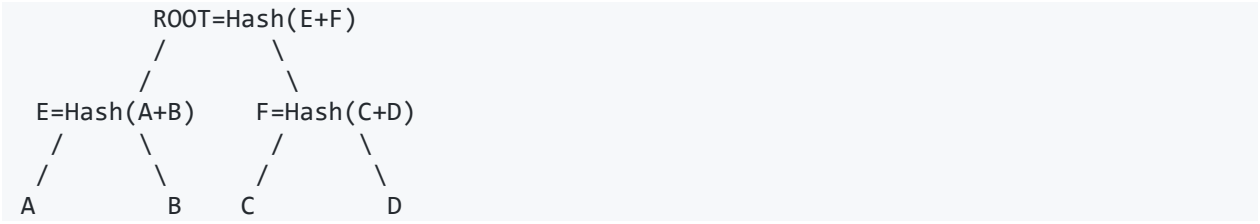
Implementation of Merkle Trees

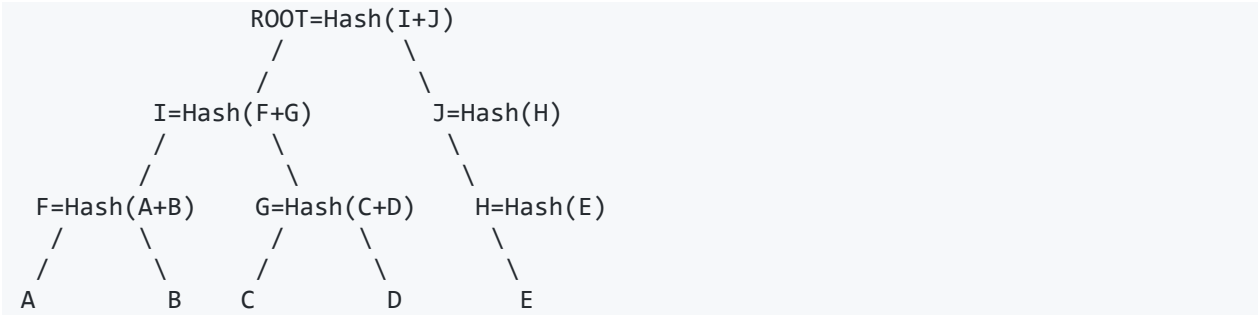
Merkle hash tree is a simple binary tree consisting of hashed leaves and nodes. Leaves are the hashes of individual nodes that have been appended to the next node. Nodes are the hashes of paired child leaves or paired child nodes. The root hash, from which all nodes and leaves stem, is known as the merkle tree hash [1].

***Tree Generation**



***Note that A, B, C, D are the hashes of the transaction or list of names in our case.*

***Tree Generation**



Validation check of Transaction (Inclusion check)

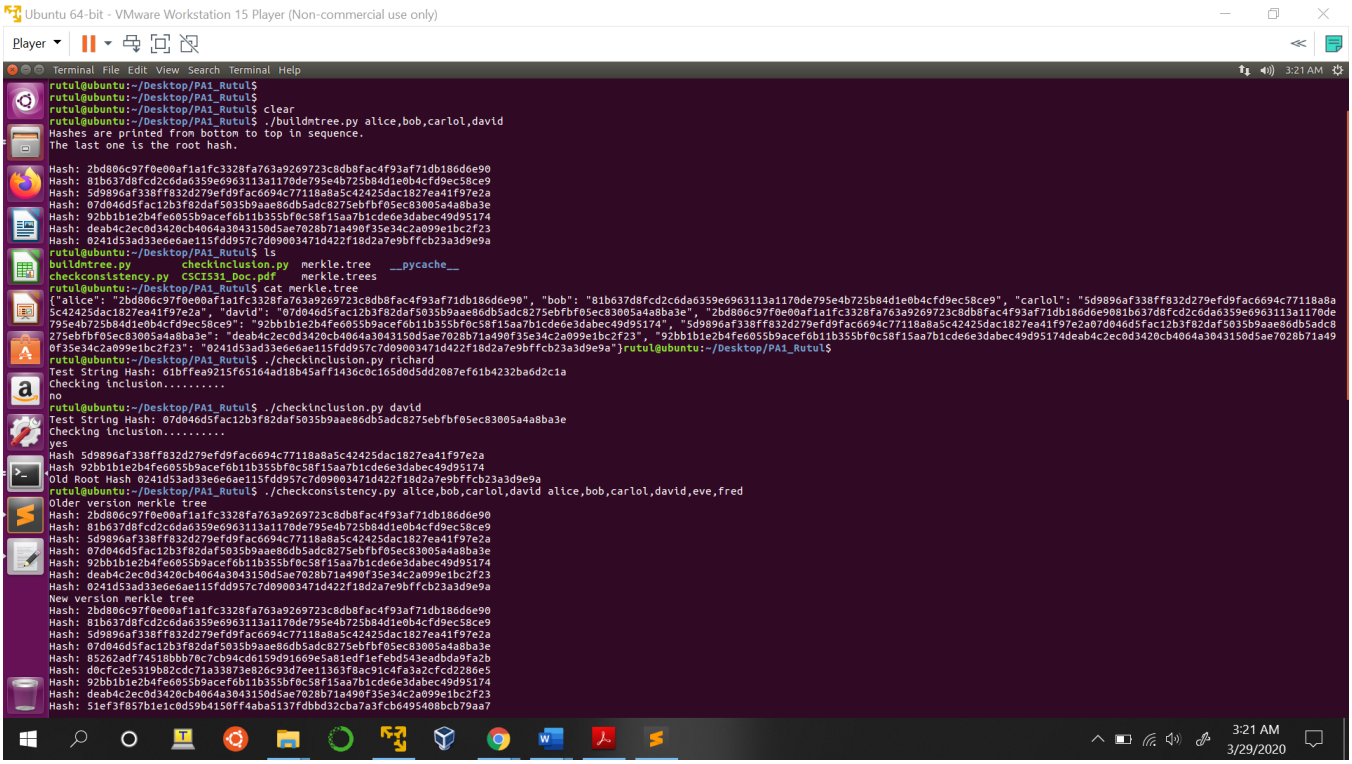
Merkle tree transaction validation is the missing node hashes required to compute all the nodes between the leaf and the tree root. If the root hash you compute from the audit path matches the currently advertised merkle tree root hash, then the leaf exists in the tree [1]. **checkinclusion.py** file checks for inclusion of a transaction only for 4 leaf merkle tree.

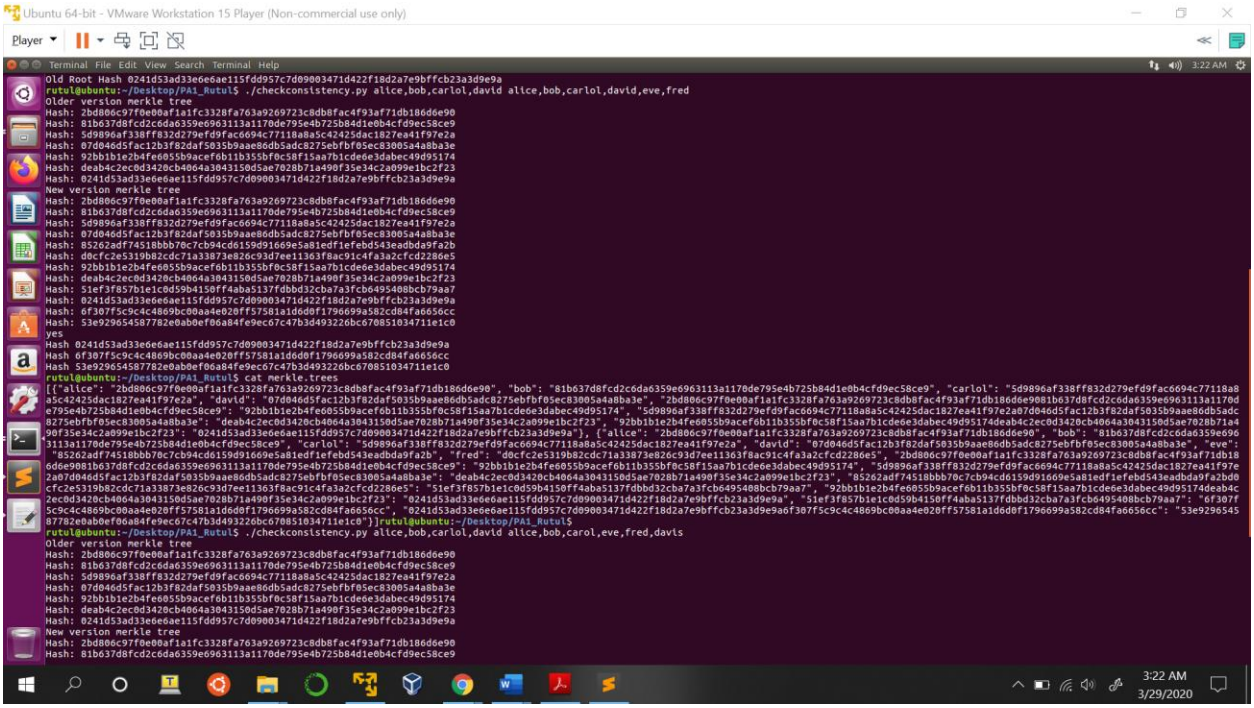
Consistency of Merkle Trees

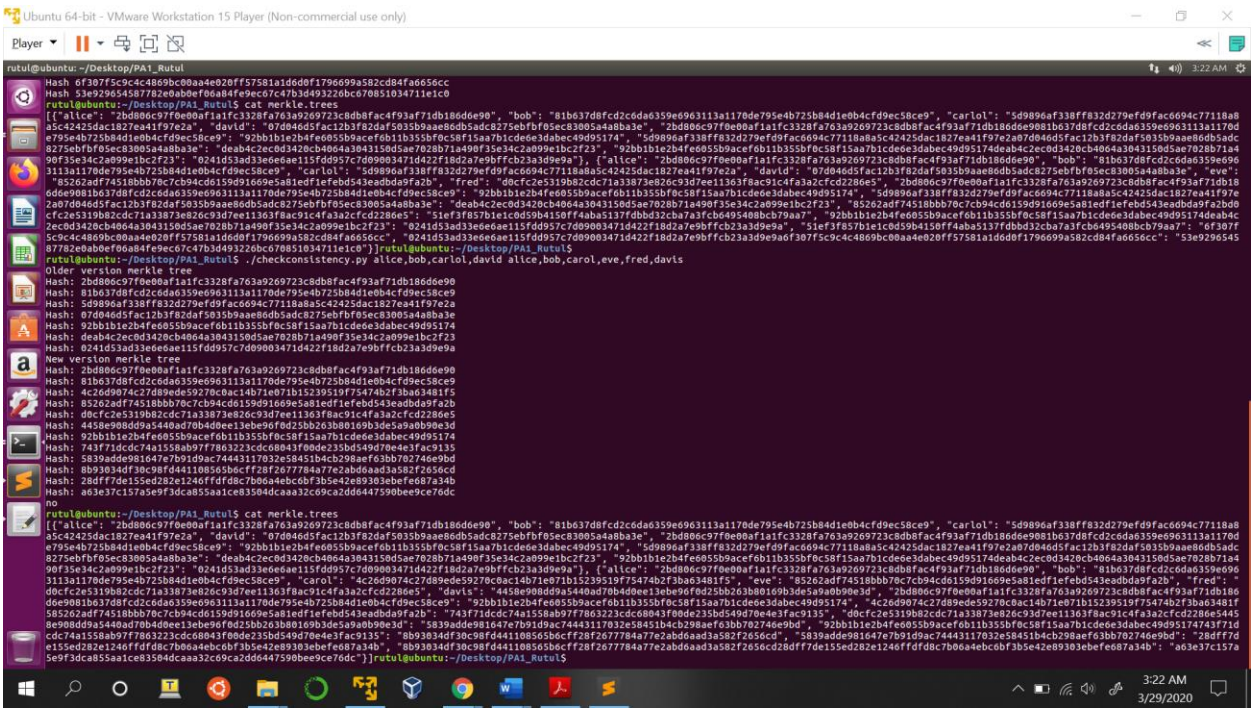
Merkle tree consistency proof lets you verify that any two versions of a tree are consistent: that is, the later version includes everything in the earlier version, in the same order, and all new entries come after the entries in the older version [1]. **checkconsistency.py** file checks for consistency between two merkle trees i.e if the new merkle tree contains all the transactions and in the same order or not.

Commands & Screenshots on Ubuntu

```
./buildmtree.py alice,bob,carlol,david
cat merkle.tree
./checkinclusion.py richard
./checkinclusion.py david
./checkconsistency.py alice,bob,carlol,david alice,bob,carlol,david,eve,fred
cat merkle.trees
./checkconsistency.py alice,bob,carlol,david alice,bob,carol,eve,fred,davis
cat merkle.trees
```







References:

[1] https://www.certificate-transparency.org/log-proofs-work

Rutul Bakulkumar Pandya [USC ID: 2154579267]