

Merkle tree

Gianluca Dini

Dept. of Ingegneria dell'Informazione

University of Pisa

Email: gianluca.dini@unipi.it

Version: 2024-04-16

1

Example: executable integrity [→]



UNIVERSITÀ DI PISA

- Executable X stored on disk as a list of blocks x_1, x_2, \dots, x_r
- OS needs to verify X integrity before execution
- Option 1
 - OS stores $t = H(X)$ on read-only memory^(*)
 - OS checks whether $t == H(X)$ before execution
 - Drawback: hashing may slow down executable launching
 - (*) Read-only memory implementation
 - Separate system that provides data to requestors
 - Digital signature with private key offline


apr. '24

Merkle Tree

2

2

Example: executable integrity [↓]



UNIVERSITÀ DI PISA

- 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?


apr. '24

Merkle Tree

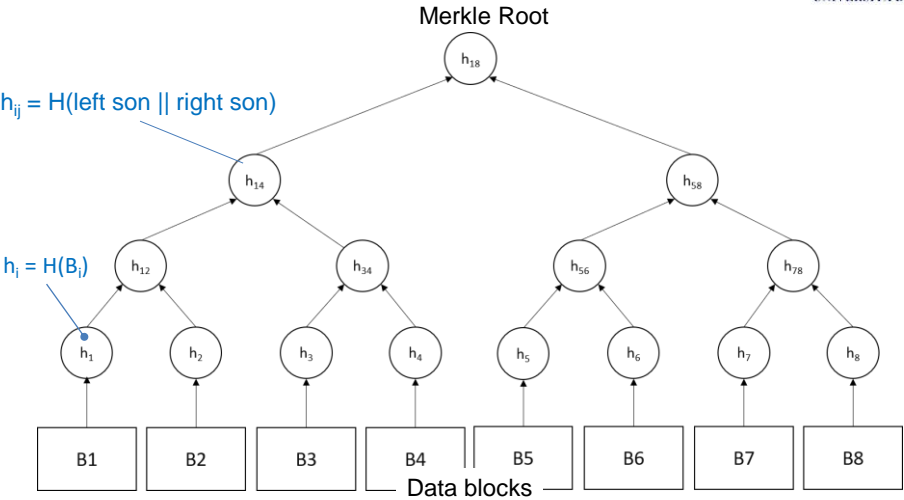
3

3

Merkle Tree (1979)



UNIVERSITÀ DI PISA



Merkle Root

h_{18}

h_{14}

h_{58}

h_{12}

h_{34}

h_{56}

h_{78}

h_1

h_2

h_3

h_4

h_5

h_6

h_7

h_8

B1

B2

B3

B4

B5

B6

B7

B8

Data blocks

$h_{ij} = H(\text{left son} || \text{right son})$

$h_i = H(B_i)$


apr. '24

Merkle Tree

4

4

Proving membership

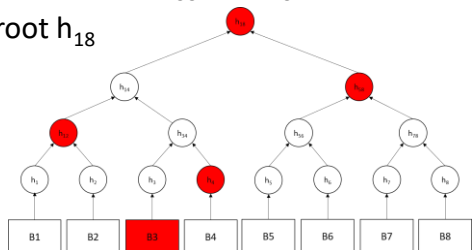


UNIVERSITÀ DI PISA

Contents and position

B3

- Verify whether B3 belongs to the data set
- List of hashes (Merkle proof): $\pi = \langle h_4, h_{12}, h_{58}, h_{18} \rangle$
- Verification algorithm
 - Check whether $H(H(h_{12}, H(H(B3), h_4)), h_{58}) == h_{18}$
 - Verify authenticity of the root h_{18}
- Complexity $O(\log n)$, with $n = \text{\#blocks}$
- The tree contains $2n - 1$ nodes (hashes)




apr. '24

Merkle Tree

5

5

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
 - 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. '24

Merkle Tree

6

6

Proving multiple/non-membership



UNIVERSITÀ DI PISA

- Proving membership of multiple elements
 - L elements
 - $L \times \log_2 n$ hashes
 - Many proofs overlap and thus can be shrinked
- Proving non-membership
 - Sorted Merkle tree hash
 - $B_1 < B_2 < \dots < B_n$ (e.g., set of TX's)
 - Proof that B does not belong to the data set
 - Determine $B_i < B < B_{i+1}$, with B_i and B_{i+1} *adjacent leaves*
 - Provide $\text{proof}(B_i)$ and $\text{proof}(B_{i+1})$

apr. '24

Merkle Tree

7

7

Merkle Tree - applications



UNIVERSITÀ DI PISA

- | | |
|--|---|
| <ul style="list-style-type: none"> • File systems <ul style="list-style-type: none"> – IPFS, Btrfs, ZFS • Content distribution protocols <ul style="list-style-type: none"> – Dat, Apache Wave • Distributed revision control system <ul style="list-style-type: none"> – Git, Mercurial • Blockchain <ul style="list-style-type: none"> – Bitcoin, Ethereum | <ul style="list-style-type: none"> • Backup Systems <ul style="list-style-type: none"> – Zeronet • P2P networks <ul style="list-style-type: none"> – Torrent • NoSQL systems <ul style="list-style-type: none"> – Apache Cassandra, Riak, Dynamo • Certificate Transparency framework |
|--|---|

apr. '24

Merkle Tree

8

8

Content distribution [→]

The diagram illustrates the content distribution process. A 'Content provider' box contains a vertical list of content blocks (1, 2, ..., B) and a hash function 'H'. An arrow points from the list to 'H', which outputs a fingerprint 'h_f'. This fingerprint is then sent to a 'Trusted server' box, which contains a 'TS' (Trusted Server) block. The 'Trusted server' also outputs 'h_f'. Below the 'Content provider' is a group of 'Untrusted peers' represented by circles labeled p_1, p_2, ..., p_n. These peers are connected to a user 'u' (circle). One peer p_i is specifically labeled with 'blk_i' pointing to 'u'. The 'Trusted server' also has a direct arrow pointing to 'u'. The diagram shows how the user 'u' receives content from untrusted peers and verifies it against the fingerprint 'h_f' provided by the trusted server.

april '24

Merkle Tree

9

Content Distribution [→]

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

april '24

Merkle Tree

10

Content Distribution [→]



UNIVERSITÀ DI PISA

- **Solution no. 1 (shown in the slide)**
 - Trusted Server stores h_f
- **Verification**
 - Upon receiving *all* blocks $\{\text{blk}_i, 1 \leq i \leq B\}$, compute $h_f' = H(\text{blk}_1 \mid \text{blk}_2 \mid \dots \mid \text{blk}_n)$.
 - Return $(h_f' == h_f)$
- **Drawback**
 - Check upon completion (possibly long delay)
 - Not possible to determine corrupted/compromised blocks

apr. '24

Merkle Tree

11

11

Content Distribution [→]



UNIVERSITÀ DI PISA

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


apr. '24

Merkle Tree

12

12

Content Distribution [↓]



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

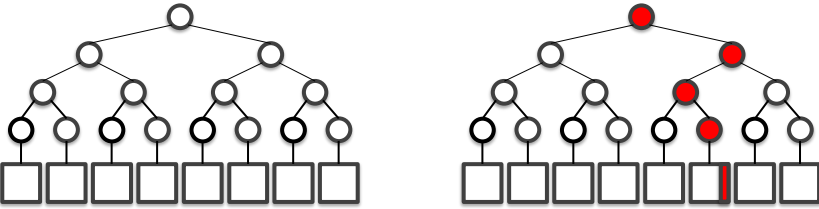

apr. '24

Merkle Tree

13

13

File comparison



- File F gets modified in a block blk_i
- Comparing files takes is $O(B)$
- Comparing MTs is $O(\log B)$


apr. '24

Merkle Tree

14

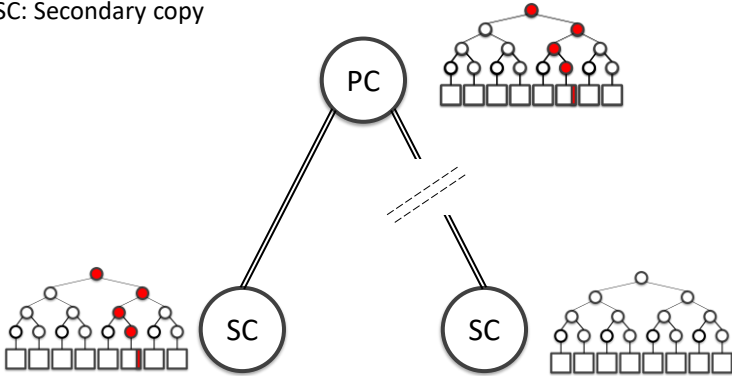
14

Replication



UNIVERSITÀ DI PISA

PC: Primary copy
SC: Secondary copy




apr. '24

Merkle Tree

15

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

apr. '24

Merkle Tree

16

Brief history



UNIVERSITÀ DI PISA

- 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

apr. '24

Merkle Tree

17

17