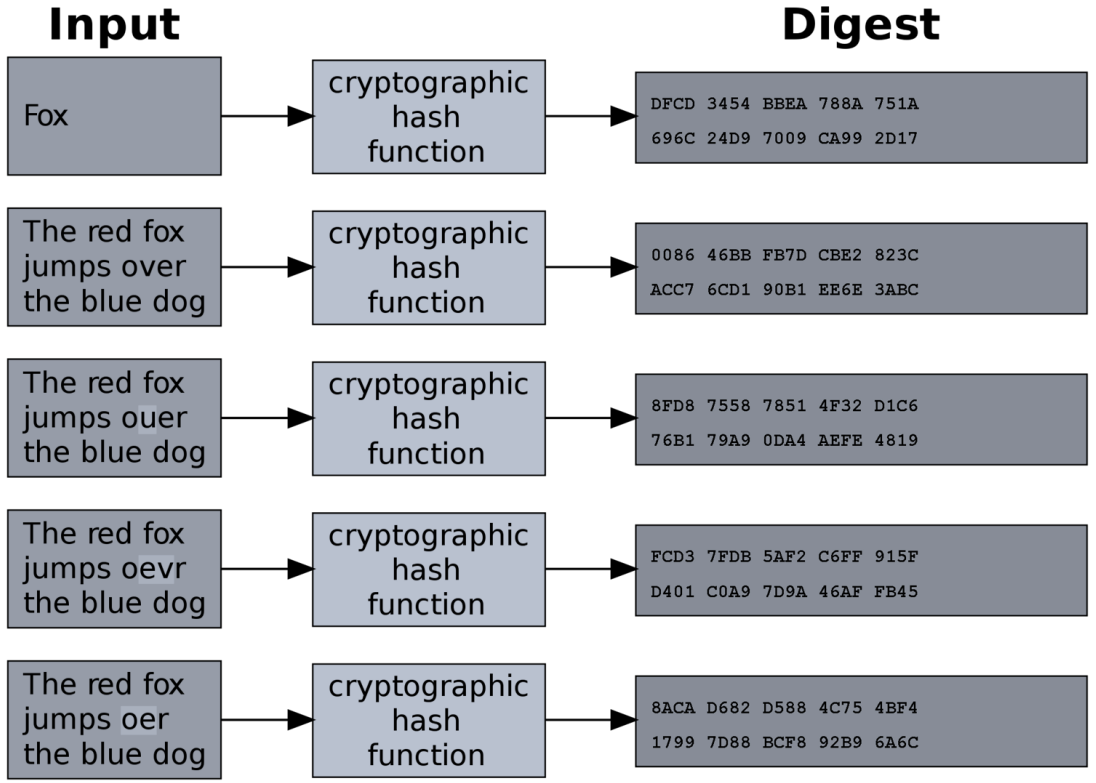
Blockchain

raed**RASHEED**

Hash Function

- A hash function is any function that can be used to map data of arbitrary size to data of fixed size.



Hash Function

- A hash function is any function that can be used to map data of arbitrary size to data of fixed size.

https://andersbrownworth.com/blockchain/hash

Structure of a Block

- The block is made of a header, containing metadata, followed by a long list of transactions that make up the bulk of its size.
- The block header is 80 bytes, whereas the average transaction is at least 250 bytes and the average block contains more than 500 transactions.
- A complete block, with all transactions, is therefore 1,000 times larger than the block header.

Structure of a Block

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

Block Header

- The block header consists of three sets of block metadata.
  - First, there is a reference to a previous block hash, which connects this block to the previous block in the blockchain.
  - The second set of metadata, namely the difficulty, timestamp, and nonce, relate to the mining competition.
  - The third piece of metadata is the merkle tree root, a data structure used to efficiently summarize all the transactions in the block.

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

Block Identifiers

- Block Header Hash
  - The primary identifier of a block is its cryptographic hash, a digital fingerprint, made by hashing the block header twice through the SHA256 algorithm.

    000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f
- Block Height
  - A second way to identify a block is by its position in the blockchain, called the block height. The first block ever created is at block height 0 (zero)

The Genesis Block

- The first block in the blockchain is called the genesis block and was created in 2009.

https://www.blockchain.com/btc/block/000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f

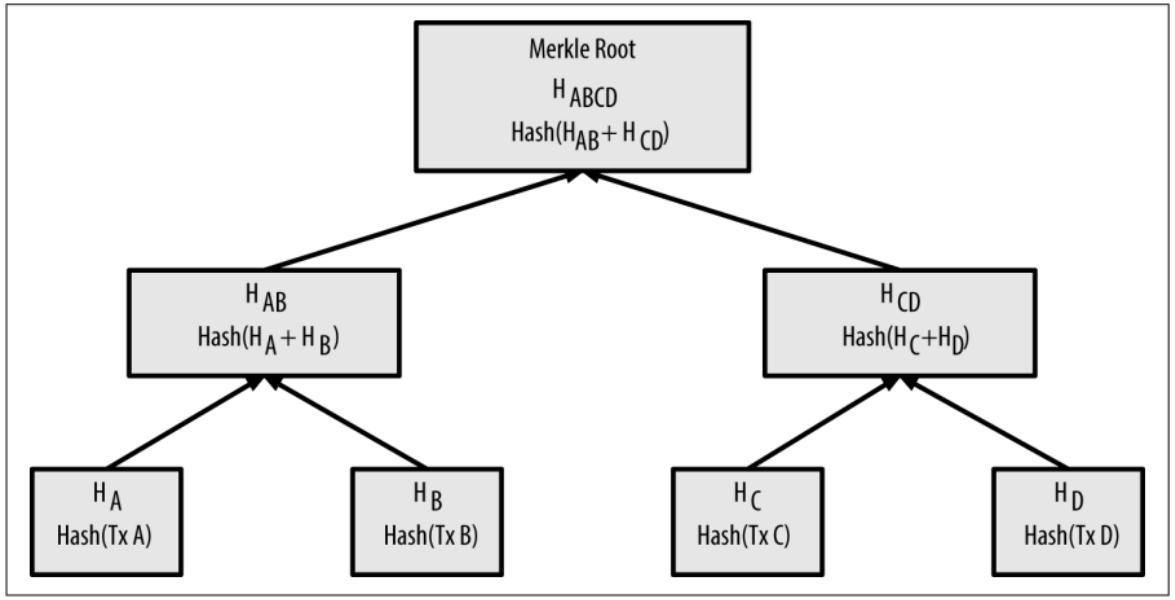
```
$ bitcoin-cli getblock
000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f

{
    "hash" : "000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f",
    "confirmations" : 308321,
    "size" : 285,
    "height" : 0,
    "version" : 1,
    "merkleroot" :
"4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b",
    "tx" : [
        "4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b"
    ],
    "time" : 1231006505,
    "nonce" : 2083236893,
    "bits" : "1d00ffff",
    "difficulty" : 1.00000000,
    "nextblockhash" :
"00000000839a8e6886ab5951d76f411475428afc90947ee320161bbf18eb6048"
}
```

Merkle Trees

- A merkle tree, also known as a binary hash tree, is a data structure used for efficiently summarizing and verifying the integrity of large sets of data.
- Merkle trees are binary trees containing cryptographic hashes.
- The merkle tree is constructed bottom-up.
- we start with four transactions, A, B, C, and D, which form the leaves of the merkle tree.
  - H_A = SHA256(SHA256(Transaction A))
  - H_{AB} = SHA256(SHA256(H_A + H_B))

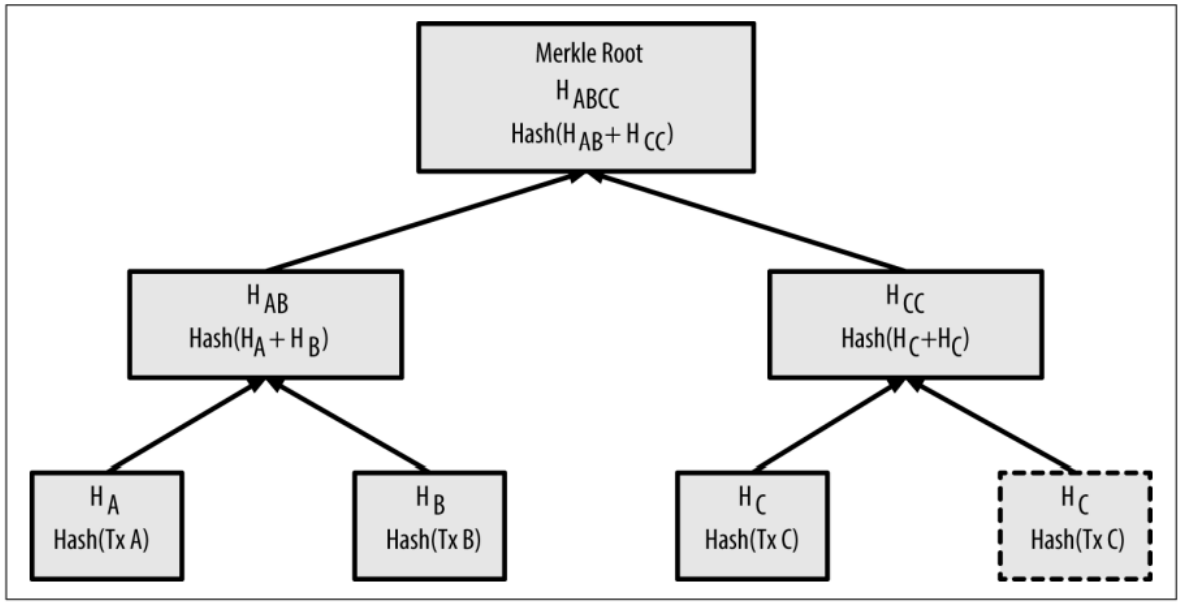
Merkle Trees

- A merkle tree



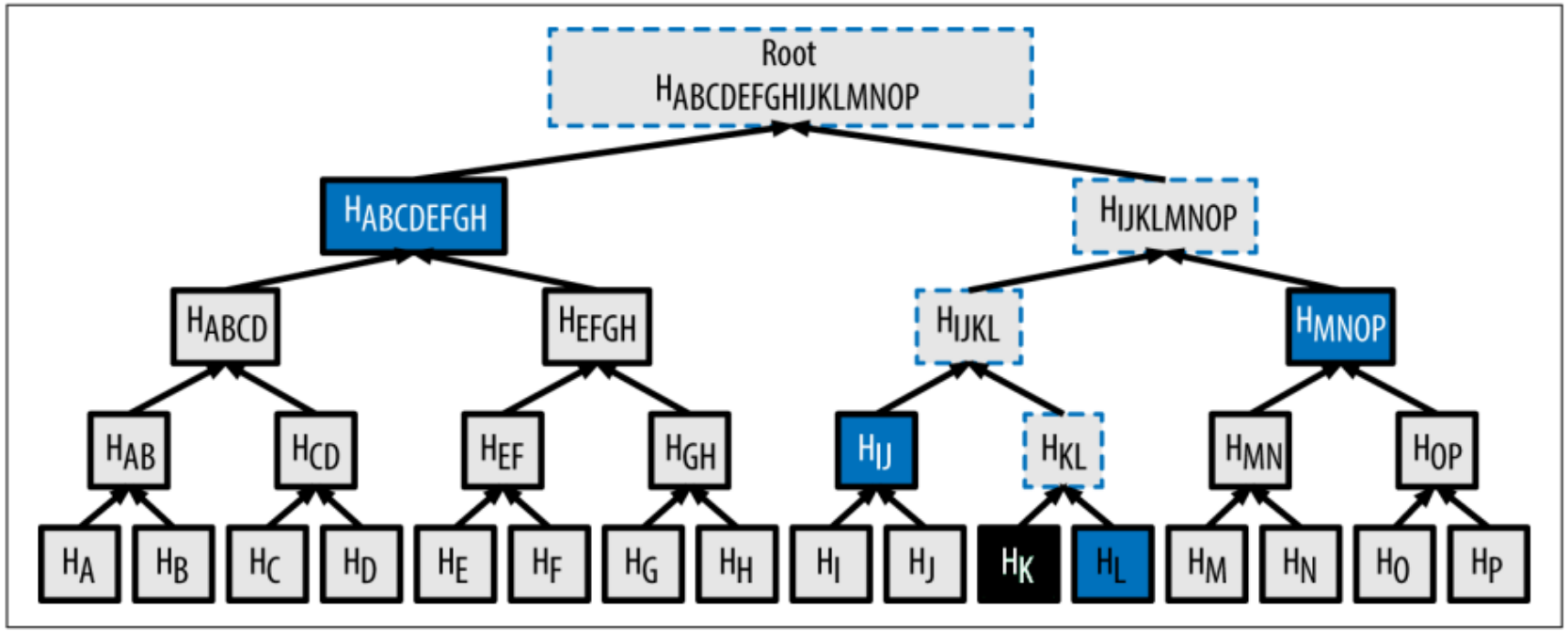
Merkle Trees

- If there is an odd number of transactions to summarize, the last transaction hash will be duplicated to create an even number of leaf nodes, also known as a balanced tree.



Merkle Trees

- 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.



Merkle Trees

- Merkle tree efficiency

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