Lecture 4 – Data Structure (Merkle tree and derivatives) overview summary – Tien Dat Le

A Merkle tree or a Hash tree is an important data structure in cryptography and computer science. It is a kind of tree structure such that every leaf nodes is labelled with the hashed of a data block and every non-leaf node is labelled with the hashed of its child nodes.
The root of the tree is called root hash and is be used to verify the integrity of any data block as the important characteristic of Merkle tree is that if a malicious modification is made on any data block, it will cause to change the root.
In order to verify if an entry belongs to the tree hash only take O(logn) complexity with n is the depth of the tree. Therefore, Merkle tree is used in Bitcoin to verify if a transaction is part of a block.
As the tree root does not indicates the tree depth. It is vulnerable to second-preimage attack: attacker can search for a second document with the same root hash. One simple fix is to append 0x00 and 0x01 to the leaf node hashes and internal node hashes accordingly.

A Merkle-patricia-tree (trie), also known as a radix tree, is a data structure that represents a space-optimized trie in which each node that is the only child is merged with its parent. This makes radix trees very efficient for set of strings that share long prefixes. Radix trees support insertion, deletion and searching operations. Unlike balanced trees, radix trees permit lookup, insertion and deletion in O(k) time rather than O(logn).

The Ethereum implementation of radix tree introduces a number of improvements. First, to make the tree secure, each node is referenced by its hash. With this scheme, the radix tree root node now becomes the fingerprint of the whole trees, which is similar to Merkle tree. Second, they introduced a number of node 'types' to enhance performance. The first type is a blank node, which is empty. The second type is the standard leaf node, which is a list of [key, value]. The 3rd type is extension nodes, which is also a list of [key, value] where value is hash of some other nodes and can be looked up in the database. The final type is branch nodes, which are lists of length 17. The first 16 elements are 16 possible hex characters in a key, and the final element holds a value if there is a [key, value] pair where the key ends at the branch node.
This implementation use a special hex-prefix (HP) encoding used for keys. A single hex character, or 4 bit binary number, is called a nibble. The HP specifications is rather simple. A nibble is appended to the key that encodes both the terminator status and parity. The lowest significant bit in the nibble encodes parity while the next significant bit encodes terminator status.
In Ethereum, from a block header there are 3 roots from 3 of these tries: State Trie, Storage Trie and Transactions Trie.