Advanced Ethereum data structure: Tries

A PRESENTATION BY JINANE AMAL, ANMOOL AMJAD AND JOSÉPHINE LAGUARDIA

Introduction

- What makes Ethereum more scalable and efficient in comparison to the other blockchains ?
- How does It keep track of the continuous flow of informations ?
- How does it ensure that each transaction, within a continuously growing network, remains verifiable and tamper proof ?

⇨ **What exactly is a Trie ?**

Merkle Patricia Trie (MPT)

PATRICIA – Key storage

PATRICIA : Practical Algorithm To Retrieve Information Coded in Alphanumeric

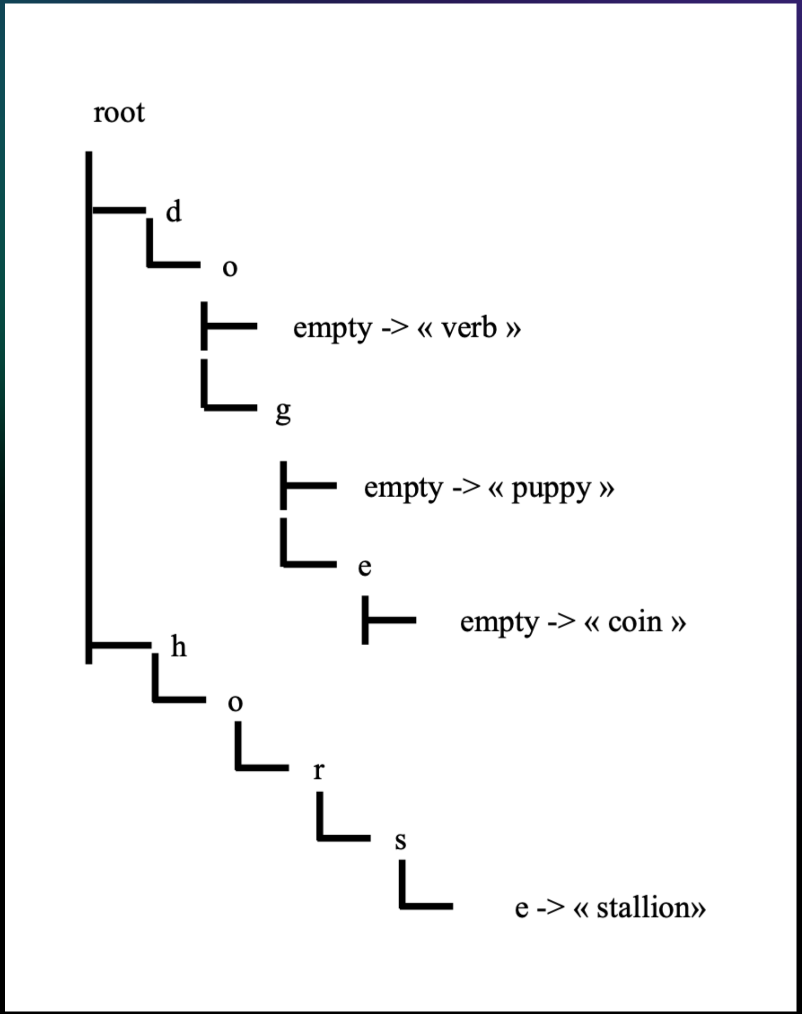
Ordered tree structure that represents keys as paths, or sequence of characters.

Illustrative example: 4 key-value pairs -> (do : verb), (dog: puppy), (doge: coin) and (horse: stallion)

(Step 0: convert to hexadecimal representation)

Key characteristics:

- Fast look-ups
- No cryptographic security (anyone can modify it)

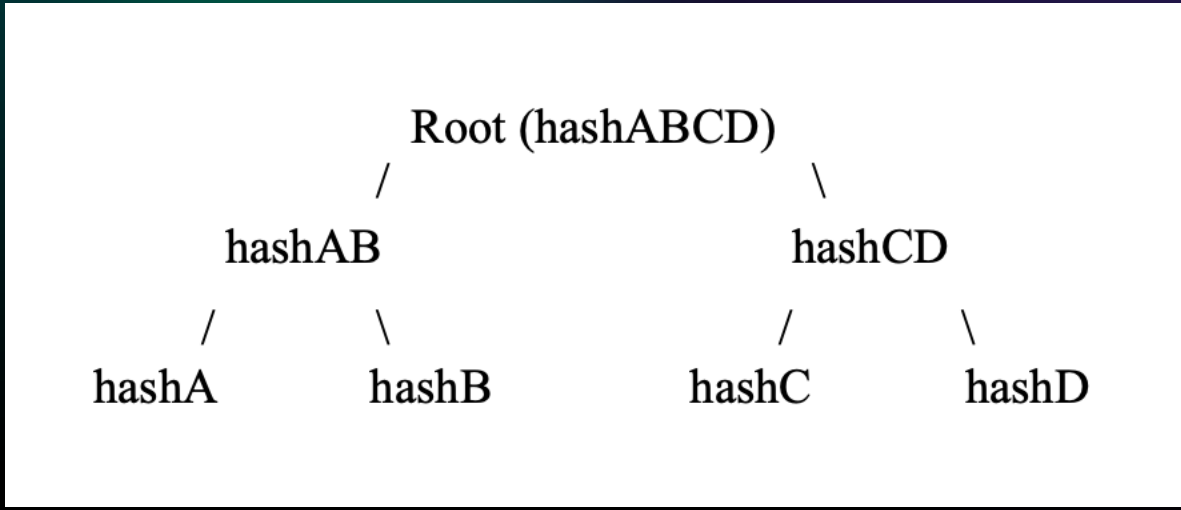


Merkle Tree – secure and tamper-proof data

Binary tree where leaf nodes store data (hashed) and parent nodes store hashes of children.

Key characteristics:

- Tamper-proof: changing one transaction changes the entire tree
- Efficient verification: only need part of the tree for verification (Merkle proof)



Merkle Patricia Trie

Combines Patricia Trie (efficient key storage) + Merkle Tree (tamper-proof security).

Ethereum's trie structure consists of three types of nodes:

1. **Branch node**
    - Used when multiple keys share a common prefix.
2. **Extension node** (path + pointer to another node)
    - Compresses long, unique paths to save space.
3. **Leaf node**
    - Stores the final value at the end of a path.

The root hash (Merkle Root) summarizes the entire trie state.

Illustrative example:

rootHash → hashA

hashA → Branch Node

- "d" (hashB)
- "h" (Leaf: 'stallion')

hashB → Extension Node "o"

hashC → Branch Node

- "g" (hashD)
- "verb" stored at "do"

hashD → Branch Node

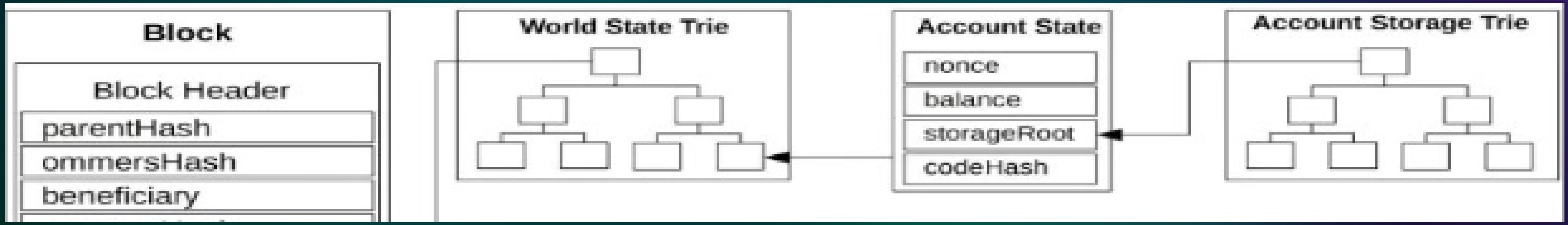
- "dog" → "puppy"
- "doge" → "coins"

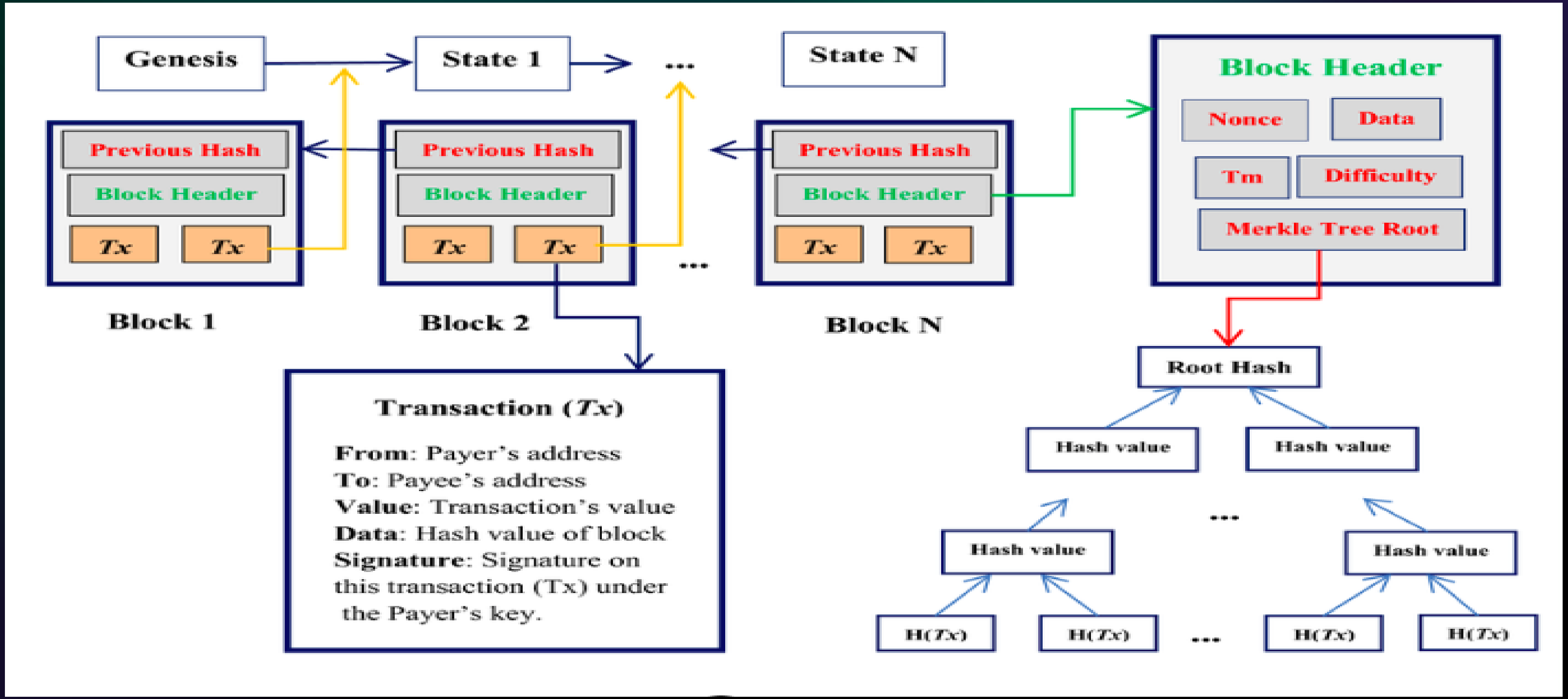
Types of tries in Ethereum

Types of Tries Comparative Table

Trie Type	Goal	Stored Data	Link to the block
State Trie	Tracks account balances and contracts' state	Account balance, contract code, storage, nonce	Located in the block header
Storage Trie	Stores smart contracts internal data	Contract variables and storage slots	Linked to specific contracts within blocks
Receipt Trie	Tracks transaction receipts	Transaction status, logs and gas usage	Located in the block body
Transaction Trie	Lists all the transactions in a block	Transaction hashes and the transaction's related details	Located in the block body

Representation of Ethereum Block





Advantages and challenges

Advantages

- **Efficient state management :**

  → Optimized organization and storage efficiency

- **Optimized updates and state changes :**

  → Efficiently handles frequent state changes

- **Verifiable and secure data integrity :**

  → Unauthorized modifications easily detectable

- **Efficient verification :**

  → Facilitates light clients

Challenges and gas cost

- **Maintenance complexity :**

  → Developpers can face challenges to maintain the structure

- **Storage bloat :**

  → State size grows significantly

- **Expensive trie updates :**

  → Gas fees on updates.

  → Example : writing to contract storage is costly because it updates the storage trie

Future improvements

Verkle Trees

Verkle Trees are a next-generation cryptographic data structure that use vector commitments instead of traditional hashing. This offers several advantages:

- Smaller proof sizes
- Faster verification
- Better scalability

Verkle Trees will replace MPT’s heavy storage with a more compact, scalable structure.

Conclusion

Conclusion

- **Efficient state management**
  - Combine Merkle trees and Patricia Tries for secure, optimized storage
- **Challenges**
  - Growing state size, high gas cost
- **Future solutions**
  - Verkle trees & stateless Ethereum

THANK YOU FOR YOUR ATTENTION

References

Ethereum documentation: https://ethereum.org/en/developers/docs/data-structures-and-encoding/patricia-merkle-trie/

https://www.lucassaldanha.com/ethereum-yellow-paper-walkthrough-2/

https://medium.com/shyft-network/understanding-trie-databases-in-ethereum-9f03d2c3325d

Consensys: https://consensys.io/blog/ethereum-explained-merkle-trees-world-state-transactions-and-more

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

W3R one : https://w3r.one/fr/blog/blockchain-web3/architecture-blockchain/markle-trees-structures-donnees/trie-patricia-et-ethereum-gestion-optimisee-etats