Module 1

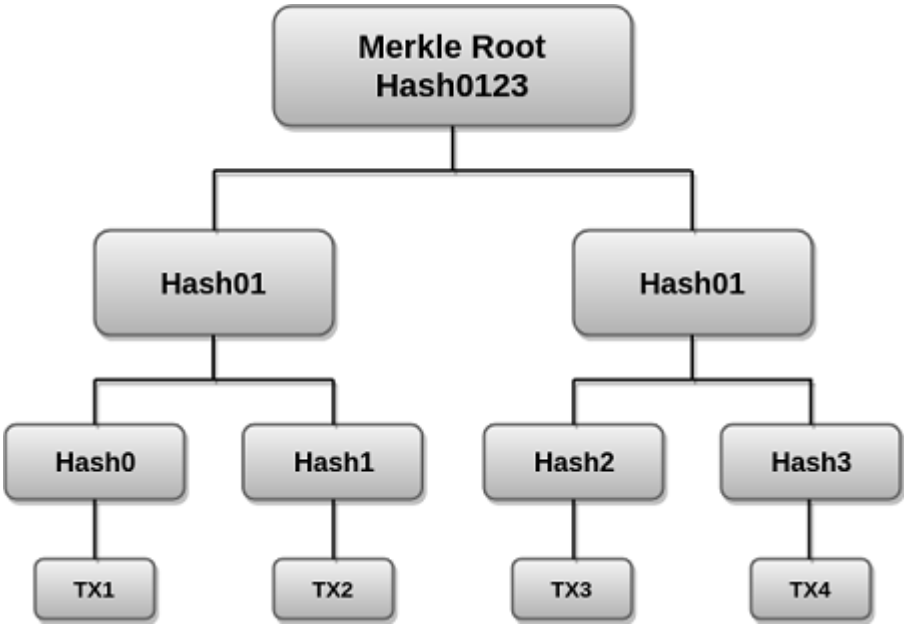
Introduction to DLT & Blockchain

Markle Tree:

- Merkle tree is a fundamental part of blockchain technology.
- It is a mathematical **data structure** composed of hashes of different blocks of data, and which serves as a summary of all the transactions in a block.
- It allows for efficient and secure verification of content in a large body of data.
- It helps to verify the consistency and content of the data. Both Bitcoin and Ethereum use Merkle Trees structure. Merkle Tree is also known as **Hash Tree**.

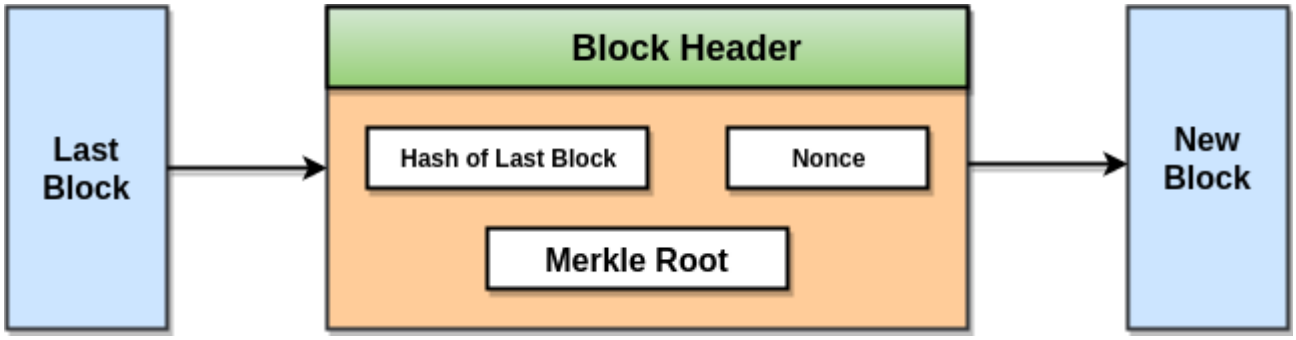
How do Merkle trees work?

- A Merkle tree stores all the transactions in a block by producing a digital fingerprint of the entire set of transactions. It allows the user to verify whether a transaction can be included in a block or not.
- Merkle trees are created by repeatedly calculating hashing pairs of nodes until there is only one hash left. This hash is called the Merkle Root, or the Root Hash. The Merkle Trees are constructed in a bottom-up approach.
- Every leaf node is a hash of transactional data, and the non-leaf node is a hash of its previous hashes. Merkle trees are in a binary tree, so it requires an even number of leaf nodes. If there is an odd number of transactions, the last hash will be duplicated once to create an even number of leaf nodes.



- The above example is the most common and simple form of a Merkle tree, i.e., **Binary Merkle Tree**. There are four transactions in a block: **TX1**, **TX2**, **TX3**, and **TX4**. Here you can see, there is a top hash which is the hash of the entire tree, known as the **Root Hash**, or the **Merkle Root**. Each of these is repeatedly hashed, and stored in each leaf node, resulting in Hash 0, 1, 2, and 3. Consecutive pairs of leaf nodes are then summarized in a parent node by hashing **Hash0** and **Hash1**, resulting in **Hash01**, and separately hashing **Hash2** and **Hash3**, resulting in **Hash23**. The two hashes (**Hash01** and **Hash23**) are then hashed again to produce the Root Hash or the Merkle Root.
- Merkle Root is stored in the **block header**. The block header is the part of the bitcoin block which gets hash in the process of mining. It contains the hash of the last block, a Nonce, and the Root Hash of all the transactions in the current block in a Merkle Tree. So having the Merkle root in block header makes the transaction **tamper-proof**. As this Root Hash includes the hashes of all the

transactions within the block, these transactions may result in saving the disk space.



- The Merkle Tree maintains the **integrity** of the data. If any single detail of transactions or order of the transaction's changes, then these changes reflected in the hash of that transaction. This change would cascade up the Merkle Tree to the Merkle Root, changing the value of the Merkle root and thus invalidating the block. So everyone can see that Merkle tree allows for a quick and simple test of whether a specific transaction is included in the set or not.

Merkle trees have three benefits:

- It provides a means to maintain the integrity and validity of data.
- It helps in saving the memory or disk space as the proofs, computationally easy and fast.
- Their proofs and management require tiny amounts of information to be transmitted across networks.