# 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 $x_1$, $x_2$,…, $x_r$
- 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 exection
  - 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

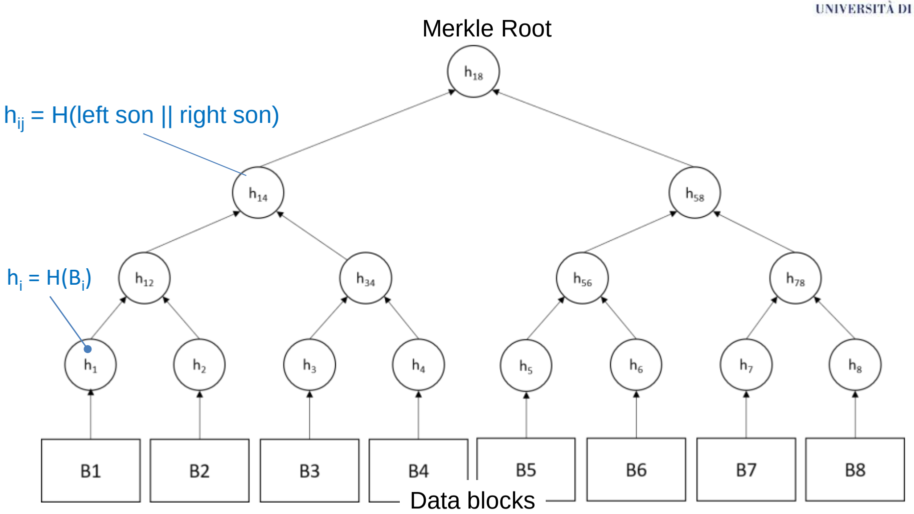
# Example: executable integrity [↓]

- 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 exection moves to $x_i$
  - Drawback: storage overhead to store $t_i$'s
- Can we do any better?





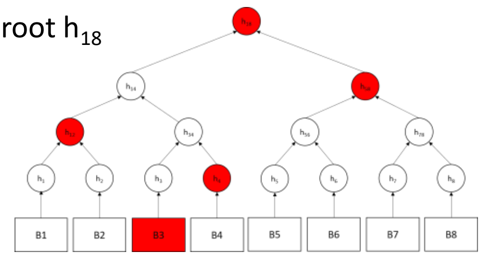
# Merkle Tree (1979)



# Proving membership

Contents and position

- Verify whether B3 belongs to the data set
- List of hashes (Merkle proof): $\pi$ = <$h_4$, $h_{12}$, $h_{58}$, $h_{18}$>
- 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 = #blocks
- The tree contains 2n – 1 nodes (hashes)






5

# Properties

- 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



6

# Proving multiple/non-membership

- 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
  - B1 < B2 < … < Bn (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 proof($B_i$) and proof($B_{i+1}$)

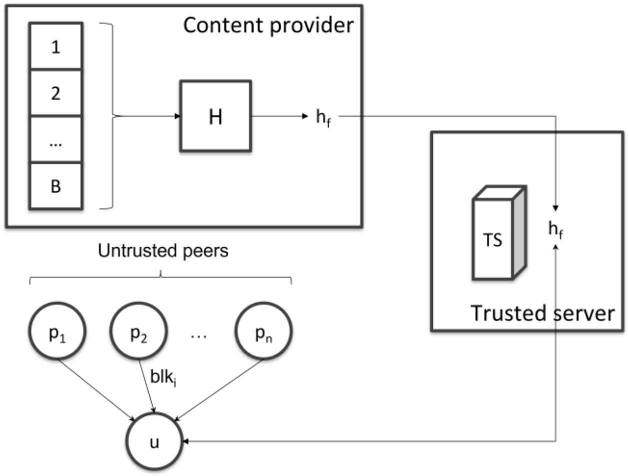




# Merkle Tree - applications

- 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

# Content distribution [→]




apr. '24 Merkle Tree 9

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)?

apr. '24 Merkle Tree 10

10

## Content Distribution [→]

UNIVERSITÀ DI PISA

- Solution no. 1 (shown in the slide)
  - Trusted Server stores $h_f$
- Verification
  - Upon receiving *all* blocks $\{blk_i, 1 \le i \le B\}$, compute $h_f' = H(blk_1 \mid blk_2 \mid \ldots \mid blk_n)$.
  - 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 $\langle h_f, h_1, h_2, \ldots, h_B \rangle$ with $h_i = H(blk_i)$, $1 \le i \le B$
  - 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. '24 Merkle Tree 12

12

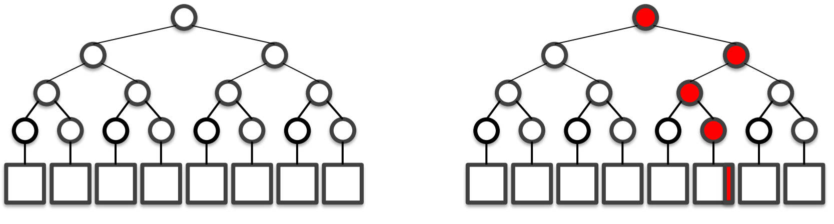
# Content Distribution [↓]

- 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





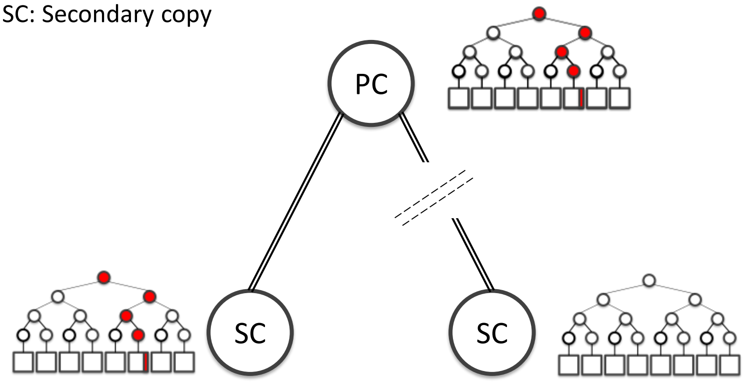
# File comparison



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

## Replication

PC: Primary copy
SC: Secondary copy






15

## Replication

- 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



16

# 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