Blockchain

PoS and Committees

DAT655 Leander Jehl

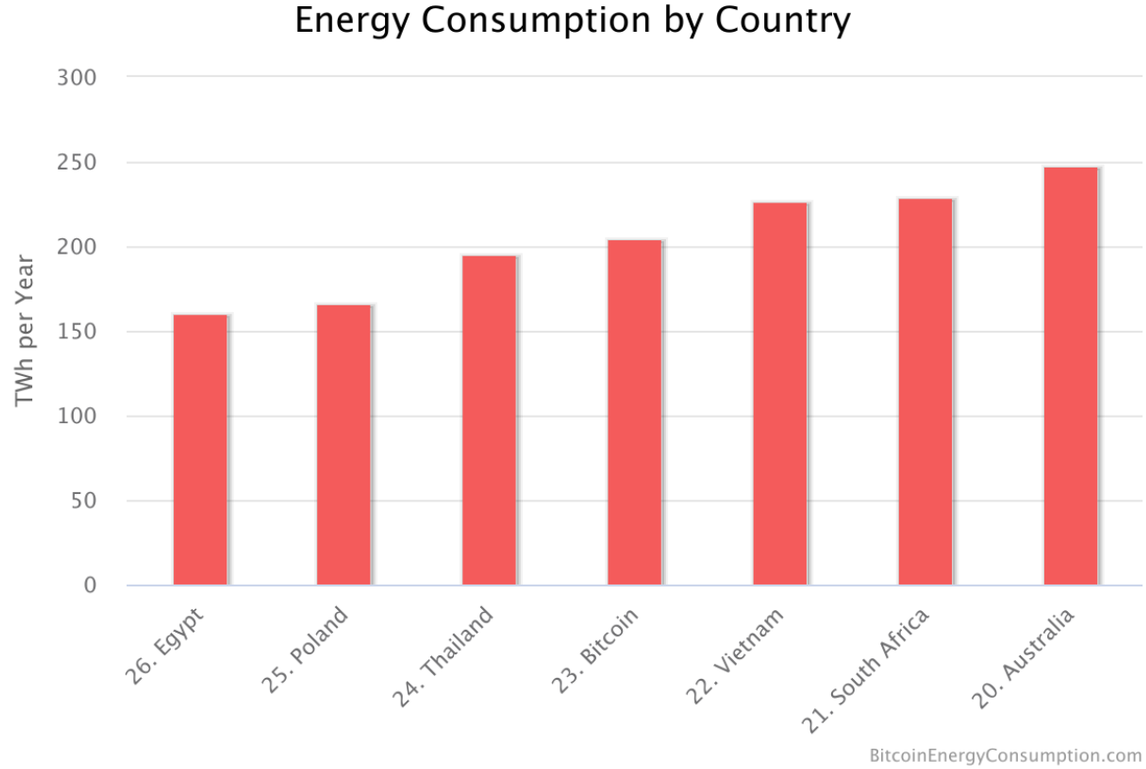
Bitcoin

Downsides

Throughput at most 7tx per second

Confirmation latency approx 1h

Enormous energy consumption



System models

Permissioned vs unpermissioned

Unpermissioned:
A system that anyone can join, usually anonymously.
Example: Bitcoin, Ethereum, PoW or PoStake blockchain

- In unpermissioned systems, typically sybils pose a problem

Permissioned: Need permission to join a network. System comprised of nodes with known identity.

Example:

- Several organisations running a system together, each running one servers.

Permissioned systems

Examples

Try to ask the chat.

Permissioned systems

Identities:

- Each peer is uniquely identified.
- Group of initial peers is created offline.
- List of peers exist, including a public key for each peer.

System models

Permissioned vs unpermissioned

Unpermissioned:
A system that anyone can join, usually anonymously.
Example: Bitcoin, Ethereum, PoW or PoStake blockchain

- In unpermissioned systems, typically sybils pose a problem
- PoW: Sybils identities do not matter, since the amount of hashing power must be devided.

Alternatives to Proof of Work

Alternatives to PoW

Challenges

Requirements

- **Open membership**
- **Large and diverse group of members**

Attacks

- **Sybil attack**
- **Aggregation of members (mining pools)**

Alternatives to PoW

Proof of X

- **Proof of useful work**
  - **Trying to compute useful things in PoW.**
- **Proof of authority**
  - **One or multiple trusted nodes append blocks.**
- **Proof of storage**
  - **One disc one vote**
- **Proof of elapsed time**
  - **One TEE one vote**
- **Proof of stake**
  - **One cryptocoin one vote**

Proof of Stake

Proof of Stake

Idea

- **Lock some amount of funds (stake) to become eligible for creating a block and receiving a reward.**
- **Stake is locked by issuing a special transaction.**
- **Unlocked after a given time.**

Proof of Stake

PeerCoin

A nodes with $addr$ and coin($addr$) much stake can create a new block if:

$$H(\texttt{prevblockhash}||addr||\texttt{timeinsec}) < d \cdot \textbf{coin}(addr)$$

as hexadecimal number

- **d is a base difficulty (hex number)**
- **coin($addr$) addjusts difficulty based on stake**

One try to solve PoW per second.

Distributing stake to multiple agents does not give benefit.

Proof of Stake

PeerCoin

A nodes with $addr$ and coin($addr$) much stake can create a new block if:

$$H(\texttt{prevblockhash}||addr||\texttt{timeinsec}) < d \cdot \textbf{coin}(addr)$$

Pros:

- **Energy efficient**
- **Easy to participate (**no special hardware)

Proof of Stake

PeerCoin

A nodes with $addr$ and coin($addr$) much stake can create a new block if:

$$H(\texttt{prevblockhash}||addr||\texttt{timeinsec}) < d \cdot \textbf{coin}(addr)$$

Cons:

- **Predictability (**look in the future**)**
- **Nothing at stake (**Can work on 2 forks**)**
- **Possibly unfair (**rich get richer**)**
- **Possible to PoW (**stake grinding**)**
- **History rewrite (**Long range attacks**)**

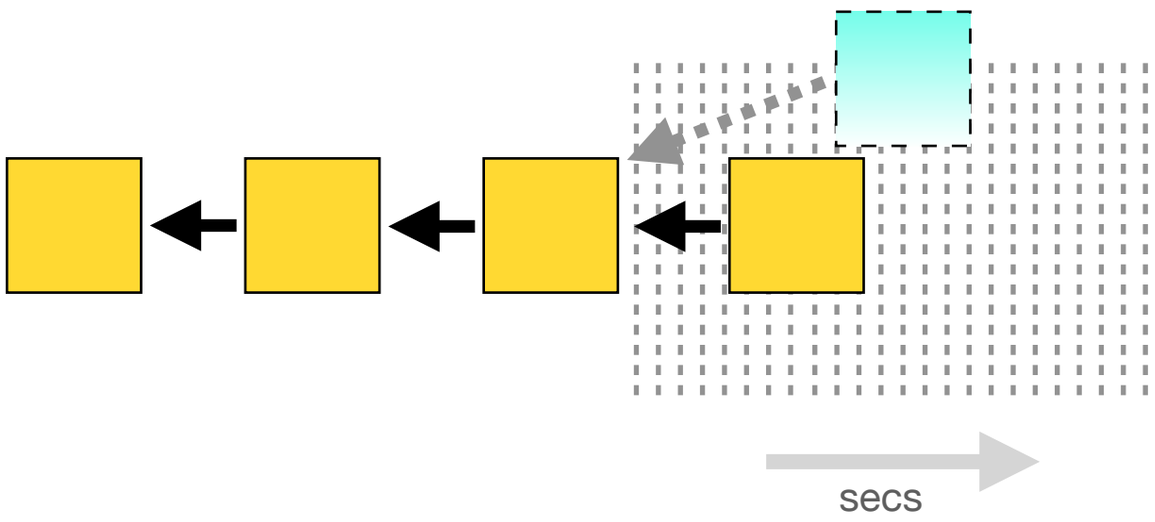
Proof of Stake

PeerCoin

$$H(\texttt{prevblockhash}||addr||\texttt{timeinsec}) < d \cdot \textbf{coin}(addr)$$

Predictability (look in the future)

- Can advance `timeinsec` faster than time.



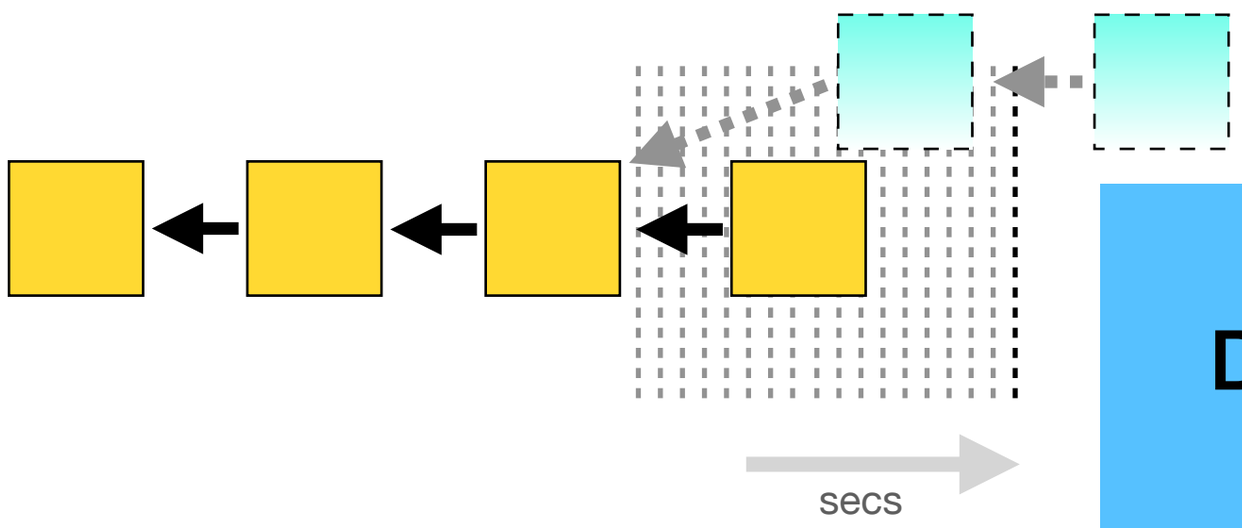
Proof of Stake

PeerCoin

$$H(\texttt{prevblockhash}||addr||\texttt{timeinsec}) < d \cdot \textbf{coin}(addr)$$

Predictability (look in the future)

- Can advance `timeinsec` faster than time.
- Can create longest chain in the future



Do not accept blocks in the future.

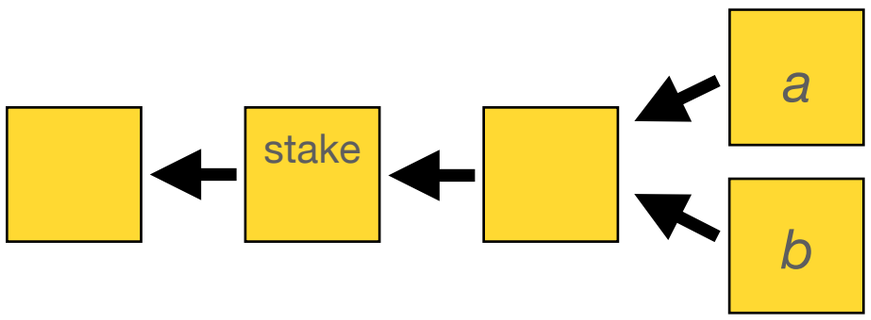
Proof of Stake

PeerCoin

$$H(\texttt{prevblockhash}||addr||\texttt{timeinsec}) < d \cdot \textbf{coin}(addr)$$

Nothing at stake

- Can work on 2 forks if they both include your stake



Slashing:
Punish nodes for misbehaviour
e.g. by taking their stake

every second, try to extend *a* and *b*

no, or only slow decision

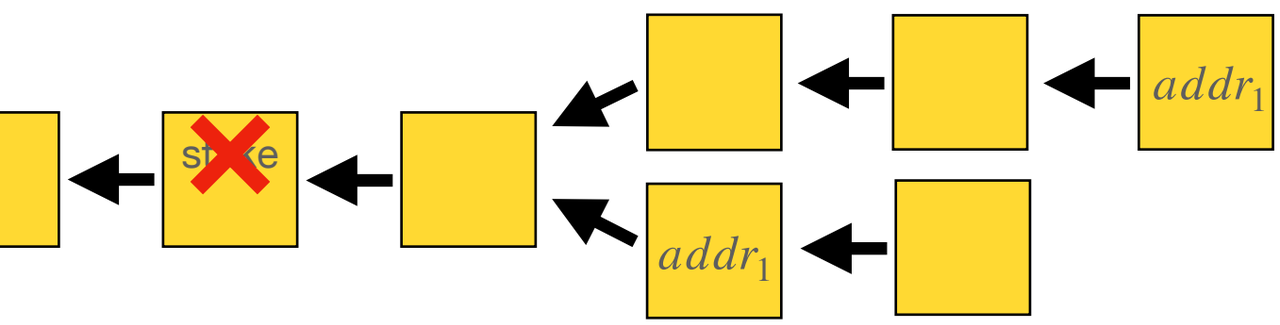
Proof of Stake

PeerCoin

$$H(\texttt{prevblockhash}||addr||\texttt{timeinsec}) < d \cdot \textbf{coin}(addr)$$

Nothing at stake

- Can work on 2 forks if they both include your stake



Slashing:
Punish nodes for misbehaviour
e.g. by taking their stake

Proof of Stake

PeerCoin

$$H(\texttt{prevblockhash}||addr||\texttt{timeinsec}) < d \cdot \textbf{coin}(addr)$$

Slashing

- If nodes misbehave they loose their deposit
- Lost deposit can be
  - destroyd (burned)
  - given to other nodes, e.g. the one reporting misbehaviour
- Deposit needs to be frozen long enough to detect misbehaviour
- Nothing at stake still possible with multiple addresses.
- Blocks need to be signed, to avoid someone else causing slashing.

Proof of Stake

PeerCoin

$$H(\texttt{prevblockhash}||addr||\texttt{timeinsec}) < d \cdot \textbf{coin}(addr)$$

Possibly unfair

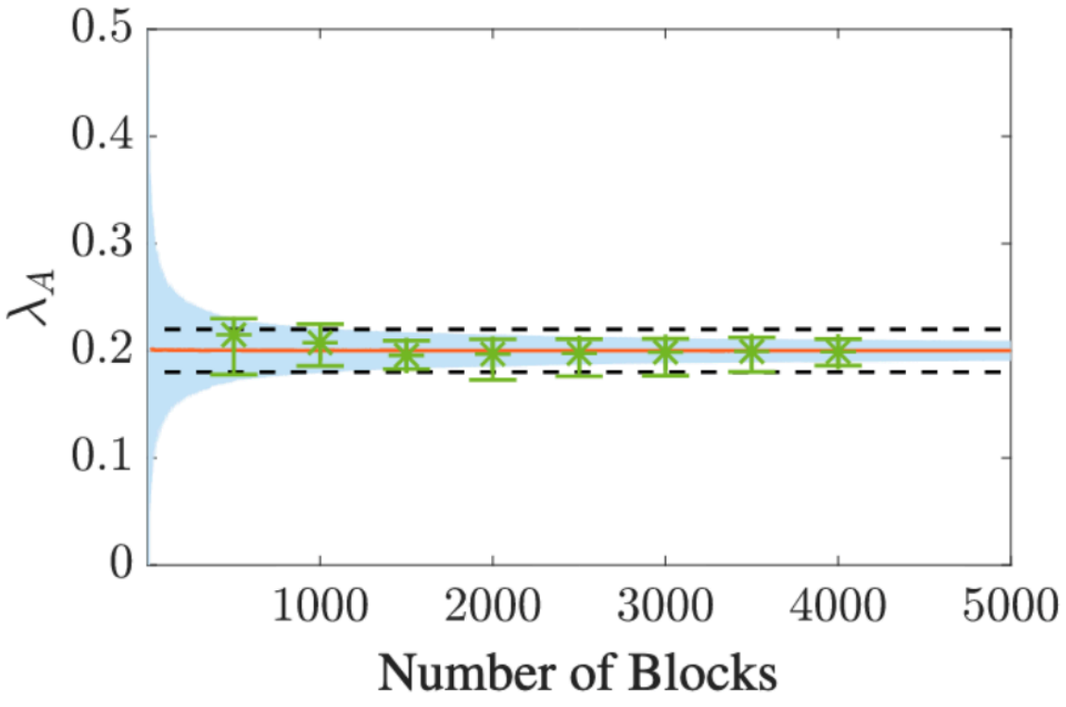
- Miner receiving first reward gets an advantage.
- Reward distribution is more likely to diverge than in PoW.

Proof of Stake

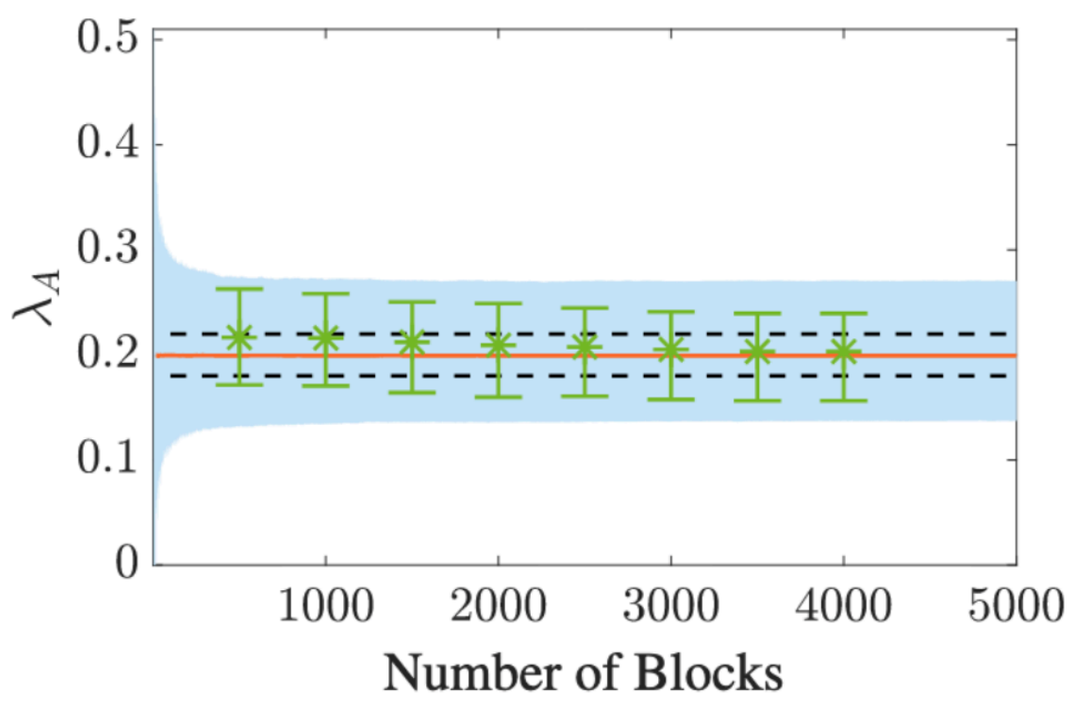
PeerCoin

$$H(\texttt{prevblockhash}||addr||\texttt{timeinsec}) < d \cdot \textbf{coin}(addr)$$

Possibly unfair



(a) PoW



(b) ML-PoS

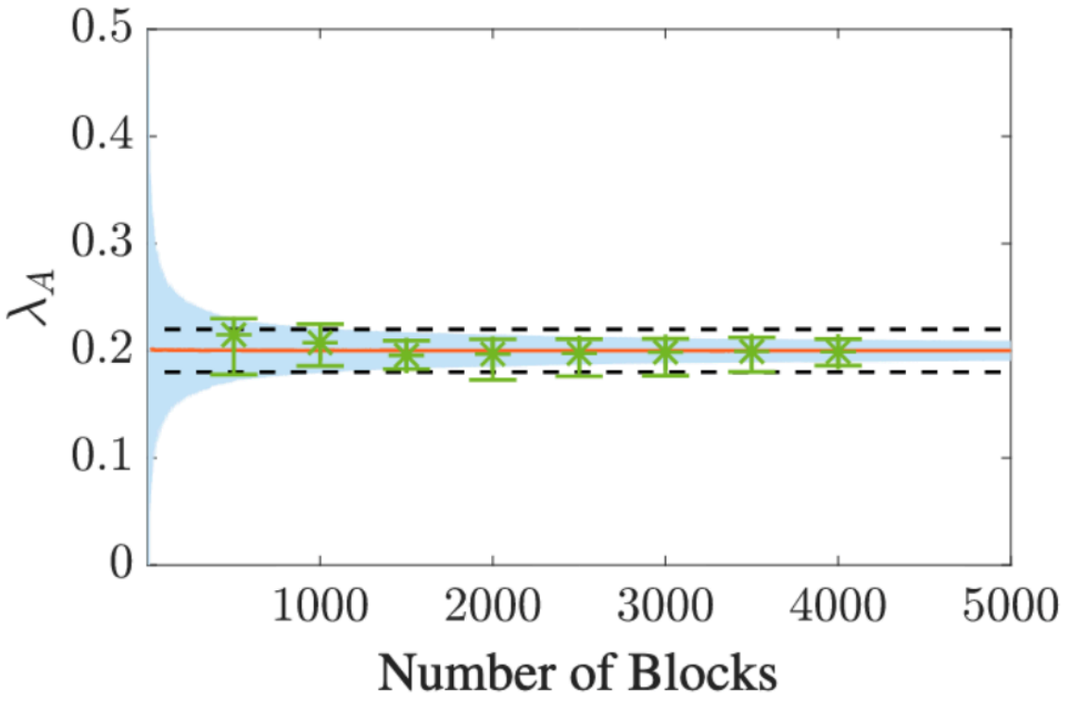
Huang et al., SIGMOD'21

Proof of Stake

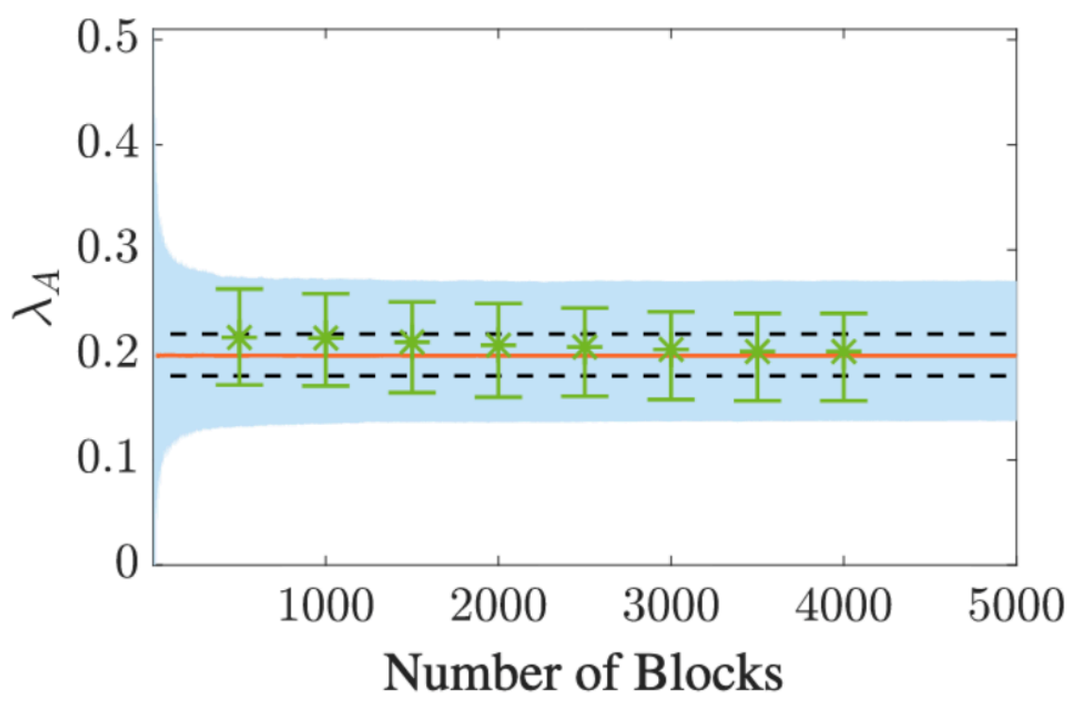
PeerCoin

$$H(\texttt{prevblockhash}||addr||\texttt{timeinsec}) < d \cdot \textbf{coin}(addr)$$

Possibly unfair



(a) PoW



(b) ML-PoS

Huang et al., SIGMOD'21

Proof of Stake

PeerCoin

$$H(\texttt{prevblockhash}||addr||\texttt{timeinsec}) < d \cdot \textbf{coin}(addr)$$

Possible to PoW (stake grinding**)**

- Try different transactions to get the next block.
- When creating a block, you can decide, which transactions to include.
- Trying different transactions you can get different hashes.
- Try to find a hash that allows you to also create the next block.

Proof of Stake

PeerCoin

$$H(\texttt{prevblockhash}||addr||\texttt{timeinsec}) < d \cdot \textbf{coin}(addr)$$

Possible to PoW (stake grinding**)**

- Try different transactions to get the next block.

Countermeasures

- Temporarily reduce stake after finding a block
- Other source then blockhash, e.g. proof
  $\pi_{i+1} = H(\pi_i||addr||\texttt{timeinsec})$

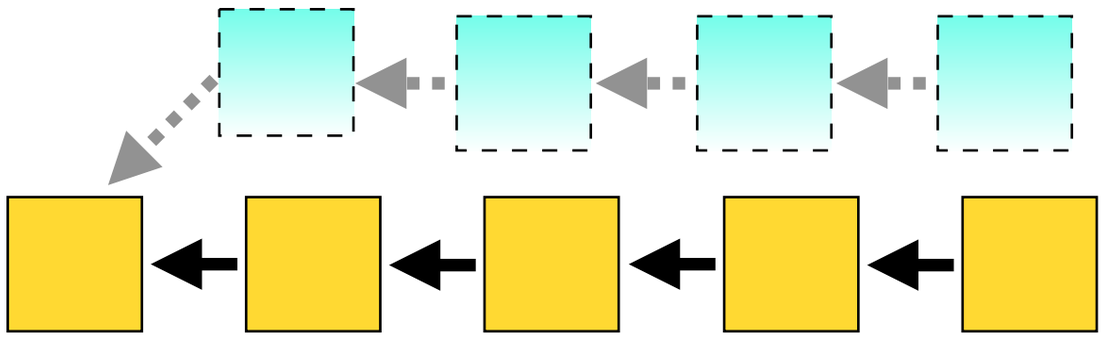
Proof of Stake

PeerCoin

$$H(\texttt{prevblockhash}||addr||\texttt{timeinsec}) < d \cdot \textbf{coin}(addr)$$

History rewrite (long range attacks**)**

- Rebuild a chain from an earlier point with



Combine with:

- Stake grinding (PoW)
- Stealing old keys

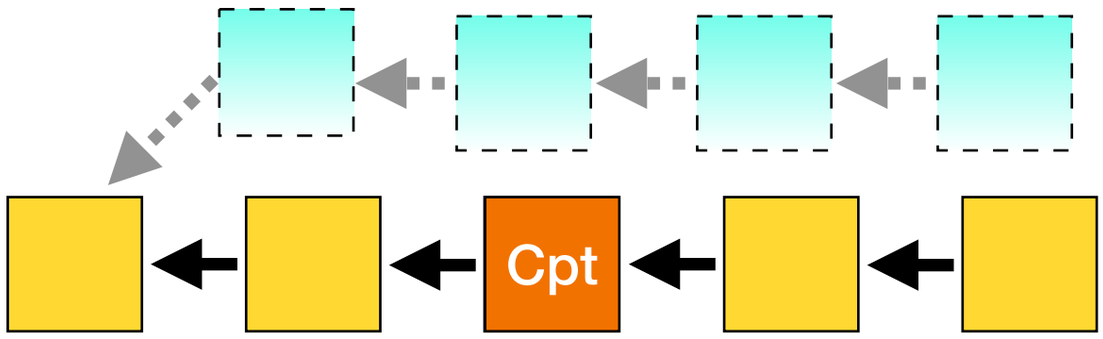
Proof of Stake

PeerCoin

$$H(\texttt{prevblockhash}||addr||\texttt{timeinsec}) < d \cdot \textbf{coin}(addr)$$

History rewrite (long range attacks**)**

- Rebuild a chain from an earlier point with



Countermeasure

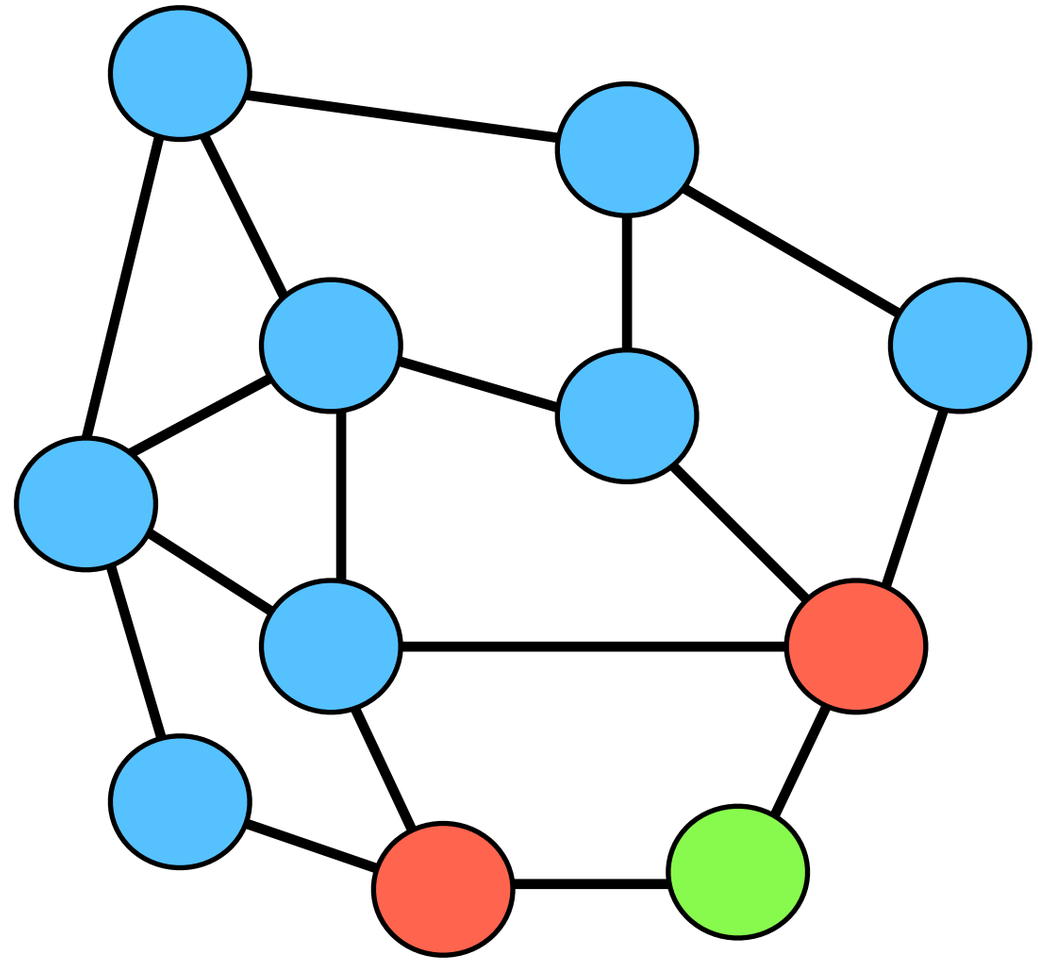
- Checkpoints

Proof of Stake

Oups: Eclipse attack dangerous in PoS

Eclipse attack:

- Single node is cut off from network by attacker
- Attacker can show him an alternative chain



Proof of Stake

Ethereum

Moved from PoW to PoS

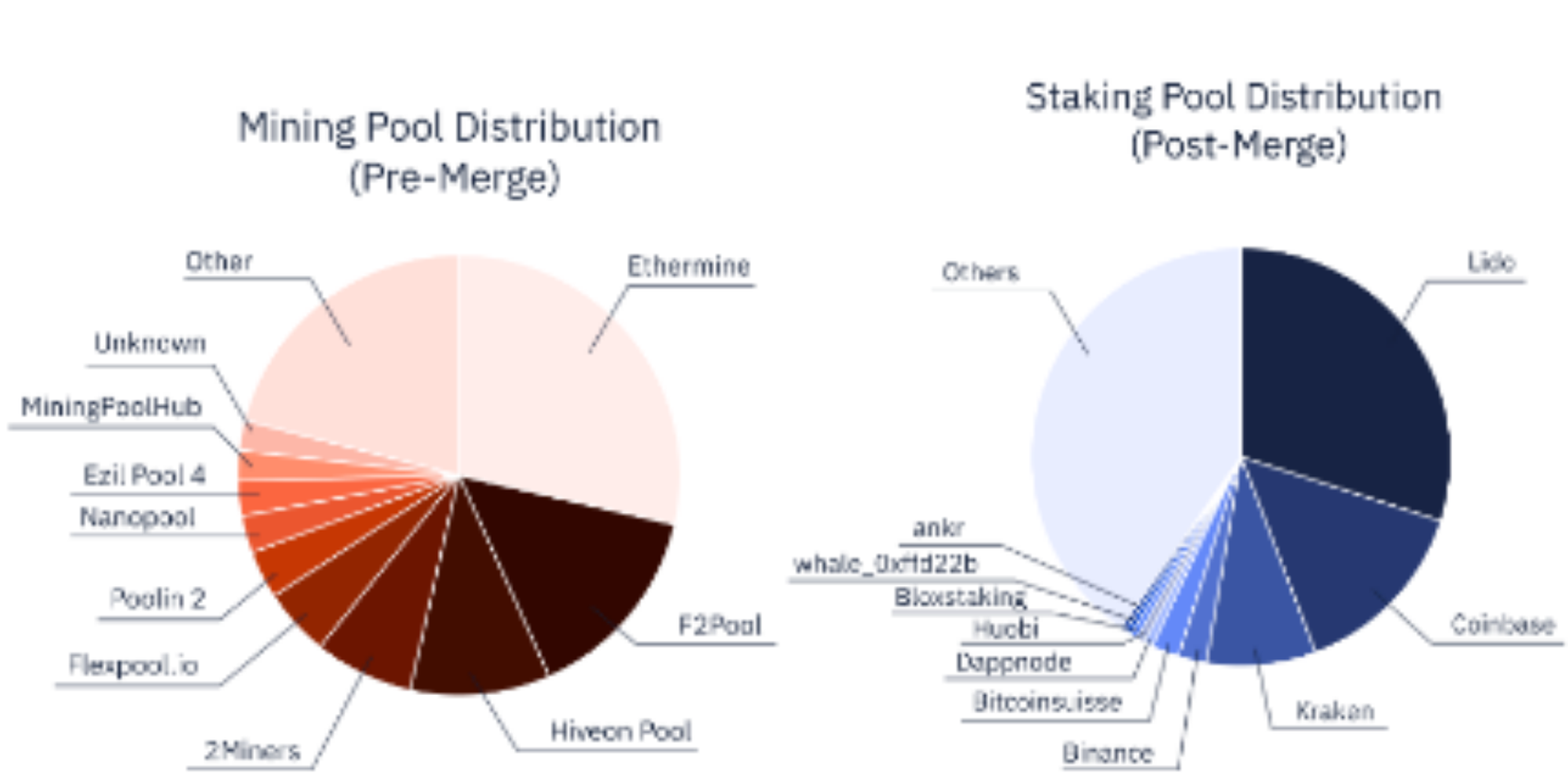


Fig. 3: Block proposal's distribution per mining and staking entities for the two months of study.

Comittee based blockchains

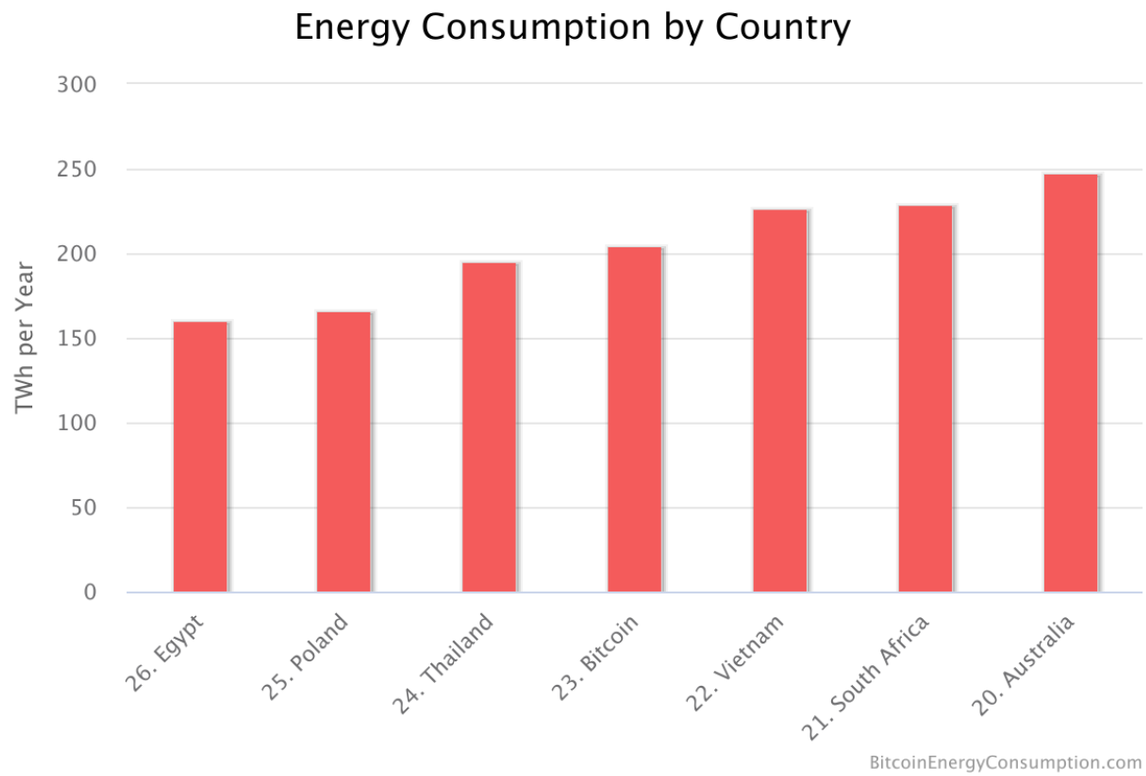
Bitcoin

Downsides

Throughput at most 7tx per second

Confirmation latency approx 1h

Enormous energy consumption



Bitcoin

Downsides

Throughput at most 7tx per second

Confirmation latency approx 1h

Enormous energy consumption → PoS

- **PoS problems**
  - Predictability (look in the future)
  - Nothing at stake (Can work on 2 forks)
  - Possibly unfair (rich get richer)
  - Possible to PoW (stake grinding)
  - History rewrite (Long range attacks)

Throughput

Decoupling performance and security

Problem in PoW and PoS:

Faster or larger blocks lead to more forks

Throughput

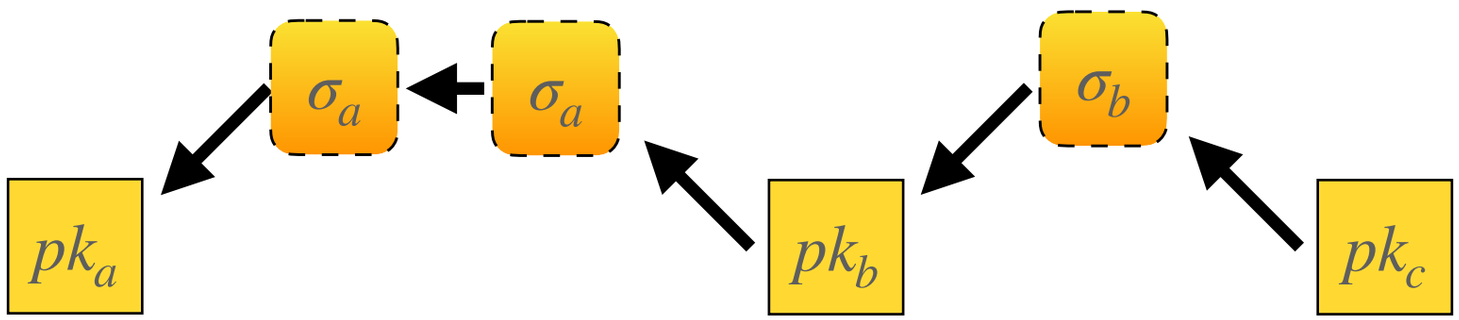
Decoupling performance and security

Problem in PoW and PoS:

Faster or larger blocks lead to more forks

Solution: Bitcoin-NG [NSDI`16]

- **Keyblocks:** Use PoW/PoS to elect leader.
- **Microblocks:** Leader publishes blocks with transactions.



Throughput

Decoupling performance and security

Problem in PoW and PoS:

Faster or larger blocks lead to more forks

Solution: Bitcoin-NG [NSDI`16]

- **Keyblocks:** Use PoW/PoS to elect leader.
- **Microblocks:** Leader publishes blocks with transactions.

Problems:

- *Leader is target for DOS.*
- Does not solve commit latency.
- No rate limit

Throughput

Decoupling performance and security

Solution: Bitcoin-NG [NSDI`16]

- **Keyblocks:** Use PoW/PoS to elect leader.
- **Microblocks:** Leader publishes blocks with transactions.

Attacks:

- Steal microblocks
- Selfish microblock creation

Need to devide block reward (fees) for microblocks

Throughput

Decoupling performance and security

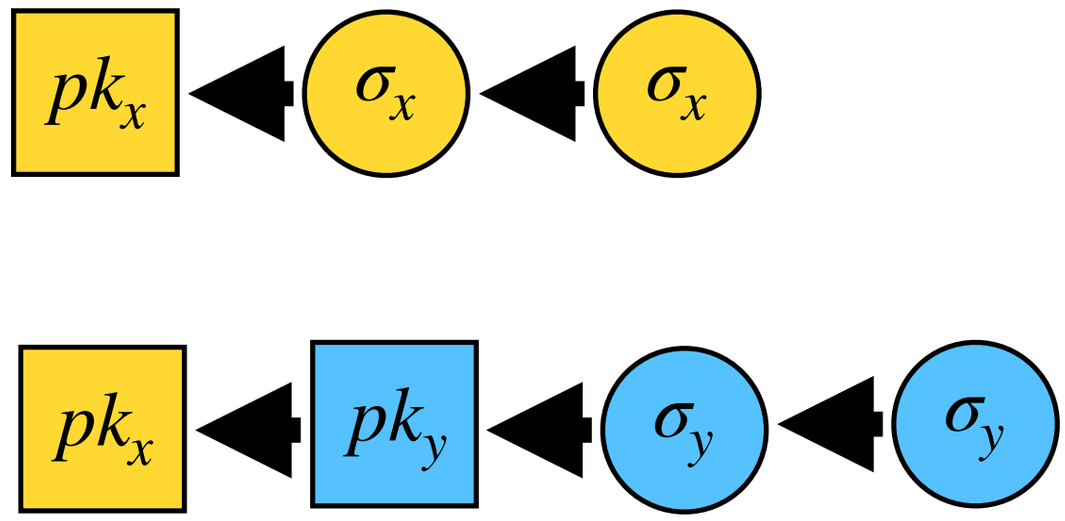
Solution: Bitcoin-NG [NSDI`16]

- **Keyblocks:** Use PoW/PoS to elect leader.
- **Microblocks:** Leader publishes blocks with transactions.

Attacks:

- Steal microblocks

Big enough reward for next leader



Throughput

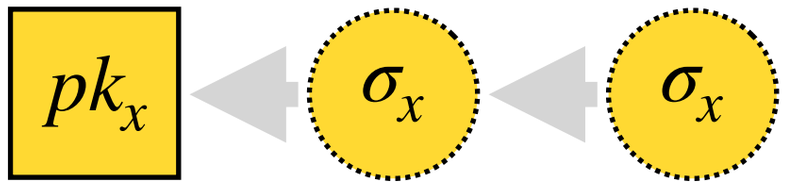
Decoupling performance and security

Solution: Bitcoin-NG [NSDI`16]

- **Keyblocks:** Use PoW/PoS to elect leader.
- **Microblocks:** Leader publishes blocks with transactions.

Attacks:

- Selfish microblock creation



Big enough reward for current leader

Throughput

Decoupling performance and security

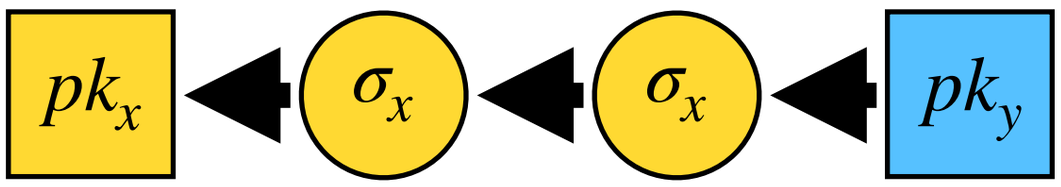
Solution: Bitcoin-NG [NSDI`16]

- **Keyblocks:** Use PoW/PoS to elect leader.
- **Microblocks:** Leader publishes blocks with transactions.

Attacks:

- Steal microblocks
- Selfish microblock creation

Solution: 40% to pk_x and 60% to pk_y



Confirmation latency

Committees confirm the block

Problem in PoW and PoS:

Need to wait for multiple blocks for confirmation.

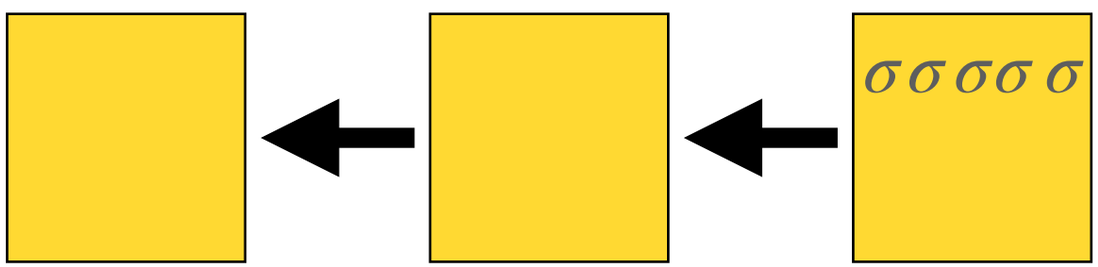
Confirmation latency

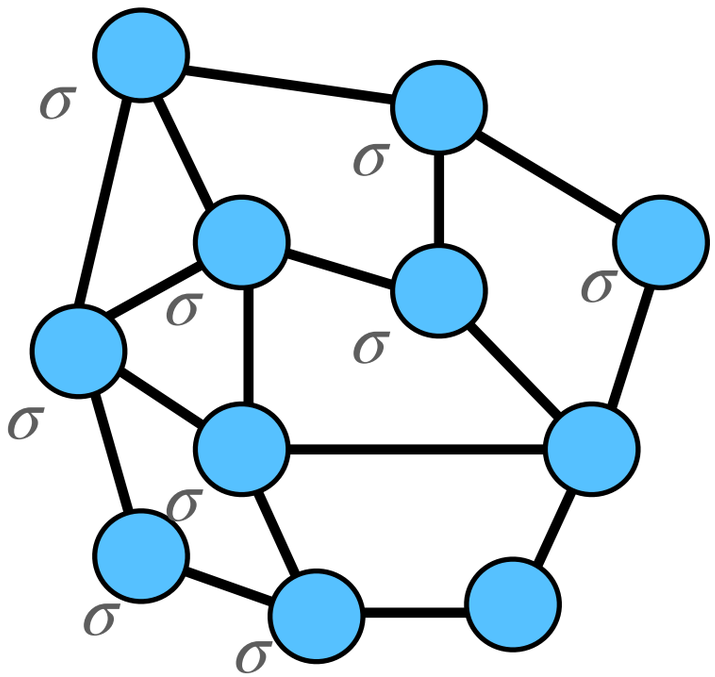
Committees confirm the block

Problem in PoW and PoS:

Need to wait for multiple blocks for confirmation.

Idea: Require large fraction (e.g. 2/3) of the nodes to confirm they are extending a certain block.





Confirmation latency

Committees confirm the block

Problem in PoW and PoS:

Need to wait for multiple blocks for confirmation.

Idea: Require large fraction (e.g. 2/3) of the nodes to confirm they are extending a certain block.

Problems:

- **Sybills (**what is 2/3?**)**
- **Enforcing promise**
- **Conflicting promises (**forks**)**

Confirmation latency

Committees confirm the block

Idea: Require large fraction (e.g. 2/3) of the nodes to confirm they are extending a certain block.

Sybills problems (what is 2/3?**)**

- 2/3 requires to know who is all nodes

Confirmation latency

Committees confirm the block

Idea: Require large fraction (e.g. 2/3) of the nodes to confirm they are extending a certain block.

Sybills problems (what is 2/3?**)**

- 2/3 requires to know who is all nodes

Solution (PoW&PoS)

- 2/3 of the nodes that found the last 100 blocks
- or in PoS: nodes that own 2/3 of the stake.

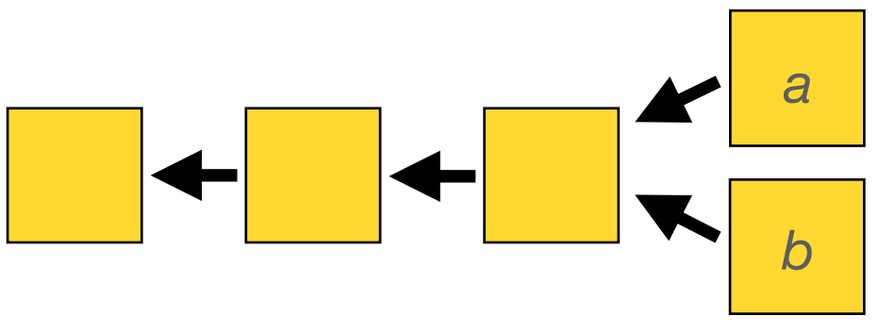
Confirmation latency

Committees confirm the block

Idea: Require large fraction (e.g. 2/3) of the nodes to confirm they are extending a certain block.

Enforce promise

- After signing *a* a node should not create a block extending *b.*



Slashing:
Punish nodes for misbehaviour
e.g. by taking their stake

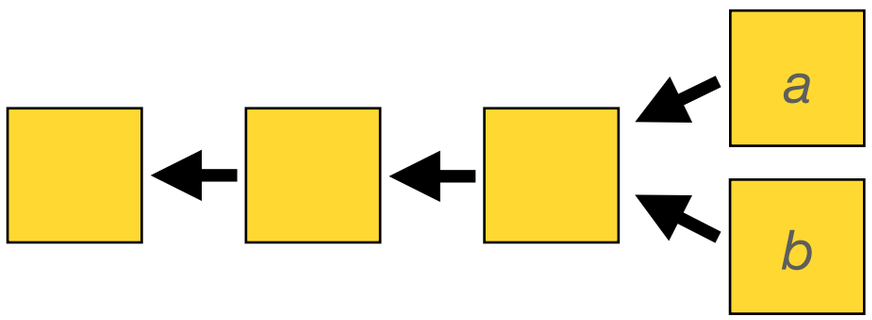
Confirmation latency

Committees confirm the block

Idea: Require large fraction (e.g. 2/3) of the nodes to confirm they are extending a certain block.

Conflicting promises (forks**)**

- What if 1/2 promises *a* and 1/2 promises *b*?



Consensus:
Employ a consensus algorithm

Committee based blockchain

- **Multiple nodes agree on the next block**
- **Uses Consensus algorithm**
- **Byzantine failure model**

Examples:

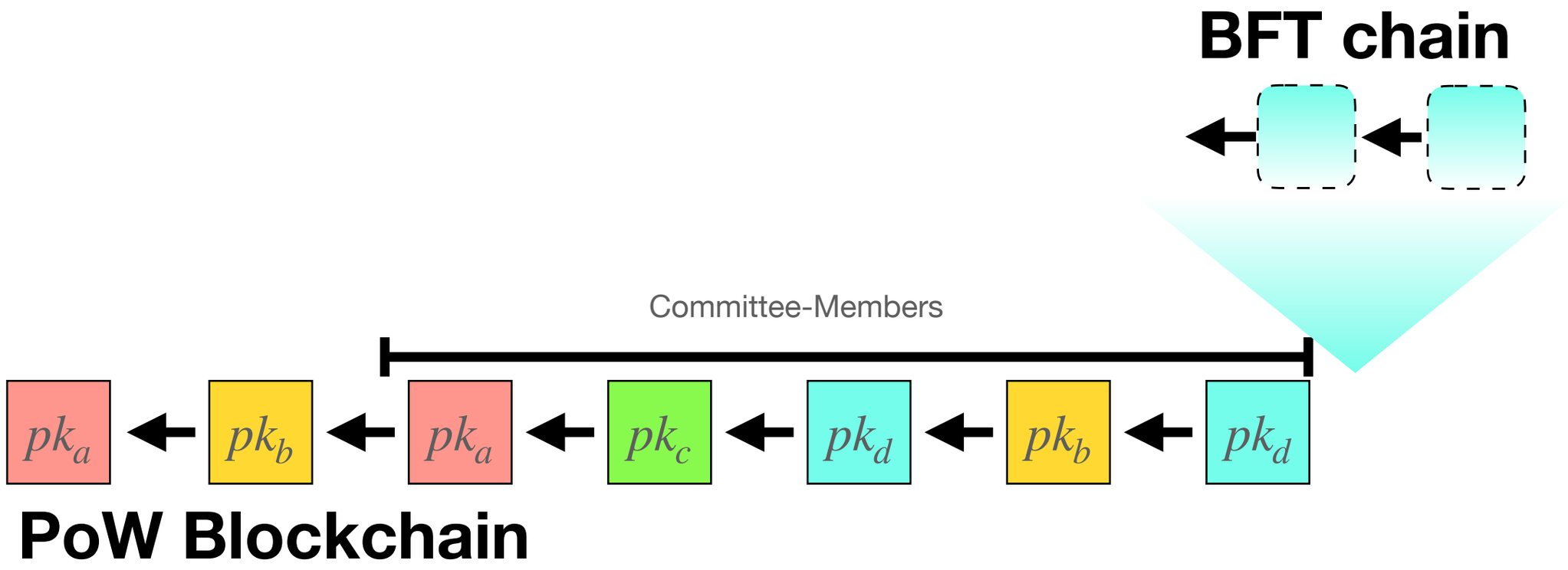
- Byzcoin (PoW)
- Cosmos (PoS static)
- Algorand (PoS with randomization)

Committee based blockchain

PoW Committee

ByzCoin [USENIX Sec'16]

- A PoW blockchain determines committee members



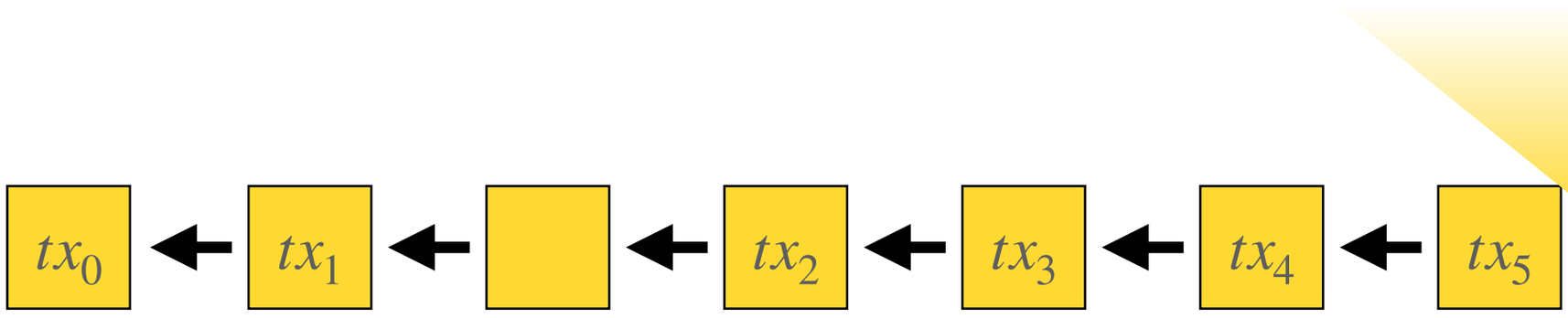
Committee based blockchain

PoS static committee

Cosmos (Tendermint)

- 100 nodes with the biggest stake are the committee

Node	Stake
A	1010
B	990
C	981



Committee based blockchain

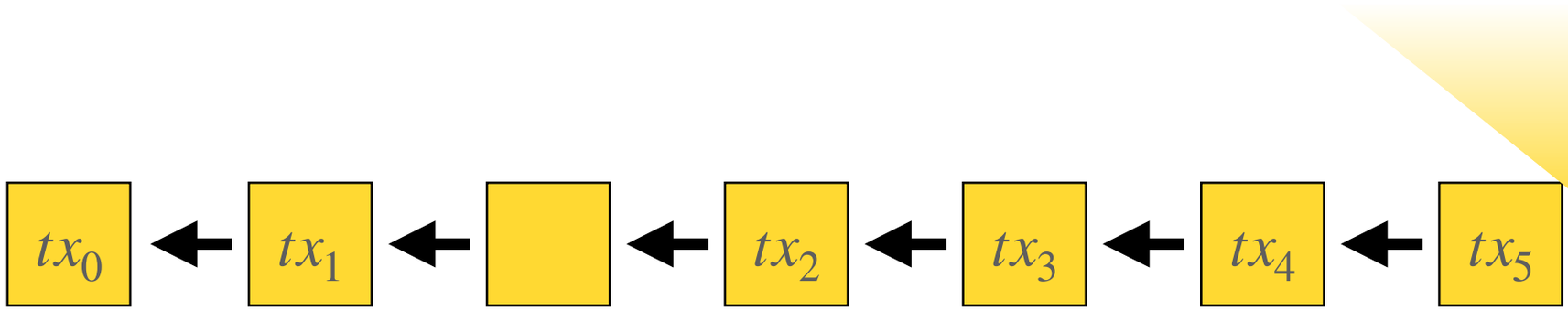
PoS random committee

Algorand

- Randomly select committee and leader for next block

$$\pi_{i+1} = H(\pi_i || addr) < d \cdot \textbf{coin}(addr)$$

Node	Stake
A	1010
B	990
C	981



Committee based blockchain

- **Multiple nodes agree on the next block**
- **Uses Consensus algorithm**
- **Byzantine failure model**

fast confirmation time

can decouple from PoW/PoS for high throughput

have rate limit since many nodes need to vote

Committee based blockchain

PoS challenges in Committee based blockchains

- **Multiple nodes agree on the next block**
- **Uses Consensus algorithm**
- **Byzantine failure model**

PoS challenges:

- Predictability (look in the future)
- Nothing at stake (Can work on 2 forks)
- Possibly unfair (rich get richer)
- Possible to PoW (stake grinding)
- History rewrite (Long range attacks)

Committee based blockchain

PoS challenges in Committee based blockchains

Predictability (look in the future**)**

- Solved if `timeinseconds` is not used.
- Can be improved by including signatures from committee

$$\pi_{i+1} = H(\pi_i || addr || [\sigma_1, \sigma_2, \ldots]) < d \cdot \text{coin}(addr)$$

- Verifiable random function: VRF

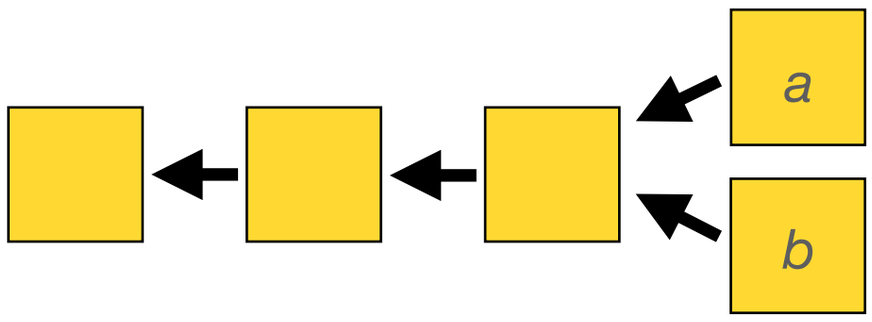
Also solves Possibility to PoW.

Committee based blockchain

PoS challenges in Committee based blockchains

Nothing at stake (can work on 2 forks**)**

- Slashing employed
- Multiple committee members participate in confirming a block



Tendermint proved that:

- **On fork, someone can be slashed**

Committee based blockchain

PoS challenges in Committee based blockchains

Possibly unfair (rich get richer**)**

- Less problematic, since all committee members get a reward

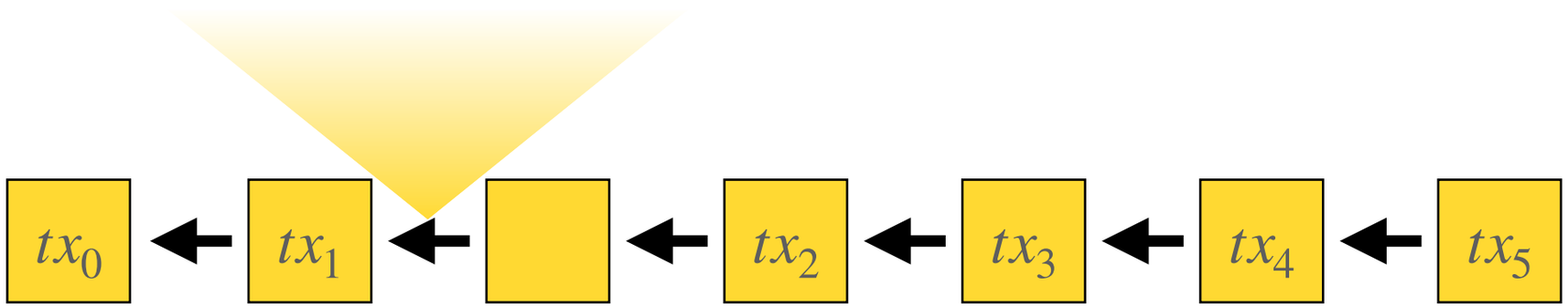
Committee based blockchain

PoS challenges in Committee based blockchains

History rewrite (Long range attacks**)**

- Cannot rewrite history, unless I control 2/3 of the share.

Node	Stake
A	879
B	870
C	830



Committee based blockchain

PoS challenges in Committee based blockchains

- **Multiple nodes agree on the next block**
- **Uses Consensus algorithm**
- **Byzantine failure model**

PoS challenges:

- Predictability (look in the future)

Consensus:
Employ a consensus algorithm

Committee based blockchain

Challenges:

- Scale to many nodes
- Fair/frequent leader election
- Reward distribution
- Create proofs for slashing

Sharding

Sharding
Ideas and potential

Shard:

Potential:

Sharding

Ideas and potential

Shard: Subsystem with a fraction of the state, processing transactions on this part of the state.

Potential: Scale throughput linearly with the number of shards.

Sharding

Problems

Sharding

Problems

A. **How to distribute state?**

B. **How to process transactions across shards?**

C. **How to avoid mining power dillusion?**
   Easier to attack a single shard than the complete system.

Sharding
Solutions

A. **How to distribute state?**

- Consistent hashing.

B. **How to process transactions across shards?**

- Atomic commit?

C. **How to avoid mining power dillusion?**

- Disallow choosing, e.g. consistent hashing (difficult).
- Allow multiple shards as in Monoxide (will there be sharding?)

Sharding example

Omniledger:
https://youtu.be/f1pyaAZ7bj0?si=g1RNtP__1ooYqCie