Blockchain technology and applications

Intro

DAT655 Blockchain tech & app

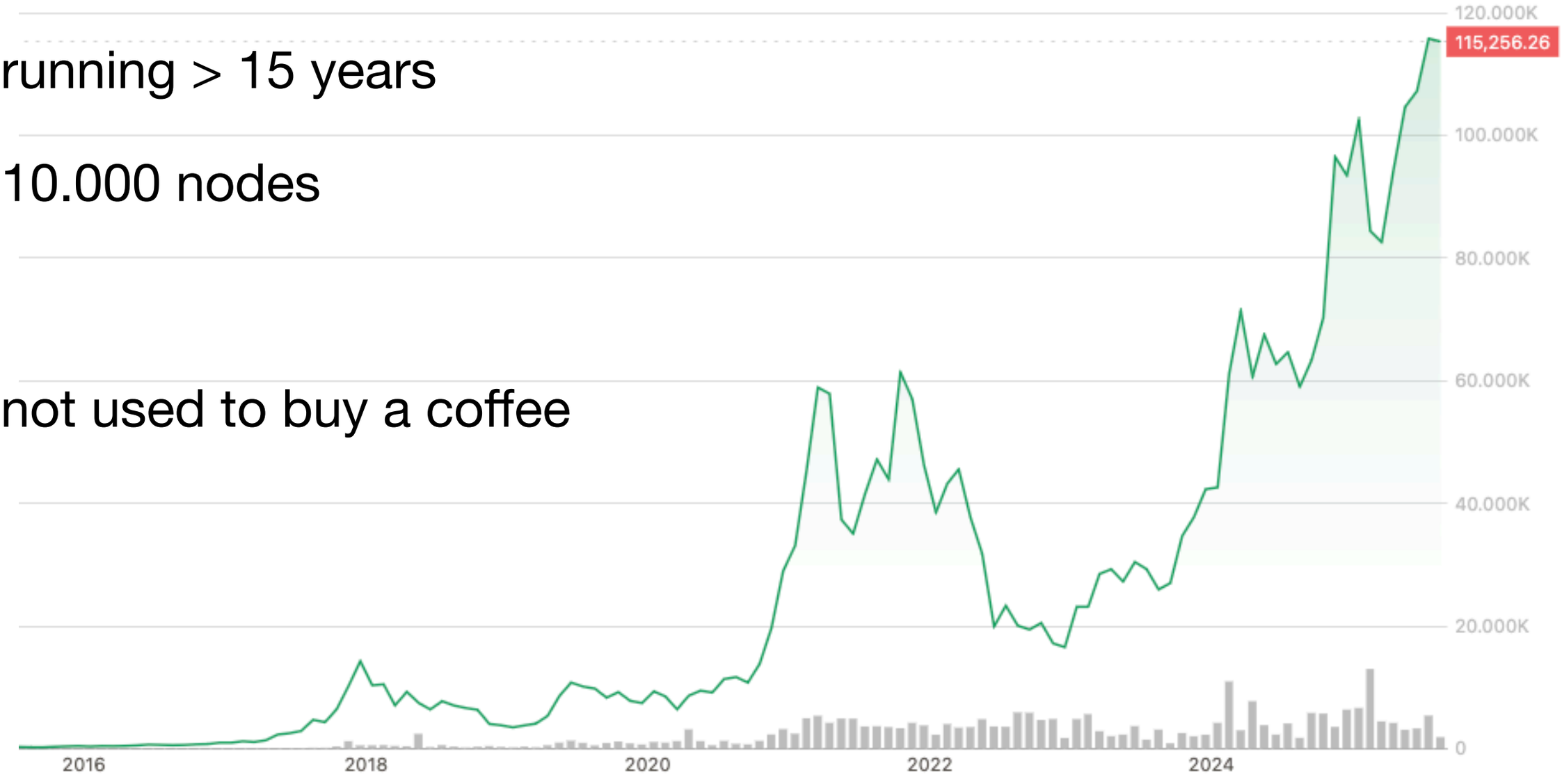
Leander Jehl

Blockchain

Motivation

Bitcoin

- running > 15 years
- 10.000 nodes
- not used to buy a coffee



https://www.coindesk.com/price/bitcoin

Blockchain and applications

Take a look at blockchain technology and applications

- What works
- What does not work

Learning goal: Know how and when to use it.

Know when not to use it.

But: No investment tips

Blockchain and applications

Where can many people, jointly store an important document?

- e.g. Will
- e.g. Ownership list
- Can changes be made?

Blockchain 1

Hashlist and Mercle trees

DAT655 Blockchain tech & app

Leander Jehl

Blockchain datastructure

What is a blockchain

A blockchain is an append only log secured against changed.

Typically a blockchain

- is stored on different nodes

Idea: Log all interactions

- Log enables anyone to reproduce/recreate state.

Cryptographic hash function

Idea

$$H(x) = y$$

Cryptographic hash function

Idea

$$H(x) = y$$

- x string or byte array
- y fixed size byte array

looks random: hashing something new gives a random value

is deterministic: hashing something twice gives the same value

Cryptographic hash function

Properties

```
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo">
</script>
```

Properties:

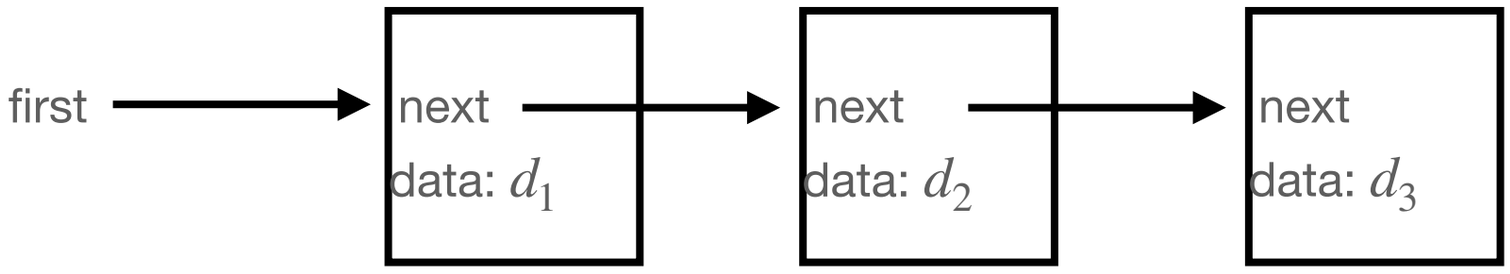
- **Pre-image resistance:**
  given y cannot find x s.t. $H(x) = y$
- **Weak collision resistance:**
  given x cannot find x' s.t. $H(x) = H(x')$
- **Strong collision resistance:**
  cannot find x and x' s.t. $H(x) = H(x')$

Use cases examples:
- **Password hashes**
- **HTML5 integrity attribute**

Hash chain

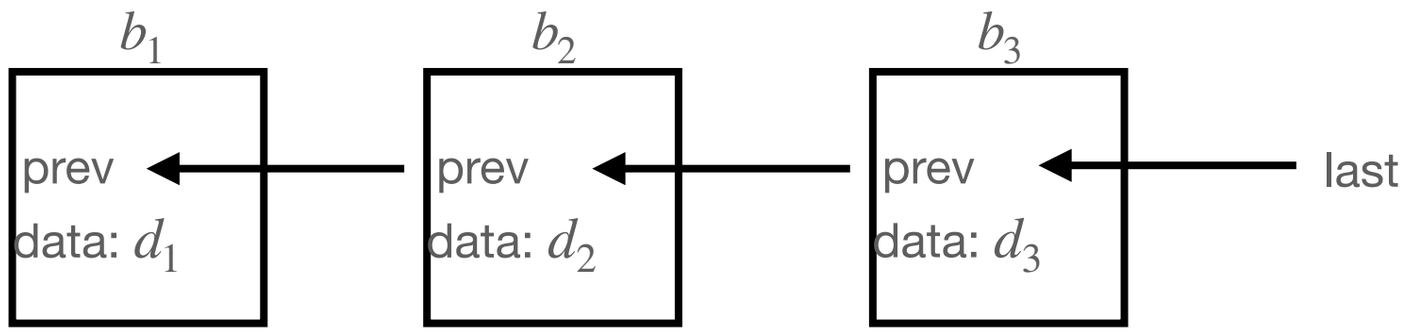
Hash chain

Linked list




```
type Node struct {
    next pointer
    data bytes
}
```

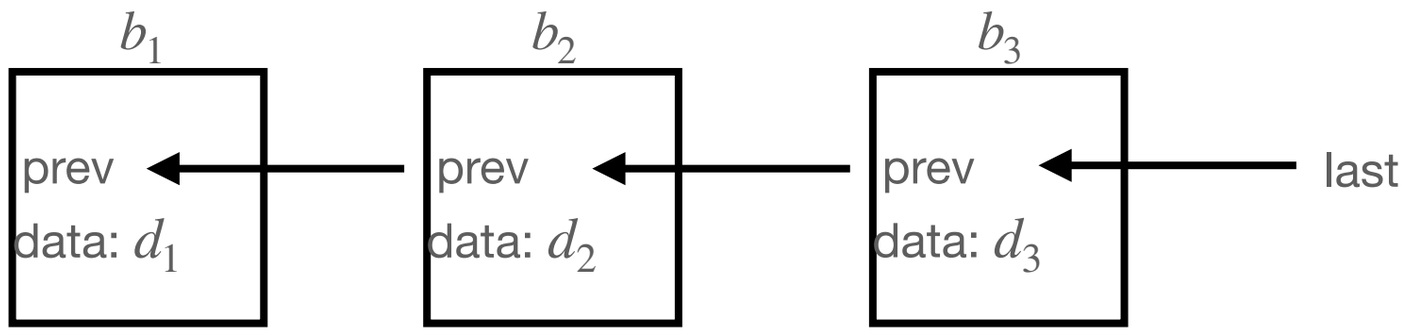
Hash chain




```
type Block struct {
    prev pointer
    data bytes
    prevhash hash
}
```

- Can hash a block by concatenating fields.
- Blockhash gives id:
  $id_b = H(b.\texttt{prevhash}||b.\texttt{data})$
- $b_2.\texttt{prevhash} = id_{b_1}$

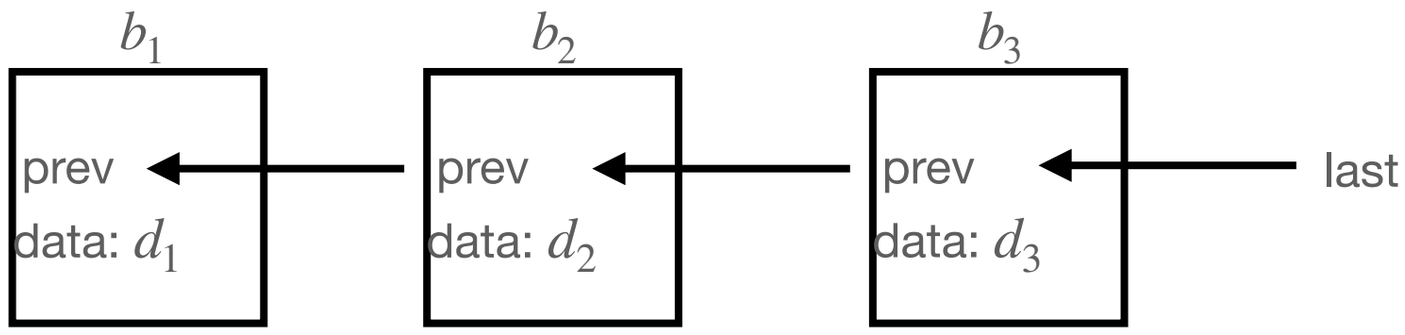
Hash chain



- $id_b = H(b.\texttt{prevhash}||b.\texttt{data})$
- Blockchain identified by id_{b_3}
- Changing d_1 changes id_{b_3}
- Removing b_2 changes id_{b_3}
- Adding b_2' changes id_{b_3}

secured against changes

Hash chain




```
type Block struct {
    prev pointer
    data bytes
    datahash hash
    prevhash hash
    timestamp
}
```

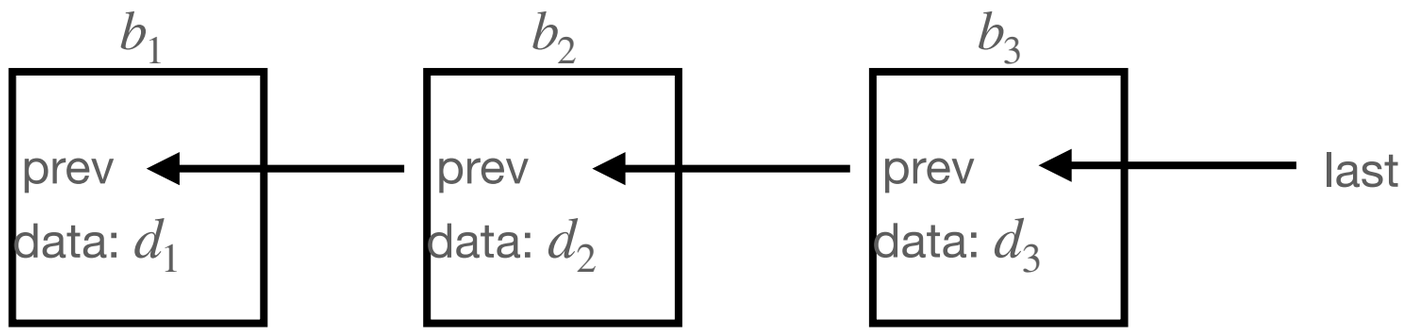
Reduce size:

- $id_b = H(b.\texttt{prevhash}||b.\texttt{datahash})$
- $b.\texttt{datahash} = H(b.\texttt{data})$

- easy to store
- can proof that data is included

Hash chain

Example: Linked timestamping



Trusted source collects data and publishes a new block, e.g. on newspaper.

Hash chain

Digital Signatures and trusted publishers

$$pk, sk \leftarrow setup(\kappa)$$

$$\sigma \leftarrow sign(sk, msg)$$

$$bool \leftarrow verify(\sigma, msg, pk)$$

Ideas:

- Require a trusted party to sign every new block
- Require m out of n trusted parties to sign a block

  Permissioned blockchains!

Merkle trees

Merkle trees

Problem

```
type Block struct {
    prev pointer
    data bytes
    datahash hash
    prevhash hash
    timestamp
}
```

Idea:

- put multiple data items into one block

- Is my item in the block?

Merkle trees

Ideas

Data items: $D_1, D_2, D_3, D_4, \ldots$

- `datahash` $= h$

Data as a hash

- $h = H(D_1 || D_2 || D_3 || \ldots)$

Data as a list of hashes

- $h = H(D_1) || H(D_2) || H(D_3) || \ldots$

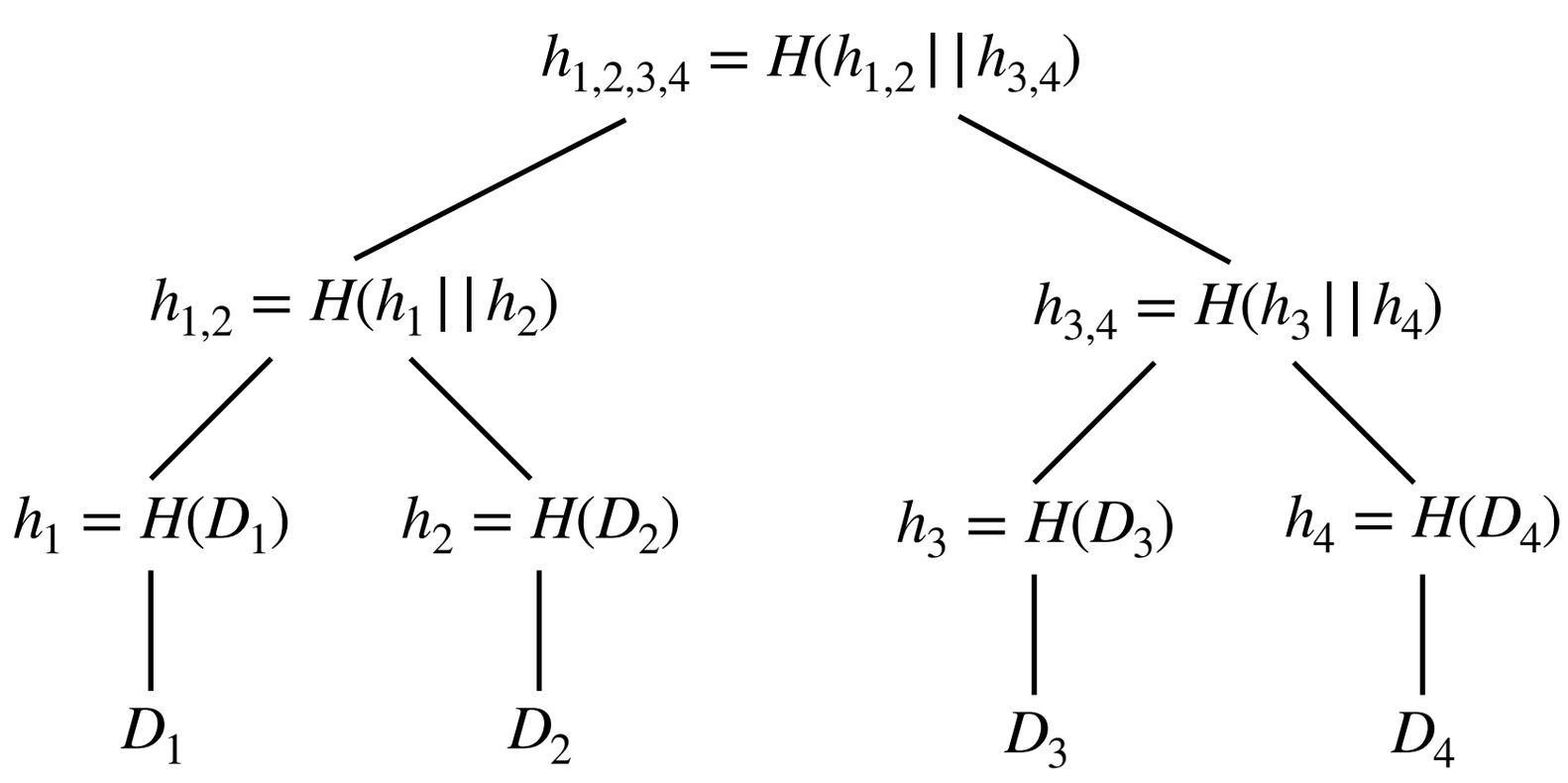
```
type Block struct {
    prev pointer
    data bytes
    datahash hash
    prevhash hash
    timestamp
}
```

Merkle trees

Design

Data items: $D_1, D_2, D_3, D_4, \ldots$

- `datahash` $= h$

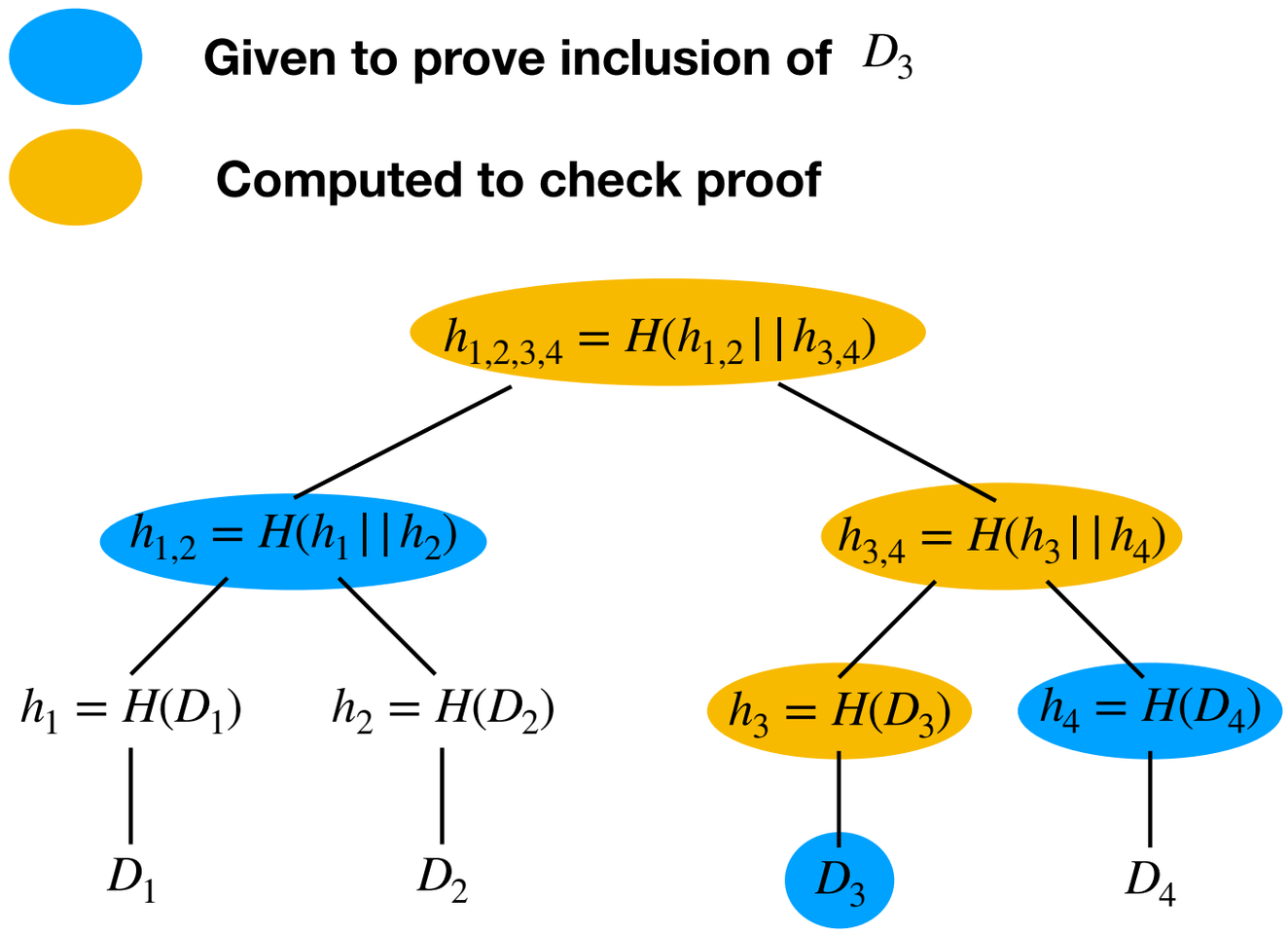


Merkle trees

Proofs

Data items: $D_1, D_2, D_3, D_4, \ldots$

- `datahash` $= h$



Merkle trees

Proofs

Data items: $D_1, D_2, D_3, D_4, \ldots$

- `datahash` $= h$

What if we have only 5 data items?

- **Duplicate D_5**
- **Add default element**