Project Topic:

Data Verification (Audit Proof) using Merkle Trees

Note:
Consistency Proof can be done in following scenarios:

1.Time is left.
2.Sir doesn’t agree with only audit poof.
PS: I understood only very basic of consistency proof.

Team Members:

1.Thejaswini DM
2.TSS Chandana
3.Arpit Jadiya
4.Saurabh Agarwala

Here we begin our journey :)

What are Merkle Trees?

A **hash tree** or **Merkle tree** is a tree in which every leaf node is labelled with the hash of a data block and every non-leaf node is labelled with the cryptographic hash of the labels of its child nodes.

PS: Merkle Trees are not necessarily binary trees.

It seems that Merkle trees in Ethereum are not binary tree.

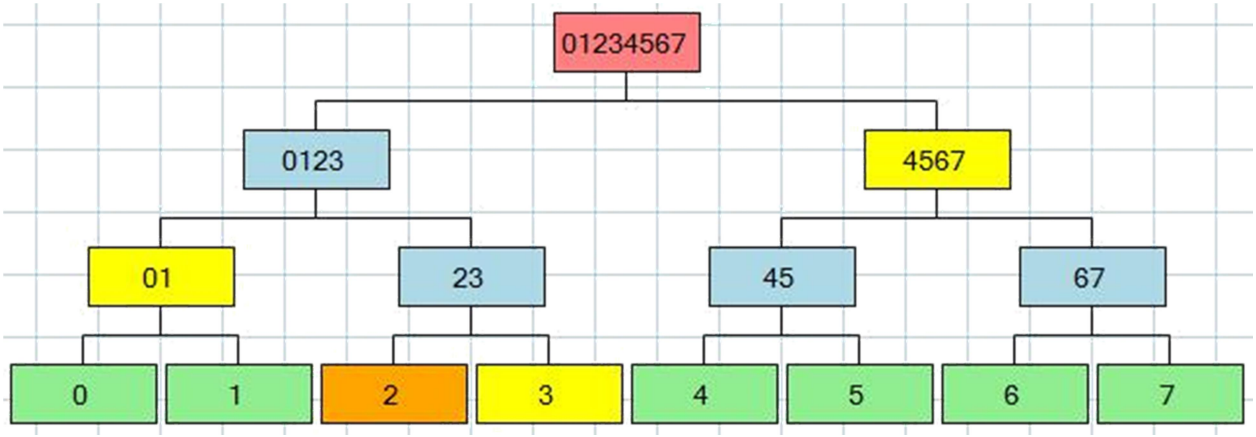
But,I feel We should restrict ourselves to binary..

What is Data Verification(Audit proof) and how does it work?

This is known as an "audit proof" because it lets you verify that a specific record has been included in the log. As with the consistency verification, the server maintaining the log provides the client with a proof that the record exists in the log. "Anyone can request a Merkle audit proof from a log and verify that a certificate [record] is in the log. Auditors routinely send these types of requests to logs so they can verify certificates for TLS clients. If a Merkle audit proof fails to produce a root hash that matches the Merkle tree hash, it means the certificate is not in the log." [7] (More on what a root has is and how an audit proof works later on.)

But there's another reason for sending the proof to the client: it proves that the server itself is not inventing a positive answer, but is instead proving to you, the client, that it knows what it's talking about. Faking a proof is computationally impossible.

In this figure,Yellow colour boxes will be the output.



Why to use Merkle Trees?

1. Significantly reduces the amount of data that a trusted authority has to maintain to proof the integrity of the data.
2. Significantly reduces the network I/O packet size to perform consistency and data verification as well as data synchronization.
3. Separates the validation of the data from the data itself -- the Merkle tree can reside locally, or on a trusted authority, or can itself reside on a distributed system (perhaps you only maintain your own tree.) Decoupling the "I can prove the data is valid" from the data itself means you can implement the appropriate and separate (including redundant) persistence for both the Merkle tree and the data store.

Practical Applications of Data Verification (Audit Proof):

1. In many peer-to-peer (P2P) systems (not just blockchains!) individuals need to be able to request data from untrusted peers with some proof that what those peers sent them is part of the real content they requested. Torrents are an example of this problem: When you download a torrent, you receive files from others “seeding” that torrent

online, but how can you be sure those files are really a part of what you're trying to download, and not garbage or malware? The Merkle Tree can authenticate the data received from your peers to solve this trust problem

2.If someone claims that in some transaction another peer paid them, how can a node on the network verify that transaction really happened? One option is that the node could store the entire history of every transaction that has ever occurred, however, this is prohibitive in terms of both time and space costs to that node. Merkle Trees provide a solution that provide time and space savings for nodes on the network. By creating a Merkle Tree out of the transaction data in each block, transactions can be audited in logarithmic time instead of linear time. Additionally, it opens the door for some bitcoin clients can save space by only storing the root of the Merkle Tree: Not needing to store every transaction that has ever happened in the history of Bitcoin is a huge value!

What is Consistency Proof?:

A Merkle consistency proof lets you verify that any two versions of a log 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. If you can prove that a log is consistent it means that no certificates have been back-dated and inserted into the log, no certificates have been modified in the log, and the log has never been branched or forked.

To see how Merkle consistency proofs work, assume you want to verify the consistency between the log in figure 2 and the log in figure 3. First, you need to verify that the old Merkle tree hash is a subset of the new Merkle tree hash. Then you need to verify that the new Merkle tree hash is the concatenation of the old Merkle tree hash plus all the intermediate node hashes of the newly appended certificates. The consistency proof is the minimum set of intermediate node hashes you need to compute these two things.

In this case, the consistency proof consists of the following intermediate node hashes: k, l, and m (see figure 4). You can use k and m to create the old Merkle tree hash, thereby verifying that the old tree exists and is unchanged. Then you can use l with kto create n, and then use n with m to create the new Merkle tree hash for the log. If your computed

Merkle tree hash matches the one advertised by the log, then you know the log is consistent.

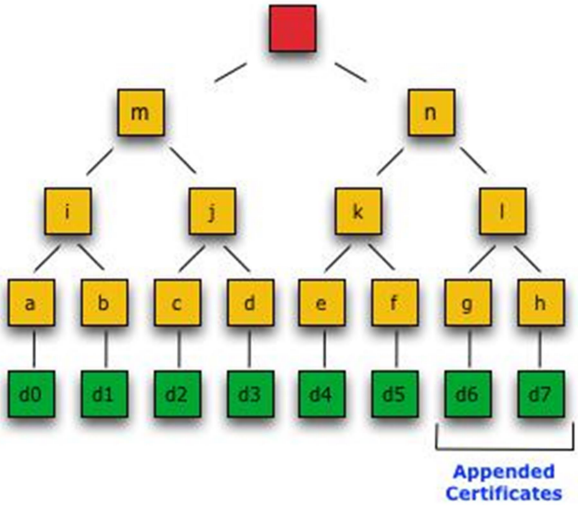


Figure 3

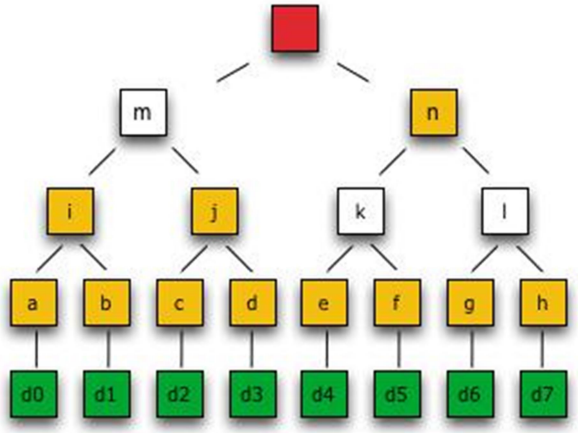


Figure 4

Use of Consistency proof:

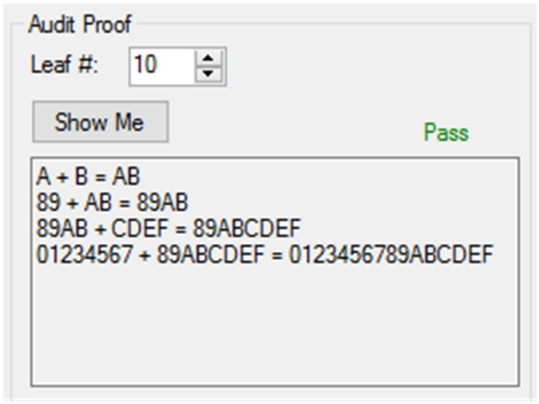
If you can prove that a log is consistent it means that:

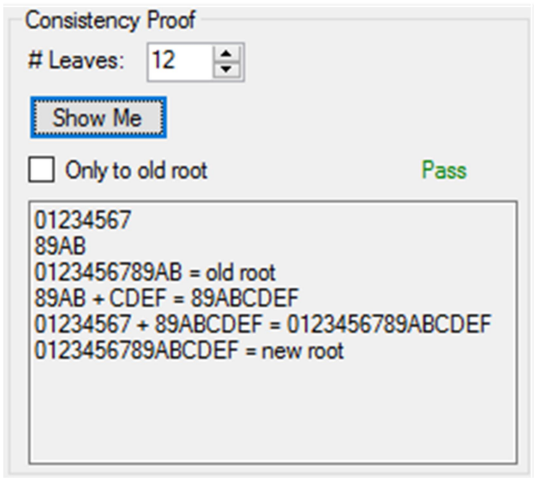
- no certificates [records] have been back-dated and inserted into the log
- no certificates have been modified in the log,
- and the log has never been branched or forked." [7]

A consistency proof is therefore important for verifying that your log has not been corrupted. "Monitors and auditors regularly use consistency proofs to verify that logs are behaving properly."

PS: I don't know what the hell these monitors and auditors are….but without it also, Its use make sense .

Future:


Audit Proof
Leaf #:
10
Show Me
Pass
A + B = AB
89 + AB = 89AB
89AB + CDEF = 89ABCDEF
01234567 + 89ABCDEF = 0123456789ABCDEF


Consistency Proof
Leaves:
12
Show Me
Only to old root
Pass
01234567
89AB
0123456789AB = old root
89AB + CDEF = 89ABCDEF
01234567 + 89ABCDEF = 0123456789ABCDEF
0123456789ABCDEF = new root

References and Readings:

These are the links which We have found ,Although,I haven't gone through all the links

https://hackernoon.com/merkle-tree-introduction-

4c44250e2da7 (Most Important)

https://www.codeproject.com/Articles/1176140/Understanding-Merkle-Tree s-Why-use-them-who-uses-t

(Next Most Important)

http://www.certificate-transparency.org/log-proofs-

work (For understanding basics of consistency proof)

https://hackernoon.com/merkle-trees-181cb4bc30b4

https://blockonomi.com/merkle-tree/

https://en.wikipedia.org/wiki/Merkle_tree