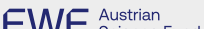
Securing Distributed Hash Tables using Proofs of Space

Christoph Günther

Krzysztof Pietrzak

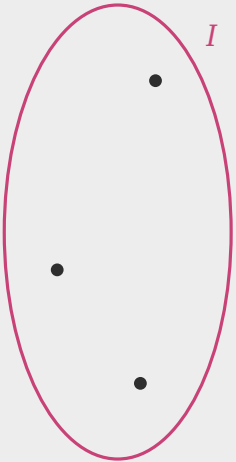
CTB Workshop @ Eurocrypt 2025



This research was funded in whole or in part by the Austrian Science Fund (FWF) 10.55776/F85.

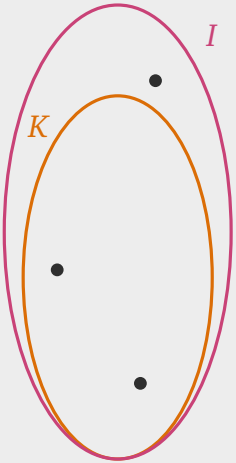
Distributed hash table (DHT)

- Network of nodes n with identifier $\mathrm{id} \in I$



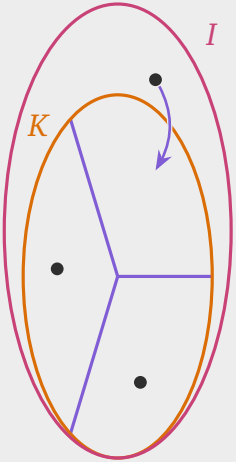
Distributed hash table (DHT)

- Network of nodes n with identifier $\mathrm{id} \in I$
- Key space $K \subseteq I$ (e.g., file hashes)



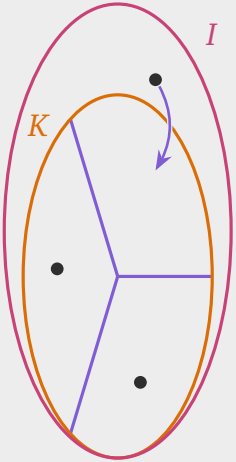
Distributed hash table (DHT)

- Network of nodes n with identifier $\mathrm{id} \in I$
- Key space $K \subseteq I$ (e.g., file hashes)
- Each node is responsible for some $K' \subseteq K$



Distributed hash table (DHT)

- Network of nodes n with identifier $\mathrm{id} \in I$
- Key space $K \subseteq I$ (e.g., file hashes)
- Each node is responsible for some $K' \subseteq K$
- Efficient protocol $\mathrm{lookup}\colon K \to I$



Applications

- Trackerless BitTorrent

Applications

- Trackerless BitTorrent
- Tor Hidden Services

Applications

- Trackerless BitTorrent
- Tor Hidden Services
- Peer discovery in Ethereum (discv4/5)

Applications

- Trackerless BitTorrent
- Tor Hidden Services
- Peer discovery in Ethereum (discv4/5)
- Data availability sampling (potentially)

Applications

- Trackerless BitTorrent
- Tor Hidden Services
- Peer discovery in Ethereum (discv4/5)
- Data availability sampling (potentially)
- Distributed storage networks (e.g., IPFS or Swarm)

Applications

- Trackerless BitTorrent
- Tor Hidden Services
- Peer discovery in Ethereum (discv4/5)
- Data availability sampling (potentially)
- Distributed storage networks (e.g., IPFS or Swarm)

IPFS: $\approx$ 45 000 DHT server nodes `https://probelab.io/ipfs/kpi/`, 2025-04-21

Sybil attacks

Adversary joins the network with multiple nodes using different identities

Sybil attacks

Adversary joins the network with multiple nodes using different identities

- IPFS content censorship attack [Sridhar et al., NDSS'24]

Sybil attacks

Adversary joins the network with multiple nodes using different identities

- IPFS content censorship attack [Sridhar et al., NDSS'24]
- Eclipse attack against IPFS [Prünster et al., USENIX'22]

Sybil attacks

Adversary joins the network with multiple nodes using different identities

- IPFS content censorship attack [Sridhar et al., NDSS'24]
- Eclipse attack against IPFS [Prünster et al., USENIX'22]
- Eclipse attack against Ethereum's peer-to-peer network [Marcus et al., ePrint 2018/236]

Outline

1. DHT constructions & attacks

Outline

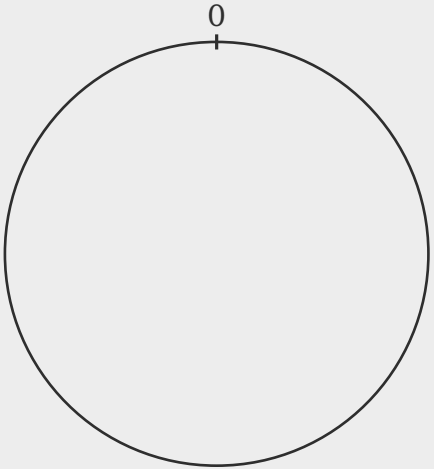
1. DHT constructions & attacks
2. Proof of work & downsides

Outline

1. DHT constructions & attacks
2. Proof of work & downsides
3. Proof of space (PoSp)
    - Why space is a better fit
    - Combining DHTs + PoSp
    - Theoretical analysis

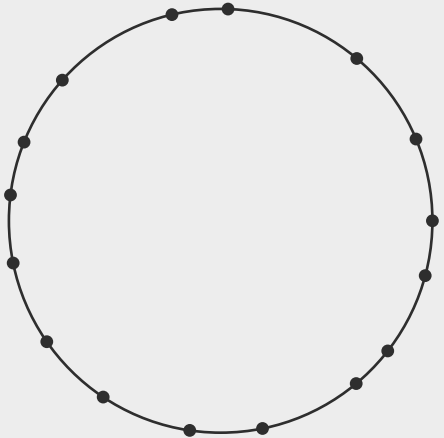
Chord

- $I = K = \mathbb{Z}_N$



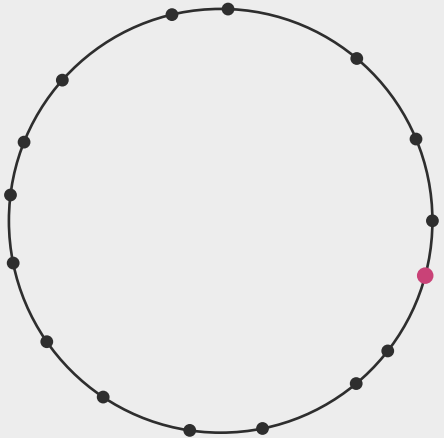
Chord

- $I = K = \mathbb{Z}_N$



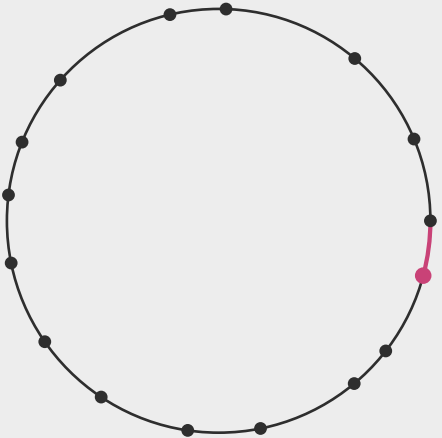
Chord

- $I = K = \mathbb{Z}_N$



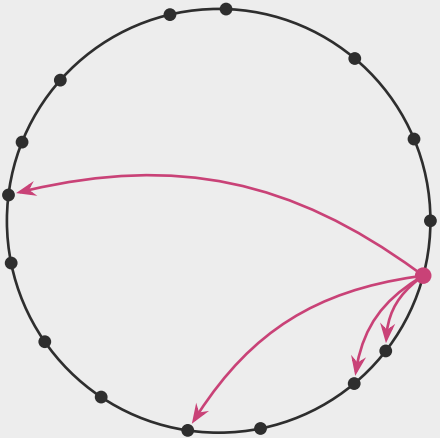
Chord

- $I = K = \mathbb{Z}_N$
- Responsible for preceding keys



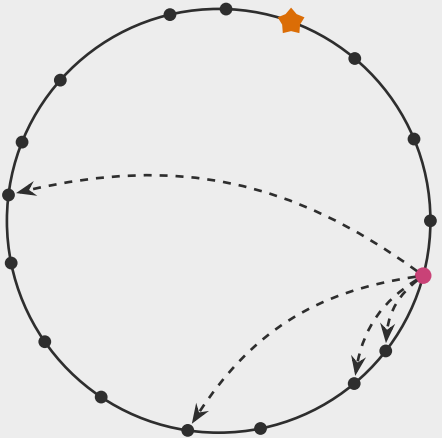
Chord

- $I = K = \mathbb{Z}_N$
- Responsible for preceding keys
- # of peers: $O(\log n)$



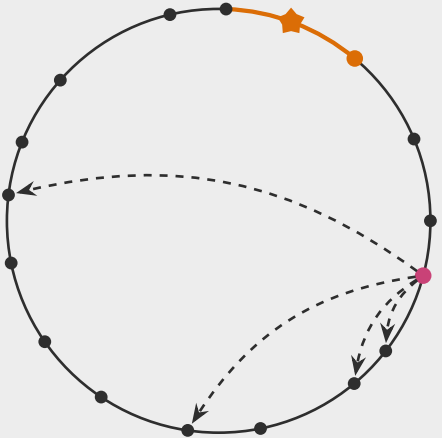
Chord

- $I = K = \mathbb{Z}_N$
- Responsible for preceding keys
- # of peers: $O(\log n)$



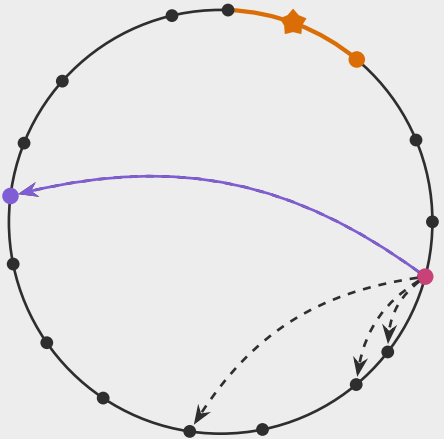
Chord

- $I = K = \mathbb{Z}_N$
- Responsible for preceding keys
- # of peers: $O(\log n)$



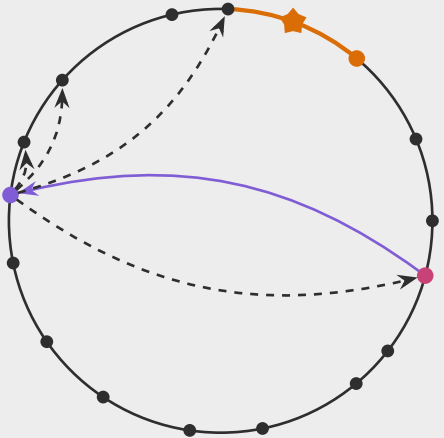
Chord

- $I = K = \mathbb{Z}_N$
- Responsible for preceding keys
- # of peers: $O(\log n)$



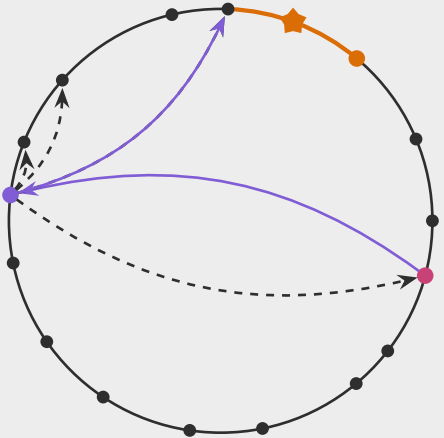
Chord

- $I = K = \mathbb{Z}_N$
- Responsible for preceding keys
- # of peers: $O(\log n)$



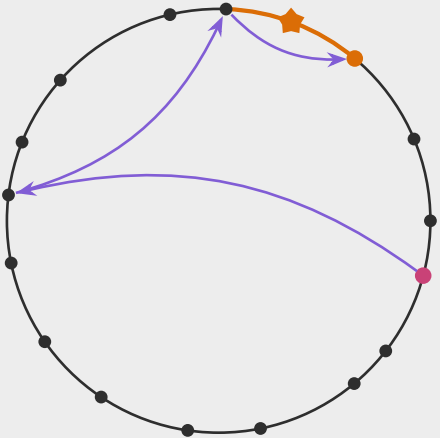
Chord

- $I = K = \mathbb{Z}_N$
- Responsible for preceding keys
- # of peers: $O(\log n)$

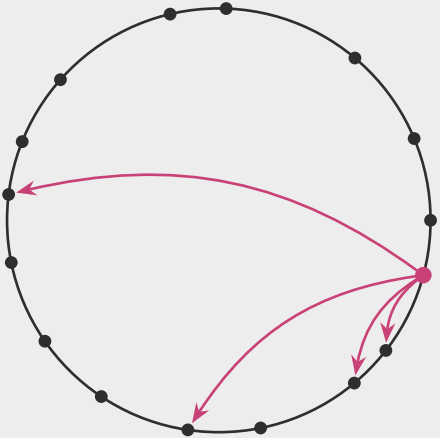


Chord

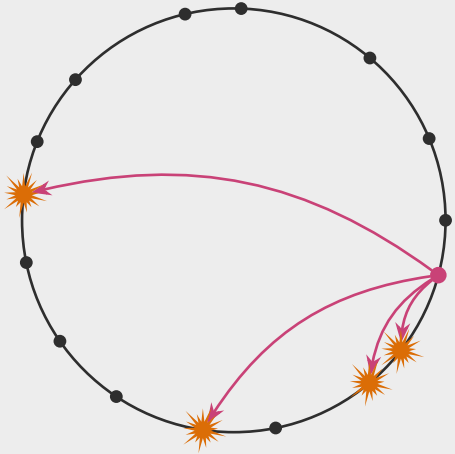
- $I = K = \mathbb{Z}_N$
- Responsible for preceding keys
- # of peers: $O(\log n)$
- # of hops: $O(\log n)$



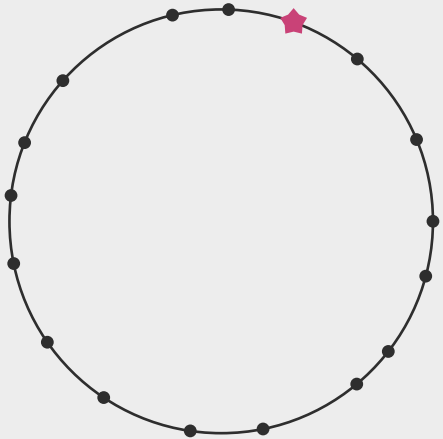
Eclipse attack



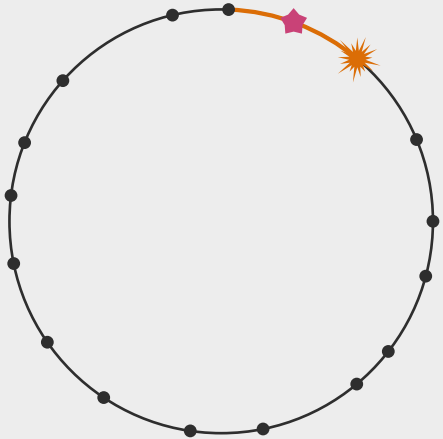
Eclipse attack



Censorship attack



Censorship attack



Kademlia

- Practically more efficient (different distance metric)

Kademlia

- Practically more efficient (different distance metric)
- Redundancy/robustness features

Kademlia

- Practically more efficient (different distance metric)
- Redundancy/robustness features
    - Multiple peers of similar distance

Kademlia

- Practically more efficient (different distance metric)
- Redundancy/robustness features
  - Multiple peers of similar distance
  - Multiple nodes responsible for the same key

Kademlia

- Practically more efficient (different distance metric)
- Redundancy/robustness features
  - Multiple peers of similar distance
  - Multiple nodes responsible for the same key
  - Concurrent lookups

Preventing Attacks

- Resistance of the DHT against Sybils

Preventing Attacks

- Resistance of the DHT against Sybils
    - DHTs like Kademlia with ad-hoc guarantees

Preventing Attacks

- Resistance of the DHT against Sybils
  - DHTs like Kademlia with ad-hoc guarantees
  - Provably secure schemes (e.g., using committees + Byzantine agreement)

Preventing Attacks

- Resistance of the DHT against Sybils
  - DHTs like Kademlia with ad-hoc guarantees
  - Provably secure schemes (e.g., using committees + Byzantine agreement)
- Ease of Sybil creation

Preventing Attacks

- Resistance of the DHT against Sybils
  - DHTs like Kademlia with ad-hoc guarantees
  - Provably secure schemes (e.g., using committees + Byzantine agreement)
- Ease of Sybil creation
  - Reducing the number of Sybils

Preventing Attacks

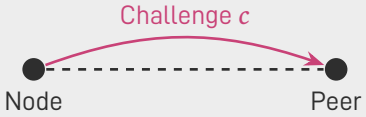
- Resistance of the DHT against Sybils
  - DHTs like Kademlia with ad-hoc guarantees
  - Provably secure schemes (e.g., using committees + Byzantine agreement)
- Ease of Sybil creation
  - Reducing the number of Sybils
  - Restricting free identifier choice

Preventing Attacks

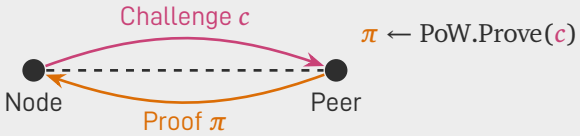
- Resistance of the DHT against Sybils
  - DHTs like Kademlia with ad-hoc guarantees
  - Provably secure schemes (e.g., using committees + Byzantine agreement)
- Ease of Sybil creation
  - **Reducing the number of Sybils**
  - Restricting free identifier choice

Proof of work (PoW)

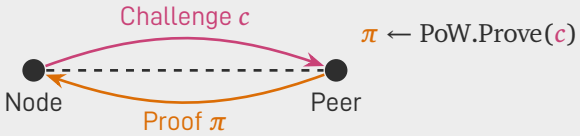
Proof of work (PoW)



Proof of work (PoW)

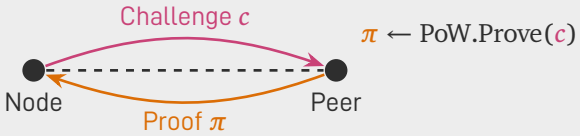


Proof of work (PoW)



Issues

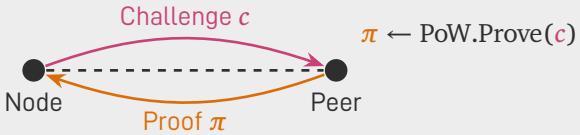
Proof of work (PoW)



Issues

1. DHT nodes usually don't have a lot of computing power

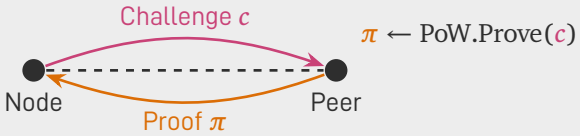
Proof of work (PoW)



Issues

1. DHT nodes usually don't have a lot of computing power
   $\Rightarrow$ PoW difficulty cannot be too high
   $\Rightarrow$ No meaningful security (resource asymmetry)

Proof of work (PoW)



Issues

1. DHT nodes usually don't have a lot of computing power
   $\Rightarrow$ PoW difficulty cannot be too high
   $\Rightarrow$ No meaningful security (resource asymmetry)
2. Wastes energy constantly

Wasting disk space

1. Synergy with storage applications (e.g., IPFS, data availability sampling)

Wasting disk space

1. Synergy with storage applications (e.g., IPFS, data availability sampling)
    - Honest nodes usually have a lot of disk space (e.g., IPFS)

Wasting disk space

1. Synergy with storage applications (e.g., IPFS, data availability sampling)
    - Honest nodes usually have a lot of disk space (e.g., IPFS)
    - Meaningful security guarantees:

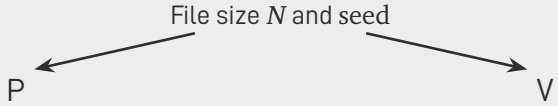
        To control a large fraction of all nodes, the adversary must contribute a large fraction of the total space dedicated to the application

Wasting disk space

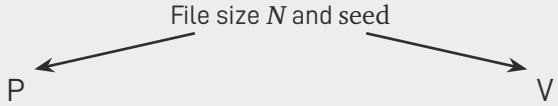
1. Synergy with storage applications (e.g., IPFS, data availability sampling)
    - Honest nodes usually have a lot of disk space (e.g., IPFS)
    - Meaningful security guarantees:

      To control a large fraction of all nodes, the adversary must contribute a large fraction of the total space dedicated to the application
2. Energy efficient (after initial setup)

Proof of space (PoSp) à la Filecoin



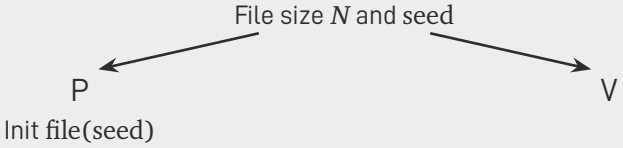
Proof of space (PoSp) à la Filecoin



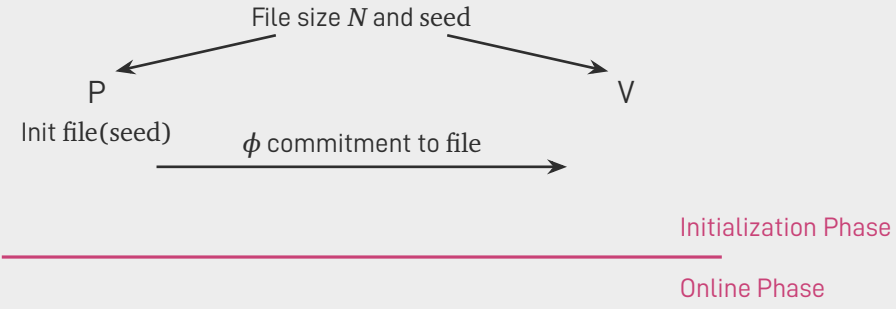
Initialization Phase

Online Phase

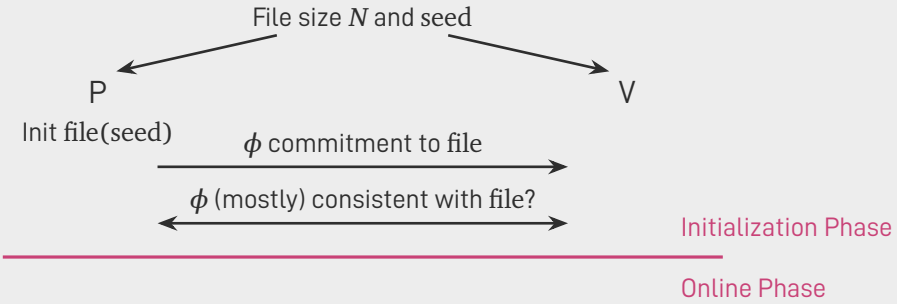
Proof of space (PoSp) à la Filecoin



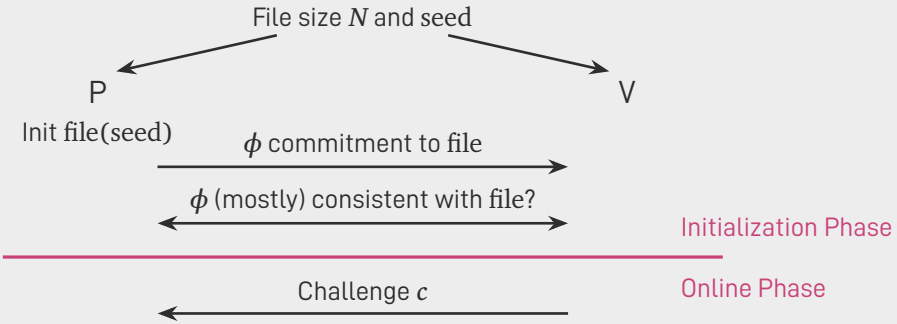
Proof of space (PoSp) à la Filecoin



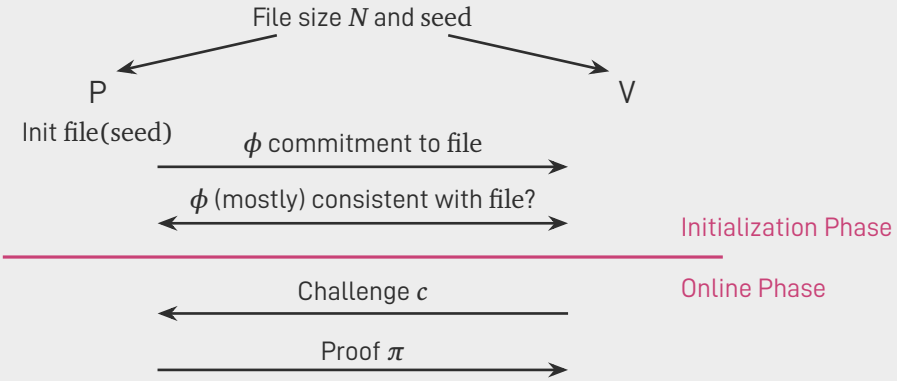
Proof of space (PoSp) à la Filecoin



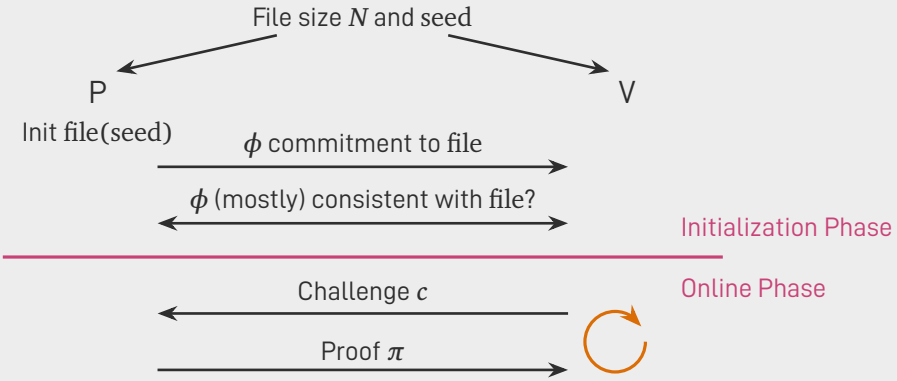
Proof of space (PoSp) à la Filecoin



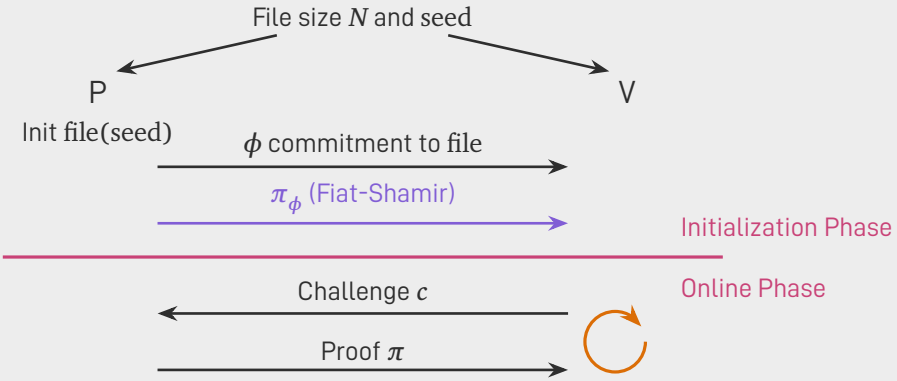
Proof of space (PoSp) à la Filecoin



Proof of space (PoSp) à la Filecoin



Proof of space (PoSp) à la Filecoin



Proofs of Space (PoSp)

- $\text{PoSp.Init}(N, \text{seed}) \rightarrow (\text{file}, \phi, \pi_\phi)$ where $|\text{file}| = N$ bits

Proofs of Space (PoSp)

- $\text{PoSp.Init}(N, \text{seed}) \rightarrow (\text{file}, \phi, \pi_\phi)$ where $|\text{file}| = N$ bits
- $\text{PoSp.Prove}(\text{file}, \phi, c) \rightarrow \pi_c$

Proofs of Space (PoSp)

- $\text{PoSp.Init}(N, \text{seed}) \rightarrow (\text{file}, \phi, \pi_\phi)$ where $|\text{file}| = N$ bits
- $\text{PoSp.Prove}(\text{file}, \phi, c) \rightarrow \pi_c$
- ...and verification algorithms

Proofs of Space (PoSp)

- $\text{PoSp.Init}(N, \text{seed}) \rightarrow (\text{file}, \phi, \pi_\phi)$ where $|\text{file}| = N$ bits
- $\text{PoSp.Prove}(\text{file}, \phi, c) \rightarrow \pi_c$
- ...and verification algorithms

Space-hardness: A cheating prover

storing *less than* αN bits

running in *less than* $\beta \, \text{time}(\text{Init})$

fails challenge c with probability *at least* p

Basic construction

Combines PoSp with existing DHT

Basic construction

Combines PoSp with existing DHT

Joining the DHT

Being added as a peer

While connected

Basic construction

Combines PoSp with existing DHT

Joining the DHT Store $(\text{file}, \phi, \pi_\phi) \leftarrow \text{PoSp.Init}(N, \text{DHT.id})$

Being added as a peer

While connected

Basic construction

Combines PoSp with existing DHT

Joining the DHT	Store $(\text{file}, \phi, \pi_\phi) \leftarrow \text{PoSp.Init}(N, \text{DHT.id})$
Being added as a peer	Send (ϕ, π_ϕ) to them
While connected	

Basic construction

Combines PoSp with existing DHT

Joining the DHT	Store $(\text{file}, \phi, \pi_\phi) \leftarrow \text{PoSp.Init}(N, \text{DHT.id})$
Being added as a peer	Send (ϕ, π_ϕ) to them
While connected	Every t seconds, receive κ PoSp challenges $c_1, \ldots, c_\kappa$
	Reply $\pi_i \leftarrow \text{PoSp.Prove}(\text{file}, \phi, c_i)$ for every $i \in [\kappa]$

Security guarantee

The number of Sybil nodes f is bounded by

$$f < \frac{S_{\text{ADV}}}{\alpha N}$$

except for exponentially small probability in λ if

- number of challenges $\kappa = \lambda / p$
- challenge interval $t < \beta \, \text{time(Init)}$

Security guarantee

The number of Sybil nodes f is bounded by

$$f < \frac{S_{\text{ADV}}}{\alpha N}$$

except for exponentially small probability in λ if

- number of challenges $\kappa = \lambda / p$ $\Rightarrow$ boost detection probability
- challenge interval $t < \beta\,\text{time(Init)}$

Security guarantee

The number of Sybil nodes f is bounded by

$$f < \frac{S_{\text{ADV}}}{\alpha N}$$

except for exponentially small probability in λ if

- number of challenges $\kappa = \lambda / p$ $\Rightarrow$ boost detection probability
- challenge interval $t < \beta\,\text{time(Init)}$ $\Rightarrow$ must use αN space

Security guarantee

The number of Sybil nodes f is bounded by

$$f < \frac{S_{\text{ADV}}}{\alpha N}$$

Every Sybil costs αN space

except for exponentially small probability in λ if

- number of challenges $\kappa = \lambda / p$ $\Rightarrow$ boost detection probability
- challenge interval $t < \beta\,\text{time(Init)}$ $\Rightarrow$ must use αN space

Virtual nodes

Problem: Often want a bound on the *fraction* of Sybils $f/n < \gamma$

Virtual nodes

Problem: Often want a bound on the *fraction* of Sybils $f/n < \gamma$

Solution: Honest node with space S participates as $\delta S/N$ distinct identities

Virtual nodes

Problem: Often want a bound on the *fraction* of Sybils $f/n < \gamma$

Solution: Honest node with space S participates as $\delta S/N$ distinct identities

$\Rightarrow n$ grows linearly in the total honest space S_{HON}

Virtual nodes

Problem: Often want a bound on the *fraction* of Sybils $f/n < \gamma$

Solution: Honest node with space S participates as $\delta S/N$ distinct identities

$\Rightarrow n$ grows linearly in the total honest space S_{HON}

Security guarantee

$f/n < \gamma$ as long as $S_{\text{ADV}} < \text{const}(\alpha, \gamma, \delta) \cdot S_{\text{HON}}$

Practicality?

Observation: Depends on the PoSp parameters α, β and p

Practicality?

Observation: Depends on the PoSp parameters α, β and p

Issue: Too many challenges (# of challenges κ large, interval t small)

Practicality?

Observation: Depends on the PoSp parameters α, β and p

Issue: Too many challenges (# of challenges κ large, interval t small)

$\Rightarrow$ bandwidth too high

Practicality?

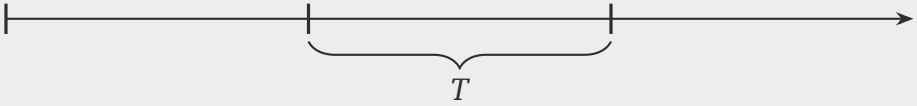
Observation: Depends on the PoSp parameters α, β and p

Issue: Too many challenges (# of challenges κ large, interval t small)

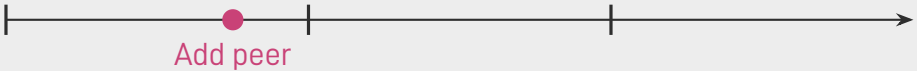
$\Rightarrow$ bandwidth too high

Solution: Improve protocol: Probabilistic challenges & time epochs

Improved protocol

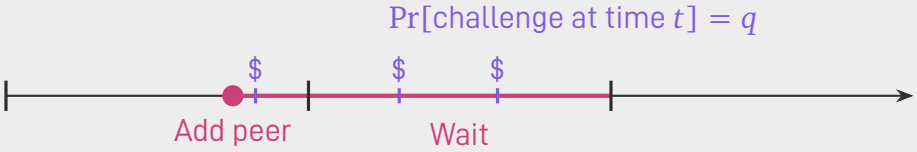


Improved protocol

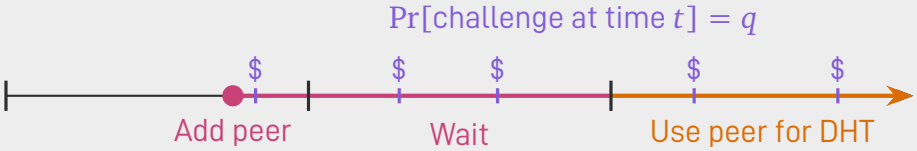


Improved protocol

Improved protocol



Improved protocol



Security guarantee

$T = 4\lambda/(pq)$ and ADV has space for $f = S_{\text{ADV}}/(\alpha N)$ Sybils

$\Rightarrow$

at most $2f$ Sybils except for negl. probability in λ

Security guarantee

$T = 4\lambda/(pq)$ and ADV has space for $f = S_{\text{ADV}}/(\alpha N)$ Sybils

$\Rightarrow$

at most $2f$ Sybils except for negl. probability in λ

Note that pq is that a probability that a peer is challenged *and* detected.

Security guarantee

$T = 4\lambda/(pq)$ and ADV has space for $f = S_{\text{ADV}}/(\alpha N)$ Sybils

$\Rightarrow$

at most $2f$ Sybils except for negl. probability in λ

Note that pq is that a probability that a peer is challenged *and* detected.

Improvements over basic scheme

1. Fewer challenges in expectation
2. No need to boost detection probability (constant p is fine)

Contributions

- Using disk space to limit Sybils in DHTs
- Simple & practical schemes
- Theoretical analysis

Future work

- Simulations/implementation
- Don't waste space (e.g., backups)
- DHTs handling large f/n

Securing Distributed Hash Tables using Proofs of Space

C. U. Günther and K. Pietrzak

https://gnthr.eu/uploads/posp_dht_draft.pdf