Verkle Trees: Bandwidth-efficient alternative to Merkle Trees

Arya Phadke, Gunjan Dhanuka, Siddharth Bansal, Vani Krishna Barla

CS581 - Blockchain Science and Technology

Merkle Trees

- The Merkle proof of a leaf in Merkle tree consists of siblings of each of the nodes in the path from leaf to Merkle root.
- The local digest can then be compared to the Merkle Proof obtained from Merkle Tree (on server).
- Merkle Tree that contains many small files can have Merkle proofs that are prohibitively large, since the hash for all siblings at each level has to be sent for verification.
- This creates a *large and expensive bandwidth overhead* in the network.

Merkle Trees

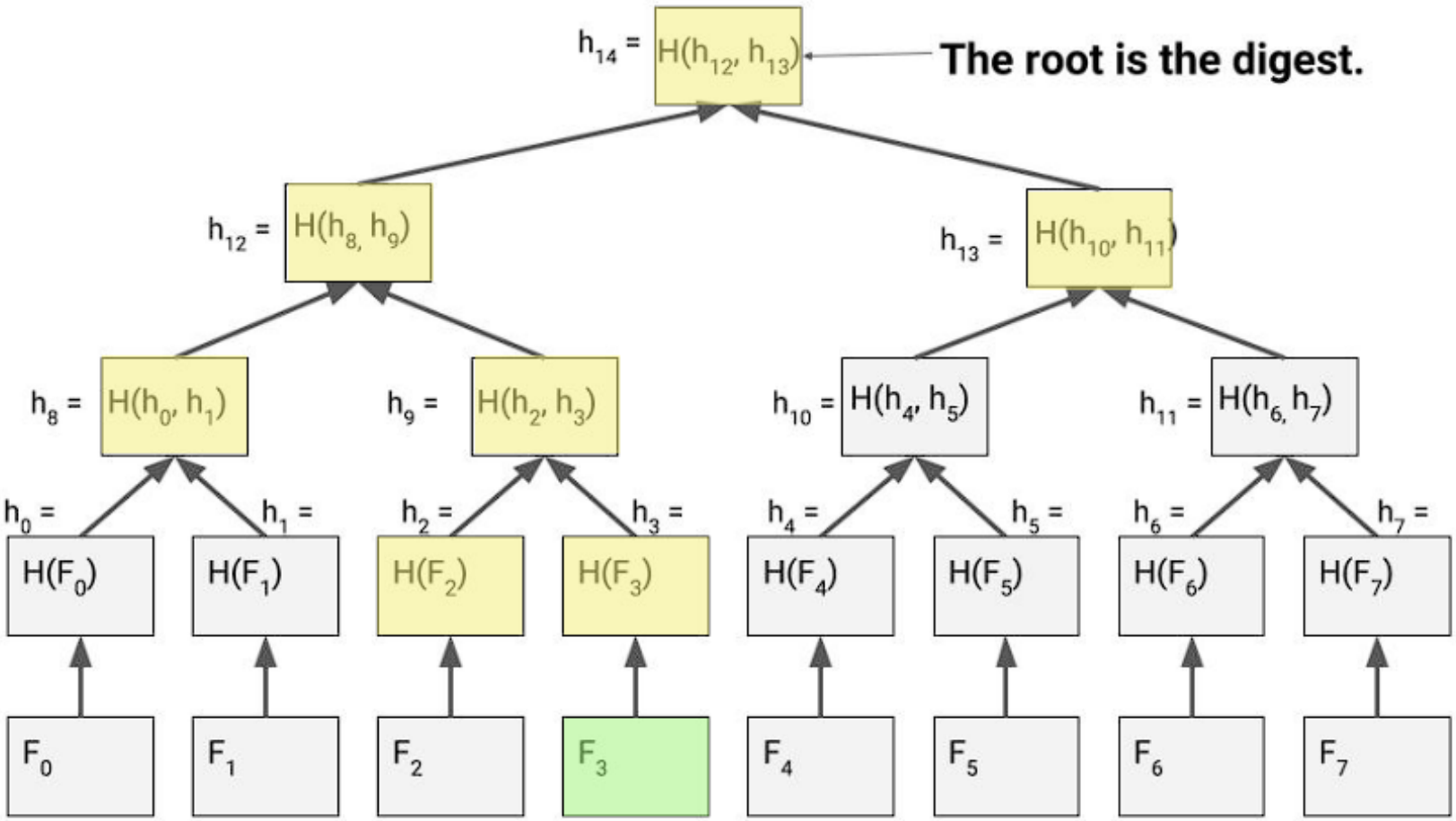

Figure: A Binary Merkle Tree ($k = 2$)

Verkle Trees

- Verkle Trees reduce the proof size from $\mathcal{O}(\log_2 n)$ to $\mathcal{O}(\log_k n)$ for a branching factor k.
- Setting $k = 1024$, the Verkle Proof will be 10 times smaller than the Merkle Tree, coming at a cost of k times more computation.
- Vector Commitments are used instead of the Cryptographic Hash functions (SHA-256) in Verkle Trees.

Verkle Trees

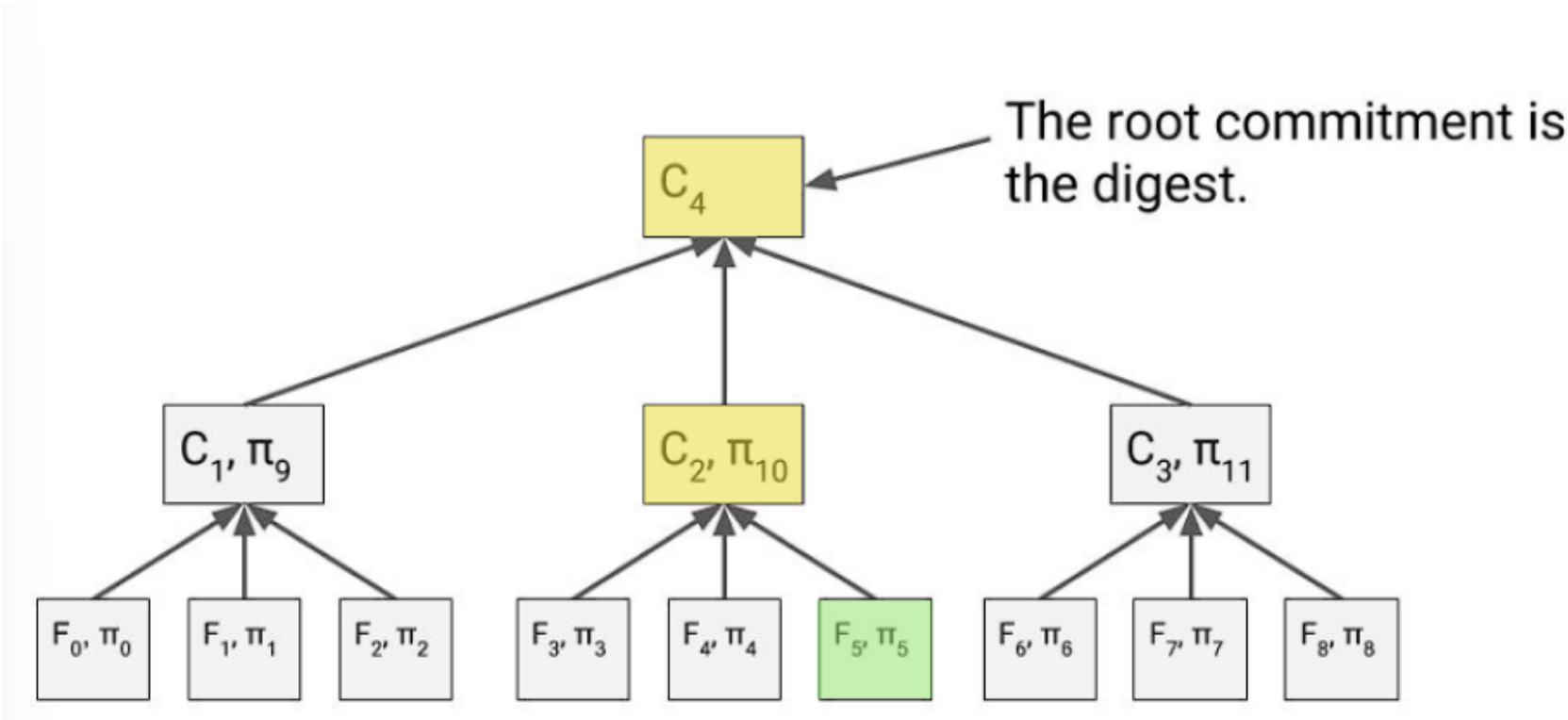

Figure: A Verkle Tree with $k = 3$

Vector Commitments

Vector Commitment (VC) [CF13] allows to commit to an ordered sequence of values in such a way that it is later possible to open the commitment only w.r.t. a specific position. The following algorithms describe VCs:

- $VC.KeyGen(1^k, q)$
- $VC.Com_{pp}(m_1, ..., m_q)$
- $VC.Open_{pp}(m, i, aux)$
- $VC.Ver_{pp}(C, m, i, \Lambda_i)$
- $VC.Update_{pp}(C, m, m', i)$
- $VC.ProofUpdate_{pp}(C, \Lambda_j, m', i, U)$

Remark

The size of the commitment string C is independent of q.

Verkle Trees v/s Merkle Trees

Scheme	Construct	Update	Proof Size
Merkle Tree	$\mathcal{O}(n)$	$\mathcal{O}(\log_2 n)$	$\mathcal{O}(\log_2 n)$
k-ary Merkle Tree	$\mathcal{O}(n)$	$\mathcal{O}(k \log_k n)$	$\mathcal{O}(k \log_k n)$
Vector Commitment	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$
k-ary Verkle Tree	$\mathcal{O}(kn)$	$\mathcal{O}(k \log_k n)$	$\mathcal{O}(\log_k n)$

Table: Time Complexities

The proof size of Merkle size is of $\mathcal{O}(\log_2 n)$ and is costly in terms of bandwidth. VC reduces the proof to a constant $\mathcal{O}(1)$ but comes at a computation cost of $\mathcal{O}(n^2)$ to construct.

Remark

The proof size for k-ary Verkle Tree is a factor of $\mathcal{O}(\log_2 k)$ times smaller than Merkle Tree's membership proofs.

Applications of Verkle Trees in Blockchain

ETHEREUM [W+14]

- Ethereum Clients currently use Merkle Tree to store state data, where information about individual accounts are stored as leaves.
- To verify blocks, Ethereum clients execute all the transactions in a block and update their local state trie.
- Verkle Trees are critical to stateless Ethereum clients.
- Stateless clients use a "witness" to the state data that arrives with the block instead of storing the entire state database.
- The witnesses need to be very small in order to broadcast across the network in time for validators.
- Verkle trees solve this problem by enabling small witnesses, removing one of the main barriers to stateless clients.

Our Implementation

- Our implementation of Verkle Trees was written in Python and used the Charm-Crypto library [AGR11] as well as the SHA-256 cryptographic hash function.
- We began by implementing the CDH-based Vector Commitment scheme by Catalano and Fiore [CF13].
- We then constructed the tree in a bottom-up manner, where the leaf nodes contain the data and internal nodes contain the commitments and proofs of their parents.
- We added the verification mechanism outside of the Tree class, to simulate real-life scenarios where the request for proof is initiated by the client on their local system.

References

- Joseph A. Akinyele, Matthew D. Green, and Avi D. Rubin, *Charm: A framework for rapidly prototyping cryptosystems*, Cryptology ePrint Archive, Paper 2011/617, 2011, https://eprint.iacr.org/2011/617.
- Dario Catalano and Dario Fiore, *Vector commitments and their applications*, Public-Key Cryptography - PKC 2013 - 16th International Conference on Practice and Theory in Public-Key Cryptography, Nara, Japan, February 26 - March 1, 2013. Proceedings (Kaoru Kurosawa and Goichiro Hanaoka, eds.), Lecture Notes in Computer Science, vol. 7778, Springer, 2013, pp. 55–72.
- Gavin Wood et al., *Ethereum: A secure decentralised generalised transaction ledger*, Ethereum project yellow paper **151** (2014), no. 2014, 1–32.