Merkle tree

Gianluca Dini
Dept. of Ingegneria dell'Informazione
University of Pisa
Email: gianluca.dini@unipi.it
Version: 2022-04-12

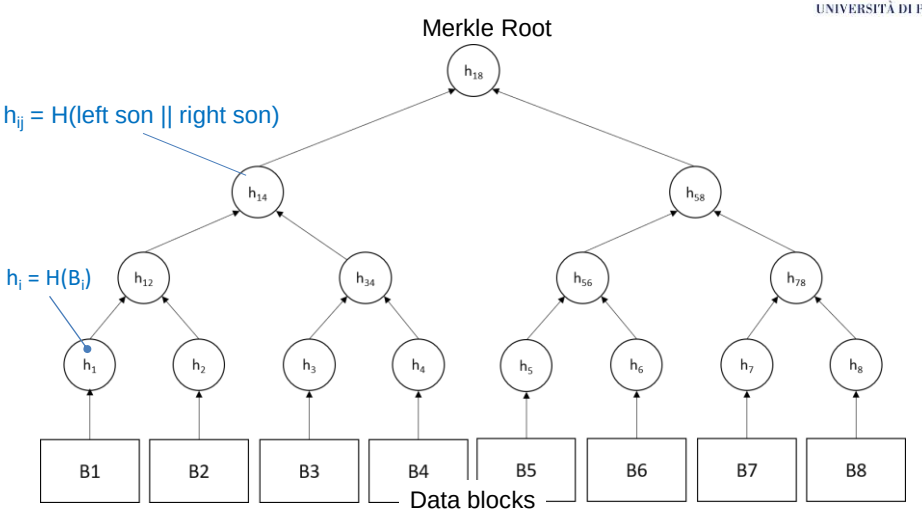
1

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

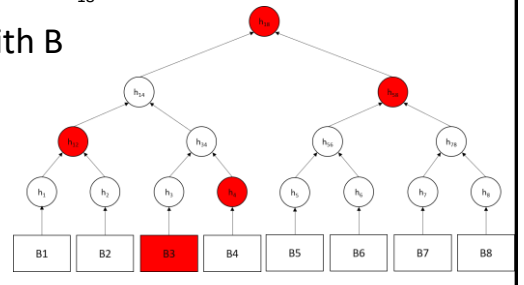
Merkle Tree (1979)



3

Verification

- Verify whether B3 belongs to the data set
  - List of hashes (proof): <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 B # of blocks



4

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 B), with B the number of leaves (blocks)

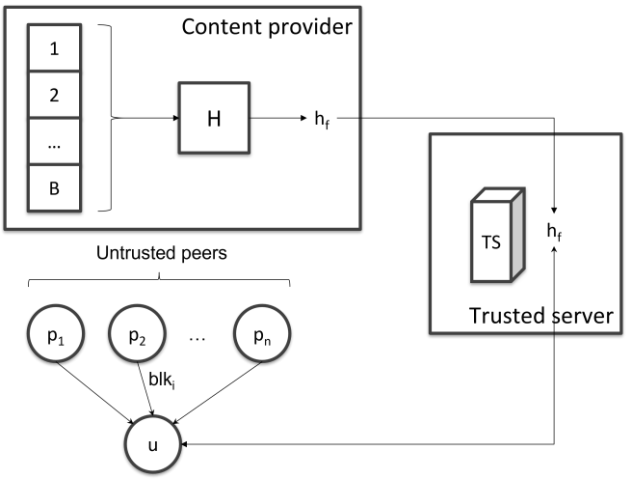
5

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

6

Distributed scenario [→]



7

Distributed scenario

- 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)?

8

Distributed scenario

- Solution no. 1 (shown in the slide)
  - Trusted Server stores h_f
- Verification
  - Upon receiving all blocks $\{blk_i, 1 \leq i \leq B\}$, compute $h_f' = H(blk_1 \,!\, blk_2 \,!\, \ldots \,|\, blk_n)$.
  - Return $(h_f' == h_f)$
- Drawback
  - Check upon completion (possibly long delay)
  - Not possible to determine corrupted/compromised blocks

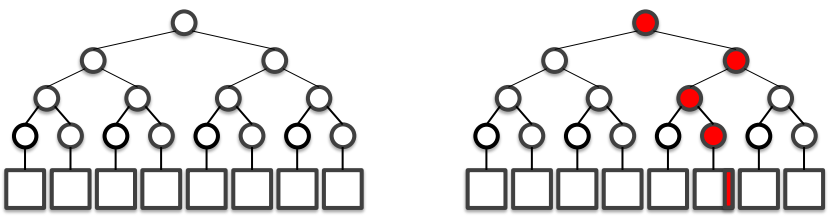
Distributed scenario

- Solution n.2
  - Trusted Server stores $\langle h_f, h_1, h_2, \ldots, h_B \rangle$ with $h_i = H(blk_i)$, $1 \leq i \leq B$
  - 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

Distributed scenario

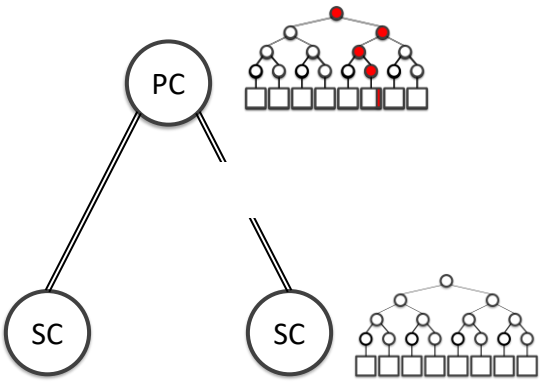
- Solution n.3: Merkle Tree
  - Trusted Server stores the root of the Merkle Tree
  - Each peer stores
    - A subset of the blocks $\{blk_i\}$;
    - For each block blk_i, $\langle blk_i, proof_i \rangle$
  - User Verification
    - Upon downloading a block blk_i, the user verifies it using $proof_i$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

File comparison



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

Replication



13

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

14