

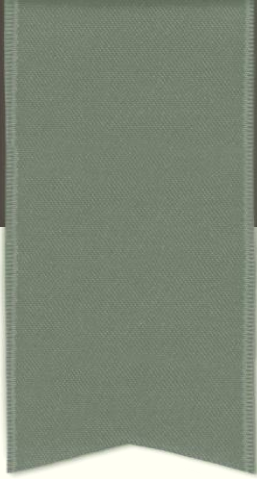
ETHEREUM

Blockchain technologies, lecture 5



Course overview

- Ethereum -- transaction-based state machine
- Merkle-Patricia Trie
- Block structure, Transactions
- Ethereum accounts
 - Smart contracts
- EVM and code execution



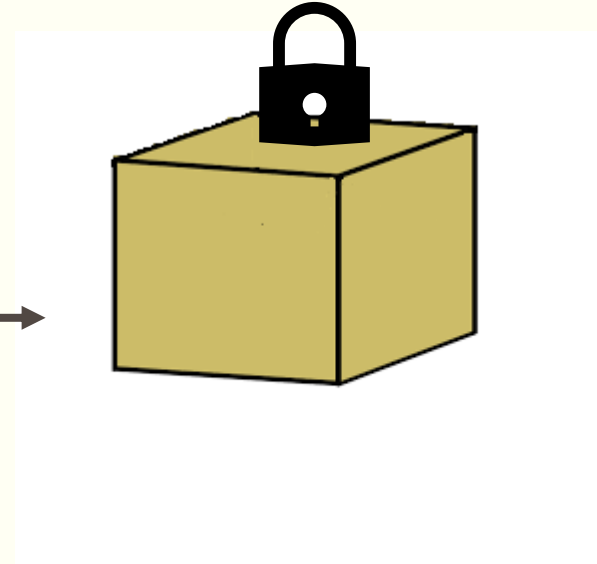
MERKLE PROOFS

MERKLE-PATRICIA TRIE

- **MERKLE TREE** or hash trees are named after Ralph Merkle, 1979.
- Every leaf node is labelled with the cryptographic hash of a data block.
- Every non-leaf node is labelled with the cryptographic hash of the labels of its child nodes.
- Allows efficient and secure verification of large data
- Example of **commitment scheme**.

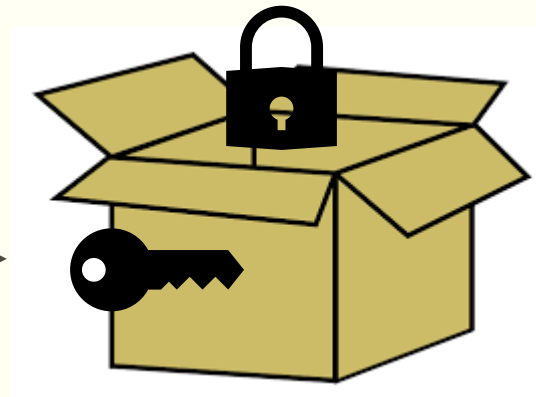
ALICE

COMMIT (binding property)

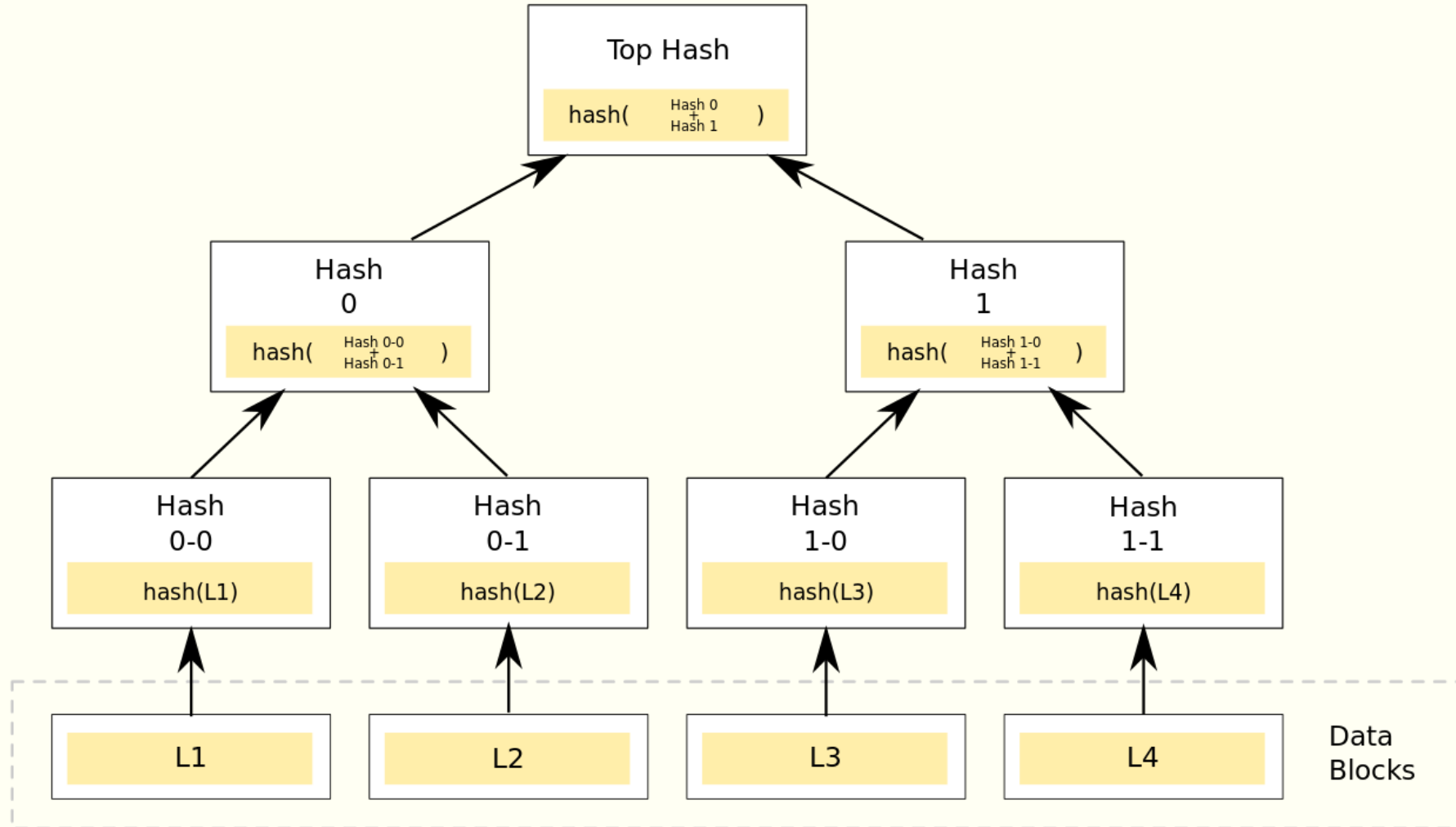


BOB

REVEAL (hiding property)



MERKLE-PATRICIA TREES – hash trees

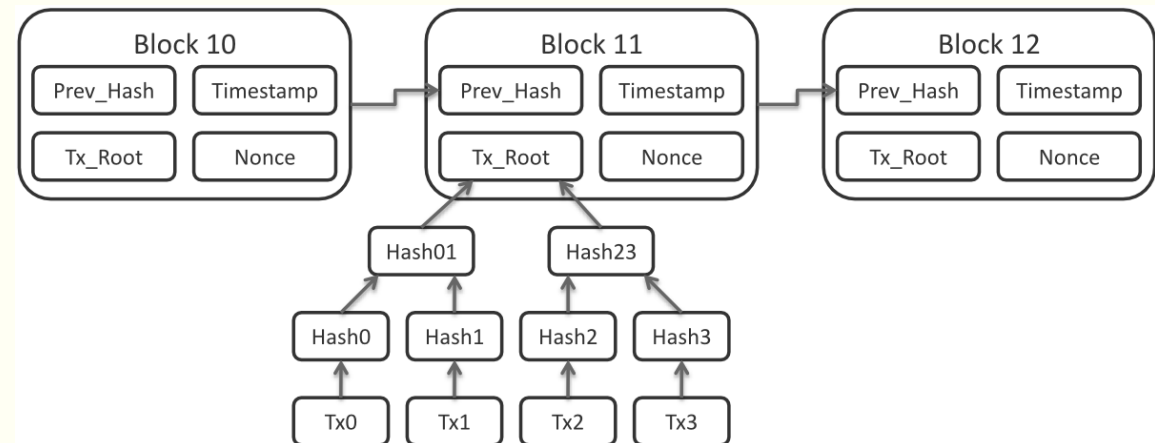


MERKLE PROOFS BITCOIN -- LightClients

Light clients download only block headers.

80-bytes chunks of data for each block that contain:

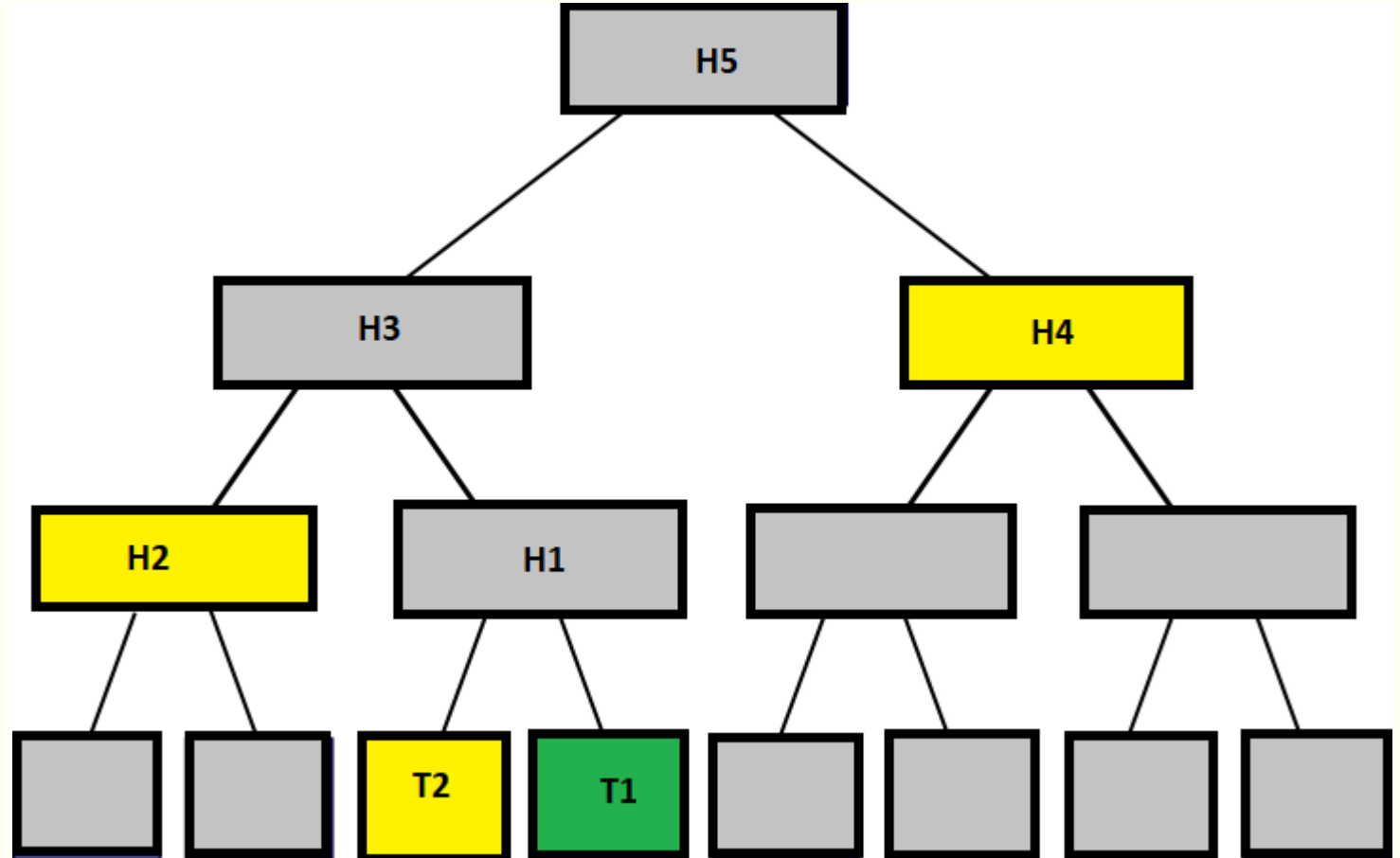
- Previous block hash
- Timestamp
- Mining difficulty
- PoW None
- Root hash for Merkle tree of transactions



MERKLE PROOFS BITCOIN -- LightClients

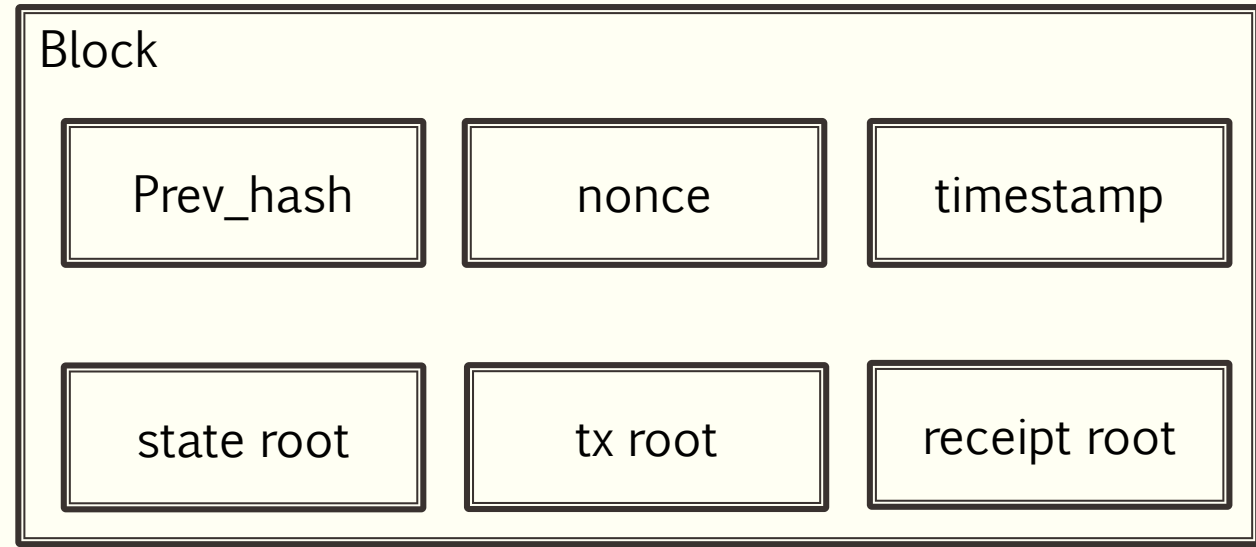
Light clients can be used to determine the status of a transaction:

- Ask for a Merkle proof showing that a transaction is in one of the Merkle trees.
- A Merkle proof consists of a chunk, the root hash of the Merkle tree, and the “branch” consisting of all of the hashes on the path from the chunk to the root.
- Bitcoin proof not enough to know current state (balance, asset current holder etc.). One must authenticate and add all transactions.



MERKLE PROOFS ETHEREUM -- LightClients

- Is the transaction included in a block (tx root)?
 - Does account X exists?
 - What is the current balance of the account X?
 - All events of type X?
 - What will be the outcome of a certain transaction, effect on balances?
-
- Transaction tree immutable.
 - **State-tree** updates: new accounts, update balance, update storage for smart contracts etc.





ETHEREUM STATE MACHINE

Ethereum state machine

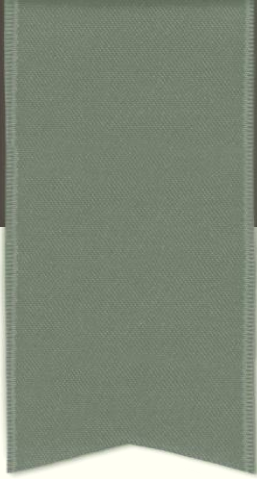
- Ethereum – **transaction-based state machine**
- Turning complete (pseudo-complete due to gas limits)
- State transition function: $Y(S,T) = S'$
 - State is a modified Merkle Patricia Tree storing hashes of all accounts.
- EVM code execution – stack machine
 - OP CODES: ADD, SUB etc. ADDRESS, BALANCE, BLOCKHASH
 - Each operation has a gas cost

EVM - STORAGE

- Any contract data must be assigned to a location (**storage** or **memory**)
- Each account has a data area called storage.
- A contract can only read or write to its own storage.
- Key-value store. Maps 256-bit words to 256-bit words.
- Values stored in storage are stored permanently on the blockchain.
- Modify storage is costly and should be avoided.

EVM – MEMORY AND STACK

- **Memory** -- Values only stored for the lifetime of a contract function's execution.
- Cheaper, not permanently stored.
- Writes can be either 8 bits or 256 bits wide.
- .
- **Stack** maximum size of 1024 elements and contains words of 256 bits
- Swap between top 16 elements and top, copy from top 16 elements, operation with topmost two elements.
- Each operation has a gas cost

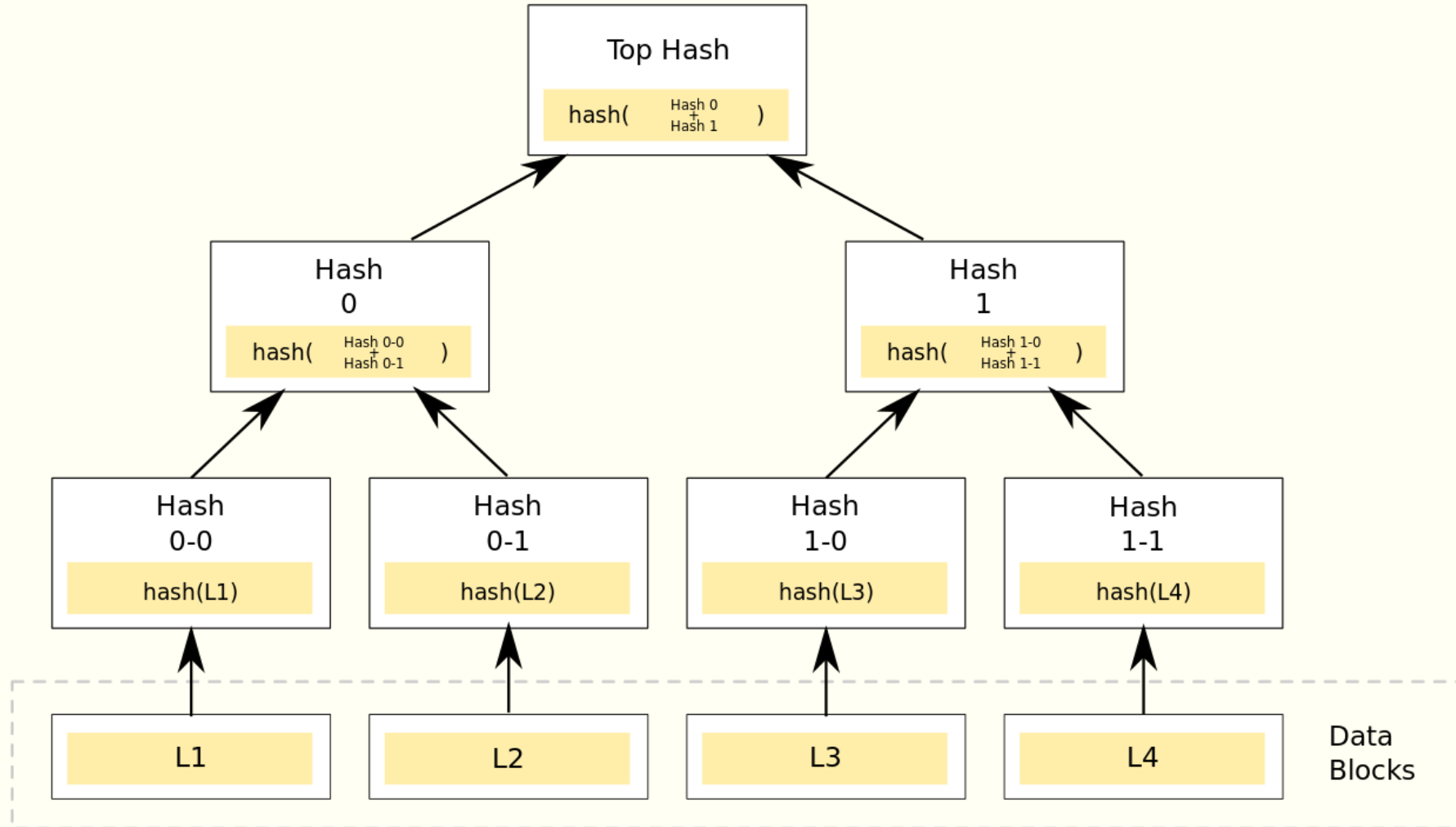


MERKLE-PATRICIA TRIE

MERKLE-PATRICIA TRIE

- **MERKLE TREE** or hash trees are named after Ralph Merkle, 1979.
- Every leaf node is labelled with the cryptographic hash of a data block.
- Every non-leaf node is labelled with the cryptographic hash of the labels of its child nodes.
- Allows efficient and secure verification of large data
- Example of **commitment scheme**.

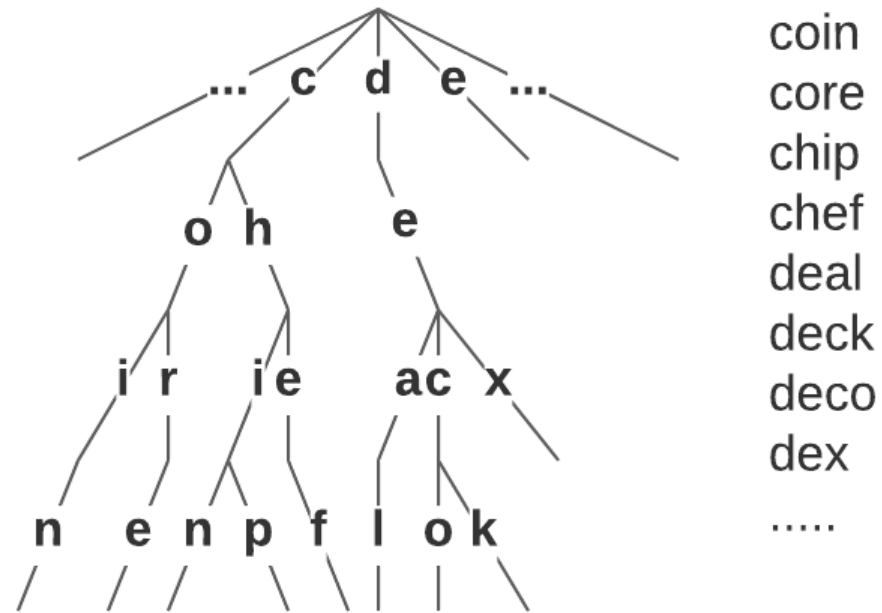
MERKLE-PATRICIA TRIE – hash trees



MERKLE-PATRICIA TRIE

- **PATRICIA** derived from the Latin word patrician, meaning "noble".

Practical Algorithm to Retrieve Information Coded in Alphanumeric, D.R.Morrison (1968)

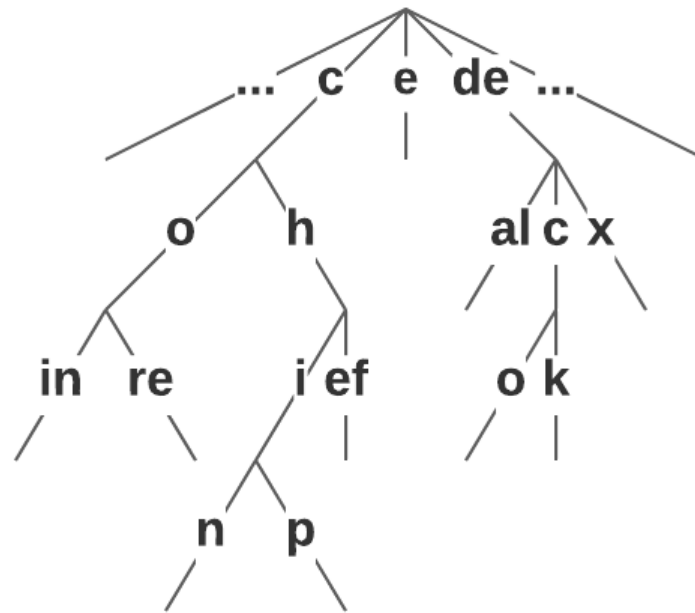


coin
core
chip
chef
deal
deck
deco
dex
.....

TRIE

MERKLE-PATRICKA TRIE

- PATRICIA optimized trie - each node that is the only child is merged with its parent.



coin
core
chip
chef
deal
deck
deco
dex
.....

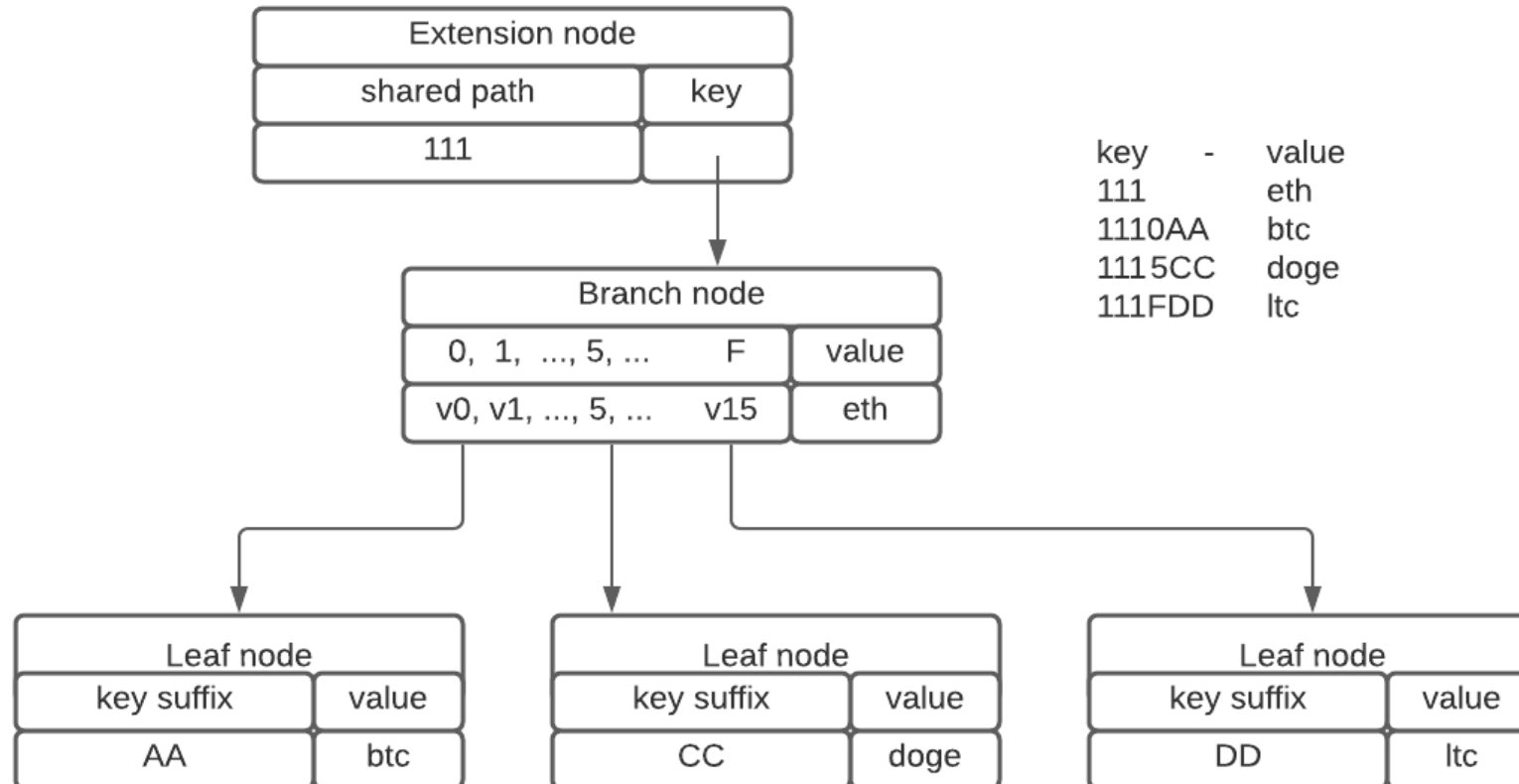
PATRICIA-TRIE

MERKLE-PATRICIA TREES

- **Hashing in the sense of MERKLE tree**, a child node is referred by its cryptographic hash.
- **Group common prefixes in the sense of PATRICIA tries**, branches only when branching is necessary.
- Allows 16 branches.
- Three types of nodes:
 - Branch node $[i_0, i_1, \dots, i_{15}, \text{value}]$ The i -position contains a link to a child node.
The 17th item (value) is used only for terminal nodes
 - Extension $[\text{path}, \text{value}]$ Compression, stores common *path* for multiple keys, *value* links to a child node.
 - Leaf $[\text{path}, \text{value}]$ Compression, terminal node.

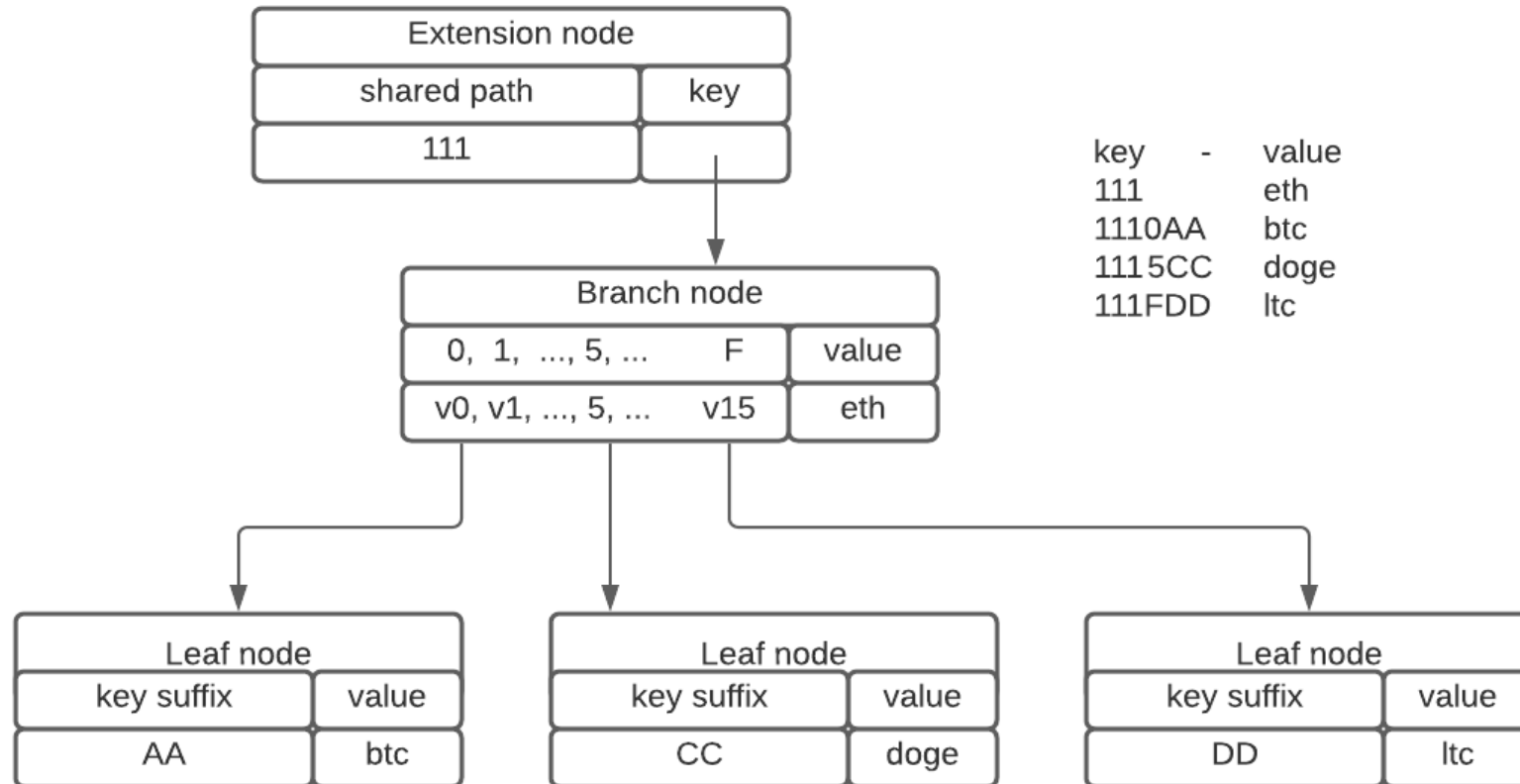
MERKLE-PATRICKIA TREES

- **MARKLE** PATRICIA TRIE - Group common prefixes in the sense of PATRICIA tries.



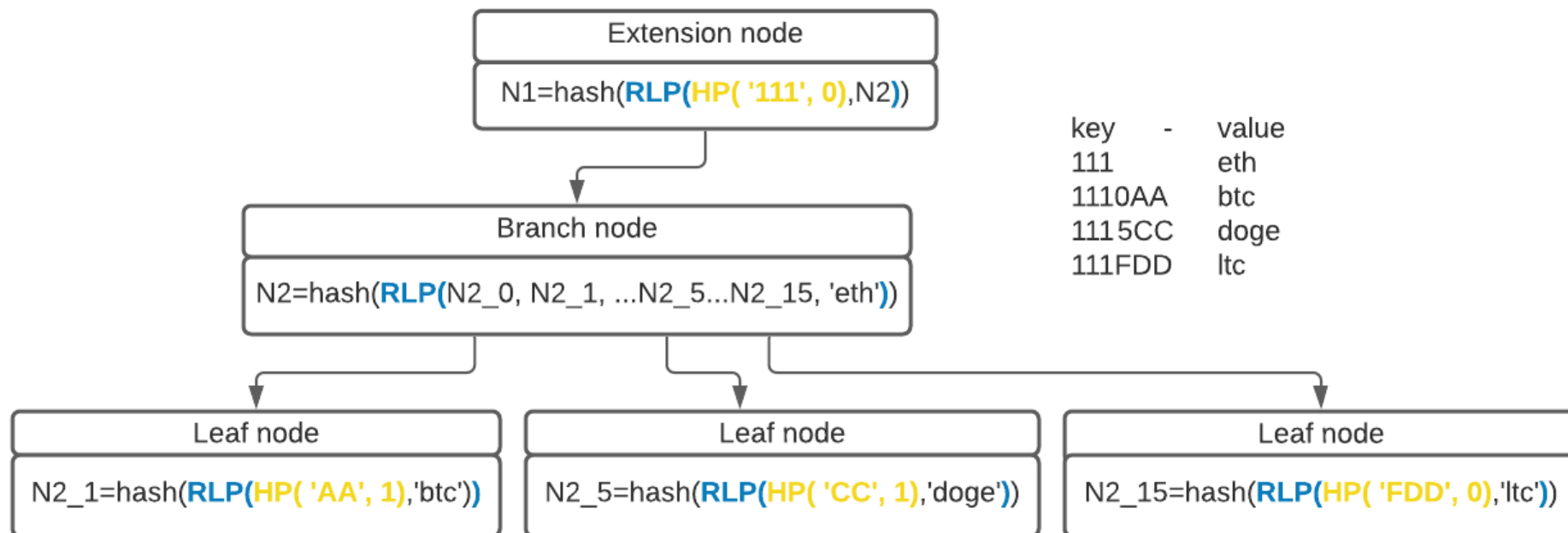
MERKLE-PATRICKIA TREES

- **MARKLE** PATRICIA TREE - Group common prefixes in the sense of PATRICIA tries.
- use key-value storages, RocksDB (Parity) LevelDB (Geth)



MERKLE-PATRICIA TREES

- **MARKLE** PATRICIA TREE - a child node is referred by its **cryptographic hash**.
- RLP function, serializes a set of input arrays into one array.
- HP encodes node structures into compressed byte arrays.
- hash = keccak implementation of SHA3



MERKLE-PATRICIA TREES Hex Prefix encoding

- $f(t) = \begin{cases} 2, & t = 1 \\ 0, & t = 0 \end{cases}$ HP encodes a path stream so that two (4-bit) nibbles compose one (8-bit) byte.
- $$\text{HP}(x, t) = \begin{cases} (16f(t), 16x[0] \oplus x[1], 16x[2] \oplus x[3], \dots), & \|x\| \text{ is even} \\ (16(f(t) + 1) + x[0], 16x[1] + x[2], 16x[3] + x[4], \dots), & \|x\| \text{ is odd} \end{cases}$$

t	$\ x\ $	x	prefix	X[0] or 0	Grup 1	Grup 2
t=0 extension	even	0xa, 0xb, 0xc, 0xd	0000	0000	1010 1011	1100 1101
	odd	0x09, 0xa, 0xb, 0xc, 0xd	0001	1001	1010 1011	1100 1101
t=1, leaf	even	0xa, 0xb, 0xc, 0xd	0010	0000	1010 1011	1100 1101
	odd	0x09, 0xa, 0xb, 0xc, 0xd	0011	1001	1010 1011	1100 1101

MERKLE-PATRICIA TREES **Recursive Length Prefix**

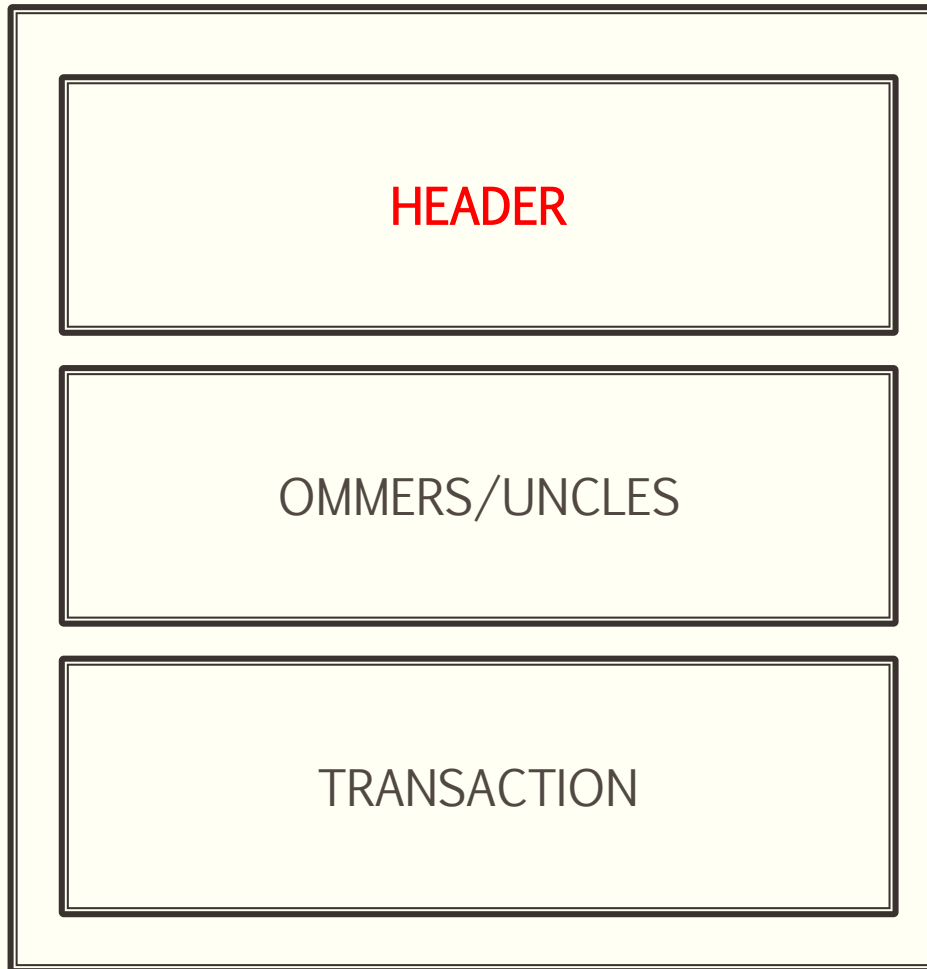
- RLP function, serializes (flattens) a set of input arrays into one byte array.
- Flattened byte array is persisted into a key-value storage.
- **Length Prefix** Every sub-sequence is prefix by a byte encoding its length and a bit-mask to determine the sub-subsequence type: array or string.
- **Recursive** Every flattened sequence is prefixed by total length of all its sub-sequences.

x	$\ x\ $	$RLP(x)$	$RLP(x)$	$\ RLP(x)\ $
XYZ	3	$128 + \ x\ $, X, Y, Z	1000 0011 X Y Z	4
AB	2	$128 + \ x\ $, A, B	1000 0010 A B	3
[XYZ, AB]	2	$192 + \ RLP(x_1), RLP(x_2)\ $, $RLP(x_1)$, $RLP(x_2)$	1100 0111 10000011 X Y Z 10000010 A B	8



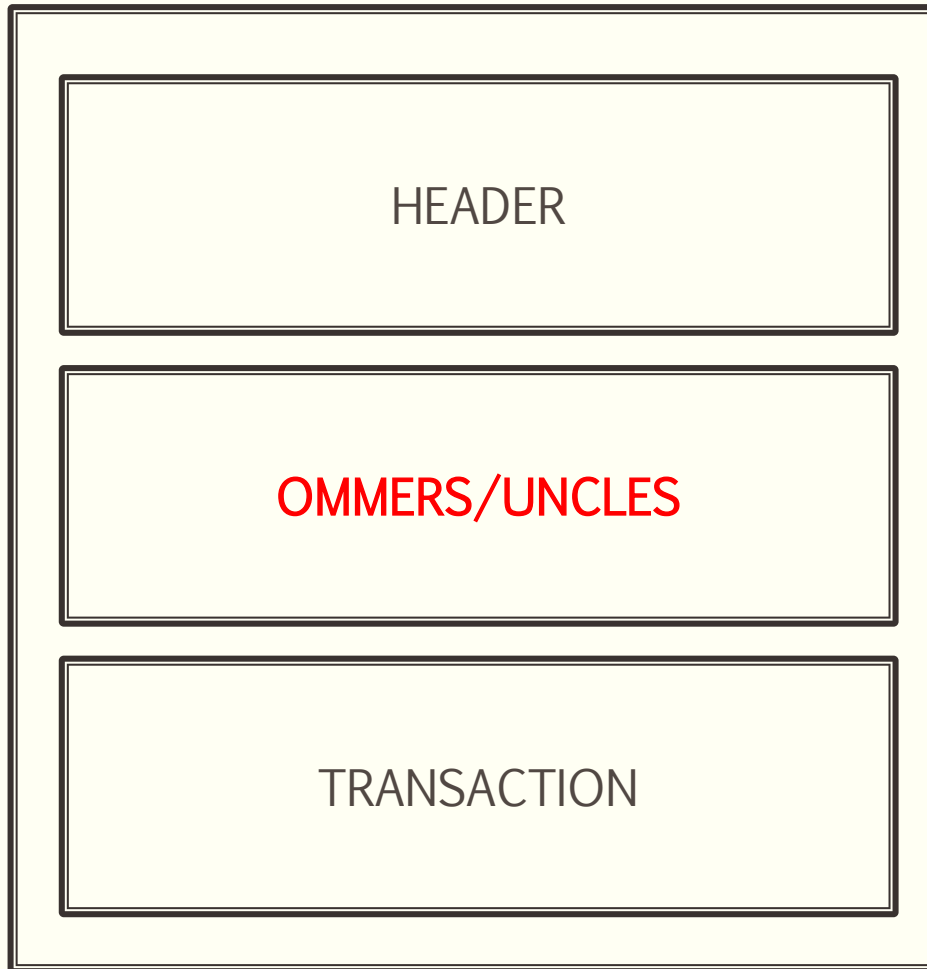
ETHEREUM BLOCK STRUCTURE

ETHEREUM BLOCK STRUCTURE



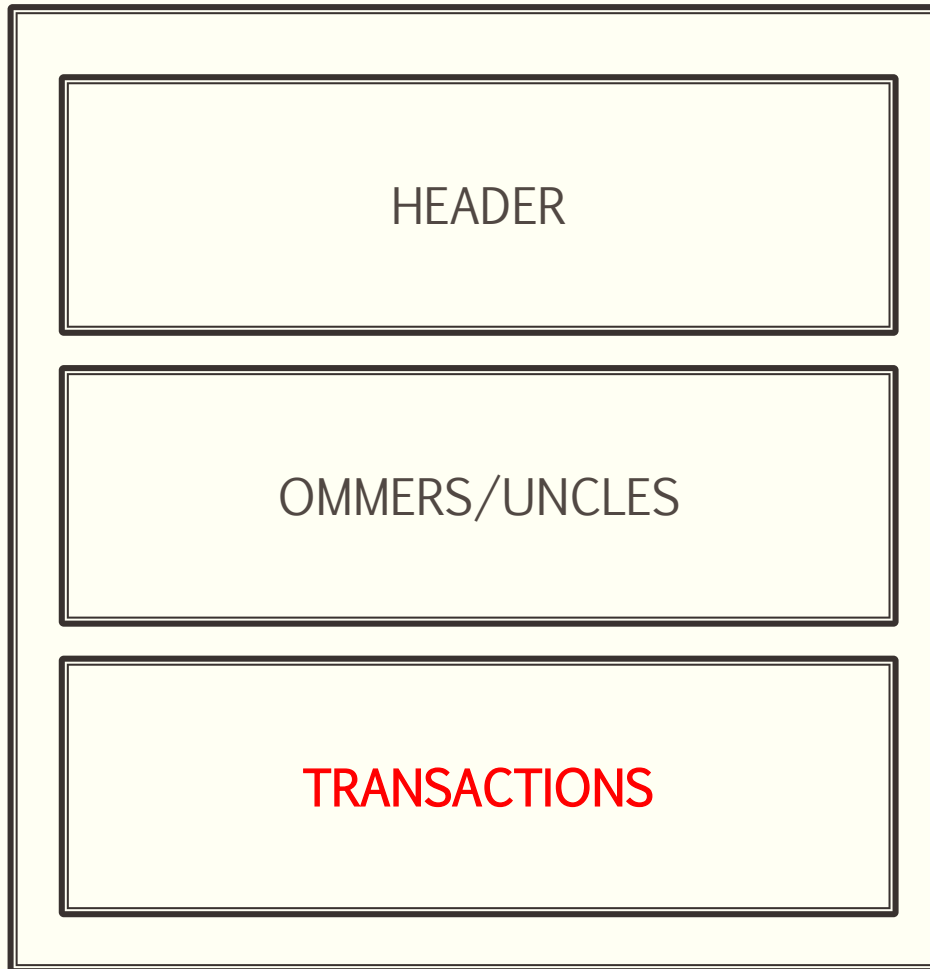
- Transactions, usually representing funds transfers, are grouped in blocks.
- Blocks are linked via cryptographic hashes.
- Block **header** stores the *hash of the previous block* in the chain.
- Miners wrap transactions into blocks and compete to append a new block into the chain via PoW protocol.

ETHEREUM BLOCK STRUCTURE



- If more blocks are produced at the same time, blocks that are not included in the mainline become ommers, with a smaller reward for miners.
- A new block is added in the mainchain approximately every 15s. Only a number of 6 consecutive ommers is rewarded, after one minute a half the miner will be notice about the consensus and be incentivized to join the mainline.
- **!!! Not used in the PoS protocol**

ETHEREUM BLOCK STRUCTURE



- **Transactions** are stored in MERKLE PATRICIA TRIE structures.
- In Ethereum state of the accounts (world state) is also stored, reflecting balances as a result of transactions execution.
- **World state** is also stored in a MERKLE PATRICIA TRIE.

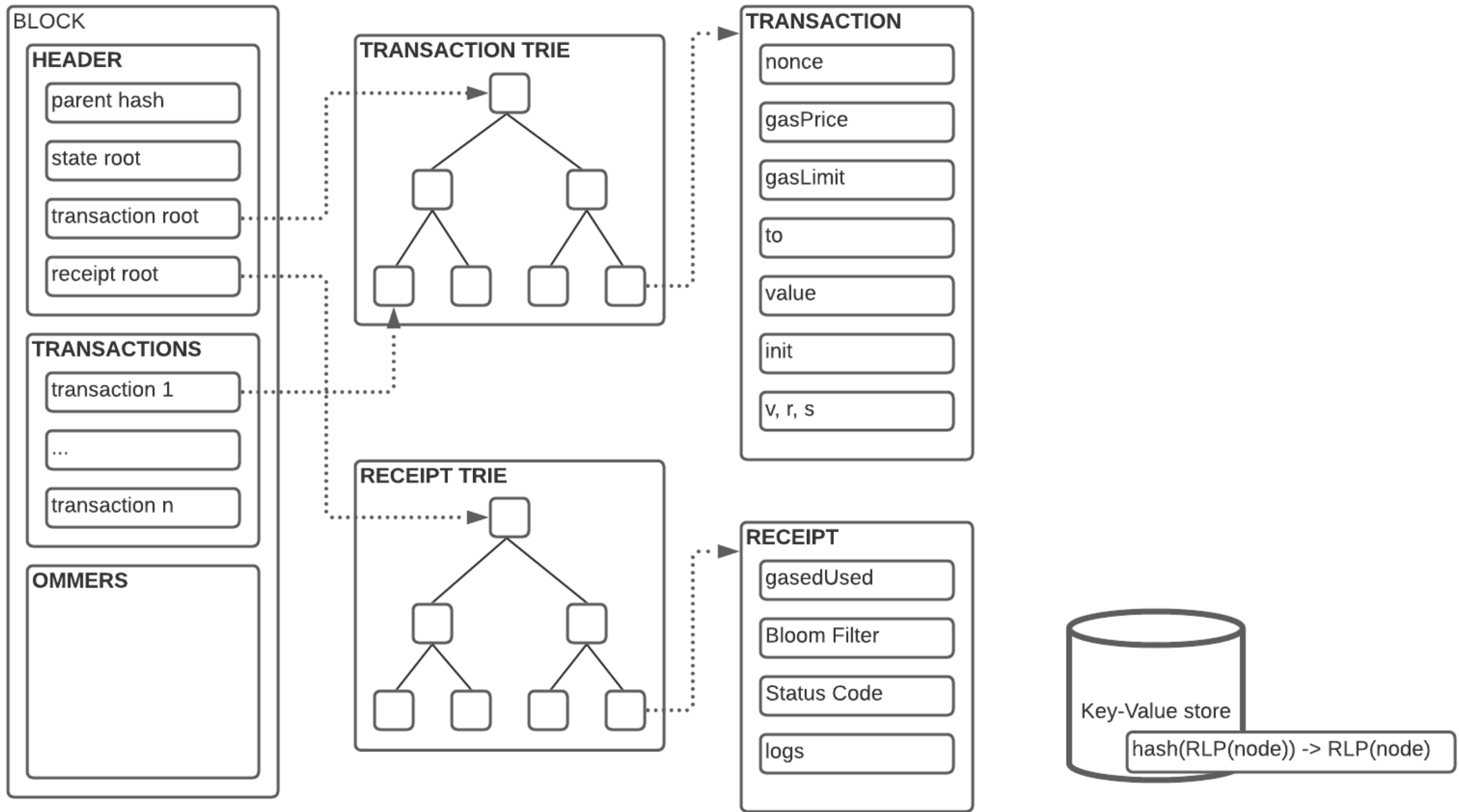
ETHEREUM BLOCK STRUCTURE -BLOCK HEADER

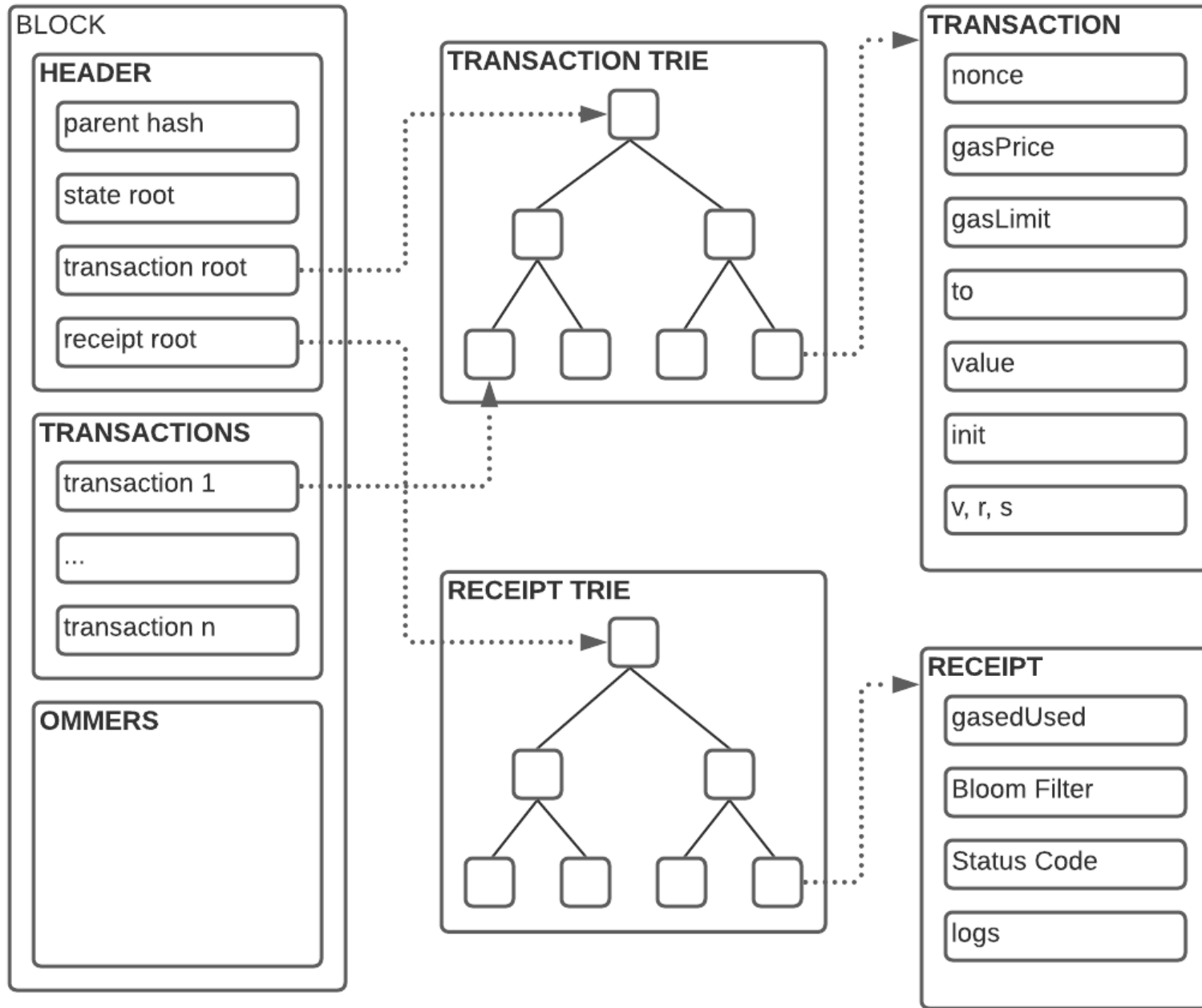
▪ parentHash	Keccak 256 bit hash of the parent block	
▪ nonce	64 bit value found by miner	
▪ mixHash	hash confirming mining	PoW consensus
▪ difficulty	scalar value measuring mining effort	
▪ ommersHash	Keccak 256 bit hash of list of ommers	
▪ Beneficiary	address of miner receiving reward	
▪ stateRoot	Keccak 256 bit hash of a root of the World State Trie	
▪ receiptsRoot	Keccak 256 bit hash of a root of the Transaction Receipt Trie	
▪ transactionsRoot	Keccak 256 bit hash of a root of the Transaction Trie	
▪ logsBloom	Bloom filter relating logs hashes with the logs	
▪ number	ordinal number of this block. every new block gets a number increased by one.	
▪ gasLimit	accumulated gas limit required to process all transactions	
▪ gasUsed	accumulated real consumed gas to process all transactions	
▪ extraData	32 bytes additional data	
▪ timestamp	scalar value equal to the reasonable output of Unix's time() at this block's inception	

ETHEREUM BLOCK STRUCTURE -BLOCK HEADER

▪ parentHash	Keccak 256 bit hash of the parent block
▪ nonce	64 bit value found by miner
▪ mixHash	hash confirming mining
▪ difficulty	scalar value measuring mining effort
▪ omersHash	Keccak 256 bit hash of list of omers
▪ beneficiary	address of miner receiving reward
▪ stateRoot	Keccak 256 bit hash of a root of the World State Trie
▪ receiptsRoot	Keccak 256 bit hash of a root of the Transaction Receipt Trie
▪ transactionsRoot	Keccak 256 bit hash of a root of the Transaction Trie
▪ logsBloom	Bloom filter relating logs hashes with the logs
▪ number	ordinal number of this block. every new block gets a number increased by one.
▪ gasLimit	accumulated gas limit required to process all transactions
▪ gasUsed	accumulated real consumed gas to process all transactions
▪ extraData	32 bytes additional data
▪ timestamp	scalar value equal to the reasonable output of Unix's time() at this block's inception

[Merkle Patricia Trie](#)





Receipts – effect of transactions

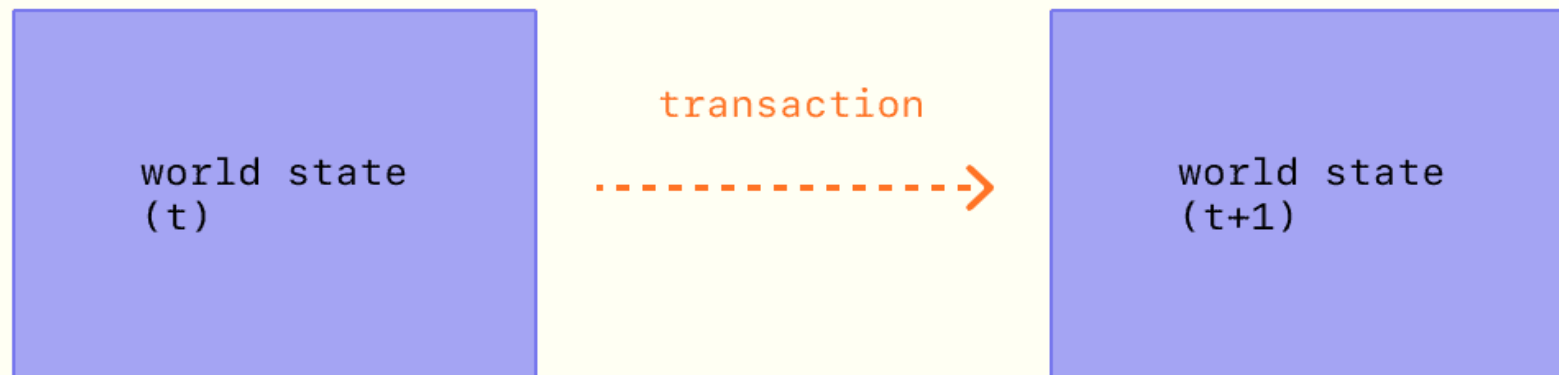
check log hash
bloom filter – false positives



ETHEREUM TRANSACTIONS AND GAS

ETHEREUM TRANSACTION

- A transaction is initiated by an externally-owned account
- The result of the transaction is changing world state.
- Lifecycle:
 - Generate transaction hash
 - Broadcast to the network, include transaction in transaction pool
 - Miner include transaction in a block
 - Transaction receive confirmations



ETHEREUM TRANSACTION

- Two types of transactions: **contract creation** (resulting in a new contract account containing compiled smart contract bytecode) and **message calls**.
- **Nonce** scalar value equal to the number of transactions sent by the sender
- **gasPrice** scalar value equal to the number of Wei(10^{-18}) to be paid per unit of gas.
- **gasLimit** scalar value equal to the maximum amount of gas that should be used when executing the transaction
- **to** 160-bit address of the message call's recipient or null.
- **value** number of Wei to be transferred.
- **init** unlimited size byte array, EVM code.
- **data** unlimited size byte array, input data.
- **signature** confirms that sender has authorize transaction

EVM - GAS

- Measures the amount of computational effort required to execute operations.
 - “Fuel” for Ethereum network – not currency!
- Each operation costs a specific amount of gas.
- For each transaction sender specify a “gas limit”, maximum amount he is willing to pay for the transaction to be processed.
- Gas limit is implicitly purchased from the sender’s account balance.
- The transaction is considered invalid if the account balance cannot support paying gas limit.
- Gas can be partially return to the sender.
- If transaction runs out of gas, then it’s reverted back to its original state.

ETHEREUM TRANSACTION

- Transaction validity
 - Transaction is well formed (RLP), no additional bytes.
 - Transaction signature is valid.
 - Transaction nonce is valid, equivalent to the sender's current nonce.
 - Gas limit is smaller than gas used by the transaction.
 - The sender account balance contains at least the cost required to pay transaction.
- Gas limit * gas cost
- Gas limit * (base fee + tip)

EVM - GAS EIP 1559

- Before EIP 1559 Gas cost set by **first price auction system**
 - Everyone submit bid, miners select transactions with the highest fee.
 - Often users pay more than 5x than necessary.
-
- After EIP 1559 block has a **base fee** determined by the blocks before. If block size is greater than block limit increase fee, if is less then the block size, decrease fee.

EVM - GAS

- Before EIP 1559 block size had fixed-size
- After EIP 1559 block size is variable and will increase or decrease depending on the network demand up until the block limit (30 million gas)
- “Block gas limit” determines that amount of gas that can be spent per block.
- “Each block has a target size of 15 million gas, but the size of blocks will increase or decrease in accordance with network demand, up until the block limit of 30 million gas.”
- Increase or decrease base fees.

EVM – GAS EIP 1559

- **Ethereum Improvement Proposals (EIP)** describe standards for the Ethereum platform
 - Core improvements requiring consensus fork, miner strategy changes.
 - Networking
 - ERC – contract standards (ERC-20, ERC-721) Ethereum Request for comment.
- (London Upgrade) August 5th, 2021
 - Better transaction fee estimation, predictable transaction fees
 - Quicker transaction inclusion
 - counteract the release of ETH by **burning a percentage of transaction fee**.
- Replace first auction system with **base fee**.
- **maxPriorityFeePerGas** amount of gas paid as tip

Bibliography

- Ethereum yellow paper <https://ethereum.github.io/yellowpaper/paper.pdf>
- Merkle in Ethereum, Vitalik Buterin <https://blog.ethereum.org/2015/11/15/merkle-in-ethereum/>
- Ethereum Patricia-tree <https://eth.wiki/en/fundamentals/patricia-tree>
- HP implementation and examples https://hexdocs.pm/hex_prefix/HexPrefix.html
- Ethereum EVM <https://ethereum.org/en/developers/docs/evm/>
- EIP standards <https://eips.ethereum.org/>
- EVM memory <https://docs.soliditylang.org/en/latest/introduction-to-smart-contracts.html?highlight=memory#storage-memory-and-the-stack>
- EVM gas <https://ethereum.org/en/developers/docs/gas/>
- <https://eth.wiki/en/concepts/ethash/ethash>
- https://commons.wikimedia.org/wiki/File:Hash_Tree.svg
- https://commons.wikimedia.org/wiki/File:Bitcoin_Block_Data.png