guardtime

What are hash trees and why you should care

Ahto Truu

guardtime

Cryptographic hash functions

A hash function takes arbitrary-sized data as input and generates a unique fixed-size bit sequence as output.

The output is known as the hash value, message digest, or digital fingerprint of the input.

A cryptographically secure hash function is one-way.

guardtime

Hashing and digital signatures I

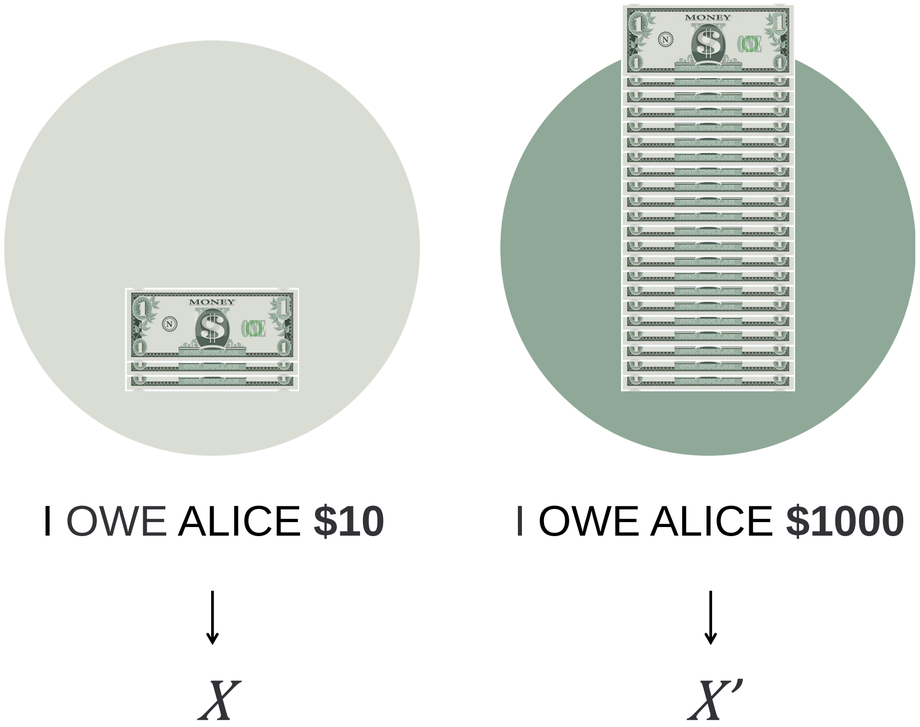
Bob

- creates a contract X,
- signs it via h(X),
- gives to Alice.

Alice

- modifies X to X’
  with h(X’) = h(X),
- claims Bob signed X’.

Forgery after signing.



guardtime

Hashing and digital signatures II

Alice

- creates contracts X and X’ with h(X) = h(X’).

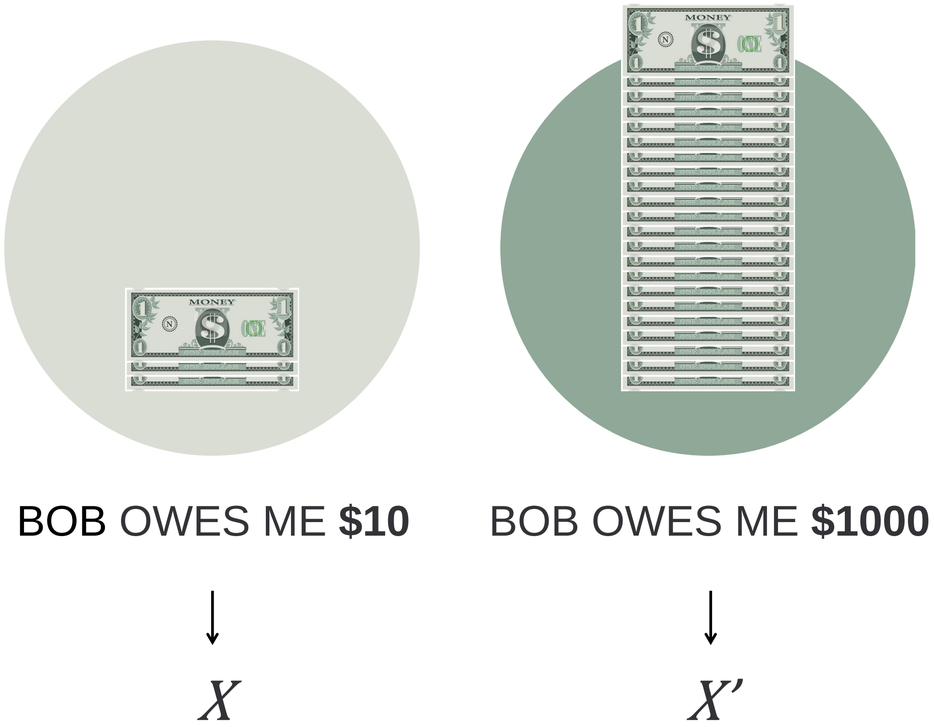
Bob

- signs X via h(X).

Alice

- claims Bob signed X’.

Forgery before signing.



Classical security properties

A secure hash function has several properties with non-obvious relationships.

Different applications may rely on different properties.

1. Pre-image resistance (one-wayness)
   Hard to find a message matching a given hash value.
2. Second pre-image resistance
   Hard to modify a message without changing its hash value.
3. Collision resistance (collision-freeness)
   Hard to find different messages with the same hash.

Collision resistance is the strongest of the three, implies the others.

CRC32 does NOT meet any of the three requirements!
It is intended to guard against accidental errors, not malicious attacks.

Same for general hash functions in most standard libraries!

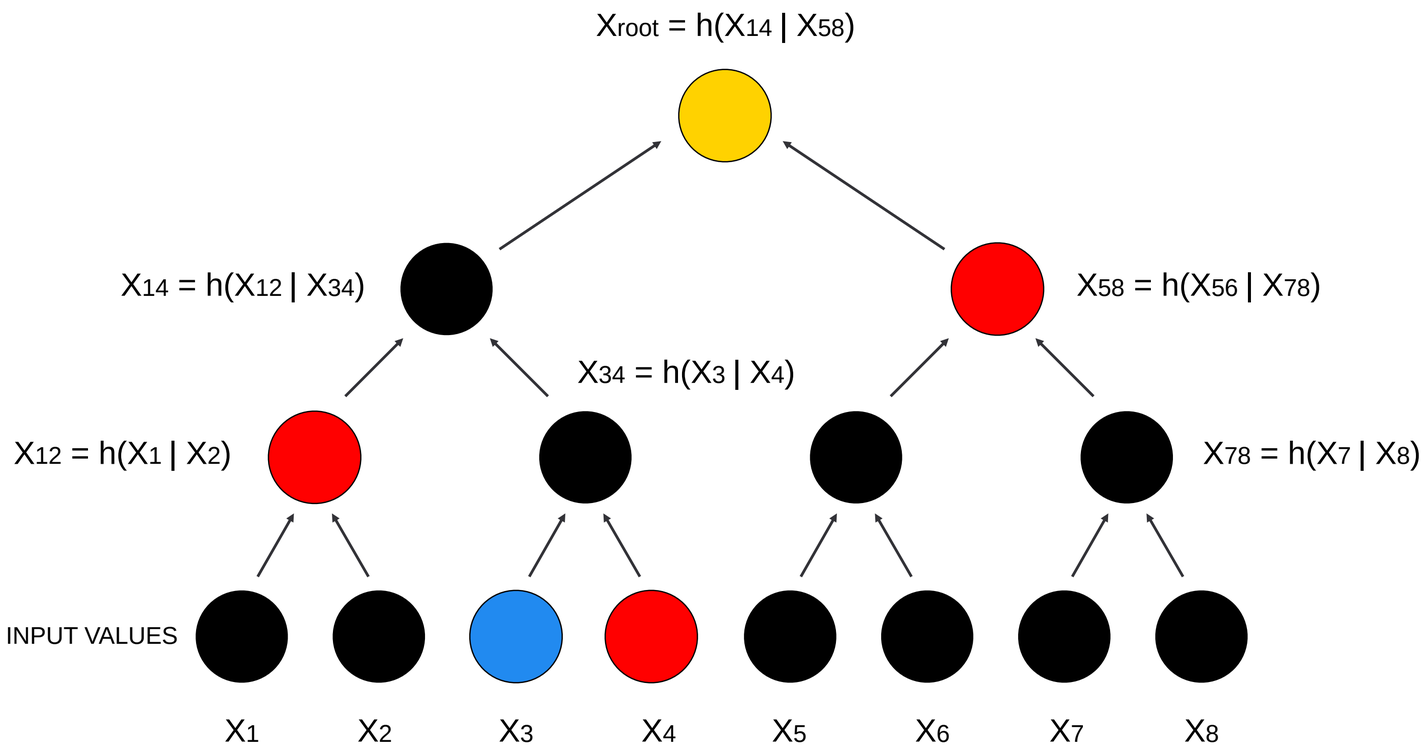
guardtime

Aggregating hash values with hash trees

A hash tree takes hash values as inputs and, via repeated hash function application, generates a single root hash value.

On the figure:

- x_1 to x_8 are the input hash values
- h(x | y) is the hash value of the concatenation of x and y



guardtime

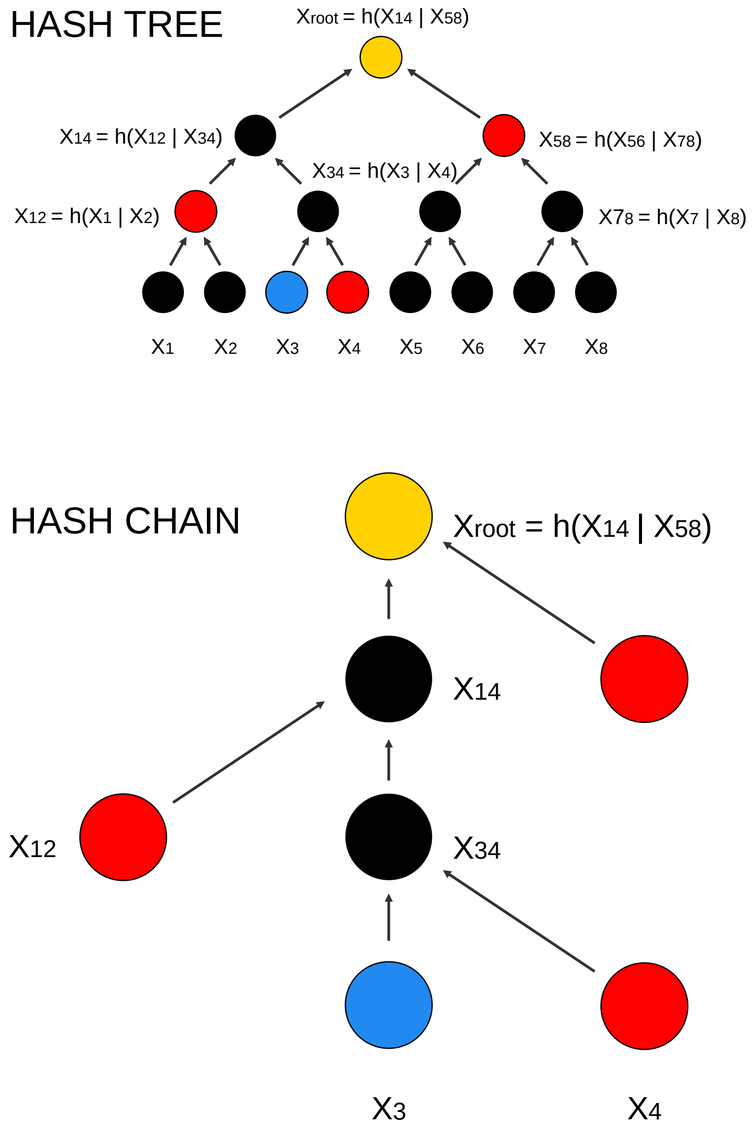
Hash chains as membership proofs

A hash chain contains all the information needed to regenerate the root hash value from any particular leaf of the tree.

Each leaf node in a hash tree with N leaves needs $\log_2(N)$ additional hash values to regenerate the root.

In the example, if the owner of x_3 knows $\{x_4, x_{12}, x_{58}\}$ then he can re-create the root hash value, thus proving that x_3 participated in the original computation that led to it.

If the data in question still hashes to x_3, then it must have been an input to the original tree.



guardtime

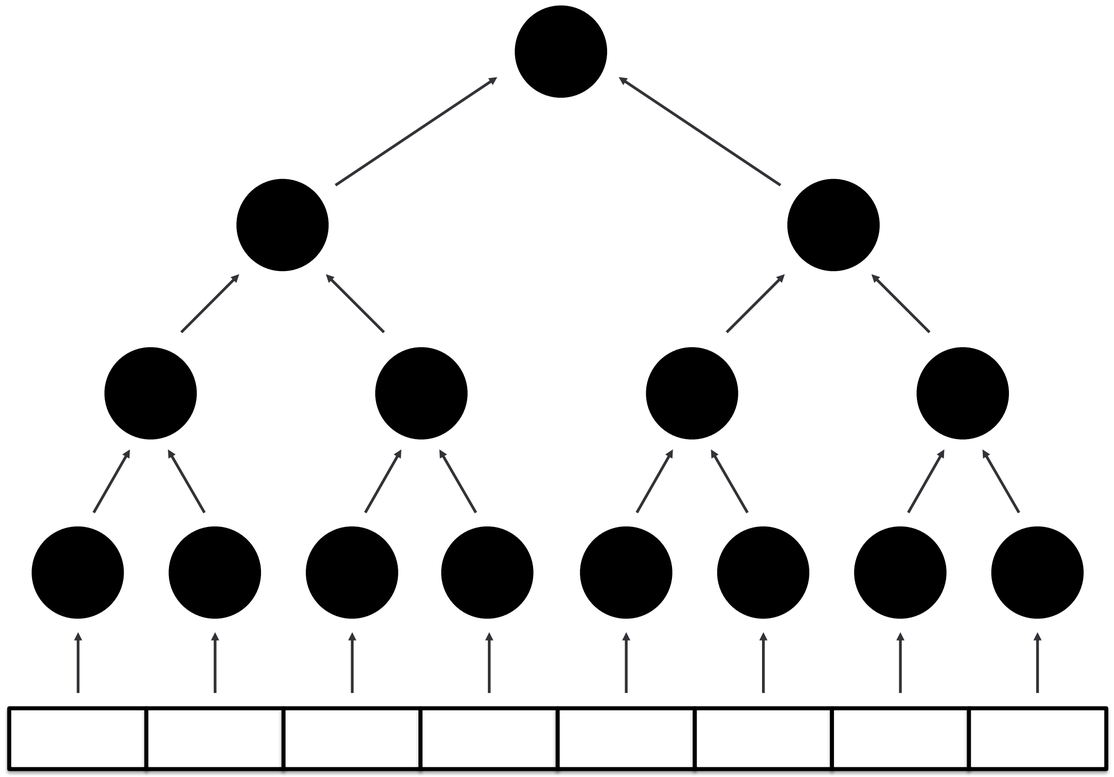
Authenticated data structures

Data structures

- Operations carried out by an untrusted prover.
- Results can be efficiently verified.

Example: BitTorrent

- File hashed in chunks.
- Root hash in the torrent file.
- Each chunk comes with chain linking it to the root.



guardtime

Blockchain as append-only log

Each record contains the hash of the previous one (in addition to the new data).

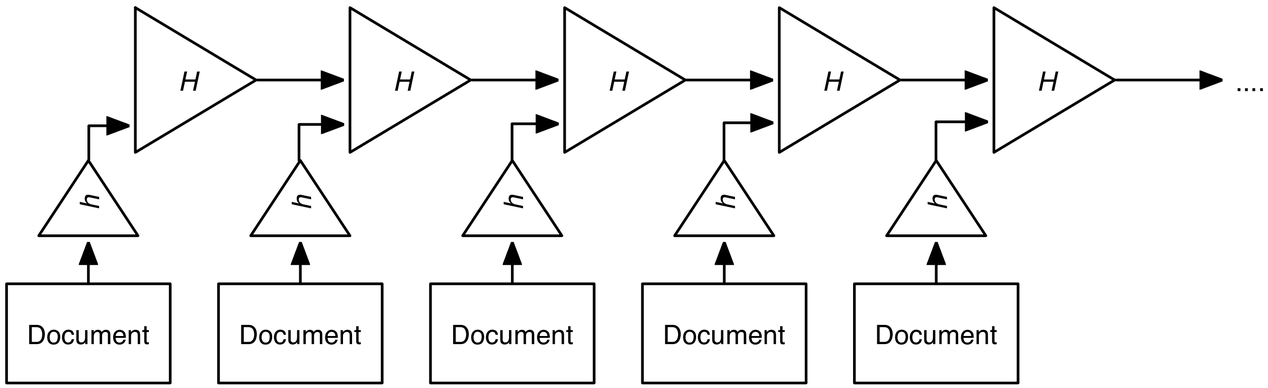
Last record “seals” the whole chain.

Request: Append X.

Response: New “head hash”.

Client only keeps the “head hash”.

Can easily verify the new “head hash” based on the old one and appended data.



guardtime

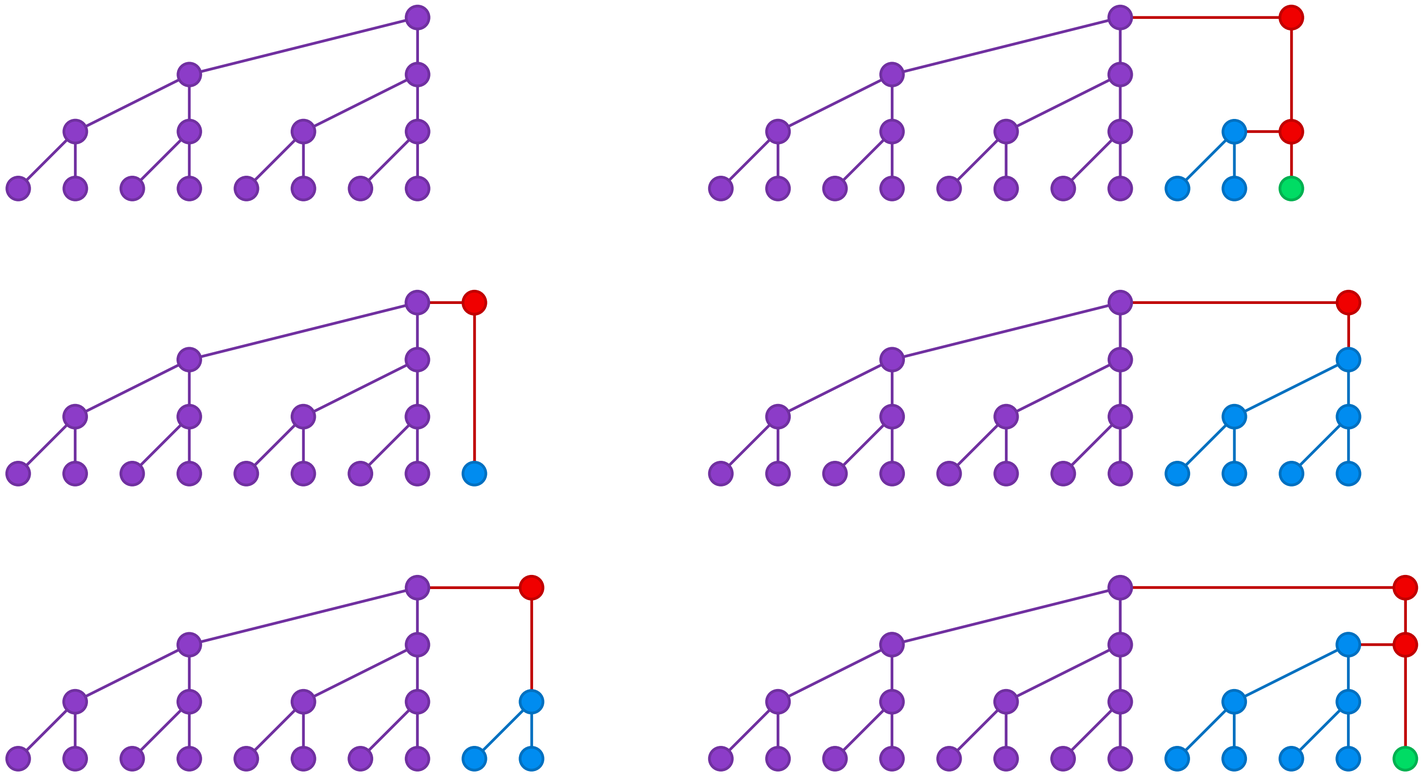
Hash trees for append-only logs

Request: Append X.

Response: Chain from added leaf to the new root.

The chain has enough information to check against the old root.

Membership of an entry verified like with regular hash tree.



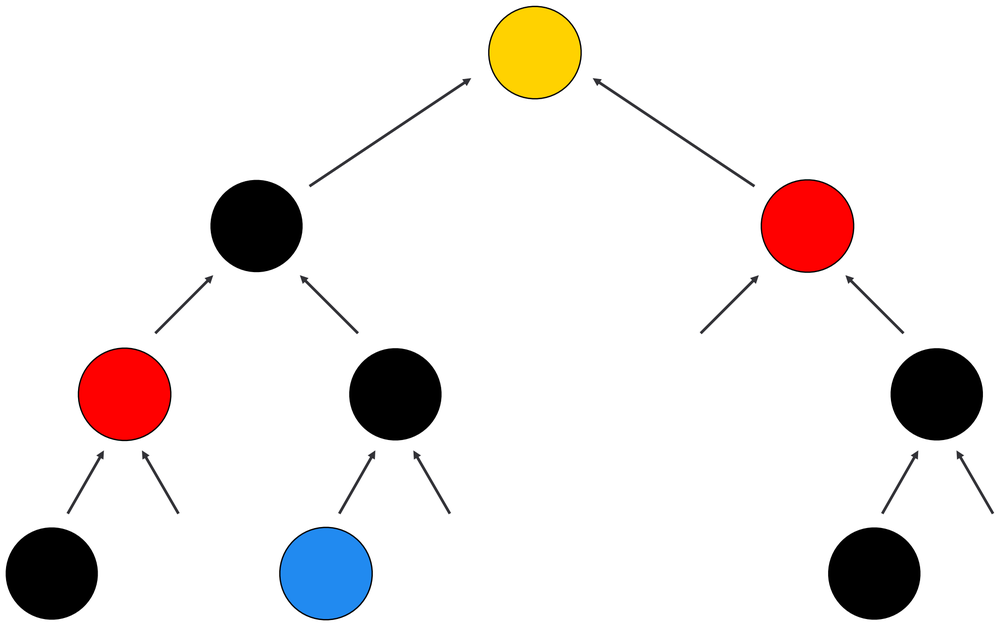
guardtime

Sparse hash trees for authenticated dictionaries

- Virtual tree with a level for each bit of hash function output.
- Each bit of hash of key gives a left or right direction.
- Corresponding leaf contains the hash of the value.

The chain linking the new value to the updated root contains enough information to check the old value against the previous root.

Allows proofs of both inclusion and omission (absence of a key).



guardtime

Log-backed authenticated dictionaries

- Dictionary on sparse hash tree.
- Mutation log based on canonical hash tree.
- Hash of the two root hashes locks complete history.
- Client authentication of latest update operation as before.
- Authentication of element lookup as before.
- Option to implement persistent dictionary using “copy-on-write”.
- Correctness of any individual operation can be verified based on old root, two hash chains, new root.
- Queries against any historical state of the dictionary.

Generic authenticated data structures

- Authenticated Data Structures, Generically
- Andrew Miller, Michael Hicks, Jonathan Katz, Elaine Shi
- POPL 2014

- Extension to ML-like functional language.
- One source compiles to client and server implementations.
- If it compiles, it’s guaranteed to be secure.

- Test implementations of naive binary search trees, red-black trees, skip lists, etc.

guardtime

Questions?
Comments?
Heckles?

Ahto Truu
ahto.truu@guardtime.com