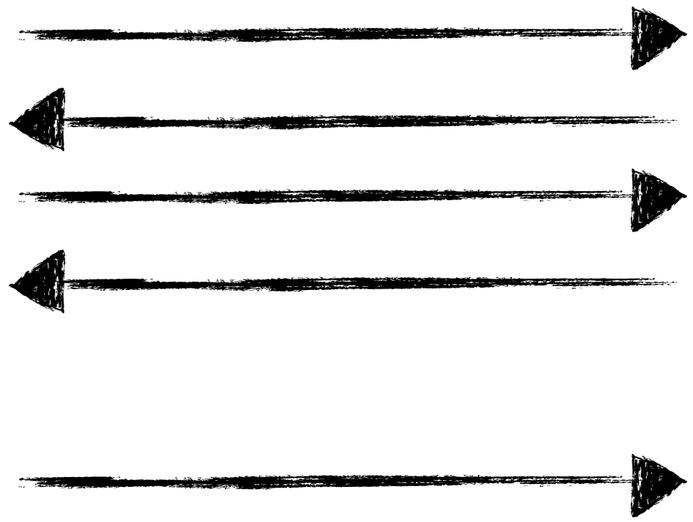
Prover

unlimited computational power

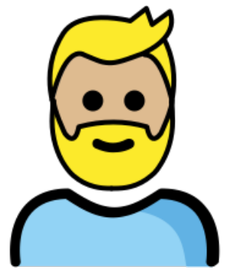
may try to cheat

I claim : `x ∈ L` because

I know a `w s.t. R(x,w) = 1`



Verifier



bounded computational power

always honest

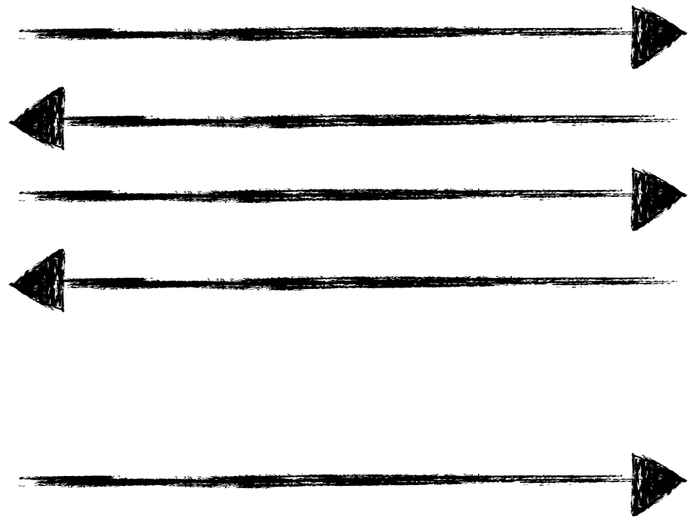
YES /

Interactive Zero Knowledge Proofs (of Knowledge)

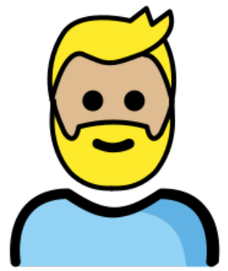
Prover

unlimited computational power

may try to cheat



Verifier



bounded computational power

always honest

I claim : `x ∈ L` because

I know a `w s.t. R(x,w) = 1`

YES /

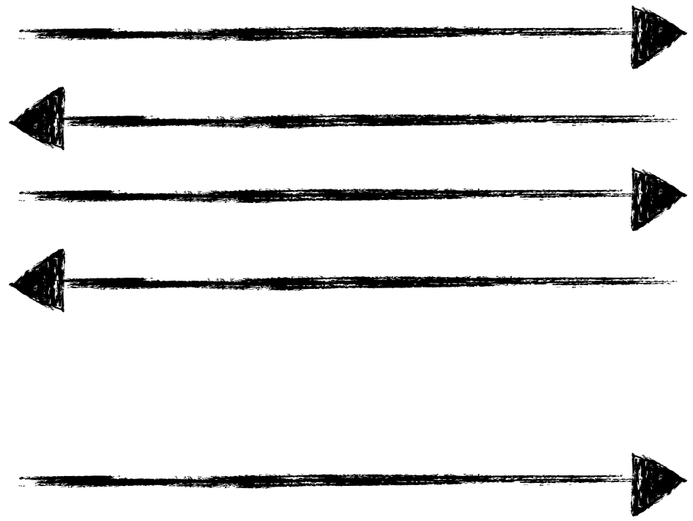
Properties

Interactive Zero Knowledge Proofs (of Knowledge)

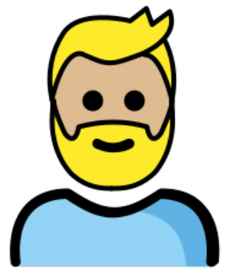
Prover

unlimited computational power

may try to cheat



Verifier



bounded computational power

always honest

YES /

I claim : $x \in L$ because

I know a w s.t. $R(x,w) = 1$

Properties

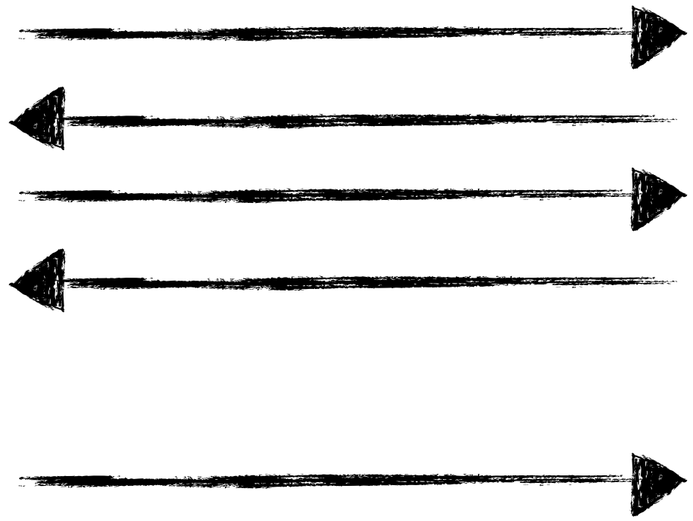
Completeness: if the Prover and the Verifier behave honestly, the proof will be accepted.

Interactive Zero Knowledge Proofs (of Knowledge)

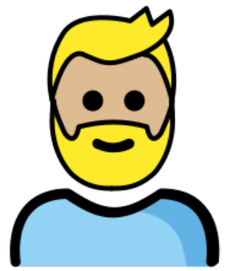
Prover

unlimited computational power

may try to cheat



Verifier



bounded computational power

always honest

I claim : `x ∈ L` because

I know a `w s.t. R(x,w) = 1`

YES /

Properties

Completeness: if the Prover and the Verifier behave honestly, the proof will be accepted.

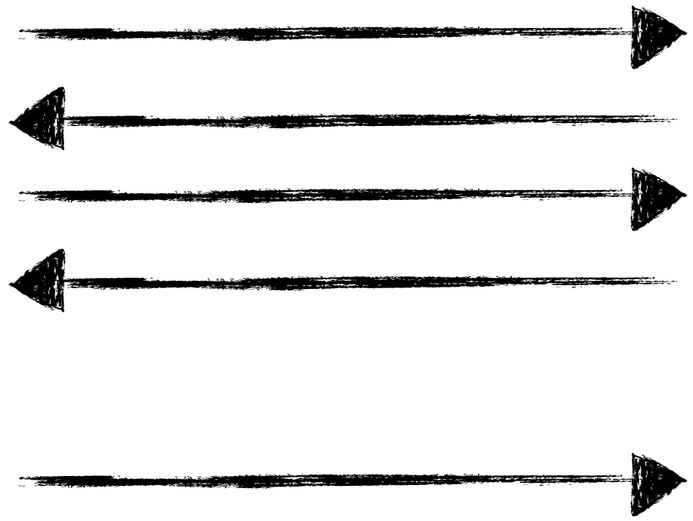
Soundness: the Prover cannot convince the Verifier of false statements.

Interactive Zero Knowledge Proofs (of Knowledge)

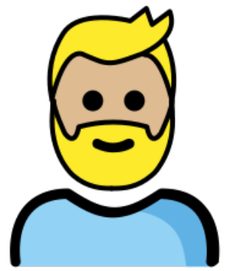
Prover

unlimited computational power

may try to cheat



Verifier



bounded computational power

always honest

YES /

I claim : `x ∈ L` because

I know a `w s.t. R(x,w) = 1`

Properties

Completeness: if the Prover and the Verifier behave honestly, the proof will be accepted.

Soundness: the Prover cannot convince the Verifier of false statements.

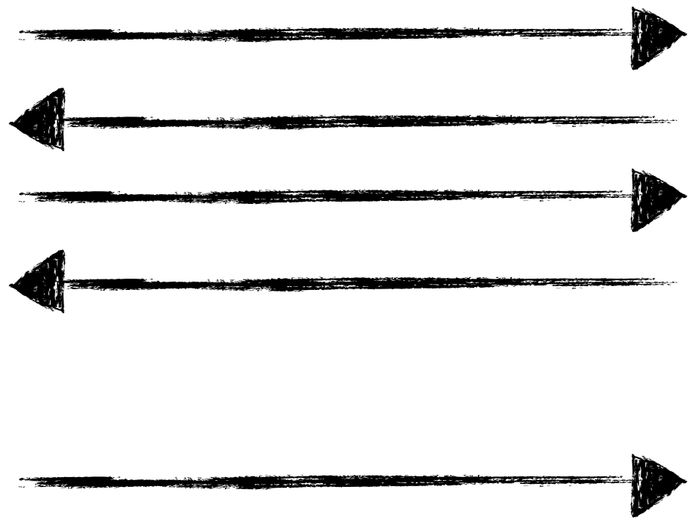
Zero-Knowledge: the Verifier learns nothing beyond the truth of the statement.

Interactive Zero Knowledge Proofs (of Knowledge)

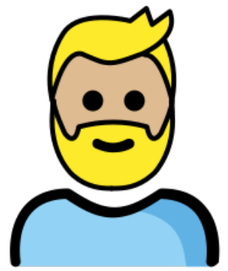
Prover

unlimited computational power

may try to cheat



Verifier



bounded computational power

always honest

I claim : `x ∈ L` because

I know a `w s.t. R(x,w) = 1`

YES /

Properties

Completeness: if the Prover and the Verifier behave honestly, the proof will be accepted.

Soundness: the Prover cannot convince the Verifier of false statements.

Zero-Knowledge: the Verifier learns nothing beyond the truth of the statement.

Of Knowledge: the Verifier is convinced that the Prover knows a witness for the statement being true.

Schnorr Protocol

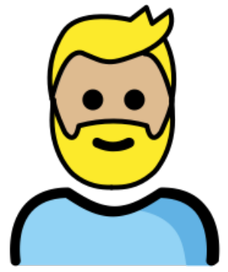
$L = \{ X \in \mathbb{G} \text{ such that } X = g^w, \exists\ w \in \mathbb{Z}_q \}$

Prover

(X, w)

Verifier

X



Schnorr Protocol

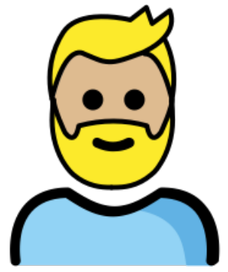
$L = \{ X \in \mathbb{G} \text{ such that } X = g^w, \exists\ w \in \mathbb{Z}_q \}$

Prover

(X,w)

Verifier

X



$r \leftarrow\$ \{0,1,..\ q-1\}$

$R = g^r \in \mathbb{G}$

Schnorr Protocol

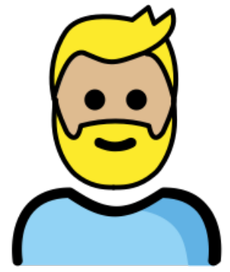
$L = \{ X \in \mathbb{G} \text{ such that } X = g^w, \exists\ w \in \mathbb{Z}_q \}$

Prover

(X,w)

Verifier

X



$r \leftarrow\$ -\{0,1,..\ q-1\}$

$R = g^r \in \mathbb{G}$

R

Schnorr Protocol

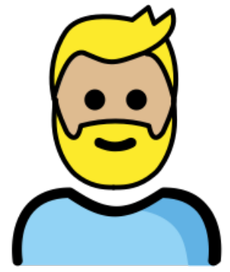
$L = \{ X \in \mathbb{G} \text{ such that } X = g^w, \exists\ w \in \mathbb{Z}_q \}$

Prover

(X,w)

Verifier

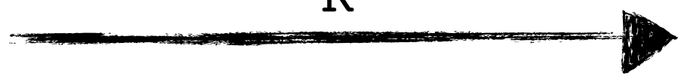
X



$r \leftarrow\$ \{0,1,..\ q-1\}$

$R = g^r \in \mathbb{G}$

R



$c \leftarrow\$ \{0,1,..\ q-1\}$

Schnorr Protocol

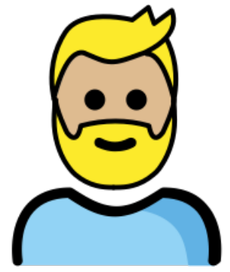
$L = \{ X \in \mathbb{G} \text{ such that } X = g^w, \exists\ w \in \mathbb{Z}_q \}$

Prover

(X,w)

Verifier

X



$r \leftarrow\$ \{0,1,..\ q-1\}$

$R = g^r \in \mathbb{G}$

R

c

c $\leftarrow\$ \{0,1,..\ q-1\}$

Schnorr Protocol

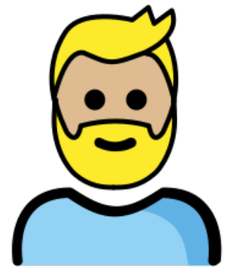
$L = \{ X \in \mathbb{G} \text{ such that } X = g^w, \exists\ w \in \mathbb{Z}_q \}$

Prover

(X,w)

Verifier

X



$r \leftarrow\$ \{0,1,..\ q-1\}$

$R = g^r \in \mathbb{G}$

R

c

c $\leftarrow\$ \{0,1,..\ q-1\}$

$z = c * w + r \in \mathbb{Z}_q$

Schnorr Protocol

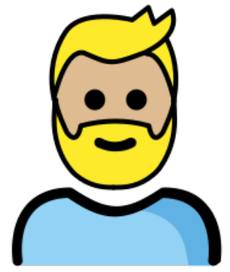
$L = \{ X \in \mathbb{G} \text{ such that } X = g^w, \exists\ w \in \mathbb{Z}_q \}$

Prover

(X,w)

Verifier

X



r ←$-{0,1,.. q-1}

R = $g^r \in \mathbb{G}$

R

c

c ←$-{0,1,.. q-1}

z = c * w + r $\in \mathbb{Z}_q$

z

Schnorr Protocol

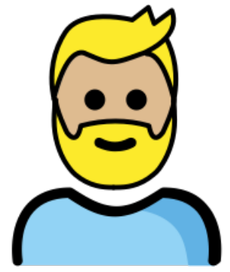
$L = \{ X \in \mathbb{G} \text{ such that } X = g^w, \exists\ w \in \mathbb{Z}_q \}$

Prover

(X,w)

Verifier

X



$r \leftarrow\$ \{0,1,..\ q-1\}$

$R = g^r \in \mathbb{G}$

R →

← c

c ←$-{0,1,.. q-1}

$z = c * w + r \in \mathbb{Z}_q$

z →

g^z =?= R * X^c

Schnorr Protocol

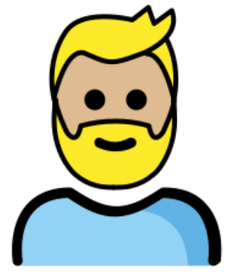
$L = \{ X \in \mathbb{G} \text{ such that } X = g^w, \exists\ w \in \mathbb{Z}_q \}$

Prover

(X,w)

Verifier

X



$r \leftarrow\$ \{0,1,..\ q-1\}$

$R = g^r \in \mathbb{G}$

R →

← c

c ←$-{0,1,.. q-1}

$z = c * w + r \in \mathbb{Z}_q$

z →

g^z =?= R * X^c

Completeness: (easy computation)

Schnorr Protocol

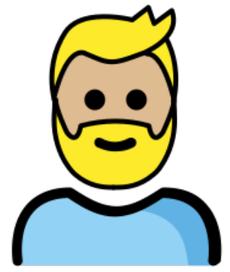
$L = \{ X \in \mathbb{G} \text{ such that } X = g^w, \exists\ w \in \mathbb{Z}_q \}$

Prover

(X,w)

Verifier

X



$r \leftarrow\$ \{0,1,..\ q-1\}$

$R = g^r \in \mathbb{G}$

R →

← c

c ←$-{0,1,.. q-1}

$z = c * w + r \in \mathbb{Z}_q$

z →

g^z =?= R * X^c

Completeness: (easy computation)

Soundness: Follows from the Discrete Logarithm Assumption

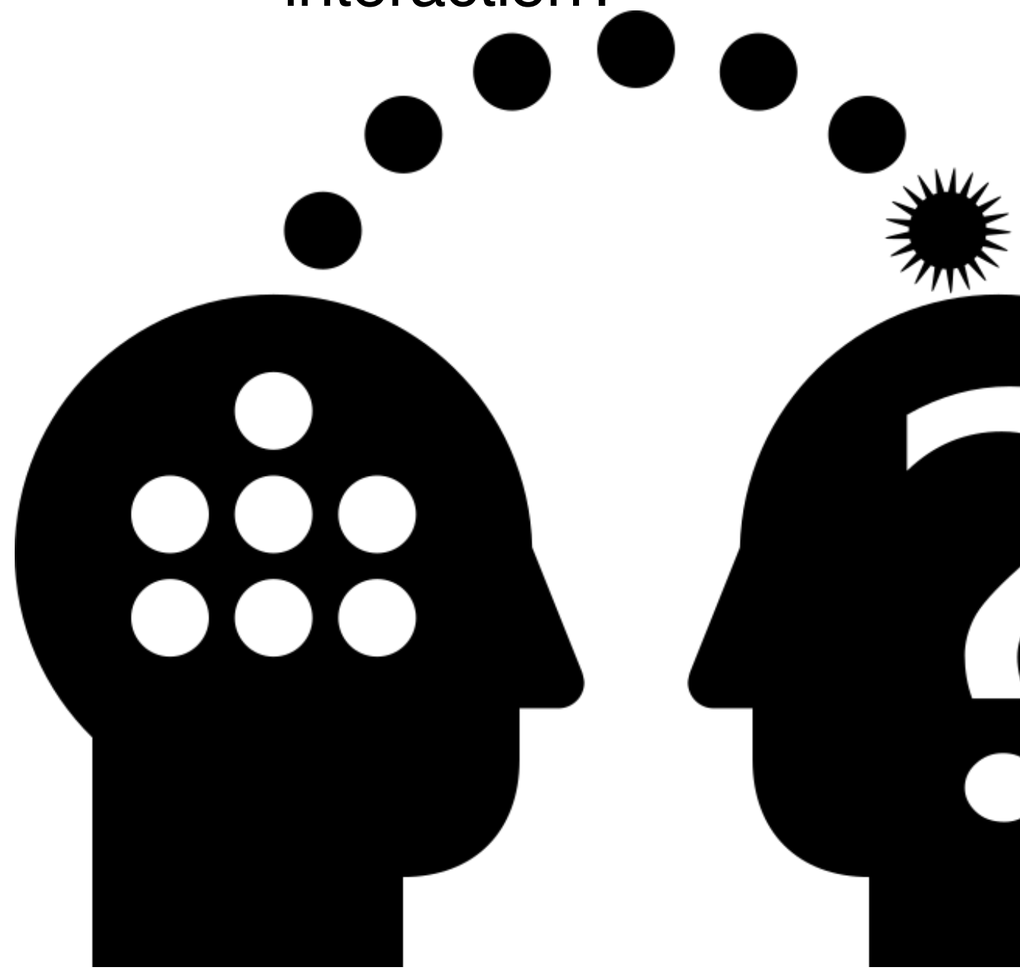
Schnorr Protocol - Proving Zero Knowledge

“the Verifier learns nothing beyond the truth of the statement.”

Schnorr Protocol - Proving Zero Knowledge

“the Verifier learns nothing beyond the truth of the statement.”

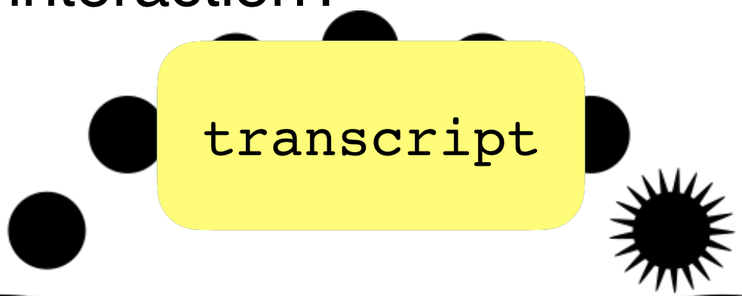
What does it mean to learn nothing from an interaction?

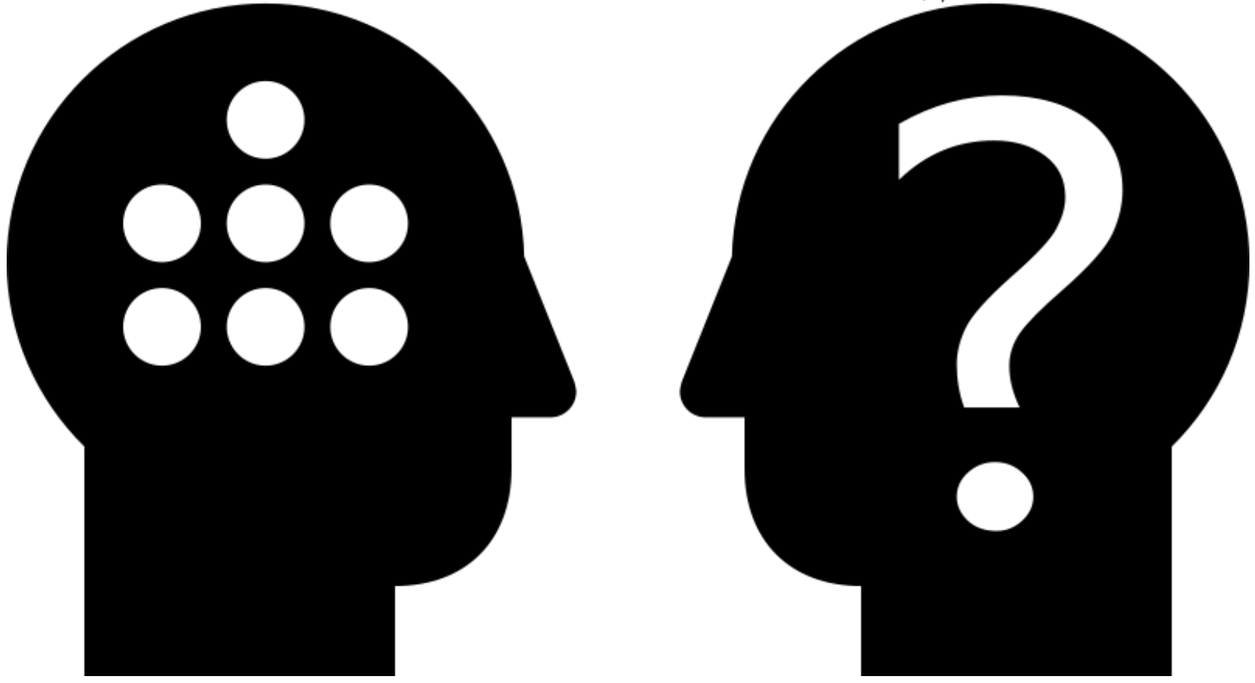


Schnorr Protocol - Proving Zero Knowledge

“the Verifier learns nothing beyond the truth of the statement.”

What does it mean to learn nothing from an interaction?

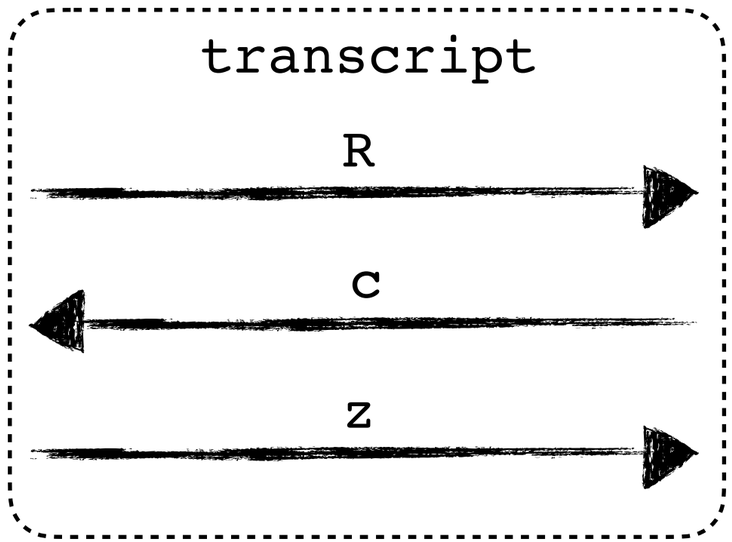




Schnorr Protocol - Proving Zero Knowledge

“the Verifier learns nothing beyond the truth of the statement.”

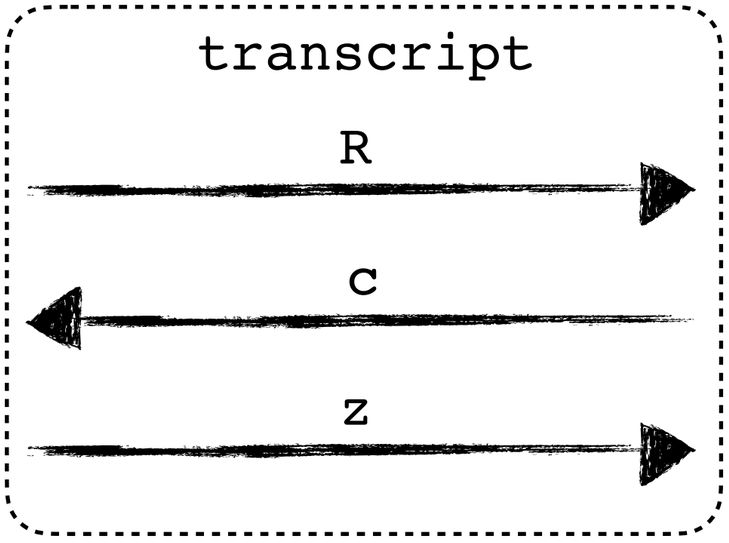
What does it mean to learn nothing from an interaction?



Schnorr Protocol - Proving Zero Knowledge

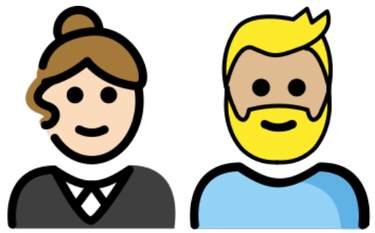
"the Verifier learns nothing beyond the truth of the statement."

What does it mean to learn nothing from an interaction?



Real Interaction

(x,w)



```
r ←$-{0,1,.. q-1}
R = g^r ∈ 𝔾
c ←$-{0,1,.. q-1}
z = c * w + r ∈ ℤ_q
```

Schnorr Protocol - Proving Zero Knowledge

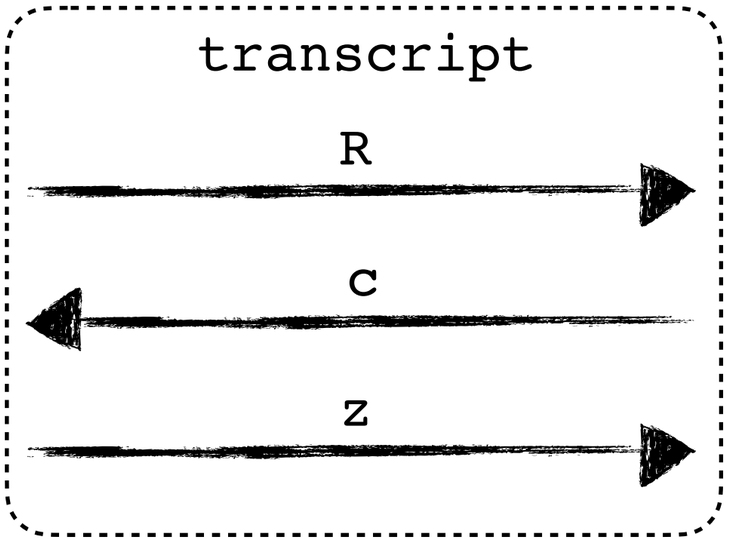
"the Verifier learns nothing beyond the truth of the statement."

What does it mean to learn nothing from an interaction?

Simulator

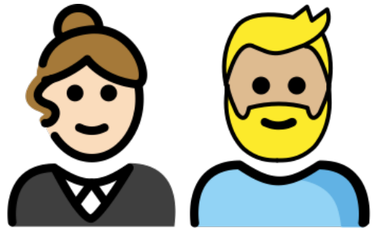
x

```
z ←$-{0,1,.. q-1}
c ←$-{0,1,.. q-1}
R = g^z*X^-c
```



Real Interaction

(x,w)



```
r ←$-{0,1,.. q-1}
R = g^r ∈ 𝔾
c ←$-{0,1,.. q-1}
z = c * w + r ∈ ℤ_q
```

Schnorr Protocol - Proving Zero Knowledge

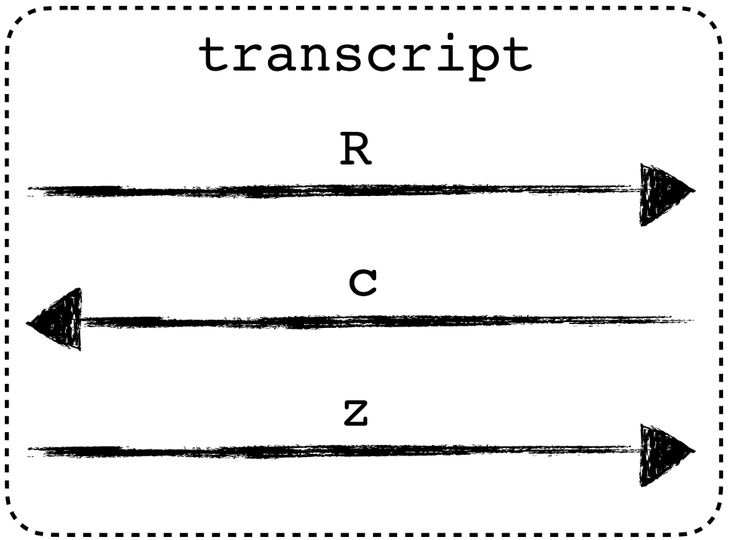
"the Verifier learns nothing beyond the truth of the statement."

What does it mean to learn nothing from an interaction?

Simulator

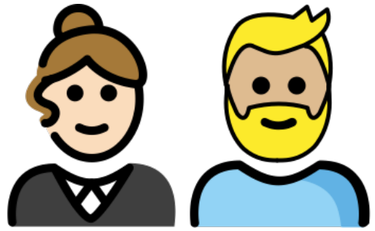
`x`

```
z ←$-{0,1,.. q-1}
c ←$-{0,1,.. q-1}
R = g^z*X^-c
```



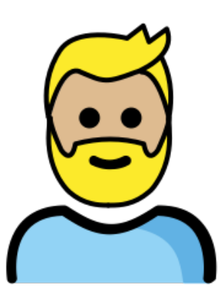
Real Interaction

`(x,w)`



```
r ←$-{0,1,.. q-1}
R = g^r ∈ 𝔾
c ←$-{0,1,.. q-1}
z = c * w + r ∈ ℤ_q
```

`( R , c, z )`



Verifier

`( R , c, z )`

Schnorr Protocol - Proving Zero Knowledge

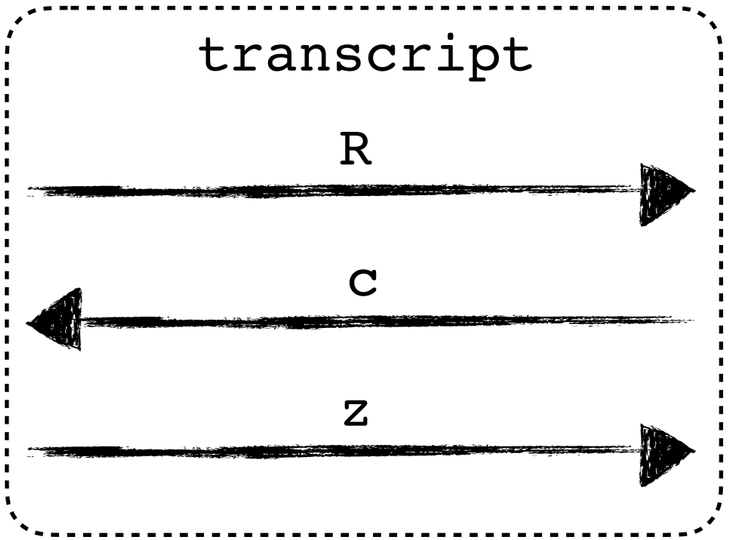
"the Verifier learns nothing beyond the truth of the statement."

What does it mean to learn nothing from an interaction?

Simulator

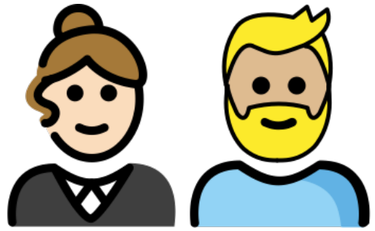
`x`

```
z ←$-{0,1,.. q-1}
c ←$-{0,1,.. q-1}
R = g^z*X^-c
```



Real Interaction

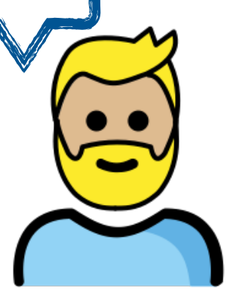
`(x,w)`



```
r ←$-{0,1,.. q-1}
R = g^r ∈ 𝔾
c ←$-{0,1,.. q-1}
z = c * w + r ∈ ℤ_q
```

the transcripts look the same to me

`( R , c, z )`



Verifier

`( R , c, z )`

Schnorr Protocol - Proving Zero Knowledge

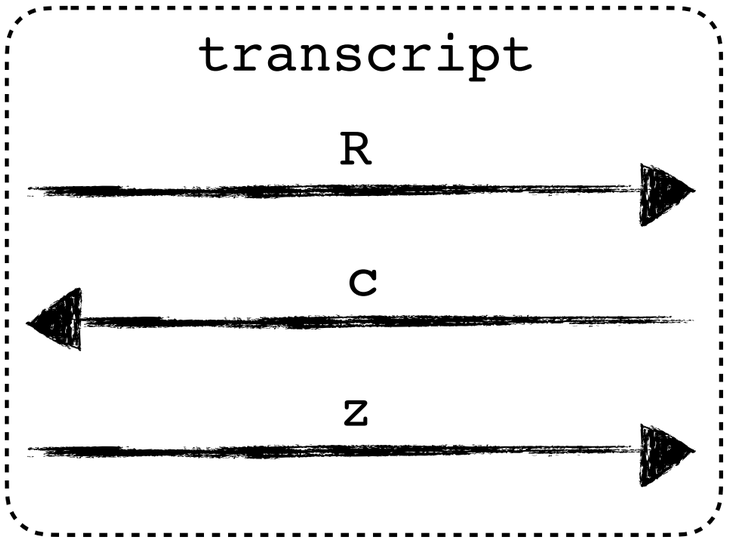
"the Verifier learns nothing beyond the truth of the statement."

What does it mean to learn nothing from an interaction?

Simulator

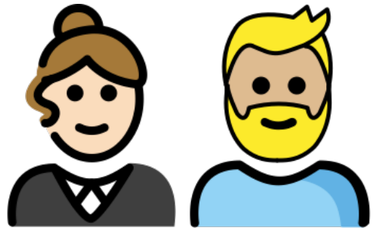
`x`

```
z ←$-{0,1,.. q-1}
c ←$-{0,1,.. q-1}
R = g^z*X^-c
```



Real Interaction

`(x,w)`

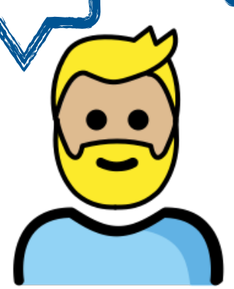


```
r ←$-{0,1,.. q-1}
R = g^r ∈ 𝔾
c ←$-{0,1,.. q-1}
z = c * w + r ∈ ℤ_q
```

the transcripts look the same to me

$\{\text{View}_V[P(w) \leftrightarrow V(x)]\} \sim \{S(x)\}$

`( R , c, z )`



Verifier

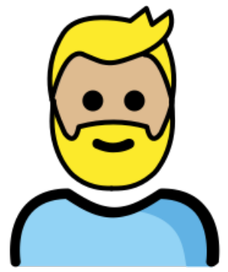
`( R , c, z )`

Schnorr Protocol - Proving Knowledge

"it is possible to extract the witness from the prover"

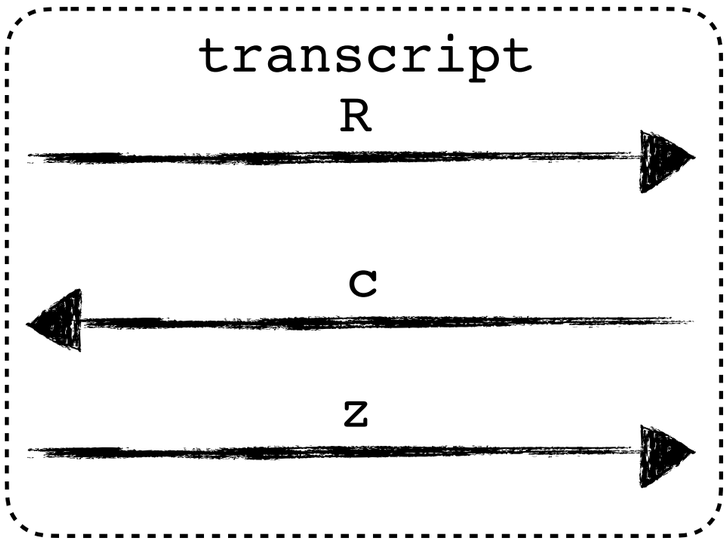
Prover

`(X,w)`



Verifier

`X`



`r ←$-{0,1,.. q-1}`

`R = g^r ∈ 𝔾`

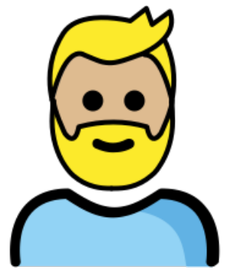
`z = c * w + r ∈` $\mathbb{Z}_q$

Schnorr Protocol - Proving Knowledge

"it is possible to extract the witness from the prover"

Prover

`(X,w)`

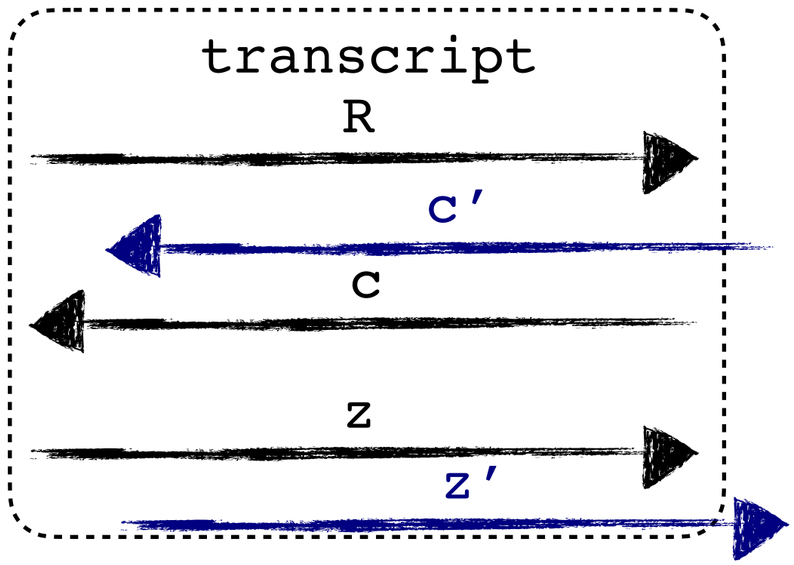


Verifier

`X`

`r ←$-{0,1,.. q-1}`

`R = g^r ∈ 𝔾`



`c' ≠ c`

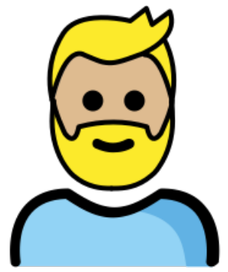
`z = c * w + r ∈` $\mathbb{Z}_q$

Schnorr Protocol - Proving Knowledge

"it is possible to extract the witness from the prover"

Prover

(X,w)

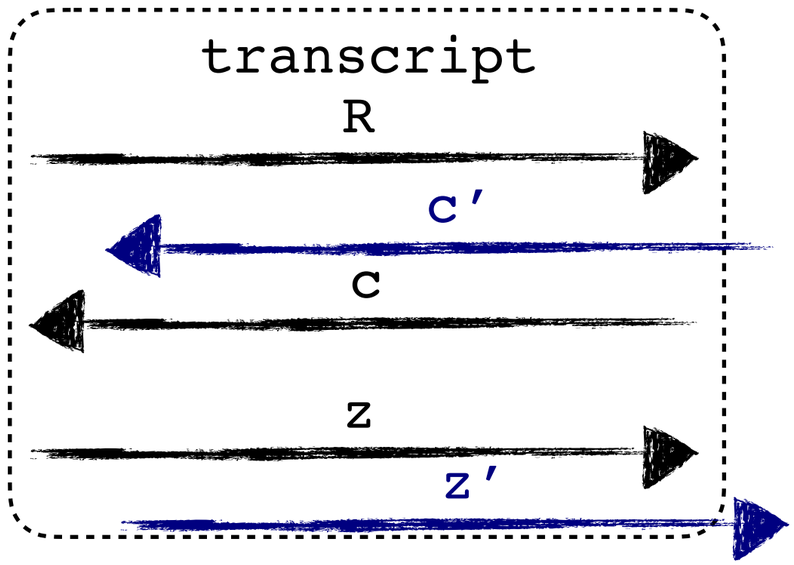


Verifier

X

r ←$-{0,1,.. q-1}

R = $g^r \in \mathbb{G}$



c' ≠ c

z = c * w + r $\in \mathbb{Z}_q$

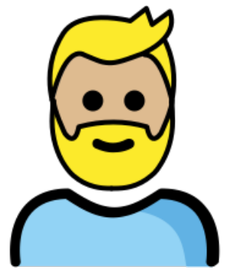
w = z-z'/(c-c')

Schnorr Protocol - Proving Knowledge

"it is possible to extract the witness from the prover"

Prover

(X,w)

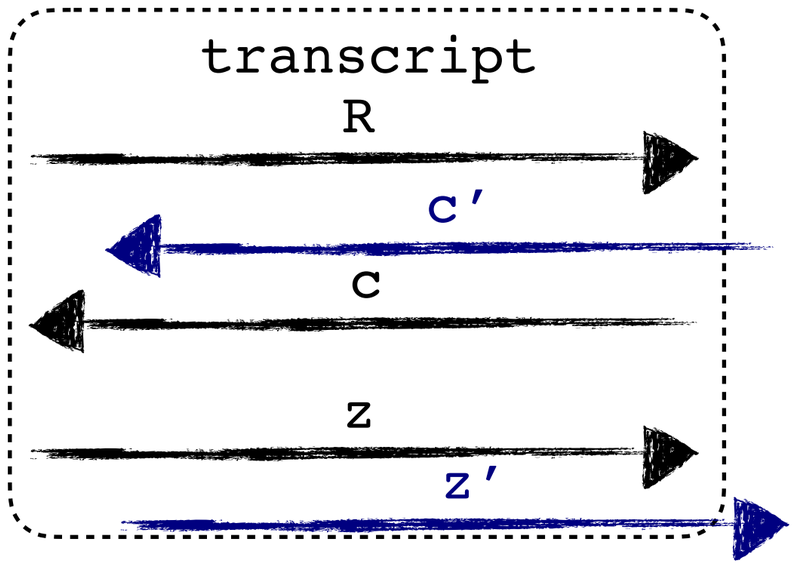


Verifier

X

r ←$-{0,1,.. q-1}

R = $g^r \in \mathbb{G}$



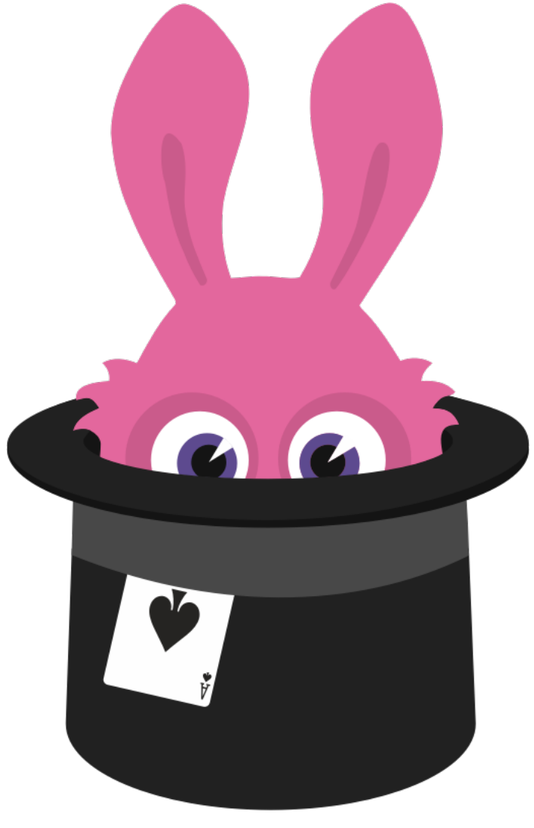
c' ≠ c

z = c * w + r $\in \mathbb{Z}_q$

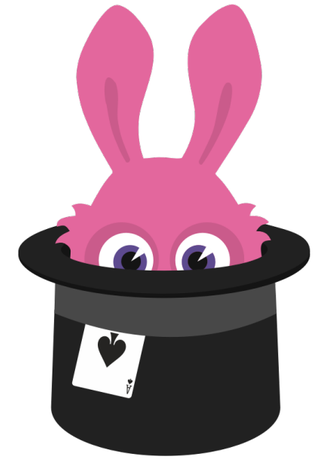
Have we seen something similar before?

w = z-z'/(c-c')

Can we construct a zero-knowledge proof system in which the proof is one single message?

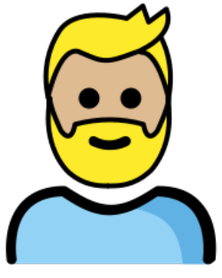


Non Interactive Zero Knowledge Proofs (NIZK)



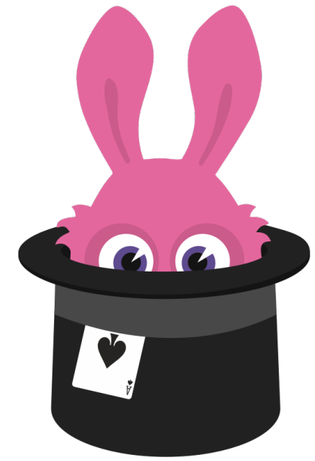
Prover

π



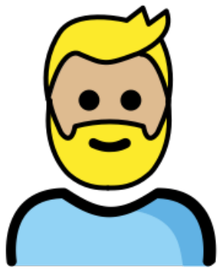
Verifier

Non Interactive Zero Knowledge Proofs (NIZK)



Prover

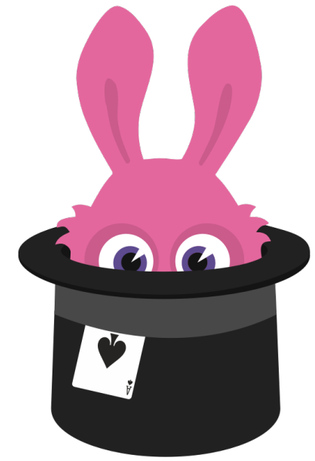
π



Verifier

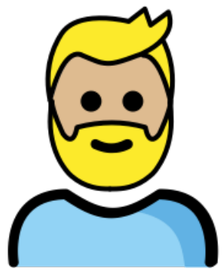
Zero-Knowledge: the Verifier learns nothing beyond the truth of the statement.

Non Interactive Zero Knowledge Proofs (NIZK)



Prover

π

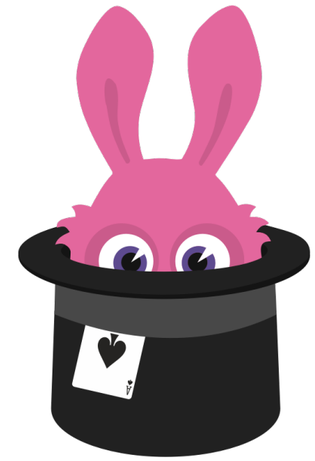


Verifier

Zero-Knowledge: the Verifier learns nothing beyond the truth of the statement.

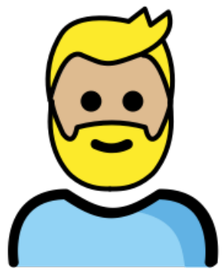
Simulator

Non Interactive Zero Knowledge Proofs (NIZK)



Prover

π



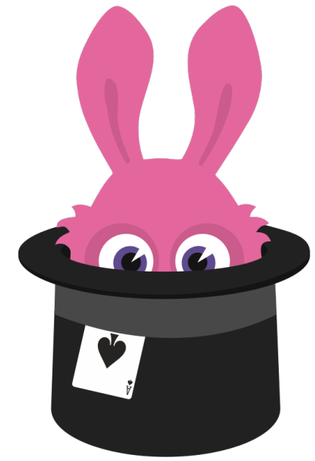
Verifier

Zero-Knowledge: the Verifier learns nothing beyond the truth of the statement.

Simulator

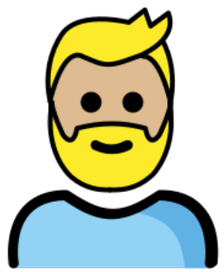
I can produce an accepting π without knowing w!

Non Interactive Zero Knowledge Proofs (NIZK)



Prover

π



Verifier

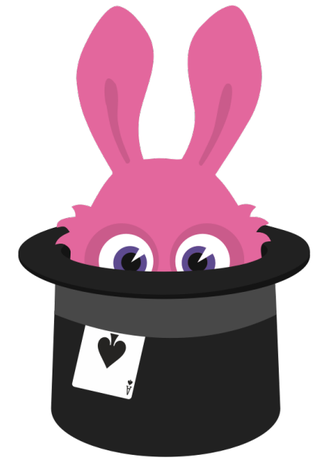
Zero-Knowledge: the Verifier learns nothing beyond the truth of the statement.

Simulator

I can produce an accepting π without knowing w!

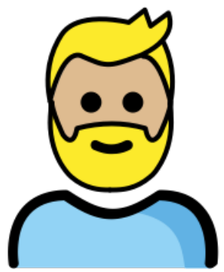
NIZK proofs exist for trivial languages

Non Interactive Zero Knowledge Proofs (NIZK)



Prover

π



Verifier

Zero-Knowledge: the Verifier learns nothing beyond the truth of the statement.

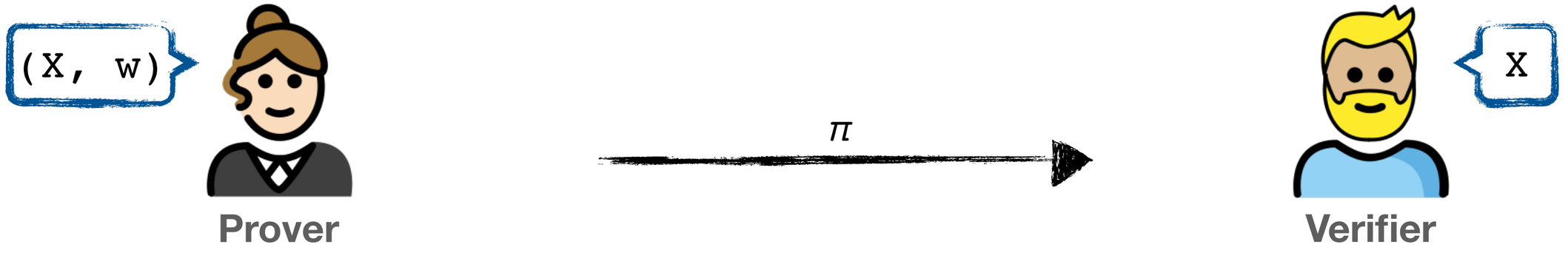
Simulator

I can produce an accepting π without knowing w!

NIZK proofs exist for trivial languages unless we make some extra assumptions

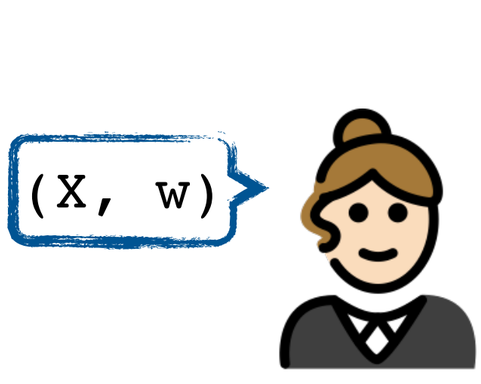
The Fiat-Shamir Heuristic

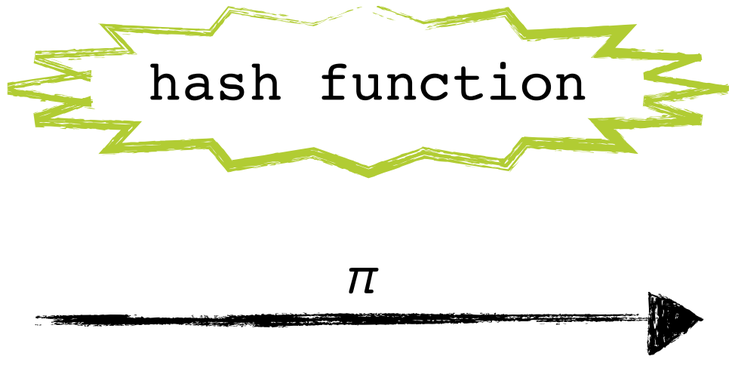
or how to turn an interactive protocol into non-int.

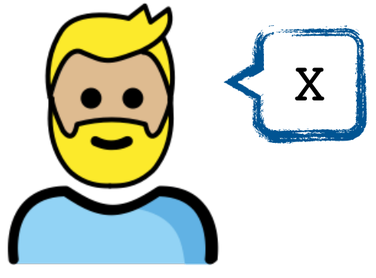


The Fiat-Shamir Heuristic

or how to turn an interactive protocol into non-int.

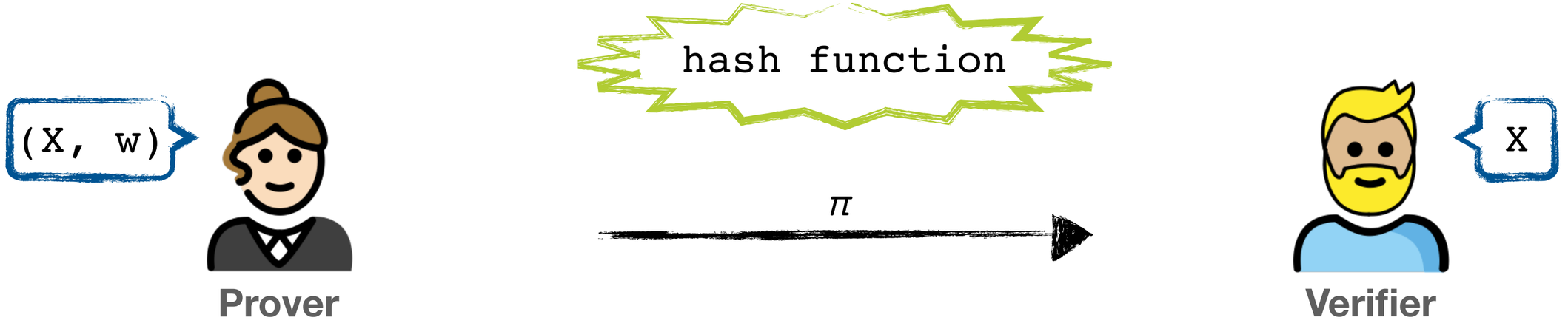






The Fiat-Shamir Heuristic

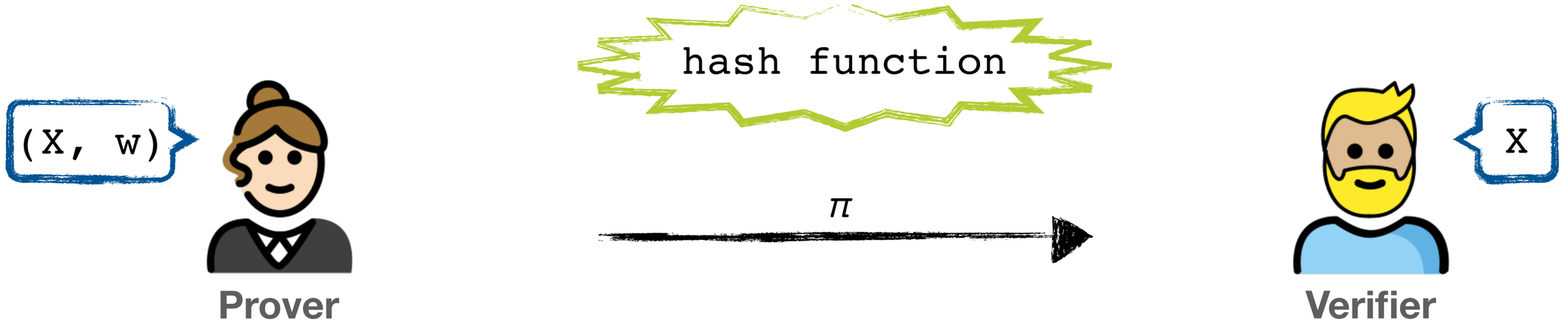
or how to turn an interactive protocol into non-int.



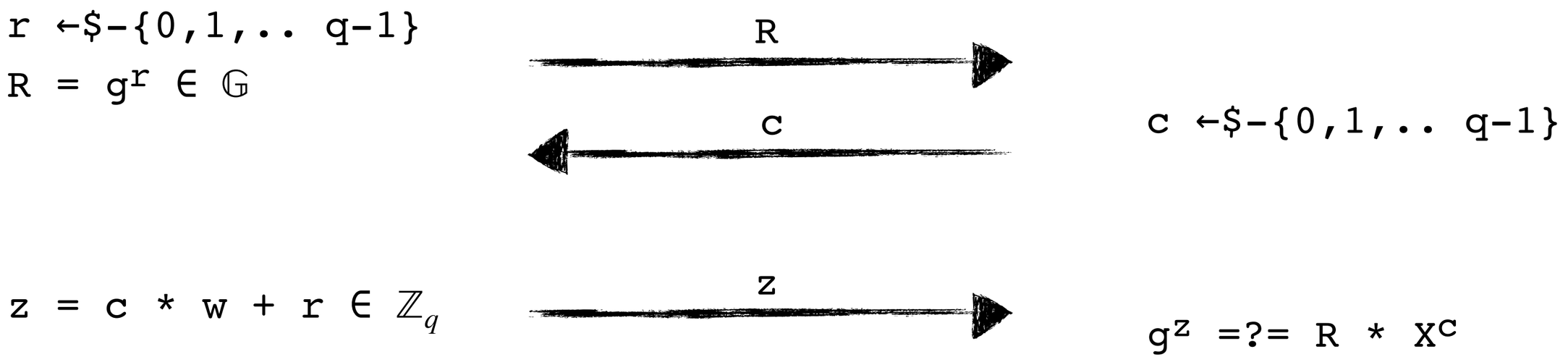
Non Interactive Schnorr Proof

The Fiat-Shamir Heuristic

or how to turn an interactive protocol into non-int.

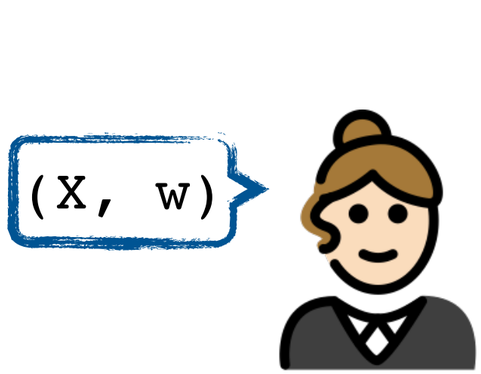


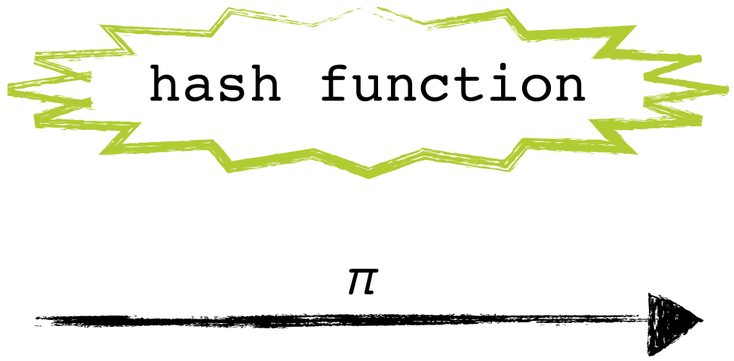
Non Interactive Schnorr Proof

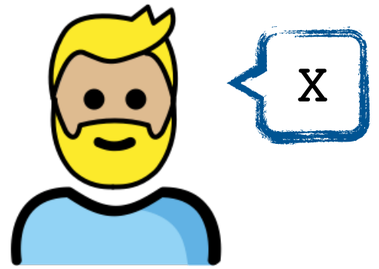


The Fiat-Shamir Heuristic

or how to turn an interactive protocol into non-int.



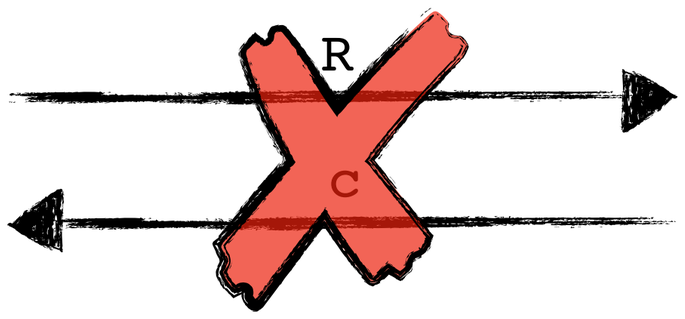




Non Interactive Schnorr Proof

$r \leftarrow\$ \{0,1,..\ q-1\}$

$R = g^r \in \mathbb{G}$



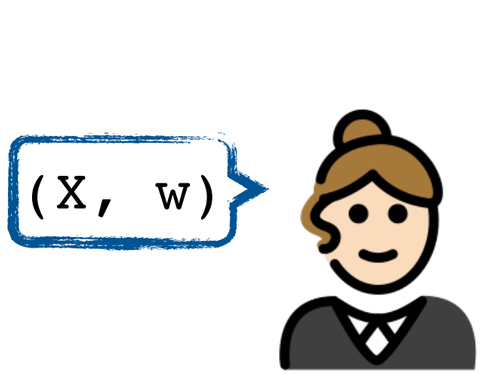
$c \leftarrow\$ \{0,1,..\ q-1\}$

$z = c * w + r \in \mathbb{Z}_q$

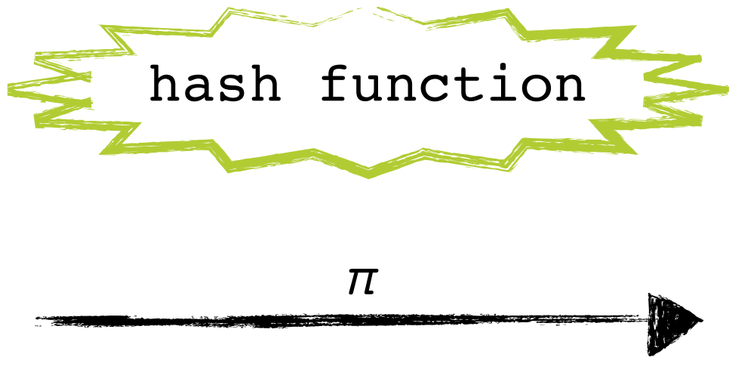
$g^z =?= R * X^c$

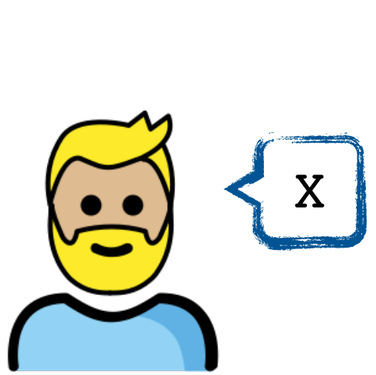
The Fiat-Shamir Heuristic

or how to turn an interactive protocol into non-int.



Prover





Verifier

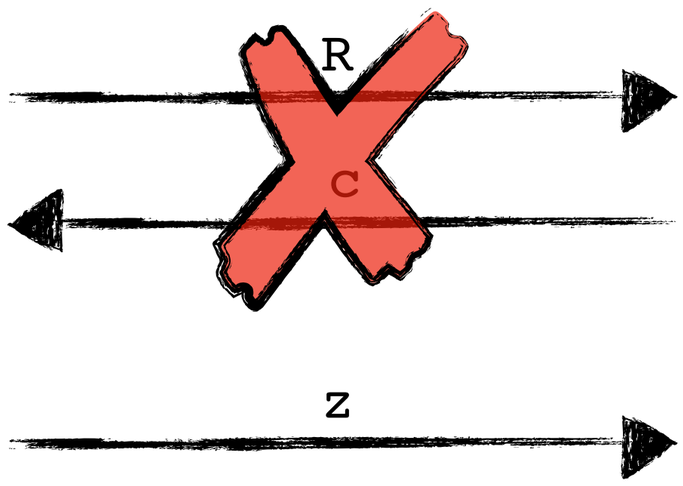
Non Interactive Schnorr Proof

$r \leftarrow\$ \{0,1,..\ q-1\}$

$R = g^r \in \mathbb{G}$

$c = Hash(g,X,R)$

$z = c * w + r \in \mathbb{Z}_q$

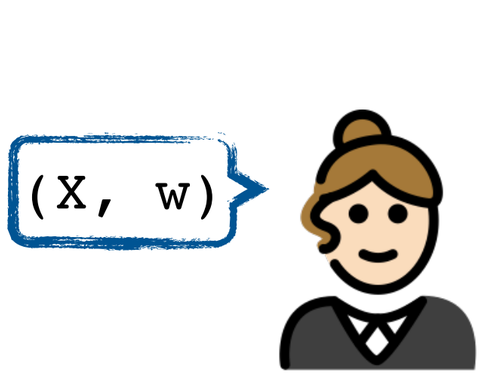


$c \leftarrow\$ \{0,1,..\ q-1\}$

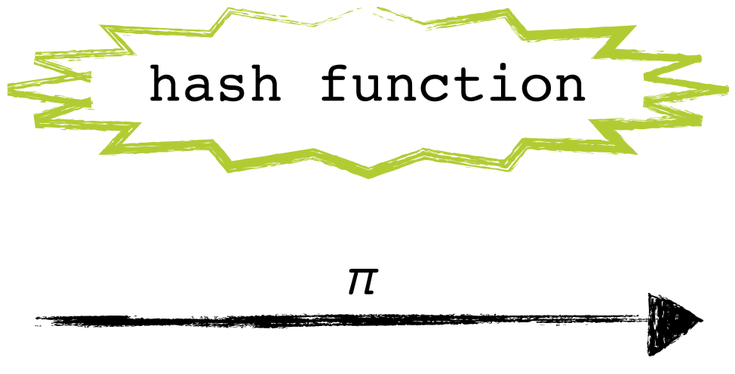
$g^z =?= R * X^c$

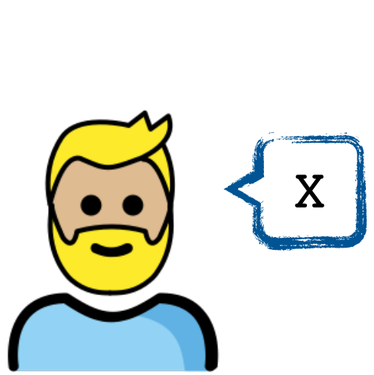
The Fiat-Shamir Heuristic

or how to turn an interactive protocol into non-int.



Prover





Verifier

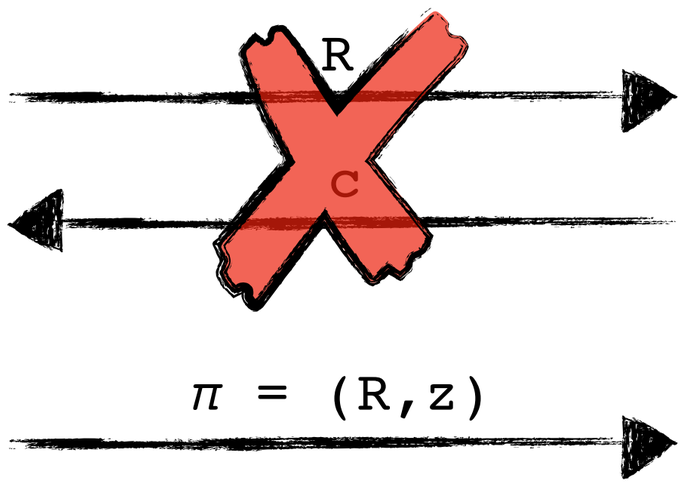
Non Interactive Schnorr Proof

`r ←$-{0,1,.. q-1}`

`R = g^r ∈ 𝔾`

`c = Hash(g,X,R)`

`z = c * w + r ∈` $\mathbb{Z}_q$

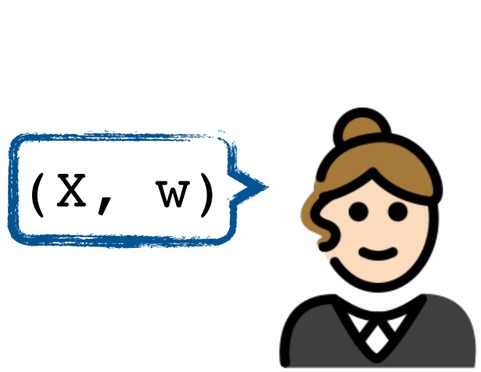


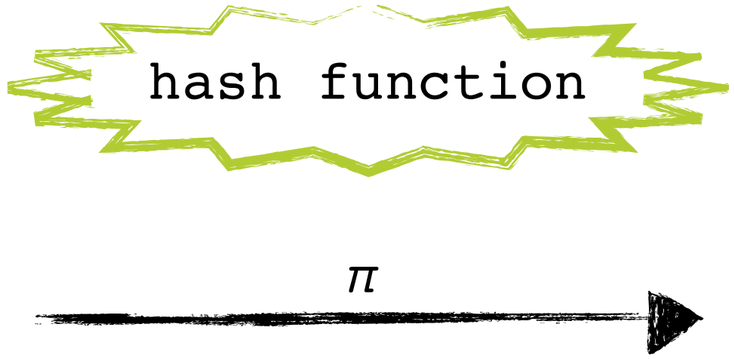
`c ←$-{0,1,.. q-1}`

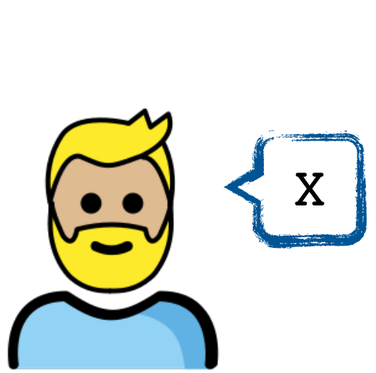
`g^z =?= R * X^c`

The Fiat-Shamir Heuristic

or how to turn an interactive protocol into non-int.







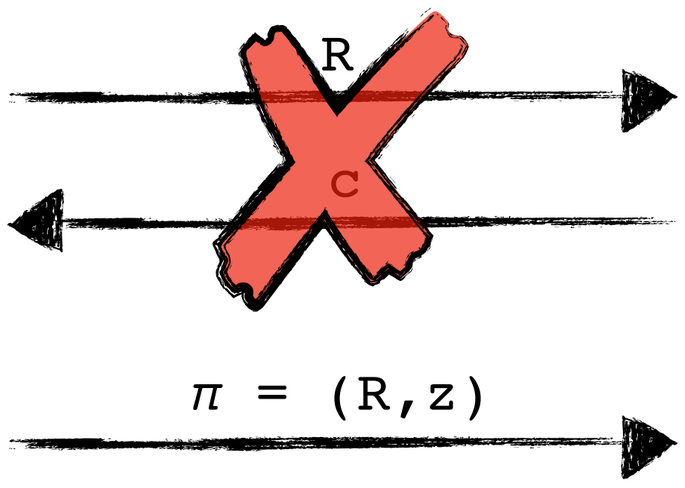
Non Interactive Schnorr Proof

$r \leftarrow\$ -\{0,1,..\ q-1\}$

$R = g^r \in \mathbb{G}$

$c = Hash(g,X,R)$

$z = c * w + r \in \mathbb{Z}_q$



$c \leftarrow\$ -\{0,1,..\ q-1\}$

$c = Hash(g,X,R)$

$g^z =?= R * X^c$

Have We Seen This Before?

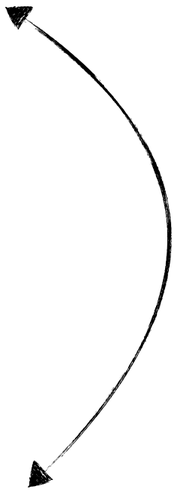
Have We Seen This Before?

```
r ←$-{0,1,.. q-1}
R = g^r ∈ 𝔾
c = Hash(g,X,R)
z = c * w + r ∈ ℤ_q
```

```
Sign(sk, msg) ⇨ sgn
k ←$── [0 ... n-1]
R = k*G
r = R_x mod n
z = sha256(msg)
s = inv(k)·(z + d·r) mod n
sgn = (r, s)
```

Have We Seen This Before?

```
r ←$-{0,1,.. q-1}
R = g^r ∈ 𝔾
c = Hash(g,X,R)
z = c * w + r ∈ ℤ_q
```

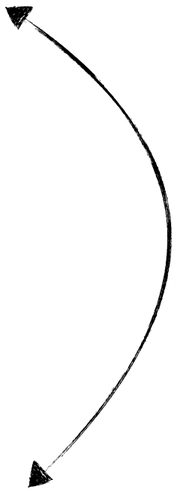


similar structure:

```
Sign(sk, msg) ⇨ sgn
k ←$— [0 ... n-1]
R = k*G
r = R_x mod n
z = sha256(msg)
s = inv(k)·(z + d·r) mod n
sgn = (r, s)
```

Have We Seen This Before?

```
r ←$-{0,1,.. q-1}
R = g^r ∈ 𝔾
c = Hash(g,X,R)
z = c * w + r ∈ ℤ_q
```



similar structure:

1. pick randomness

```
Sign(sk, msg) ⇨ sgn
k ←$— [0 ... n-1]
R = k*G
r = R_x mod n
z = sha256(msg)
s = inv(k)·(z + d·r) mod n
sgn = (r, s)
```

Have We Seen This Before?

```
r ←$-{0,1,.. q-1}
R = g^r ∈ 𝔾
c = Hash(g,X,R)
z = c * w + r ∈ ℤ_q
```

```
Sign(sk, msg) ⇨ sgn
k ←$— [0 ... n-1]
R = k*G
r = R_x mod n
z = sha256(msg)
s = inv(k)·(z + d·r) mod n
sgn = (r, s)
```

similar structure:

1. pick randomness
2. generate new (unpredictable) randomness using the hash function

Have We Seen This Before?

```
r ←$-{0,1,.. q-1}
R = g^r ∈ 𝔾
c = Hash(g,X,R)
z = c * w + r ∈ ℤ_q
```

```
Sign(sk, msg) ⇨ sgn
k ←$— [0 ... n-1]
R = k*G
r = R_x mod n
z = sha256(msg)
s = inv(k)·(z + d·r) mod n
sgn = (r, s)
```

similar structure:

1. pick randomness
2. generate new (unpredictable) randomness using the hash function
3. use the secret and hide it with the randomness

Have We Seen This Before?

```
r ←$-{0,1,.. q-1}
R = g^r ∈ 𝔾
c = Hash(g,X,R)
z = c * w + r ∈ ℤ_q
```

```
Sign(sk, msg) ⇨ sgn
k ←$— [0 ... n-1]
R = k*G
r = R_x mod n
z = sha256(msg)
s = inv(k)·(z + d·r) mod n
sgn = (r, s)
```

similar structure:

1. pick randomness
2. generate new (unpredictable) randomness using the hash function
3. use the secret and hide it with the randomness
4. return a proof of knowledge of a secret value

Zk-SNARK

Zero Knowledge Succinct Non-interactive ARguments of Knowledge

Zk-SNARK

Zero Knowledge Succinct Non-interactive ARguments of Knowledge

$|\pi| = o(|w|)$

Schnorr: π = (R,z), |π| > |w|

Zk-SNARK

Zero Knowledge Succinct Non-interactive ARguments of Knowledge

$|\pi| = o(|w|)$

Schnorr: π = (R,z), |π| > |w|

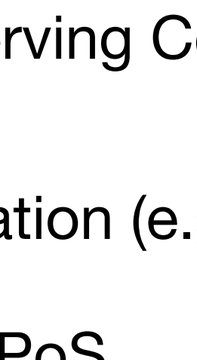
what?

Privacy-Preserving Coins (Zcash)

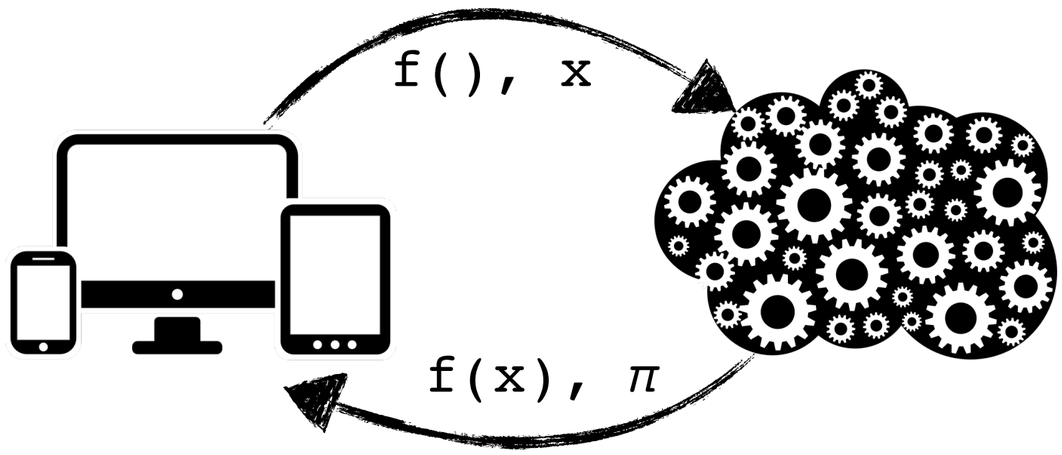
Ledger Verification (e.g. Coda)

Privacy-Preserving PoS
(e.g. Ouroboros Crypsinous)

Privacy-Preserving
Smart Contracts
(Hawk, Gyges)



Verifiable Computations



Verifiable Document Redacting

Zk-SNARK

Zero Knowledge Succinct Non-interactive ARguments of Knowledge

$|\pi| = o(|w|)$

Schnorr: π = (R,z), |π| > |w|

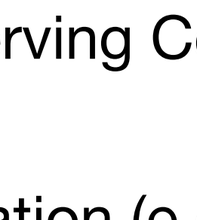
what?

Privacy-Preserving Coins (Zcash)

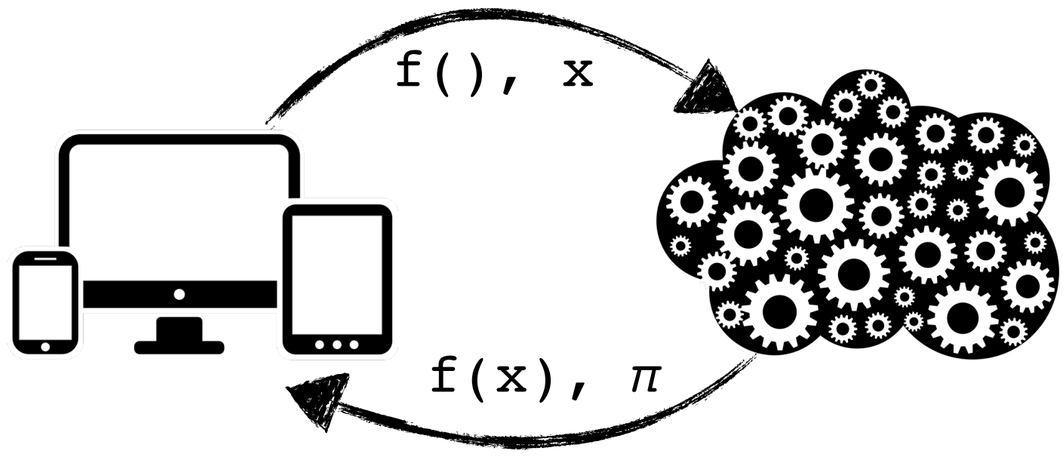
Ledger Verification (e.g. Coda)

Privacy-Preserving PoS
(e.g. Ouroboros Crypsinous)

Privacy-Preserving
Smart Contracts
(Hawk, Gyges)



Verifiable Computations



Verifiable Document Redacting

how? elliptic curves, pairings, R1CS, QAP + non-falsifiable assumptions (KoE)