ETH zürich

OmniLedger

A Scale-Out, Decentralized, Secure Ledger via Sharding

Presented by Felix Laufenberg

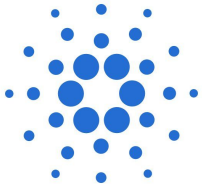
A **Scale-Out**, Decentralized, Secure Ledger via Sharding

But do we need scalability?

7 tx/s
50$ tx fee
Energy of Czech Rep.

1.7 k - 56k tx/s
(Centralized)

56 tx/s



276 tx/s
(Permissioned)

20 tx/s
(with PoW)

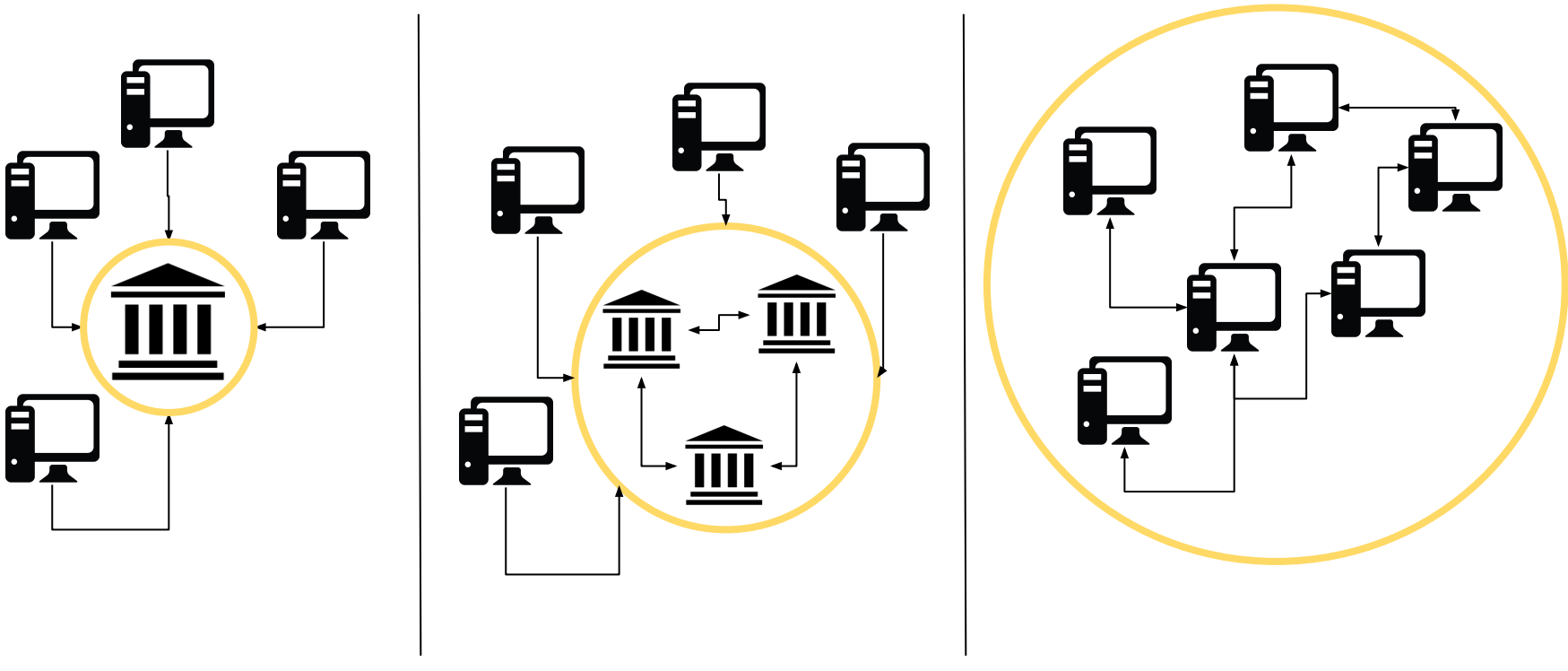
OmniLedger

10k tx/s*

*(12.5% adversary, 72 nodes/committee, 25 committees)

A Scale-Out, Decentralized Ledger, Secure via Sharding

Permissionless setting



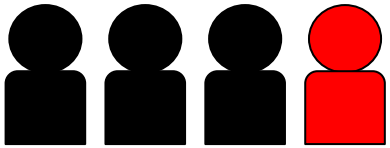
Centralized

Permissioned

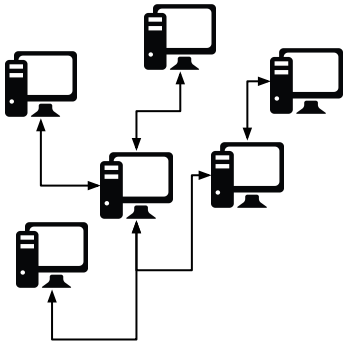
Permissionless/
Decentralized

A Scale-Out, Decentralized, Secure Ledger via Sharding

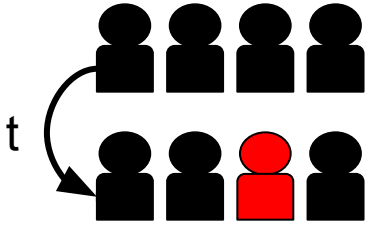
What is the adversary/network model for OmniLedger?



< 25% Malicious Validators



Synchronous Network



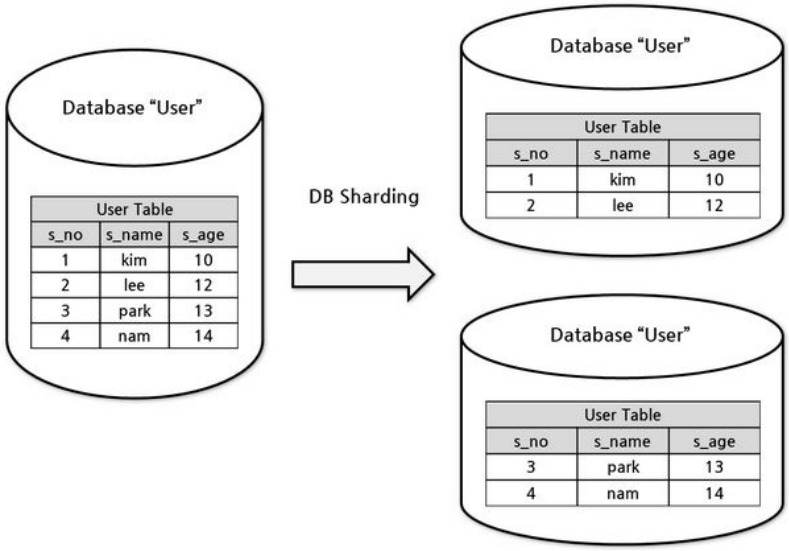
Mildly Adaptive

How realistic are these assumptions?
A:...

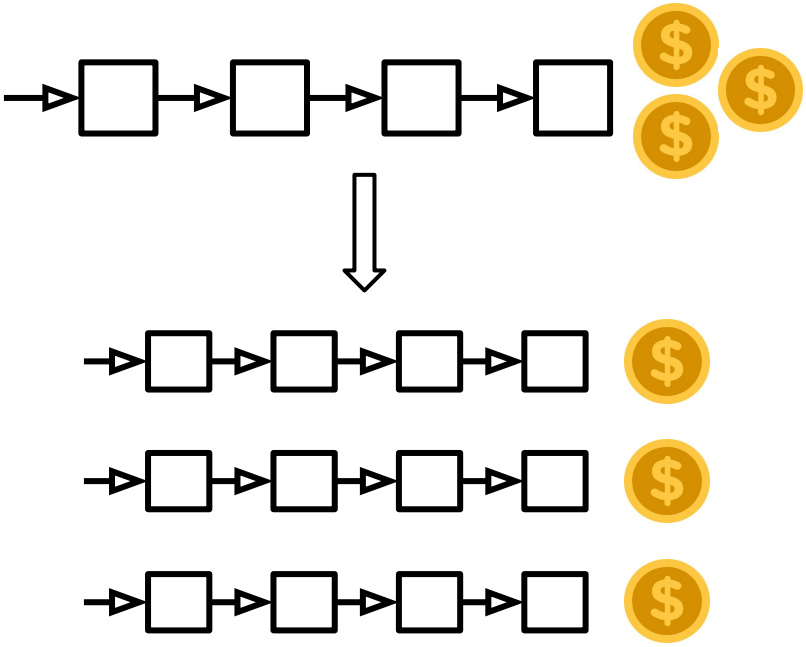
A Secure, Scale-Out, Decentralized Ledger via Sharding

From DB Architecture to Blockchain

In Database Systems:

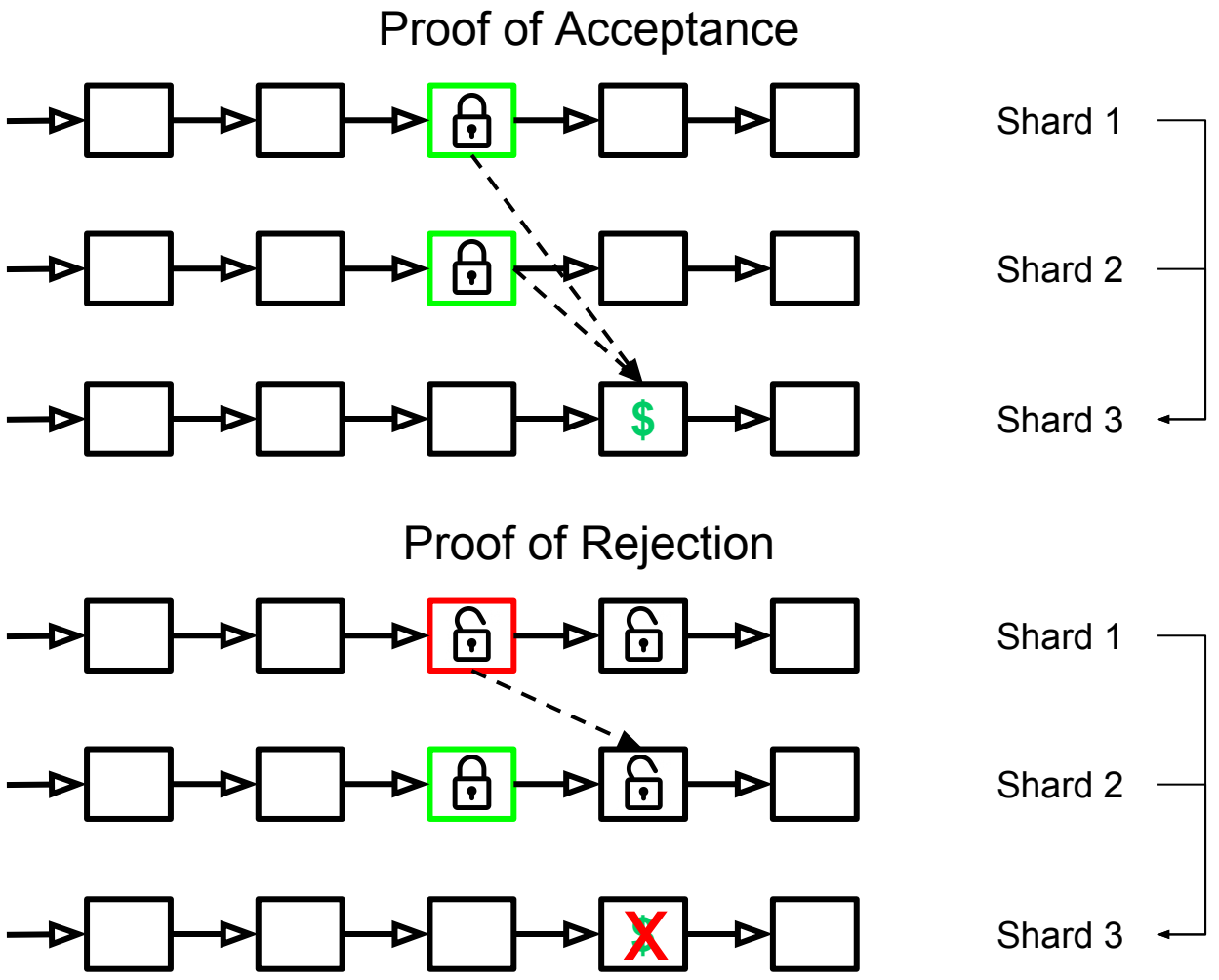


In Blockchain:



A Secure, Scale-Out, Decentralized Ledger via Sharding

Handling Cross-Shard Transactions



A Secure, Scale-Out, Decentralized Ledger via Sharding

Sharding vs Channels vs Sidechains

Proof-of-Work

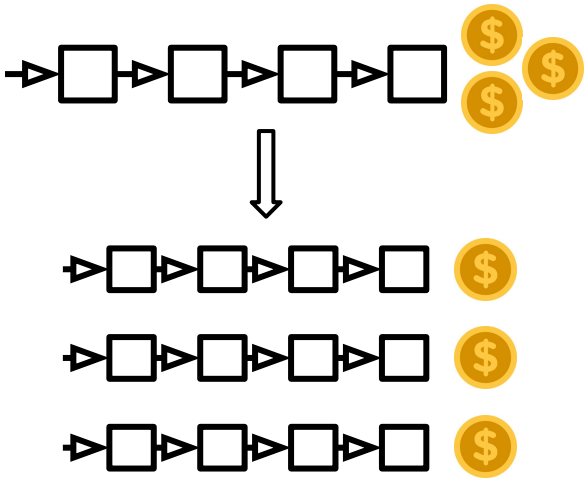
Proof-of-Stake

Proof-of-Space

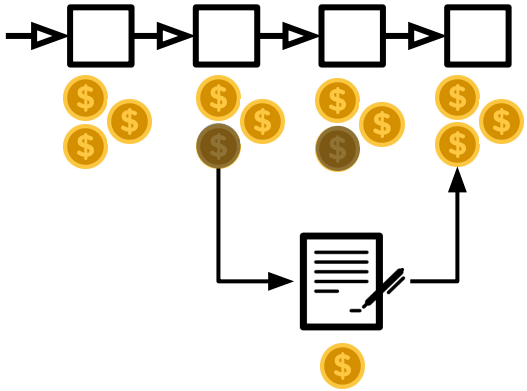
Proof-of-X

\+

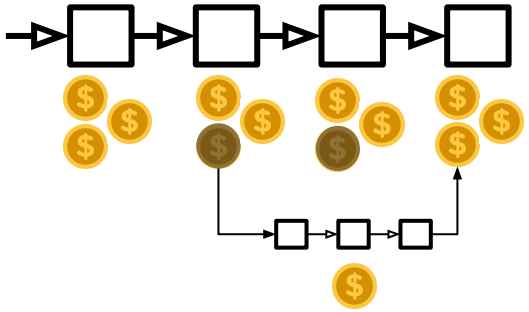
Sharding



Channels

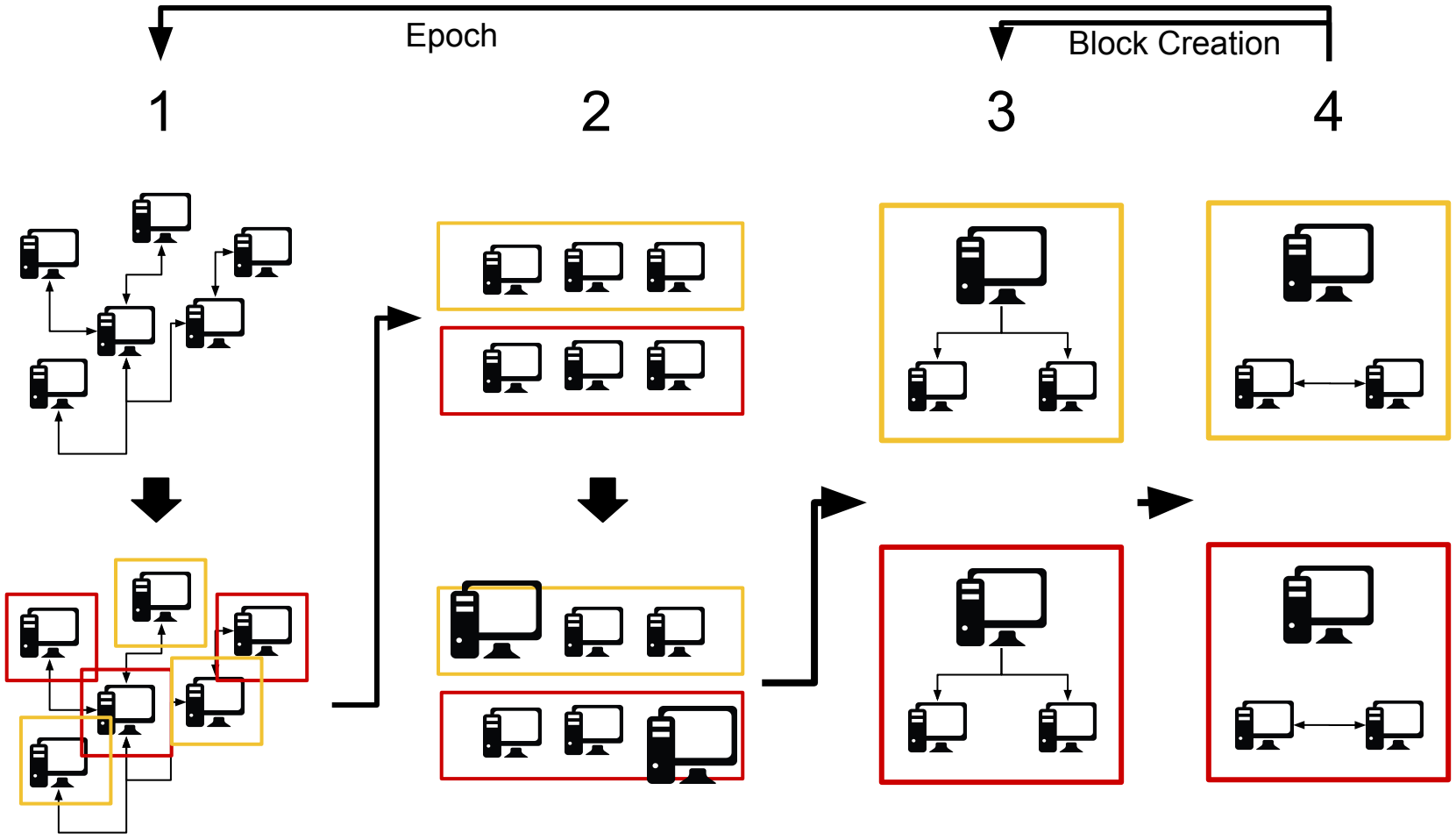


Sidechains



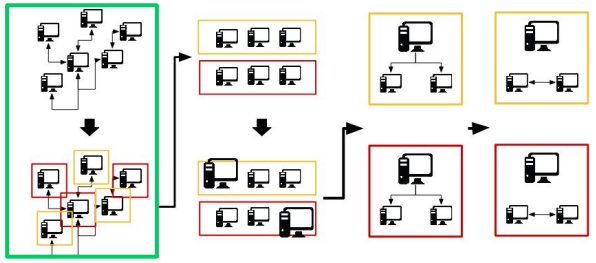
The OmniLedger Protocol

High Level Overview



Step 1: Assigning Nodes to Shards

RandHound



Problem 1: Randomness Needed!

- Grinding attacks
- DOS attacks
- Corruption of Nodes

Solution: RandHound

- Bias resistant public randomness
- Interpreted as permutation of nodes

Problem 2: Bootstrapping

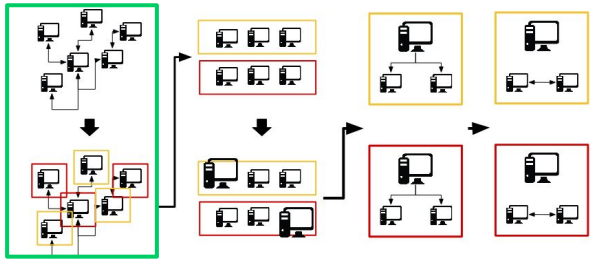
- Validators need to up to date on assigned shard (expensive/ congestion)
- Protocol should run during reconfiguration of validator - shard assignment

Solution: Gradual changing shard assignments

- Too many: need ⅔ Honest Majority participating
- Too little: -> Problem 1
- log(n) nodes per epoch

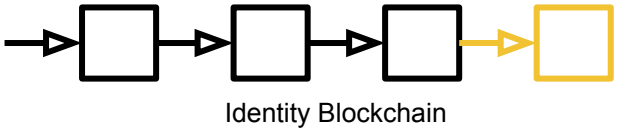
Step 1: Assigning Nodes to Shards

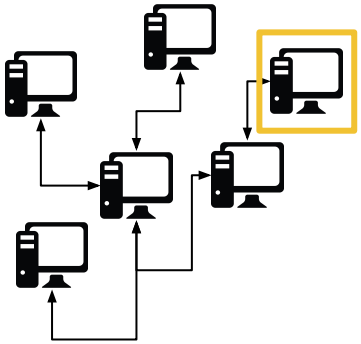
Sybil Attacks & Identity Blockchain



Problem : Sybil Attacks

- Adversary creating arbitrary many nodes



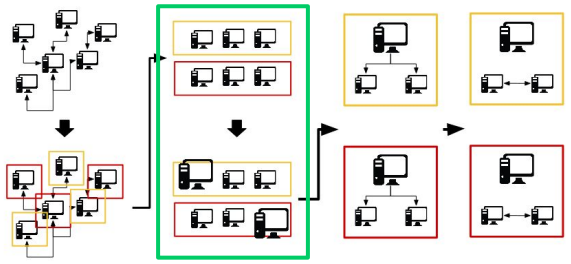


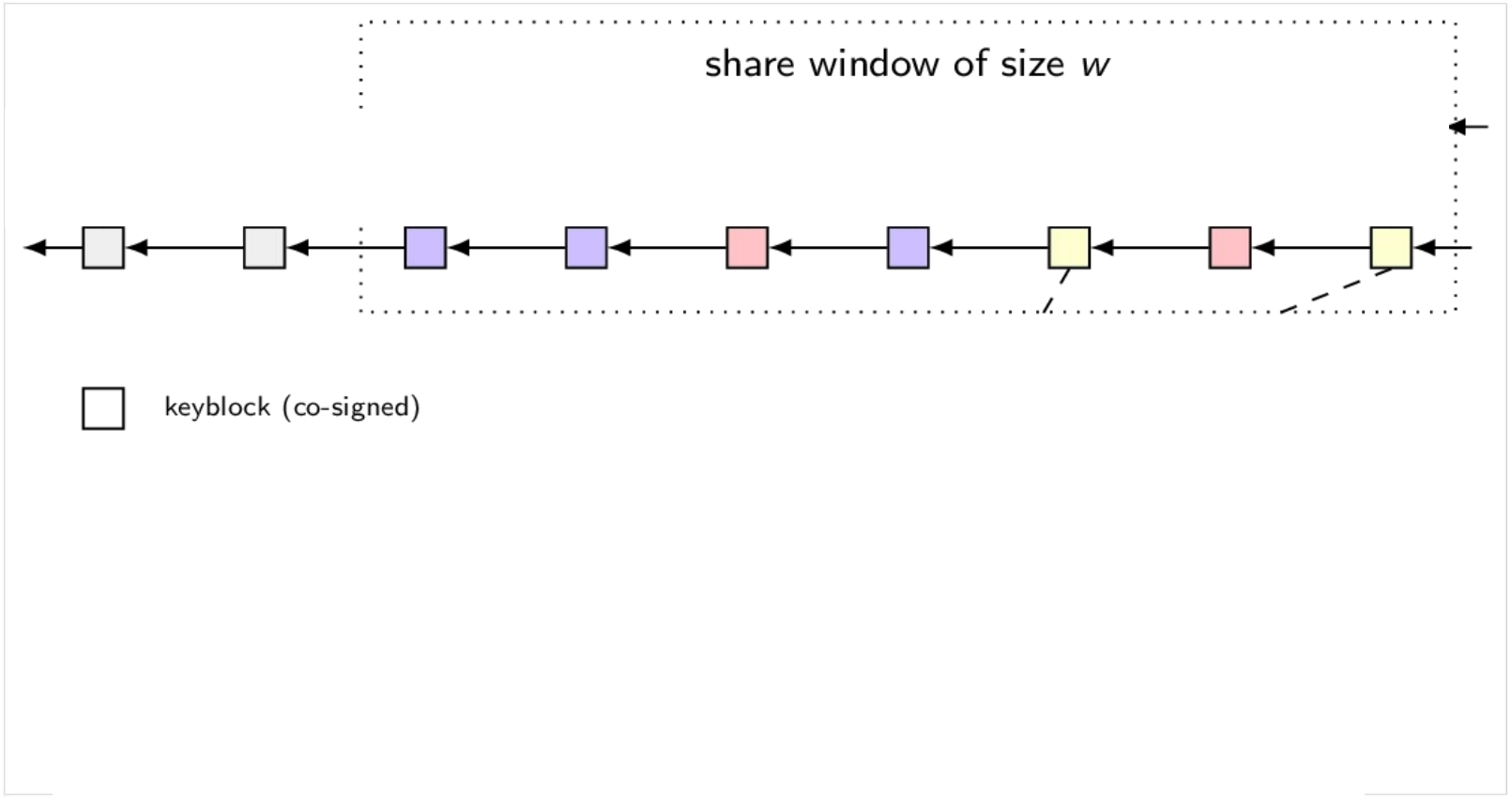
Solution : Sybil Resistant Identities

- PoW/PoP/PoB + Consensus to join

Step 2: Leader + Validator Election

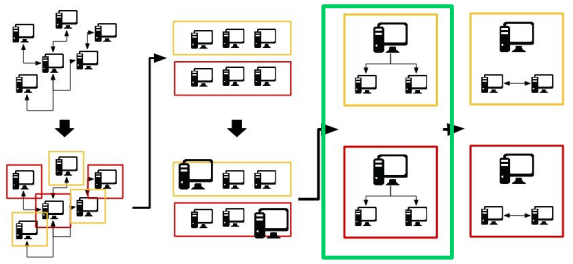
ByzCoin

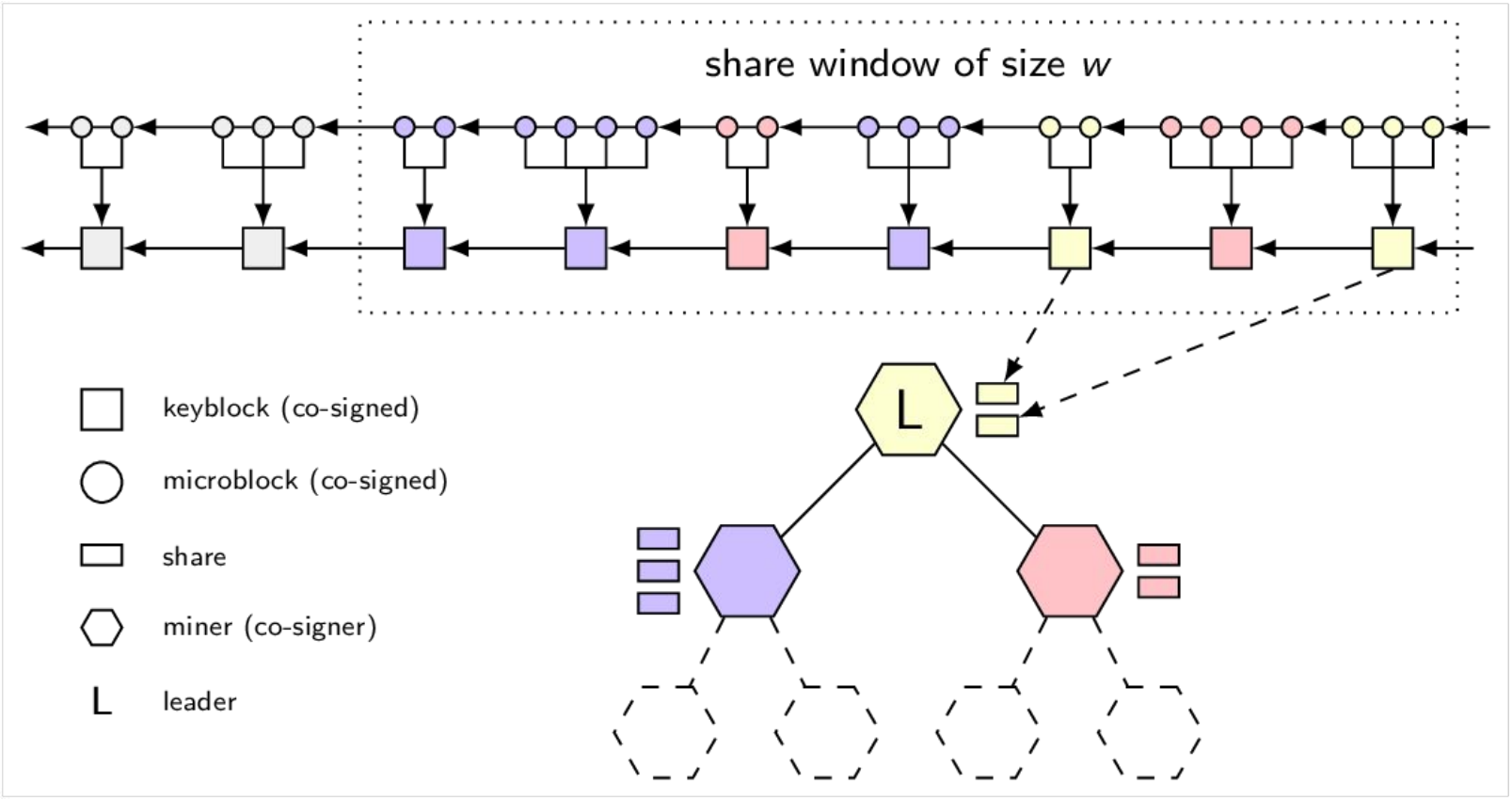




Step 3: Block Proposal

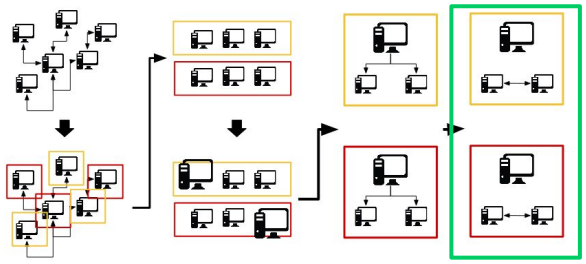
ByzCoin





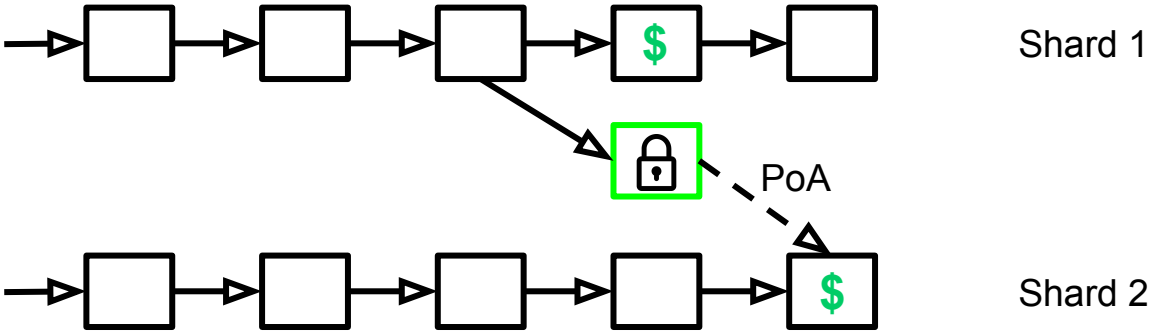
Step 4: PBFT on proposed Block

ByzCoin



Problem : Forking

- Double spending
- Waiting to be able to confirm cross-shard transactions



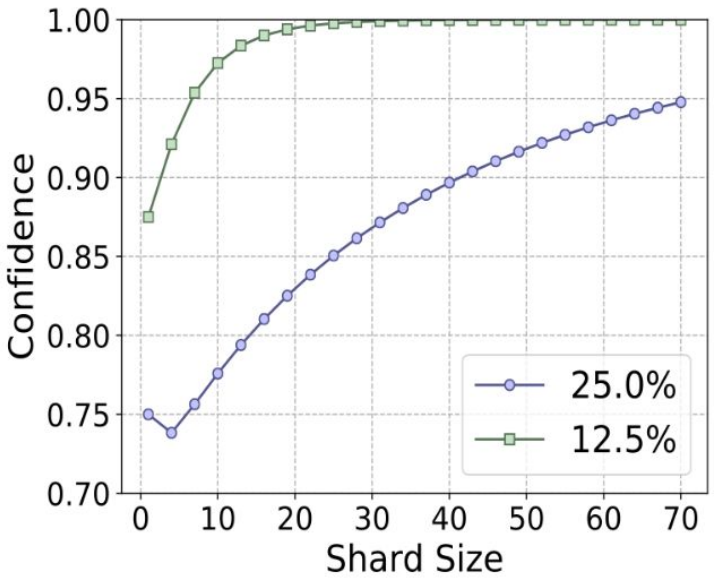
Solution : Byzantine Agreement

- Established PBFT (+Optimizations)

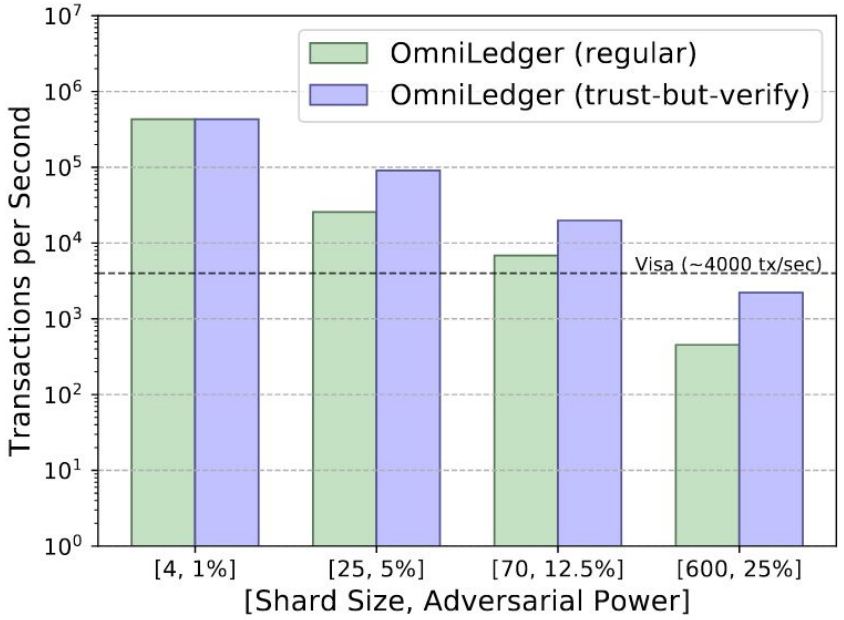
Performance vs Security

Shard Security

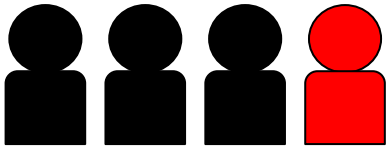
Probability that ⅔ honest majority holds



Measured Performance (Simulation)

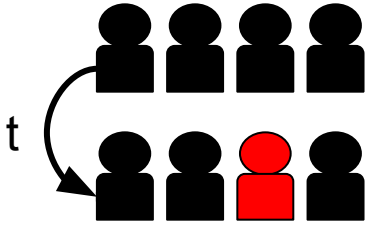


Revisiting Security



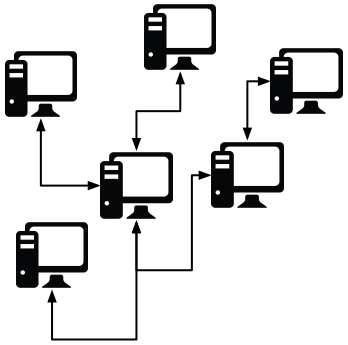
< 25% Malicious Validators

Weak assumption + needs to be adapted for performance



Mildly Adaptive

t in the order of O(n/log(n) * e)
Prune to DOS attacks



Synchronous Network

Unrealistic assumption, prune to network layer attacks

+ Almost no resistance to Sibyl attacks

How realistic are these assumptions?
A: Not very realistic

Conclusion

- Constant probability of failure -> expected to fail
- Fails badly -> double spending
- PoW seems like wrong approach

Good foundation for future work, but not practical yet.

Thank You

To Add:

Comparison to channels

RandHound

Simulation

Security arguments

Trust-but-verify

PBFT + CoSi

Shard size - performance - security Tradeoff