File Upload and Authentic Download

MEHDI VALINEJAD, *Blockchain Protocol Engineer*

***Abstract*—In today's world, file storage and sharing are essential. However, managing large amounts of data on local storage devices like Hard Drives can be cumbersome. To address this, online storage solutions allow us to access files from anywhere without physically carrying storage devices. But how can we ensure that the data remains unaltered during upload? Enter the Merkle Tree Hash and its associated proof—a method to verify the integrity and correctness of downloaded data. In this report, we explore various implementation and verification approaches for achieving this goal. Implementation and Notes can be found on file-upload-merkle**

***Index Terms*—File Upload, File Download, File Server, Merkle Tree, Merkle Proof, Zero Knowledge Proof.**

I. Experiment

THE experiment of client-server file upload and Merkle Tree verification with the following steps:

- File Upload and Merkle Hash Calculation:
  1) Users upload a set of files using the client application.
  2) Locally, the client calculates a Merkle Hash for the Merkle Tree root based on these files.
  3) The client then sends an upload request to the server application.
- Server Logic:
  1) The server application creates an identical Merkle Tree using the same files as the client.
  2) It stores this Merkle Tree in server memory storage.
- Client Trust and Validation:
  1) After uploading, the client can delete the local files.
  2) When needed, the client can download a file from the server.
  3) The client validates the file's content integrity by comparing its proof against the stored Merkle root.

This setup ensures data consistency and trust between the client and server during file transfers.

II. Planned Improvements

1. In-Place Tree Modification

The current file tree implementation lacks the ability to insert or delete files. To address this limitation, we propose replacing the existing Merkle Tree with a Binary Merkle Tree. Unlike the traditional Merkle Tree, the Binary Merkle Tree allows for efficient addition and deletion of nodes. Each individual tree element is hashed, and each node is associated with a full key hash. The benefits of this approach:

1) Fast Updates: The number of hash operations for adding or deleting nodes is proportional to the hash function's bit length (independent of the tree's size).
2) Authenticity: The root commits to the entire tree structure, indirectly validating all its leaves.
3) Deterministic: The order of insertions and deletions does not impact the tree structure1

2. Alternative Verification Methods

1) Verkle Tree: a more efficient way of cryptographic structure with more compact proofs of data.
2) MMR: Better performance and flexibility for the future proof selection for the more sophisticated file upload and download features.

3. Alternative Academic Merkle Tree replacement potential

- HEX-BLOOM: Authenticity and Integrity verification using Bloom Filter and Exclusive-OR [1]. This can be implemented and integrated to the file upload and download solution.
- MTFS: Merkle-Tree-Based File System using distributed random notes without a centralized controller to distribute files in a P2P network. [2]
- Snarl: Uses merkle tree to reduce the impact of hierarchical dependency and cascading chunk failures. [3]. The method is not implemented, so it can be verified and be implemented support large or chunked file storage.

4. File Download Considerations

- Sandboxed File Download: Files need to be downloaded and their hash need to be recalculated to be able to verify their authenticity, so it the file is manipulated maliciously they need to be downloaded in a sandboxed environment locally before the hash and proof verification.

III. Literature

IPFS

Decentralized distributed file system protected by Merkle DAG (Directed Acyclic Graph) as a replacement of Merkle tree which can be constructed from the leaves directly. the nodes can carry a payload and there is no balance restriction.

Cassandra Anti-Entropy repair

Anti-entropy repair involves comparing data across all replicas and updating each replica to the most recent version. In Cassandra, this process consists of two phases: building a Merkle tree for each replica and then comparing these trees to identify any differences

REFERENCES

[1] R. Patgiri and M. D. Borah, "HEX-BLOOM: An efficient method for authenticity and integrity verification in privacy-preserving computing," Cryptology ePrint Archive, Paper 2021/773, 2021, https://eprint.iacr.org/2021/773. [Online]. Available: https://eprint.iacr.org/2021/773

[2] J. Kan and K. S. Kim, "Mtfs: Merkle-tree-based file system," in *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2019, pp. 43–47.

[3] R. Nygaard, V. Estrada-Galiñanes, and H. Meling, "Snarl: entangled merkle trees for improved file availability and storage utilization," in *Proceedings of the 22nd International Middleware Conference*, ser. Middleware '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 236–247. [Online]. Available: https://doi.org/10.1145/3464298.3493397