Privacy & libp2p

Double Hashing in the DHT

Gui Michel
@guissou

ProbeLab,
Protocol Labs

IPFS Camp

Protocol Labs

IPFS Camp
30th October 2022

Agenda

- Goals
- DHT Content Routing in IPFS
- Double Hashing DHT Design
  - Double Hash
  - Prefix Lookup
  - Provider Records Encryption
  - Provider Records Authentication
- Demo
- Transition

IPFS Camp

Goals & non-goals

What this solution improves

- Reader Privacy when looking up content in the DHT
- From adversary able to observe network (e.g DHT server nodes)

What this solution does NOT target

- Writer Privacy in the DHT
- Data Transfer Privacy (e.g Bitswap)

IPFS Camp

Privacy & Content Routing - DHT Challenges

- **Content Routing** needs the content to be discoverable
  - DHT server nodes need to know what you are looking for to help you find it
- **Privacy** needs the content to be private
  - You don't want DHT server nodes to know which content you are looking for

Definitions

MultiHash $\leftarrow$ Hash(): `mHash = 01010111`

CID $\leftarrow$ `prefix0 + mHash = bafybeiaqr6csdcnxrkpx23...`

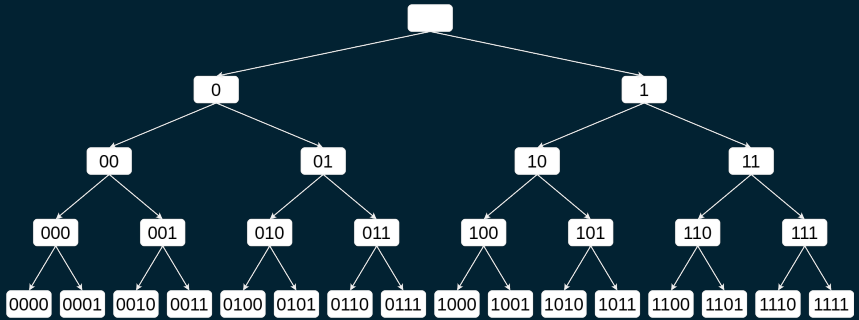
Content second hash $\leftarrow$ Hash(`prefix1 + mHash`) = `10110001`

- Note: *For simplicity we represent the second hash as* `Hash(CID)`

Provider Record: Pointer `CID` $\rightarrow$ `PeerID` stored in the DHT

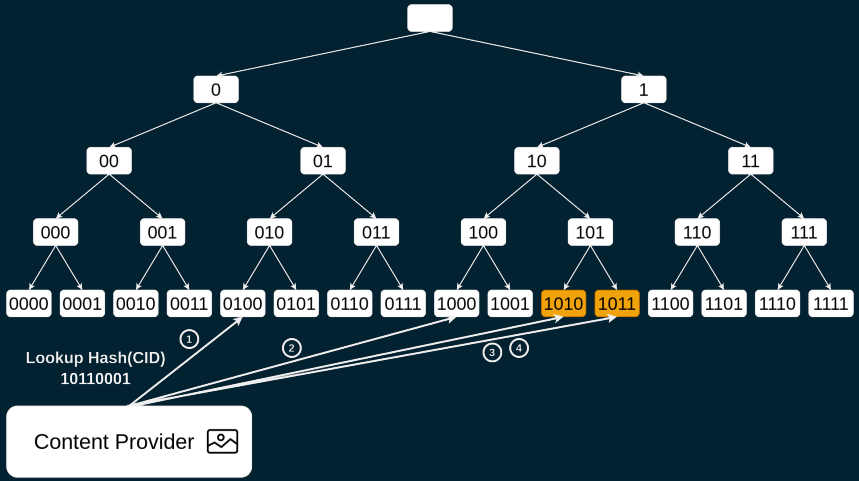
IPFS Camp

Pinning content to the DHT



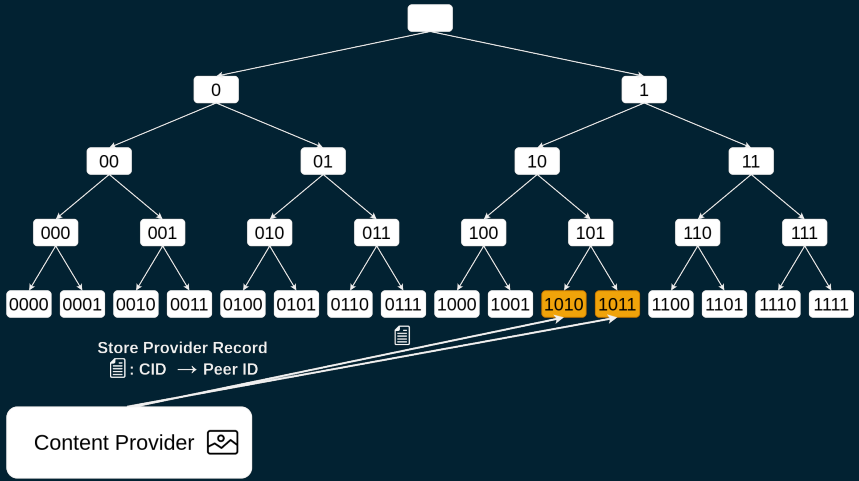
IPFS Camp

Pinning content to the DHT



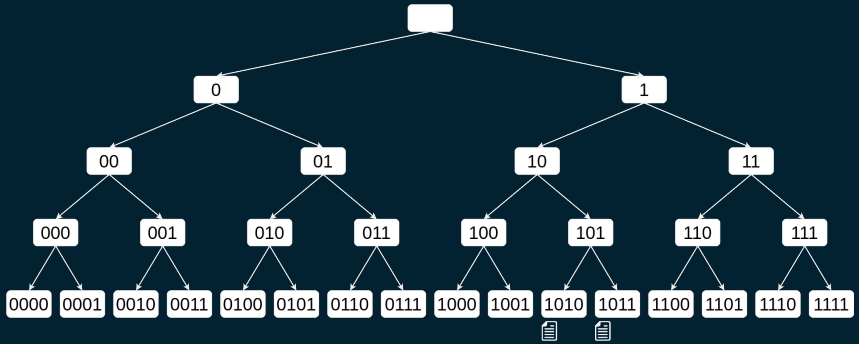
IPFS Camp

Pinning content to the DHT



IPFS Camp

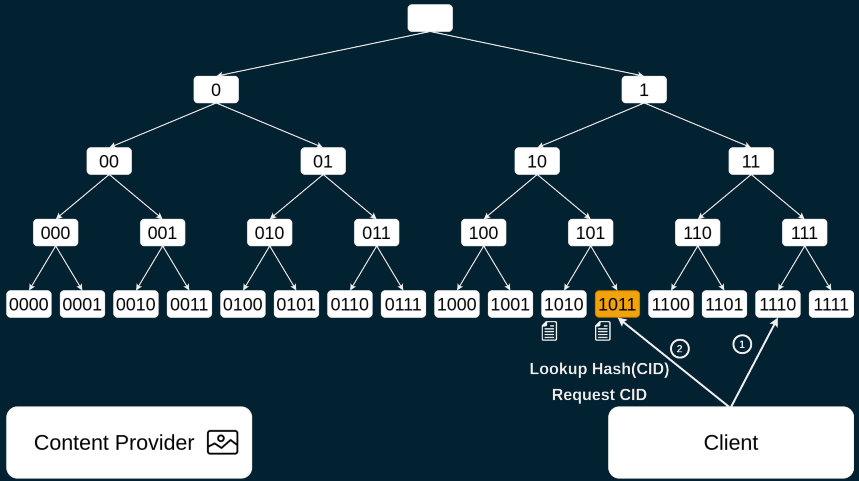
Pinning content to the DHT



Content Provider

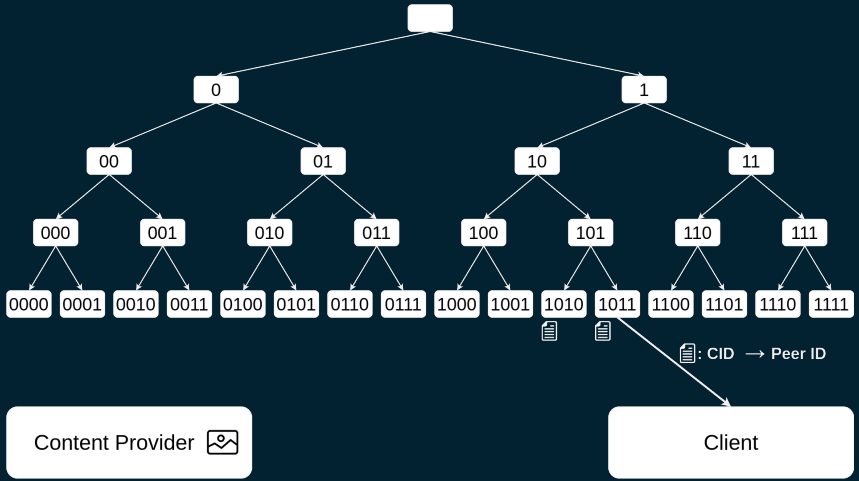
IPFS Camp

Looking up content in the DHT



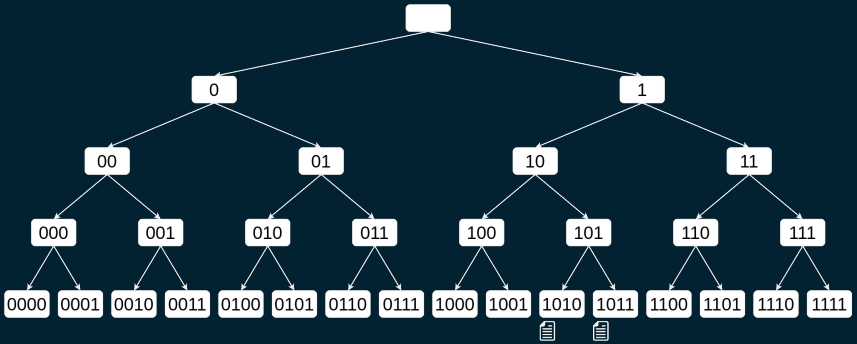
IPFS Camp

Looking up content in the DHT



IPFS Camp

Looking up content in the DHT



IPFS Camp

Content Routing in the IPFS DHT

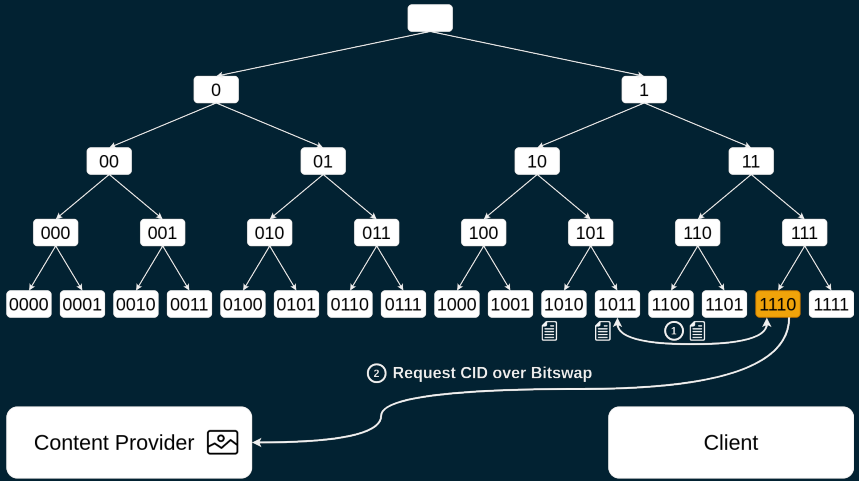
What does the DHT learn from each request?

- Client's `PeerID`
- Client's IP address
- CID of the content we want to access

What can DHT server nodes do with this information?

- Request CID to fetch the content
- They learn which content was accessed by the client

Spying on the accessed content



IPFS Camp

1) Double Hashing

How did we get to double hashing?

- Content is addressed by its CID, derived from the content's hash
- Looking up content leaks the CID
- We need another DHT content identifier hiding the CID

Properties of `Hash(CID)`

- `Hash(CID)` can efficiently be computed from `CID`
- `CID` cannot be recovered from `Hash(CID)`

2) Prefix requests

- `k-anonymity` & plausible deniability
- Request a `prefix` of `Hash(CID)`
- The DHT will return all provider records matching this `prefix`

But:

- No `l-diversity` nor `t-closeness`
- Prefix requests correlation is still possible

IPFS Camp

3) Provider Record Encryption

- `PeerID` is encrypted by the Content Provider
- Encryption key is derived from the `CID`
- Symmetric encryption `AES-GCM`
- Knowledge of `CID` is required to read the `PeerID`
- Peers with knowledge of `Hash(CID)` cannot read the `PeerID`

IPFS Camp

4) Provider Record Authentication

Bad news

- Provider Records are encrypted, DHT server nodes cannot authenticate them
- Peers can create Provider Records for any known CIDs pointing to any `PeerID`

Solution

- Provider Records get signed by the publisher private key
- DHT server nodes verify the signatures against the publisher's public key
- Clients verify the signature against the `PeerID` decrypted from the Provider Record

Specs

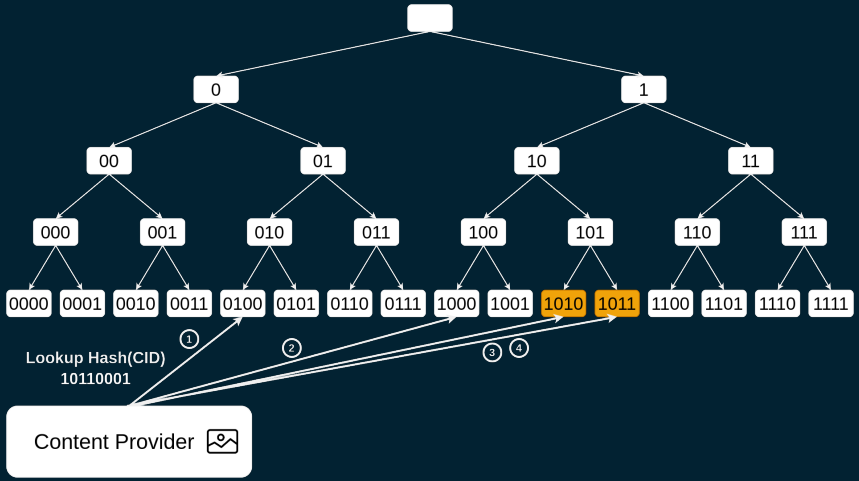
- $EncPeerID \leftarrow Enc_{CID}(PeerID)$
- $Signature \leftarrow Sign_{PeerSK}(EncPeerID)$
- Provider Record $\leftarrow [Hash(CID) \rightarrow (EncPeerID, Signature)]$

In the implementation we use the Multihash `MH` instead of the CID directly

- $EncPeerID \leftarrow Enc_{MH}(PeerID)$
- Provider Record $\leftarrow [Hash(\text{"}CR\_DOUBLEHASH\text{"} + MH) \rightarrow (EncPeerID, Signature)]$

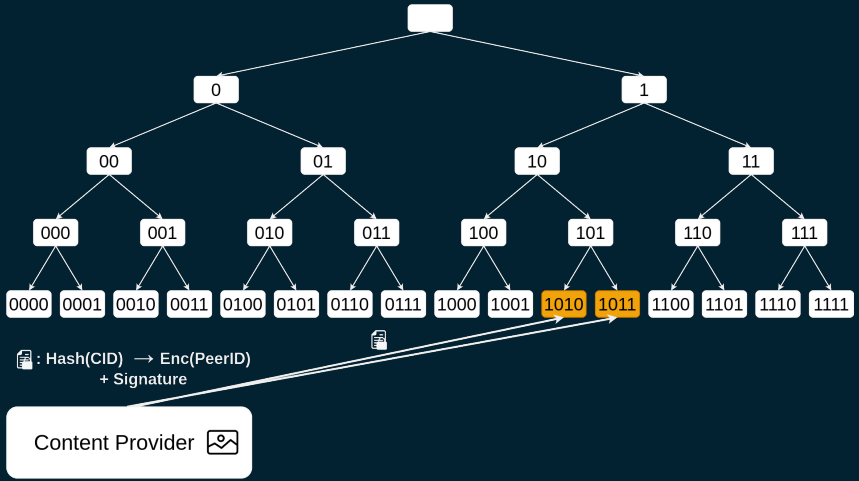
IPFS Camp

Everything together: Pinning Content



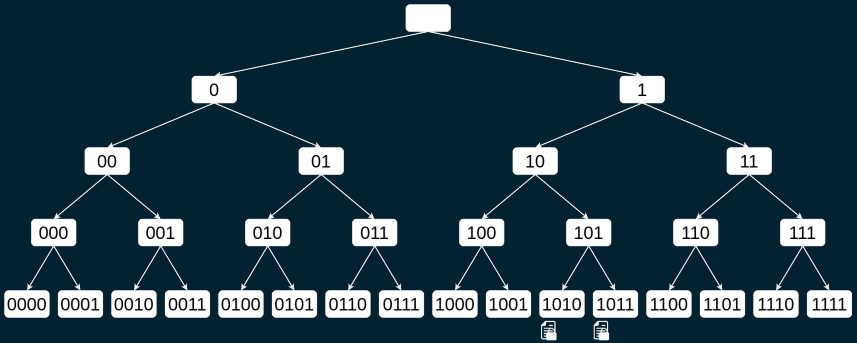
IPFS Camp

Everything together: Pinning Content



IPFS Camp

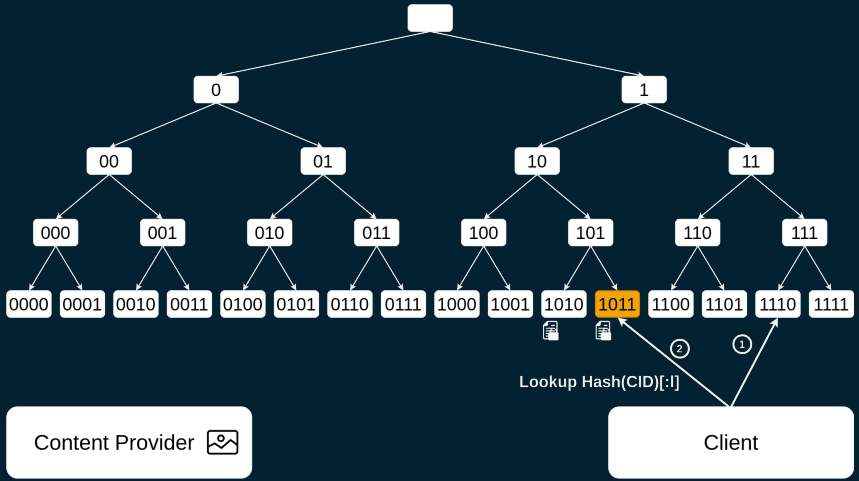
Everything together: Pinning Content



Content Provider

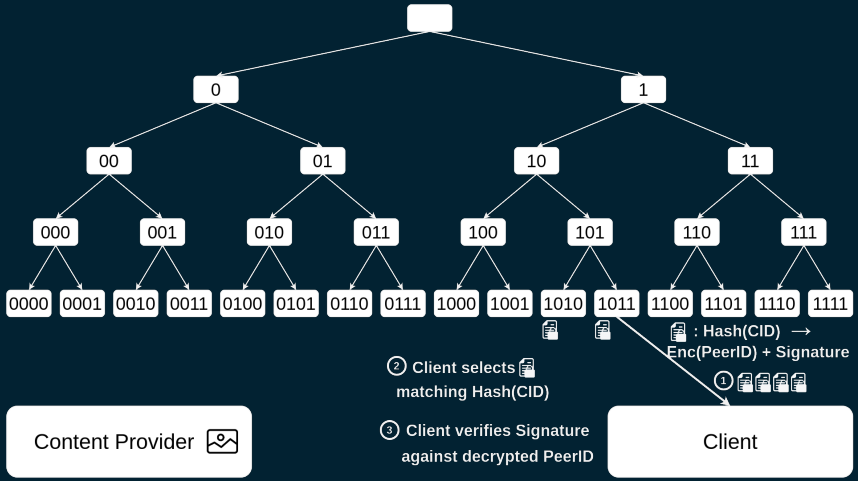
IPFS Camp

Everything together: Retrieving Content



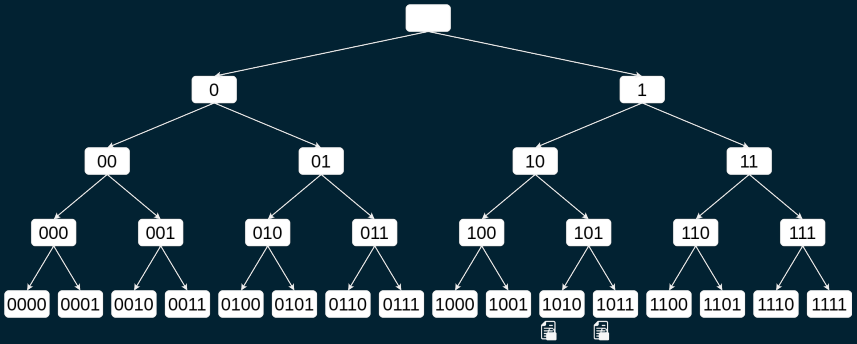
IPFS Camp

Everything together: Retrieving Content



IPFS Camp

Everything together: Retrieving Content



IPFS Camp

Privacy guarantees

- `k-anonymity` and plausible deniability provided by the prefix requests
- Requester `PeerID` can be associated with `Hash(CID)` but not `CID`
- If the adversary doesn't know the `CID` it cannot fetch the content, nor discover the content provider

But:

- If the adversary knows a `CID` whose `Hash(CID)` matches the requested `prefix`, they can try to guess the requested content
- DHT server nodes storing Provider Records can associate `Hash(CID)` with publisher's `PeerID`

IPFS Camp

Overhead

- **Network**: k Provider Records returned (instead of 1)
- **Storage**: Provider Records are larger as they now contain a Signature
- **Computation**: Signature is required for each republish

Changes

- *Only* `libp2p` has to be upgraded
- Applications built on top of `libp2p` automatically benefit from the upgrade
- Integrates in IPFS Reframe
- Double Hashing implemented by the Indexers

- Elizabeth Binks from Chainsafe working on the implementation in `github.com/ChainSafe/go-libp2p-kad-dht/`

IPFS Camp

Demo

- Demo prepared by Elizabeth Binks from Chainsafe

IPFS Camp

Transition challenges

- Protocol change breaks compatibility with the current DHT
- Legacy DHT server nodes cannot serve private requests
- Migration is required
- Only updated nodes can server private requests

Booting up a new DHT

- We currently have 4 DHTs. Client LAN, Server LAN, Client WAN & Server WAN
- *Creating a new DHT* would double the number of active DHTs
- Routing Table size is expected to double
- Future changes means creating new DHTs at each protocol upgrade

IPFS Camp

Upgradable DHT

- Change bucket replacement policy to prefer peers with the same protocol version
- Nodes need to keep track of each other's protocol version
- `k-buckets` with high ID will NOT change
- `k-buckets` with low ID are expected to include only peers with the same protocol version

Analogy

IPFS Camp

DHT *Tram* upgrade

Let's make the Interplanetary routes upgradable for all future vehicules!

IPFS Camp

RPCs

- New `PrivateProvide` 
- New `PrivateLookup` 
- `FindPeer` doesn't change 
- `Provide` and `Lookup` sill available 

DHT load calculations

- Current average load per peer: $\frac{\#all\_requests}{\#all\_peers}$
- Expected load for legacy peers: $\frac{\#legacy\_requests}{\#all\_peers} = \frac{\#all\_requests - \#private\_requests}{\#all\_peers}$
- Expected load for upgraded peers:

$$\frac{\#legacy\_requests}{\#all\_peers} + \frac{\#private\_requests}{\#upgraded\_peers} = \frac{\#all\_requests - \#private\_requests}{\#all\_peers} + \frac{\#private\_requests}{\#upgraded\_peers}$$

Remarks

We may want to stop supporting legacy requests in the future and don’t want to break legacy nodes

- Young prefer young ✅
- Old prefer old ❌
- Ultimate migration required

IPFS Camp

Transition period

1. Support for the Double Hashing DHT has to be pushed to `libp2p` nodes
2. Content Providers publish provider records to both legacy and DH DHTs
3. New client perform private lookup (and can failover to legacy lookup)
4. Content Providers stop publishing Provider Records to legacy DHT
5. New versions of `libp2p` stop supporting legacy content requests

Callout for breaking DHT protocol changes

IPFS Camp

Conclusion

- Double Hashing DHT brings significant reader-privacy improvements
- Provider Records authentication
- Bonus: small writer-privacy improvement
- Low overhead (k Provider Records, Signature)
- Can be applied to other Content Routers (e.g Indexers)
- Double Hashing DHT implementation almost done
- We can make the DHT upgradable!

Specs

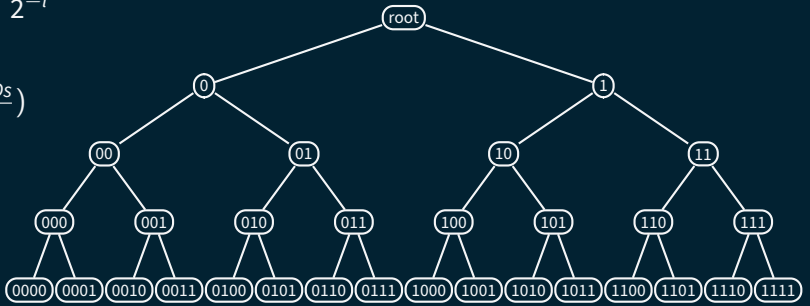
IPFS Camp

Additional slides

IPFS Camp

Prefix Length Selection

- Prefix Length: l
- `k-anonymity`: The requested Provider Record can not be distinguished from at least $k - 1$ other Provider Records
- $k = \#CIDs \times 2^{-l}$
- $\frac{\#CIDs}{k} = 2^l$
- $l = log_2(\frac{\#CIDs}{k})$



IPFS Camp

Threat Model

- **Adversary**: DHT server nodes, observers
- **Target**: Client requesting content to the DHT
- **Results**:
  - Adversaries without CID cannot learn which content the client is accessing
  - Adversaries with CID can learn that client is trying to access content associated with one of `k` provider records
  - DHT server nodes can associate Provider Record with publisher `PeerID`

IPFS Camp