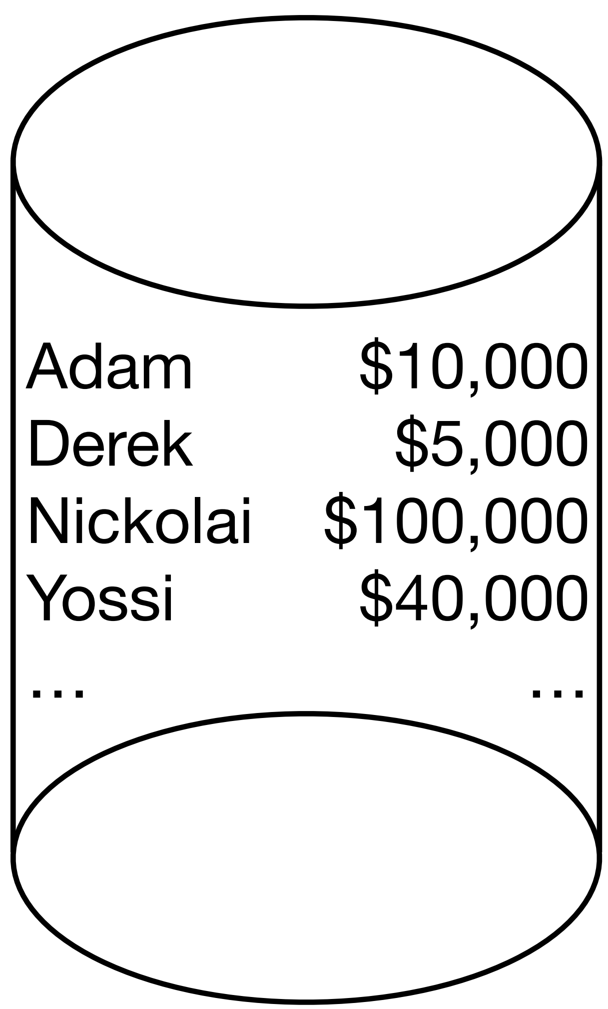
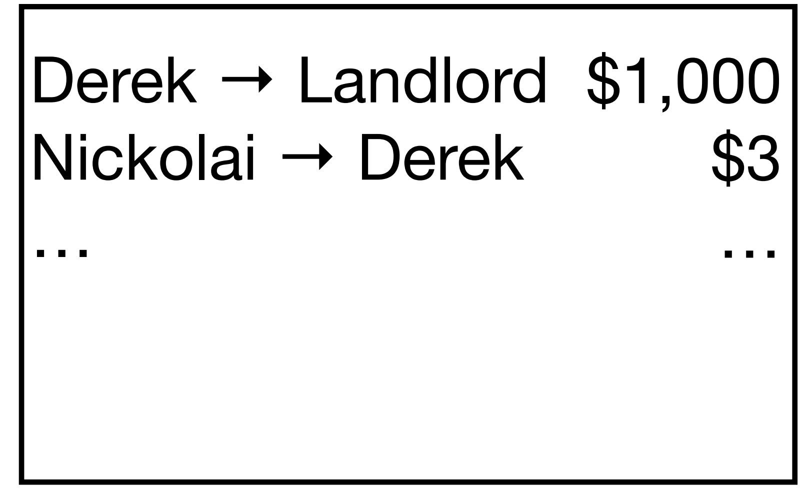
Vault: Fast Bootstrapping for the Algorand Cryptocurrency

Derek Leung, Adam Suhl, Yossi Gilad, Nickolai Zeldovich

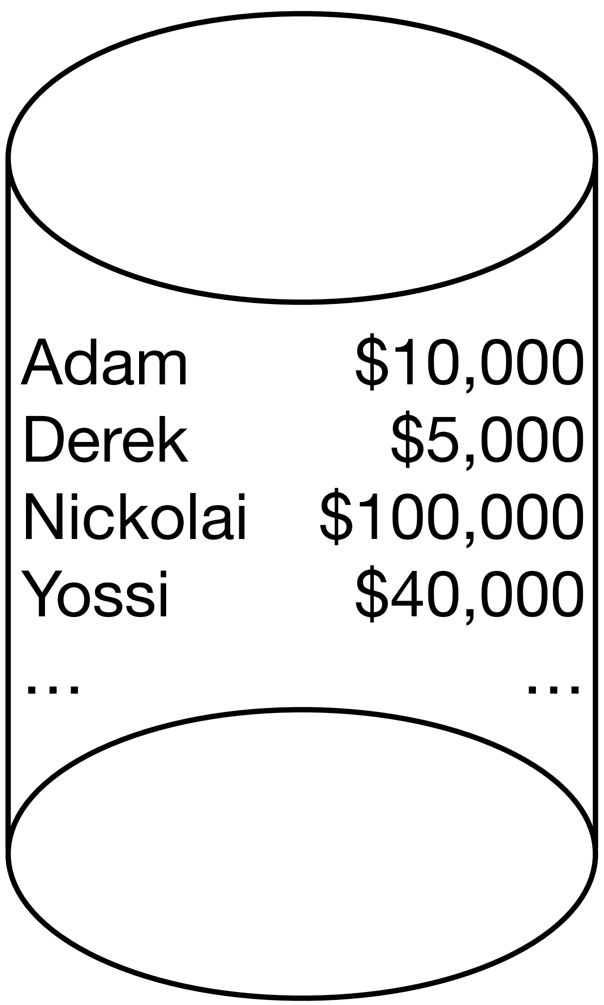
Algorand / MIT CSAIL



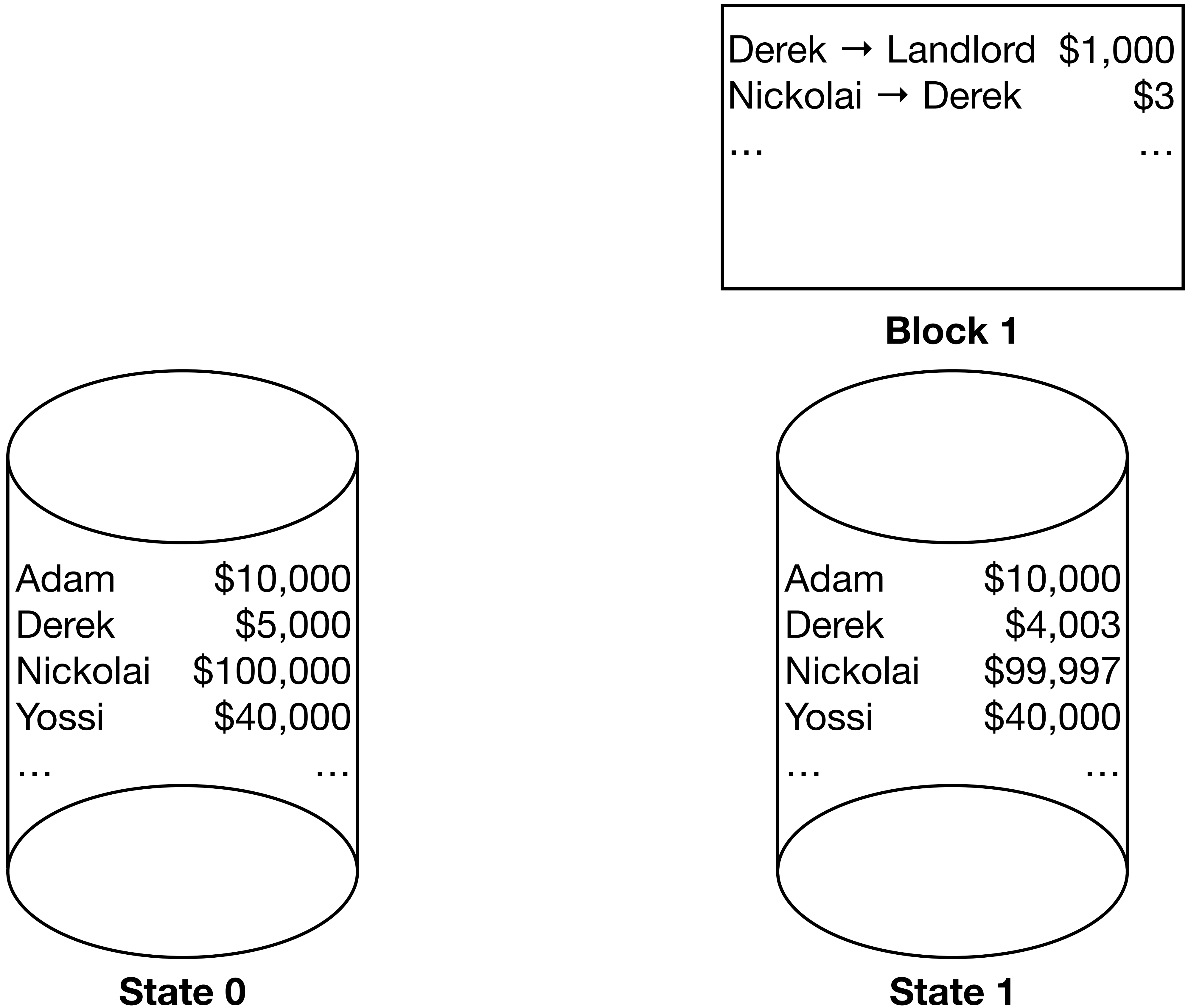
State 0

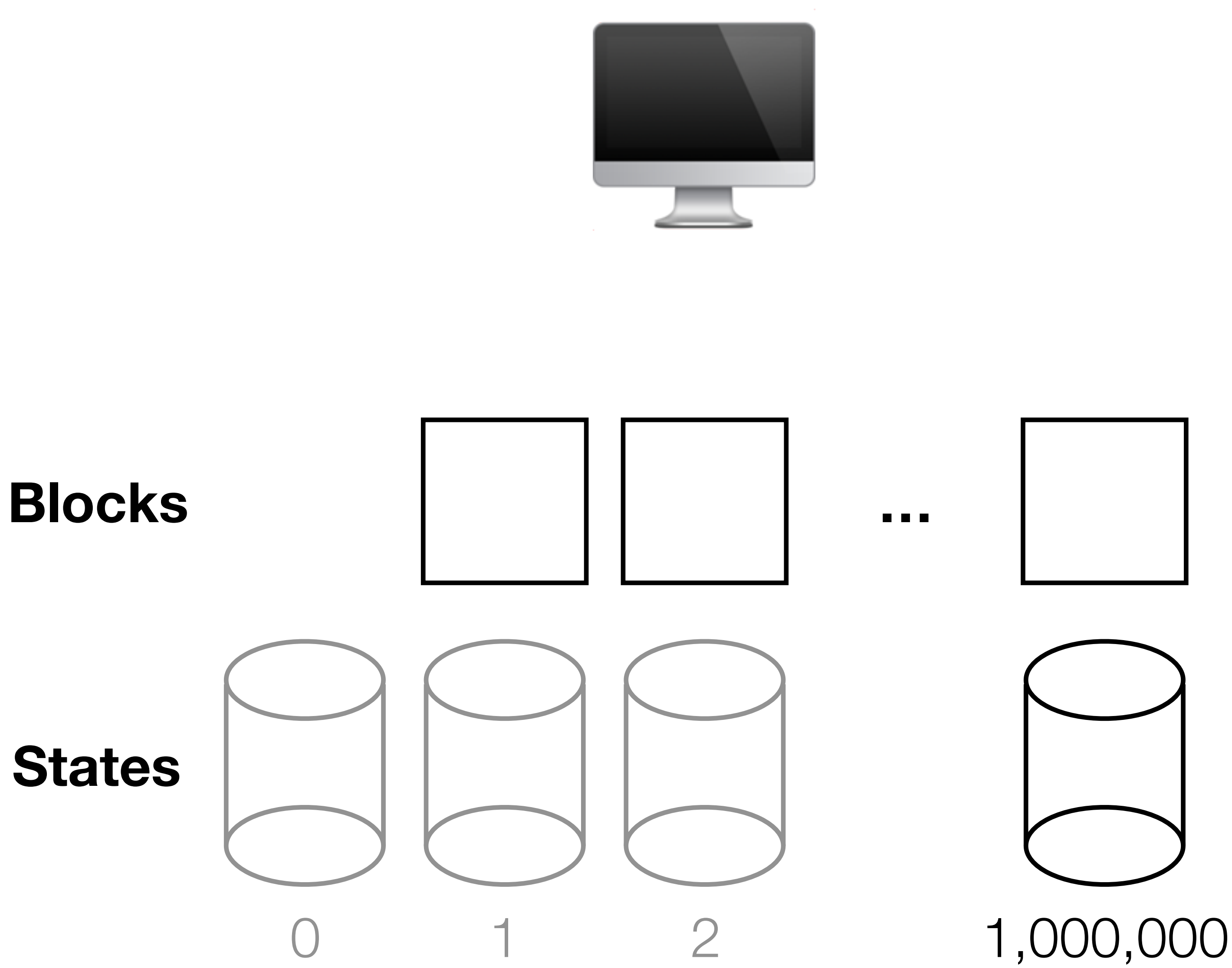


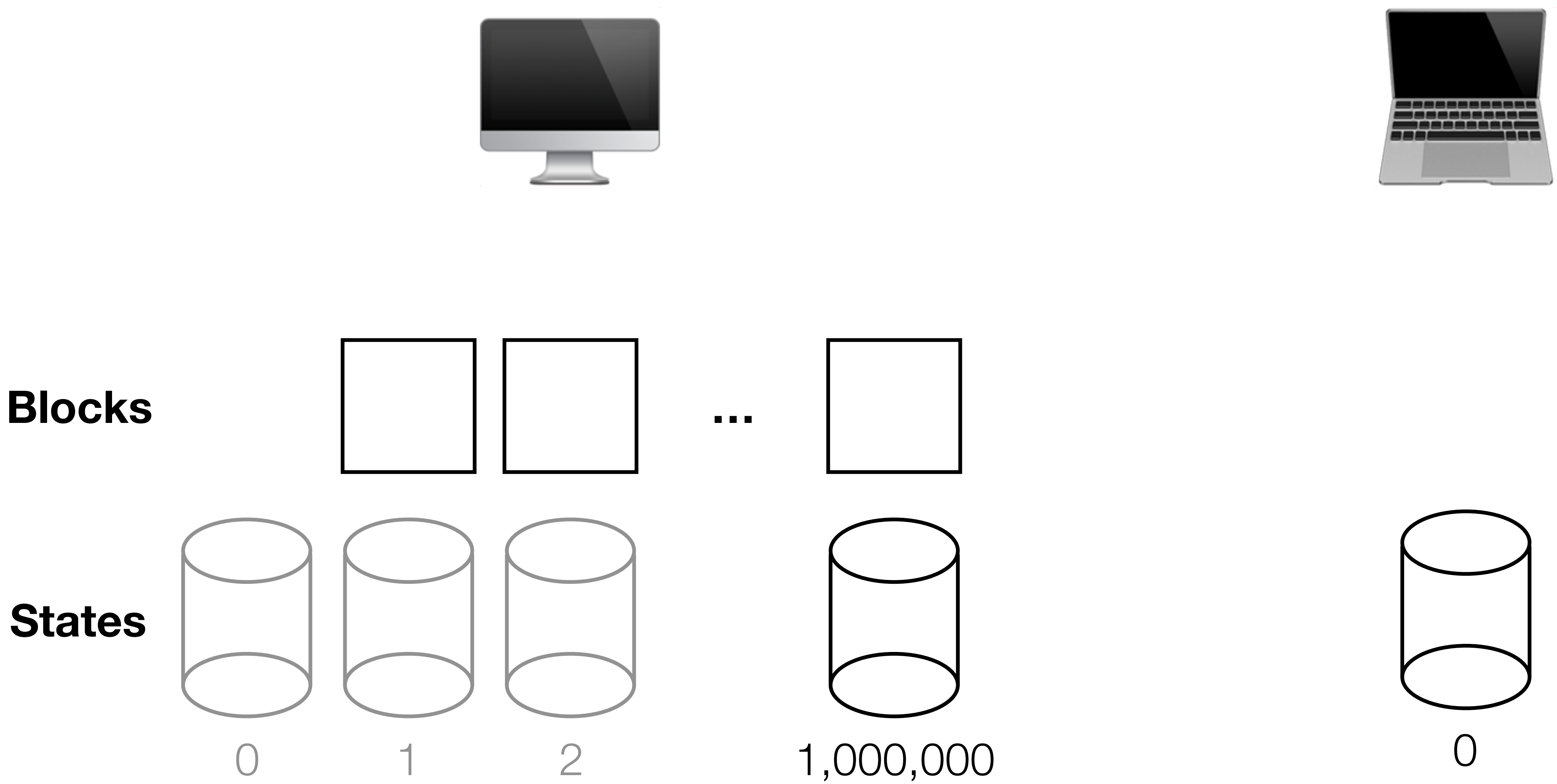
Block 1

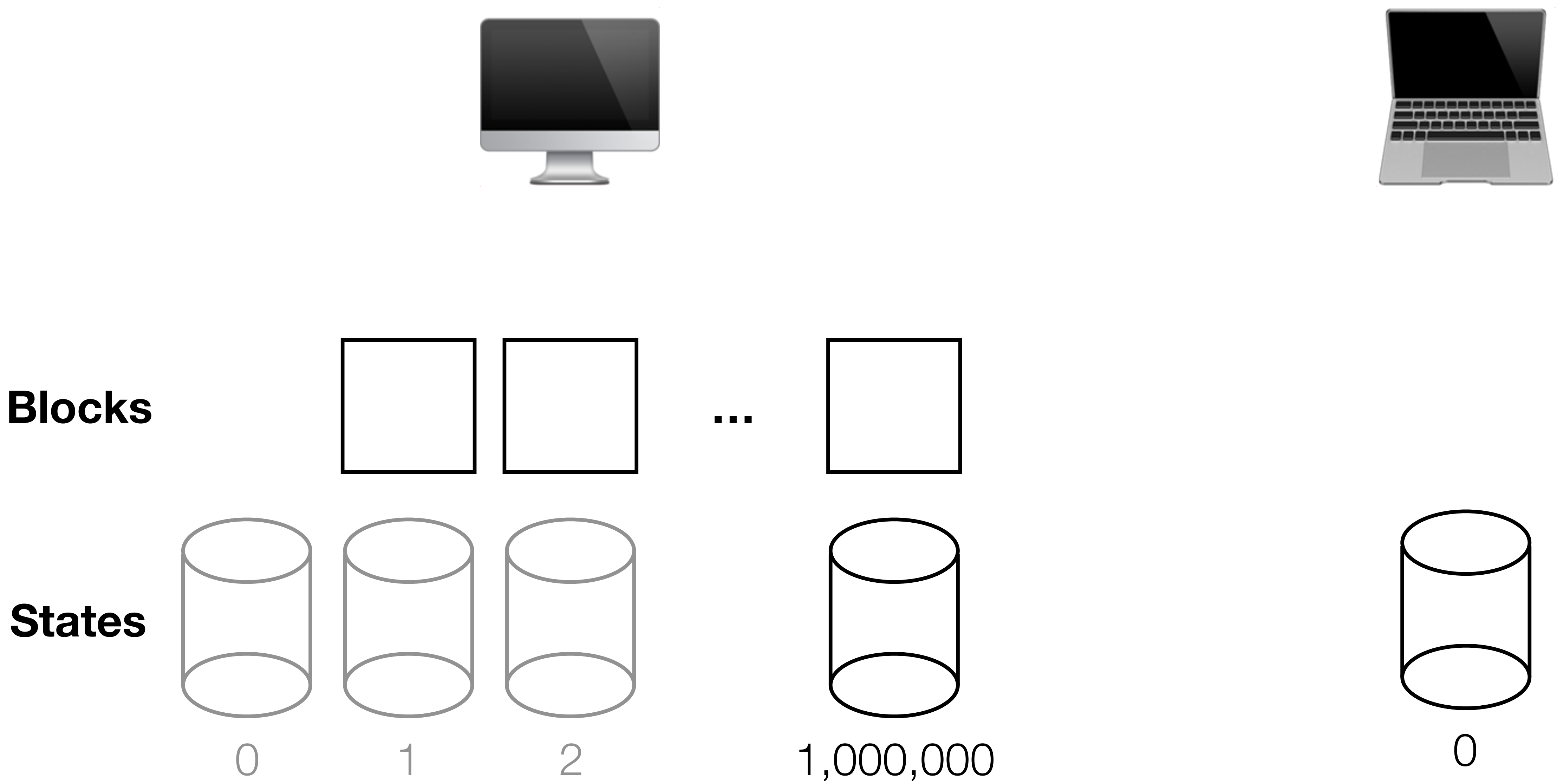


State 0


Derek → Landlord $1,000
Nickolai → Derek $3
… …
Block 1
Adam $10,000
Derek $5,000
Nickolai $100,000
Yossi $40,000
… …
State 0
Adam $10,000
Derek $4,003
Nickolai $99,997
Yossi $40,000
… …
State 1

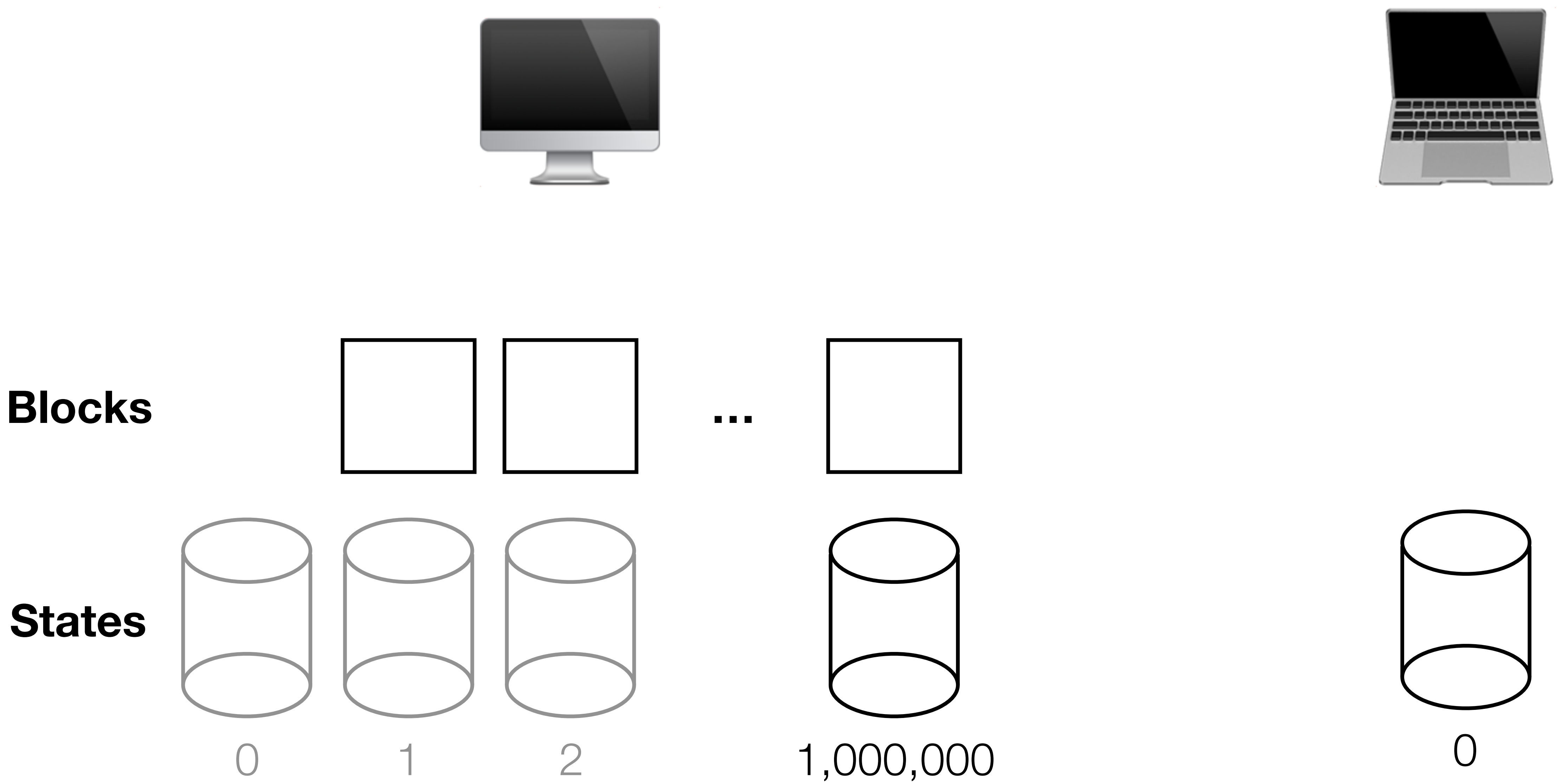

Blocks
...
States
0
1
2
1,000,000


Blocks
...
States
0
1
2
1,000,000
0



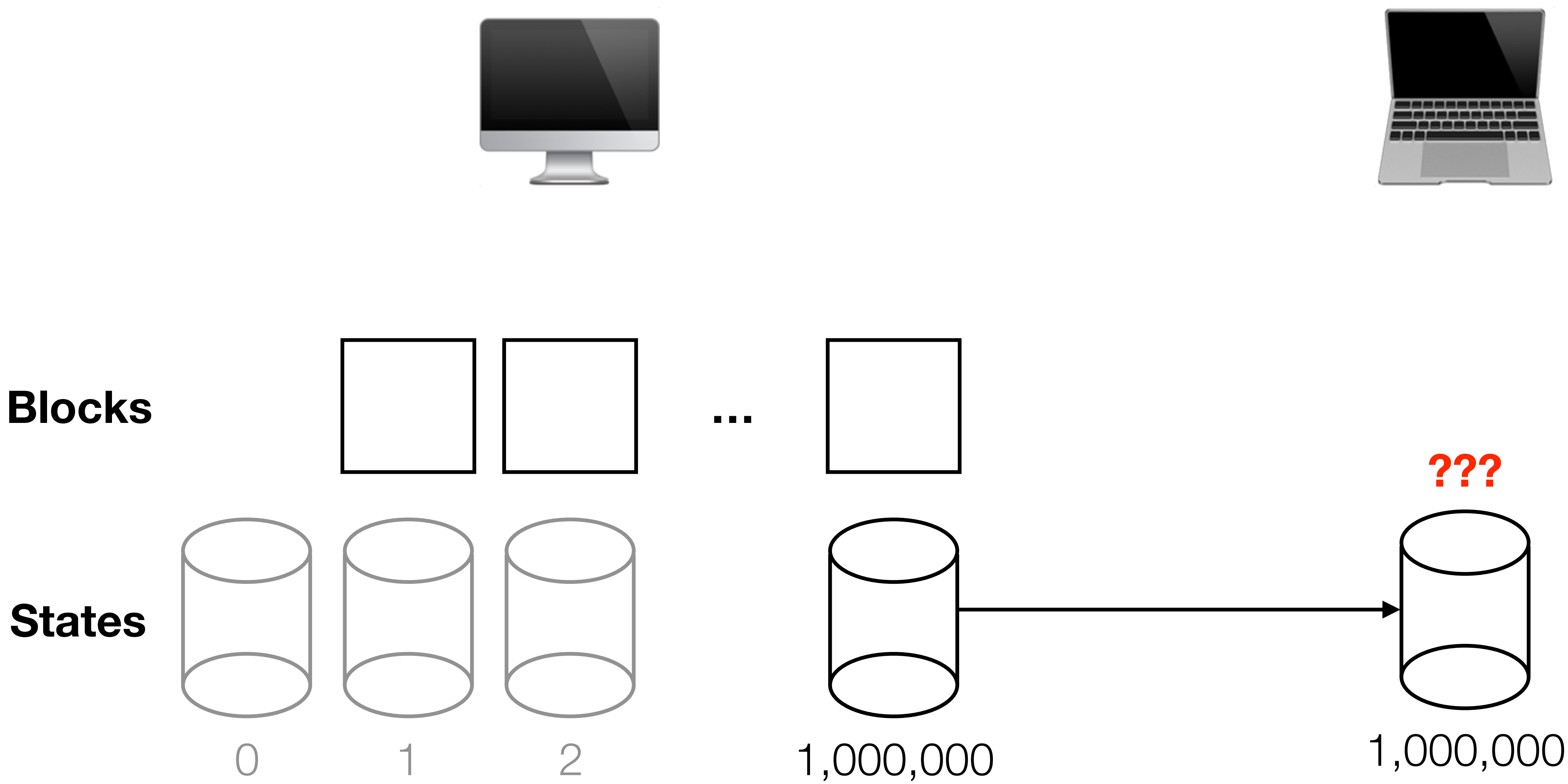
How does a new user join the system?

Bootstrapping



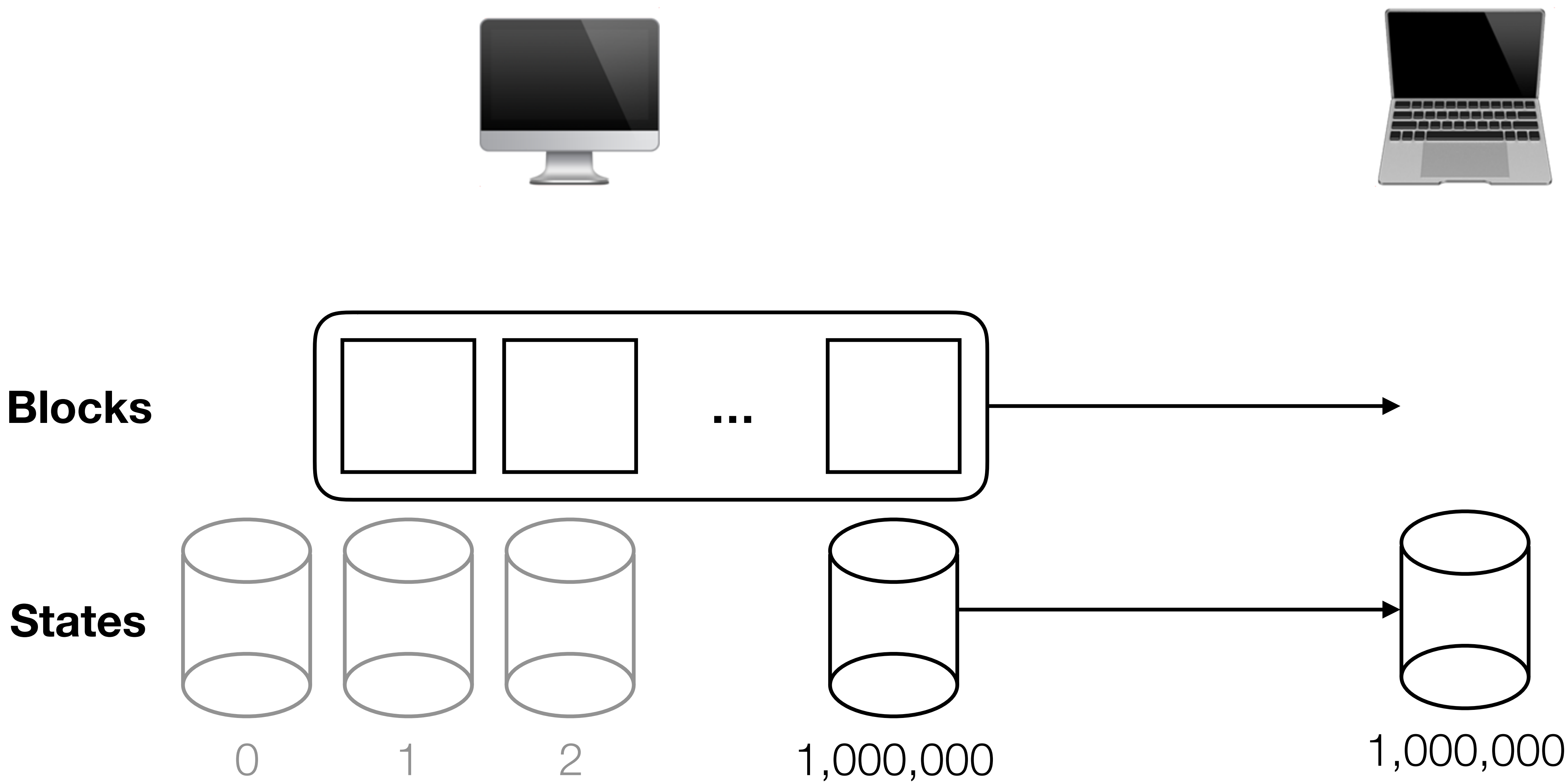
How does a new user join the system?

Bootstrapping



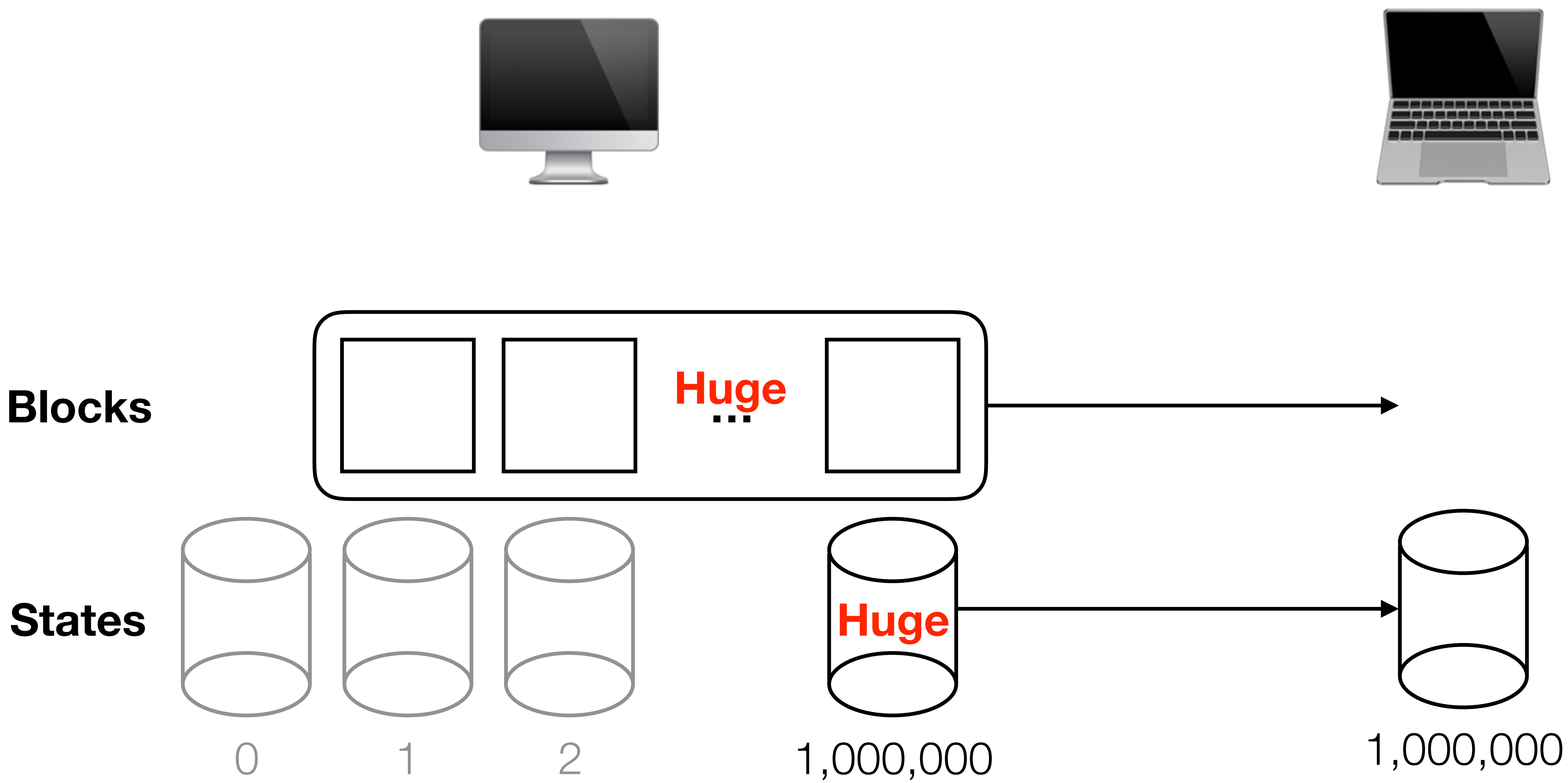
How does a new user join the system?

Bootstrapping



How does a new user join the system?

Bootstrapping



How does a new user join the system?

Bootstrapping

Goal: Securely and efficiently enable a new user to join, given initial state

- Minimize state transmitted
- Minimize proof that state is valid

Contribution

- Design of Vault, a system with secure and efficient bootstrapping
- 3 techniques for reducing sizes of state and proof of state
- 477MB data transfer cost for 500M transactions (Bitcoin: 143GB)

Vault Techniques

Approach	Challenge	Vault's Solution
Reduce state transmitted: Garbage collection	Transaction replay attacks	Force transactions to expire

Vault Techniques

Approach	Challenge	Vault's Solution
Reduce state transmitted: Garbage collection	Transaction replay attacks	Force transactions to expire
Reduce state transmitted: Shard state	Small shards lose security	Adaptive Merkle Tree sharding

Vault Techniques

Approach	Challenge	Vault's Solution
Reduce state transmitted: Garbage collection	Transaction replay attacks	Force transactions to expire
Reduce state transmitted: Shard state	Small shards lose security	Adaptive Merkle Tree sharding
Reduce size of state proof: Compress history	Attacker tampers with history	Succinct certificates

Vault Techniques

Approach	Challenge	Vault's Solution
Reduce state transmitted: Garbage collection	Transaction replay attacks	Force transactions to expire
Reduce state transmitted: Shard state	Small shards lose security	Adaptive Merkle Tree sharding
Reduce size of state proof: Compress history	**Attacker tampers with history**	**Succinct certificates**

Background

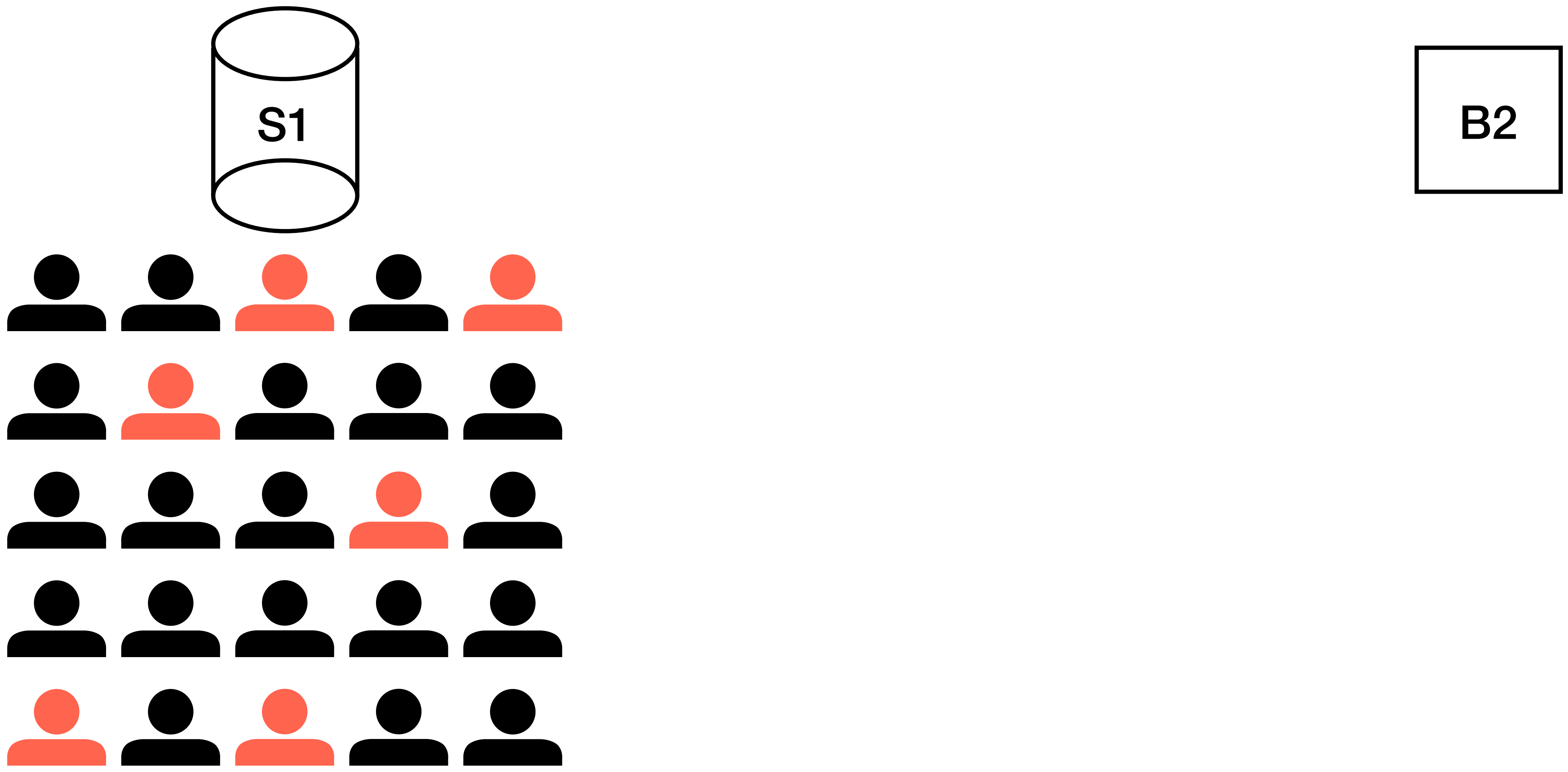
Algorand Background

- Permissionless
- Proof-of-stake
- Cryptographic proof that state is correct

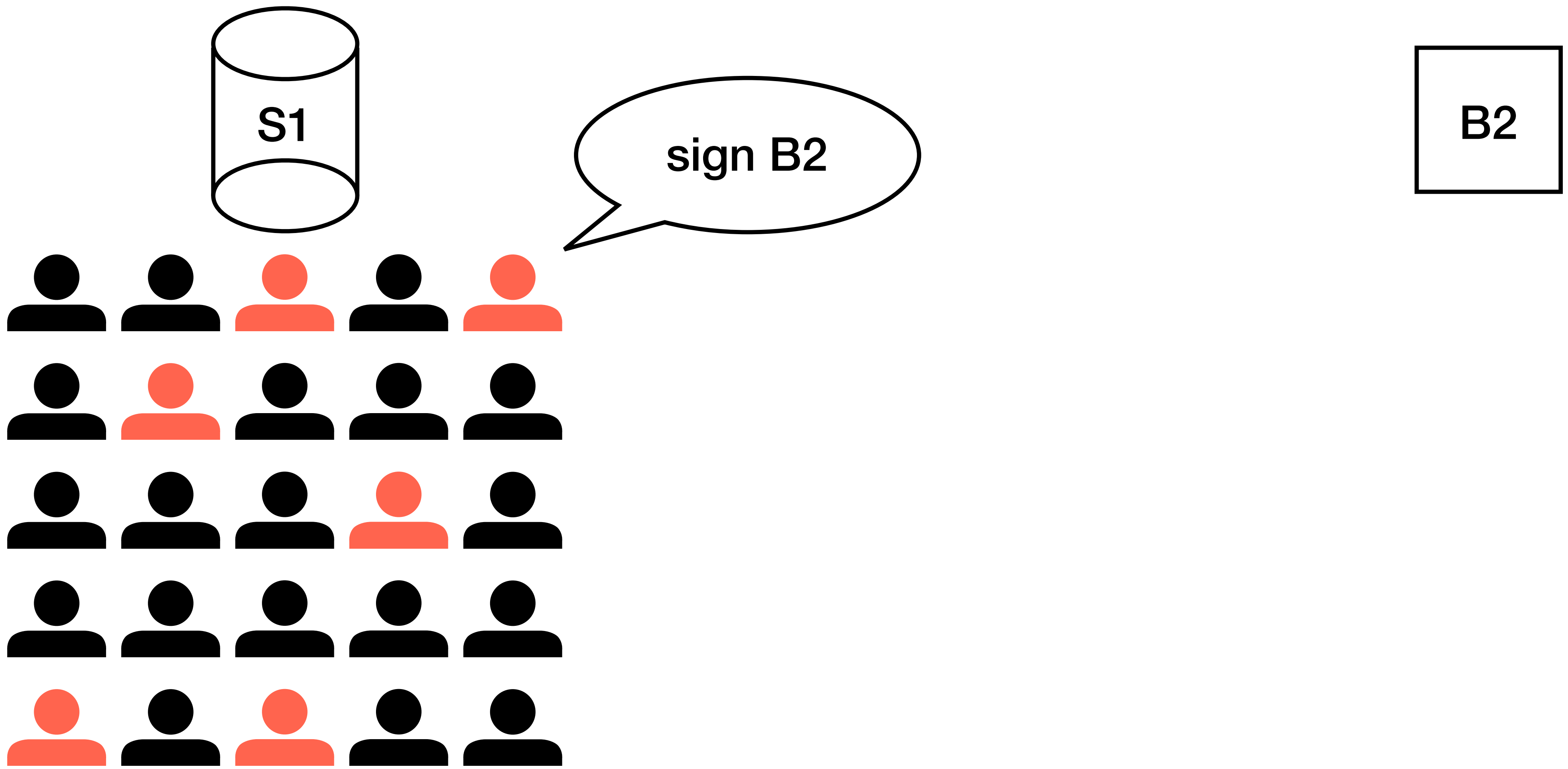
Security Model

- Standard cryptographic assumptions
- New user knows state 0 (i.e., the “genesis”)
- $f \leq 20\%$ of stake is malicious

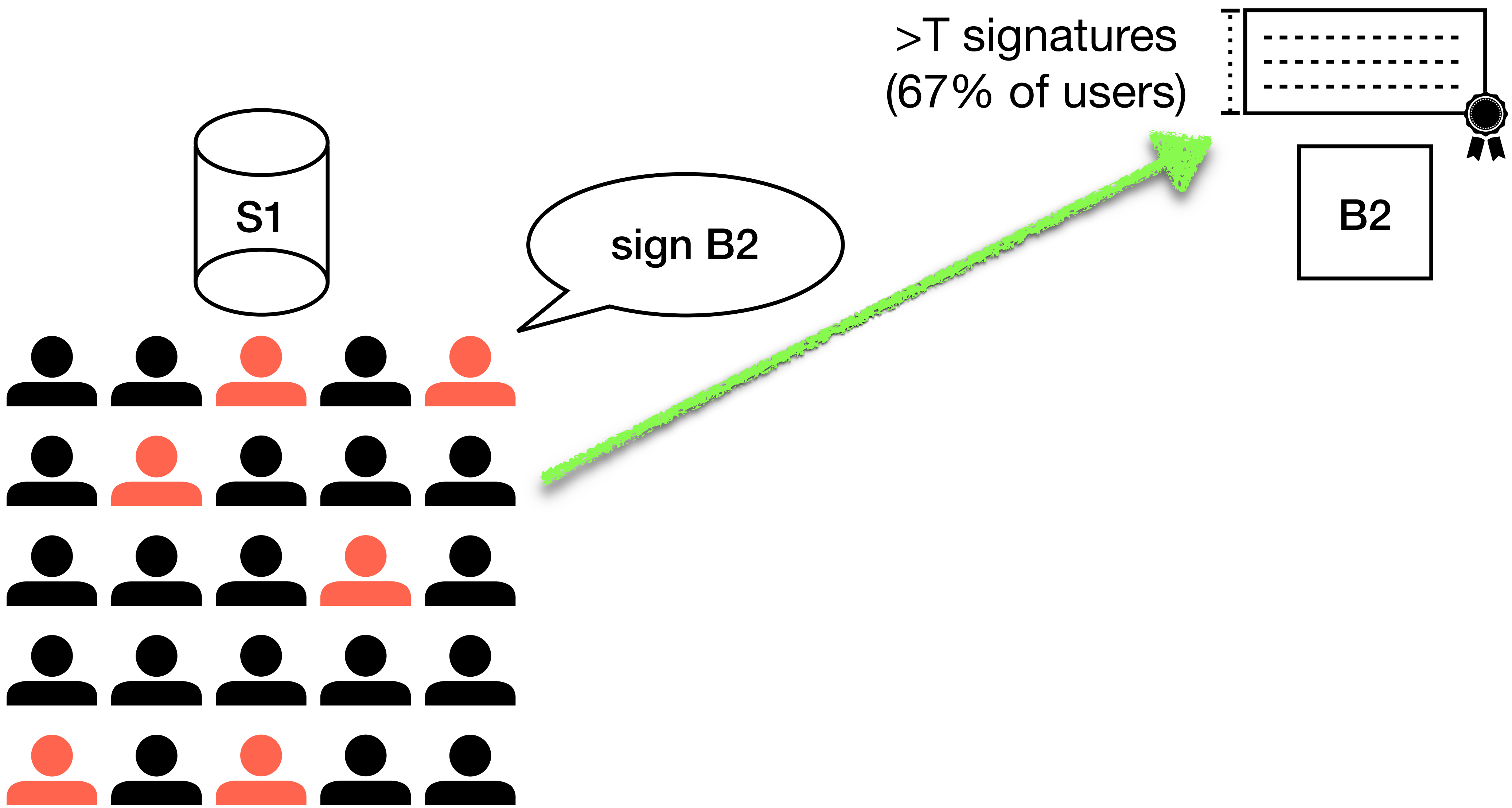
Proving a Block Correct



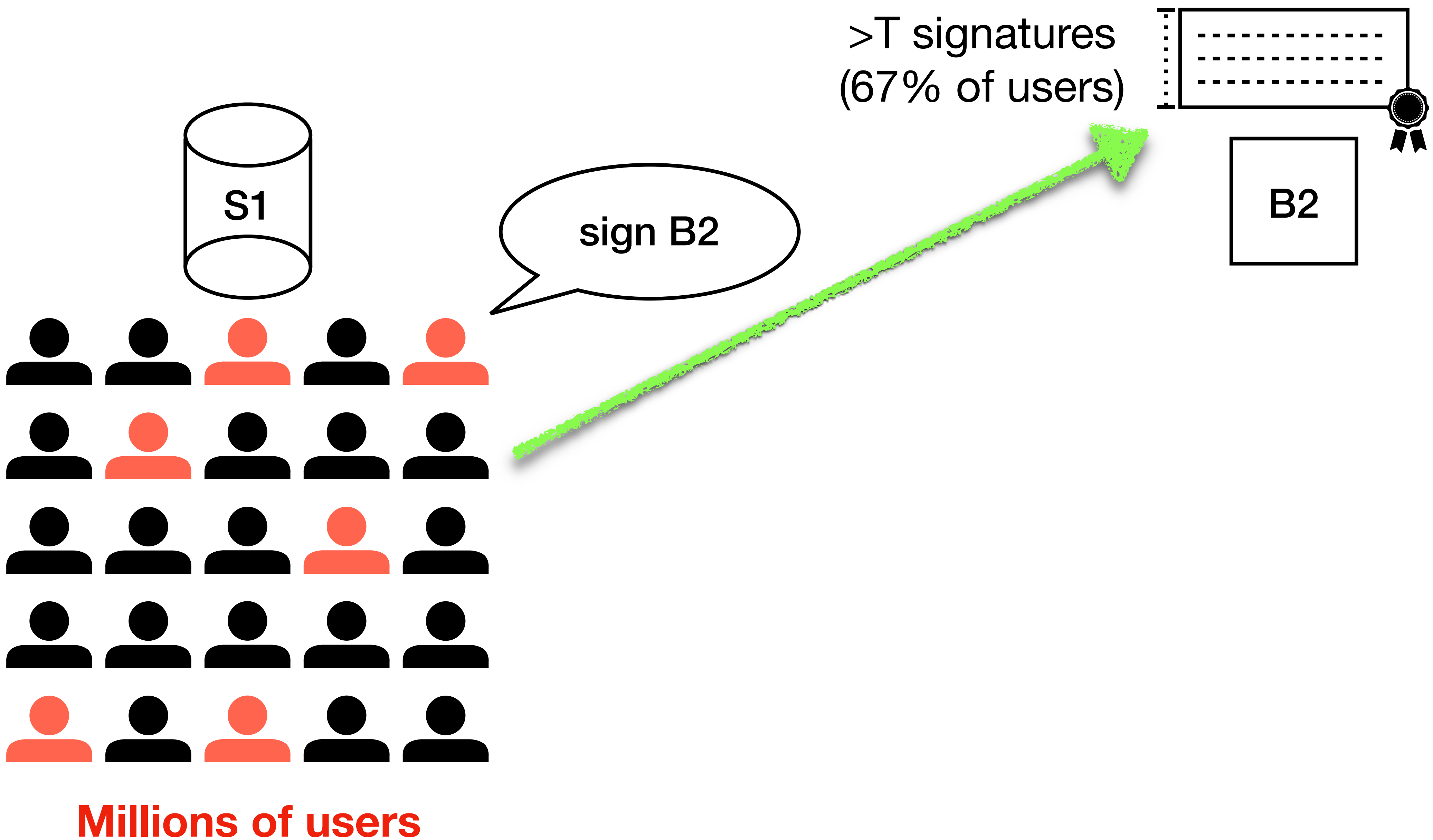
Proving a Block Correct



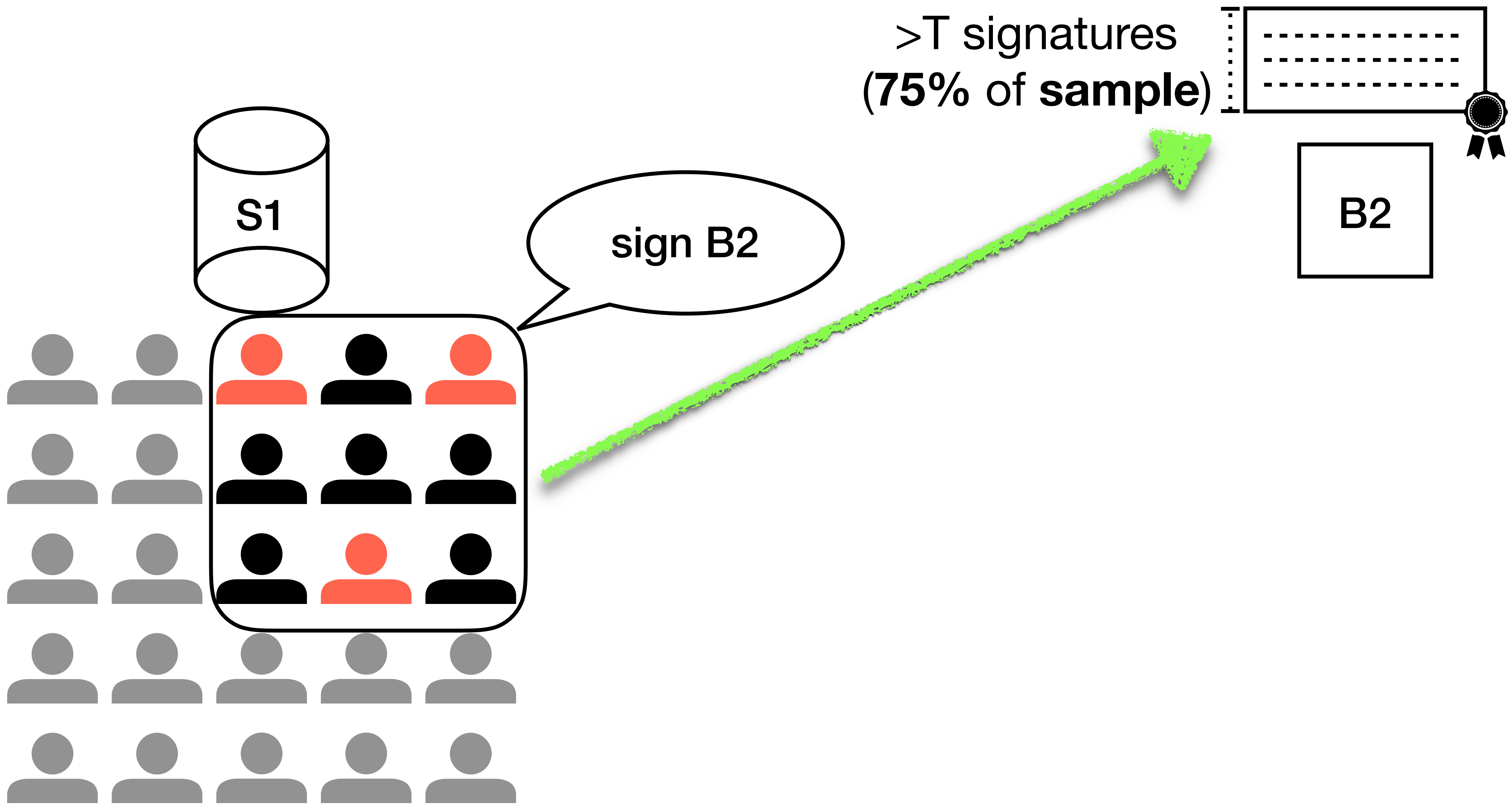
Proving a Block Correct



Proving a Block Correct



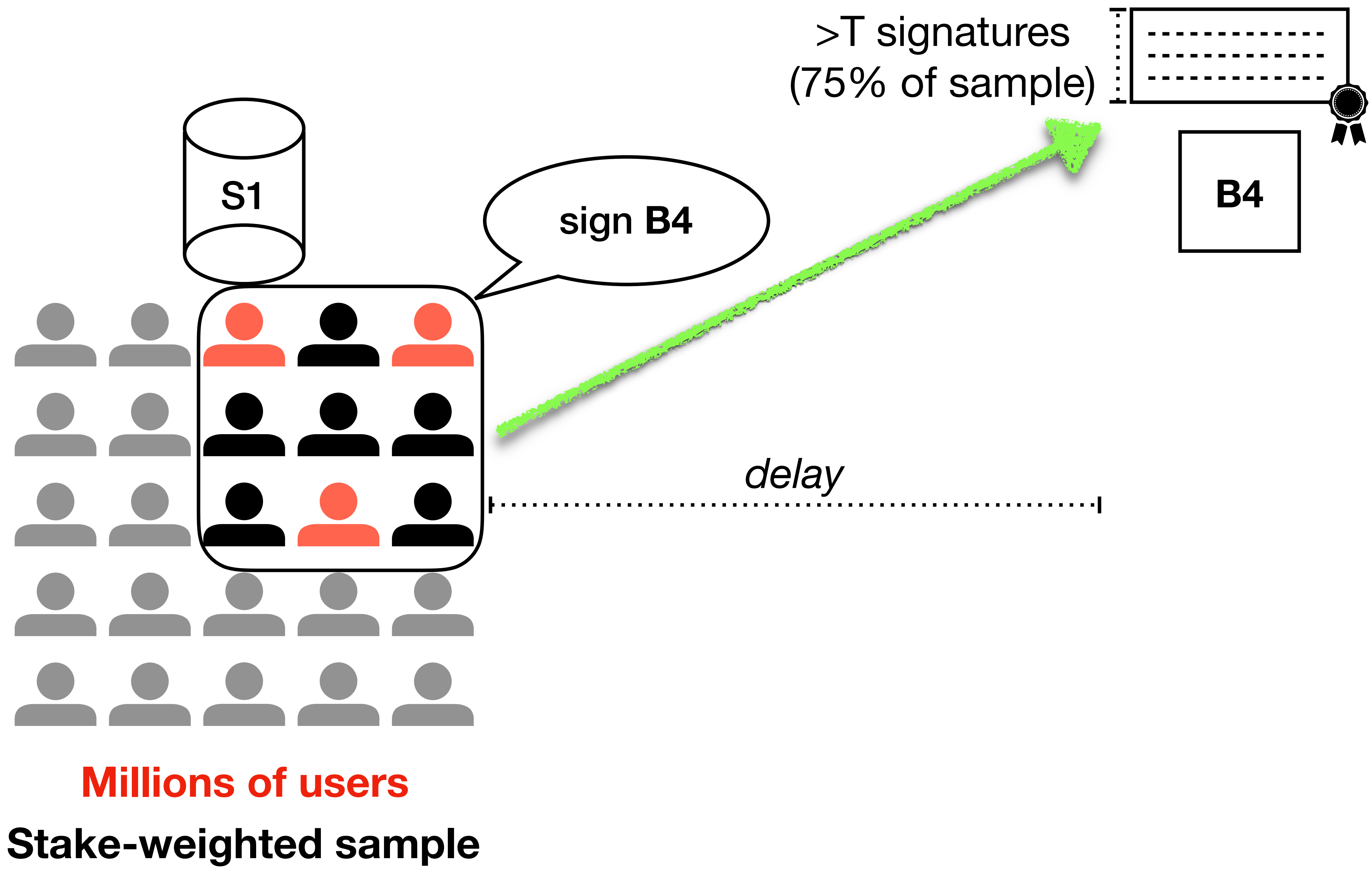
Proving a Block Correct



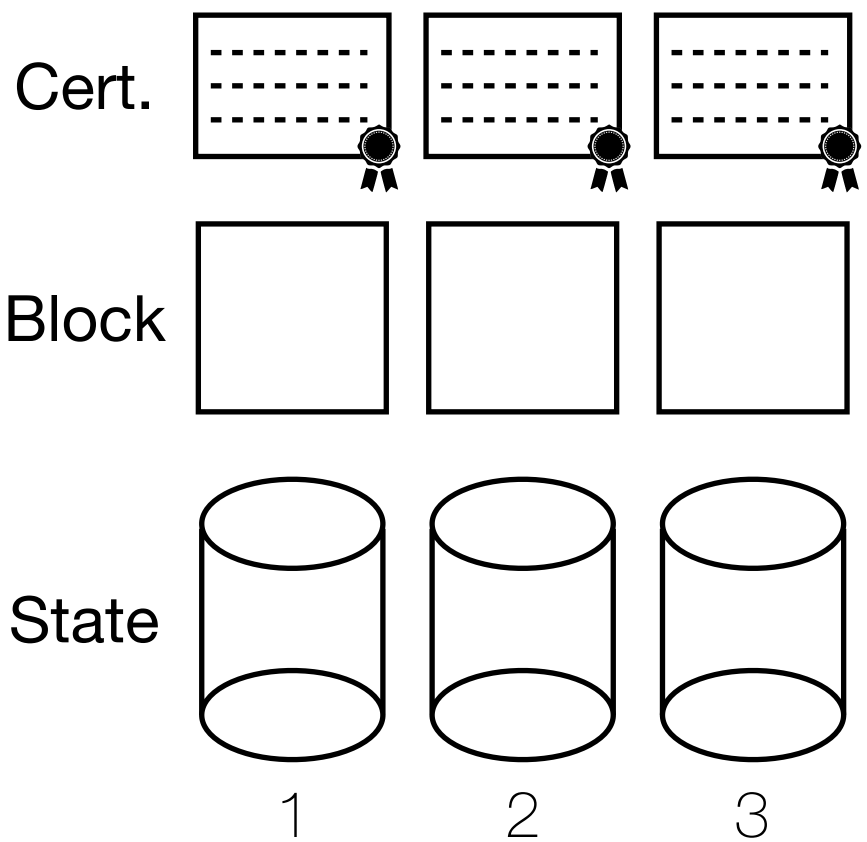
Millions of users

Stake-weighted sample

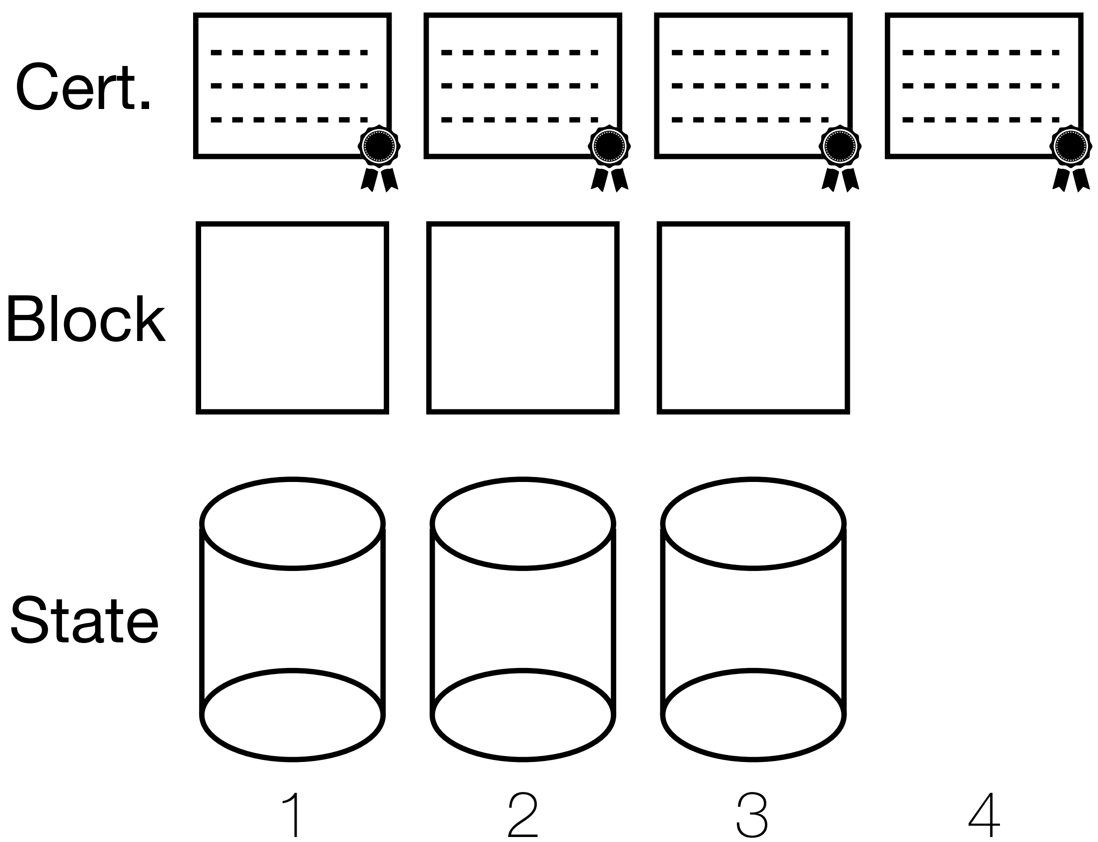
Proving a Block Correct



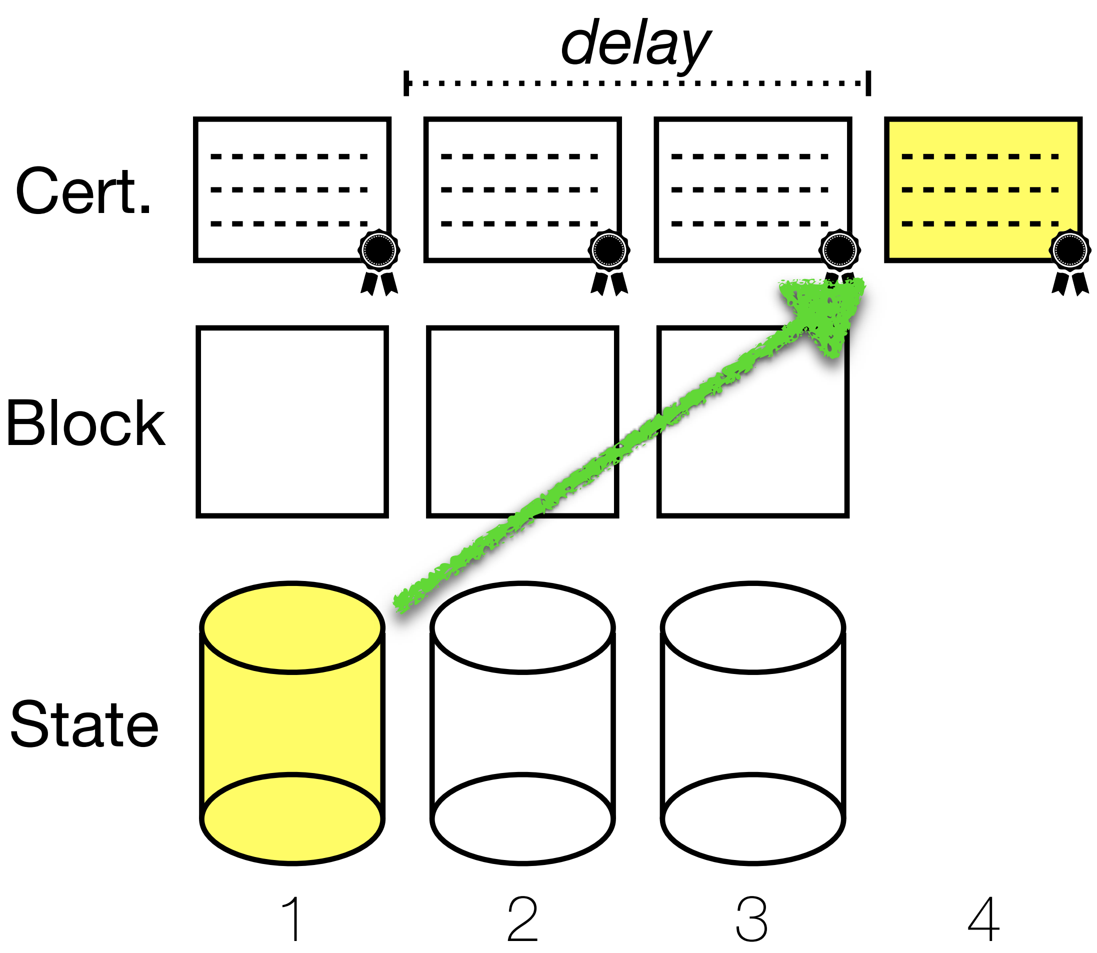
Base Bootstrapping



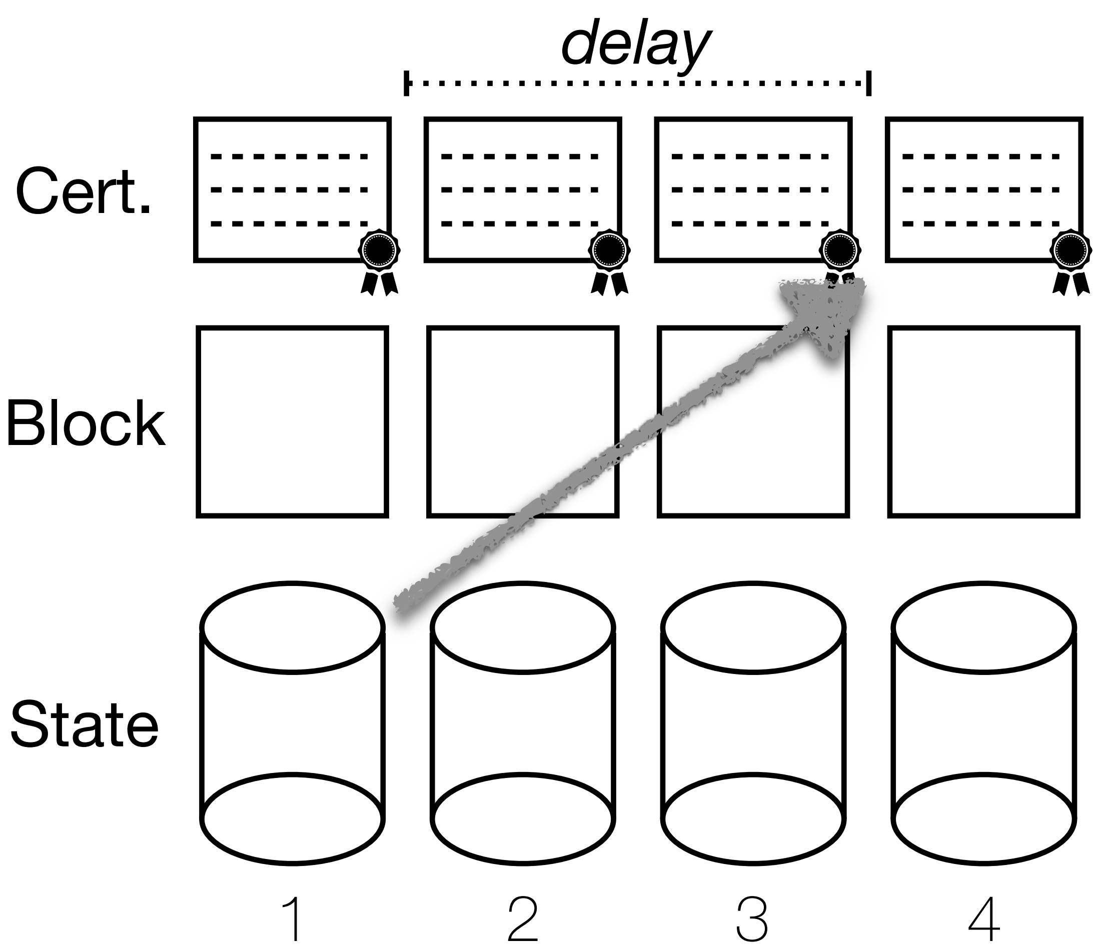
Base Bootstrapping



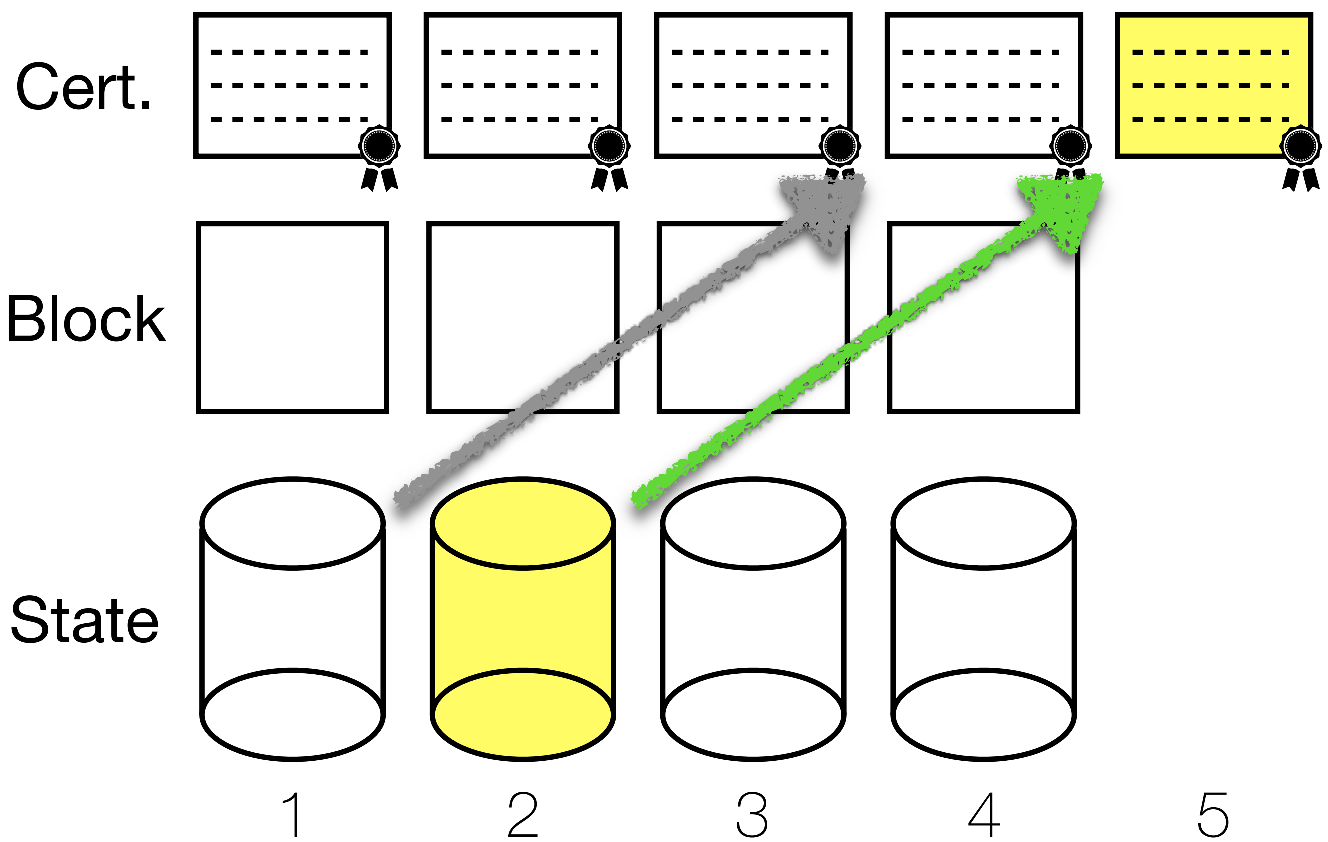
Base Bootstrapping



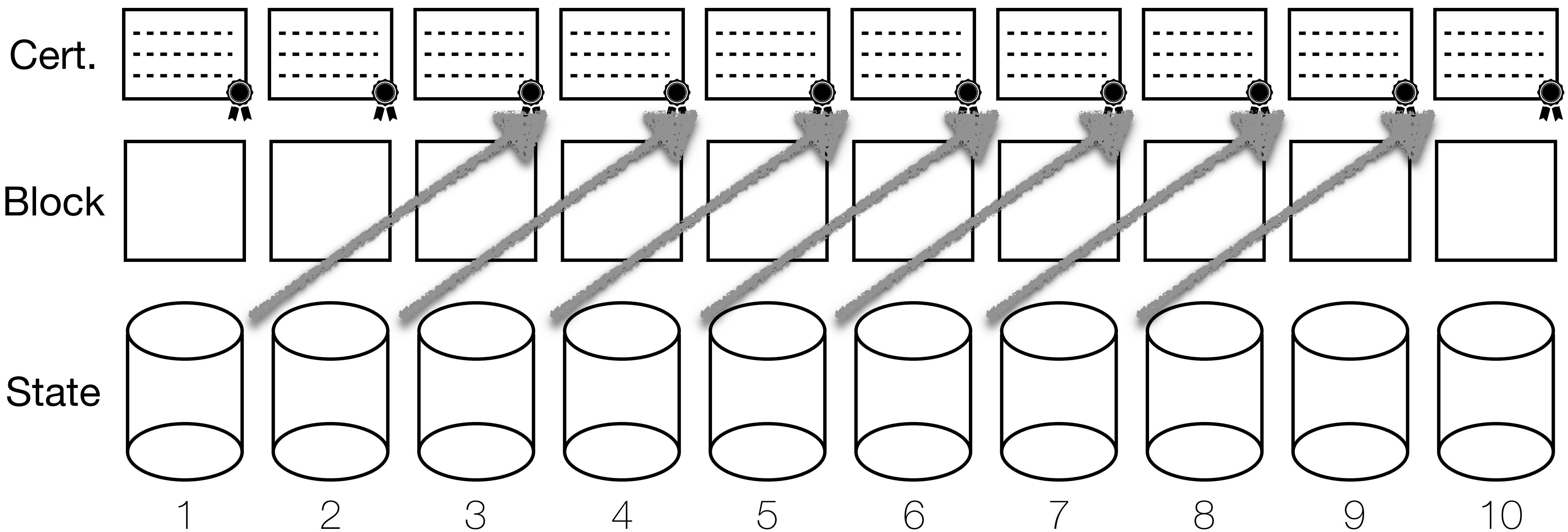
Base Bootstrapping



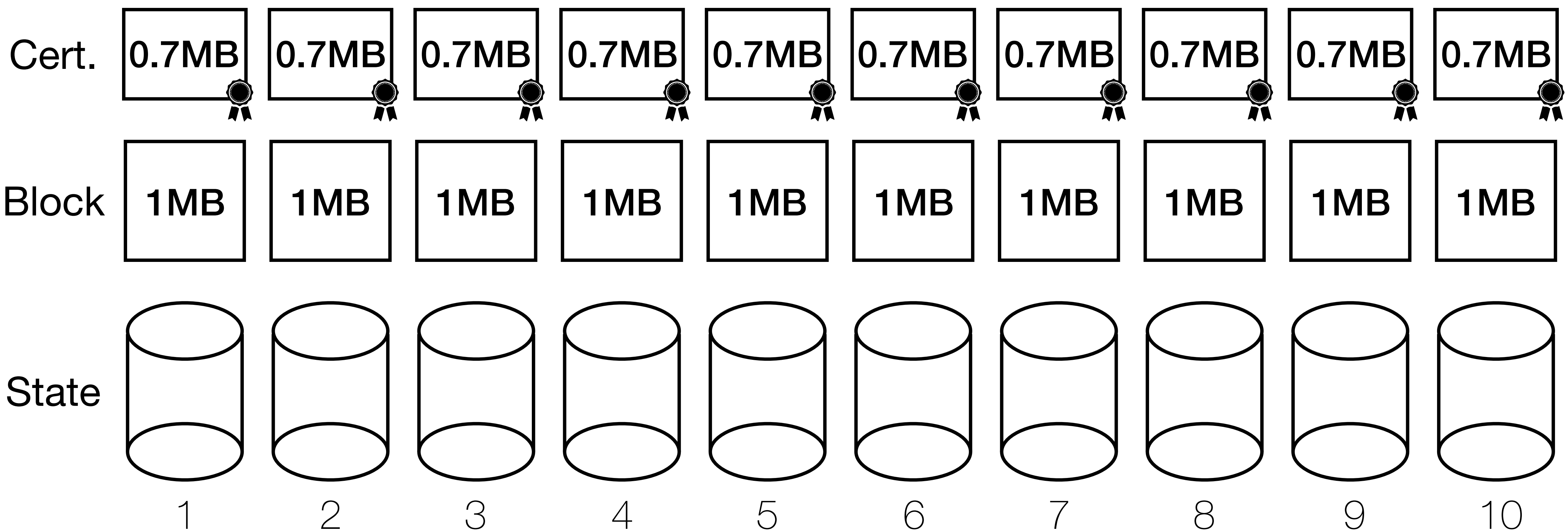
Base Bootstrapping



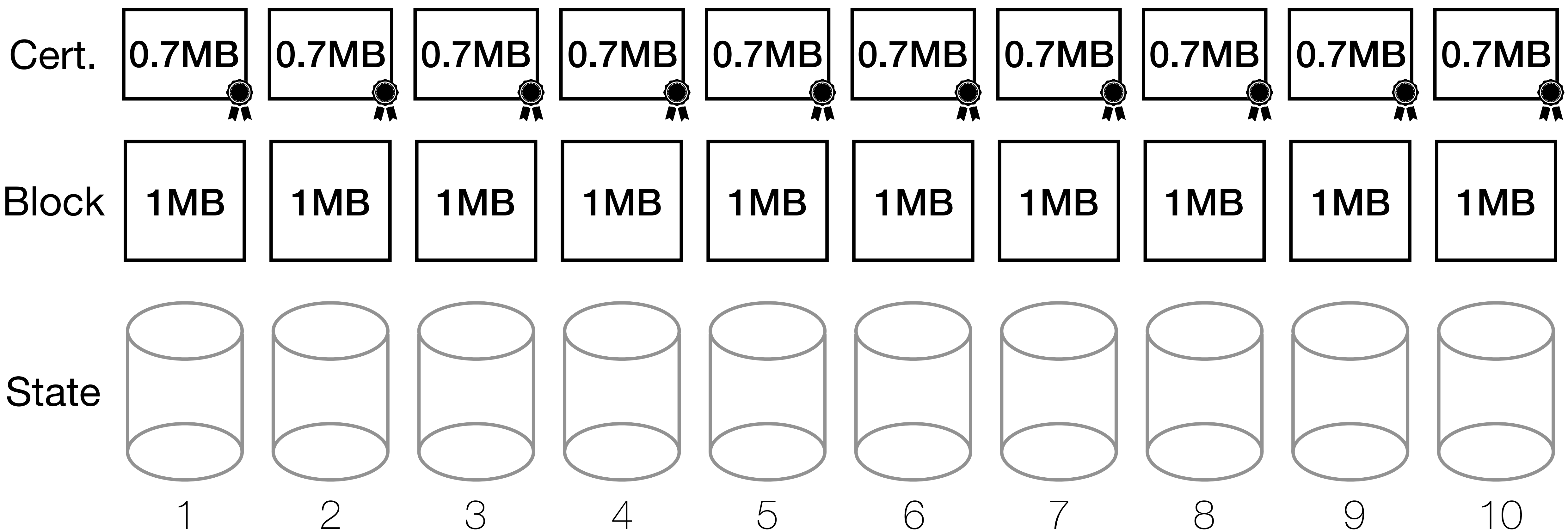
Base Bootstrapping



Base Bootstrapping



Base Bootstrapping



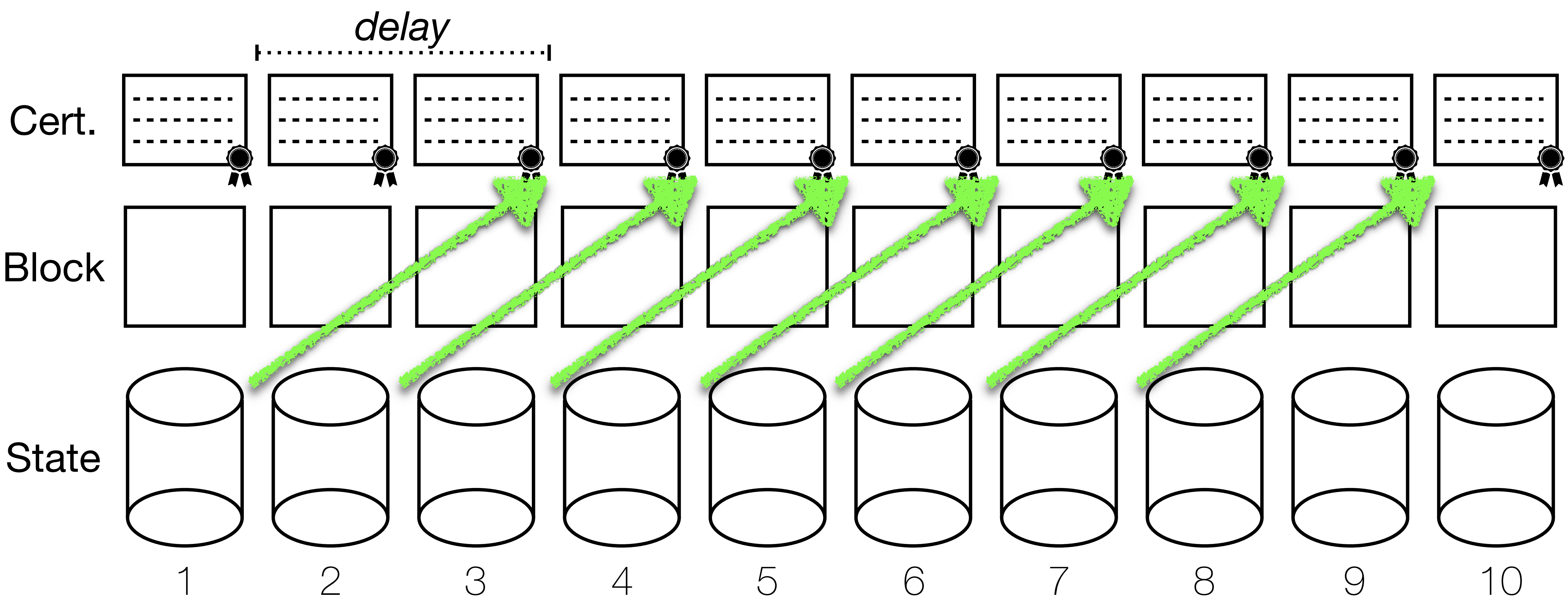
Must download every block and certificate

Vault: Compress History

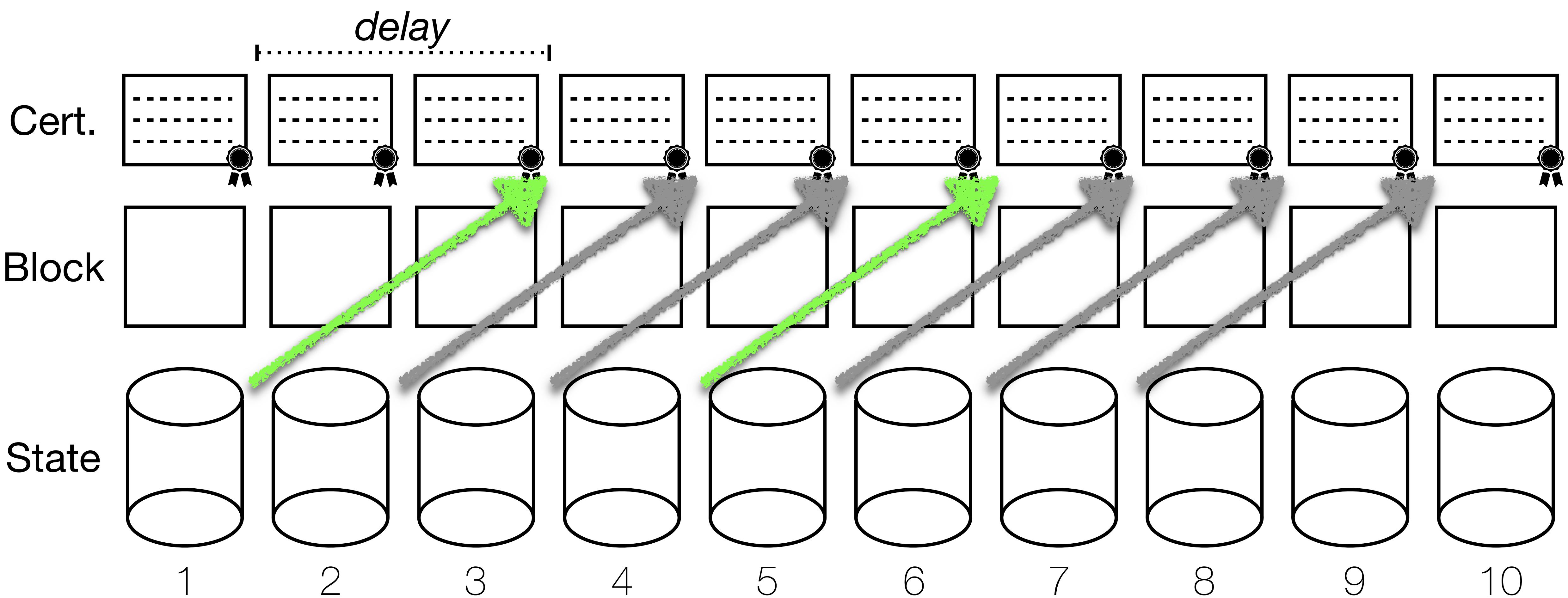
1. Vault skips blocks
2. Vault shrinks certificates

1. Skipping Blocks

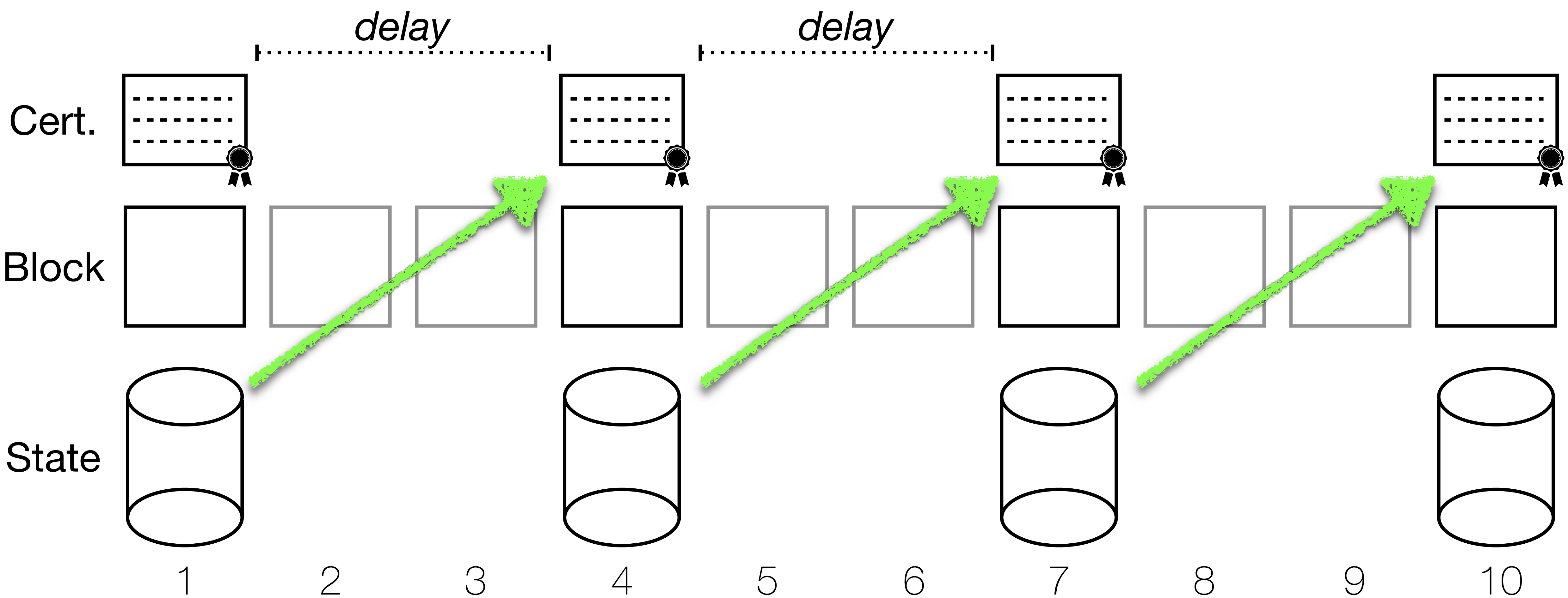
Base Bootstrapping



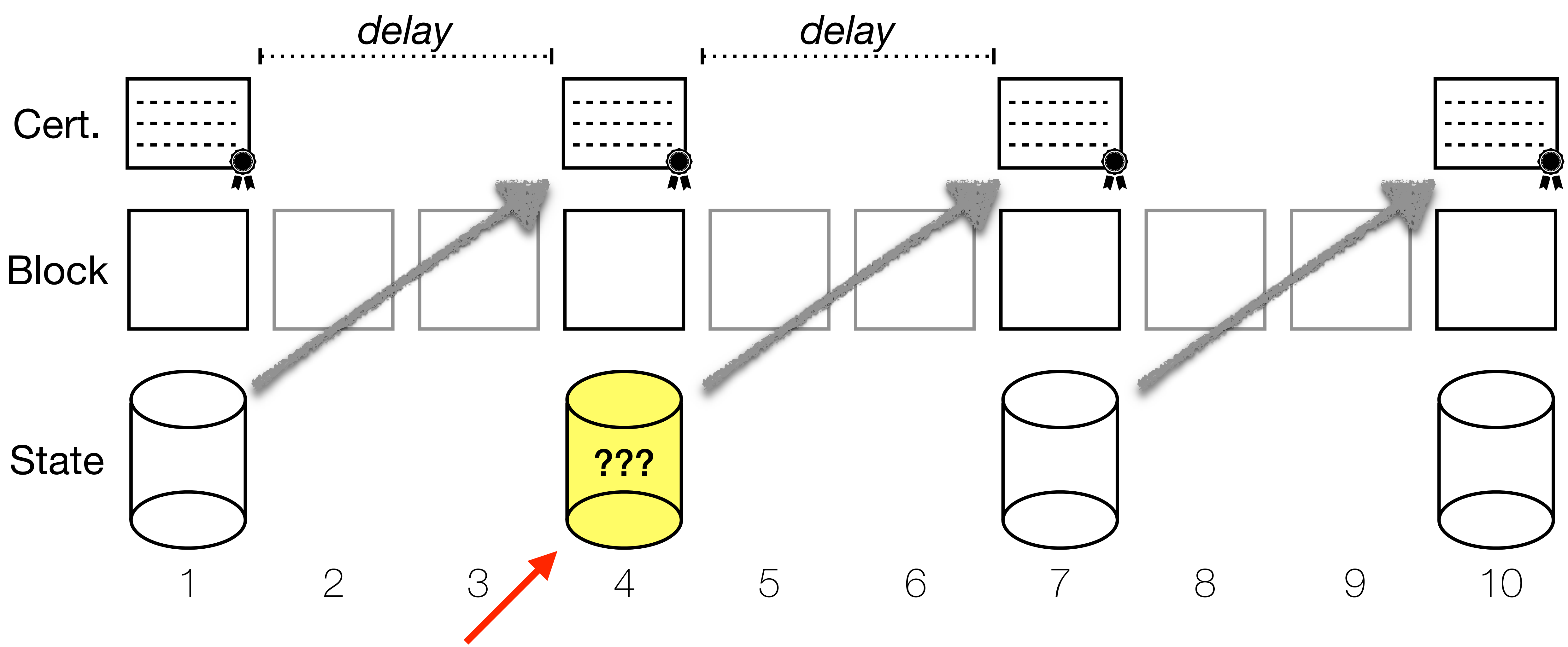
Base Bootstrapping



Skipping Blocks

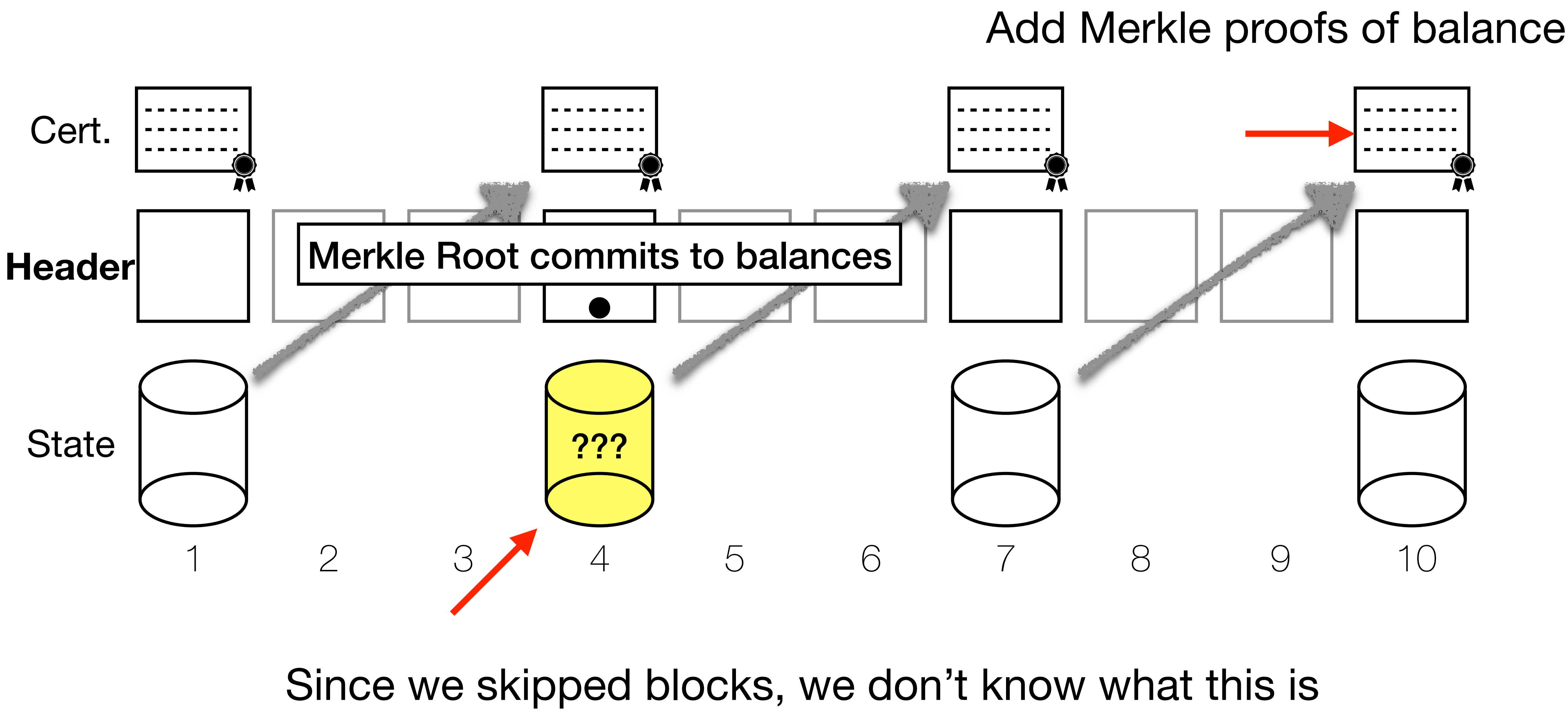


Skipping Blocks

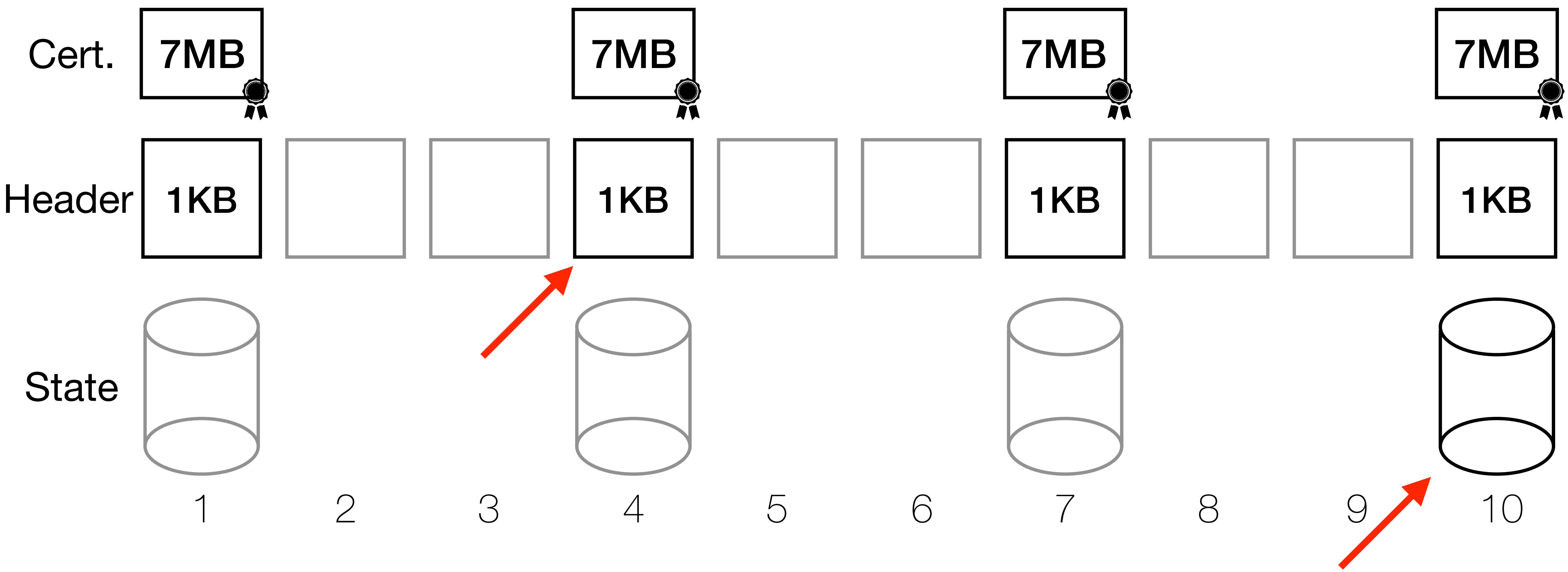


Since we skipped blocks, we don’t know what this is

Skipping Blocks



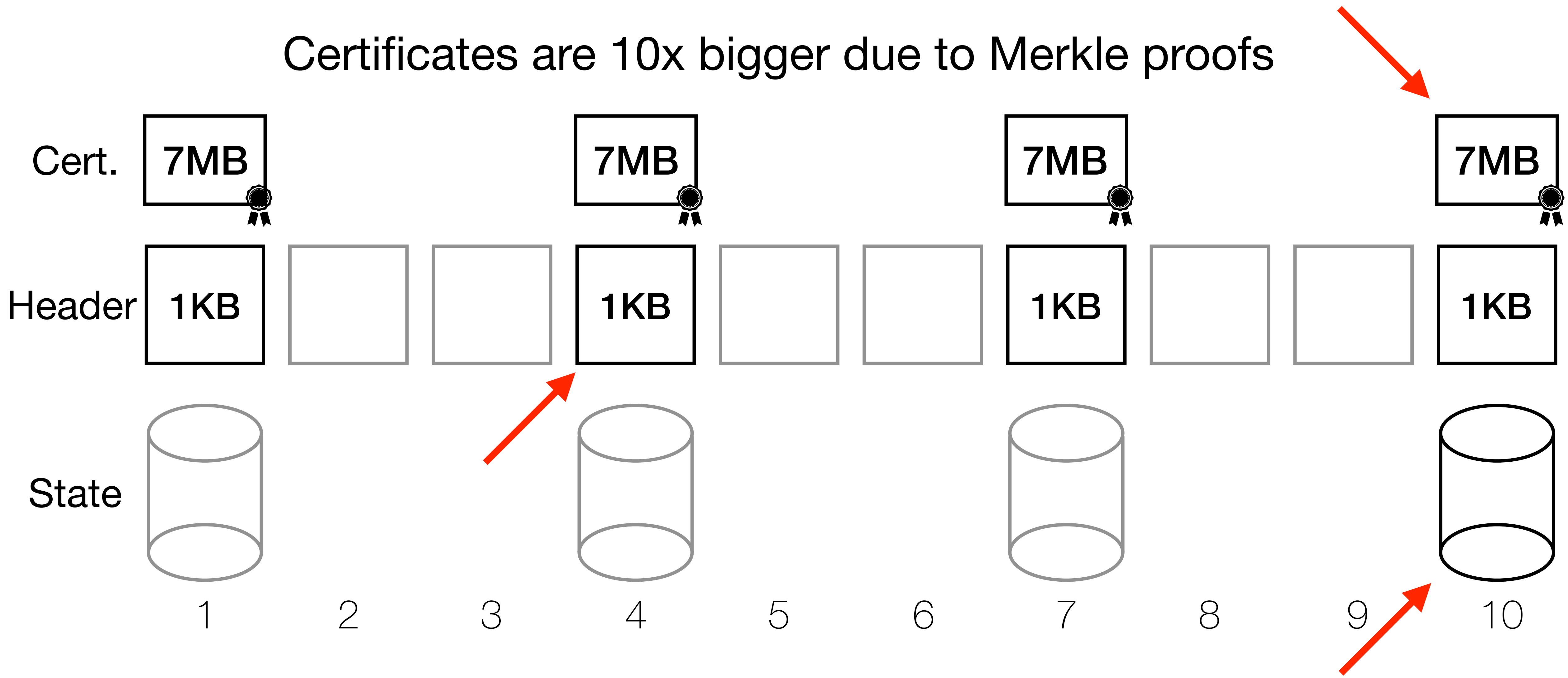
Skipping Blocks



Only need headers, certificates, and final state

Skipping Blocks

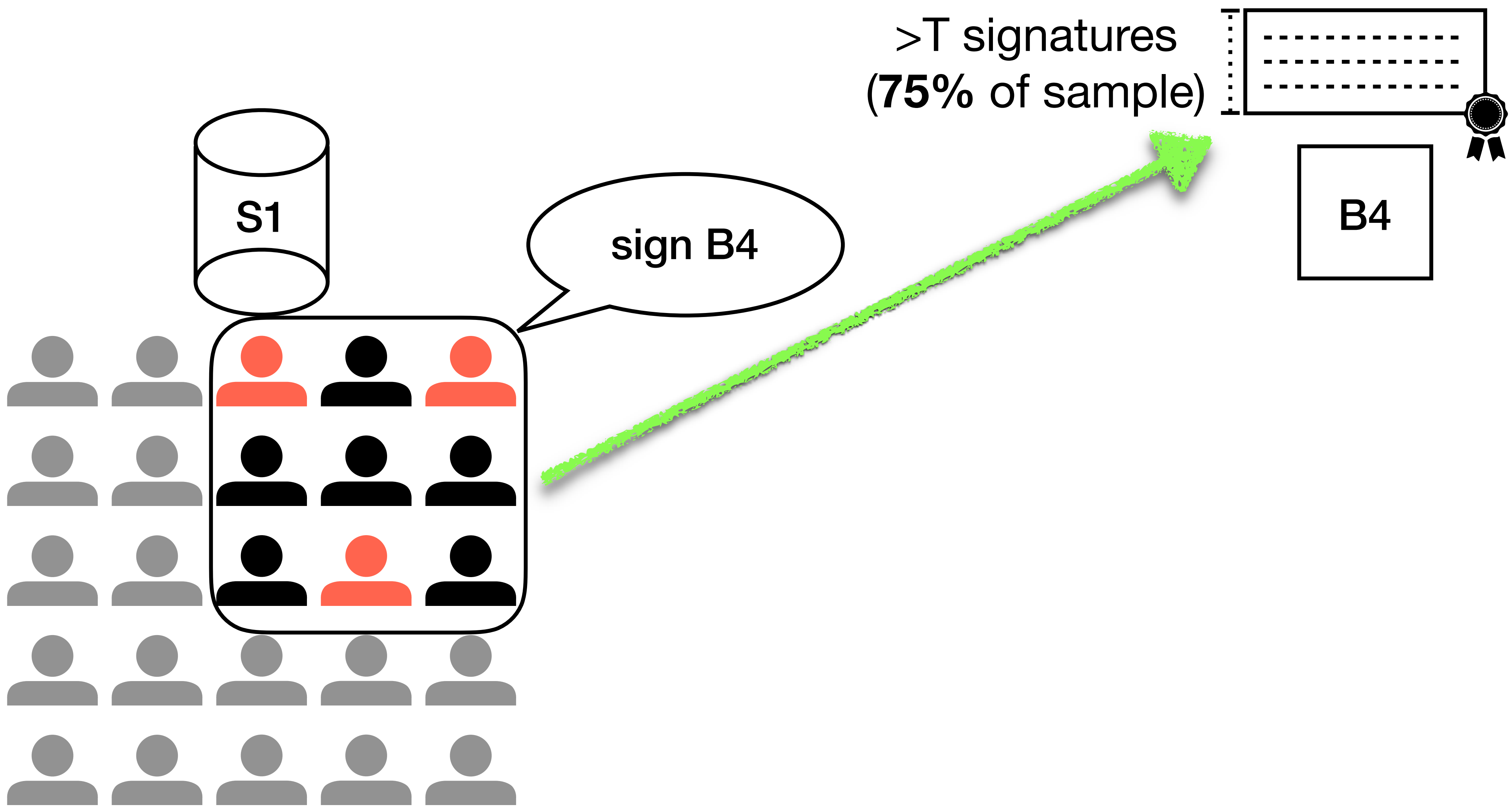
Certificates are 10x bigger due to Merkle proofs



Only need headers, certificates, and final state

2. Shrinking Certificates

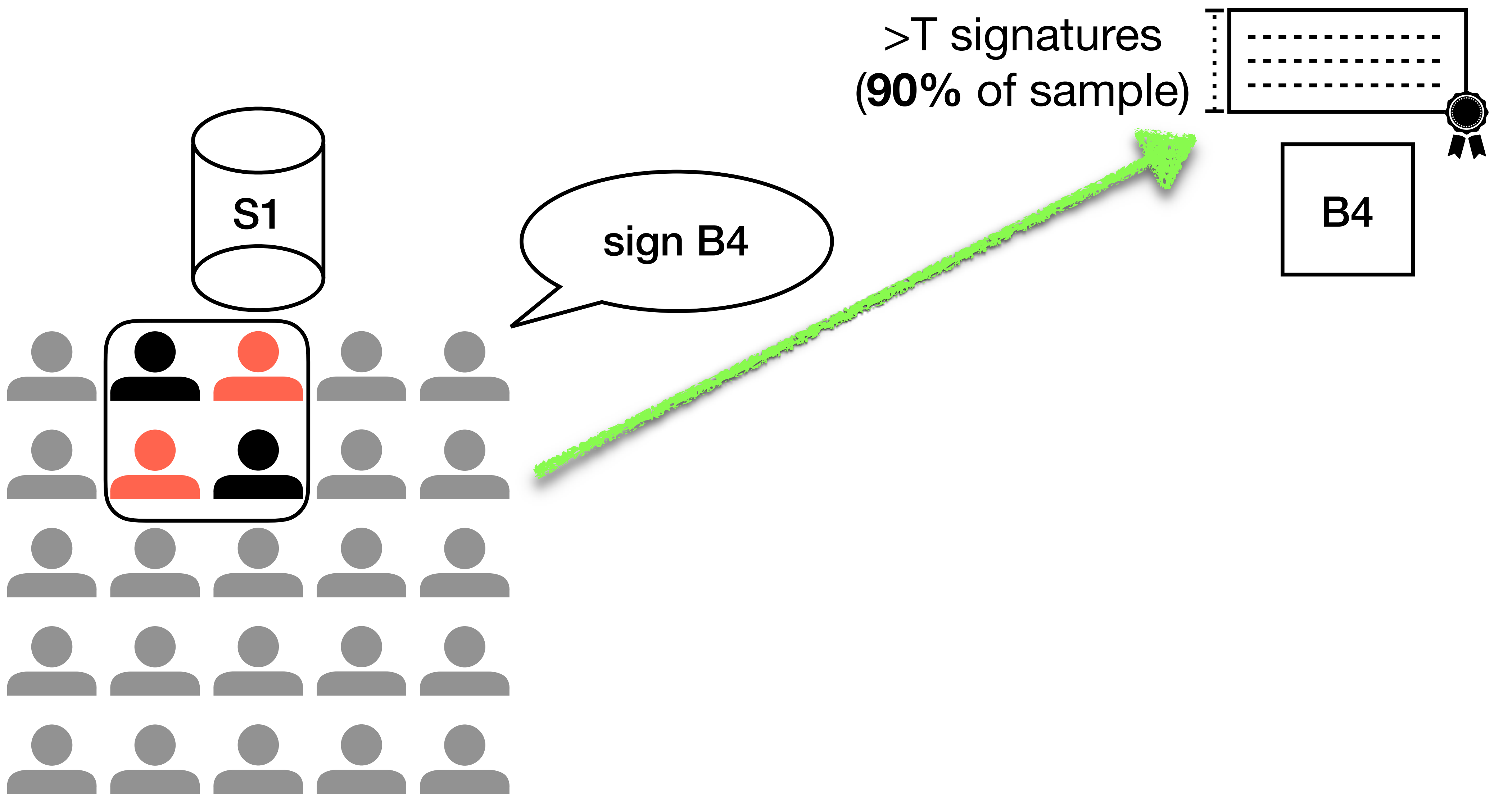
Proving a Block Correct



Millions of users

Stake-weighted sample

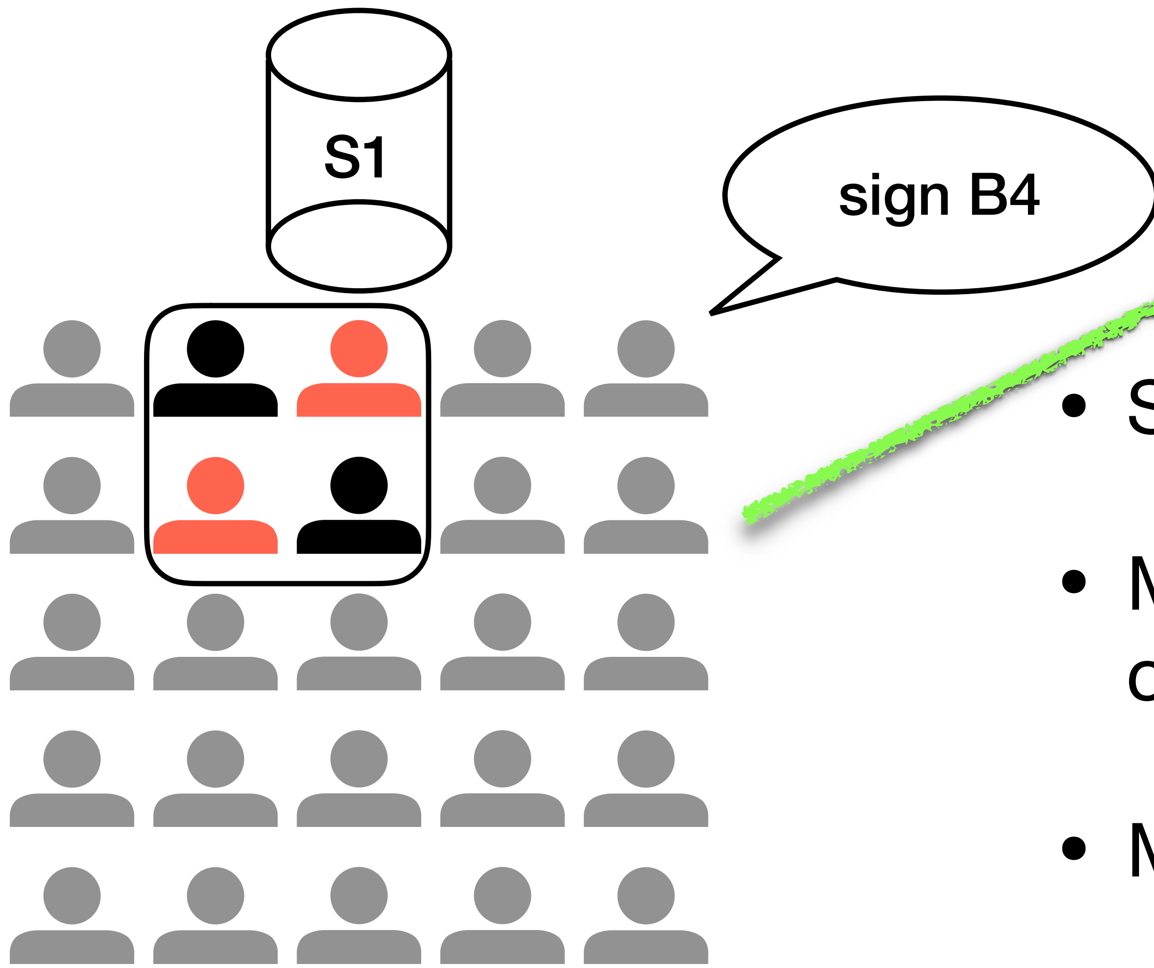
Smaller Proofs



Millions of users

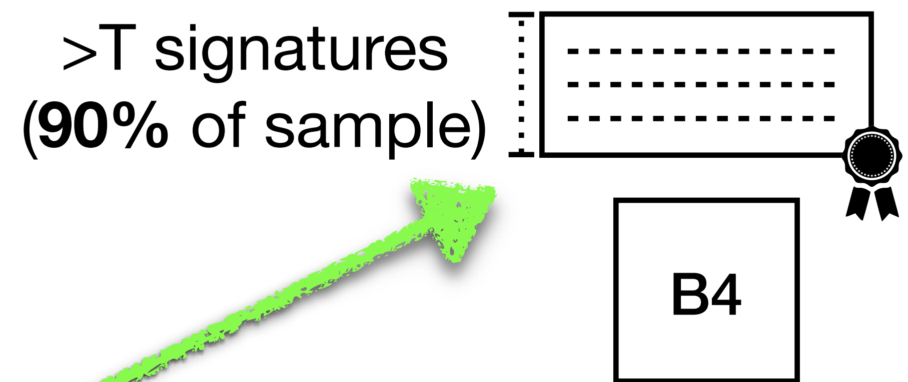
Stake-weighted sample

Smaller Proofs



Millions of users

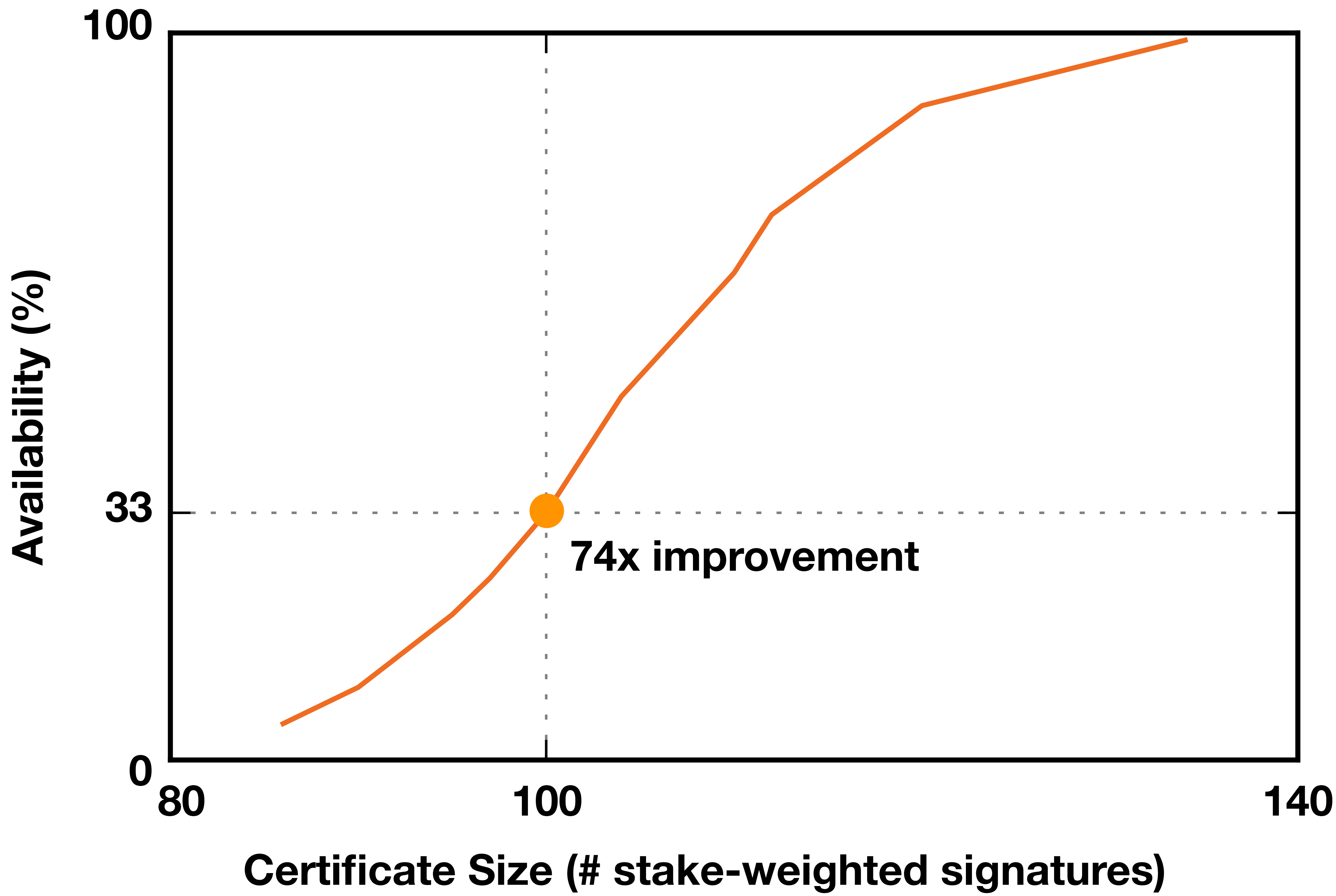
Stake-weighted sample



- Smaller sample: higher variance
- Must raise certificate threshold to compensate (e.g., 75% to 90%)
- May fail to meet threshold!

During bootstrapping, trade off availability for size

Trading Away Availability



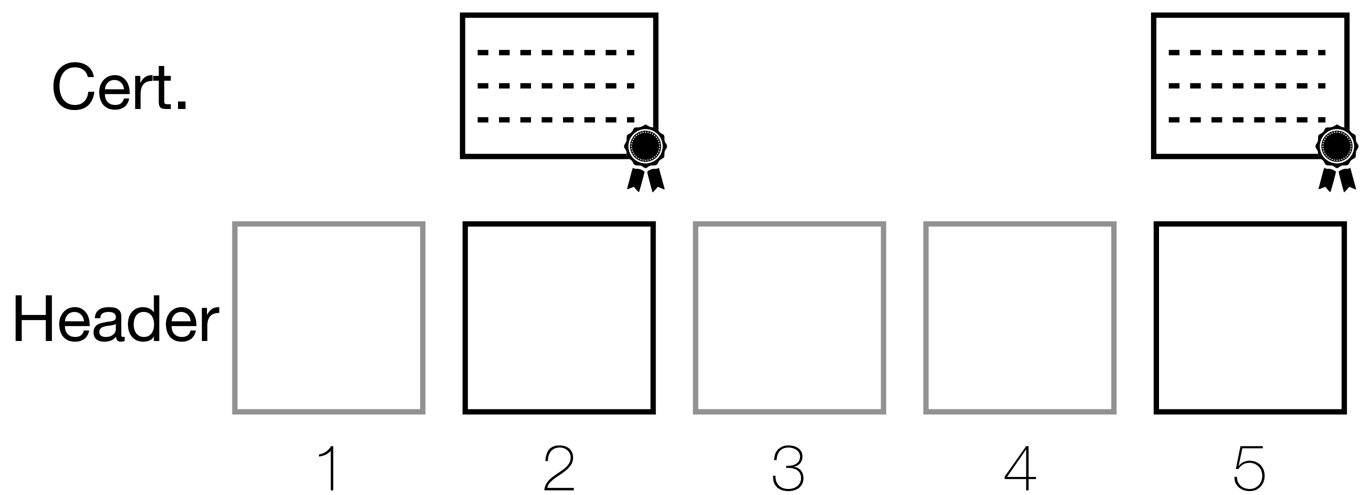
Trading Away Availability

Shrink certificates by 74x (tunable)

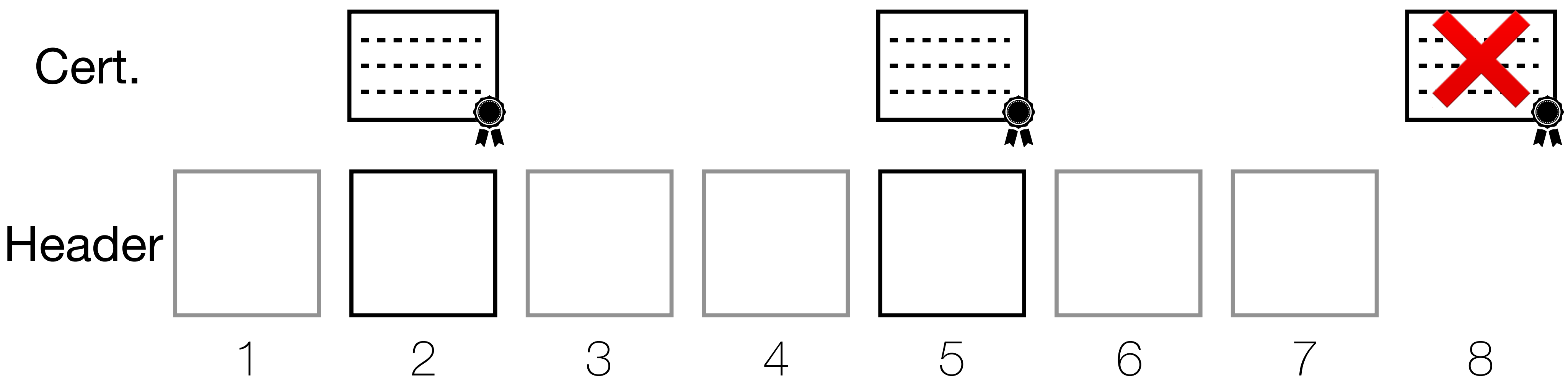
On failure, use block header hashes to bypass failure

(0.1MB to skip < *delay* blocks)

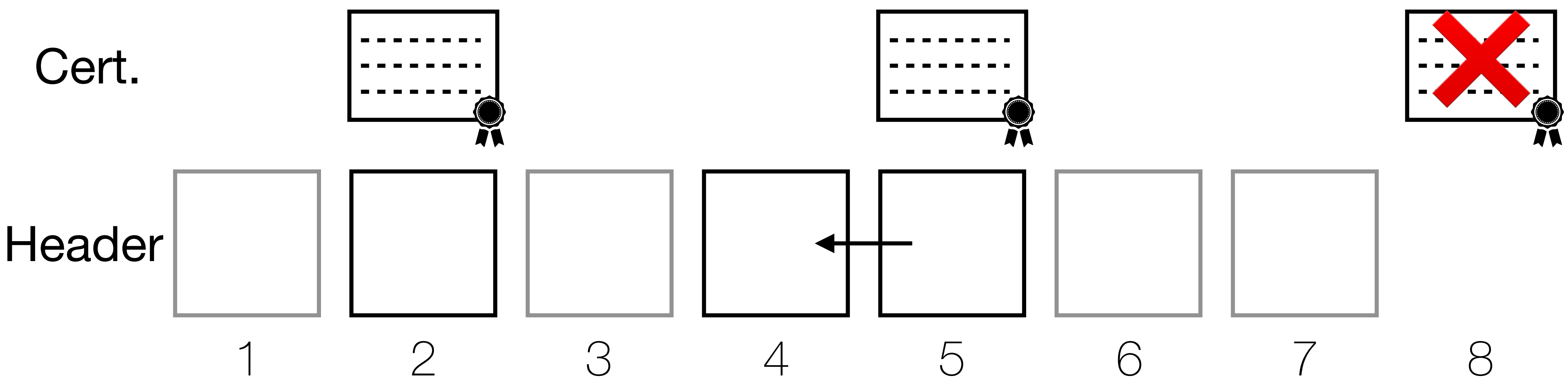
Block Hash Vaulting



Block Hash Vaulting

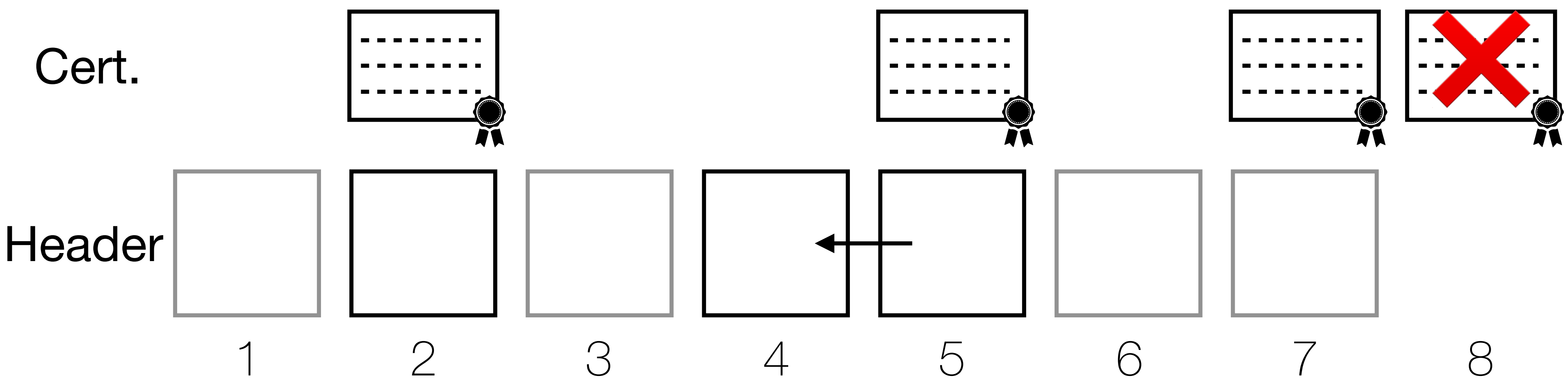


Block Hash Vaulting



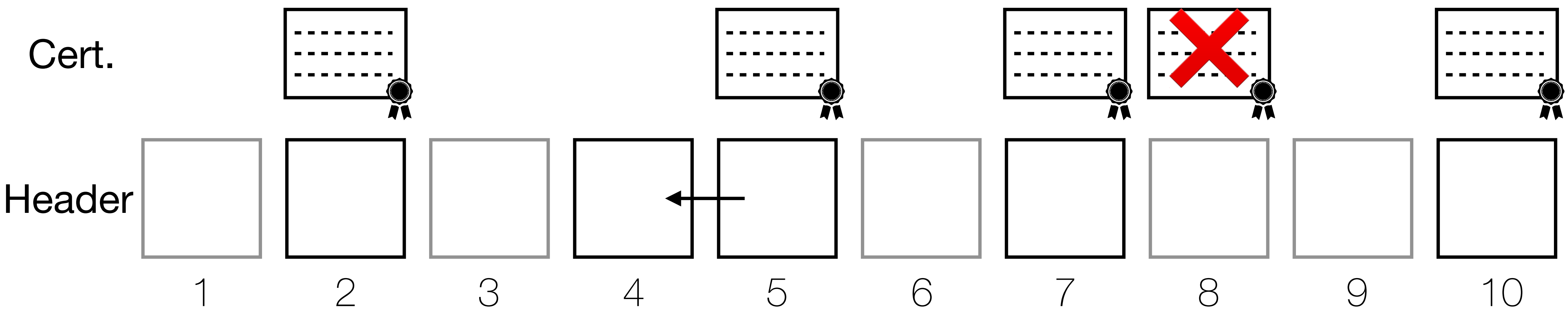
Block headers contains hash of previous block headers

Block Hash Vaulting



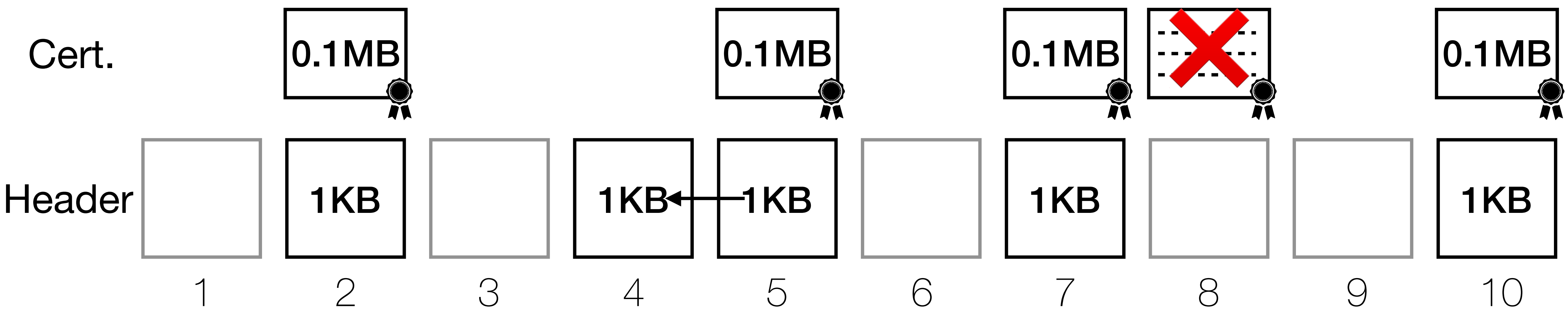
Block headers contains hash of previous block headers

Block Hash Vaulting



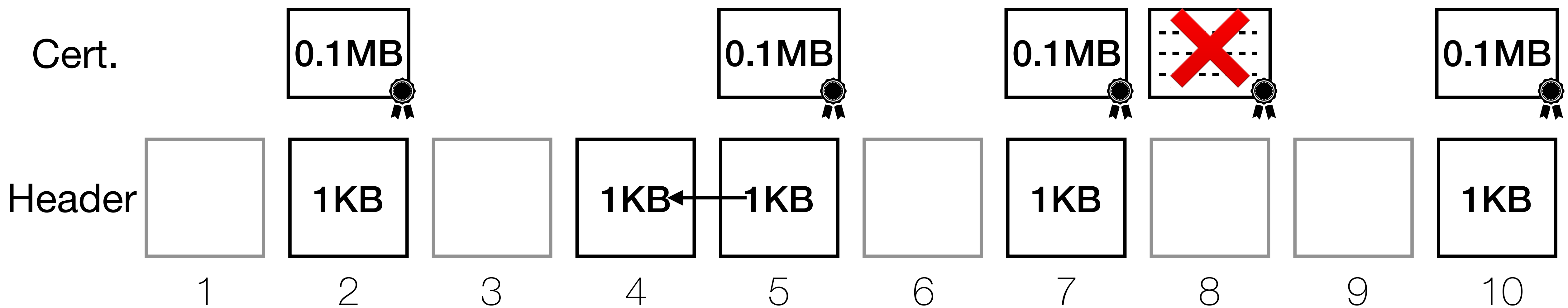
Block headers contains hash of previous block headers

Block Hash Vaulting



Block headers contains hash of previous block headers

Block Hash Vaulting



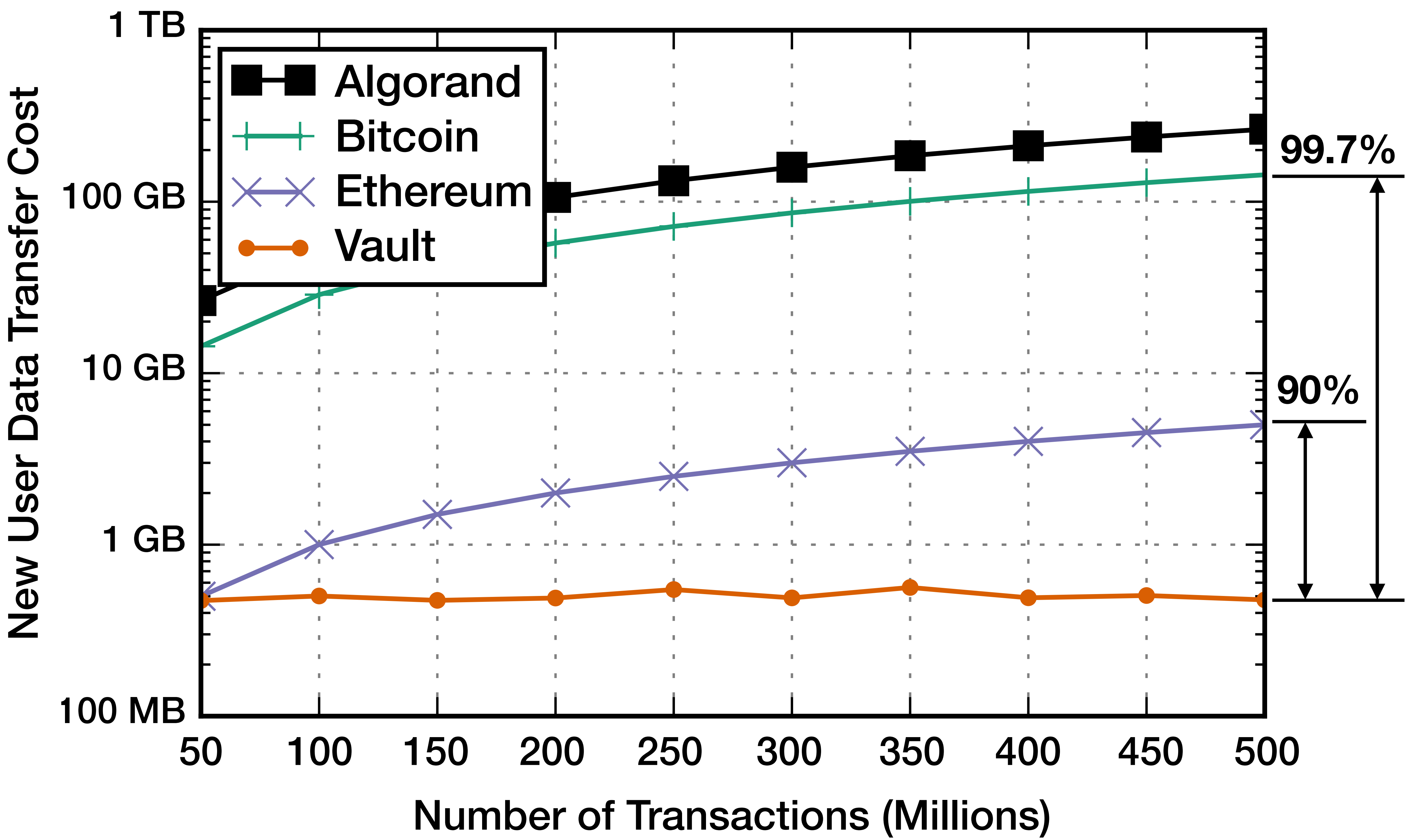
Block headers contains hash of previous block headers

Fall back to large certificates (7MB to skip *delay* blocks)

Evaluation

- Simulate trace of 500 million simple transactions
- Prototype data structures in Bitcoin, Ethereum, Algorand, and Vault
  - All 3 Vault techniques: transaction expiration, adaptive sharding, and succinct certificates
  - *delay* = 1000, #*shards* = 1000
- Measure bootstrapping data transfer cost

Data Transfer Cost



Related Work

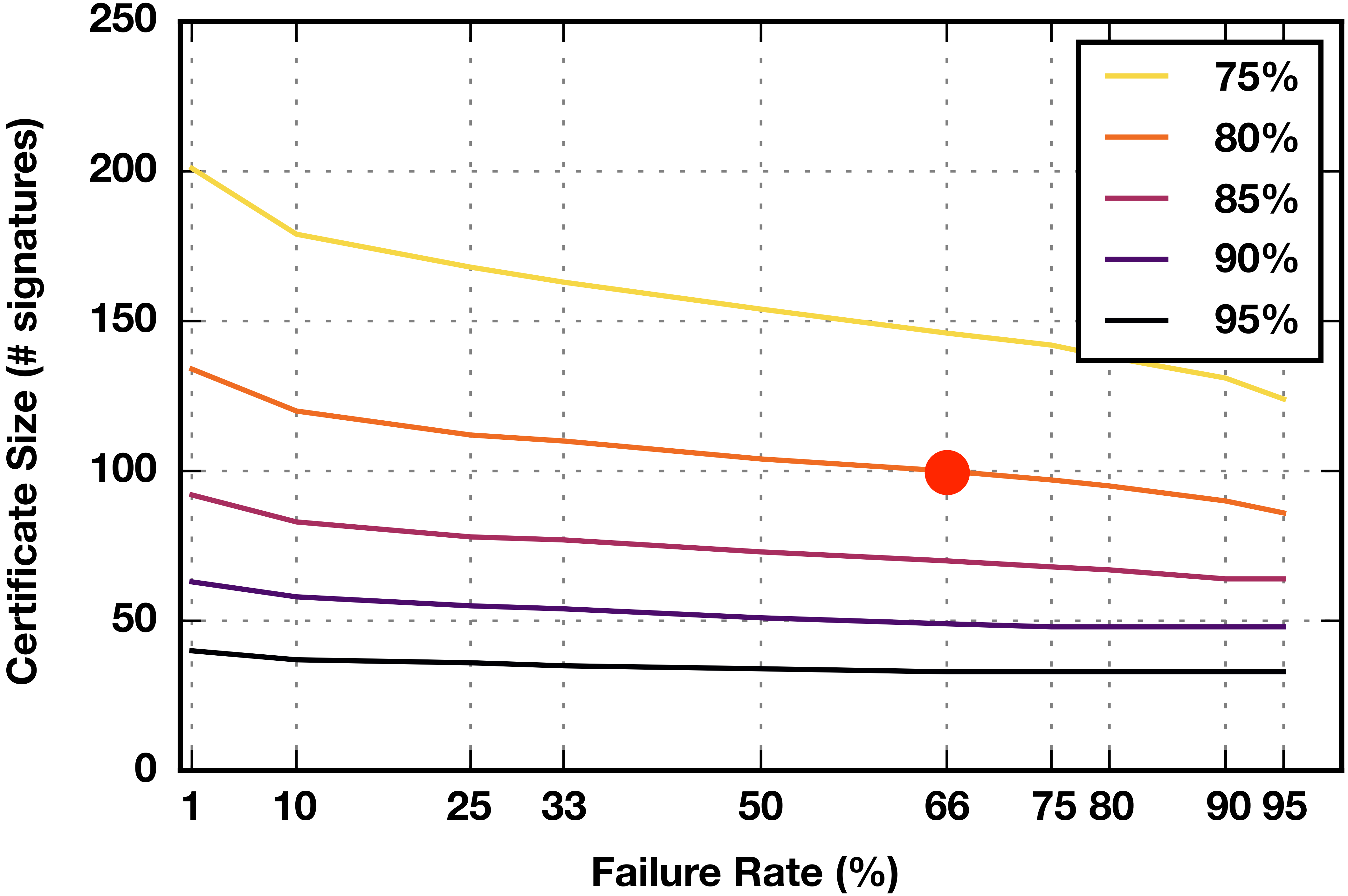
- Minimize State:
  - Lightning Network (Poon and Dryja)
  - Edrax (Chepurnoy *et al*.)
  - OmniLedger (Kokoris-Kogias *et al*.)
- Minimize Proof of State:
  - MimbleWimble (Poelstra)
  - Chainiac (Nikitin *et al*.)

Conclusion

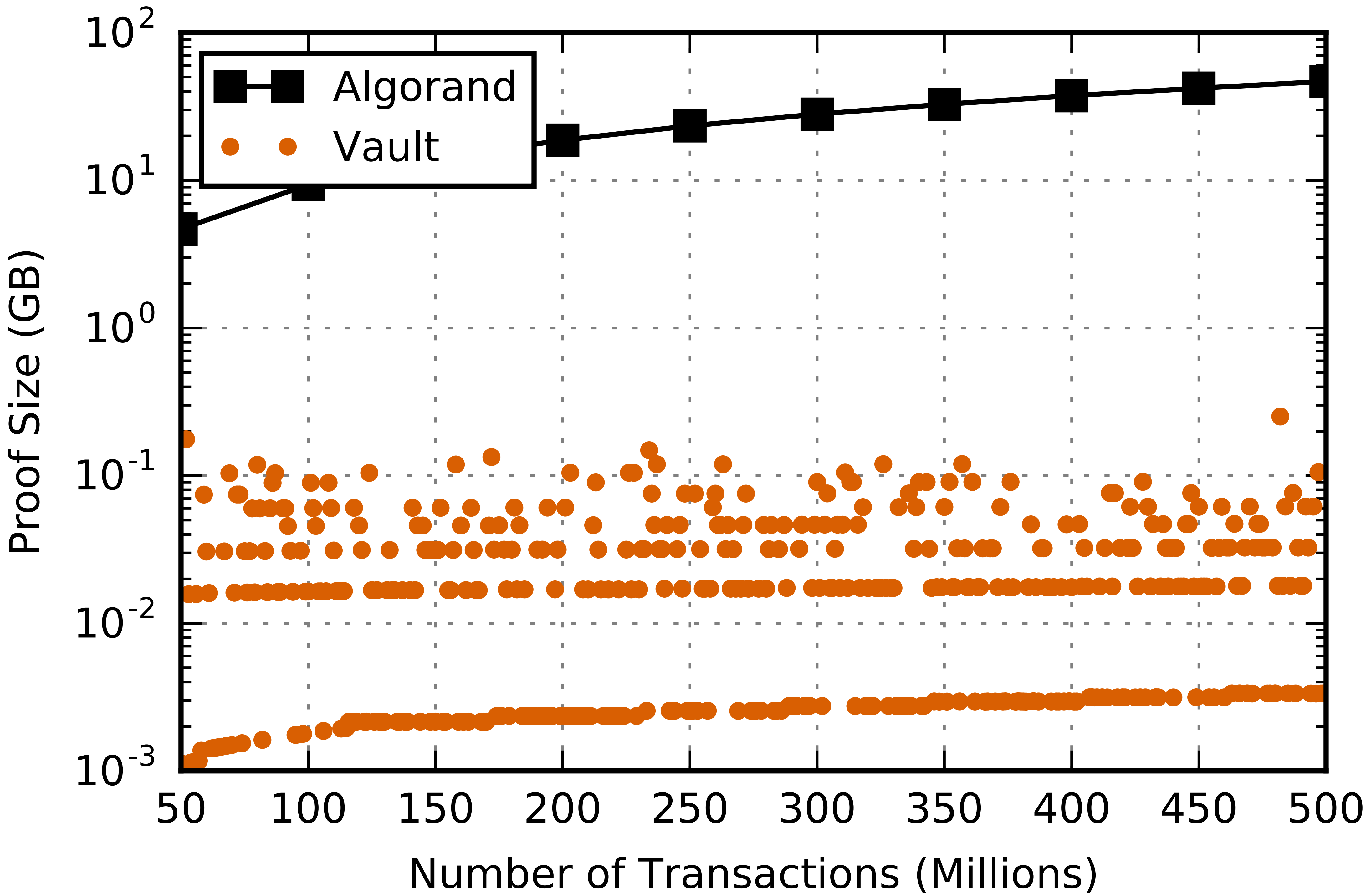
- Bootstrapping costs prevent scaling
- Vault techniques address these costs
  - Succinct certificates securely compress history, trading off availability
- Reduction in bootstrapping costs to 477MB:
  lower by 90.5% (Ethereum) or 99.7% (Bitcoin)

Backup slides

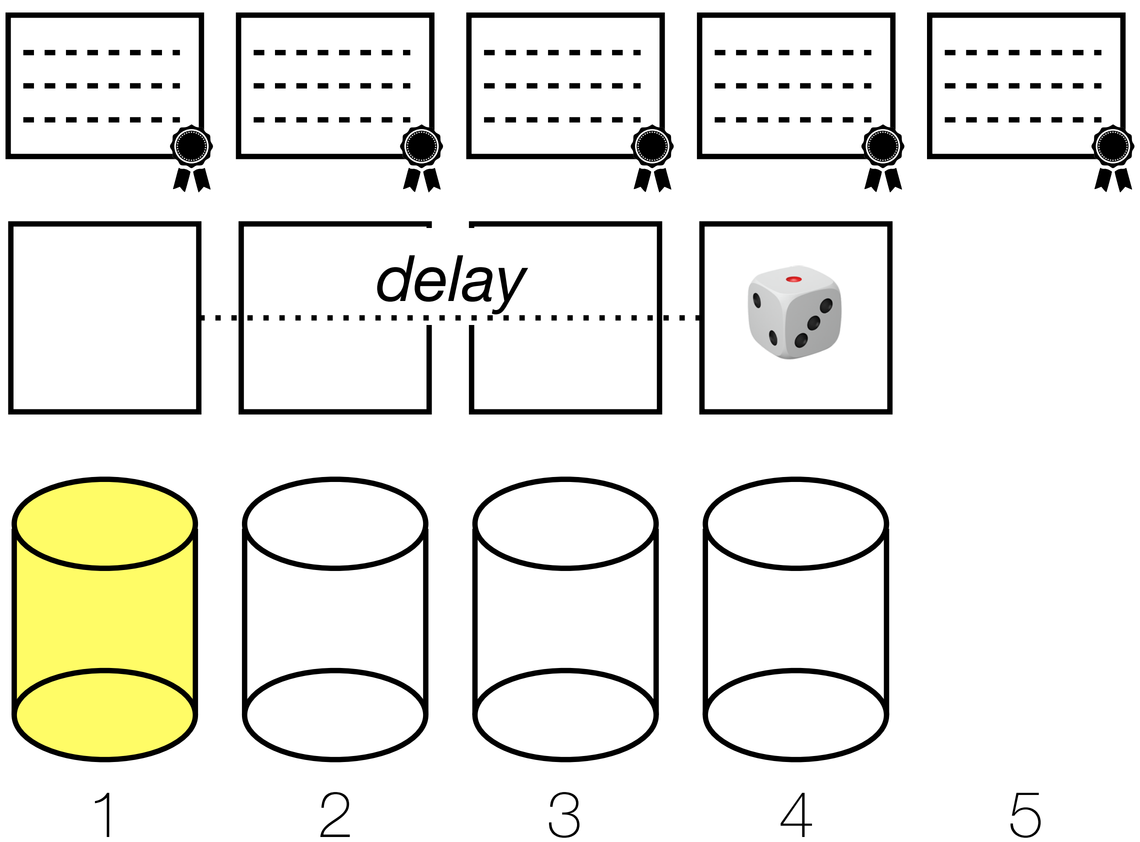
Different settings of *f*



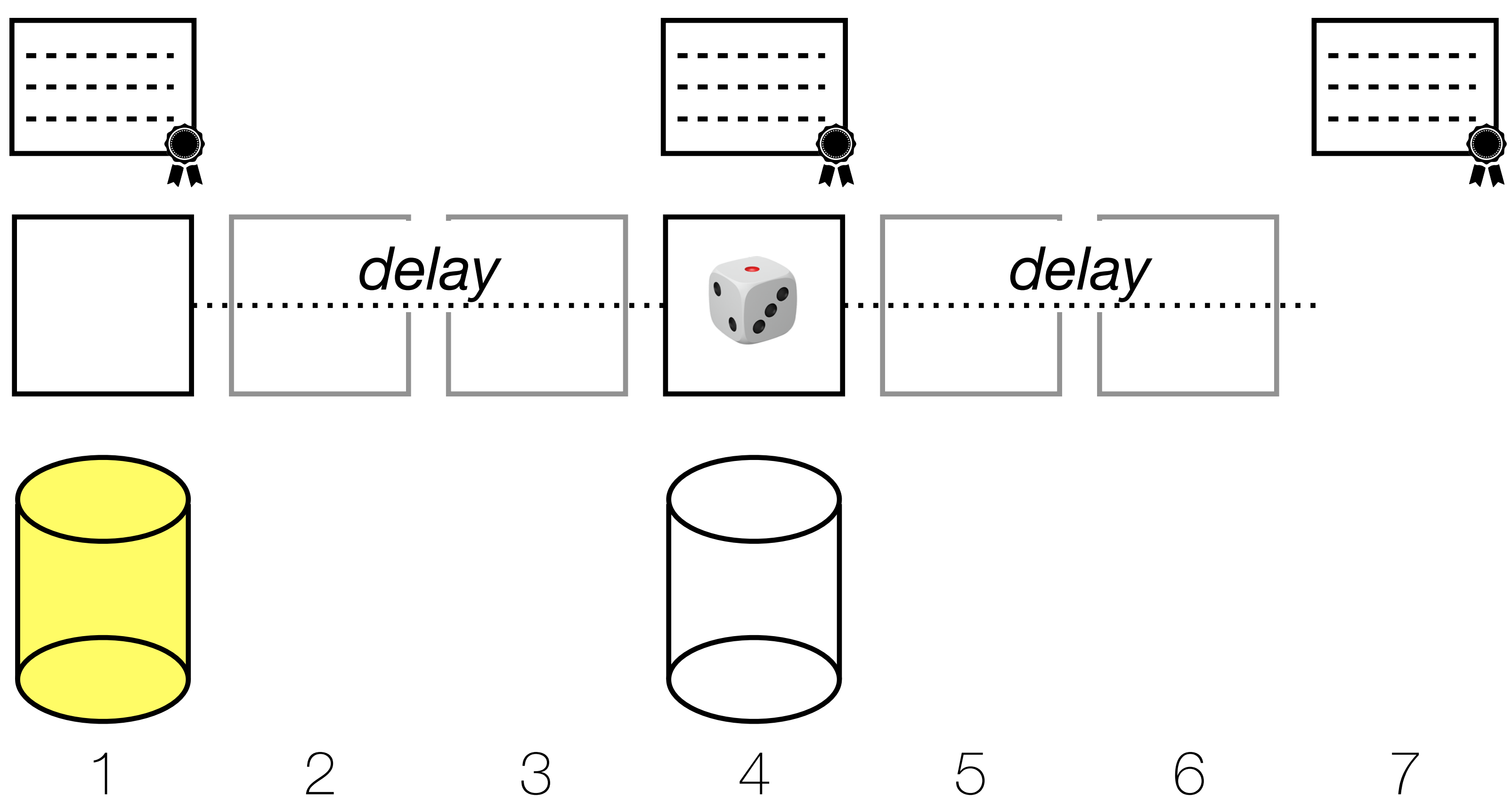
Bootstrapping Proof Size



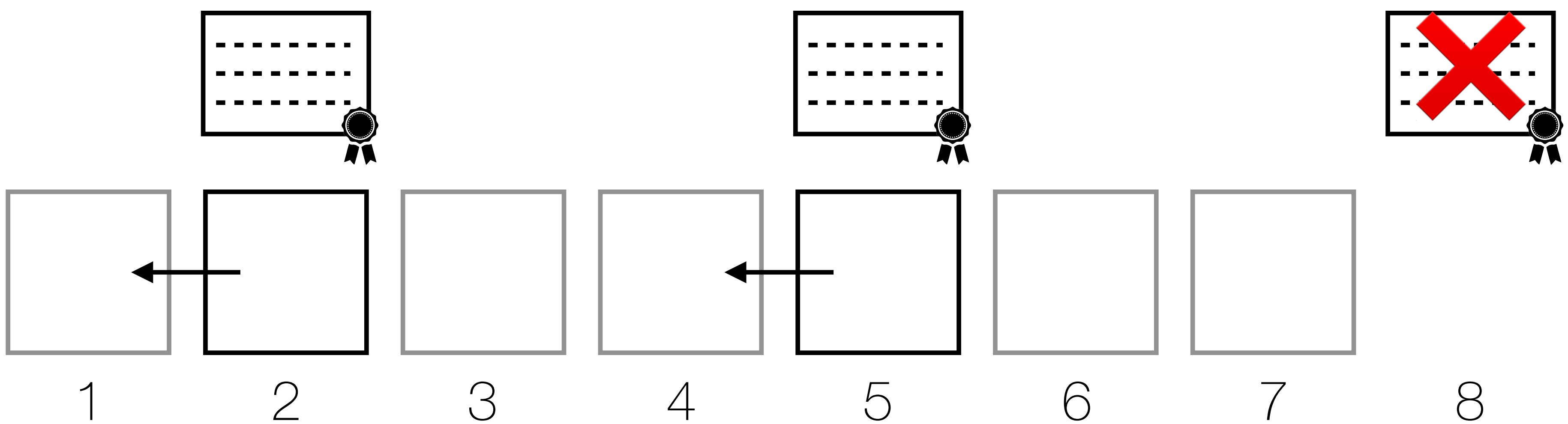
Bootstrapping with Randomization



Skipping Blocks with Randomization

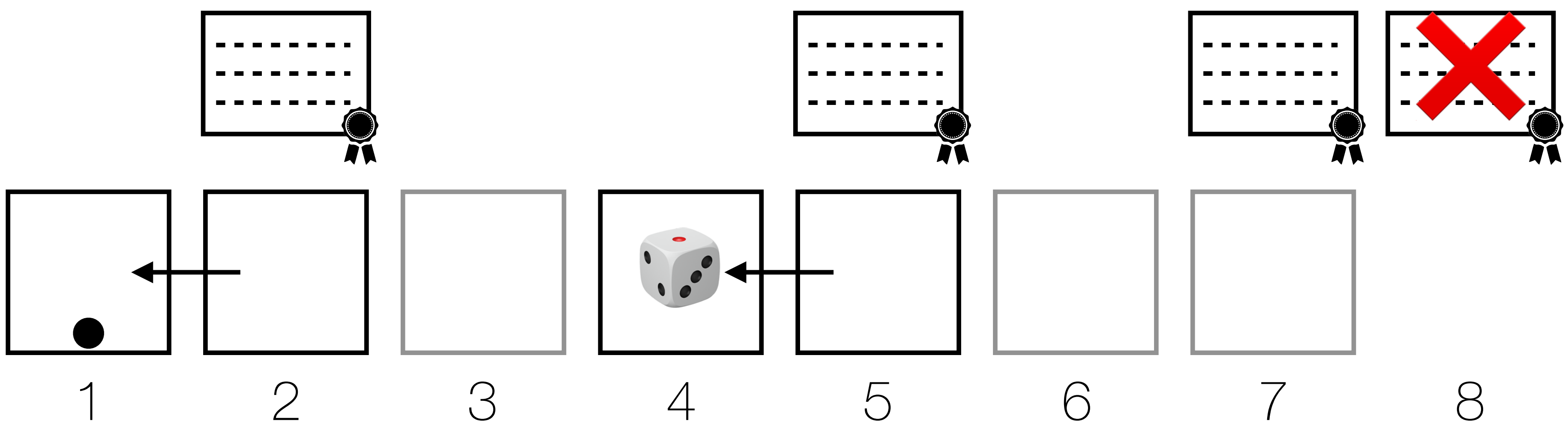


Vaulting with Randomization



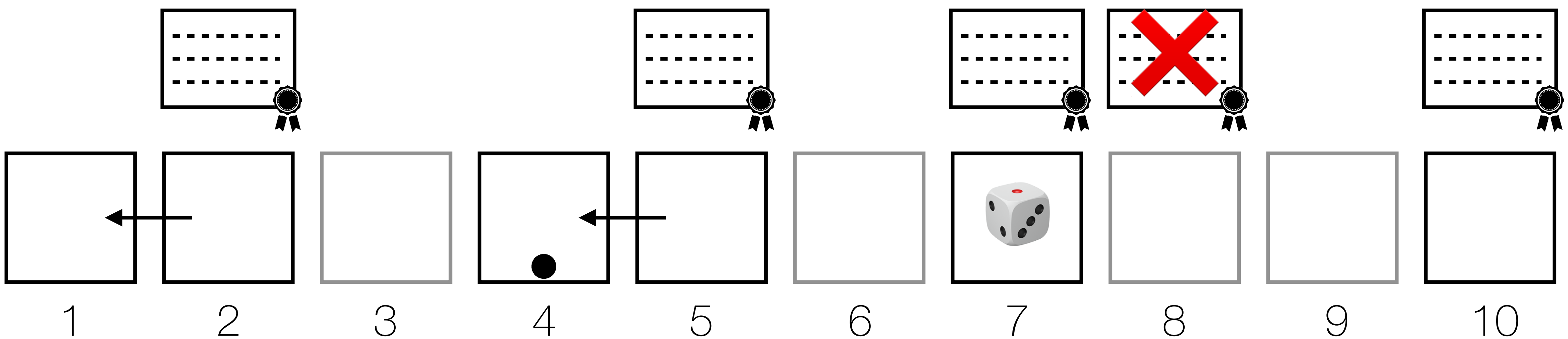
Block headers contains hash of previous block headers

Vaulting with Randomization



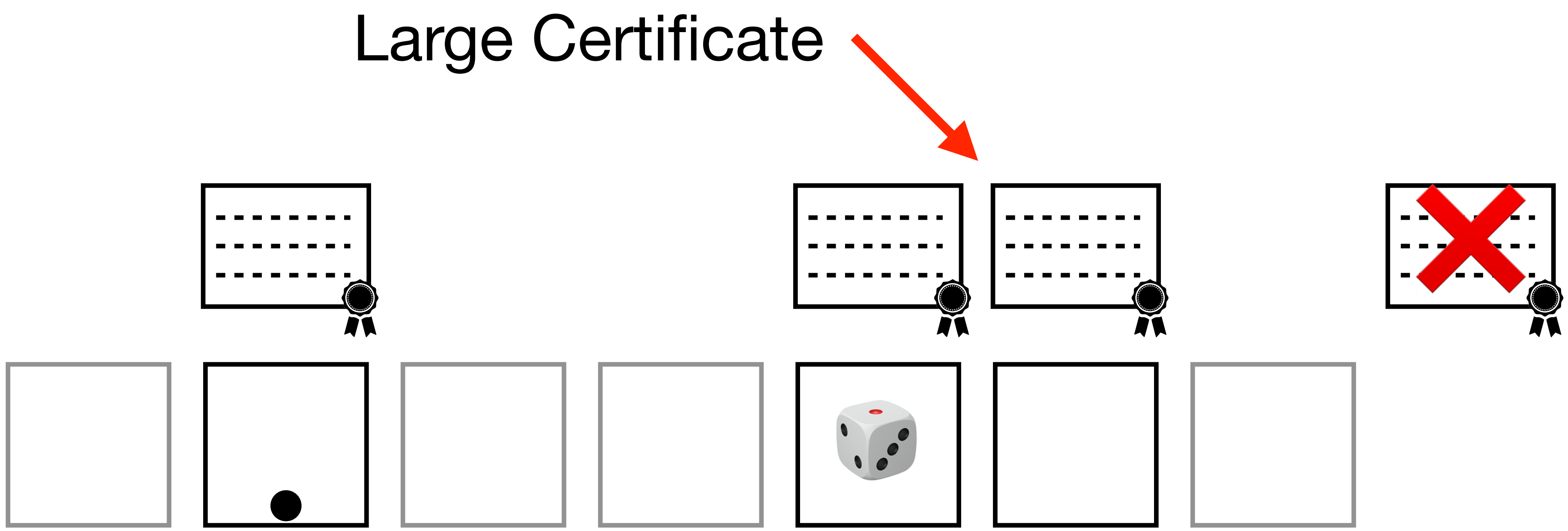
Block headers contains hash of previous block headers

Vaulting with Randomization



Block headers contains hash of previous block headers

Fallback with Algorand



Fallback with Algorand