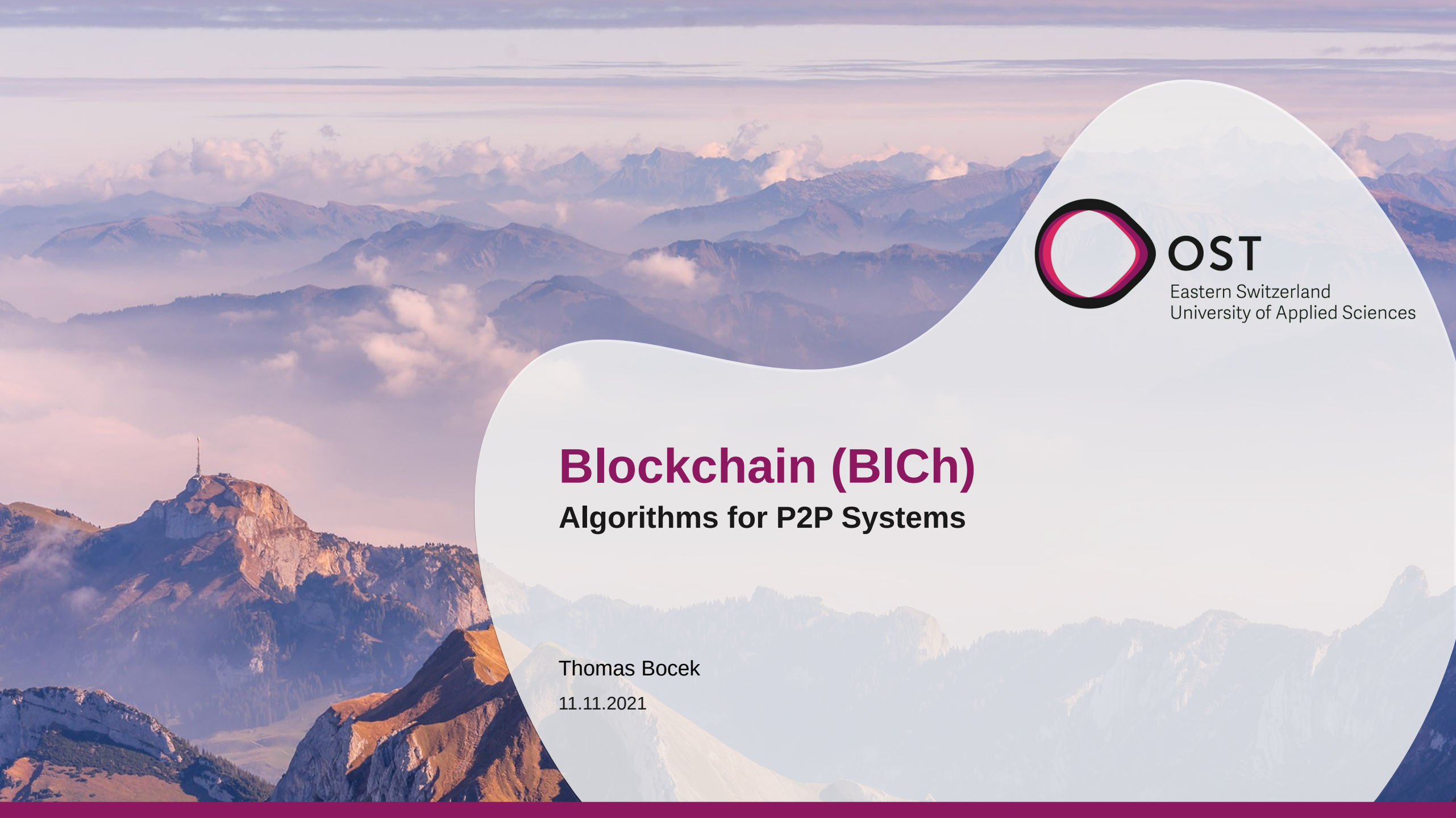


OST
Eastern Switzerland
University of Applied Sciences

Blockchain (BlCh)

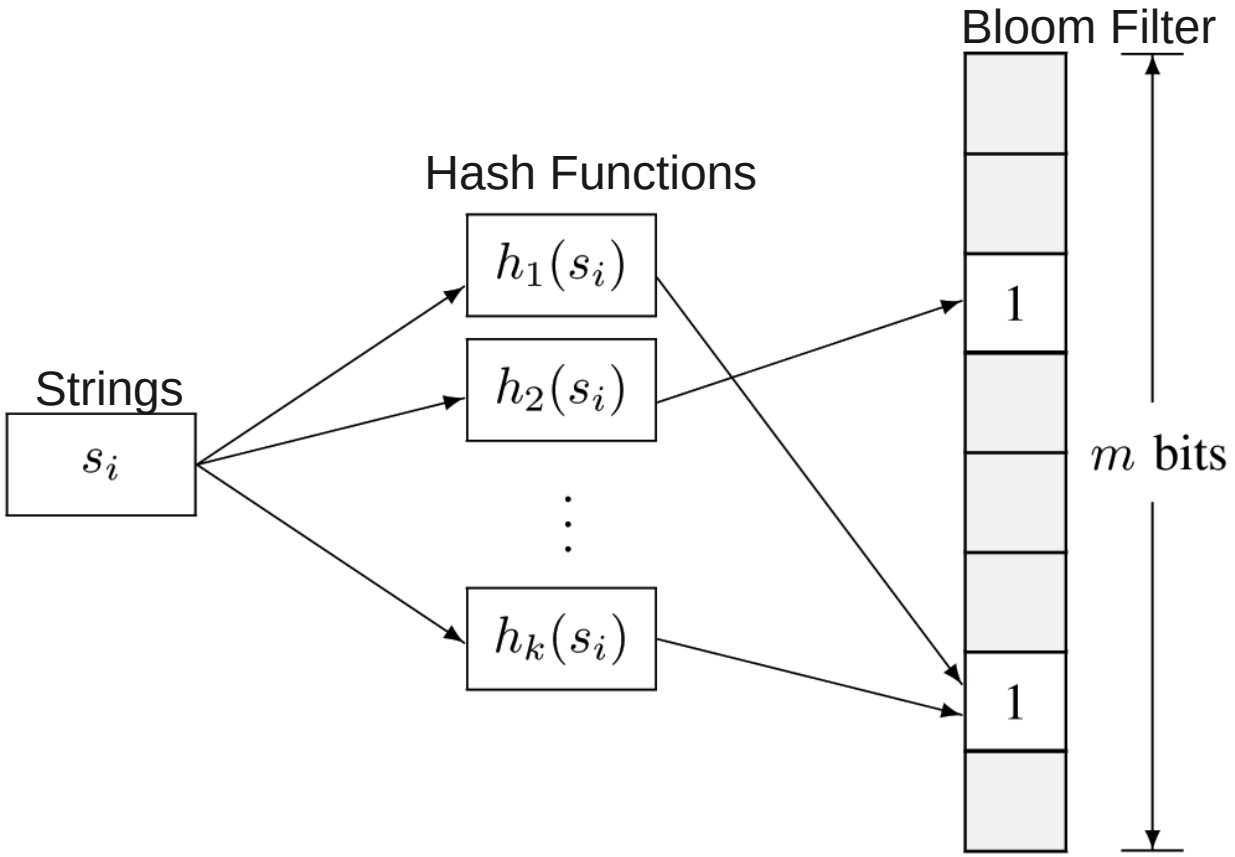
Algorithms for P2P Systems

Thomas Bocek

11.11.2021

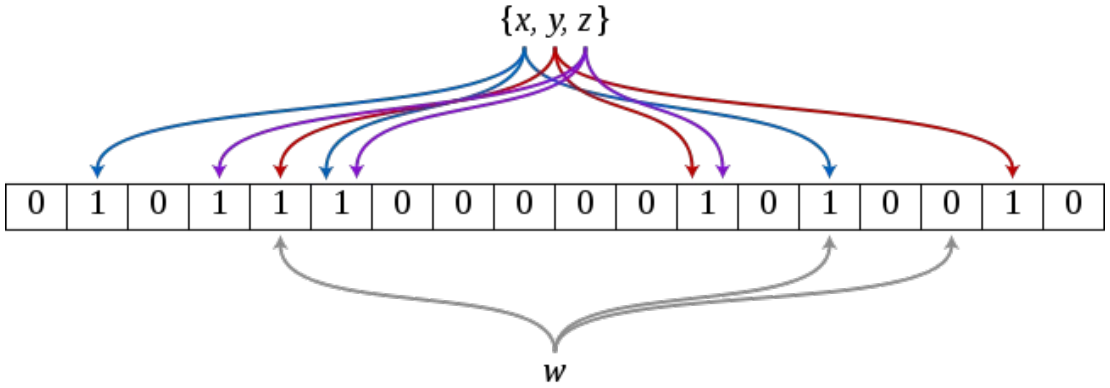
Bloom Filter

- An array of m bits, initially all bits set to 0
- A bloom filter uses k independent hash functions
  - h1, h2, …, hk with range {1, …, m}
- Each input is hashed with every hash function
  - Set the corresponding bits in the vector
- Operations
  - Insertion
    - The bit A[hi(x)] for 1 < i < k are set to 1
  - Query
    - Yes if all of the bits A[hi(x)] are 1, no otherwise
  - Deletion
    - Removing an element from this simple Bloom filter is impossible



Query of an Element, m=18, k=3

- Insert x, y, z
- Query w



http://en.wikipedia.org/wiki/Bloom_filter

- Example for False-positives
  - Insertions
    - Hash („color printer“) => (1,4,6)
    - Hash („digital camera“) => (3,4,5)
    - Bloom filter (1,3,4,5,6)
  - Query
    - Hash („heat sensor“) => (3,4,6)
    - Matches since bits 3,4,6 are all set to 1
  - Online
- False-negative
  - Query
    - Hash (“color printer”) => (1,4,6) , matches (1,3,4,5,6) → no false-negative

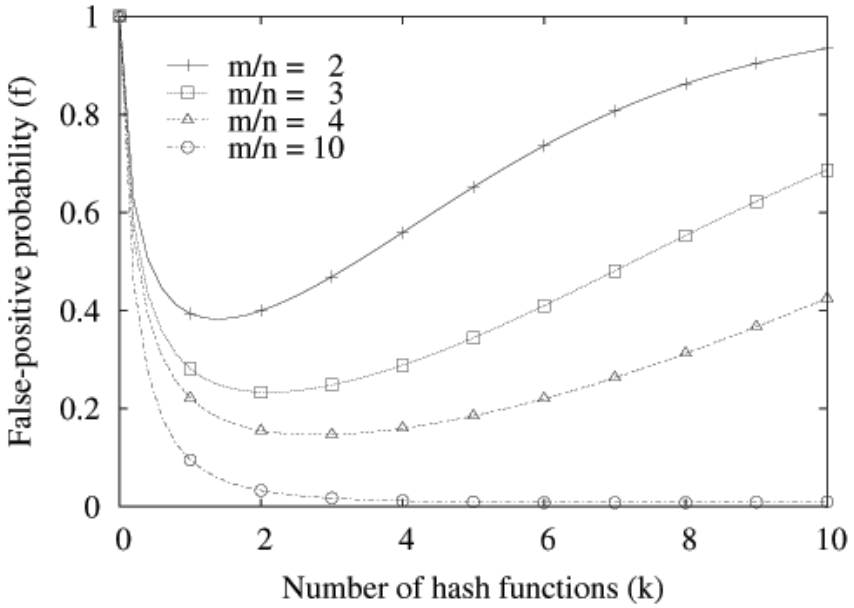
OST

Properties

- Space Efficiency
  - Any Bloom filter can represent the entire universe of elements
    - In this case, all bits are 1
- No Space Constraints
  - Add never fails
  - But false positive rate increases steadily as elements are added
- Simple Operations
  - Union of Bloom filters: bitwise OR
  - Intersection of Bloom filters: bitwise AND

- No false negative, but false positive
- False-positive probability:
  - *n* number of strings; *k* hash functions; *m*-bit vector

$$f = (1 - e^{-\frac{nk}{m}})^k$$



=> Given m/n, there is an optimal number of hash functions (opt. k = m/n ln 2) (when 50% of the bits are set)

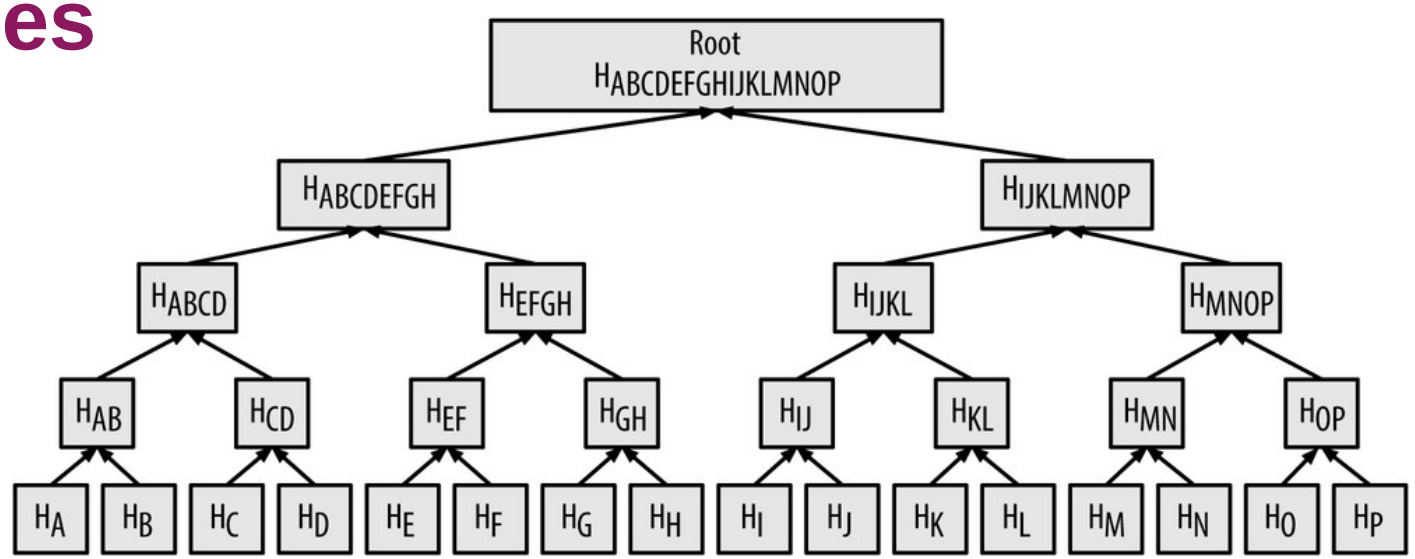
OST

Bloom Filter Variants

- Compressed Bloom Filters
  - When the filter is intended to be passed as a message
  - False-positive rate is optimized for the compressed bloom filter (uncompressed bit vector m will be larger but sparser)
  - However, compression/decompression, more memory
- Generalized Bloom Filter
  - Two type of hash functions gi (reset bits to 0) and hj (set bits to 1)
  - Start with an arbitrary vector (bits can be either 0 or 1)
  - In case of collisions between gi and hj, bit is reset to 0
  - Store more info with low false positive
  - Produces either false positives or false negatives
- Counting Bloom Filters
  - Entry in the filter not be a single bit but a counter
  - Delete operation possible (decrementing counter)
  - Variable-Increment Counting Bloom Filter
- Scalable Bloom Filter
  - Adapt dynamically to number of elements, consist of regular Bloom filters
  - “A SBF is made up of a series of one or more (plain) Bloom Filters; when filters get full due to the limit on the fill ratio, a new one is added; querying is made by testing for the presence in each filter”
- Others, e.g., Cuckoo filter
- Usage: e.g., fast search at LinkedIn

OST

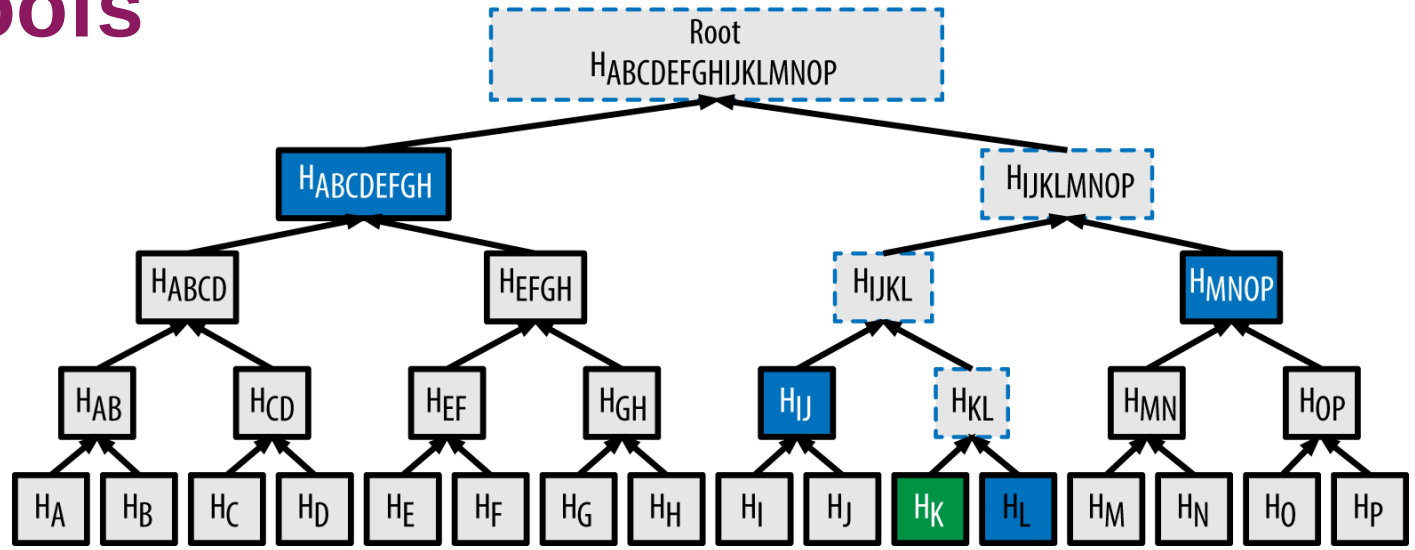
Merkle Trees



- A Merkle tree is a binary hash tree containing leaf nodes
- Constructed bottom-up, i.e.,
- Used to summarize all transactions in a block
- To prove that a specific transaction is included in a block, a node only needs to produce hashes, constituting a merkle path connecting the specific transaction to the root of the tree.

OST

Merkle Proofs

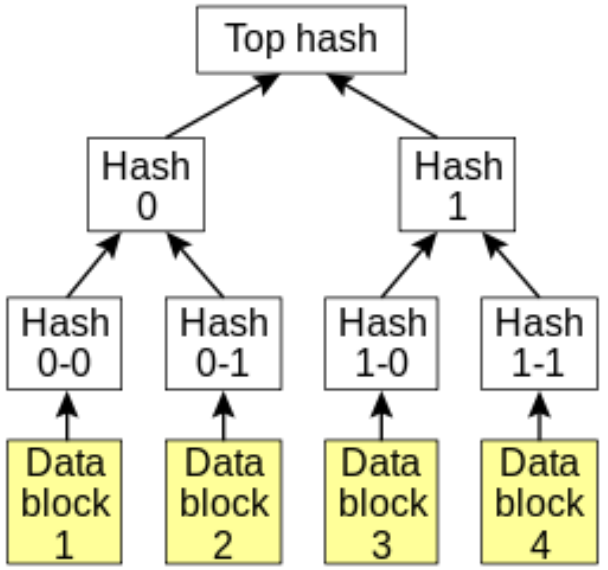


- A node can prove that transaction K is included in the block by producing a merkle path
  - $\boldsymbol{log_2 16 = 4\ hashes}$ long

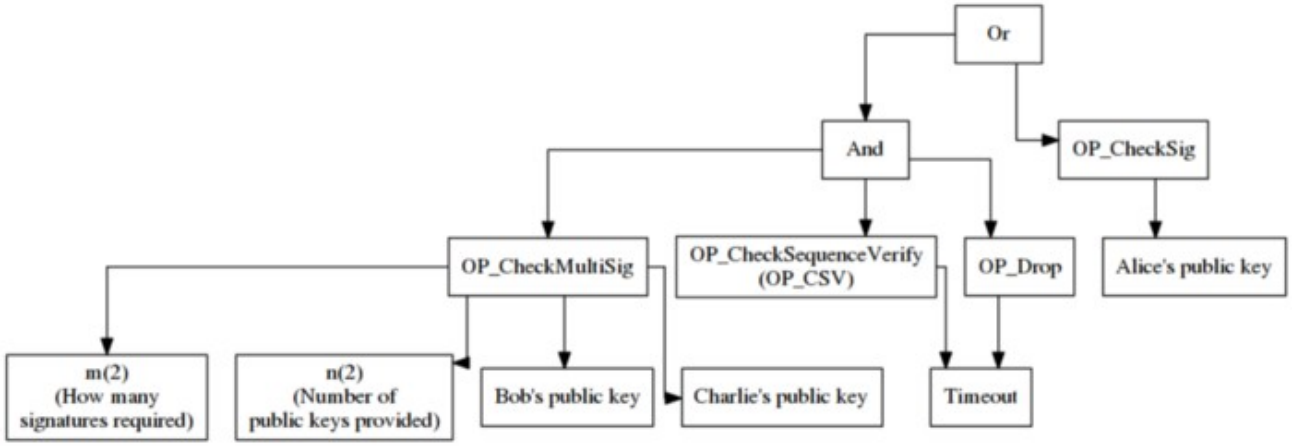
OST

BitTorrent: Mechanisms

- Magnet links
  - Magnet is URI scheme, does not point to a centralized tracker
    - No centralized tracker: pointer to DHT
    - General purpose, not only for BT
    - magnet:?xl=1000&dn=song1.mp3&xt=urn:tree:tiger:2A3B…
  - tree:tiger → Hash Tree
    - Tree of hashes (|| → concatenation)
    - hash 0 = hash(hash 0-0 || hash 0-1)
    - hash 1 = hash(hash 1-0 || hash 1-1)
    - Top hash = hash(hash 0 || hash 1)
- Merkle hash / hash tree also seen in Bitcoin blocks (transactions), MAST (Merklized Abstract Syntax Tree)



http://en.wikipedia.org/wiki/Hash_tree



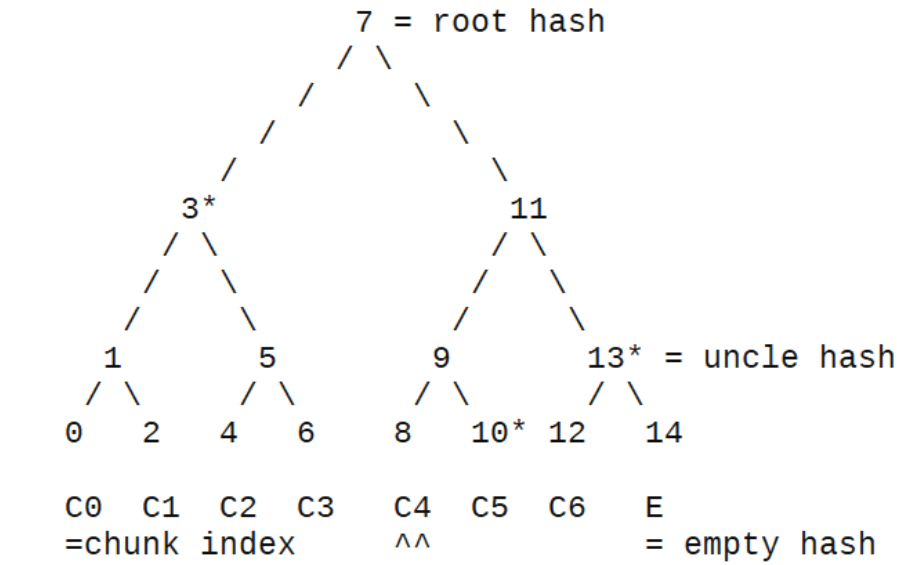
https://bitcointechtalk.com/what-is-a-bitcoin-merklized-abstract-syntax-tree-mast-33fdf2da5e2f

OST

BitTorrent: Mechanisms

- Verification
  - Peer A has top hash (root hash)
  - Peer downloads C4 from peer B
    - create hash 8
  - Need hash 10, 13, 3 (uncle hash)
    - Can be from peer B
  - With 8,10,13,3 can create root hash

    → verify this root hash
- Usage: Blockchain, P2P filesharing, git, Amazons Dynamo, ZFS



The Merkle hash tree of an interval of width W=8

http://datatracker.ietf.org/doc/draft-ietf-ppsp-peer-protocol/ Section 5.2

OST