CSE446: Blockchain & Cryptocurrencies

Lecture – 14: Ethereum - 3

BRAC UNIVERSITY

Inspiring Excellence

Agenda

- Ethereum Transactions
- Gas and Gas price
- Ethereum Block

Ethereum DoS protection

- EVM code is “Turing complete", e.g., has loops and may not halt
- In order to ensure safety, a type of DoS protection, the concept of fee is used
- Ethereum utilises a “pay as you execute” model
    - Meaning users need to pay for both computation and storage of data
- This payment is collected by the respective miner

Ethereum gas and gas price

- Each transaction contains a "gas limit", and a "gas price" which is the price the sender will pay per unit of gas, denominated in ETH (currency)
- Along with sufficient ETH to pay the fee
- But, how much do you need to pay?
- It depends on how much unit of work, i.e. code execution, is carried out or the amount of data stored in the network
  - The unit of work in Ethereum is defined with the concept of Gas
  - Ether buys GAS to fuel the EVM, like hour(s) of labour

Ethereum gas and gas price

- Each operation in Ethereum utilises some amount of gas
- For example:
  - If you add two numbers, 3 gas is used
  - If you multiply two numbers, 5 gas is used
  - One hash call requires 30 gas
- Ethereum also uses gas for contract creation
- By calculating total operations, we can estimate how much gas might require
- It is difficult to know the exact gas required for contract execution
  - So, set a higher limit using the notion of gasLimit, that implies the maximum amount of gas you are willing to buy

Ethereum gas and gas price

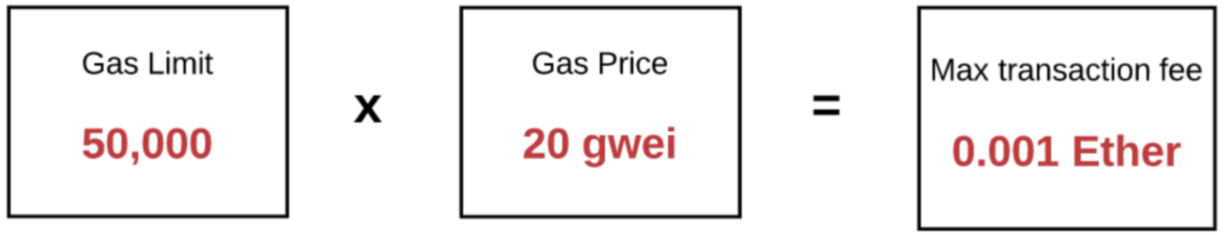
- But, still we don't know how much Ether we have to pay the miners
- That is determined by Gas Price, which is like the hourly labour rate: 500 taka/hour
- Fee for computation in Ether = gasLimit x gasPrice;
    - gasLimit (gas) and gasPrice are specified in the transaction
    - The calculated Fee is placed in Escrow
- Gas price is determined by the sender
- If gasPrice is too low, no miner will bother about processing a transaction

Ethereum gas and gas price

- If the fee is less than required, gas runs out before a transaction is completed
  - State is reverted back to the previous state
  - Fee is consumed from the escrowed Ether
- If execution completes
  - consumed gas x gasPrice is paid to the miner
  - Remaining gas (gasLimit – consumed gas) x gasPrice is returned back to the caller
- The higher the gas price you set, the sooner your transaction gets mined
  - Most miners tend to choose transactions with higher fee
- If price is relatively low, your transaction will eventually get included in the block but you might have to wait for a bit

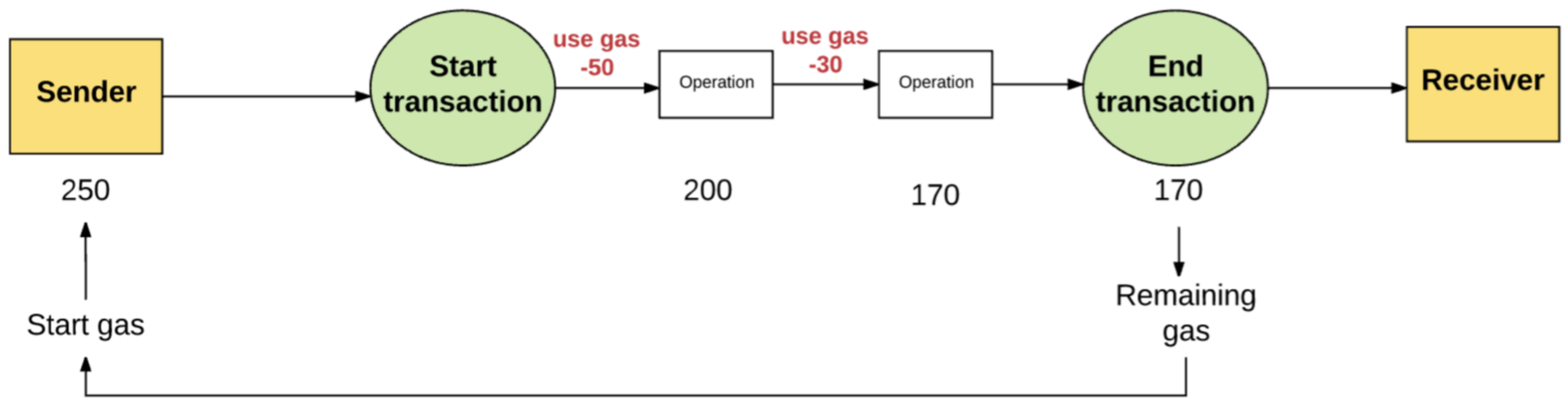
Ethereum gas and gas price

- Gas limit: Max no. of computational steps the transaction is allowed.
- Gas Price: Max fee the sender is willing to pay per computation step.



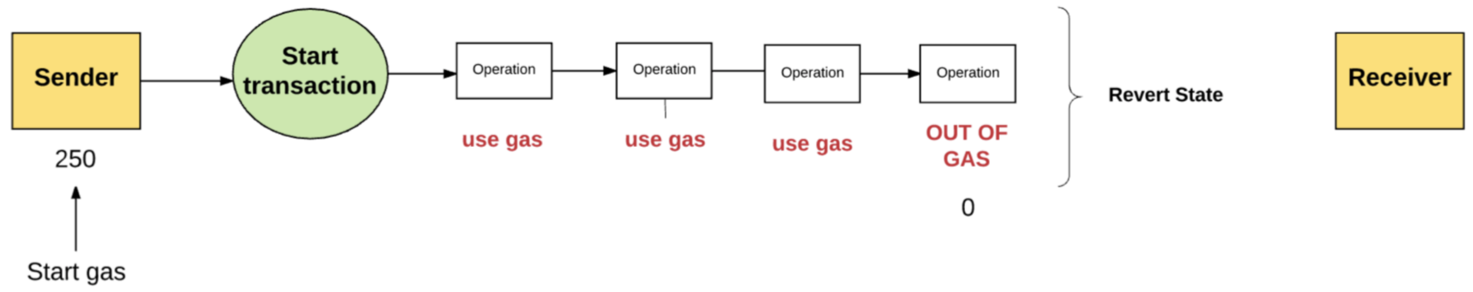
Ethereum gas and gas price

The sender is refunded for any unused gas at the end of the transaction.



Ethereum gas and gas price

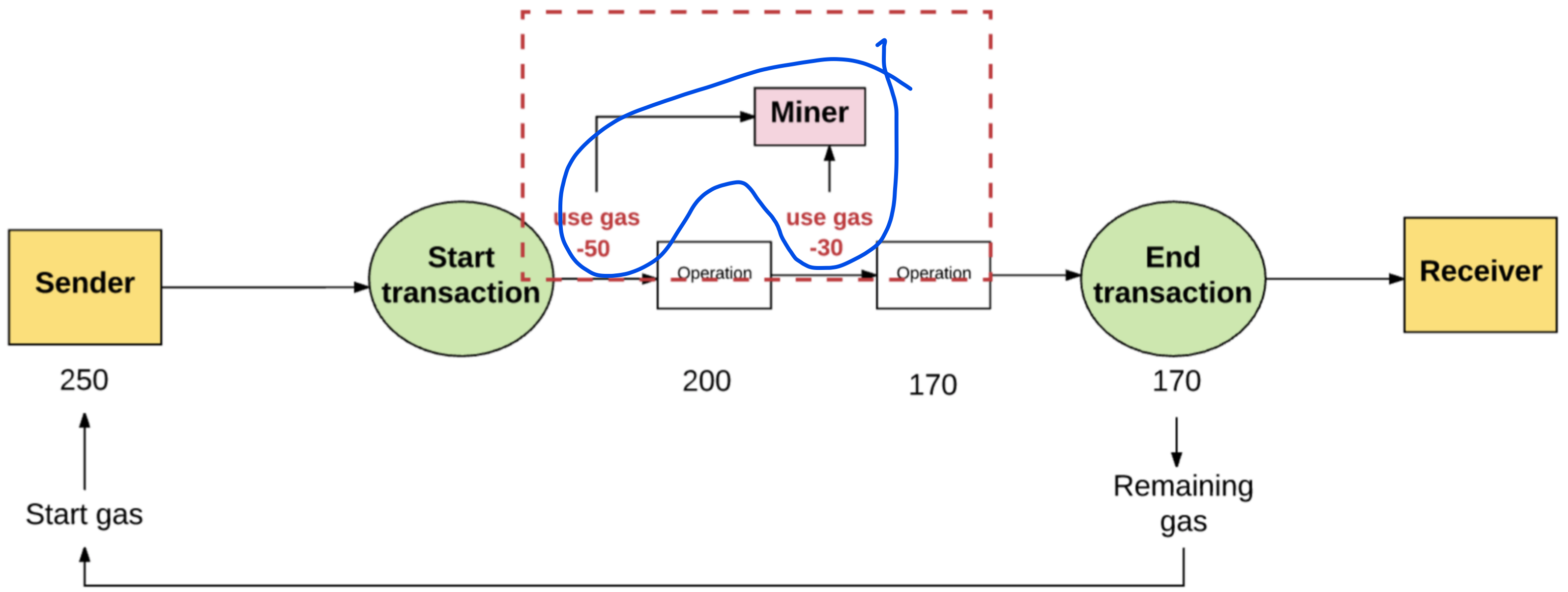
If sender does not provide the necessary gas to execute the transaction, the transaction runs “out of gas” and is considered invalid.



- The changes are reverted.
- None of the gas is refunded to the sender.

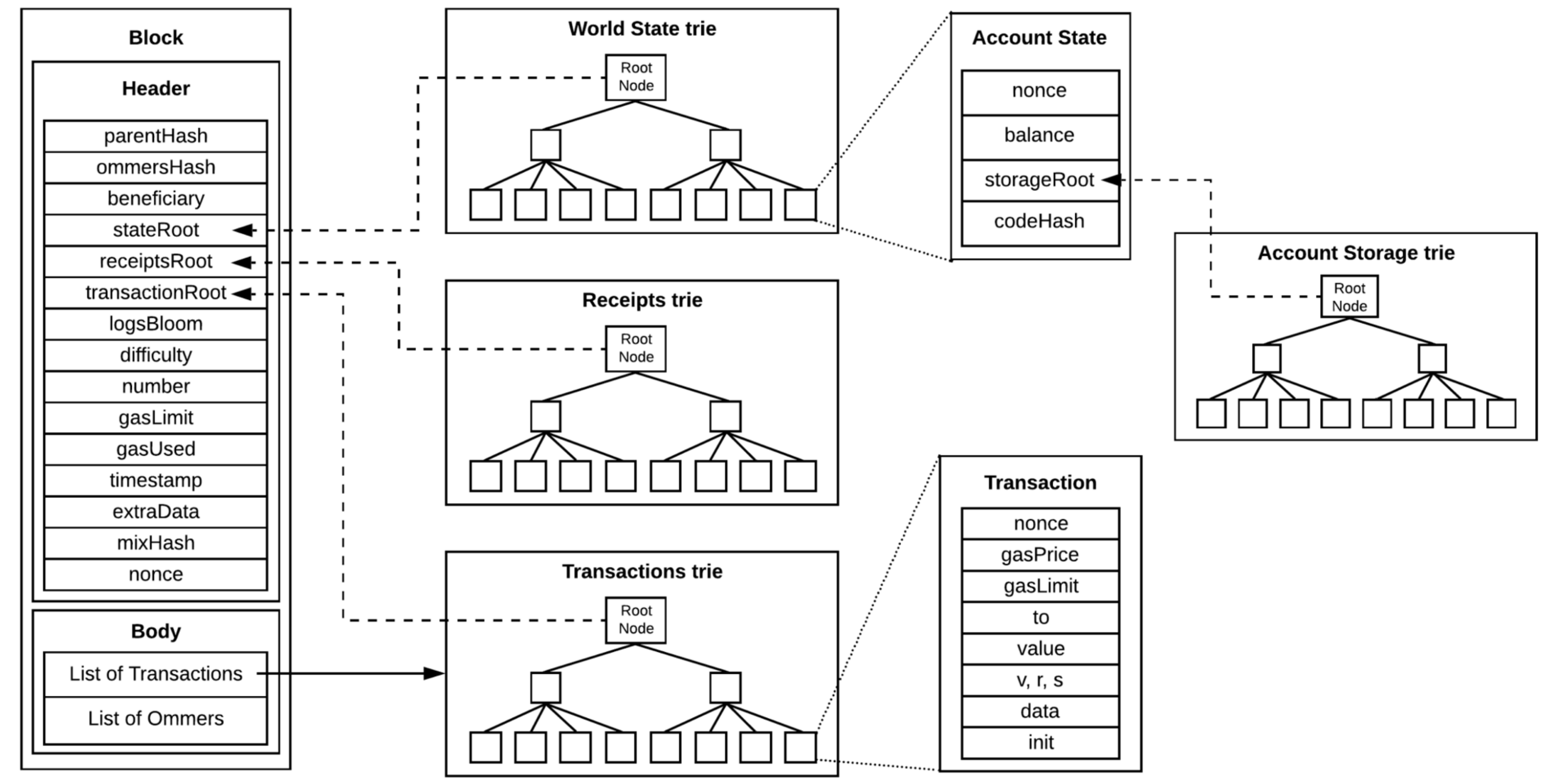
Ethereum gas and gas price

All the money spent on gas by the sender is sent to the miner's address.



From Lecture Slides of Blockchains & Cryptocurrencies Course at Johns Hopkins University

Ethereum block



Block, transaction, account state objects and Ethereum tries

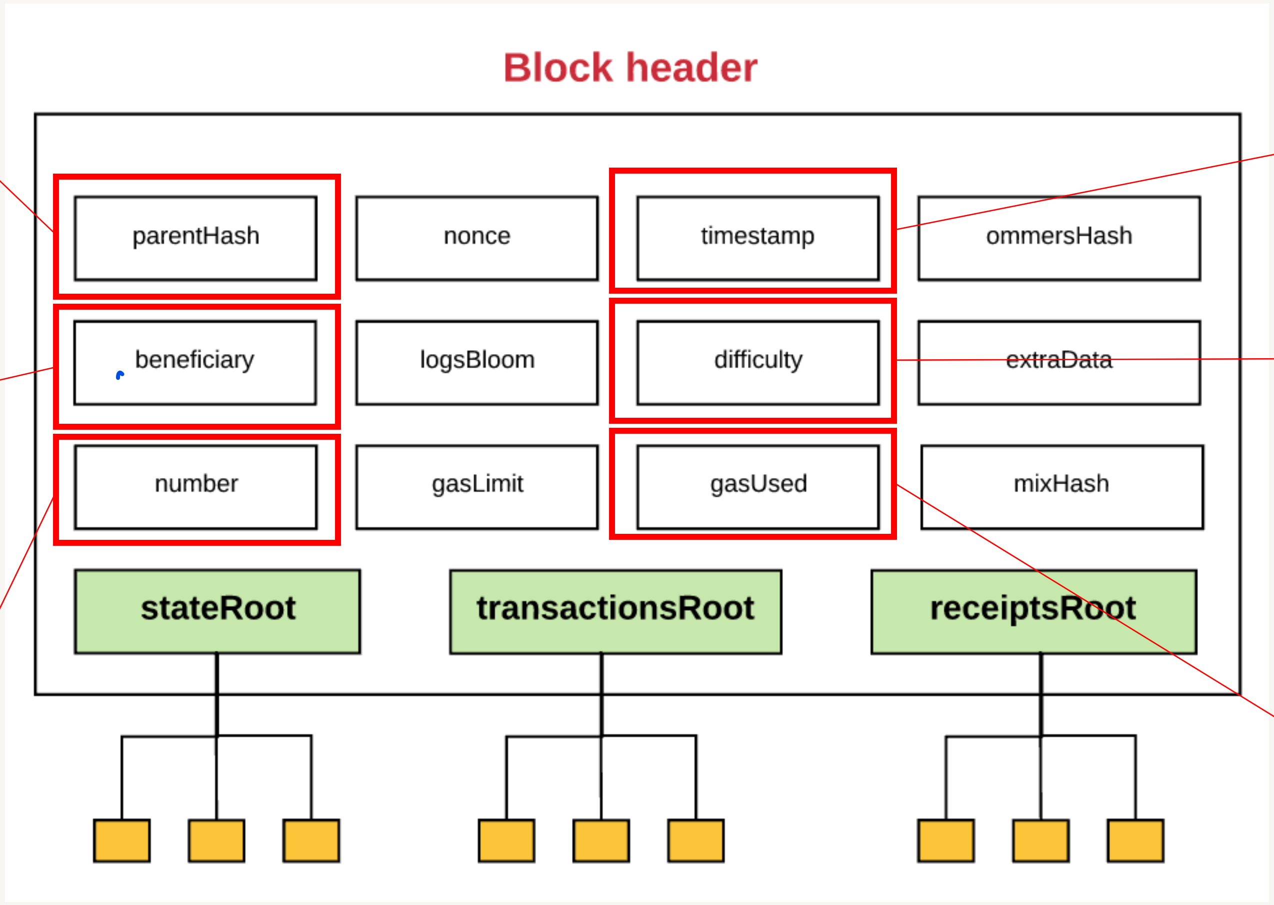
https://miro.medium.com/max/4800/1*BVvVSUIj4A9fmmVBNs4WhQ.png

Ethereum block header

A hash of the parent block's header

The account address that receives the fees for mining this block

The count of the current block



The time stamp of this block's inception

The difficulty level of this block

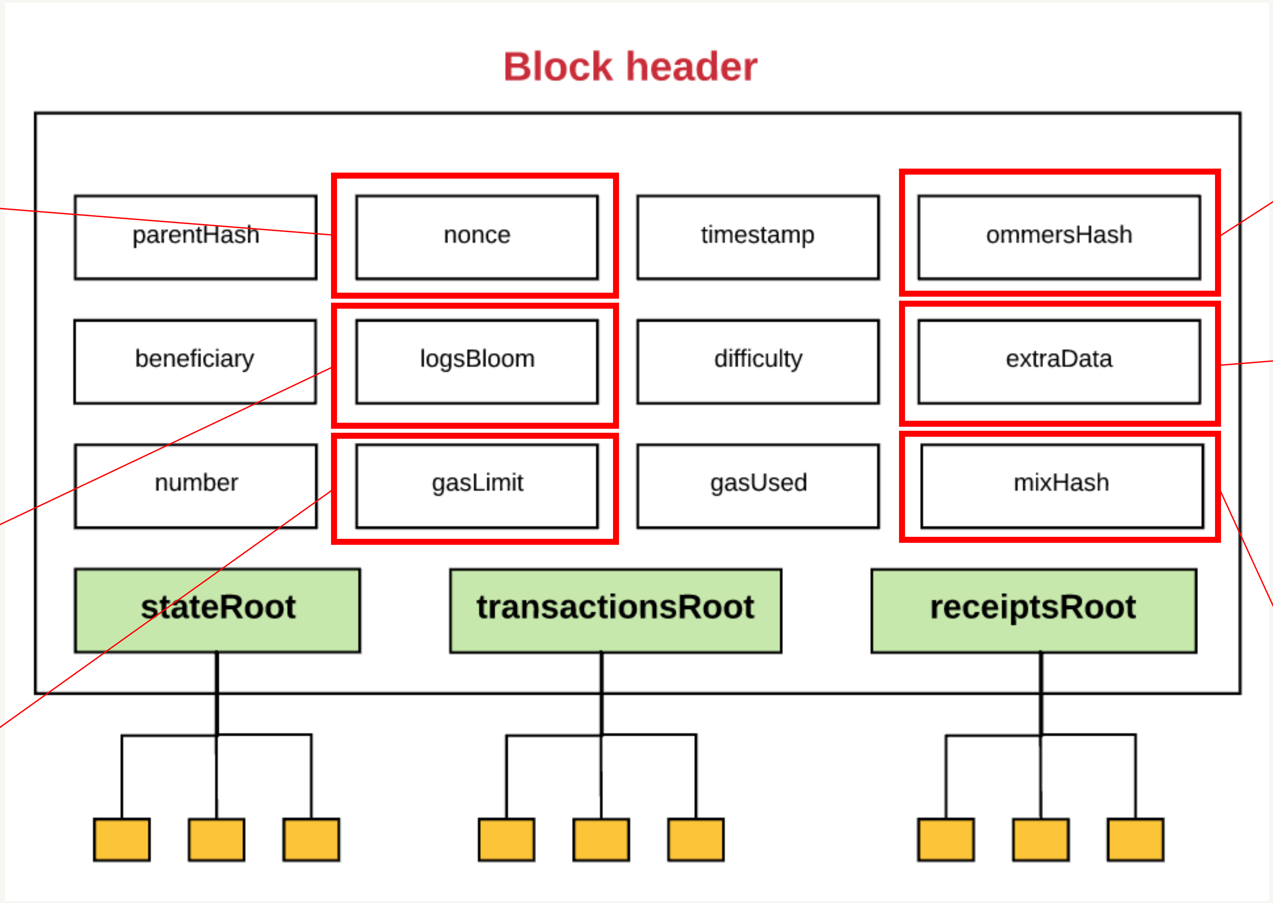
The sum of the total gas used by transactions in this block

Ethereum block header

A value that, when combined with the mixHash, proves that this block has carried out enough computation

A Bloom Filter(data structure) that consists of log information

The current gas limit per block



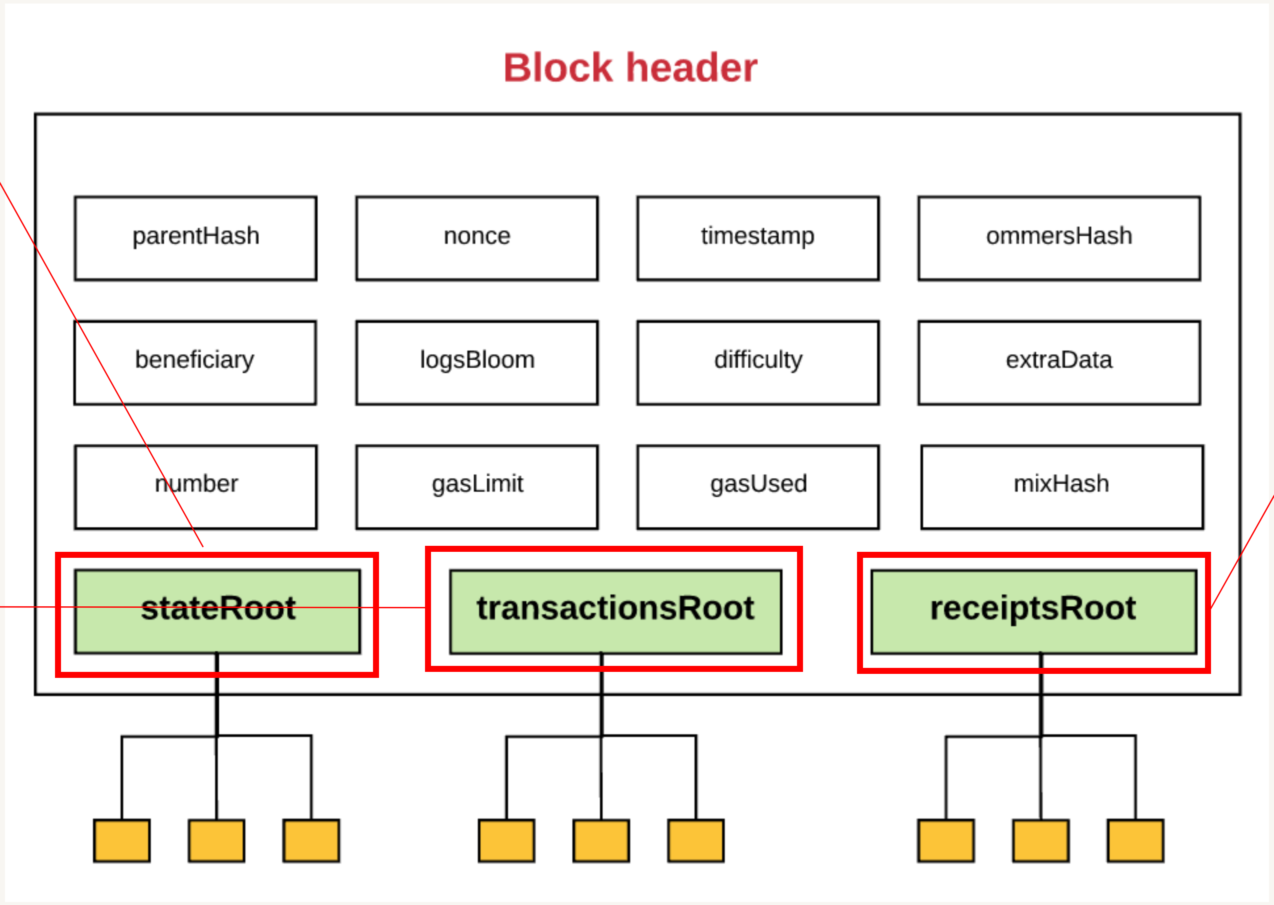
A hash of the current block's list of ommers

Extra data related to this block

A hash that, when combined with the nonce, proves that this block has carried out enough computation

Ethereum block header

The hash of the root node of the state tree



The hash of the root node of the tree that contains all transactions listed in this block

The hash of the root node of the tree that contains the receipts of all transactions listed in this block

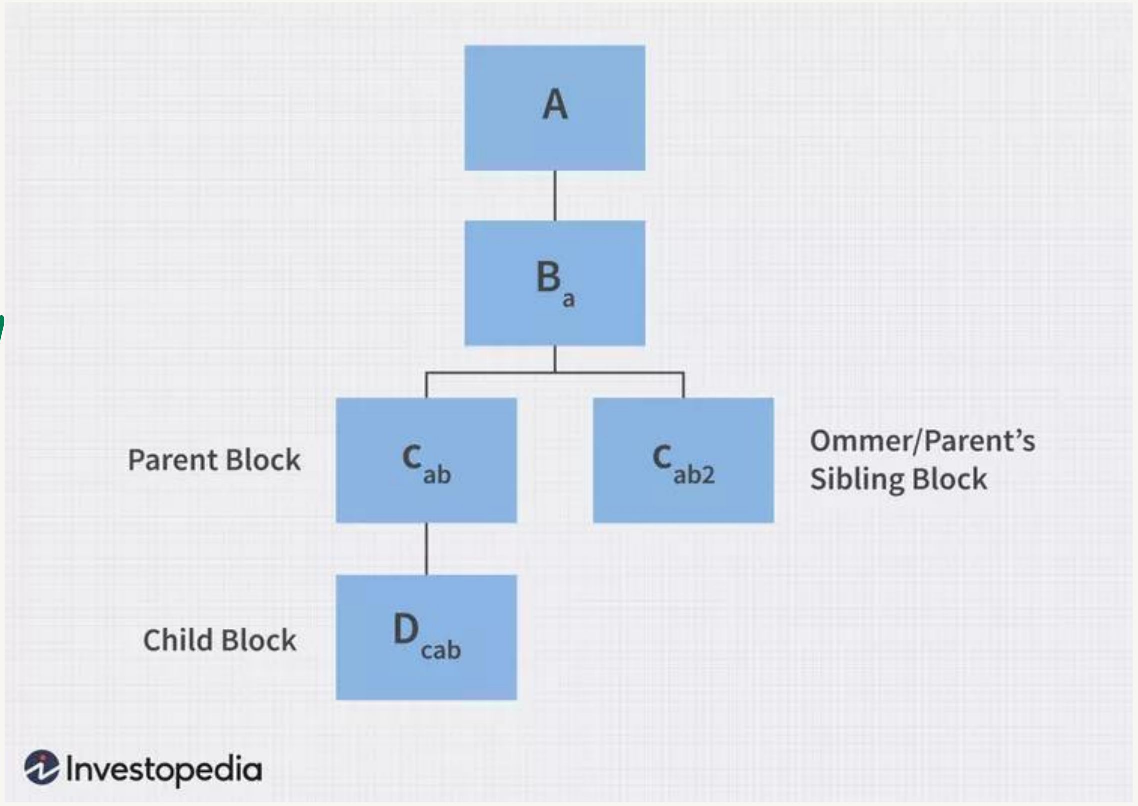
Ethereum ommers

- It is possible for two blocks to be created simultaneously by a network
- When this happens, a fork happens and eventually one block is left out
- This leftover block is called an ommer block
- In the past, they were called uncle blocks
  - referring to the familial relationships used to describe block positions within a blockchain
- In Bitcoin, there is no reward for this omner block
  - Ethereum provides a minimum amount of reward to the omner miner

Ethereum ommers

- An ommer is a block whose parent is equal to the current block's parent's parent
- Block times in Ethereum are around 15 sec
  - This is much lower than that in Bitcoin (10 min)
- This enables faster transaction

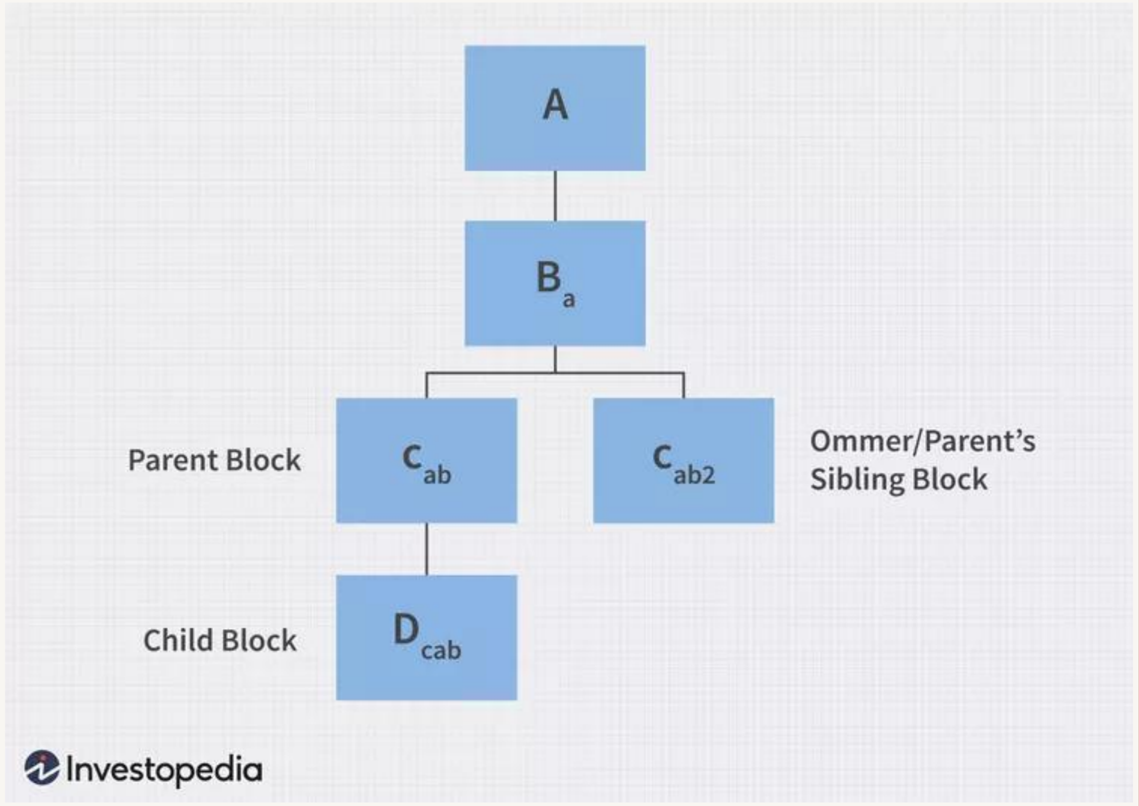
15s
ETH



https://www.investopedia.com/thmb/Bf315VzUb4nwViDSK36Piv55D1Q=/750x0/filters:no_upscale():max_bytes(150000):strip_icc():format(webp)/INV-uncle-block-cryptocurrency-fdc913eee3bb40aebba7e499bfceeada.jpg

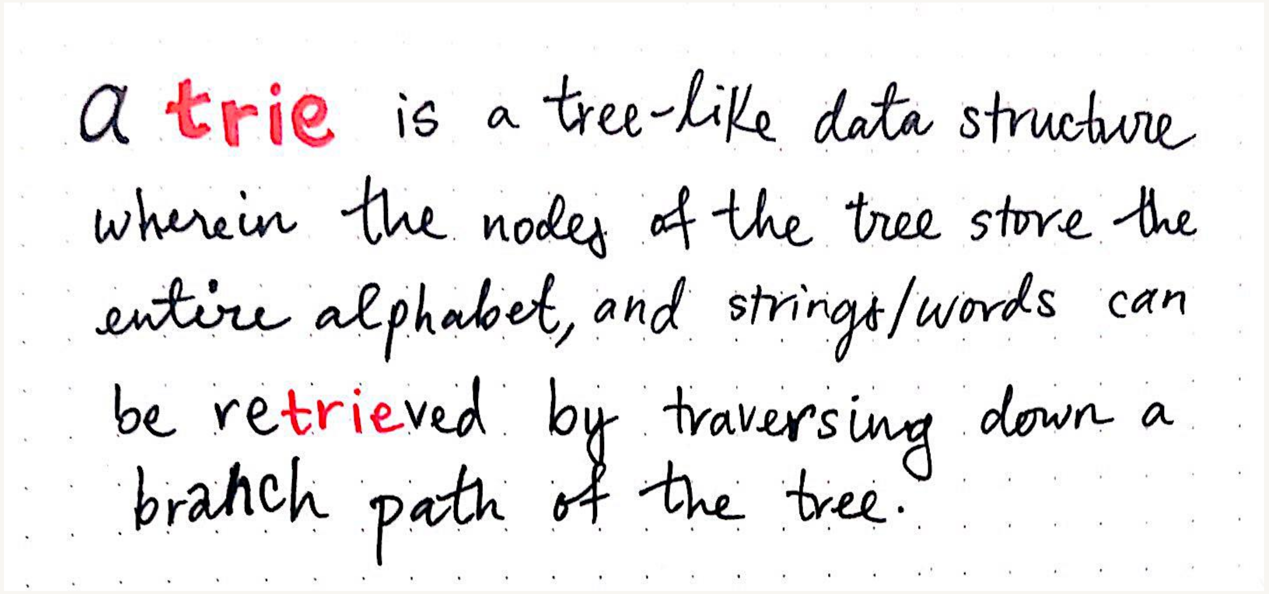
Ethereum ommers

- But there are more competing blocks, hence a higher number of orphaned blocks
- The purpose of ommers is to help reward miners for including these orphaned blocks
  - Compensating the miners for their computation



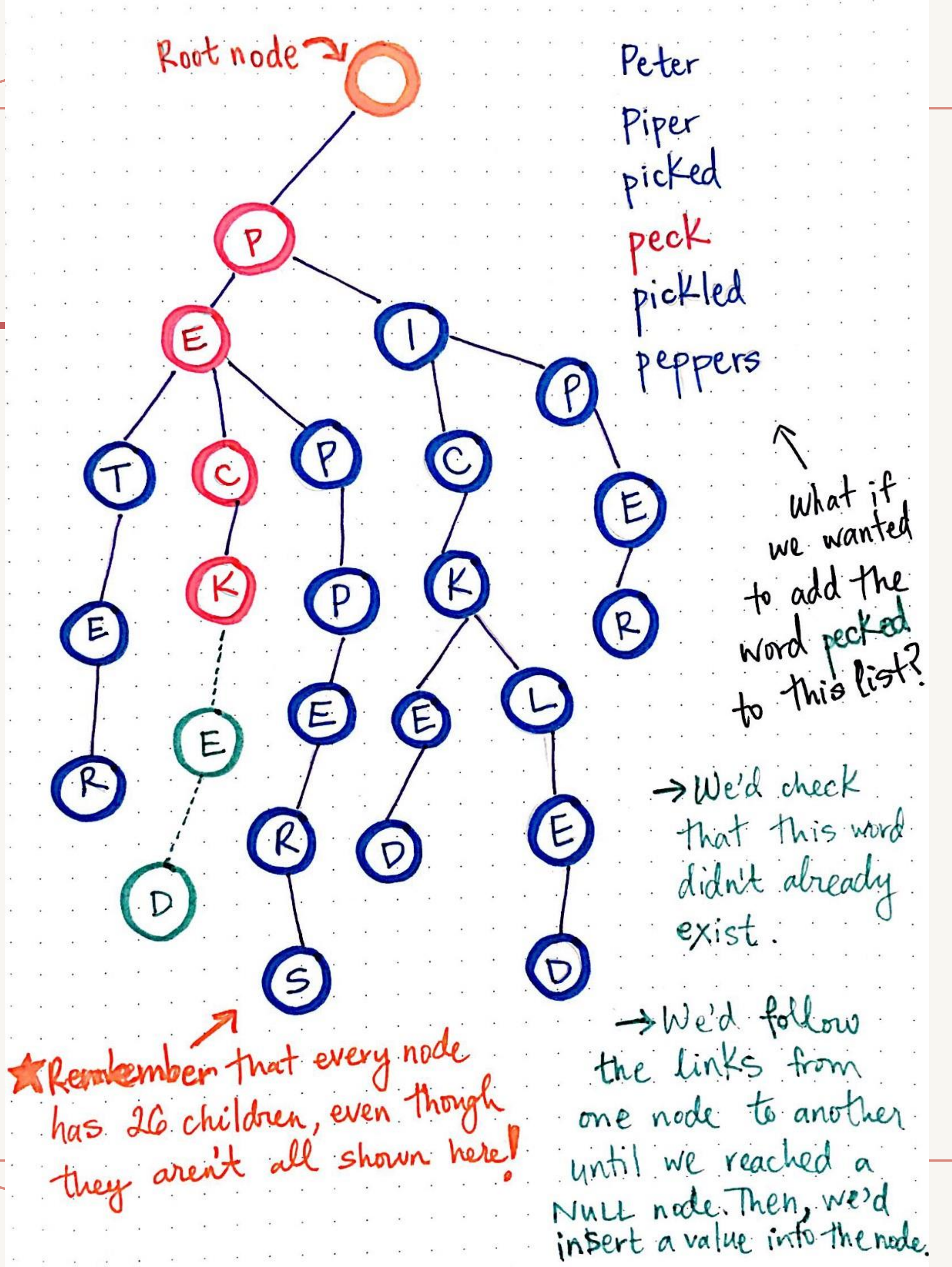
https://www.investopedia.com/thmb/Bf315VzUb4nwViDSK36Piv55D1Q=/750x0/filters:no_upscale():max_bytes(150000):strip_icc():format(webp)/INV-uncle-block-cryptocurrency-fdc913eee3bb40aebba7e499bfceeada.jpg

Trie



a trie is a tree-like data structure wherein the nodes of the tree store the entire alphabet, and strings/words can be retrieved by traversing down a branch path of the tree.

https://cdn-images-1.medium.com/max/1600/1*rkanFIU4G_tmuC939_txhA.jpeg

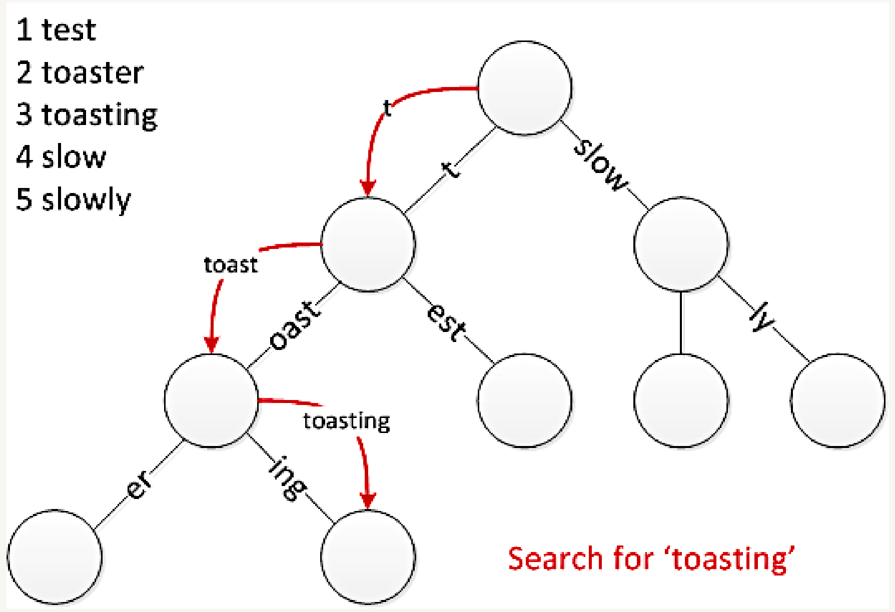


Root node
P
E
I
P
T
C
P
C
E
K
P
K
R
E
R
E
E
E
L
R
D
R
D
E
D
S
D
Peter
Piper
picked
peck
pickled
peppers
what if we wanted to add the word pecked to this list?
→We'd check that this word didn't already exist.
→We'd follow the links from one node to another until we reached a NULL node. Then, we'd insert a value into the node.
★Remember that every node has 26 children, even though they aren't all shown here!

https://cdn-images-1.medium.com/max/1200/1*sZOrNXzlQICVv5ePpav1-g.jpeg

Merkle Patricia trie

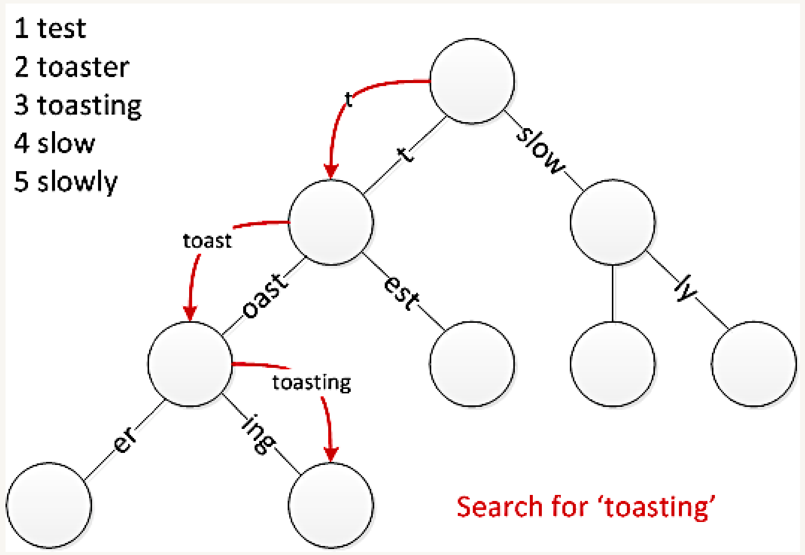
- A Patricia (Practical Algorithm To Retrieve Information Coded In Alphanumeric) trie is a binary radix trie
    - binary choice at each node when traversing the trie
- Modified in Ethereum with the concept of Merkle Patricia trie
    - the root node becomes a cryptographic fingerprint of the entire data structure, just like a Merkle tree



https://i.stack.imgur.com/d2w07.png

PT

- PT is a data structure which uses a key as a path so the nodes that share the same prefix can also share the same path
- This structure is fastest at finding common prefixes, simple to implement, and requires small memory
- It is commonly used for implementing routing tables, systems that are used in low specification machines like the router



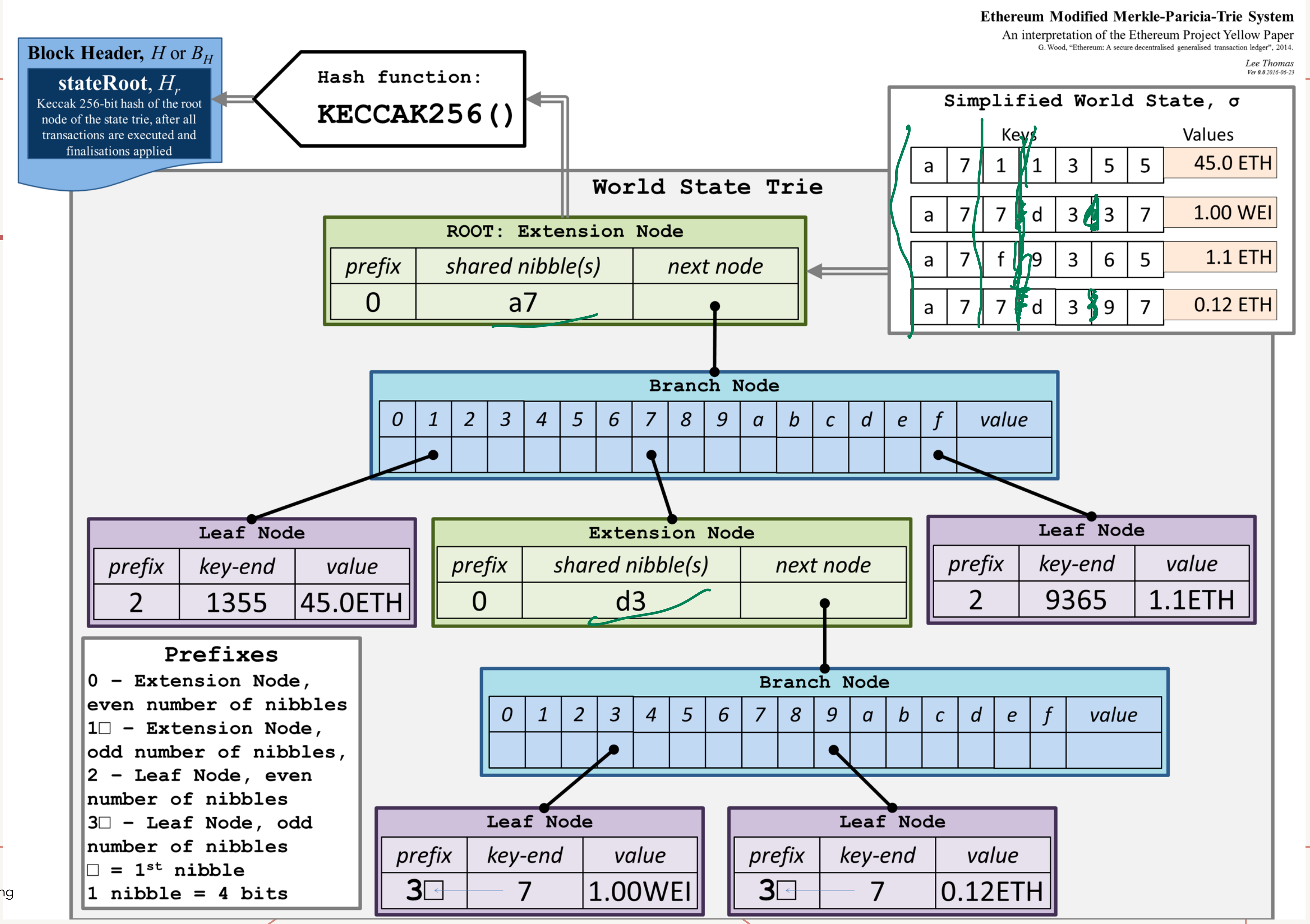
Merkle Patricia Trie (MPT)

- An MPT is a data structure for storing key value pairs in a cryptographically authenticated manner
- In the MPT every node has a hash value
- This hash is also used as the key that refers to the node
- The content of the node is stored in the Ethereum blockchain
- A node that does not have a child node is called a leaf node

MPT

- Nodes in MPT can have 16 child nodes
  - Plus it has its value, totalling 17 fields
- In Ethereum, hexadecimal is used – a 16 characters "alphabet"
- Note a hex character is referred to as a "nibble"
- Three different node types: extension, branch and leaf

MPT

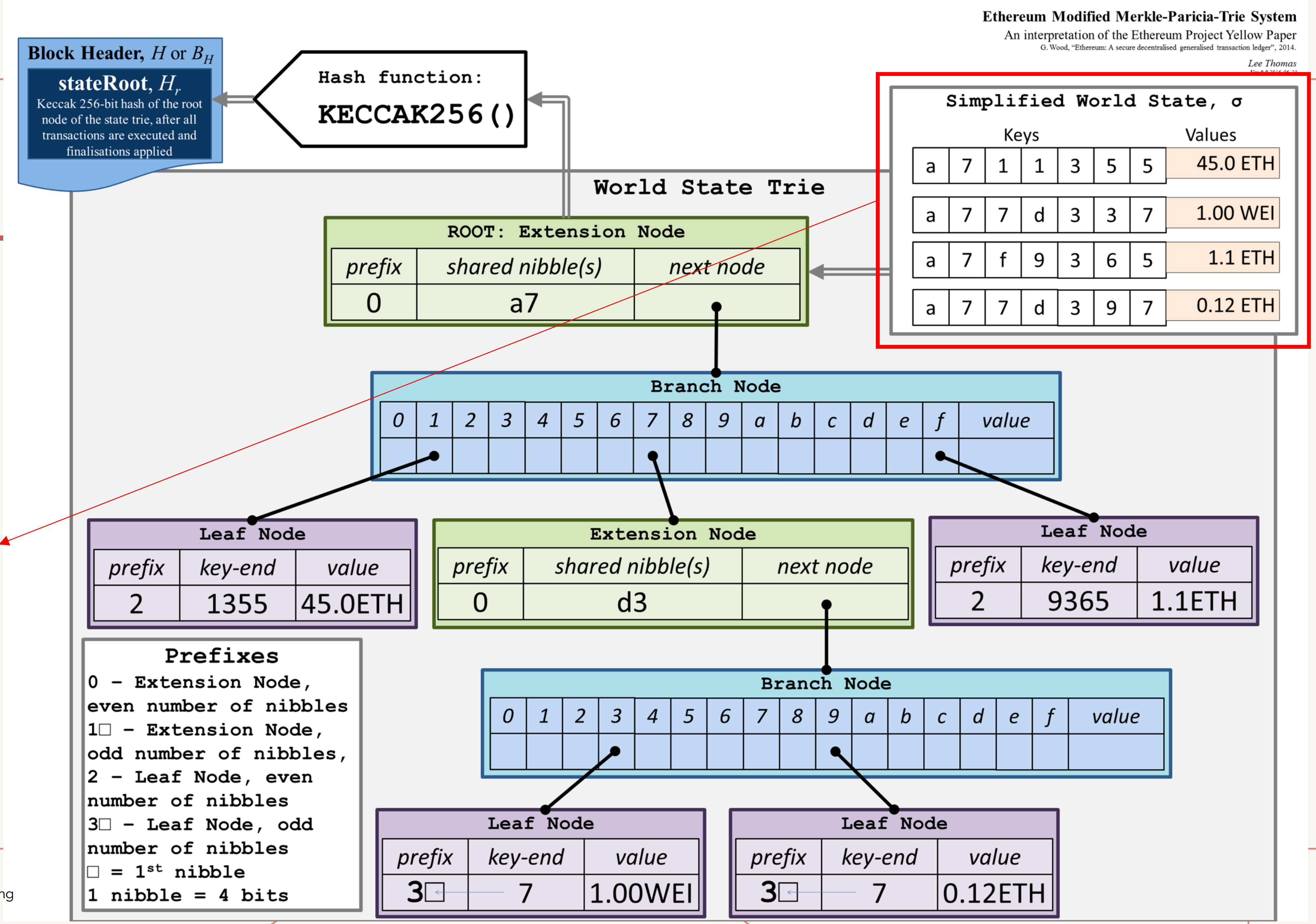


Ethereum Modified Merkle-Paricia-Trie System
An interpretation of the Ethereum Project Yellow Paper
G. Wood, "Ethereum: A secure decentralised generalised transaction ledger", 2014.
Lee Thomas
Ver 0.0 2016-06-23
Block Header, H or B_H
stateRoot, H_r
Keccak 256-bit hash of the root node of the state trie, after all transactions are executed and finalisations applied
Hash function:
KECCAK256()
Simplified World State, σ
Keys
Values
a 7 1 1 3 5 5 — 45.0 ETH
a 7 7 d 3 3 7 — 1.00 WEI
a 7 f 9 3 6 5 — 1.1 ETH
a 7 7 d 3 9 7 — 0.12 ETH
World State Trie
ROOT: Extension Node
prefix | shared nibble(s) | next node
0 | a7 |
Branch Node
0 1 2 3 4 5 6 7 8 9 a b c d e f value
Leaf Node
prefix | key-end | value
2 | 1355 | 45.0ETH
Extension Node
prefix | shared nibble(s) | next node
0 | d3 |
Leaf Node
prefix | key-end | value
2 | 9365 | 1.1ETH
Branch Node
0 1 2 3 4 5 6 7 8 9 a b c d e f value
Leaf Node
prefix | key-end | value
3□ | 7 | 1.00WEI
Leaf Node
prefix | key-end | value
3□ | 7 | 0.12ETH
Prefixes
0 – Extension Node, even number of nibbles
1□ – Extension Node, odd number of nibbles,
2 – Leaf Node, even number of nibbles
3□ – Leaf Node, odd number of nibbles
□ = 1st nibble
1 nibble = 4 bits

https://i.stack.imgur.com/YZGxe.png

MPT

- This represents a simplified world state of accounts
- Instead of storing each account in the blockchain, MPT is used

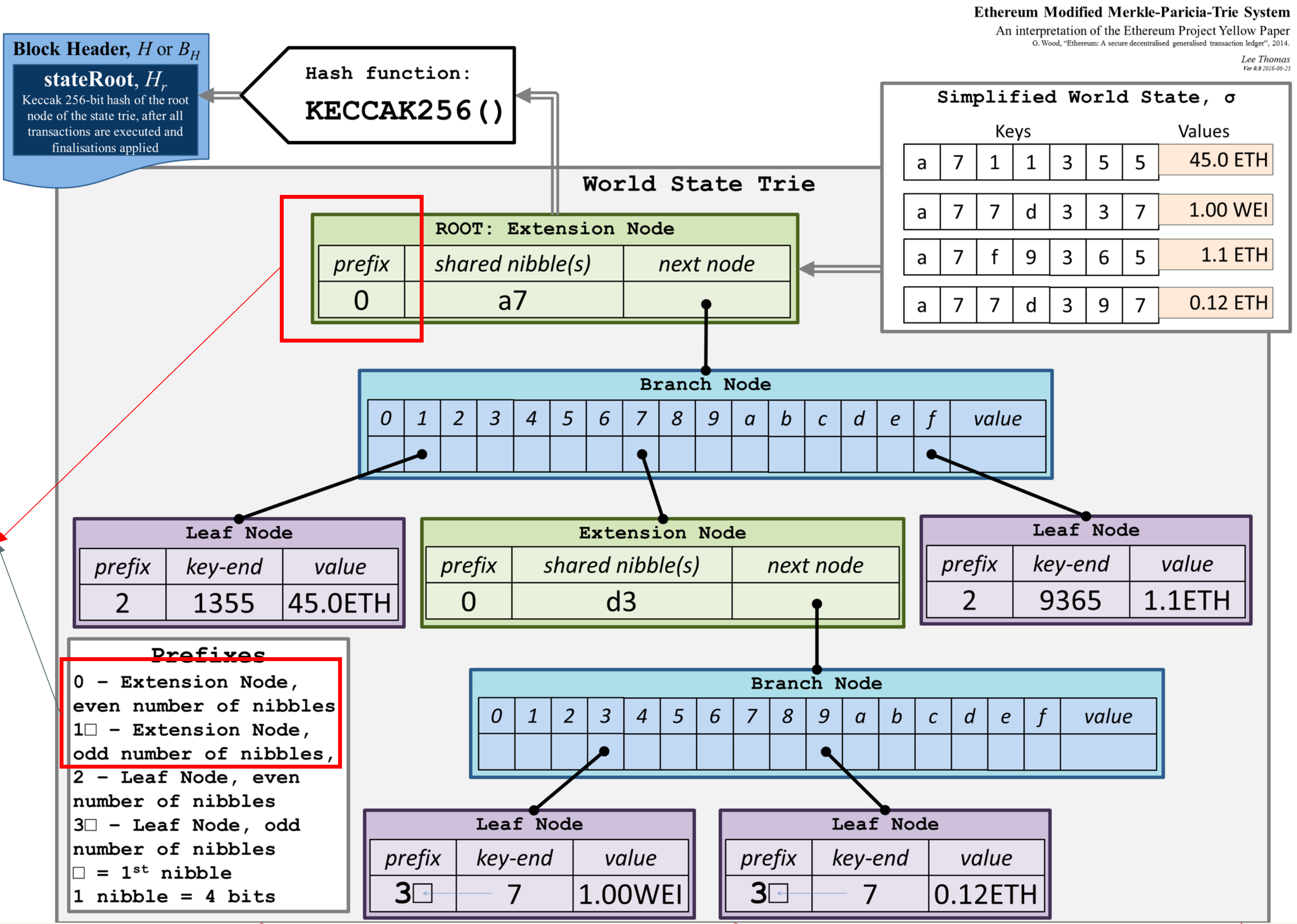


Ethereum Modified Merkle-Paricia-Trie System
An interpretation of the Ethereum Project Yellow Paper
G. Wood, "Ethereum: A secure decentralised generalised transaction ledger", 2014.
Lee Thomas
Block Header, H or B_H
stateRoot, H_r
Keccak 256-bit hash of the root node of the state trie, after all transactions are executed and finalisations applied
Hash function:
KECCAK256()
World State Trie
ROOT: Extension Node
prefix | shared nibble(s) | next node
0 | a7
Branch Node
0 1 2 3 4 5 6 7 8 9 a b c d e f value
Leaf Node
prefix | key-end | value
2 | 1355 | 45.0ETH
Extension Node
prefix | shared nibble(s) | next node
0 | d3
Leaf Node
prefix | key-end | value
2 | 9365 | 1.1ETH
Branch Node
0 1 2 3 4 5 6 7 8 9 a b c d e f value
Leaf Node
prefix | key-end | value
3□ | 7 | 1.00WEI
Leaf Node
prefix | key-end | value
3□ | 7 | 0.12ETH
Simplified World State, σ
Keys
Values
a 7 1 1 3 5 5 | 45.0 ETH
a 7 7 d 3 3 7 | 1.00 WEI
a 7 f 9 3 6 5 | 1.1 ETH
a 7 7 d 3 9 7 | 0.12 ETH
Prefixes
0 – Extension Node, even number of nibbles
1□ – Extension Node, odd number of nibbles,
2 – Leaf Node, even number of nibbles
3□ – Leaf Node, odd number of nibbles
□ = 1st nibble
1 nibble = 4 bits

https://i.stack.imgur.com/YZGxe.png

MPT

- Prefix determines the type of node
- 0/1 indicates an extension node
- If there are even number of nibbles then 0, otherwise 1

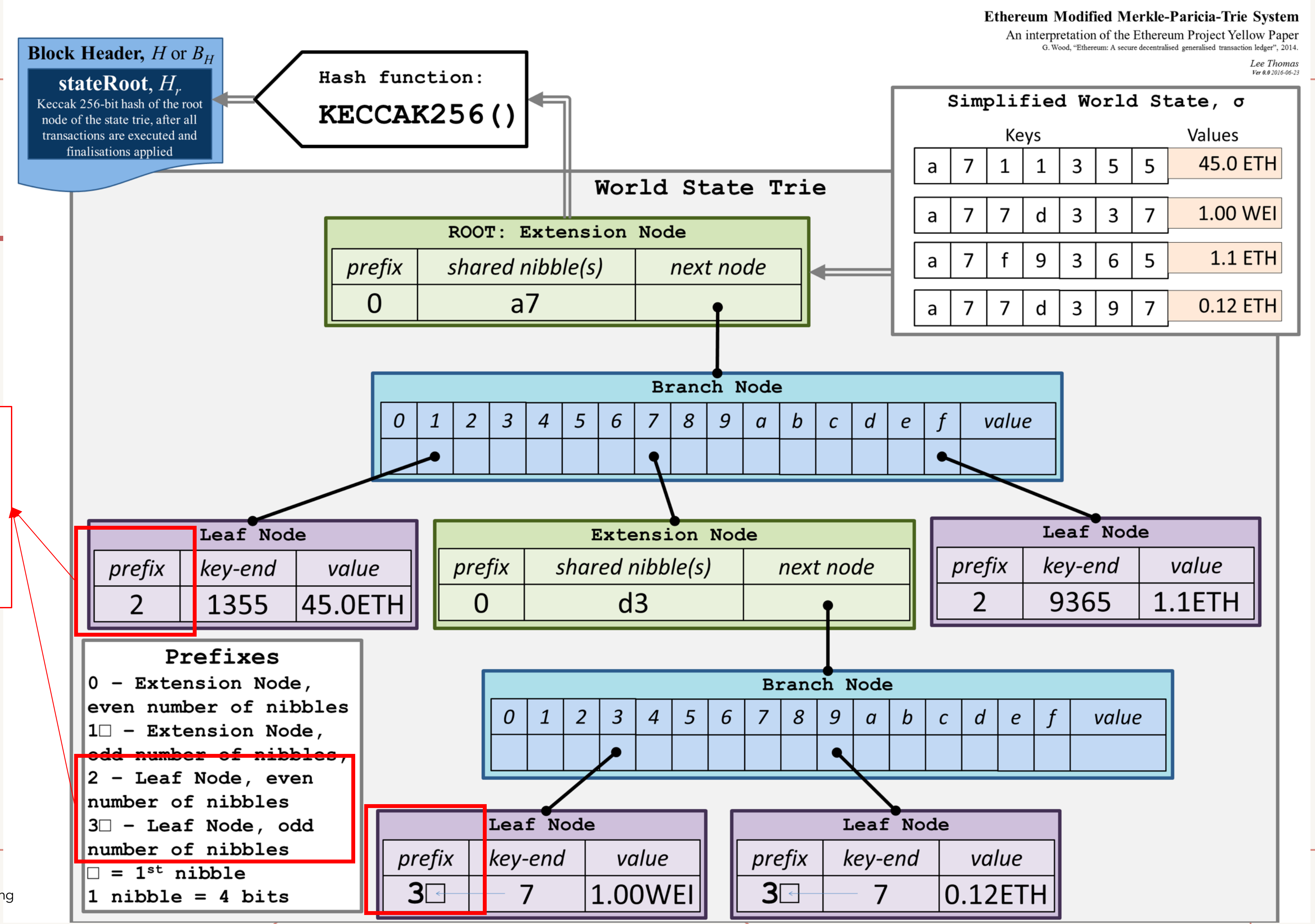


Ethereum Modified Merkle-Paricia-Trie System
An interpretation of the Ethereum Project Yellow Paper
G. Wood, "Ethereum: A secure decentralised generalised transaction ledger", 2014.
Lee Thomas
Block Header, H or B_H
stateRoot, H_r
Keccak 256-bit hash of the root node of the state trie, after all transactions are executed and finalisations applied
Hash function:
KECCAK256()
World State Trie
Simplified World State, σ
Keys
Values
a 7 1 1 3 5 5 — 45.0 ETH
a 7 7 d 3 3 7 — 1.00 WEI
a 7 f 9 3 6 5 — 1.1 ETH
a 7 7 d 3 9 7 — 0.12 ETH
ROOT: Extension Node
prefix 0 | shared nibble(s) a7 | next node
Branch Node
0 1 2 3 4 5 6 7 8 9 a b c d e f value
Leaf Node
prefix 2 | key-end 1355 | value 45.0ETH
Extension Node
prefix 0 | shared nibble(s) d3 | next node
Leaf Node
prefix 2 | key-end 9365 | value 1.1ETH
Prefixes
0 – Extension Node, even number of nibbles
1□ – Extension Node, odd number of nibbles,
2 – Leaf Node, even number of nibbles
3□ – Leaf Node, odd number of nibbles
□ = 1st nibble
1 nibble = 4 bits
Branch Node
0 1 2 3 4 5 6 7 8 9 a b c d e f value
Leaf Node
prefix 3□ | key-end 7 | value 1.00WEI
Leaf Node
prefix 3□ | key-end 7 | value 0.12ETH

https://i.stack.imgur.com/YZGxe.png

MPT

- 2/3 indicates a leaf node
- If there are even number of nibbles then 2, otherwise 3

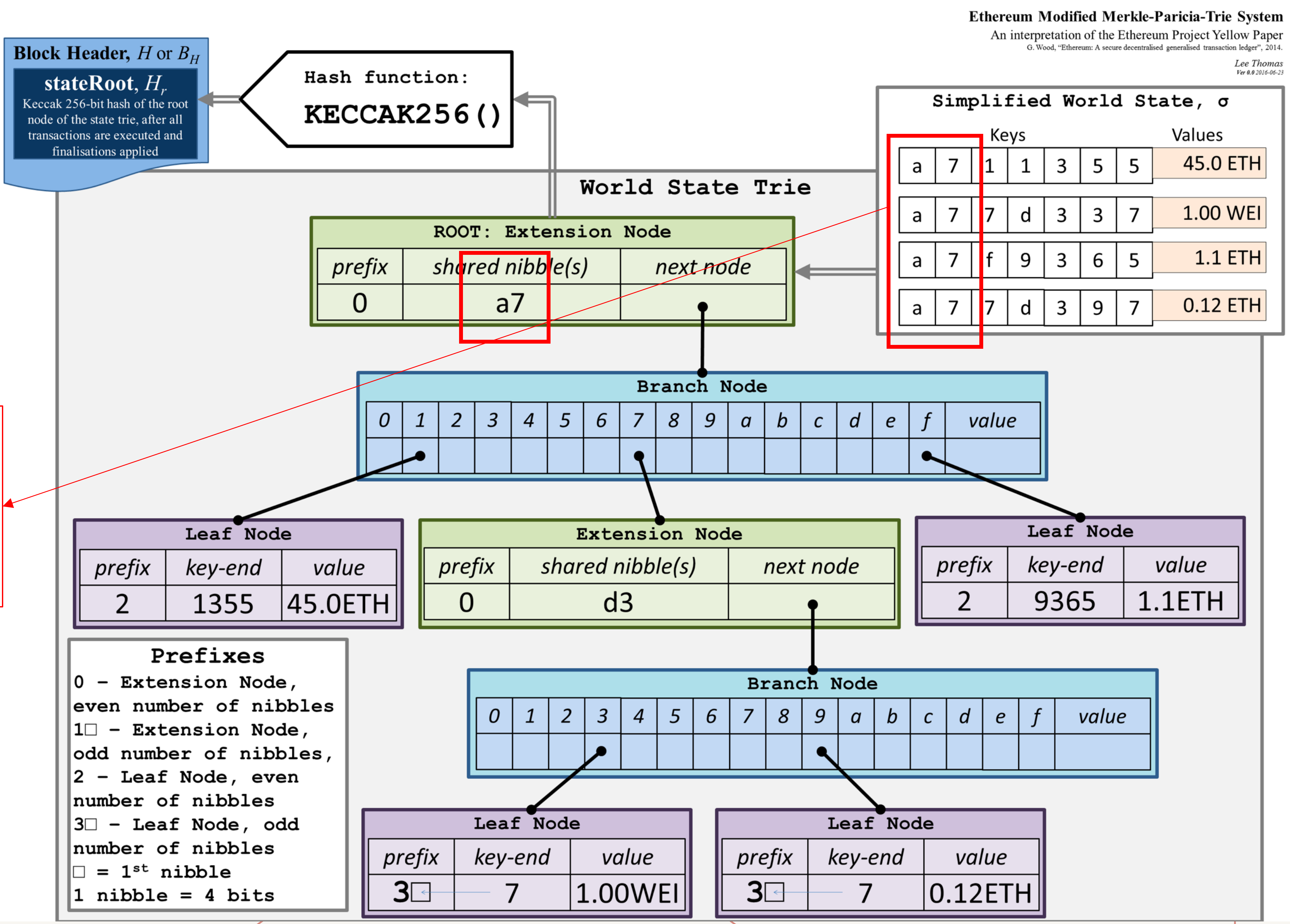


Ethereum Modified Merkle-Paricia-Trie System
An interpretation of the Ethereum Project Yellow Paper
G. Wood, "Ethereum: A secure decentralised generalised transaction ledger", 2014.
Lee Thomas
Ver 0.0 2016-06-23
Block Header, H or B_H
stateRoot, H_r
Keccak 256-bit hash of the root node of the state trie, after all transactions are executed and finalisations applied
Hash function:
KECCAK256()
Simplified World State, σ
Keys
Values
a 7 1 1 3 5 5 — 45.0 ETH
a 7 7 d 3 3 7 — 1.00 WEI
a 7 f 9 3 6 5 — 1.1 ETH
a 7 7 d 3 9 7 — 0.12 ETH
World State Trie
ROOT: Extension Node
prefix | shared nibble(s) | next node
0 | a7
Branch Node
0 1 2 3 4 5 6 7 8 9 a b c d e f value
Leaf Node
prefix | key-end | value
2 | 1355 | 45.0ETH
Extension Node
prefix | shared nibble(s) | next node
0 | d3
Leaf Node
prefix | key-end | value
2 | 9365 | 1.1ETH
Branch Node
0 1 2 3 4 5 6 7 8 9 a b c d e f value
Leaf Node
prefix | key-end | value
3□ | 7 | 1.00WEI
Leaf Node
prefix | key-end | value
3□ | 7 | 0.12ETH
Prefixes
0 – Extension Node, even number of nibbles
1□ – Extension Node, odd number of nibbles
2 – Leaf Node, even number of nibbles
3□ – Leaf Node, odd number of nibbles
□ = 1st nibble
1 nibble = 4 bits

https://i.stack.imgur.com/YZGxe.png

MPT

- All these accounts share a common prefix: a7
- That is why a7 remains in the root

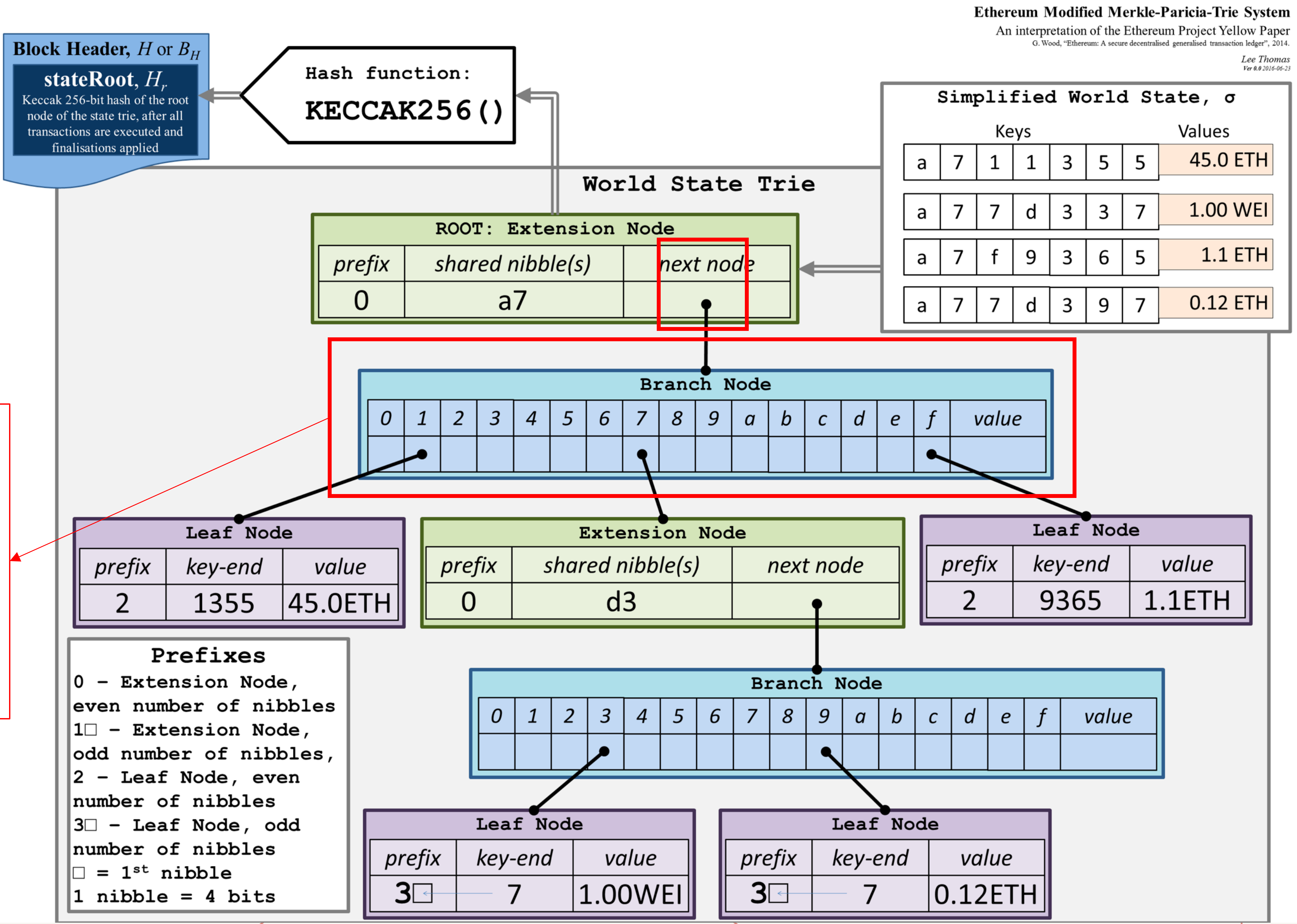


Ethereum Modified Merkle-Paricia-Trie System
An interpretation of the Ethereum Project Yellow Paper
G. Wood, "Ethereum: A secure decentralised generalised transaction ledger", 2014.
Lee Thomas
Ver 0.0 2016-06-23
Block Header, H or B_H
stateRoot, H_r
Keccak 256-bit hash of the root node of the state trie, after all transactions are executed and finalisations applied
Hash function:
KECCAK256()
Simplified World State, σ
Keys
Values
a 7 1 1 3 5 5 — 45.0 ETH
a 7 7 d 3 3 7 — 1.00 WEI
a 7 f 9 3 6 5 — 1.1 ETH
a 7 7 d 3 9 7 — 0.12 ETH
World State Trie
ROOT: Extension Node
prefix | shared nibble(s) | next node
0 | a7
Branch Node
0 1 2 3 4 5 6 7 8 9 a b c d e f value
Leaf Node
prefix | key-end | value
2 | 1355 | 45.0ETH
Extension Node
prefix | shared nibble(s) | next node
0 | d3
Leaf Node
prefix | key-end | value
2 | 9365 | 1.1ETH
Branch Node
0 1 2 3 4 5 6 7 8 9 a b c d e f value
Leaf Node
prefix | key-end | value
3□ | 7 | 1.00WEI
Leaf Node
prefix | key-end | value
3□ | 7 | 0.12ETH
Prefixes
0 – Extension Node, even number of nibbles
1□ – Extension Node, odd number of nibbles,
2 – Leaf Node, even number of nibbles
3□ – Leaf Node, odd number of nibbles
□ = 1st nibble
1 nibble = 4 bits

https://i.stack.imgur.com/YZGxe.png

MPT

- Next node field in the extension node points to a branch node
- A branch node has 16 hexadecimal characters and a value field

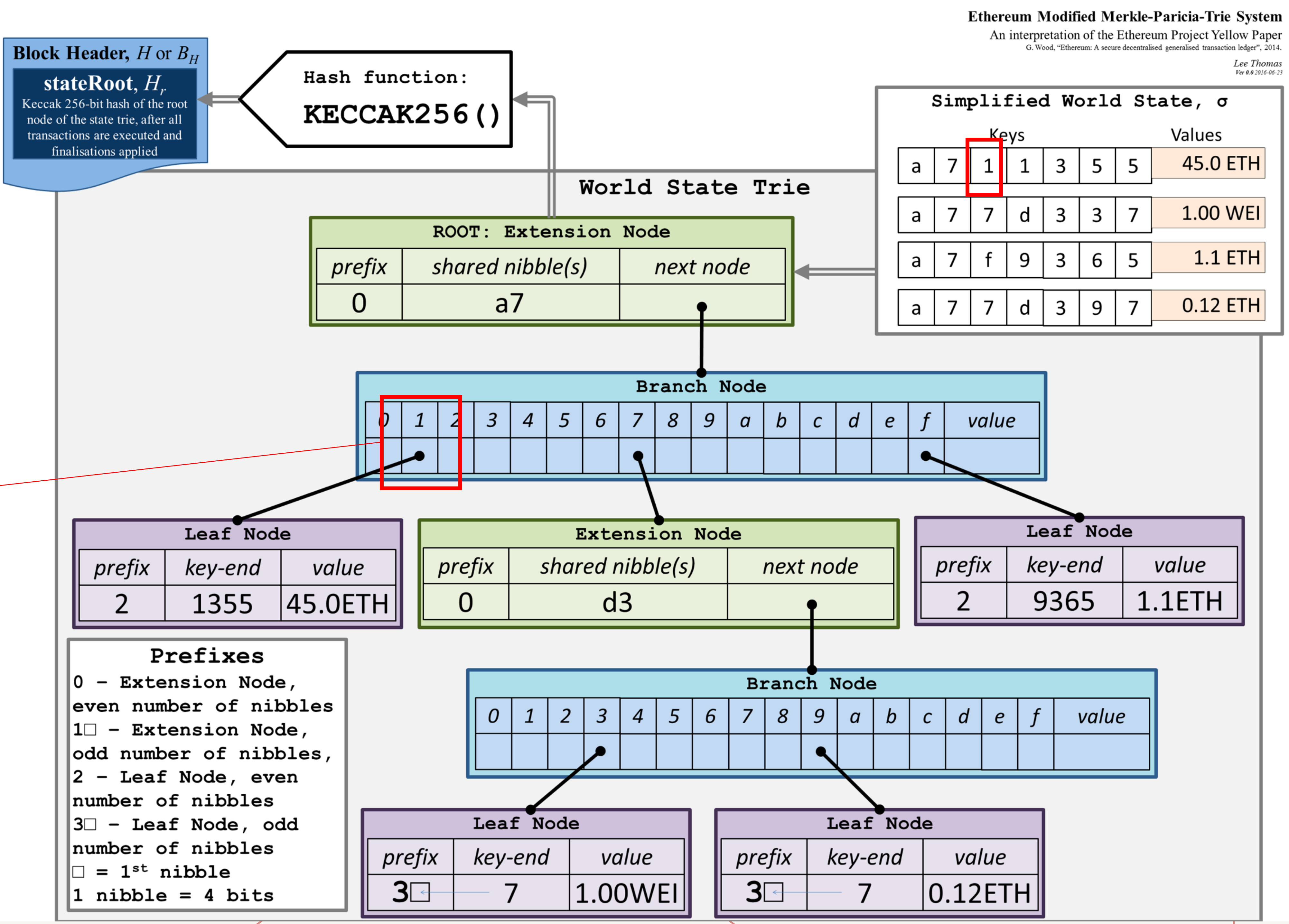


Ethereum Modified Merkle-Paricia-Trie System
An interpretation of the Ethereum Project Yellow Paper
G. Wood, "Ethereum: A secure decentralised generalised transaction ledger", 2014.
Lee Thomas
Ver 0.0 2016-06-23
Block Header, H or B_H
stateRoot, H_r
Keccak 256-bit hash of the root node of the state trie, after all transactions are executed and finalisations applied
Hash function:
KECCAK256()
Simplified World State, σ
Keys
Values
a 7 1 1 3 5 5 — 45.0 ETH
a 7 7 d 3 3 7 — 1.00 WEI
a 7 f 9 3 6 5 — 1.1 ETH
a 7 7 d 3 9 7 — 0.12 ETH
World State Trie
ROOT: Extension Node
prefix | shared nibble(s) | next node
0 | a7
Branch Node
0 1 2 3 4 5 6 7 8 9 a b c d e f value
Leaf Node
prefix | key-end | value
2 | 1355 | 45.0ETH
Extension Node
prefix | shared nibble(s) | next node
0 | d3
Leaf Node
prefix | key-end | value
2 | 9365 | 1.1ETH
Branch Node
0 1 2 3 4 5 6 7 8 9 a b c d e f value
Leaf Node
prefix | key-end | value
3□ | 7 | 1.00WEI
Leaf Node
prefix | key-end | value
3□ | 7 | 0.12ETH
Prefixes
0 – Extension Node, even number of nibbles
1□ – Extension Node, odd number of nibbles,
2 – Leaf Node, even number of nibbles
3□ – Leaf Node, odd number of nibbles
□ = 1st nibble
1 nibble = 4 bits

https://i.stack.imgur.com/YZGxe.png

MPT

- Each field except the value field represents a key character

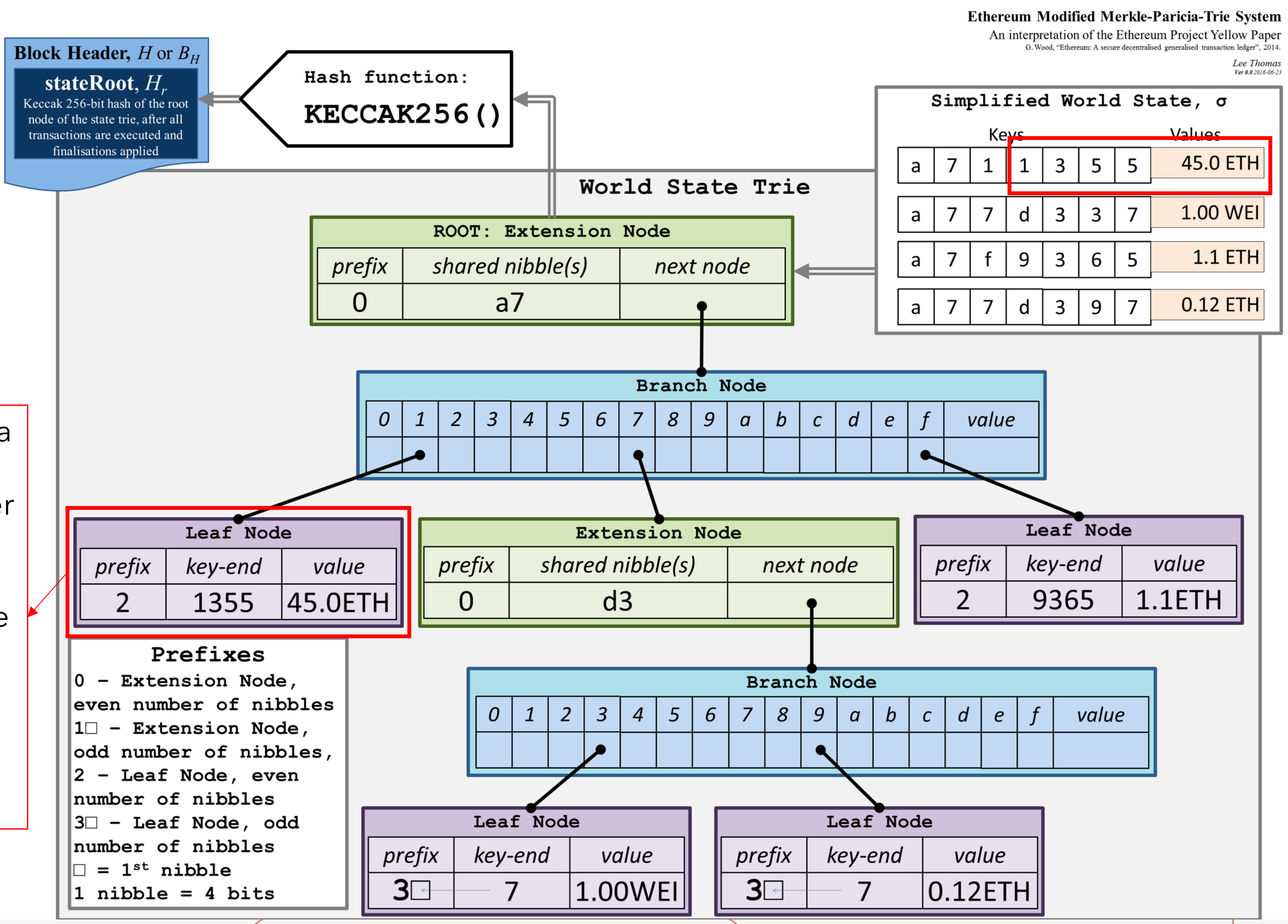


Ethereum Modified Merkle-Paricia-Trie System
An interpretation of the Ethereum Project Yellow Paper
G. Wood, "Ethereum: A secure decentralised generalised transaction ledger", 2014.
Lee Thomas
Ver 0.0 2016-06-23
Block Header, H or B_H
stateRoot, H_r
Keccak 256-bit hash of the root node of the state trie, after all transactions are executed and finalisations applied
Hash function:
KECCAK256()
Simplified World State, σ
Keys
Values
a 7 1 1 3 5 5 — 45.0 ETH
a 7 7 d 3 3 7 — 1.00 WEI
a 7 f 9 3 6 5 — 1.1 ETH
a 7 7 d 3 9 7 — 0.12 ETH
World State Trie
ROOT: Extension Node
prefix | shared nibble(s) | next node
0 | a7
Branch Node
0 1 2 3 4 5 6 7 8 9 a b c d e f value
Leaf Node
prefix | key-end | value
2 | 1355 | 45.0ETH
Extension Node
prefix | shared nibble(s) | next node
0 | d3
Leaf Node
prefix | key-end | value
2 | 9365 | 1.1ETH
Branch Node
0 1 2 3 4 5 6 7 8 9 a b c d e f value
Leaf Node
prefix | key-end | value
3□ | 7 | 1.00WEI
Leaf Node
prefix | key-end | value
3□ | 7 | 0.12ETH
Prefixes
0 – Extension Node, even number of nibbles
1□ – Extension Node, odd number of nibbles,
2 – Leaf Node, even number of nibbles
3□ – Leaf Node, odd number of nibbles
□ = 1st nibble
1 nibble = 4 bits

https://i.stack.imgur.com/YZGxe.png

MPT

- Each leaf node has a prefix indicating its even or odd number of nibbles
- A key-end to store the last values of the key
- Finally the corresponding balance for the account



Ethereum Modified Merkle-Paricia-Trie System
An interpretation of the Ethereum Project Yellow Paper
G. Wood, "Ethereum: A secure decentralised generalised transaction ledger", 2014.
Lee Thomas
Ver 0.0 2016-06-23

Block Header, H or B_H

stateRoot, H_r
Keccak 256-bit hash of the root node of the state trie, after all transactions are executed and finalisations applied

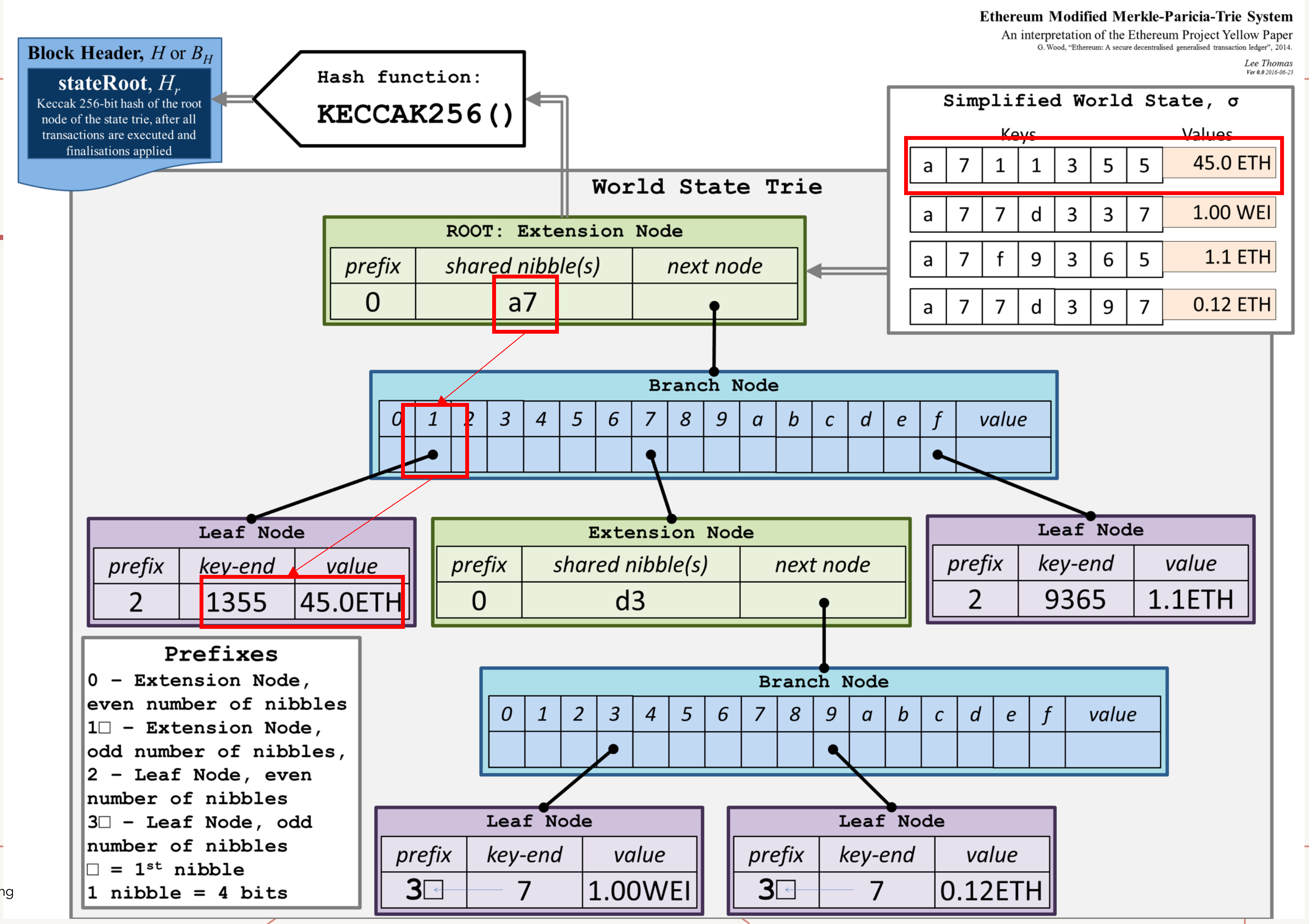
Simplified World State, σ

Keys							Values
a	7	1	1	3	5	5	45.0 ETH
a	7	7	d	3	3	7	1.00 WEI
a	7	f	9	3	6	5	1.1 ETH
a	7	7	d	3	9	7	0.12 ETH

Prefixes
0 – Extension Node, even number of nibbles
1□ – Extension Node, odd number of nibbles,
2 – Leaf Node, even number of nibbles
3□ – Leaf Node, odd number of nibbles
□ = 1st nibble
1 nibble = 4 bits

https://i.stack.imgur.com/YZGxe.png

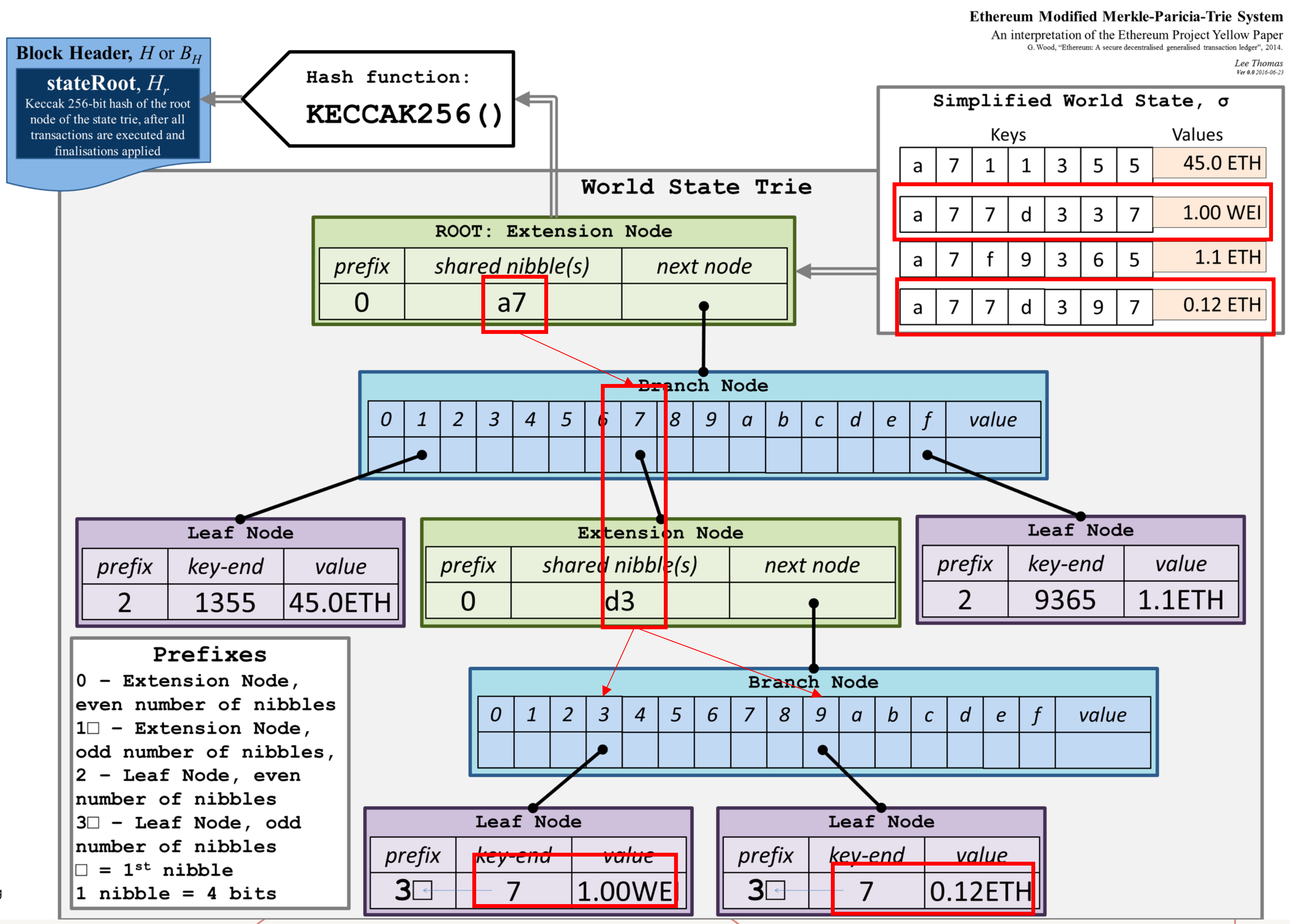
MPT



Ethereum Modified Merkle-Paricia-Trie System
An interpretation of the Ethereum Project Yellow Paper
G. Wood, "Ethereum: A secure decentralised generalised transaction ledger", 2014.
Lee Thomas
Ver 0.0 2016-06-23
Block Header, H or B_H
stateRoot, H_r
Keccak 256-bit hash of the root node of the state trie, after all transactions are executed and finalisations applied
Hash function:
KECCAK256()
Simplified World State, σ
Keys
Values
a 7 1 1 3 5 5 — 45.0 ETH
a 7 7 d 3 3 7 — 1.00 WEI
a 7 f 9 3 6 5 — 1.1 ETH
a 7 7 d 3 9 7 — 0.12 ETH
World State Trie
ROOT: Extension Node
prefix | shared nibble(s) | next node
0 | a7 |
Branch Node
0 1 2 3 4 5 6 7 8 9 a b c d e f value
Leaf Node
prefix | key-end | value
2 | 1355 | 45.0ETH
Extension Node
prefix | shared nibble(s) | next node
0 | d3 |
Leaf Node
prefix | key-end | value
2 | 9365 | 1.1ETH
Branch Node
0 1 2 3 4 5 6 7 8 9 a b c d e f value
Leaf Node
prefix | key-end | value
3□ | 7 | 1.00WEI
Leaf Node
prefix | key-end | value
3□ | 7 | 0.12ETH
Prefixes
0 – Extension Node, even number of nibbles
1□ – Extension Node, odd number of nibbles,
2 – Leaf Node, even number of nibbles
3□ – Leaf Node, odd number of nibbles
□ = 1st nibble
1 nibble = 4 bits

https://i.stack.imgur.com/YZGxe.png

MPT


Ethereum Modified Merkle-Paricia-Trie System
An interpretation of the Ethereum Project Yellow Paper
G. Wood, "Ethereum: A secure decentralised generalised transaction ledger", 2014.
Lee Thomas
Ver 0.0 2016-06-23
Block Header, H or B_H
stateRoot, H_r
Keccak 256-bit hash of the root node of the state trie, after all transactions are executed and finalisations applied
Hash function:
KECCAK256()
World State Trie
ROOT: Extension Node
prefix | shared nibble(s) | next node
0 | a7
Branch Node
0 1 2 3 4 5 6 7 8 9 a b c d e f value
Leaf Node
prefix | key-end | value
2 | 1355 | 45.0ETH
Extension Node
prefix | shared nibble(s) | next node
0 | d3
Leaf Node
prefix | key-end | value
2 | 9365 | 1.1ETH
Branch Node
0 1 2 3 4 5 6 7 8 9 a b c d e f value
Leaf Node
prefix | key-end | value
3□ | 7 | 1.00WEI
Leaf Node
prefix | key-end | value
3□ | 7 | 0.12ETH

Simplified World State, σ

Keys							Values
a	7	1	1	3	5	5	45.0 ETH
a	7	7	d	3	3	7	1.00 WEI
a	7	f	9	3	6	5	1.1 ETH
a	7	7	d	3	9	7	0.12 ETH

Prefixes

0 – Extension Node, even number of nibbles
1□ – Extension Node, odd number of nibbles,
2 – Leaf Node, even number of nibbles
3□ – Leaf Node, odd number of nibbles
□ = 1st nibble
1 nibble = 4 bits

https://i.stack.imgur.com/YZGxe.png

Ethereum tries

- There are four different tries used in Ethereum
  - State Trie
    - Contains an account information with respect to their address
  - Transaction Trie
    - Contains transaction information
  - Transaction Receipt Trie
    - Contains information regarding transaction receipt
  - Account storage Trie
    - Contains storage information with respect a smart contract

Question?