Merkle tree

Gianluca Dini
Dept. of Ingegneria dell'Informazione
University of Pisa
Email: gianluca.dini@unipi.it
Version: 07/04/2025

1

Brief history

- Ralph Merkle patented Merkle Trees in 1979
- Merkle published the paper in 1987
  – R.Merkle. A digital signature based on a conventional encryption function. CRYPTO 1987.
- Patent expired in 2002

2

Example: executable integrity [→]

- Executable X stored on disk as a list of blocks $\{x_1, x_2, \ldots, x_r\}$
- OS needs to verify X integrity before execution

3

Example: executable integrity [→]

- Option 1
  - OS stores t = H(X) on read-only memory(*)
  - OS checks whether t == H(X) before exection
  - Drawback: hashing the whole executable may slow down executable launching
  - (*) Read-only memory implementation
    - Separate system that provides data to requestors
    - Digital signature with private key offline either by digitally sign hash or by HW trick that protects certain memory area.

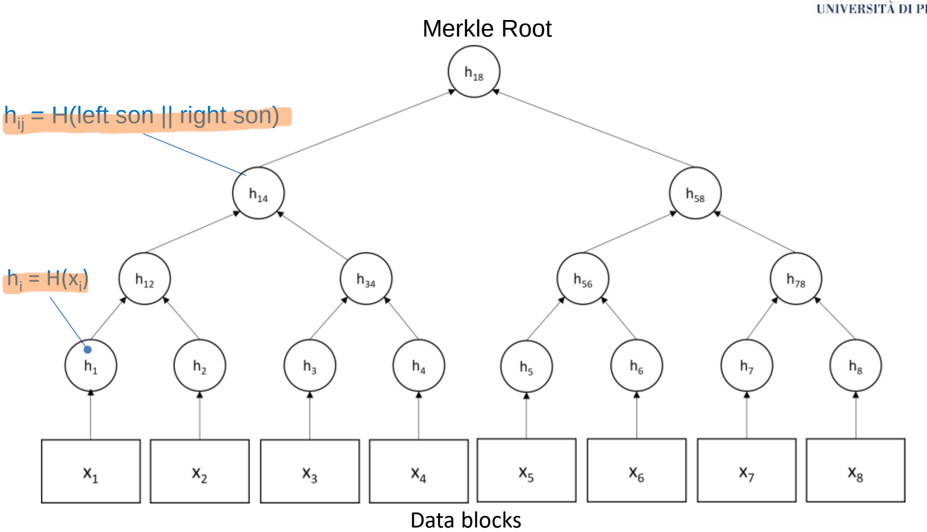
4

Example: executable integrity [↓]

- Option 2
  - For each block x_i, OS stores $t_i = H(x_i)$ on read-only memory
  - OS checks whether $t_i == H(x_i)$ when execution moves to x_i
  - Drawback: storage overhead to store t_i's
- Can we do any better?

5

Merkle Tree (1979)

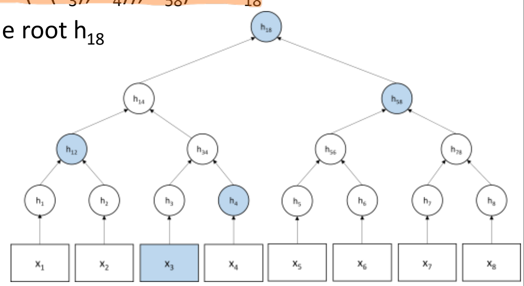


6 Assumption: tree is complete, otherwise you add leaves

Proving membership

UNIVERSITÀ DI PISA

- Verify whether x_3 (contents and position) belongs to the data set
  - List of hashes: $\pi_3 = \langle h_4, h_{12}, h_{58}, h_{18} \rangle$ (**Merkle proof**)
  - Verification algorithm
    - Check whether $H(H(h_{12}, H(H(x_3), h_4)), h_{58}) == h_{18}$
    - Verify authenticity of the root h_{18}
- Complexity O(log n), with n = #blocks
- The tree contains 2*n* – 1 nodes (hashes)



apr. '25 Merkle Tree 7

7

- It is log n in storage and time. Block needs to be stored with merkle proof.

Properties

UNIVERSITÀ DI PISA

- MT (or hash tree) allows efficient and secure verification of the contents of large data structures
- The root must be trusted : only thing I have to protect
  - Digitally signed
  - Maintained on a trusted source/storage
- Verifying whether a leaf node is part of the MT requires computing a #hashes proportional to the logarithm of the #leaves
  - O(log n), with n the number of leaves (blocks)
- MT does not require online secrets

apr. '25 Merkle Tree 8

8

Proving multiple membership

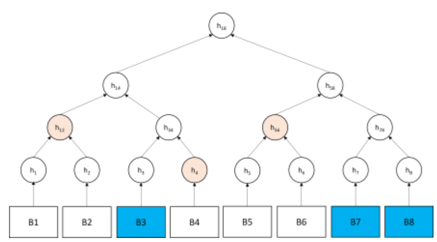
UNIVERSITÀ DI PISA

- Proving membership of multiple (L) elements
  - **Trivial solution:** L proofs ➔ L × $\log_2 n$ hashes
  - **Intuition**: Many proofs overlap and thus can be shrinked

Proving multiple membership

UNIVERSITÀ DI PISA

- The proof
  - T set of blocks; L ⊆ T; | L | = r
  - S = T – L
  - W = cover(S)
  - Merkle proof = hash values corresponding to *W*
  - W = cover(S) if every leaf in S is descendant of some node in W, and leaves outside S are not
- At most $r \cdot \log_2(n/r)$ hashes



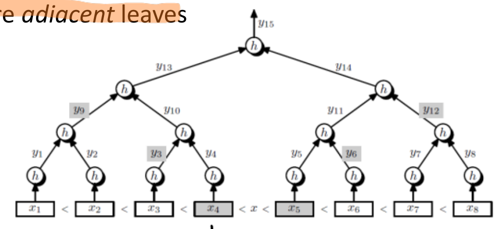
T = B3, B7, B8
Merkle proof: h_{12}, h_4, h_{56}

↓ I need specific blocks as proofs.

Merkle Proof can be easily determined: orange nodes are nodes that cover the whole blocks without covering blue blocks.

Proving non-membership

- Proving non-membership
  - Sorted Merkle Tree: $x_1 < x_2 < \ldots < x_n$ (e.g., set of TX's)
  - Prover provides
    - Proof π_i and proof π_j (merkle Proof of adjacent nodes)
  - Verifier
    - Verifies π_i and π_j
    - Verifies that x_i and x_j are *adjacent* leaves
    - Verifies that $x_i < x < x_j$



Imagine # of transactions

apr. '25 Merkle Tree 11

11

The values must be sorted in one order, and I have to find adjacent values that contain X, and verify if they are part of the tree and are adjacent.

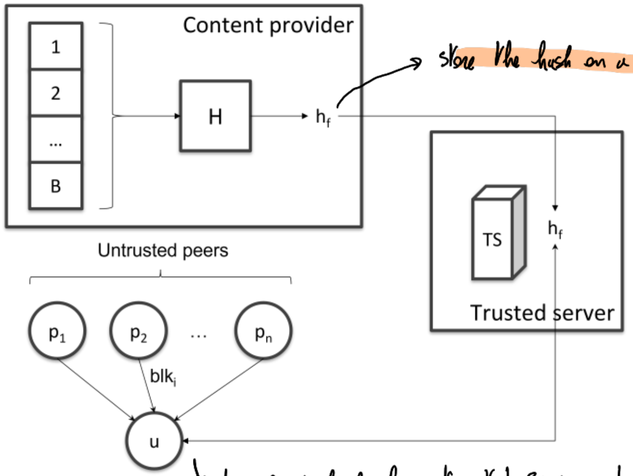
Merkle Tree - applications

- File systems
  - IPFS, Btrfs, ZFS
- Content distribution protocols
  - Dat, Apache Wave
- Distributed revision control system
  - Git, Mercurial
- Blockchain
  - Bitcoin, Ethereum
- Backup Systems
  - Zeronet
- P2P networks
  - Torrent
- NoSQL systems
  - Apache Cassandra, Riak, Dynamo
- Certificate Transparency framework

apr. '25 Merkle Tree 13

13

Content distribution [→]



store the hash on a trusted server

download hash from trusted server and compare with your hash.

Content Distribution [→]

- How does the user know that the information that (s)he is getting from some untrusted peer is genuine and hasn't been tampered with (or corrupted)?

Content Distribution [→]

- Solution no. 1 (shown in the slide)
  - Trusted Server stores h_f
- Verification
  - Upon receiving **all** blocks $\{x_i, 1 \leq i \leq B\}$, compute $h_f' = H(x_1 \mid x_2 \mid \dots \mid x_n)$.
  - Return $(h_f' == h_f)$
- Drawback
  - Check upon completion (possibly long delay)
  - Not possible to determine corrupted/compromised blocks

apr. '25 Merkle Tree 16

16

Content Distribution [→]

- Solution n.2
  - Trusted Server stores $\langle h_f, h_1, h_2, \dots, h_B \rangle$ with $h_i = H(x_i)$, $1 \leq i \leq B$
  - Number of hashes B = sizeof(file)/sizeof(block)
    - Torrent: block size is 16 kbytes
- User Verification
  - The user can verify each block
- Drawback
  - Increase storage/bandwidth overhead

apr. '25 Merkle Tree 17

17

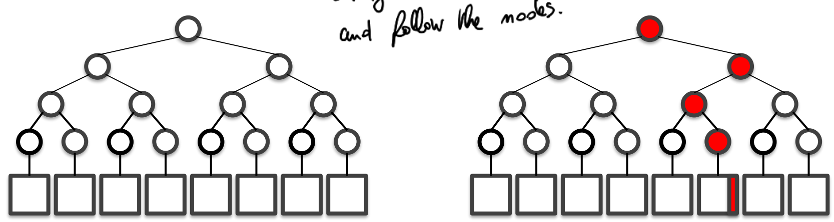
Content Distribution [↓]

- Solution n.3: Merkle Tree
  - Trusted Server stores the root of the Merkle Tree
  - Each peer stores
    - A subset of the blocks $\{x_i\}$;
    - For each block x_i, a peer stores the corresponding Merkle proof π_i
  - User Verification
    - Upon downloading a block x_i, the user verifies it using π_i and the tree root

apr. '25 Merkle Tree 18

18

File comparison : you can check wether file has changed and where : you compare root and follow the nodes.



- File F gets modified in a block x_i
- Comparing files takes is O(B)
- Comparing MTs is O(log B) , REPEAT ∀B?

apr. '25 Merkle Tree 19

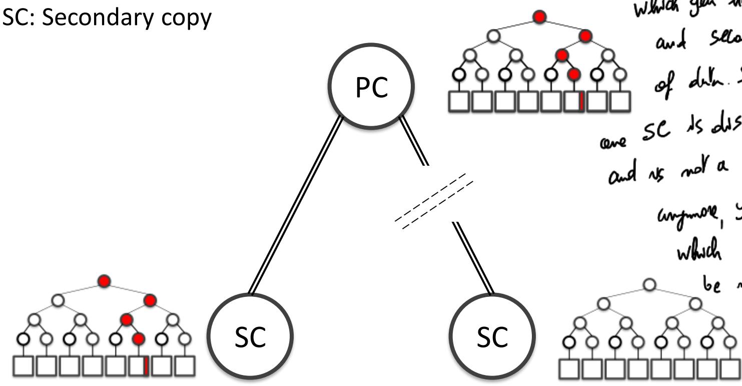
19

Replication

UNIVERSITÀ DI PISA

PC: Primary copy
SC: Secondary copy

Useful in replication systems in which you have primary and secondary copy of data. Suppose that one SC is disconnected and is not a perfect copy anymore, you can use which block has to be modified.



20

Replication

UNIVERSITÀ DI PISA

- How can the primary replica determine whether a disconnected secondary replica has to be updated?
- Upon reconnection, the primary replica compares its MT with the secondary replica's MT in order to determine the modified blocks

21