# Blockchain Data Structure (O.Reilly, 2023)

## Introduction

An ordered back-linked list of blocks of transactions. Backlinked means each block references the previous block in the chain. This data structure is a vertical stack with blocks layered on each other, where the first block is the stack’s foundation. This stack allows us to calculate the structure’s height, the distance from the first block to the top/tip of the most recently added block.

Each block is identified by a hash generated using the SHA256 cryptographic hash algorithm. This hash is found on the block’s header. Along with its hash in the header, the previous block hash is found, which references the parent block (previous block). The sequence of header hashes links each block to its parent, creating a chain back to the genesis block (the first block to be created.). A block only has a single parent. However, a block can have multiple children temporarily. This happens during a blockchain fork where candidate blocks are proposed by miners/block proposers. Eventually, a single block is chosen, and the fork is resolved.

The current block’s hash is affected by the previous block’s hash in the current block’s header. If something changes in the parent block, thus leading to the hash change, it also changes the child block’s identity, thus changing its hash. This means that if a long chain of blocks exists, one change in the contents of a block will change its identity, changing its hash. This changes the child block’s hash since the pointer hash of the parent hash has changed, affecting the other descendants’ hashes. This requires enormous computation. This means that if we have a long blockchain, the history of transactions is immutable. This increases security. [6 blocks are the minimum for an unchangeable blockchain]. The more transactions occur over time, the more stable the blockchain becomes and the harder it becomes to modify the structure.

## Structure of a block

A block is a container for transactional data. It comprises a header(metadata) and a list of transactions. The header is 80 bytes[[1]](#footnote-1), and the transaction is at least 250 bytes. An average block contains more than 500 transactions. The block also contains the nonce, difficulty target and timestamp.

Block size is 4 bytes, the block header (contains important metadata) is 80 bytes, transaction counter (Var-int) is 1-9 bytes and shows how many transactions follow, and transactions change the size.

### Block Identifiers – Block header hash and block height

A block is identified by its cryptographic hash or its digital fingerprint. This hash is obtained by hashing the block header twice using the SHA256 algorithm. This results in a 32-byte hash called the block hash or the block header hash. This hash identifies a block uniquely. The block hash is stored in a different database to help with faster indexing and retrieval.

Another way to identify a block is by its position in the blockchain, a block height. The genesis block has a block height of 0. Each following block added on top of the previous block is higher. Also, not part of the data structure but on a separate file for faster indexing and retrieval.

### The Genesis block

This is the first block. This block is statically encoded within a system and cannot be altered. Every node knows the genesis blocks hash and structure, the creation time and the single transaction within. This is a root of a trusted blockchain.

### Linking Blocks in the blockchain

The current block contains the previous block’s hash and its hash.

### Merkle Trees

A Merkle or binary hash tree summarises and verifies all transactions in a block.

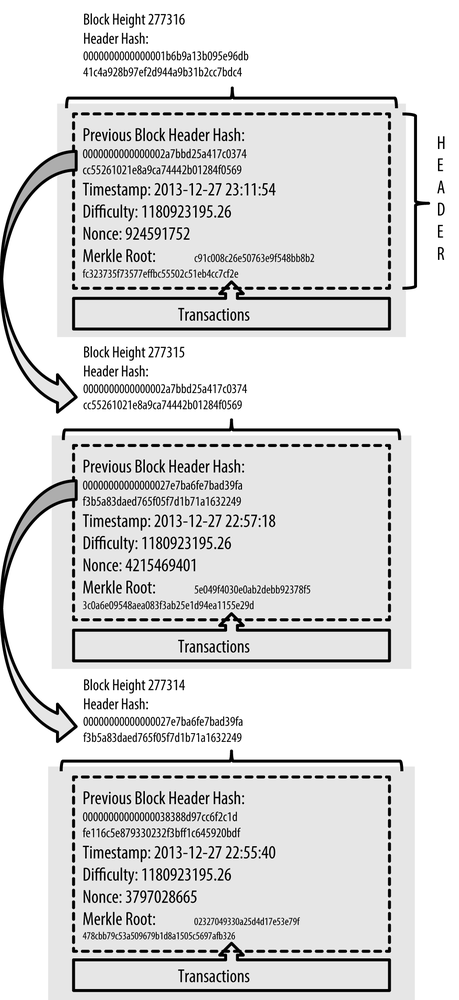
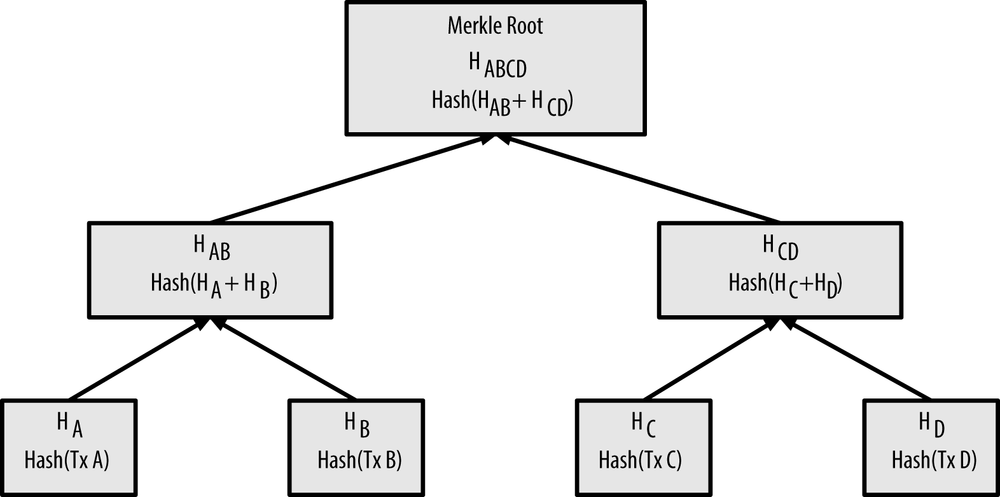


Figure : Blocks linked in a chain by reference to the previous block header hash

A Merkle tree stores the hash of transactions. Each transaction hash is a leaf in a Merkle tree. Consecutive leaf nodes are then summarised in a parent node by adding the two hashes and hashing them together. Thus, this is done multiple times until we’re left with a single node at the top or the Merkel root.



Since the Merkle tree is a binary tree, it only works with an even number of leaf nodes. If there’s an odd number, the last transaction is duplicated to create an even number of leaf nodes, a balanced tree. The Merkel root is always 32 bytes, regardless of the tree size. Any hash in a Merkle tree is 32 bytes. Merkle trees are used to verify transactions without downloading full blocks. Its very storage is efficient and very fast.

All this info about the data structure was found here: https://www.oreilly.com/library/view/mastering-bitcoin/9781491902639/ch07.html#chain\_of\_blocks

Looking at simple data structures

ARRAY: contains several enumerated elements. Each element is indexed, meaning access to a particular element is instant if its position is known. If the position is not known, a sequential search is used. This is the simplest form of data storage. However, the size of the array needs to be known beforehand, along with a size of a particular element. This means memory is set aside for the array, but it is challenging to change the size of the array.

Linked List: data elements are nodes. Each node references/ points to the next node. It’s easier to add data, and it’s very easy to increase the size of the list. Instant access is not available. Only sequential access. The first is called the head, and the last node is the tail. Individual nodes are stored in memory and can reference the next node.

Hash Table: Data elements are called keys. These keys, however, to be stored, need to be hashed. The key is stored in a bucket(hash). These buckets can be stored in an array and can be indexed instantly using its index. The hash functions are not collision-resistant thus can produce the same hash/bucket. A linked list is used to store the buckets. The bucket can contain a key that has a point to another key. Hash tables give you instant access to elements without knowing the location of every element in memory.

<https://www.horizen.io/academy/blockchain-as-a-data-structure/> : This page also gives a simple introduction to what a transaction has.

1. The Bit: represents a state: 1 or 0. 1 for on and 0 for Off.

   The Byte: a collection of bits: 8 bits to be precise. Since we have two states (1or0 or) for each bit, a byte is a collection of these and 2^8 = 256 packages can be formed. Each byte can form any character found In nature (a,e, &,+etc). To create a character, you need 8 bits or 1 byte. Bits are used for interface speeds and Bytes are used for data storage. [↑](#footnote-ref-1)