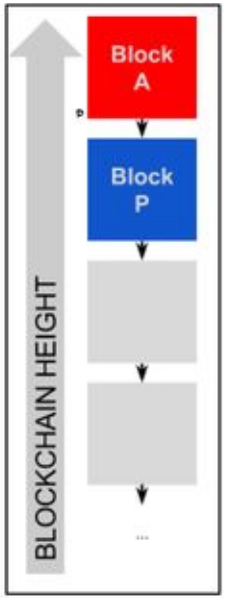
Blockchain Technology

Lecture 3: Block Structure

Blockchain (as a datastructure)

- is an ordered, back-linked list of blocks of transactions
- can be stored as a flat file, or in a simple database.
- are linked “back,” each referring to the previous block in the chain
- visualized as a vertical stack, with blocks layered on top of each other
- “Block height” refer to its distance from the first block
- “top” or “tip” refer to the most recently added block
- The first block in the blockchain - **genesis** block



Structure of a Block

A block is a container data structure that aggregates transactions for inclusion in the public ledger, the blockchain.

The block is made of a header, containing metadata, followed by a long list of transactions that make up the bulk of its size.

Block structure

Size	Field	Description
4 bytes	Block Size	The size of the block, in bytes, following this field
80 bytes	Block Header	Several fields form the block header
1-9 bytes (VarInt)	Transaction Counter	How many transactions follow
Variable	Transactions	The transactions recorded in this block

Structure of block header

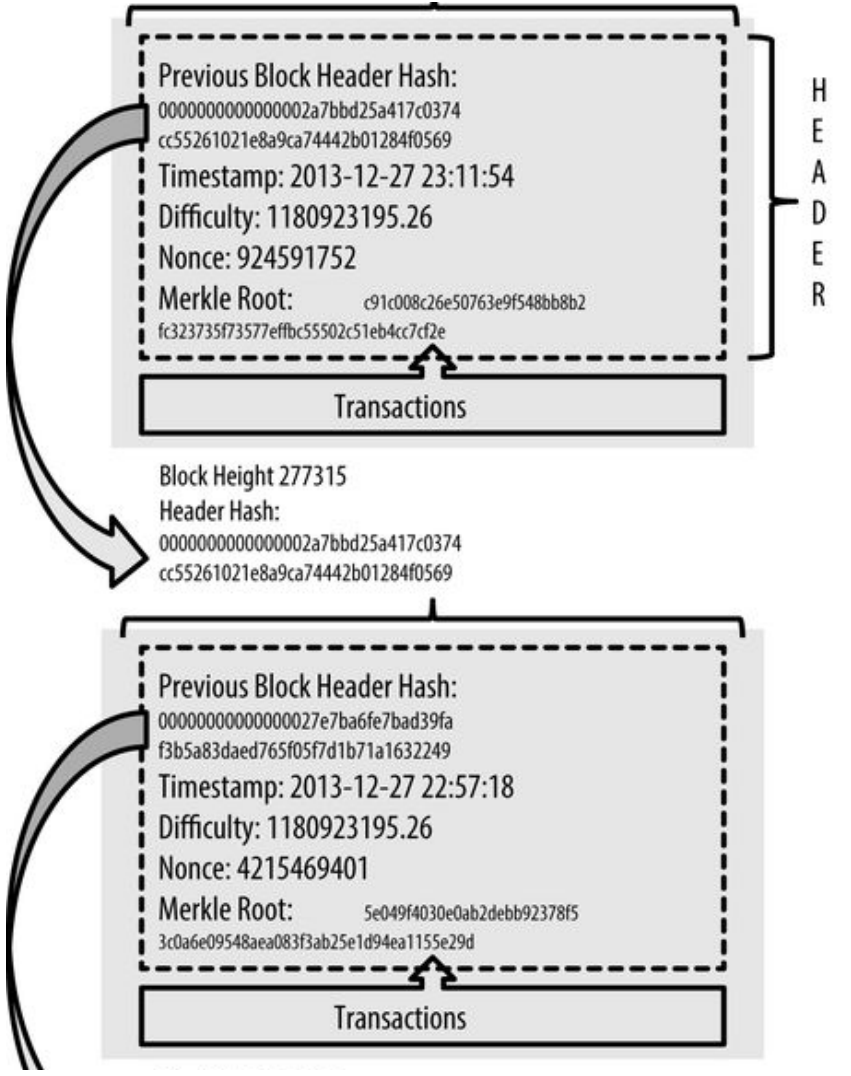
Size	Field	Description
4 bytes	Version	A version number to track software/protocol upgrades
32 bytes	Previous Block Hash	A reference to the hash of the previous (parent) block in the chain
32 bytes	Merkle Root	A hash of the root of the merkle tree of this block's transactions
4 bytes	Timestamp	The approximate creation time of this block (seconds from Unix Epoch)
4 bytes	Difficulty Target	The proof-of-work algorithm difficulty target for this block
4 bytes	Nonce	A counter used for the proof-of-work algorithm

Example


```
{
    "size" : 43560,
    "version" : 2,
    "previousblockhash" :
        "0000000000000027e7ba6fe7bad39faf3b5a83daed765f05f7d1b71a1632249",
    "merkleroot" :
        "5e049f4030e0ab2debb92378f53c0a6e09548aea083f3ab25e1d94ea1155e29d",
    "time" : 1388185038,
    "difficulty" : 1180923195.25802612,
    "nonce" : 4215469401,
    "tx" : [
        "257e7497fb8bc68421eb2c7b699dbab234831600e7352f0d9e6522c7cf3f6c77",

 #[... many more transactions omitted ...]

        "05cfd38f6ae6aa83674cc99e4d75a1458c165b7ab84725eda41d018a09176634"
    ]
}
```

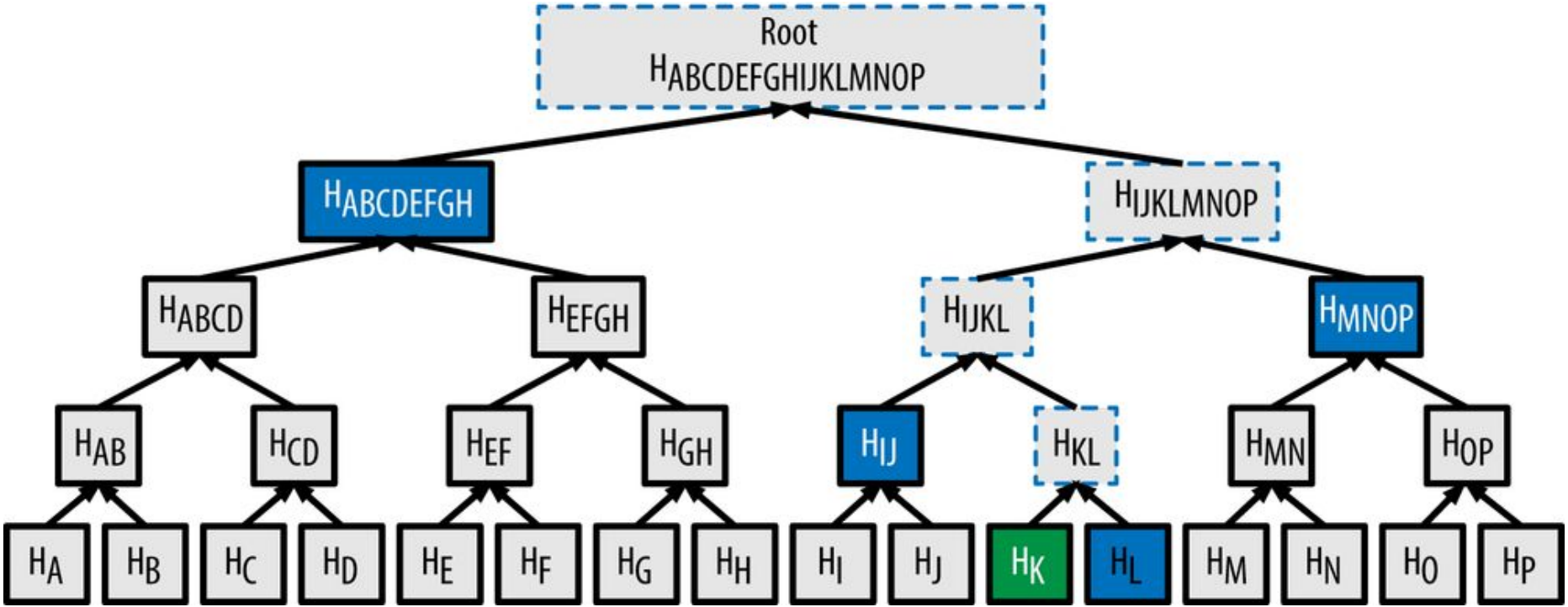



Merkle Trees

- also known as a binary hash tree, used for efficiently summarizing and verifying the integrity of large sets of data.
- used in bitcoin to summarize all the transactions in a block, producing an overall digital fingerprint of the entire set of transactions,
- providing a very efficient process to verify whether a transaction is included in a block.
- cryptographic hash algorithm used in bitcoin’s merkle trees is SHA256 applied twice, also known as double-SHA256.

H~A~ = SHA256(SHA256(Transaction A))

Merkle Path


Root
HABCDEFGHIJKLMNOP
HABCDEFGH
HIJKLMNOP
HABCD
HEFGH
HIJKL
HMNOP
HAB
HCD
HEF
HGH
HIJ
HKL
HMN
HOP
HA
HB
HC
HD
HE
HF
HG
HH
HI
HJ
HK
HL
HM
HN
HO
HP

Proof of Inclusion

To prove that a specific transaction is included in a block, **a node only needs to produce $\log_2$(N) 32-byte hashes**, constituting an ***authentication path*** or ***merkle path*** connecting the specific transaction to the root of the tree.

A node can prove that a transaction K is included in the block by producing a merkle path that is only four 32-byte hashes long (128 bytes total). The path consists of the four hashes (noted in blue in Figure 7-5) $H_{L,}$ H_{IJ}, H_{MNOP} and $H_{ABCDEFGH}$. With those four hashes provided as an authentication path, any node can prove that H_K (noted in green in the diagram) is included in the merkle root by computing four additional pair-wise hashes H_{KL}, H_{IJKL}, $H_{IJKLMNOP}$, and the merkle tree root (outlined in a dotted line in the diagram).

Efficiency of Merkle trees

Number of transactions	Approx. size of block	Path size (hashes)	Path size (bytes)
16 transactions	4 kilobytes	4 hashes	128 bytes
512 transactions	128 kilobytes	9 hashes	288 bytes
2048 transactions	512 kilobytes	11 hashes	352 bytes
65,535 transactions	16 megabytes	16 hashes	512 bytes