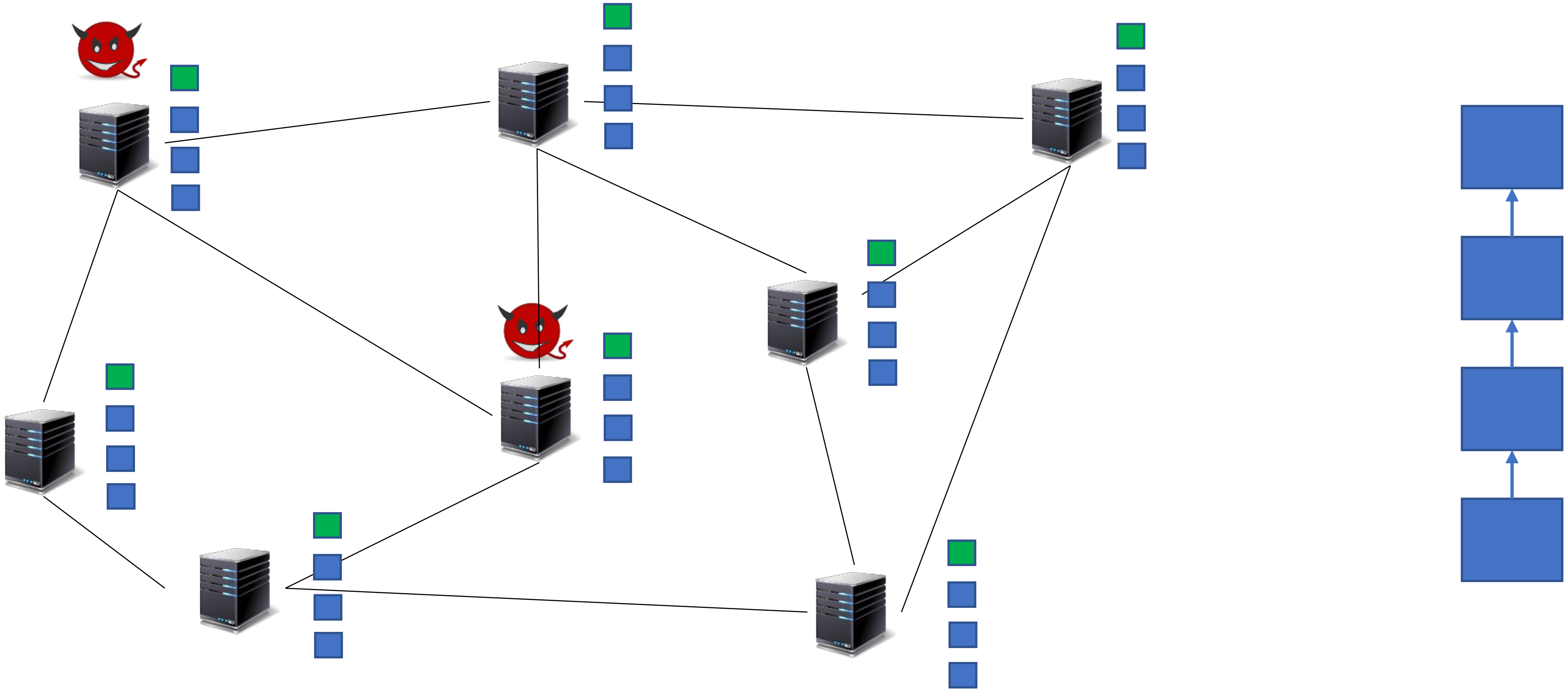
Lecture 10: Sharding

https://web3.princeton.edu/principles-of-blockchains/

Professor Pramod Viswanath
Princeton University

This lecture:
Horizontal scaling in Blockchains
Scaling Storage, Compute and Communication requirements of Bitcoin

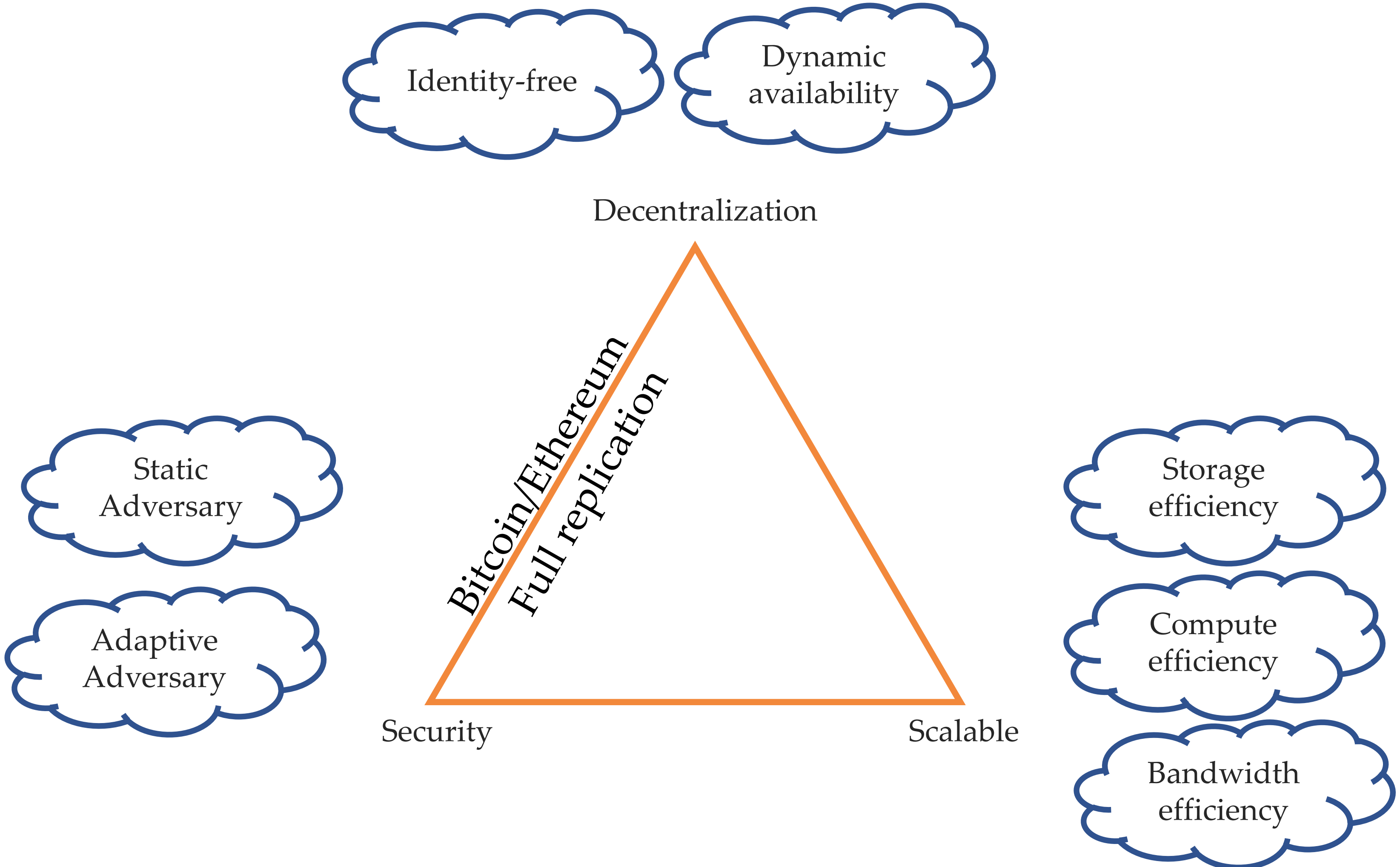
Blockchains & Full replication



Full replication – resource usage

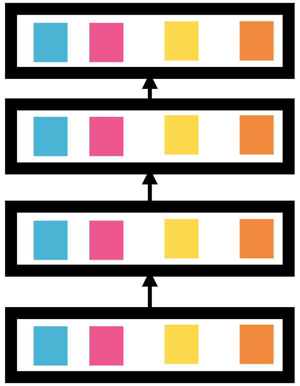
- All nodes process the same transactions
- **Communication**: The transaction has to traverse the complete network at least once
- **Storage**: All nodes have to store the complete state, the account details of everyone!
- **Compute**: All nodes have to validate all transactions and update the ledger every block

Trilemma

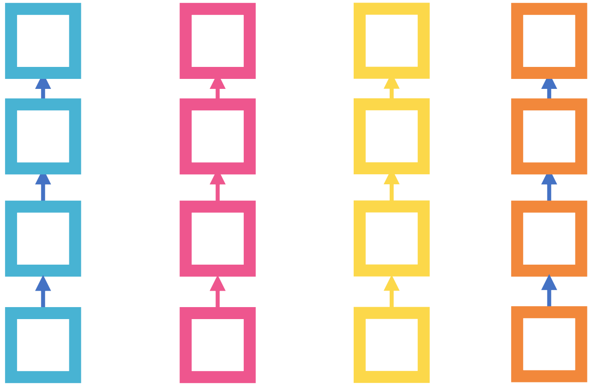

Identity-free
Dynamic availability
Decentralization
Bitcoin/Ethereum
Full replication
Static Adversary
Adaptive Adversary
Security
Scalable
Storage efficiency
Compute efficiency
Bandwidth efficiency

First Approach– Maintain multiple blockchains

- Divide ledger into K shards
- Each shard is a separate blockchain



Complete ledger



K Shards

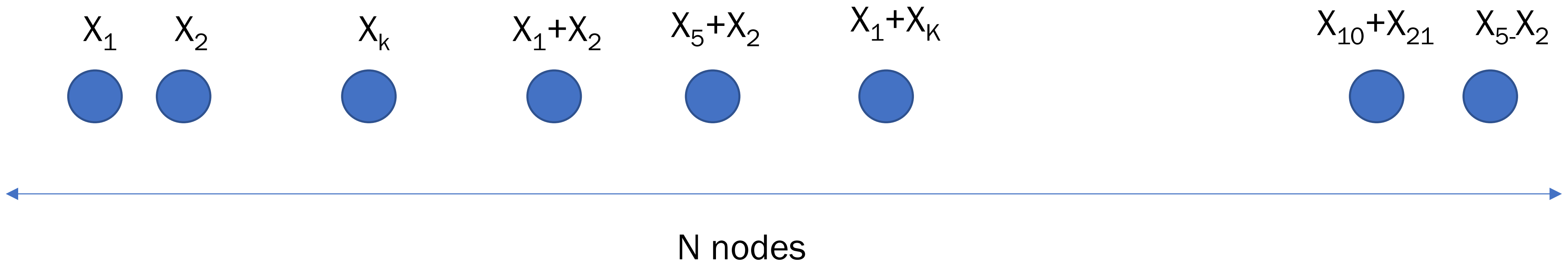
Problems

1. Reduced security – An adversary can concentrate on one shard

2. How to transfer money from one shard to another?

3. If such transfer is possible, adversary can transfer non-existent funds from infected shard to non-infected shard.

Coded Sharding: Polyshard

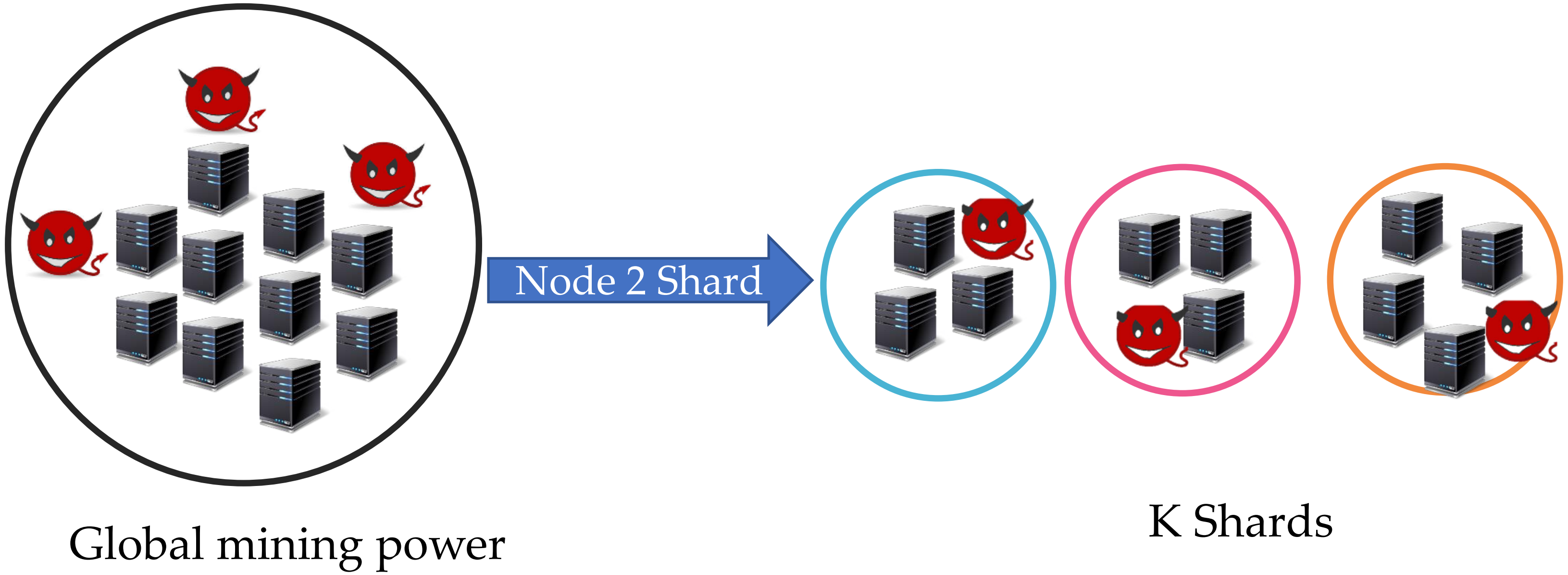


- Each node maintains a coded shard
- The coded shard is a combination of various uncoded shards

- Theoretical solution
  - Linear updates of ledger possible (for UTXO state management)
  - Nonlinear state management impractical

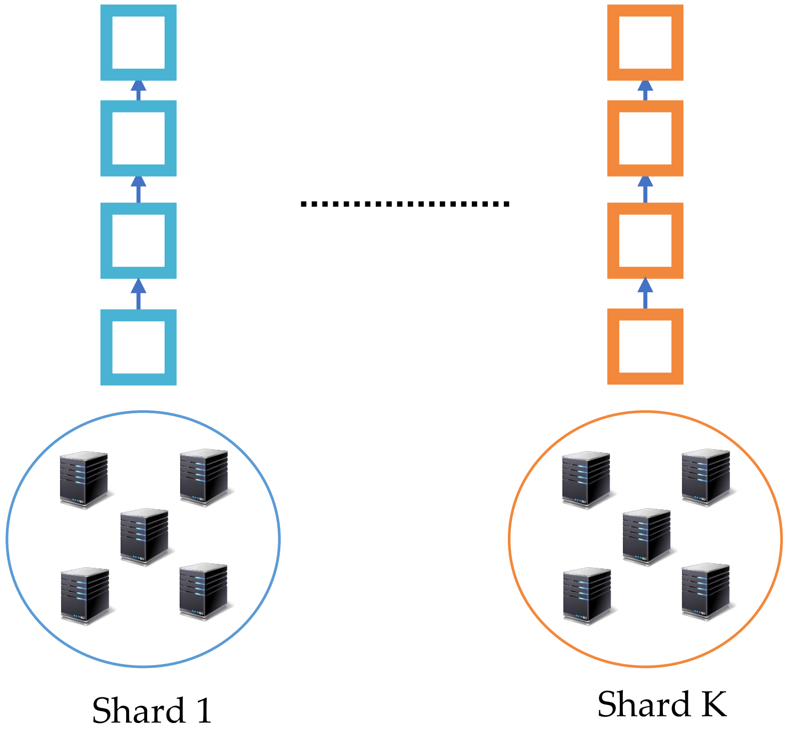
Node to shard allocation (N2S)

- Extension of the first order approach
- Allocates each consensus nodes to one shard randomly



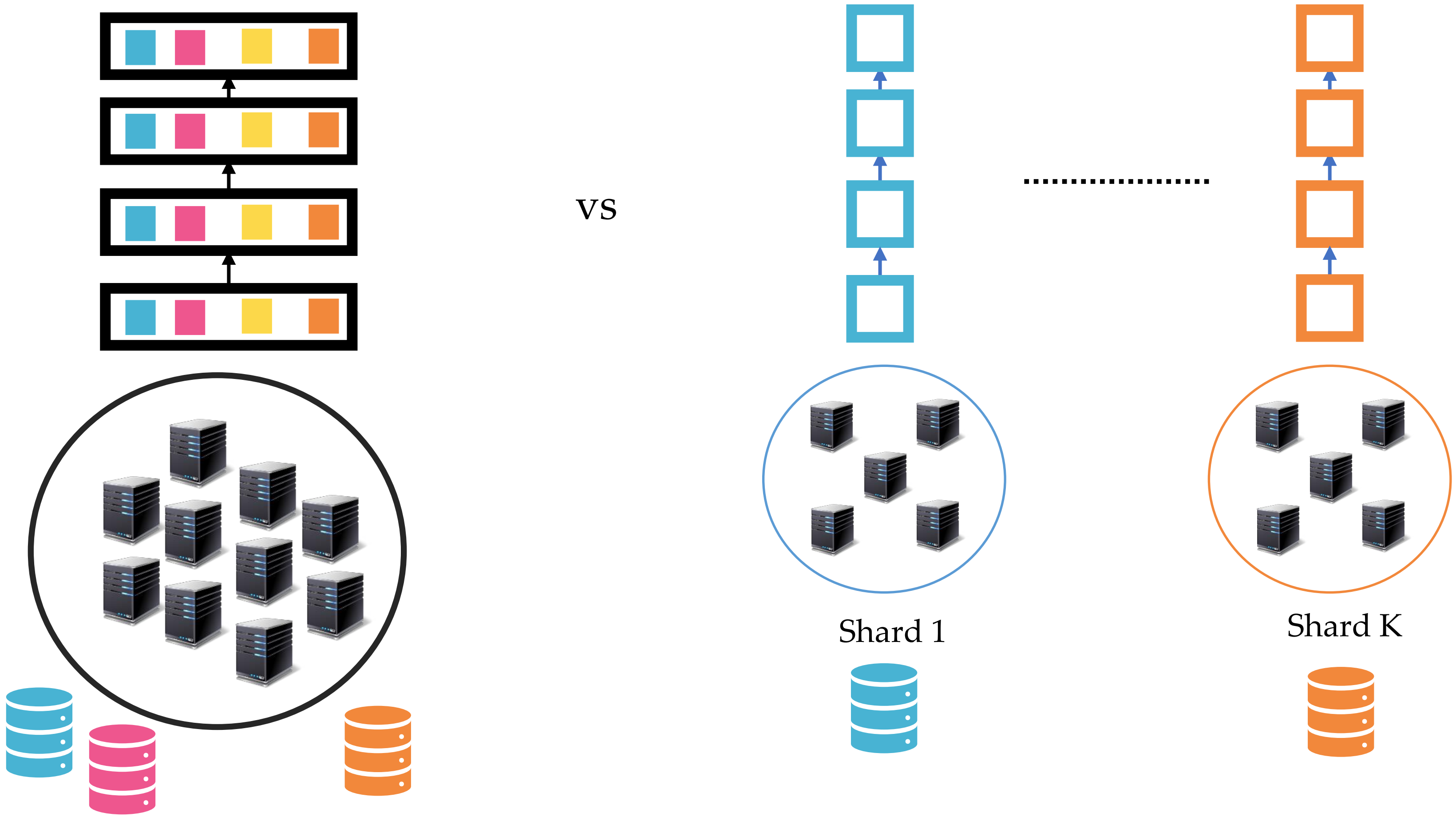
Multiconsensus

- Each shard runs its own consensus



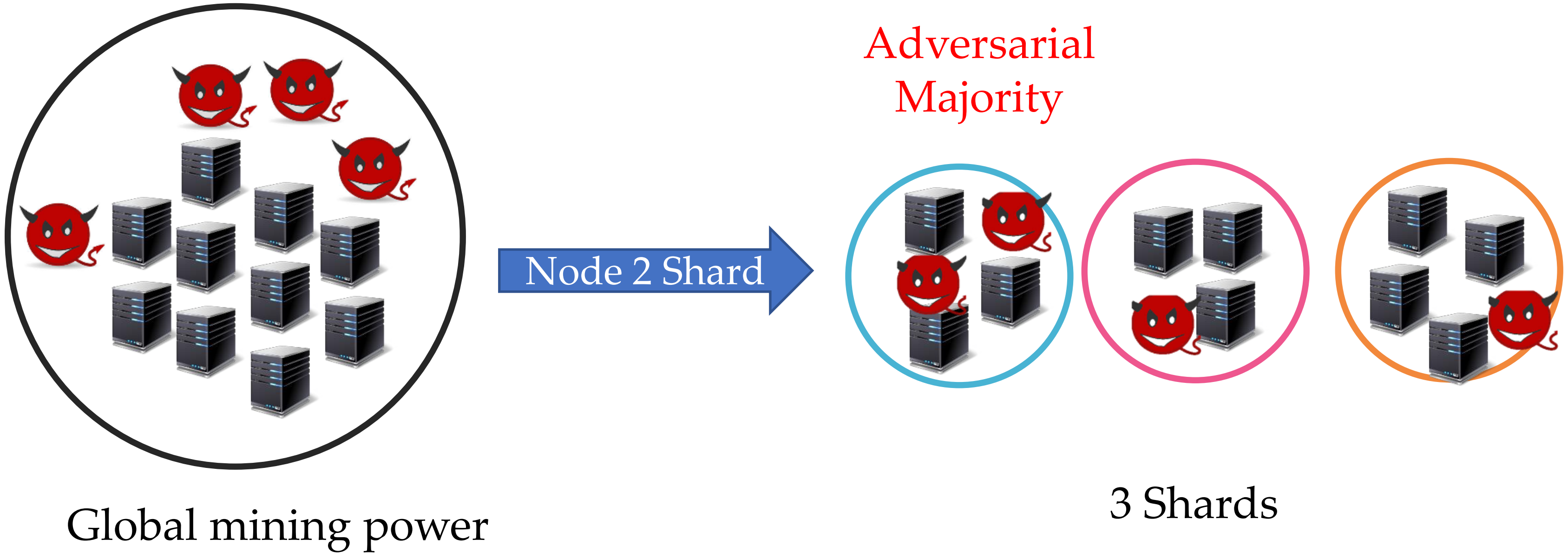
Scaling: O(K)

- Nodes only maintain the state of their own shard



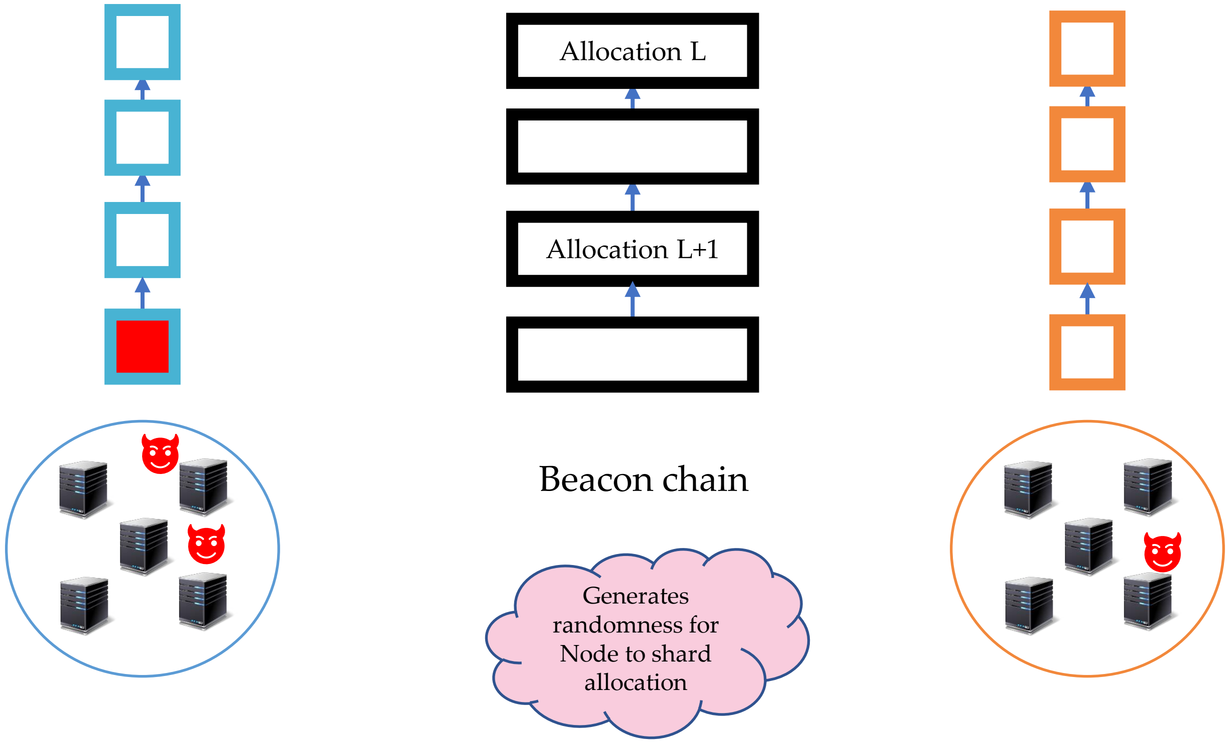
Drawback 1: Proportional representation

- Need large number of nodes per shard to ensure honest majority in a shard



Drawback 2: Security

- Not resilient to O(1/K) adaptive adversary

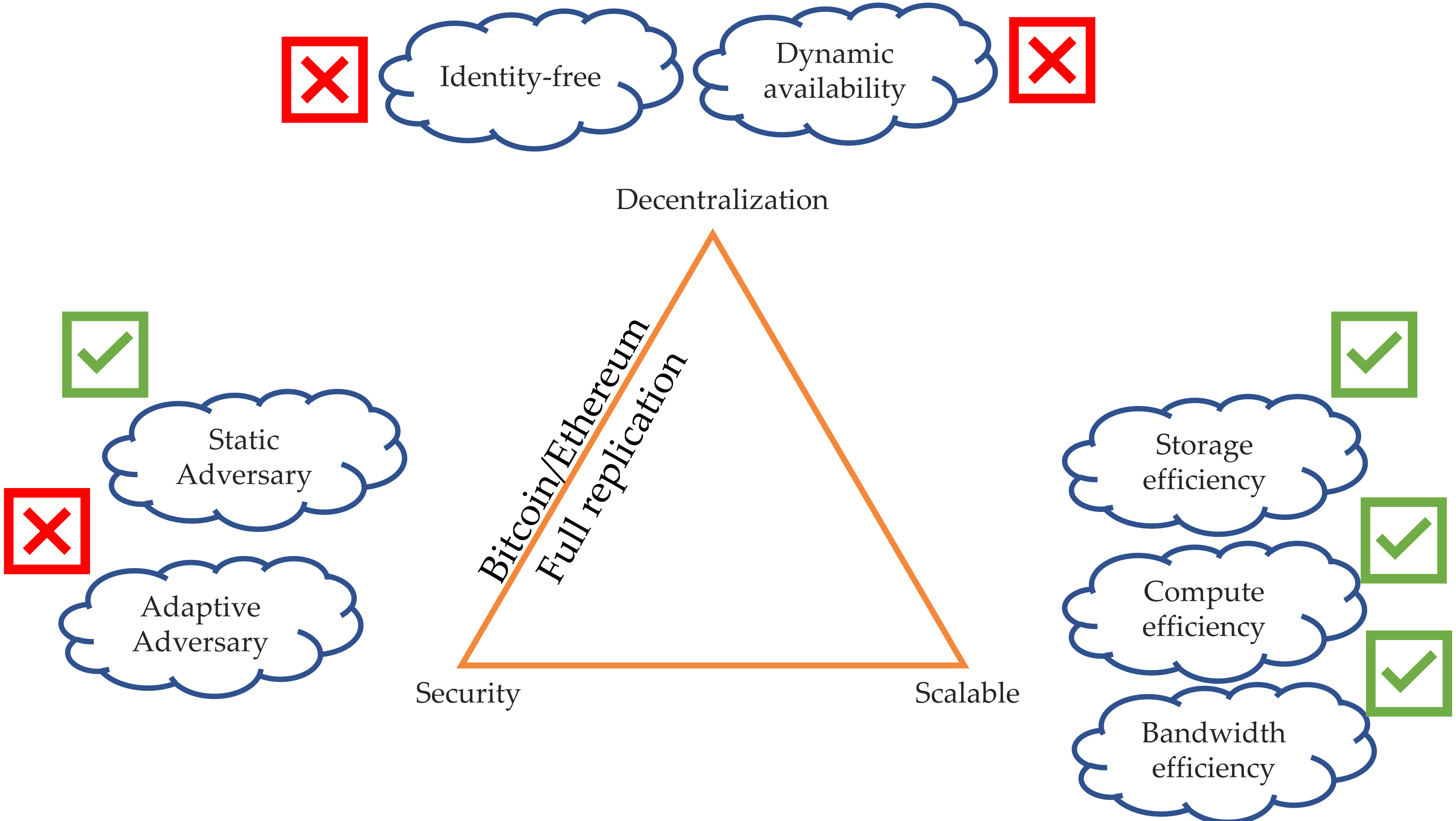


Drawback 3: Node identity

- Existing works vary in the implementation of Node to shard allocation
  - Different ways of randomness generation
  - Different rate of re-allocation

- All Node to shard allocation algorithms require consensus node identity

Trilemma revisited



Identity-free sharding

- If the protocol is identity free, we cannot use Node to Shard allocation algorithm

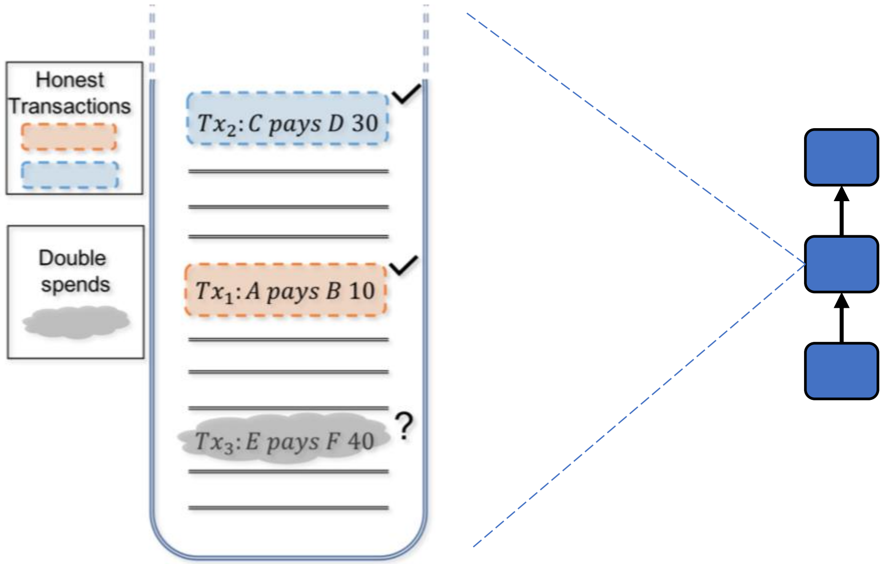
- Only choice is to allow nodes to **self-allocate**

- Adversaries can congregate on one shard; safety and liveness can be easily broken

- Solution: Uniconsensus architecture

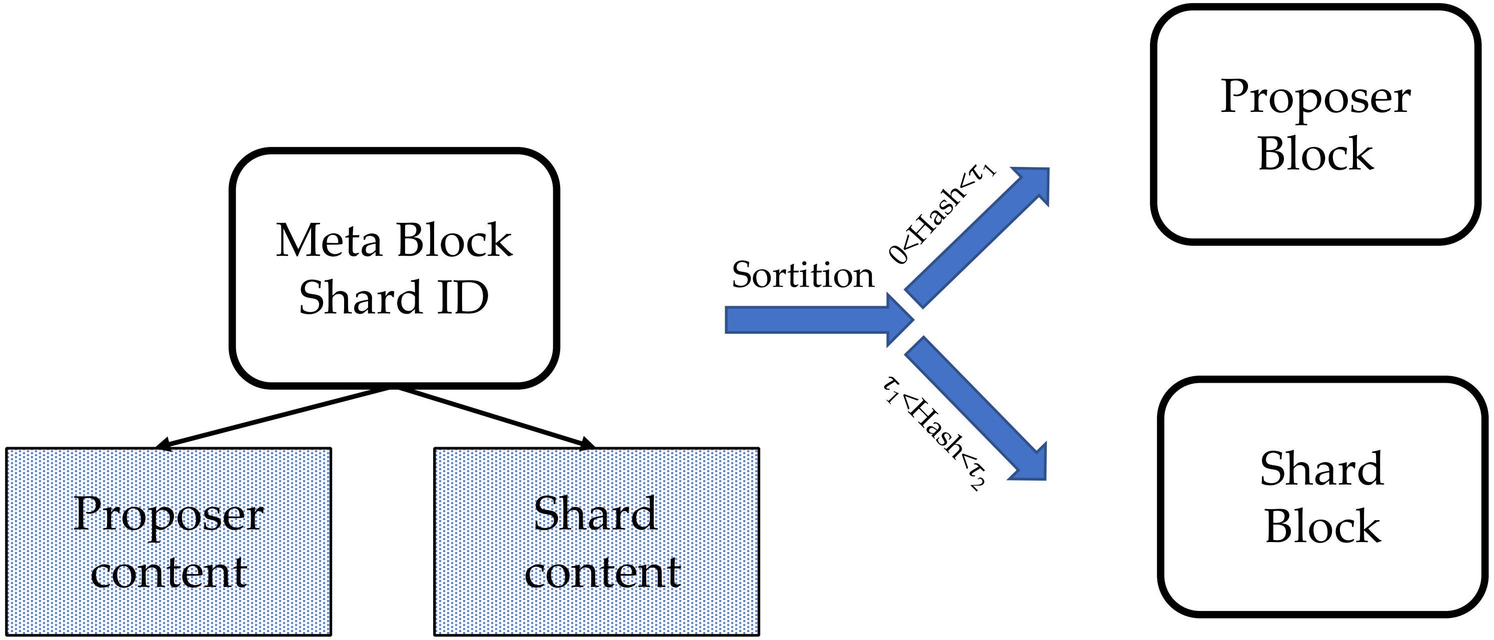
Uniconsensus: Decoupled validation

- Shard transaction ordering can be decoupled from validation
- Extension of the deconstruction ideas from Lecture 7 (Fruitchains) and Lectures 8 and 9 (Prism)

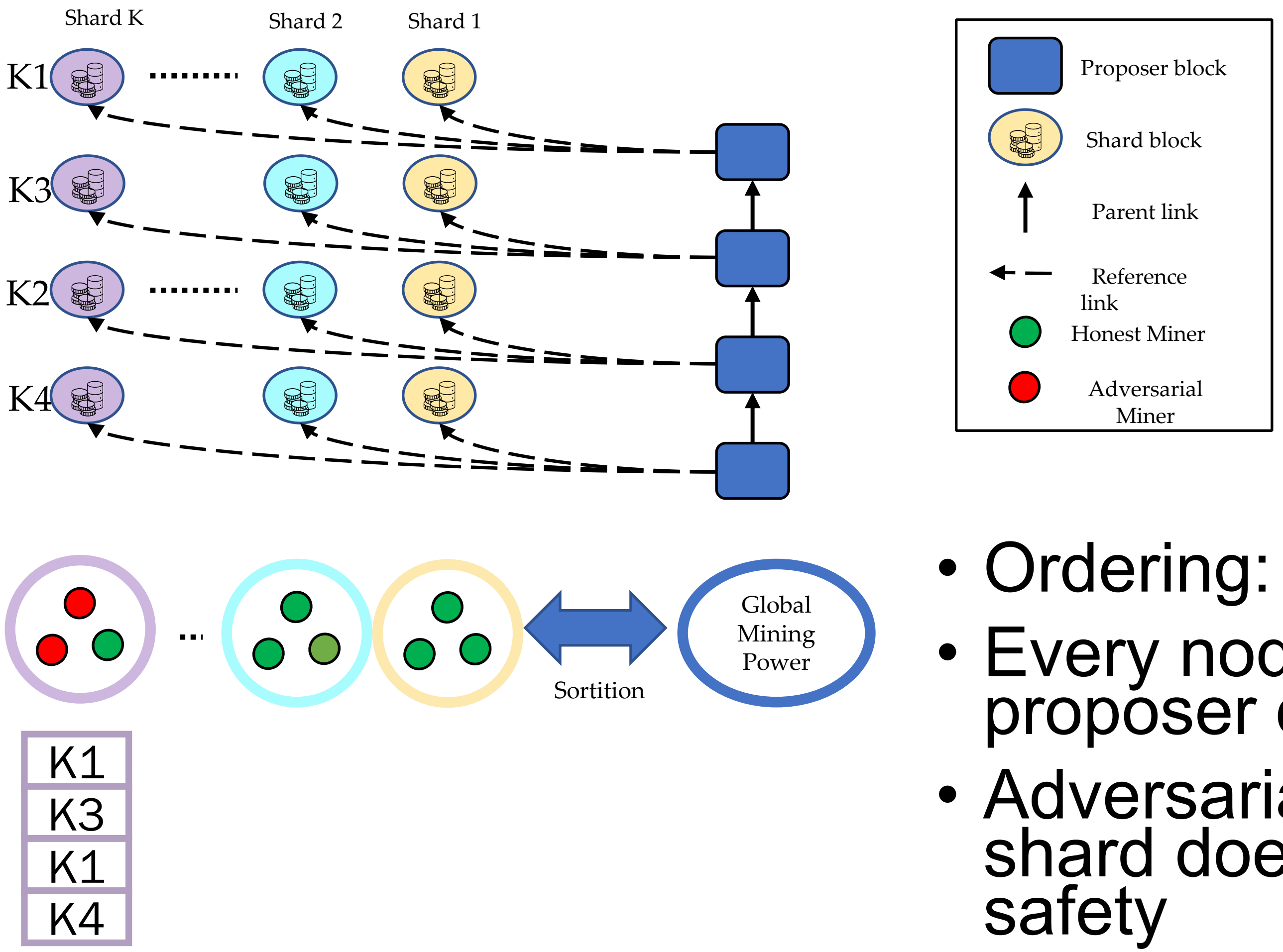


Uniconsensus: Sortition

- A node can maintain any shard of its choice
- It will maintain an ordering chain in parallel
- All nodes mine shard block and proposer block together



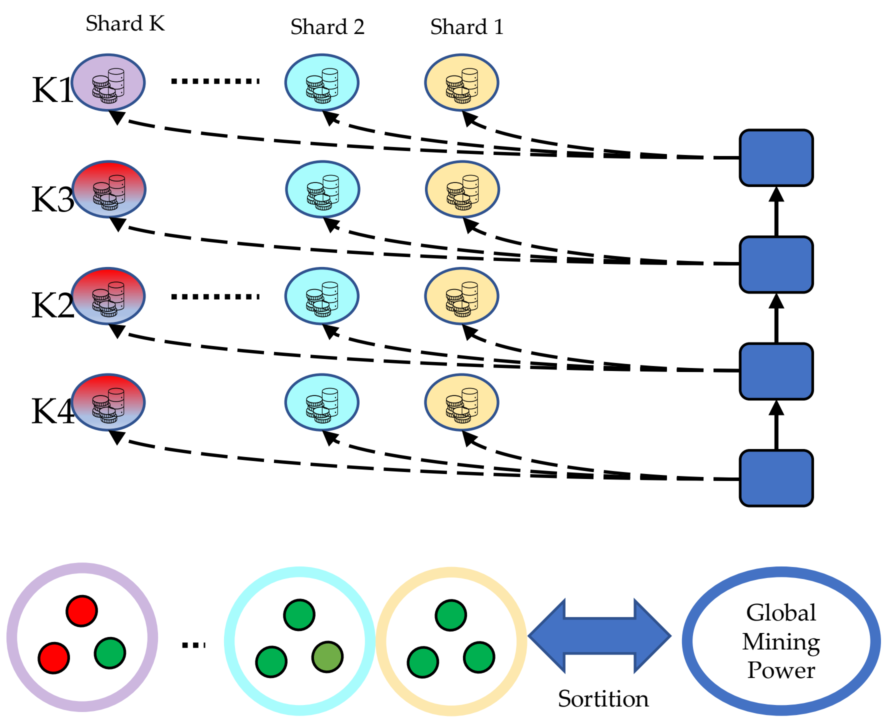
Uniconsensus: Safety



- Ordering: Proposer chain
- Every node maintains proposer chain
- Adversarial majority in a shard does not violate safety

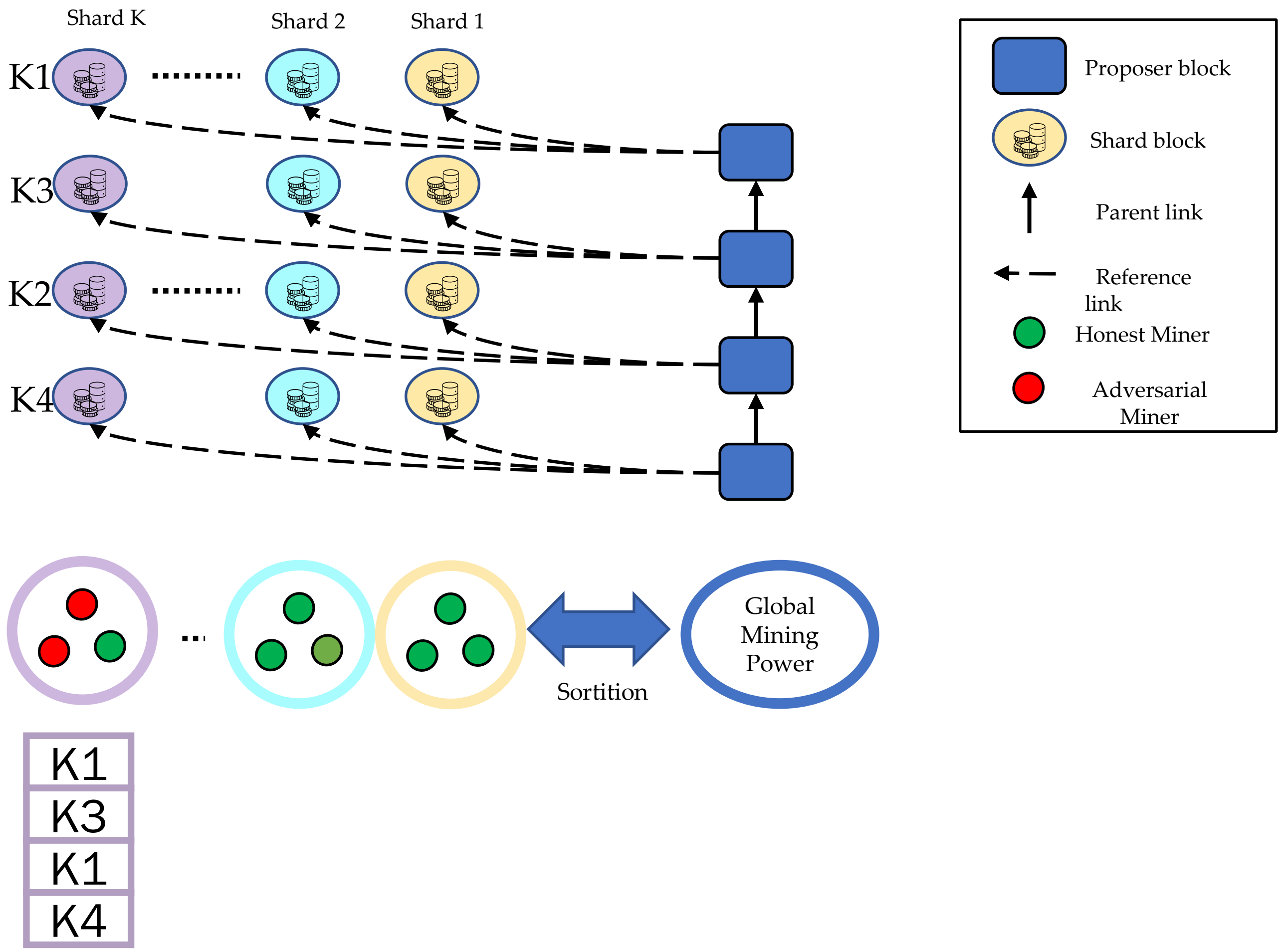
Uniconsensus: Liveness

Chain-quality visualization



- Adversaries congregate: drown out honest miners
- Worst case shard chain-quality is O(1/K)
- Dynamic self allocation can prevent such attacks

Uniconsensus Architecture



- Safety shared
- Liveness is sacrificed

Dynamic Self-allocation

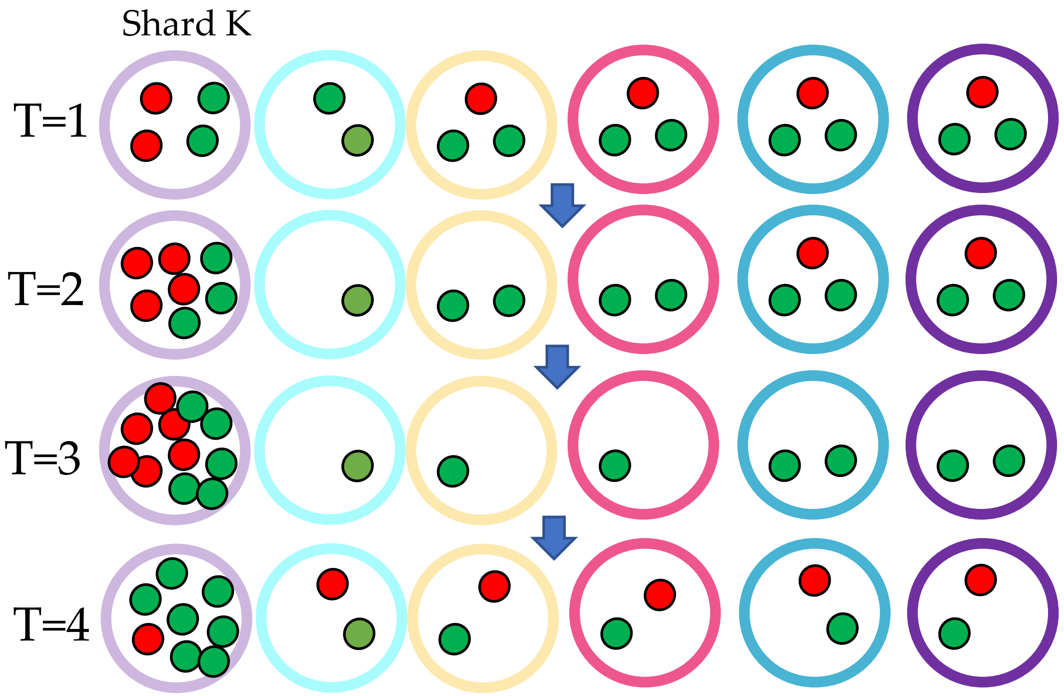
- Honest nodes adapt to adversaries and allocate themselves to new shards

- We want honest nodes to (re)allocate themselves to shards under attack

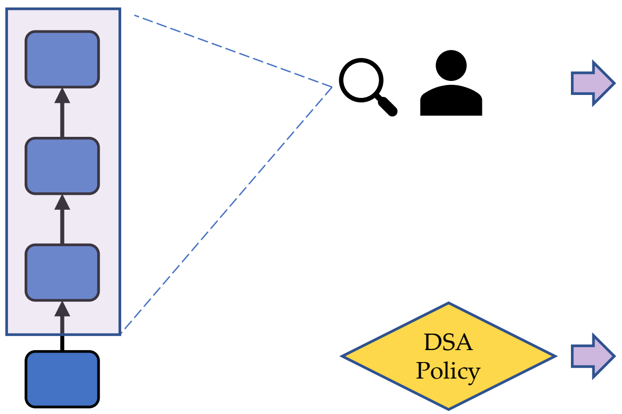
Simple DSA

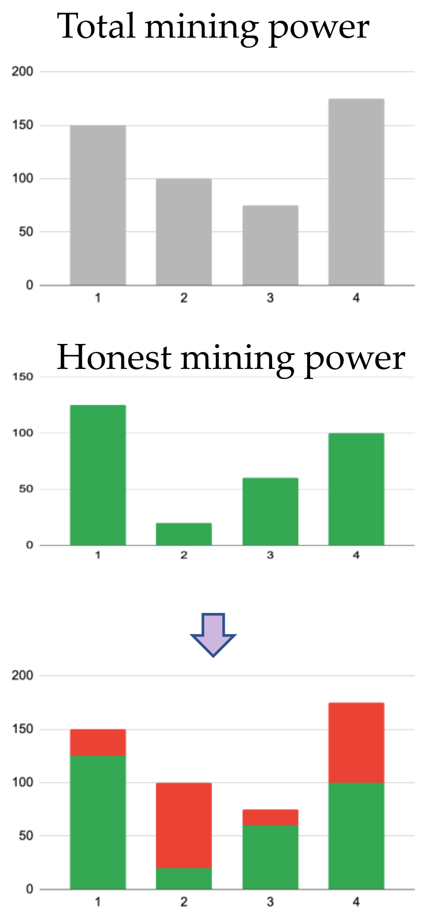
- Adapt to the adversary by following the adversary's last move
- Assume that the adversary's past allocations are known

- Worst-case shard chain quality is O(1/log(K))



Practical Implementation

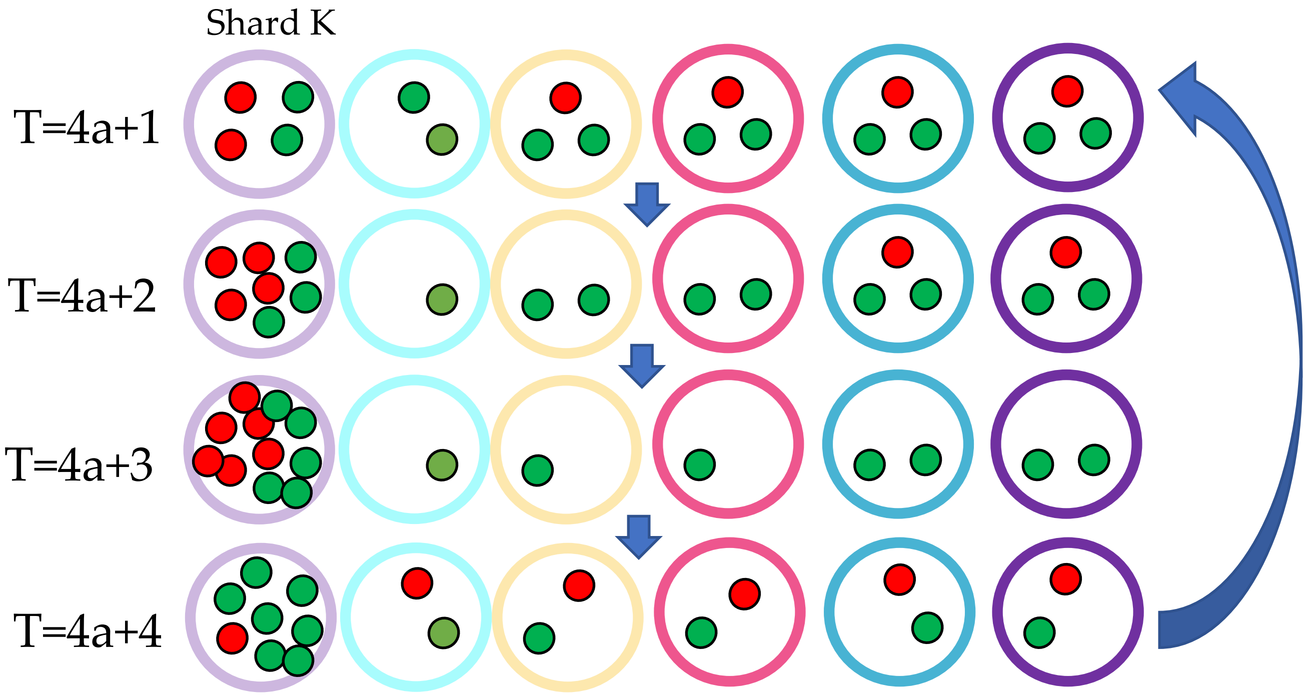




- Honest miners estimate total mining power from proposer chain
- Honest mining power allocation can be estimated from DSA policy

Simple DSA: Optimal adversary

- Optimal adversary attacks a shard by gradually increasing it's mining power by a factor of log(K) over each epoch
- The adversary is cyclic



Free2Shard DSA

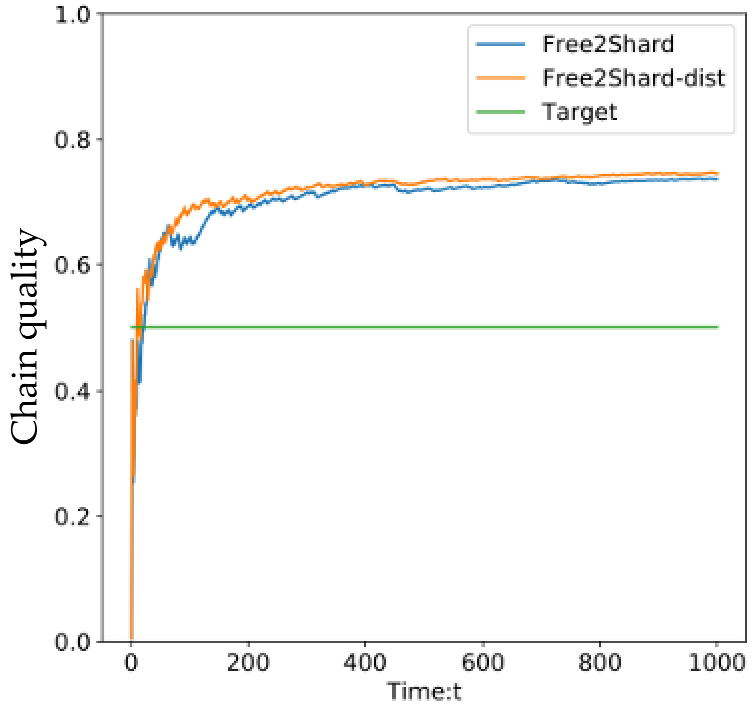
- Information theoretic bound: We cannot achieve worst-case shard chain quality better than the global honest mining power.
- Simple DSA policy is myopic
- Free2Shard DSA allocates honest mining power to shards proportional to how far behind the average chain-quality of the shard is to a target value

Free2Shard DSA

- Theorem: Performs arbitrarily close to information theoretic bounds
- Stackelberg game between adversary and honest nodes
  - Honest nodes make the first move, adversary can allocate later
- Proof has parallels with classical Blackwell approachability ‘56

Free2Shard simulations

- Theoretical convergence at the rate of $O(\frac{\sqrt{K}}{\sqrt{T}})$
- Experiments show convergence in less than 50 epochs (N=1000 nodes and K=100 shards)



State commitments: Bootstrapping

- Both Uniconsensus and Multiconsensus depend on nodes relocating to new shards
- Efficient methods of bootstrapping should be used

1. Download latest state of a shard shard (instead of the whole ledger)
2. Verify state from the latest state-commitment
3. State-commitment is a root of the Merkle tree of execution state

State-commitments: Multiconsensus

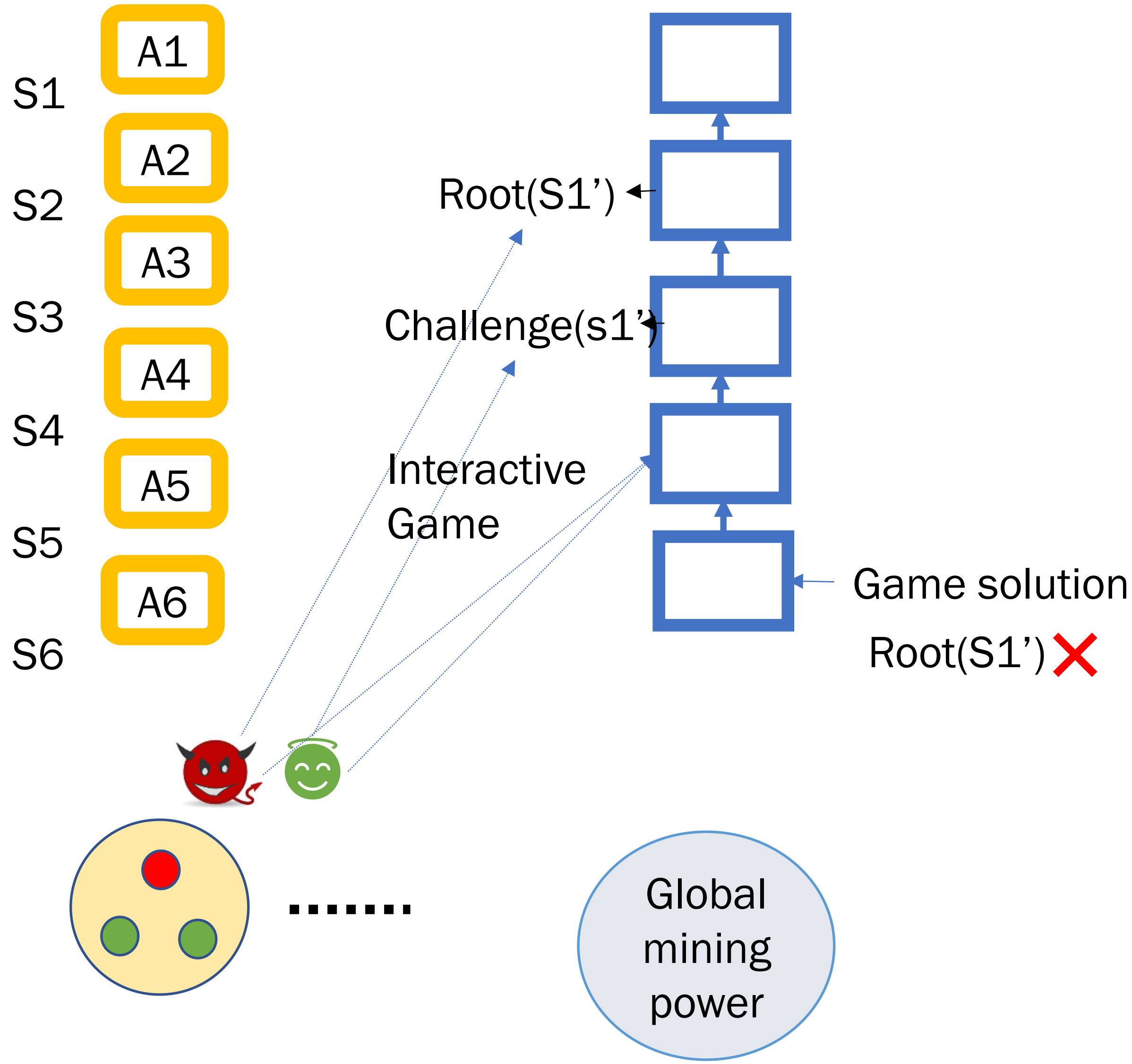
- Assumption: Each shard has honest majority

- Two possible approaches:

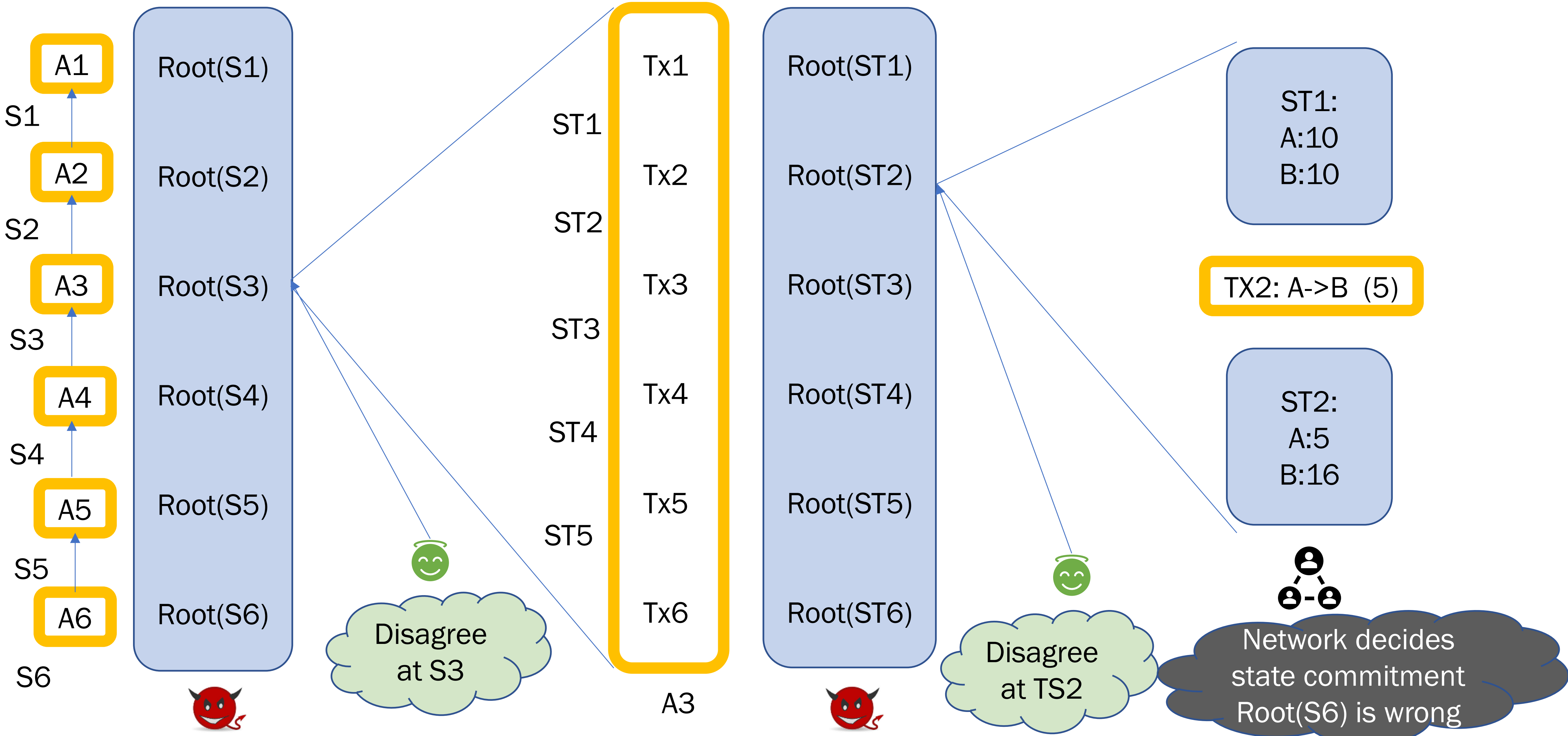
1. A state-commitment is correct if signed by majority in a shard
2. A state-commitment is correct if a transaction including one is finalized in the shard’s ledger

State-commitments: Uniconsensus

- **Problem**: No honest majority assumption and no coupled validation
- **Solution**: Interactive fraud proof mechanism for detecting incorrect state-commitments
- Logarithmic fraud search between two parties: Proposer and Challenger



Interactive fraud-proof: Uniconsensus



Conclusion

- Sharding fell out of fashion
  - Changes at the consensus layer (L1 layer) is too onerous
  - Both engineering and operational viewpoints

- Rollups (Layer 2) is the preferred approach for scaling
  - Aribtrum, Optimism

Attendance : NFT Drop

https://poap.website/likely-box-organization

- Mint token to Metamask.
- Submit tx hash for attendance claim

- Instructions in Ed pinned posts.