CS-458-Assignment#01: Implementing a Merkle Tree in Python

Due date: Friday, May 24, at 11:59 pm.

Total points: 50 points

Objective: Understand how Blockchain commitments work and how to implement an efficient commitment to a list of items using Merkle Trees. See week 1 lecture 2 for more information on commitments and Merkle Trees.

In this assignment, you will implement a Merkle tree in Python. You will create a class for Merkle tree functions (add and remove nodes, generate proofs, and return commitments). The tree will take files and append a leaf node for each of them to the tree, hash them, and create/update parent nodes as needed up to the root. You will also implement a function to check if any of the files have been tampered with.

Instructions:

- **Setup**:
  - Create a new Python project.
  - Ensure you have the "hashlib" library available, which is part of the standard library.
- **Task 1**: Hash Function
  - Write a function hash_file(file_path) that takes the path of a file, reads its contents, and returns the SHA-256 hash of the file.
- **Task 2**: Merkle Tree Construction
    - **Note**: In a Merkle Tree, there are two types of nodes: leaf nodes and non-leaf nodes. Leaf nodes are at the bottom of the bottom of the tree and the non-leaf nodes are the internal nodes that contain the hash of the concatenation of their children's hashes.
        i. Leaf Node contains the SHA-256 hash of the file content and the path to the file which can be useful for identification and generation of proofs.
        ii. Non-Leaf Node contains the SHA-256 hash resulting from the concatenation of the hashes of its child nodes and in addition to that it references to its left and right child nodes. These can be actual pointers or indices, depending on the implementation.
    - **TASK:** Write a class MerkleTree that has the following methods:

- AddFileCommitment(): Takes a file path as a parameter and reads the file and adds its hash to the tree as a leaf node, then creates/updates parent nodes as needed.
- GenerateProof(): Takes a leaf node index and generates a proof to its commitment.
- GetRoot(): returns the committed hash at the root node

- **Task 3**: Integrity Check
  Create a function VerifyFile() that takes a Merkle Tree root commitment (just the has), a proof, and a file path and checks if the file was included in the commitment using the given proof. The method should return a boolean value to reject or accept the proof.

- **Task 4**: Testing
  - Create a list of files and write a python test script that:
  - Adds a given list of files to a Merkle tree
  - Get the root commitment from the tree
  - Generate a proof for one of the files (test file)
  - Call the Verify function with the proof, the root commitment and two versions of the test file (one with changes-modified file, and the original test file) then check the integrity of the two files

- **Submission:**
  Work alone on this assignment and include your name in the submitted files
  Submit on canvas assignment #1 dropbox
  - Your Python script contains the hash_file function, Verify function and MerkleTree class.
  - A test script demonstrating the creation of the Merkle tree and the integrity checks.
  - Provide a PDF, which will include screenshot of the result when you execute the test script.
- **Evaluation Criteria:**
  - Correct implementation of file hashing.
  - Correct construction of the Merkle tree.
  - Proper implementation of the integrity check.
  - Clear and functional test script.

- Code readability and comments explaining the key parts of the implementation.

Starter Code Template: Starter code template is attached in Canvas under Assignment#01. Refer to that and complete the assignment.