What is Merkle Tree?

BLOCKCHAIN® EXPERT

blockchainexpert.uk

INTRODUCTION

While having studied blockchain technology, definitely we can see the term ‘Merkle Tree’ in so many places. Merkle Tree is the backbone of the blockchain. To understand the basics of the blockchain, one should be aware of Merkle Tree and the related terminologies. In this attempt, we are trying to give you the basics of Merkle Tree and a simple implementation of the same using Python script.

Merkle Tree is a special type of data structure which is completely built using Cryptographic Hash function. Before going deep into Merkle Tree, let’s have a glance on a hash function. The hash function is a function which converts the input data into a fixed length data regardless of the length of input data. The output of the hash function is called ‘hash value’, ‘hashcode’ or ‘hash’ in short. A hash function generates a completely unique hash with a fixed length for each input data. It is guaranteed that two hash functions will never collide for two or more different input data. Let’s see an example of the hash function.

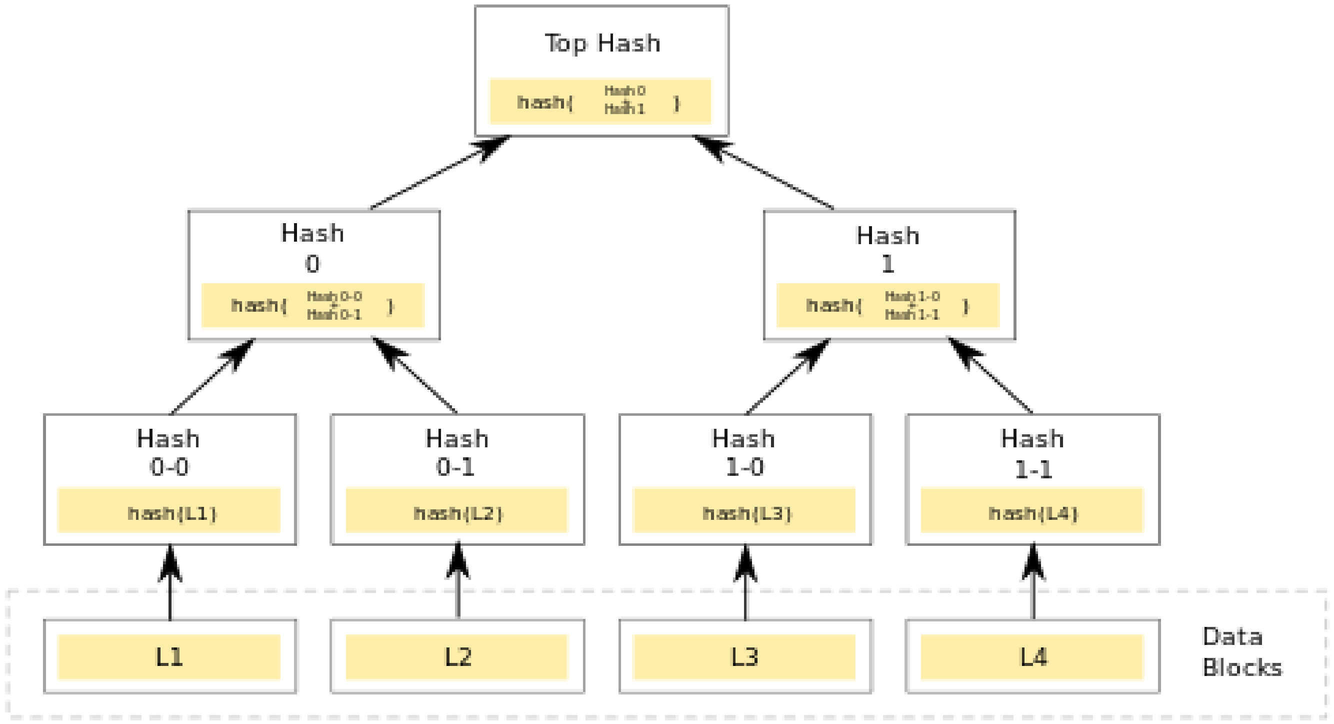
Input Data	Hash Code
Hash Me	644D45DD9DF9D3B92E5D773E13DF5215
HashMe	54FF7B00CC8493E7948F5DDF08396CBC

When hashing the data ‘Hash Me’, you can see a hexadecimal code is generated which is 16 bytes long(32 characters).

In the second example, the data to be hashed is ‘HashMe’. The only change we made is the removal of space between the words. But in the result, you can see the hash code is entirely different. But with the same length of 16 bytes. Unlike normal encryption algorithms, no hash can be decoded into original data(plain text). Which means hashing is a one-way cryptographic method or we can say hashing is irreversible.

Different popular hash functions are, MD5, SHA256, SHA1, and SHA512. Each of them follows different cryptographic algorithms. SHA256 algorithm is the popular one which followed by most of the blockchains including Bitcoin.

As already mentioned, Merkle tree is totally build using a hashing function. In short, a Merkle tree formation is the process of making a single hash from a group of hashes. Let’s see how it works with an illustration.

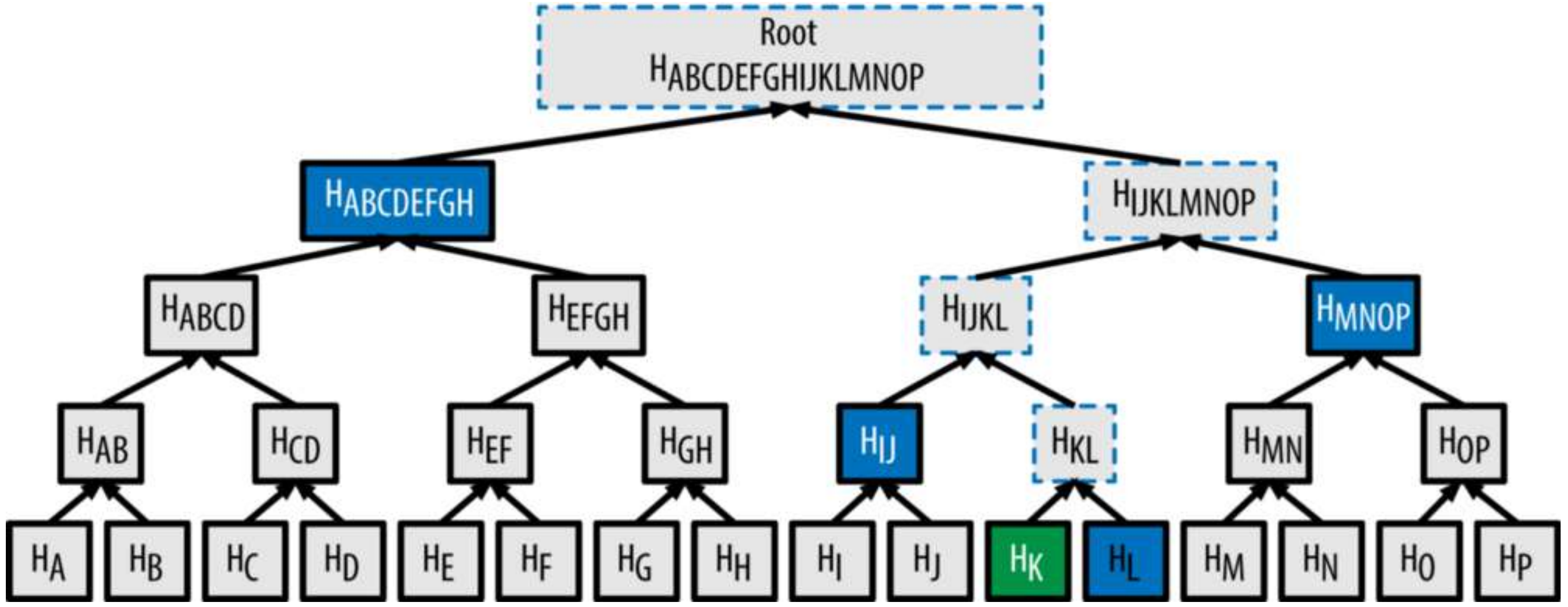


The bottom blocks L1, L2, L3, and L4 are the data blocks. The first step is to hash each data block using a specific hashing algorithm say SHA256. So that the blocks Hash 0-0, Hash 0-1, Hash 1-0, Hash 1-1 are formed. This is the initial step for building a Merkle Tree. These blocks can be called as Leaves of the Merkle Tree. The minimum number of Leaves should be two and there is no upper limit.

Next step is, hashing two adjacent hashes into a single hash. That is, Hash 0-0 is concatenated with Hash 0-1 and again hashed it by SHA256 to get Hash 0. The same process is done for Hash 1-0 and Hash 1-1 blocks to produce Hash 1. The process will continue until a single hash is formed. The final hash is called Root of the Merkle Tree. It is called Merkle Root.

The validation of the existence of a data in a Merkle Tree is an important process. The data to be validated is called Target Hash. As already mentioned, hashes are irreversible. That is, it is impossible to derive the target hashes from the Merkle Root. So that, no data can be validated by decoding the Merkle root. The only way to validate a data in a Merkle tree is to rebuild the tree. Target Hash alone is not enough to rebuild the tree. One method to rebuild the Merkle Tree is by collecting all the leaves and arrange them in the same order and build the tree again.

But this a constraint for many applications and also time and storage consuming. If we study a Merkle Tree formation in little more depth, we can see that all of the leaves are not required for building the tree. Instead, a proof to reach at Merkle Root from Target Hash can be formed. This proof is called Merkle Proof. Merkle Proof is nothing but a collection(array) of hashes which is explained below.



n this Merkle Tree, HK(red block) is our target hash. HKL is formed by hashing HK with HL. HIJKL is formed by hashing HIJ with HKL And so on to reach HABCDEFGHIJKLMNOP. It is clear that, to validate HK in the Merkle tree, the blocks HL, HIJ, HMNOP, HABCDEFGH (yellow blocks) are the only required data instead of all the leaves HA, HB, HC,........HP. Here, the Merkle Proof of HK in this tree is the array of hashes HL, HIJ, HMNOP, and HABCDEFGH.

Merkle Tree Implementation Using Python

Merkletools is a library in python for performing Merkle Tree operations. Install merkletools library for Python. Type the below command in Terminal.

```
pip install merkletools
```

Step-1: Create a merkletools object

```
import merkletools
mt = MerkleTools(hash_type="md5")
```

An object ‘mt’ is created for the merkletools library. In this step, you can specify which hashing algorithm has to be done by setting it to parameter ‘hash_type’. If nothing is specified, SHA256 will be set by default. Here MD5 is given. 'MD5', 'SHA1', 'SHA224', 'SHA256', 'SHA384', 'SHA512' 'SHA3-224', 'SHA3-256', 'SHA3-384', and 'SHA3-512' are supported by merkletools.

Step-2: Add leaves(data) to the tree.

```
hashed_data = '1aae04314577b27f3b4be982eed1b724a6d59e9c413cfb2afa2f0c68e0a93911'
plain_data = ['Lorem ipsum', ‘dolor sit amet']
mt.add_leaf(hashed_data)
mt.add_leaf(plain_data, True)
```

The first argument is the data and second argument is, do_hash. If data is plain text, then set do_hash as ‘True’. If data is already hashed, then no need to set do_hash argument.

• To get the number of leaves:

leaf_count = mt.get_leaf_count();

•To get data from the leaves at index n:

leaf_value = mt.get_leaf(n)

•Remove all the leaves and reset the tree:

mt.reset_tree()

Step-3: Build the tree.

After setting the leaves, execute below code to build the Merkle tree.

mt.make_tree()

Step-4: Checking tree is ready or not.

After executing make_tree, check whether the tree is completed and ready to supply its root and proofs.

is_ready = mt.is_ready()

is_ready returns true when the tree is ready. It returns false if the tree is reset or changing the leaves.

Step-5: Take Merkle root.

root_value = mt.get_merkle_root()

The Merkle root will be returned if the tree is ready. If the tree is not ready, None will be returned.

Step-6: Get proof of each leaf.

proof = mt.get_proof(n)

This will return the proof of the nth index. If there is no leaf at the nth index, then null will be returned.

Step-7: Validation of proof.

validate_proof(proof, target_hash, merkle_root)

This validates the proof with target hash and merkle root. Returns true if proof is valid and correctly connects the target_hash to the merkle_root. The parameter proof is an array which is the output of get_ptoof(). target_hash and merkle_root should be in hex string form

Click here to read more about : **What is Merkle Tree?**

Thank You !

UK

Cybrosys Limited
Alpha House,
100 Borough High Street,
London, Greater London
SE1 1LB, United Kingdom

Mail Us

info@blockchainexpert.uk

Phone

+44 7414439281