NATIONAL INSTITUTE OF TRANSPORT

Blockchain Technology Short Training

Introduction to Ethereum

Facilitator: Dr. Cleverence Kombe (PhD)

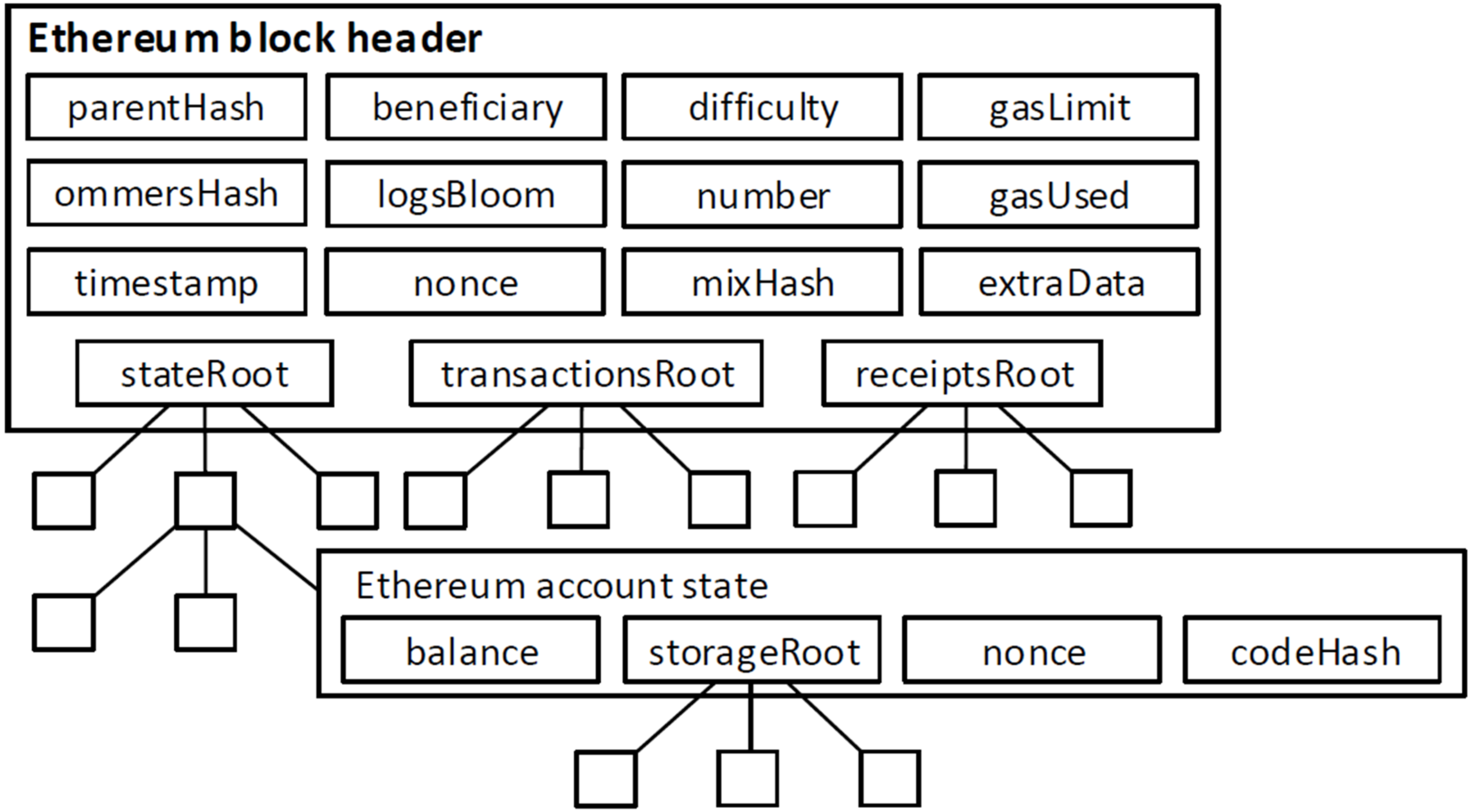
Introduction to Ethereum | Overview

- **Ethereum blocks and blockchain**

Blocks and blockchain | Block header

- **A block** is a container of information that contains a list of transactions, along with other metadata. In Ethereum, each block contains a **unique header**.
- Block headers are an essential component of Ethereum blocks, as they contain critical information that enables the blockchain to function efficiently and securely.
- The block header is the first part of the block, and it contains various elements that play a crucial role in the blockchain's operation.
- Understanding the various elements of an Ethereum block header is crucial for anyone interested in developing applications on the Ethereum network or participating in mining activities.

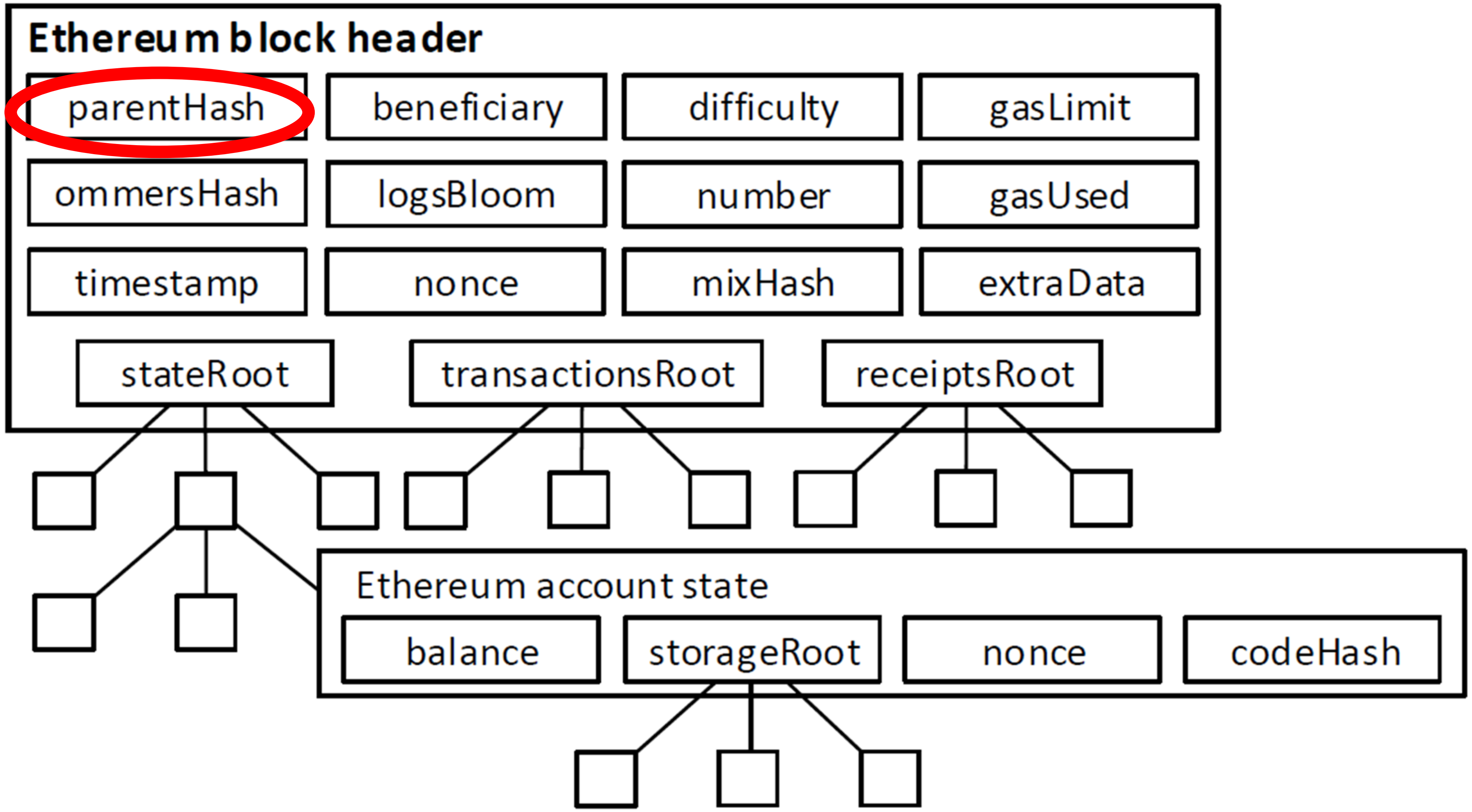
Blocks and blockchain | Block header



Blocks and blockchain | Block header

- **A block** is a container of information that contains a list of transactions, along with other metadata. In Ethereum, each block contains a **unique header**.
- Block headers are an essential component of Ethereum blocks, as they contain critical information that enables the blockchain to function efficiently and securely.
- The block header is the first part of the block, and it contains various elements that play a crucial role in the blockchain's operation.
- Understanding the various elements of an Ethereum block header is crucial for anyone interested in developing applications on the Ethereum network or participating in mining activities.

Blocks and blockchain | Block header : parentHash



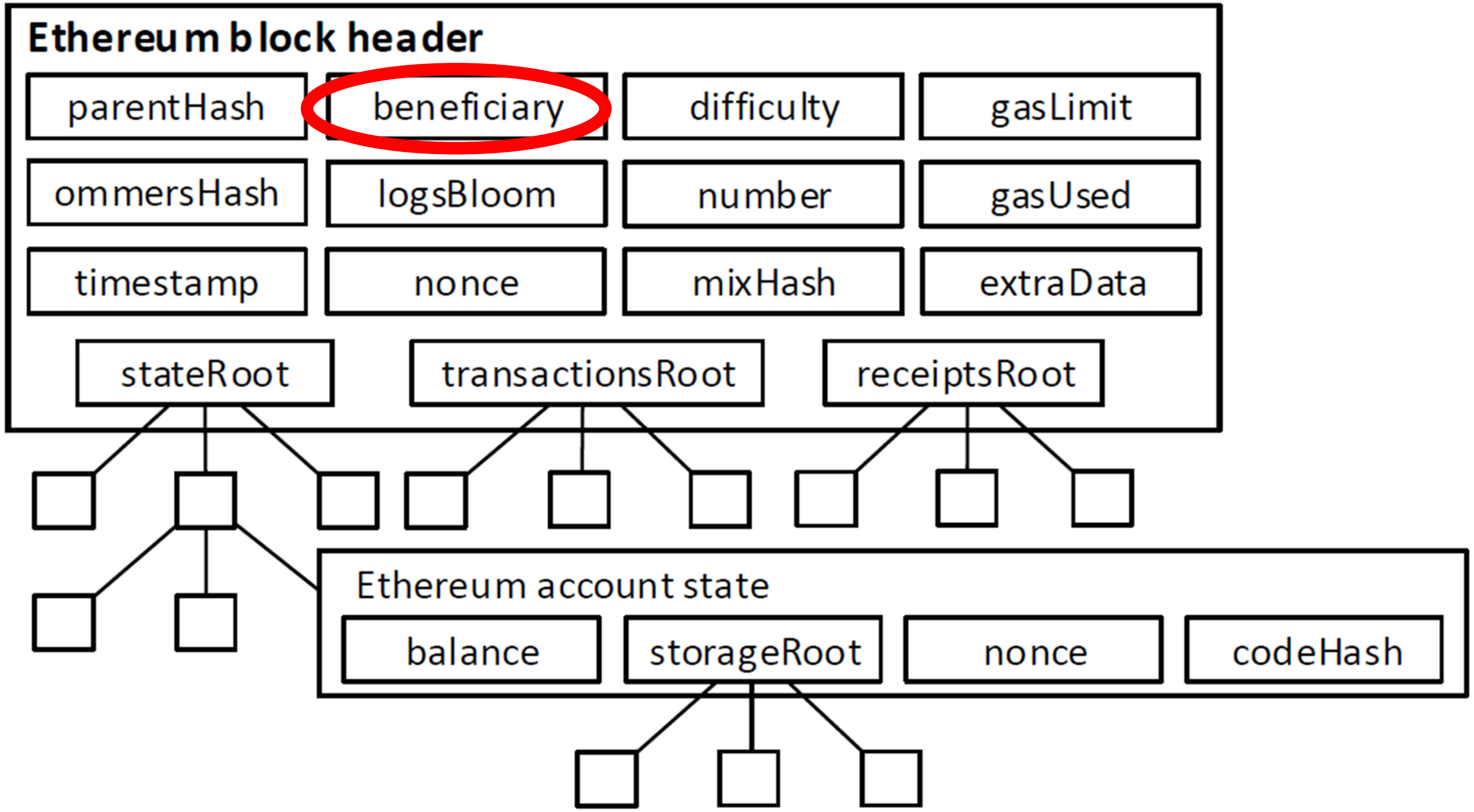
Blocks and blockchain | Block header : parentHash

- The parentHash field in the Ethereum block header is the hash of the block that precedes the current block.

- This field is used to maintain the integrity of the blockchain by ensuring that blocks are connected in a specific order.

- The parentHash field is also used to validate the current block, as it is included in the hash of the current block.

Blocks and blockchain | Block header : beneficiary



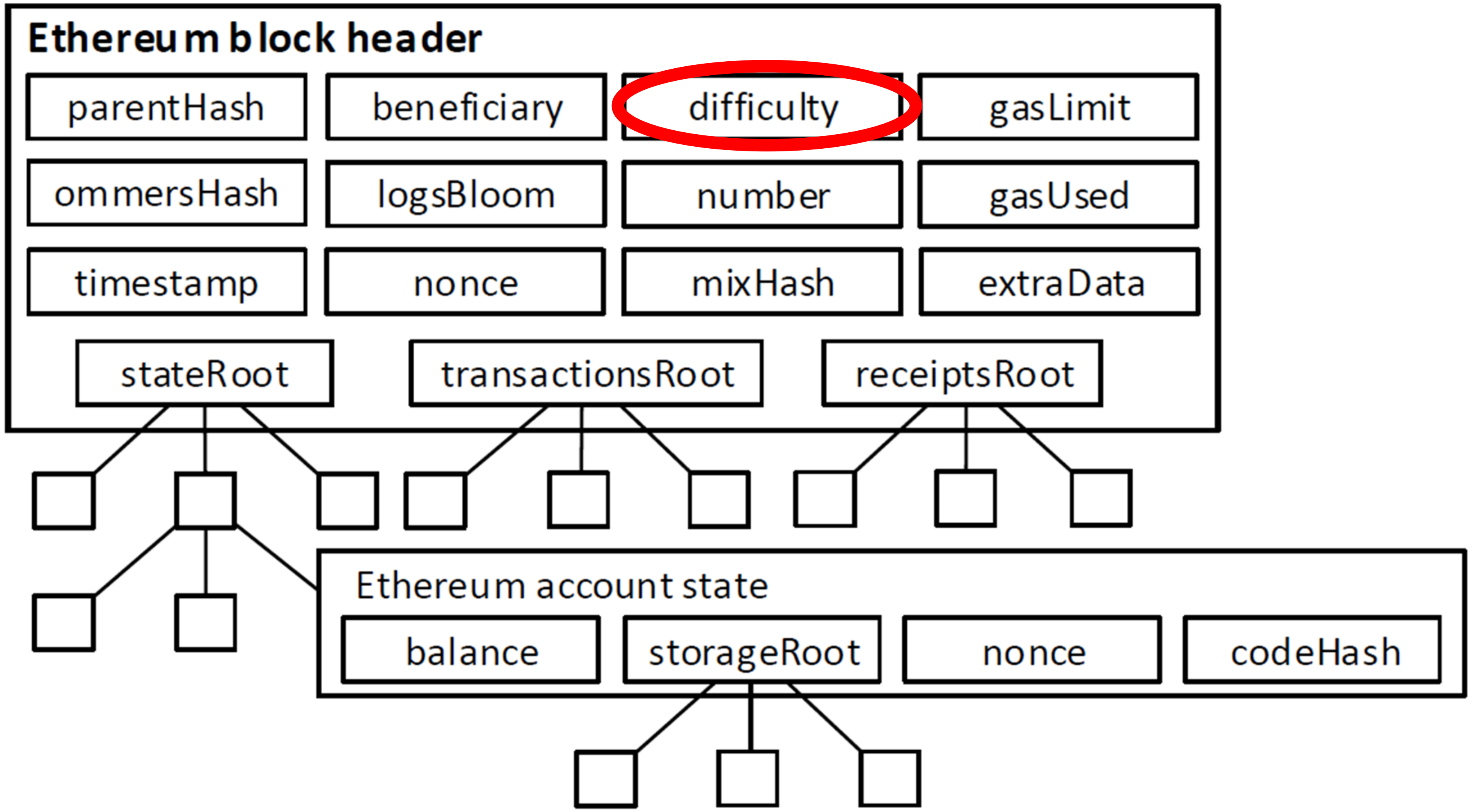
Blocks and blockchain | Block header : beneficiary

- The beneficiary field in the Ethereum block header is the address of the account that will receive the block reward.

- In Ethereum, mining a block results in the creation of new ether, which is given as a reward to the miner who solves the cryptographic puzzle.

- The beneficiary field specifies the account that will receive this reward.

Blocks and blockchain | Block header : difficulty



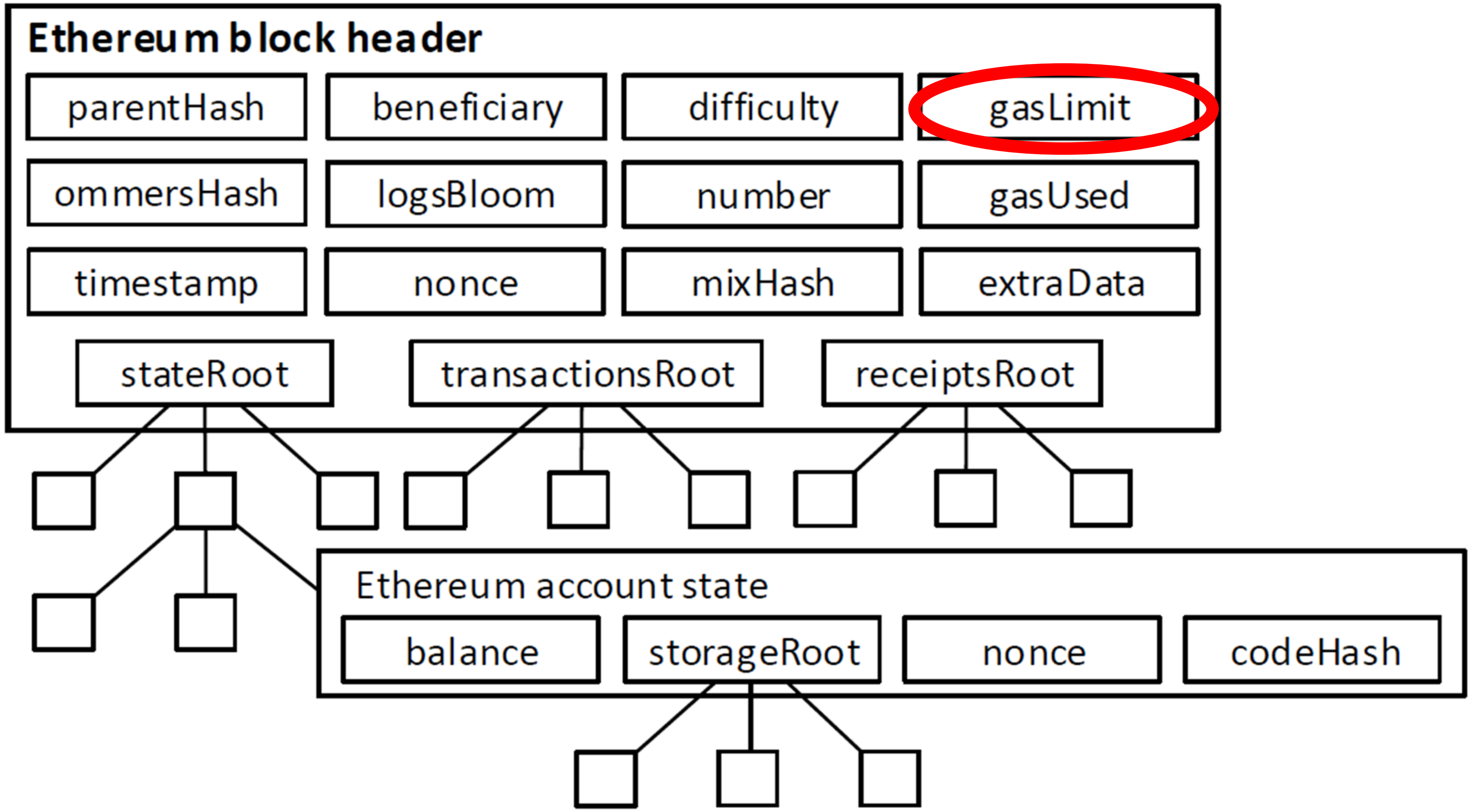
Blocks and blockchain | Block header : difficulty

- The difficulty field in the Ethereum block header represents the computational effort required to create a valid hash for the block.

- The difficulty is adjusted dynamically to ensure that blocks are added to the blockchain at a consistent rate.

- A higher difficulty level means that it is more challenging to create a valid hash for the block.

Blocks and blockchain | Block header : gasLimit



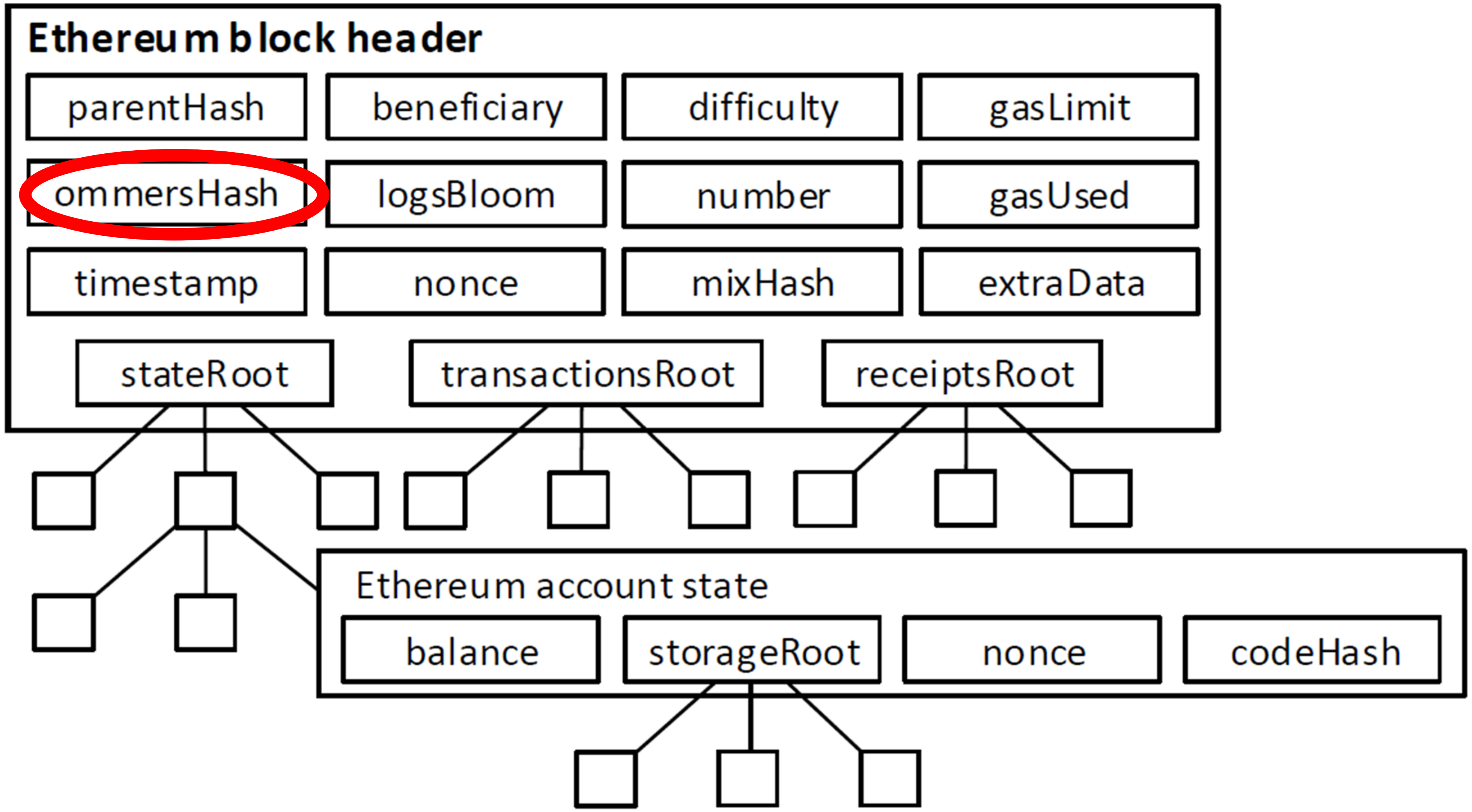
Blocks and blockchain | Block header : gasLimit

- The gasLimit field in the Ethereum block header is the maximum amount of gas that can be used by transactions included in the block.

- Gas is used to pay for the computational resources required to execute transactions on the Ethereum network.

- The gasLimit field is adjusted dynamically to ensure that the network remains efficient and secure.

Blocks and blockchain | Block header : ommersHash



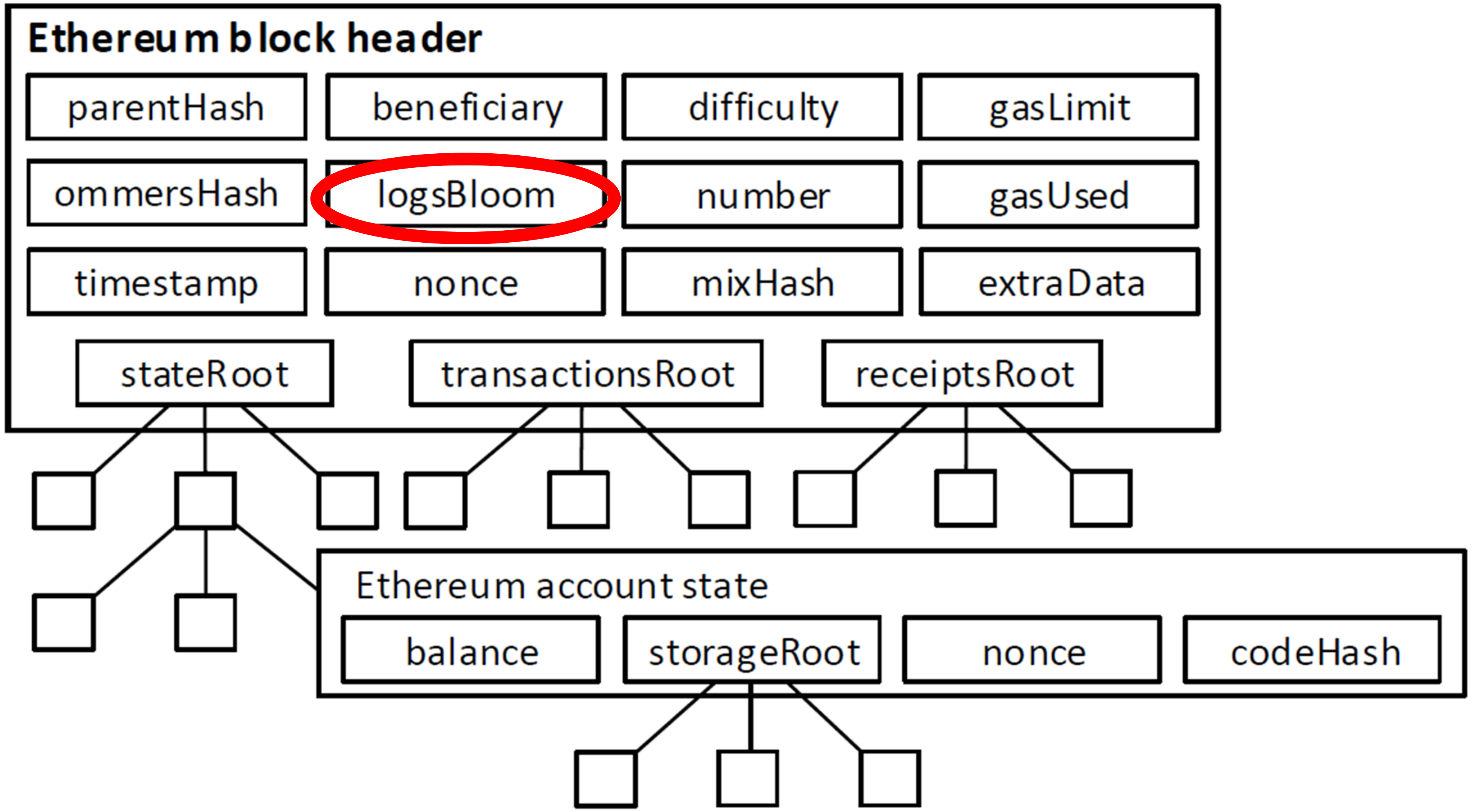
Blocks and blockchain | Block header : ommersHash

- The ommersHash field in the Ethereum block header is the hash of the block's "uncles."

- Uncle blocks are blocks that are not part of the main blockchain but are still valid. They are included in a block to reward miners who contribute to the network but do not create a valid block in time.

- The ommersHash field is used to ensure that uncle blocks are included correctly.

Blocks and blockchain | Block header : logsBloom



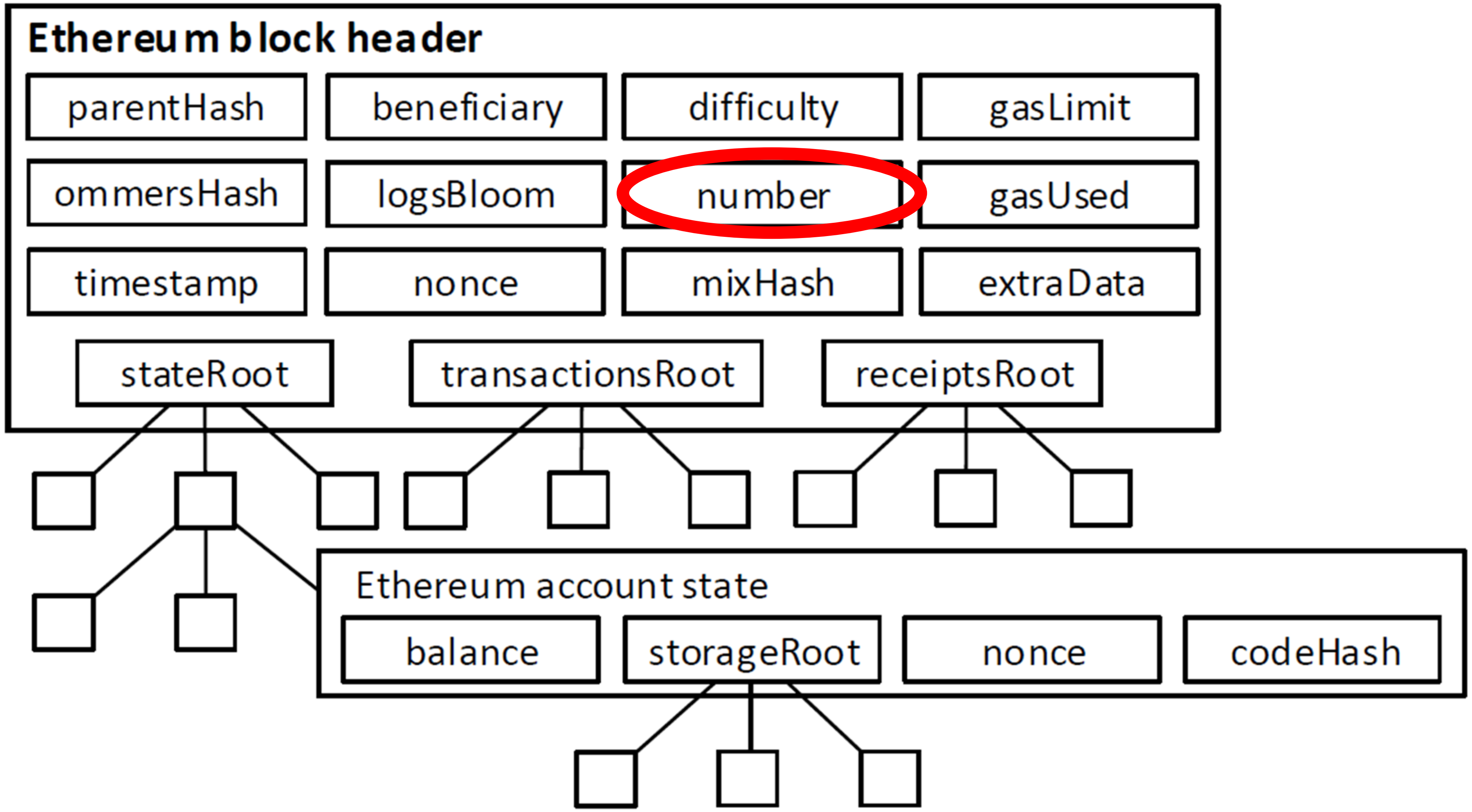
Blocks and blockchain | Block header : logsBloom

- The logsBloom field in the Ethereum block header is a data structure that is used to filter transaction logs efficiently.

- Transaction logs contain information about the state changes resulting from a transaction.

- The logsBloom field is used to help nodes quickly find the logs they need to process transactions.

Blocks and blockchain | Block header : number

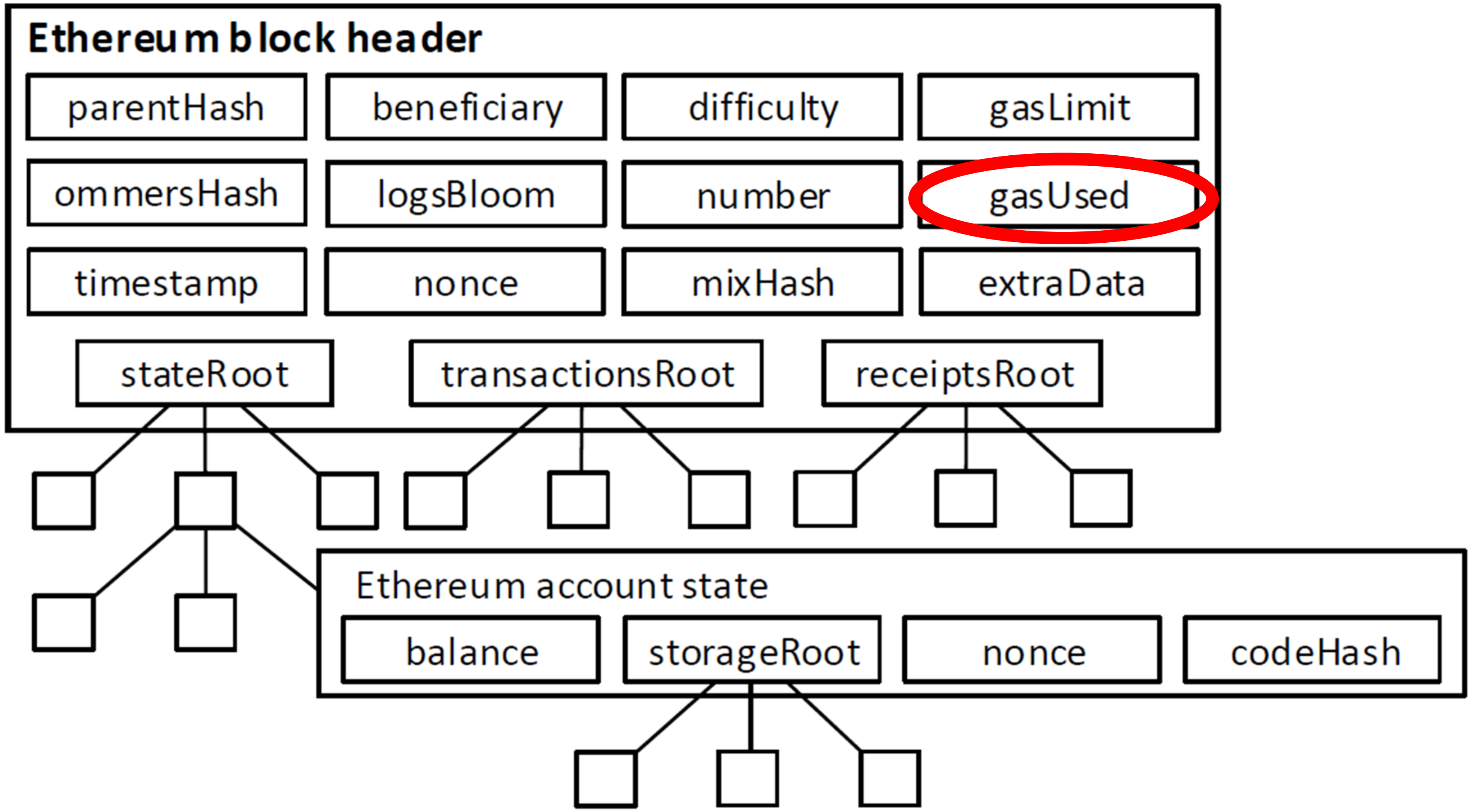


Blocks and blockchain | Block header : number

- The number field in the Ethereum block header represents the block number, a unique identifier that represents the position of the block in the blockchain.

- The genesis block, which is the first block in the blockchain, has a block number of zero.

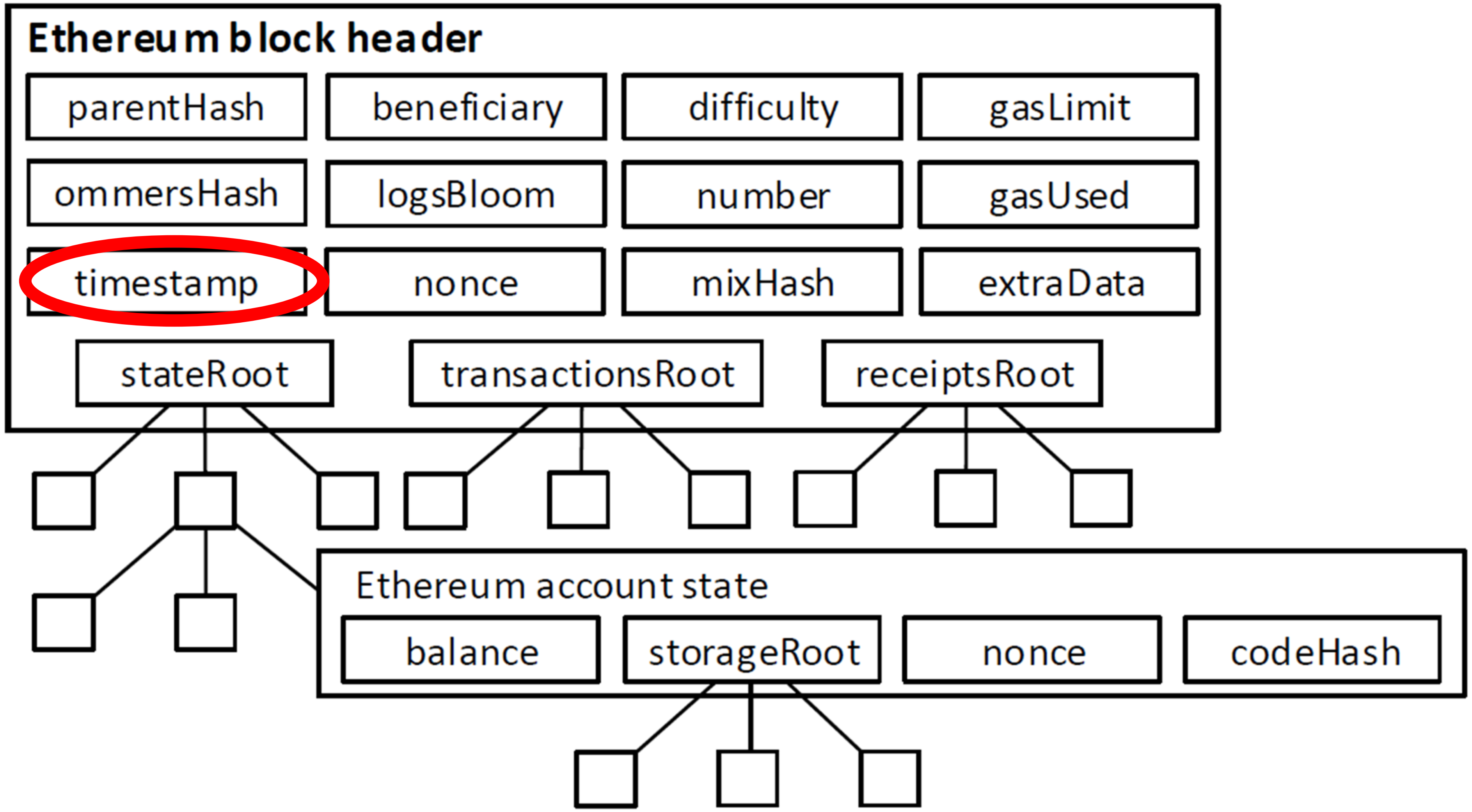
Blocks and blockchain | Block header : gasUsed



Blocks and blockchain | Block header : gasUsed

- The gasUsed field in the Ethereum block header represents the total amount of gas used by the transactions included in the block.
- This is used to calculate transaction fees and adjust the gas limit for future blocks.

Blocks and blockchain | Block header : timestamp

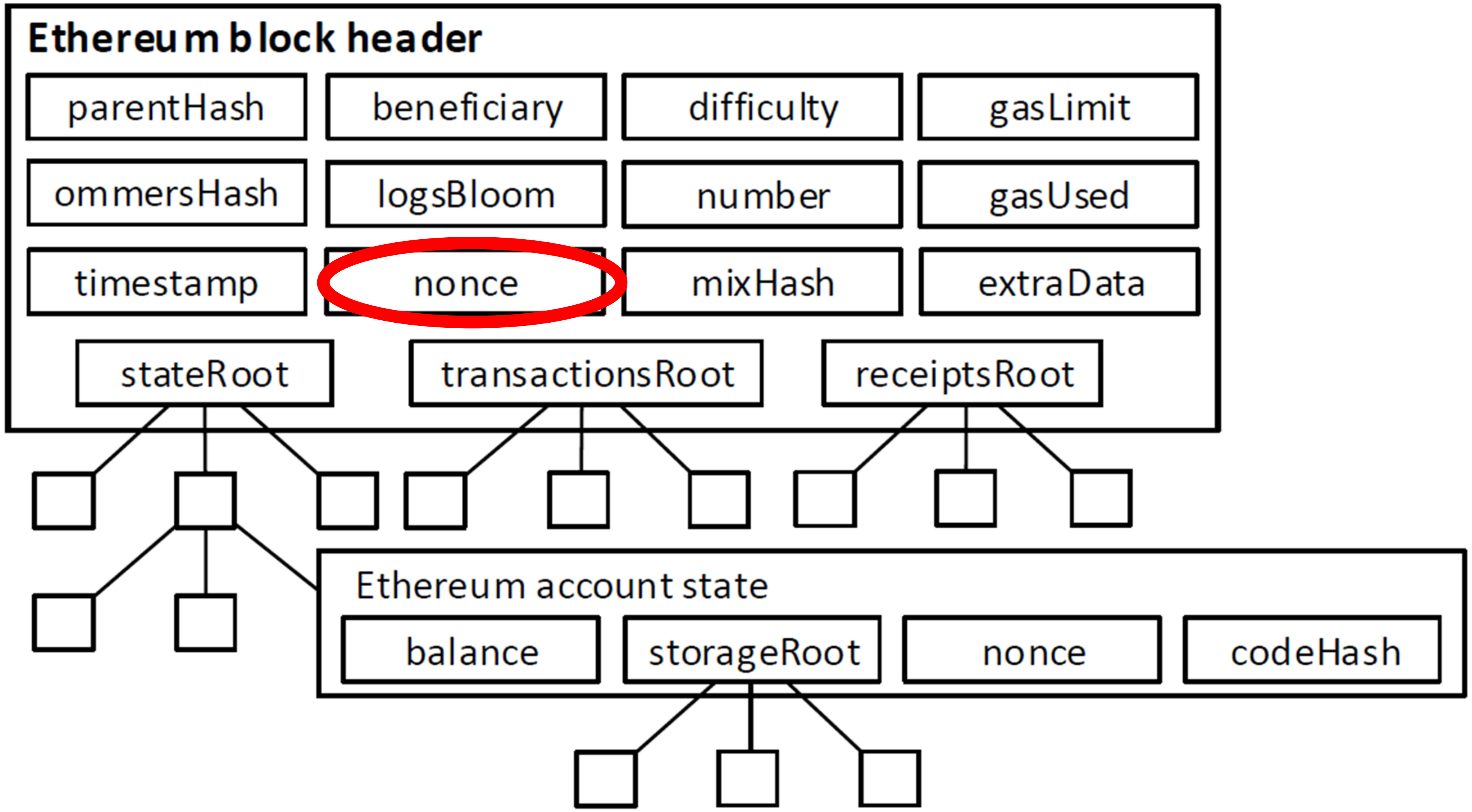


Blocks and blockchain | Block header : timestamp

- The timestamp field in the Ethereum block header represents the time when the block was created.

- This is a crucial element in maintaining the order of the blocks in the blockchain.

Blocks and blockchain | Block header : nonce



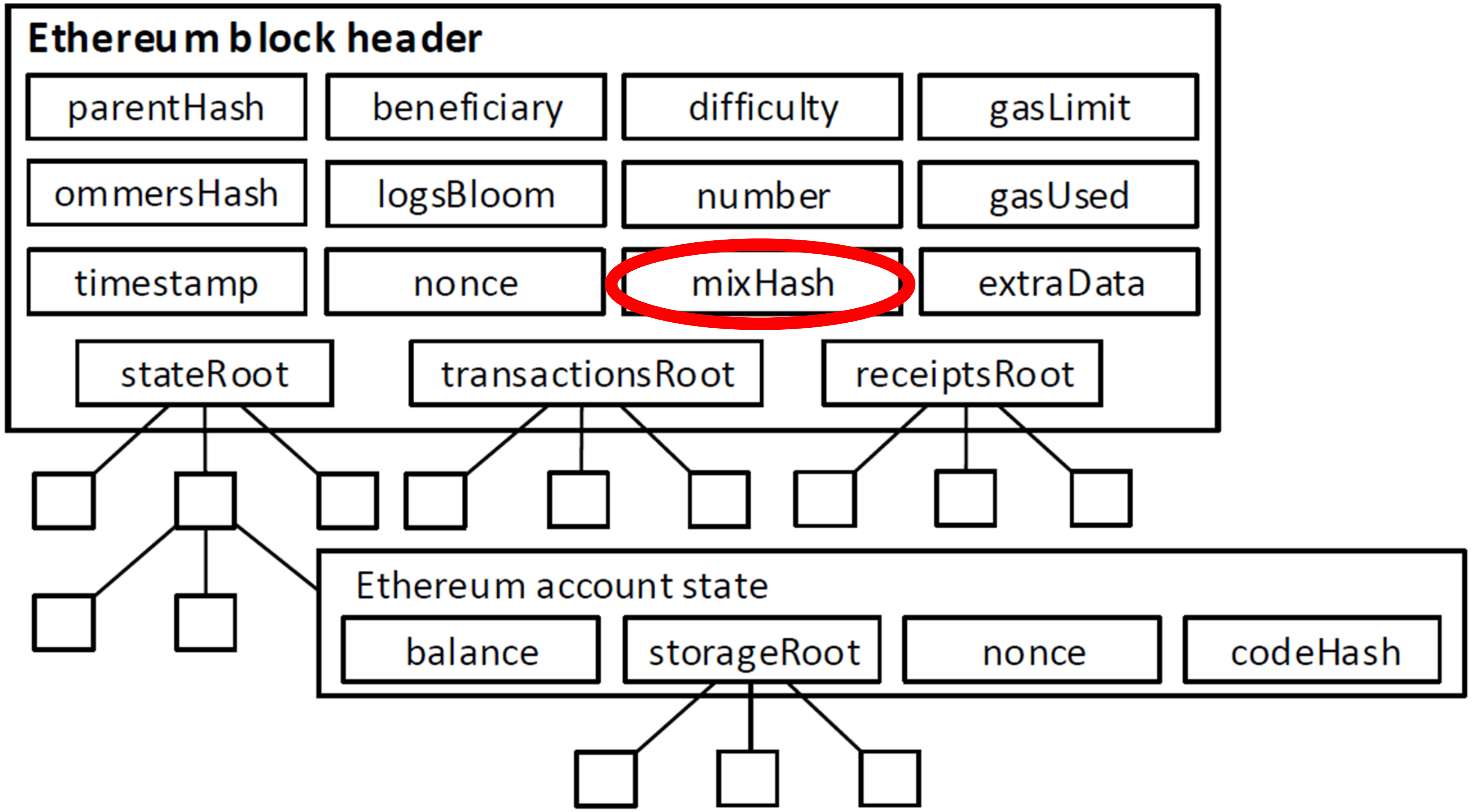
Blocks and blockchain | Block header : nonce

- The nonce field in the Ethereum block header is a random value used to prevent replay attacks.

- A replay attack is an attack in which a transaction is repeated, resulting in multiple identical transactions.

- The nonce ensures that each transaction is unique and prevents replay attacks.

Blocks and blockchain | Block header : mixHash

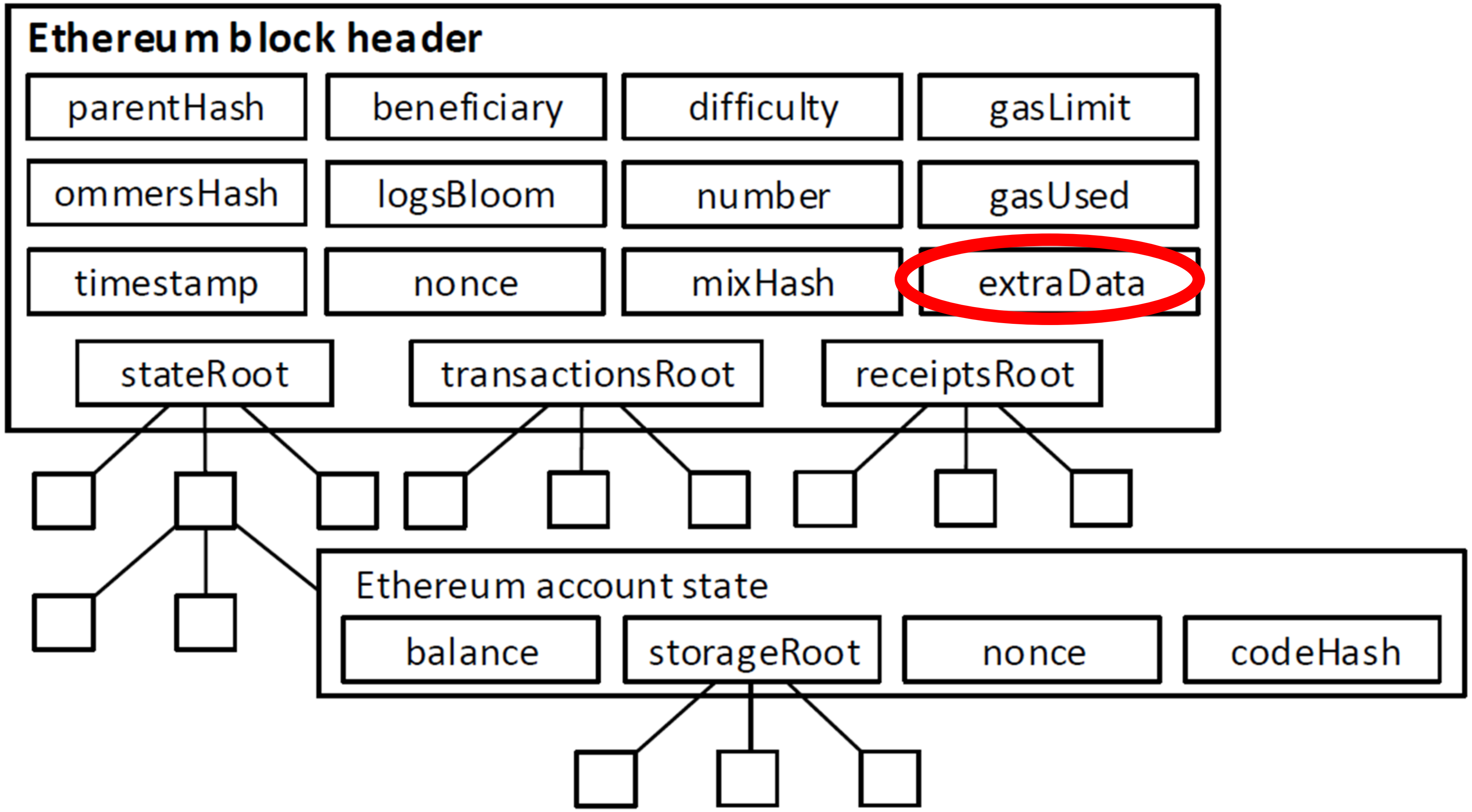


Blocks and blockchain | Block header : mixHash

- The mixHash field in the Ethereum block header is a 256-bit hash that, when combined with the nonce, is used to prove that the block creator has spent sufficient computational effort (Proof of Work) to create the block.

- The mixHash is combined with the nonce to create a unique hash for the block.

Blocks and blockchain | Block header : extraData



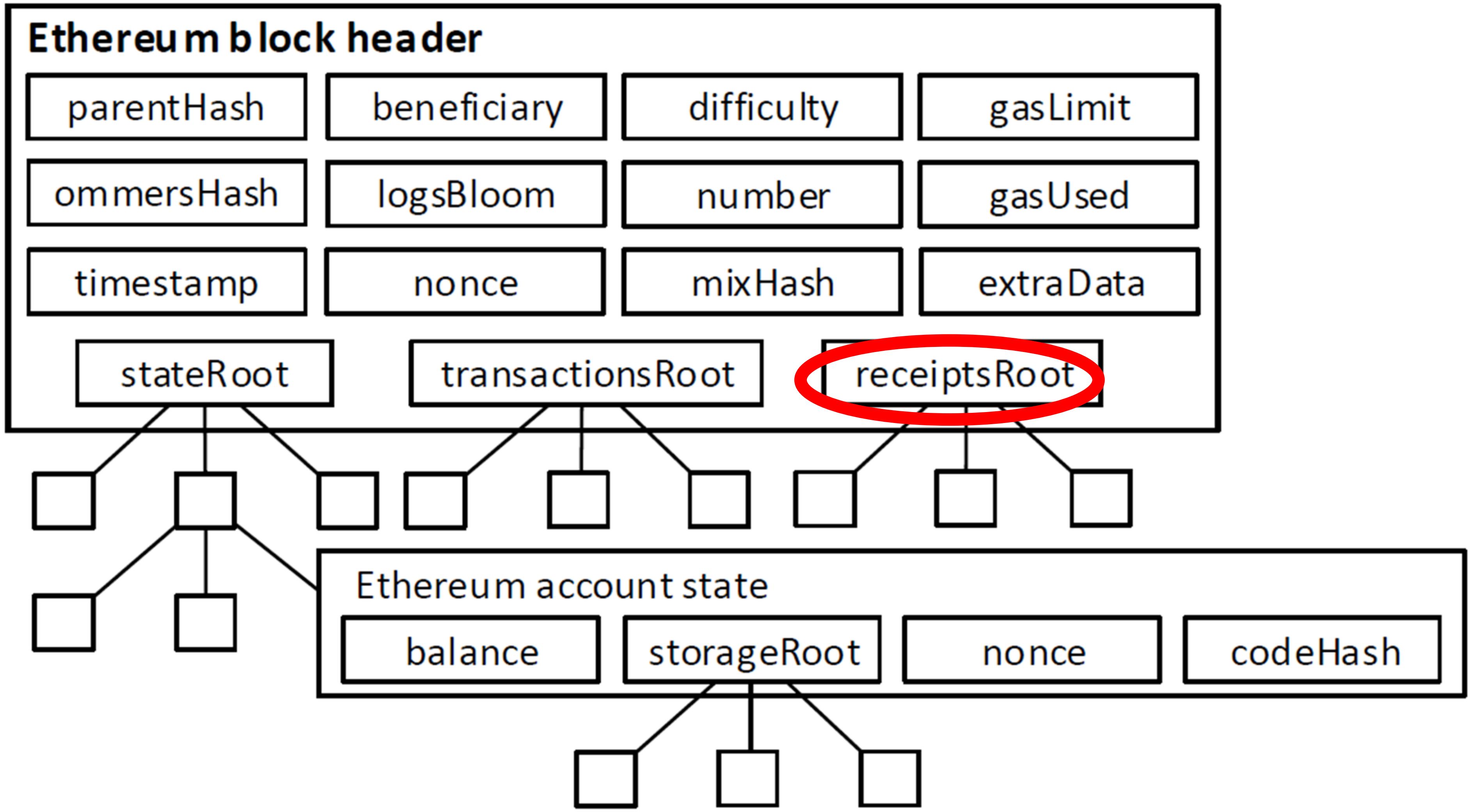
Blocks and blockchain | Block header : extraData

- The extraData field in the Ethereum block header can be used to store arbitrary data related to the block.

- Only up to 32 bytes are allowed in this field, and it can be used for various purposes such as identifying the miner or including metadata about the block.

- This field is entirely optional, and if not used, it will be empty.

Blocks and blockchain | Block header : receiptsRoot



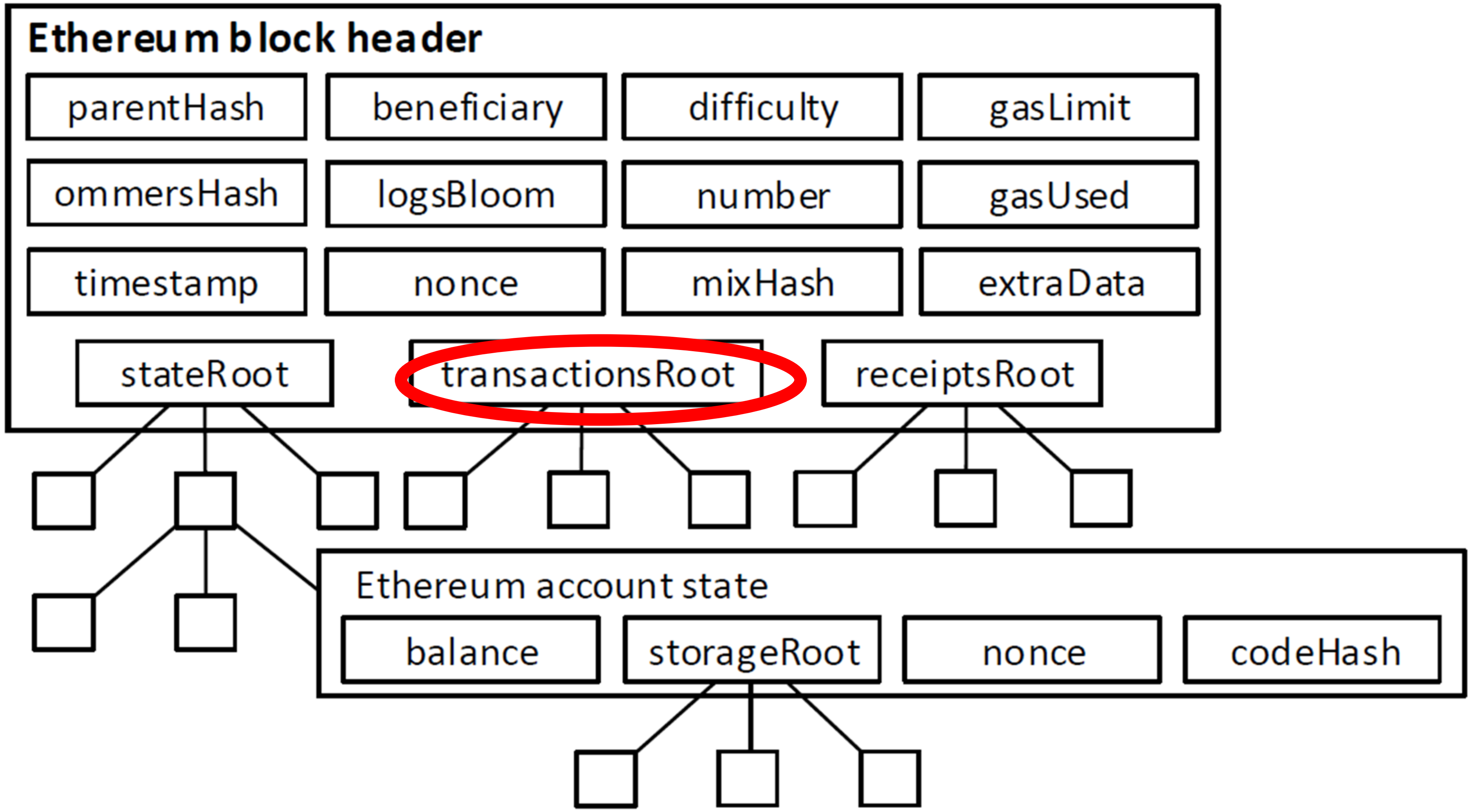
Blocks and blockchain | Block header : receiptsRoot

- The receiptsRoot field in the Ethereum block header is the root of the Merkle tree of receipts.

- A receipt is a data structure that contains information about the result of a transaction, such as whether it was successful or not.

- The receiptsRoot is used to validate the transactions included in the block.

Blocks and blockchain | Block header : transactionsRoot



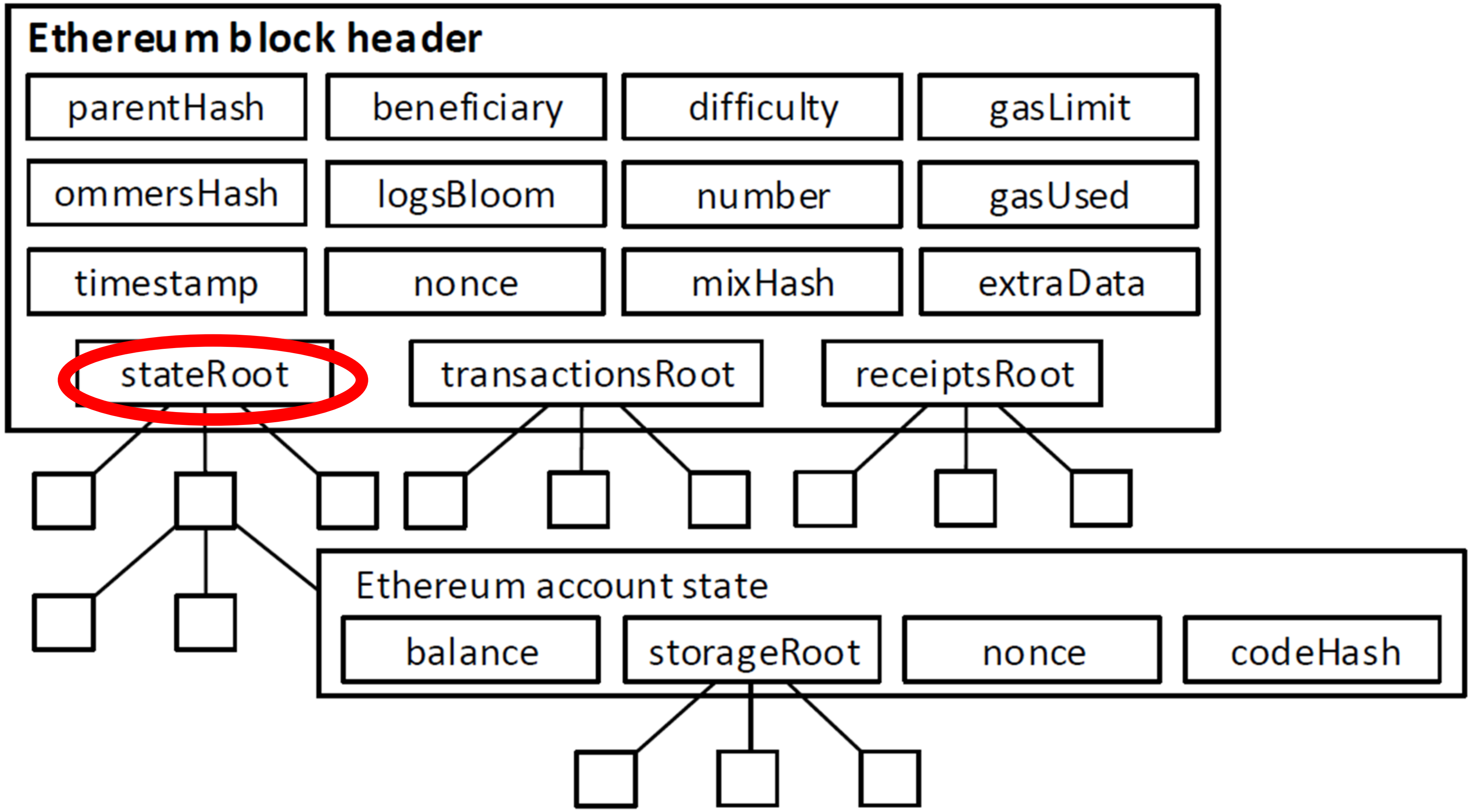
Blocks and blockchain | Block header : transactionsRoot

- The transactionsRoot field in the Ethereum block header is the root of the Merkle tree of transactions.

- The transactions included in the block are organized in a Merkle tree, with the transactionsRoot as the root of the tree.

- This field is used to validate the transactions included in the block.

Blocks and blockchain | Block header : stateRoot



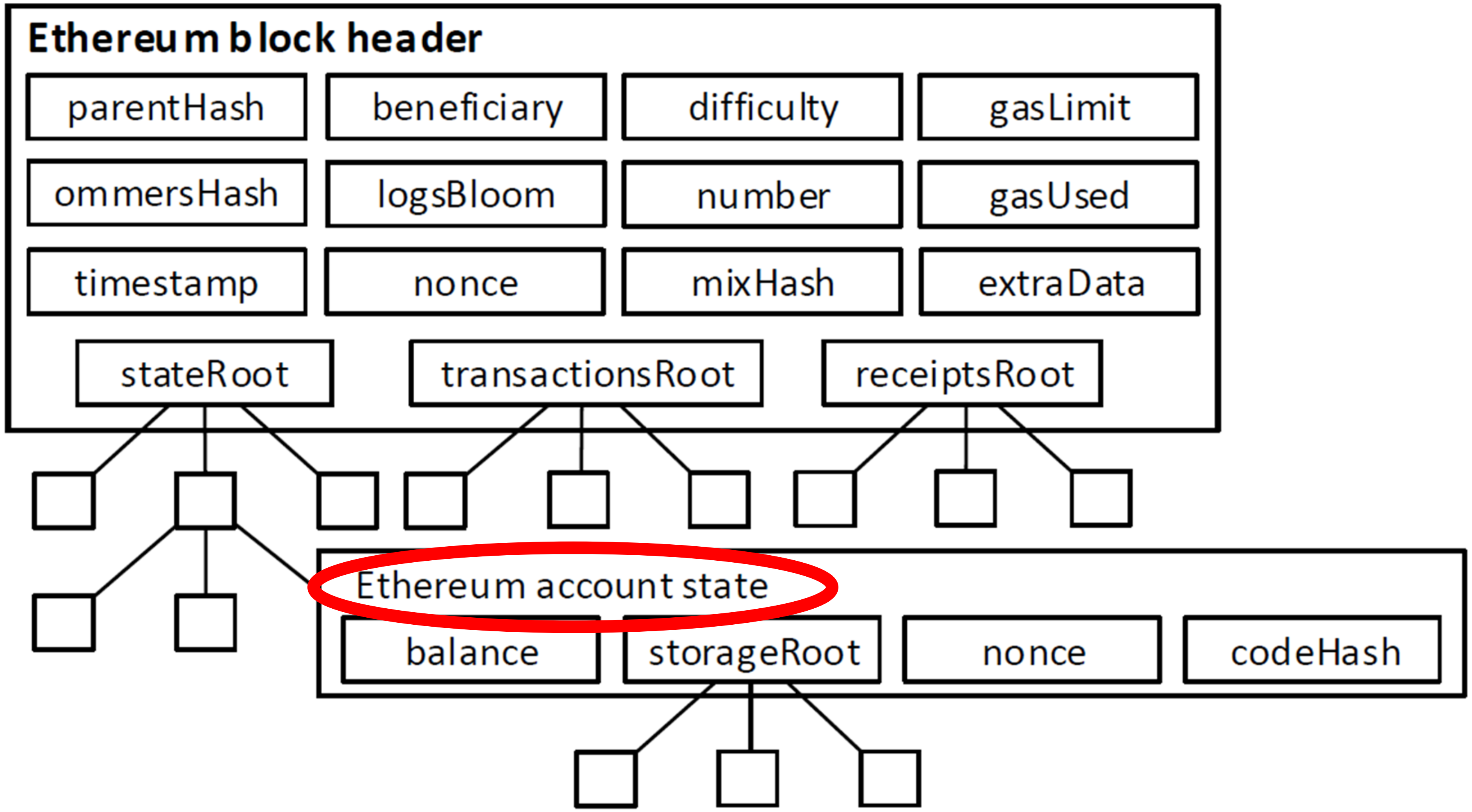
Blocks and blockchain | Block header : stateRoot

- The stateRoot field in the Ethereum block header is the root of the Merkle tree of the Ethereum account state.

- The Ethereum account state contains the balance, storageRoot, nonce, and codeHash fields.

- The stateRoot is used to validate the account state of the transactions included in the block.

Blocks and blockchain | Block header : Ethereum account state

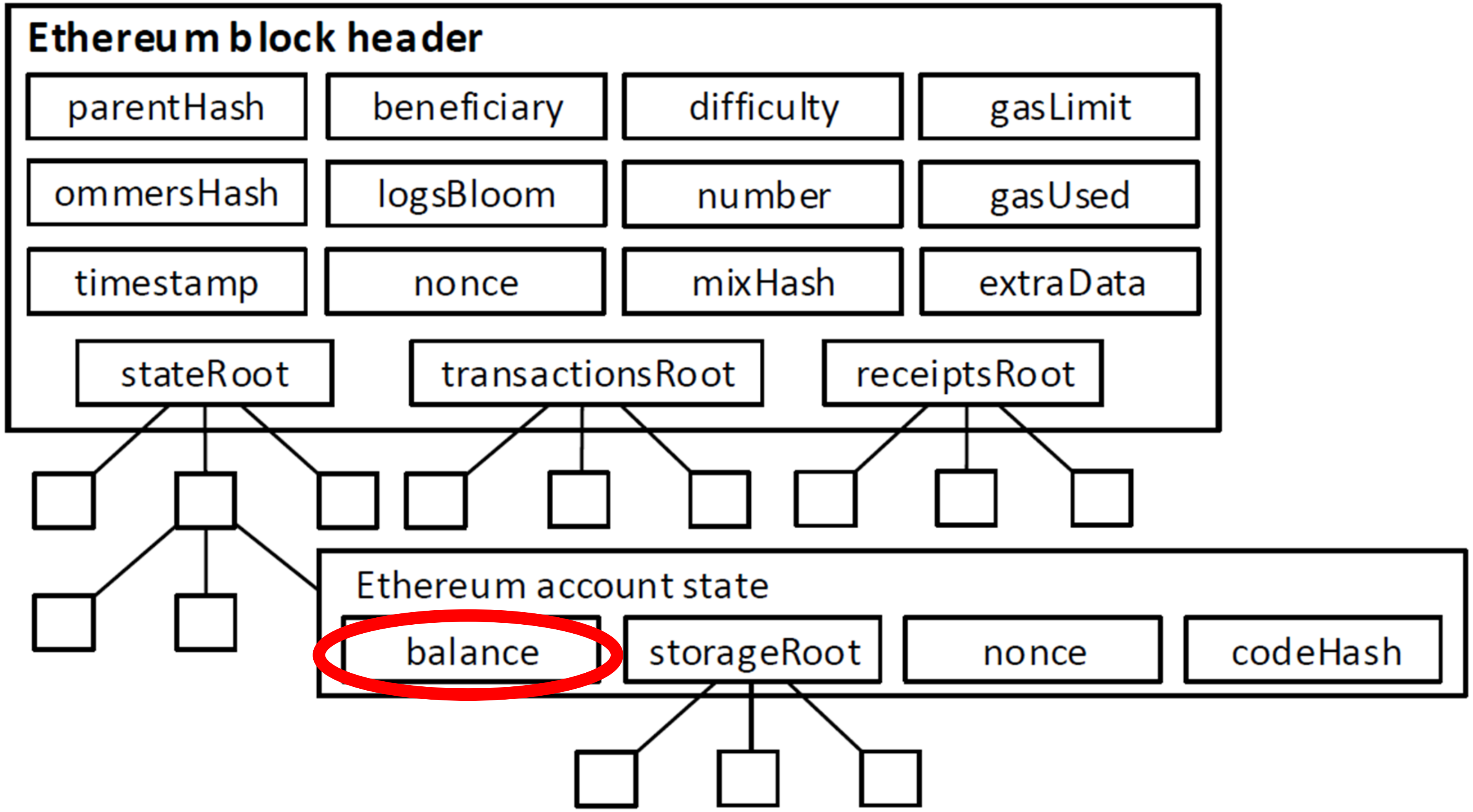


Blocks and blockchain | Block header : Ethereum account state

- The Ethereum account state is the state of all accounts on the Ethereum network.

- Each account on the Ethereum network has a unique address and contains four fields: balance, storageRoot, nonce, and codeHash.

Blocks and blockchain | Ethereum account state: balance

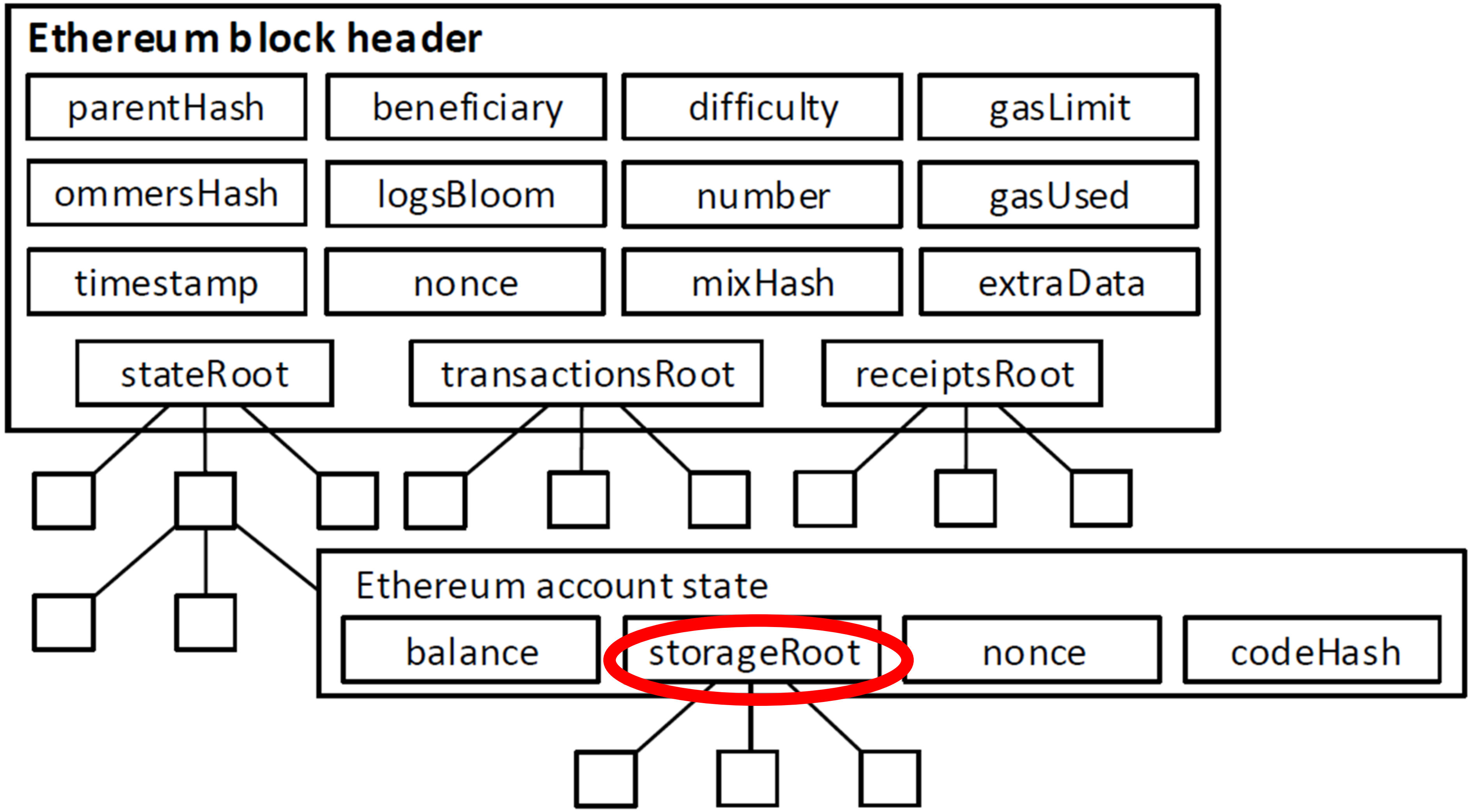


Blocks and blockchain | Ethereum account state: balance

- The balance field in an account represents the amount of ether owned by the account.

- Ether is the native cryptocurrency of the Ethereum network and is used to pay for transaction fees and smart contract executions.

Blocks and blockchain | Ethereum account state: storageRoot

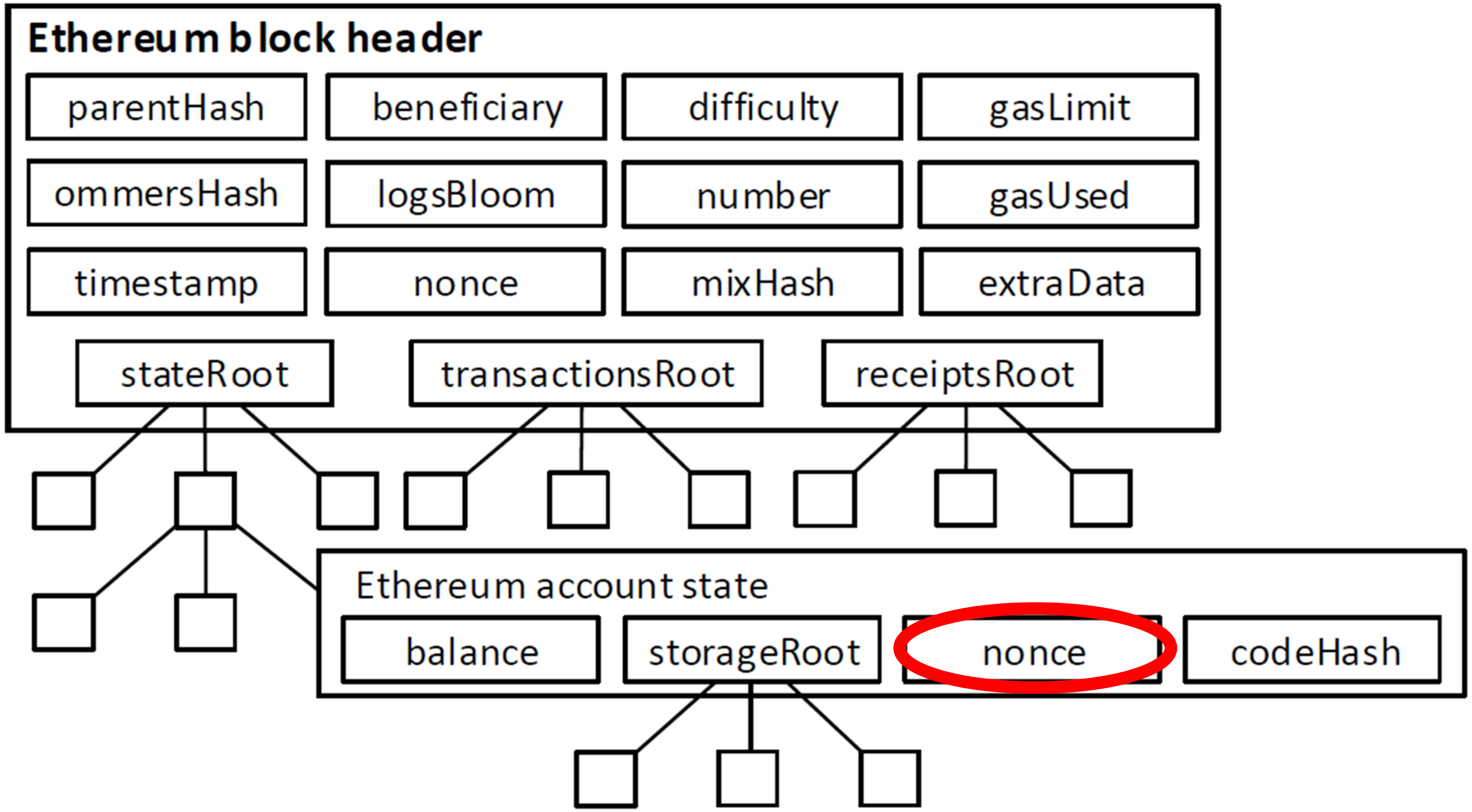


Blocks and blockchain | Ethereum account state: storageRoot

- The storageRoot field in an account represents the root of the Merkle tree that stores the account's storage data.

- Storage data includes smart contract state variables, which are persistent data stored on the blockchain.

Blocks and blockchain | Ethereum account state: nonce

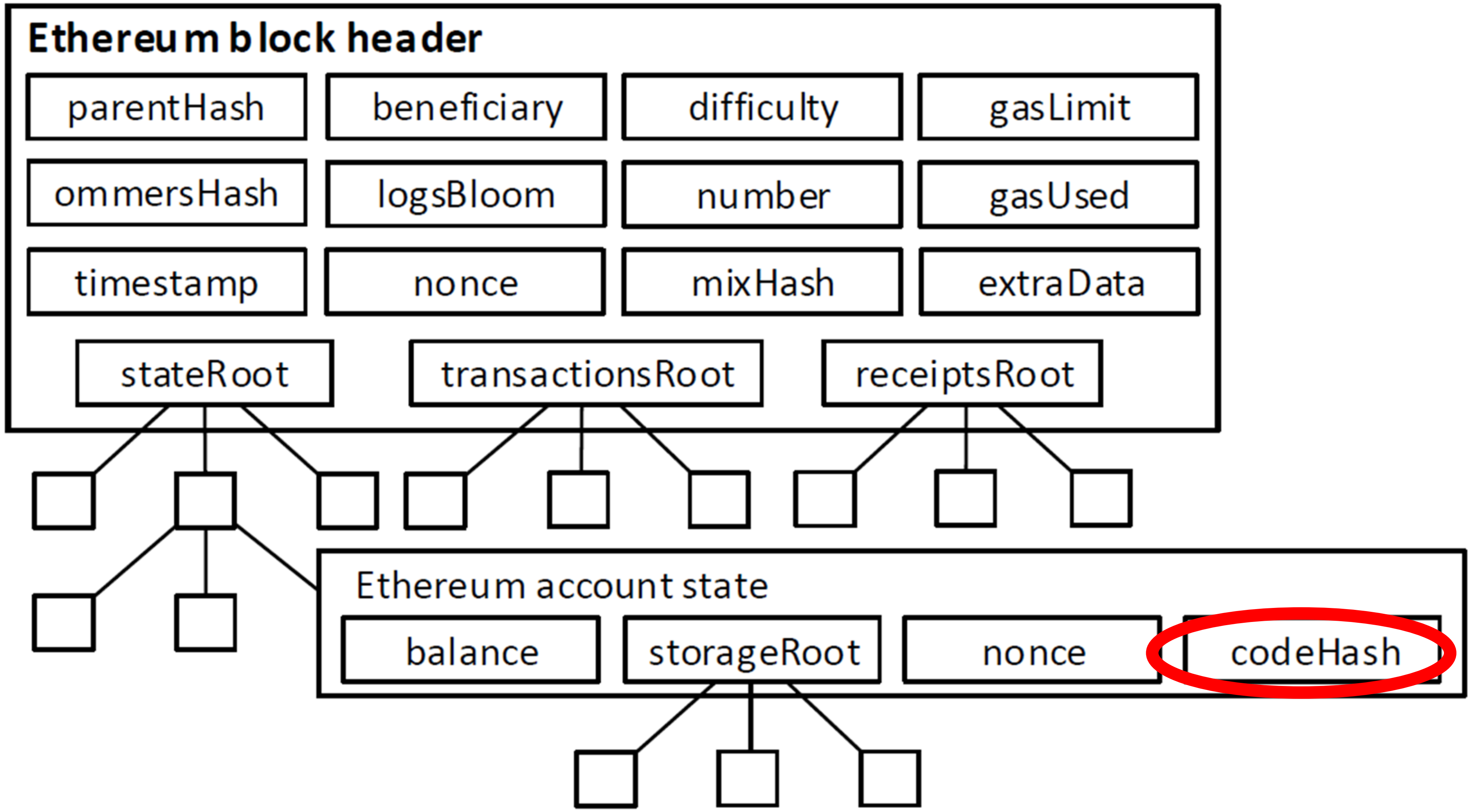


Blocks and blockchain | Ethereum account state: nonce

- The nonce field in an account is a value that is incremented each time a transaction is sent from the account.

- This ensures that each transaction from an account is unique and prevents replay attacks.

Blocks and blockchain | Ethereum account state: codeHash

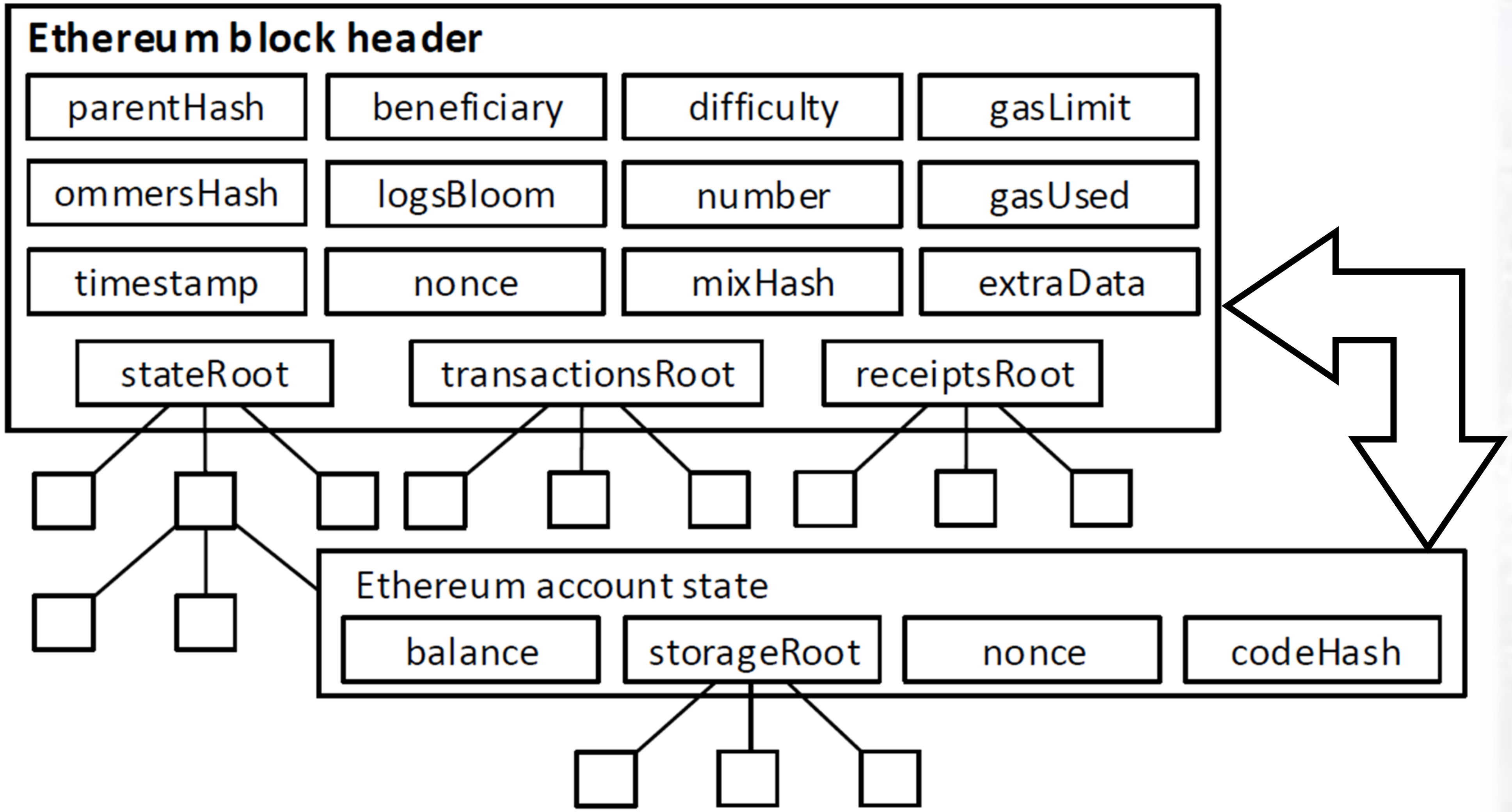


Blocks and blockchain | Ethereum account state: codeHash

- The codeHash field in an account represents the hash of the smart contract code stored on the blockchain.

- Smart contract code is executed on the Ethereum Virtual Machine (EVM), and the codeHash field ensures that the code is secure and has not been tampered with.

Blocks and blockchain | Block header and Account state relationship



Blocks and blockchain | Block header and Account state relationship

- The block header and the Ethereum account state are closely related. The stateRoot field in the block header is the root of the Merkle tree that contains the Ethereum account state.

- The stateRoot is used to validate the account state of the transactions included in the block. If the stateRoot is incorrect, the block is invalid, and the network will reject it.

Blocks and blockchain | Block header and Account state relationship

- When a transaction is executed, it changes the account state on the Ethereum network.

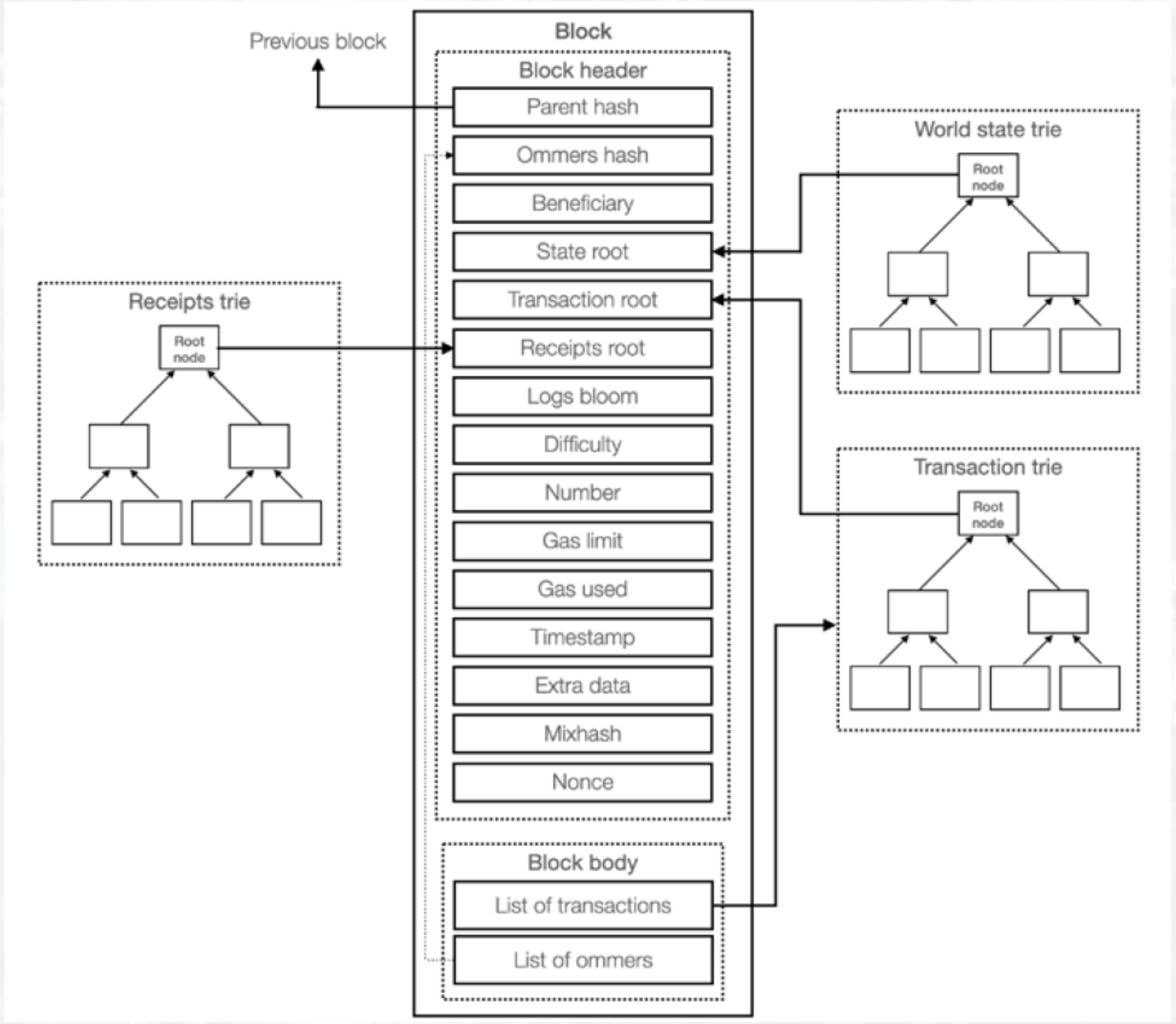
- The changes to the account state are recorded in the blockchain, and the stateRoot field in the block header is updated to reflect the new state.

- The stateRoot field in the block header is used to validate the transactions included in the block, ensuring that they are executed correctly and that the account state is maintained.

Ethereum Blockchain Elements | Blocks and blockchain

Block header

- The following diagram shows the detailed structure of the block and block header:



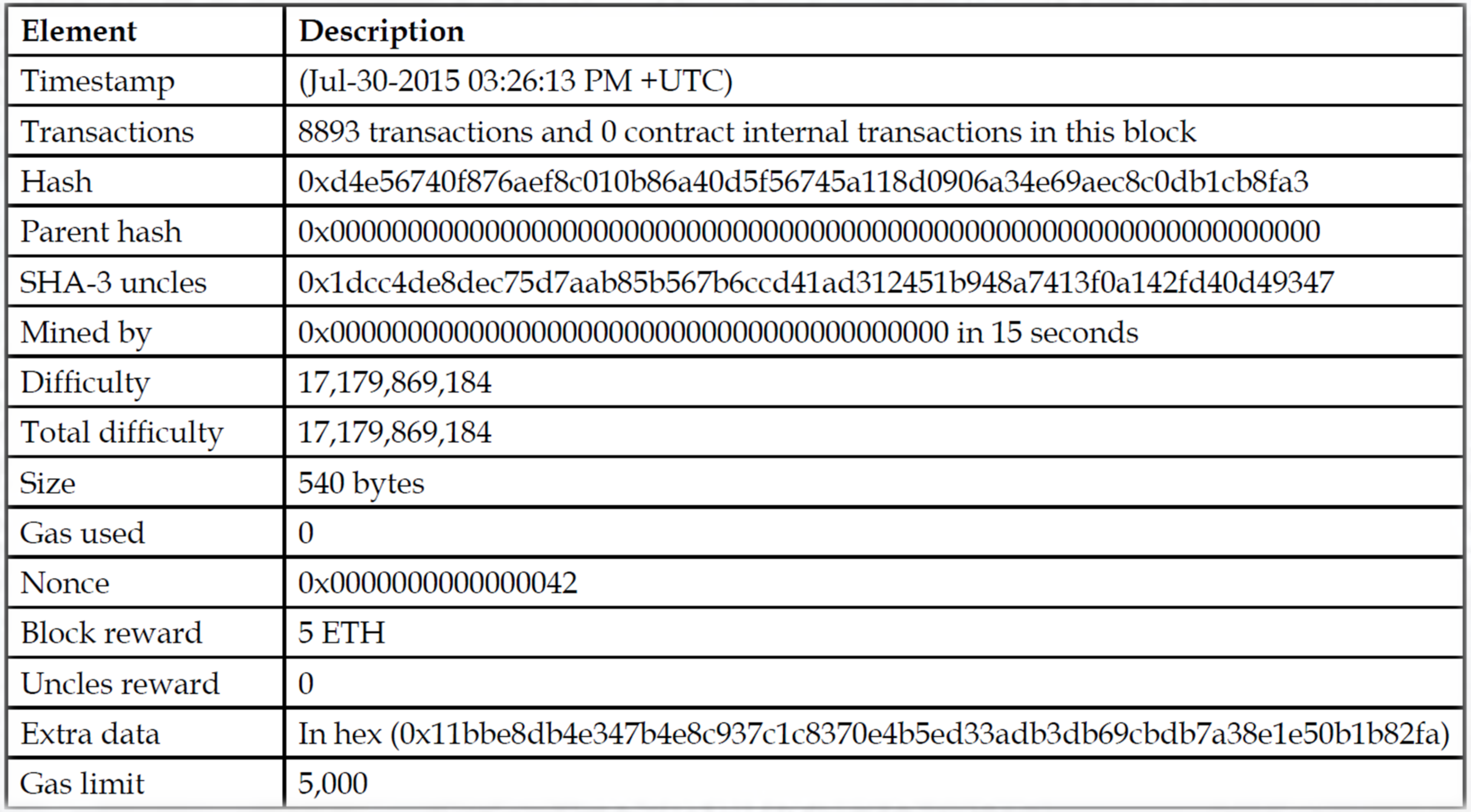
Ethereum Blockchain Elements | Blocks and blockchain

The genesis block

- The genesis block is the first block in a blockchain network.
- It varies slightly from normal blocks due to the data it contains and the way it has been created.
- It contains 15 items that are described in the next slide:

Ethereum Blockchain Elements | Blocks and blockchain

The genesis block



Element	Description
Timestamp	(Jul-30-2015 03:26:13 PM +UTC)
Transactions	8893 transactions and 0 contract internal transactions in this block
Hash	0xd4e56740f876aef8c010b86a40d5f56745a118d0906a34e69aec8c0db1cb8fa3
Parent hash	0x0000000000000000000000000000000000000000000000000000000000000000
SHA-3 uncles	0x1dcc4de8dec75d7aab85b567b6ccd41ad312451b948a7413f0a142fd40d49347
Mined by	0x0000000000000000000000000000000000000000 in 15 seconds
Difficulty	17,179,869,184
Total difficulty	17,179,869,184
Size	540 bytes
Gas used	0
Nonce	0x0000000000000042
Block reward	5 ETH
Uncles reward	0
Extra data	In hex (0x11bbe8db4e347b4e8c937c1c8370e4b5ed33adb3db69cbdb7a38e1e50b1b82fa)
Gas limit	5,000

Ethereum Blockchain Elements | Blocks and blockchain

The block validation mechanism

- An Ethereum block is considered valid if it passes the following checks:
  - If it is consistent with uncles and transactions. This means that all ommers satisfy the property that they are indeed uncles and also if the PoW for uncles is valid.
  - If the previous block (parent) exists and is valid.
  - If the timestamp of the block is valid. This means that the current block's timestamp must be higher than the parent block's timestamp. Also, it should be less than 15 minutes into the future. All block times are calculated in epoch time (Unix time).

Ethereum Blockchain Elements | Blocks and blockchain

The block validation mechanism

- An Ethereum block is considered valid if it passes the following checks:
  - If any of these checks fails, the block will be rejected. A list of errors for which the block can be rejected is presented here:
    - The timestamp is older than the parent.
    - There are too many uncles.
    - There is a duplicate uncle.
    - The uncle is an ancestor.
    - The uncle's parent is not an ancestor.
    - There is non-positive difficulty.
    - There is an invalid mix digest.
    - There is an invalid PoW.

Ethereum Blockchain Elements | Blocks and blockchain
Block finalization

- Block finalization is a process that is run by miners to validate the contents of the block and apply rewards. It results in four steps being executed:

1. **Ommers validation:** In the case of mining, determine ommers. The validation process of the headers of stale blocks checks whether the header is valid and whether the relationship between the uncle and the current block satisfies the maximum depth of six blocks. A block can contain a maximum of two uncles.

2. **Transaction validation:** In the case of mining, determine transactions. This process involves checking whether the total gas used in the block is equal to the final gas consumption after the final transaction, in other words, the cumulative gas used by the transactions included in the block.

Ethereum Blockchain Elements | Blocks and blockchain

Block finalization

- Block finalization is a process that is run by miners to validate the contents of the block and apply rewards. It results in four steps being executed:

3. **Reward application:** Apply rewards, which means updating the beneficiary's account with a reward balance. In Ethereum, a reward is also given to miners for stale blocks, which is 1/32 of the block reward. Uncles that are included in the blocks also receive 7/8 of the total block reward. The current block reward is 2 ether. It was reduced first from 5 ether to 3 with the Byzantium release of Ethereum. Later, in the Constantinople release (https://blog.ethereum.org/2019/02/22/ethereum-constantinople-st-petersburgupgrade-announcement/), it was reduced further to 2 ether. A block can have a maximum of two uncles..
4. **State and nonce validation:** Verify the state and block nonce. In the case of mining, compute a valid state and block nonce.

Ethereum Blockchain Elements | Blocks and blockchain

Block finalization

- After the high-level view of the block validation mechanism, we now look into how a block is received and processed by a node. We will also see how it is updated in the local blockchain:

  1. When an Ethereum full node receives a newly mined block, the header and the body of the block are detached from each other. Now remember, in the last session, when we introduced the fact that there are three Merkle Patricia tries (MPTs) in an Ethereum blockchain. The roots of those MPTs or tries are present in each block header as a state trie root node, a transaction trie root node, and a receipt trie root node. We will now learn how these tries are used to validate the blocks.

  2. A new MPT is constructed that comprises all transactions from the block.

Ethereum Blockchain Elements | Blocks and blockchain

Block finalization

- After the high-level view of the block validation mechanism, we now look into how a block is received and processed by a node. We will also see how it is updated in the local blockchain:

3. All transactions from this new MPT are executed one by one in a sequence. This execution occurs locally on the node within the Ethereum Virtual Machine (EVM). As a result of this execution, new transaction receipts are generated that are organized in a new receipts MPT. Also, the global state is modified accordingly, which updates the state MPT (trie).

4. The root nodes of each respective trie, in other words, the state root, transaction root, and receipts root are compared with the header of the block that was split in the first step. If both the roots of the newly constructed tries and the trie roots that already exist in the header are equal, then the block is verified and valid.

Ethereum Blockchain Elements | Blocks and blockchain

Block finalization

- After the high-level view of the block validation mechanism, we now look into how a block is received and processed by a node. We will also see how it is updated in the local blockchain:

  5. Once the block is validated, new transaction, receipt, and state tries are written into the local blockchain database.

Ethereum Blockchain Elements | Blocks and blockchain

Block difficulty mechanism

- Block difficulty is increased if the time between two blocks decreases, whereas it increases if the block time between two blocks decreases.
- This is required to maintain a roughly consistent block generation time.
- The difficulty adjustment algorithm in Ethereum's Homestead release is as follows:

  block_diff = parent_diff + parent_diff // 2048 * max(1 - (block_timestamp - parent_timestamp) // 10, -99) + int(2**((block.number // 100000) - 2)).

Ethereum Blockchain Elements | Blocks and blockchain

Block difficulty mechanism

- The preceding algorithm means that, if the time difference between the generation of the parent block and the current block is less than 10 seconds, the difficulty goes up by parent_diff // 2048 * 1.
- If the time difference is between 10 and 19 seconds, the difficulty level remains the same.
- Finally, if the time difference is 20 seconds or more, the difficulty level decreases. This decrease is proportional to the time difference from parent_diff // 2048 * -1 to a maximum decrease of parent_diff // 2048 * -99.

Ethereum Blockchain Elements | Blocks and blockchain

Block difficulty mechanism

- In addition to timestamp difference-based difficulty adjustment, there is also another part (shown in the last line of the preceding algorithm) that increases the difficulty exponentially after every 100,000 blocks.
- This is the so-called difficulty time bomb, or ice age, introduced in the Ethereum network, which will make it very hard to mine on the Ethereum blockchain at some point in the future.
- This will encourage users to switch to Proof of Stake (PoS), since mining on the PoW chain will eventually become prohibitively difficult.

Ethereum Blockchain Elements | Blocks and blockchain

Block difficulty mechanism

- According to the original estimates based on the algorithm, the block generation time would have become significantly higher during the second half of 2017, and in 2023, it would become so high that it would be virtually impossible to mine on the PoW chain, even for dedicated mining centers.
- This way, miners will have no choice but to switch to the PoS scheme proposed by Ethereum.
- In the Byzantium release, the difficulty adjustment formula has been changed to take uncles into account for difficulty calculation. This new formula is shown here:

  adj_factor = max((2 if len(parent.uncles) else 1) - ((timestamp - parent.timestamp) // 9), -99).

Ethereum Blockchain Elements | Blocks and blockchain

Gas

- We know that blocks are composed of transactions and that there are various operations associated with transaction creation, validation, and finalization.
- Each of these operations on the Ethereum network costs some amount of ETH and is charged using a fee mechanism. This fee is also called gas.
- Gas is required to be paid for every operation performed on the Ethereum blockchain.
- This is a mechanism that ensures that infinite loops cannot cause the whole blockchain to stall due to the Turing-complete nature of the EVM.
- A transaction fee is charged as an amount of Ether and is taken from the account balance of the transaction originator.

Ethereum Blockchain Elements | Blocks and blockchain

Gas

- A fee is paid for transactions to be included by miners for mining. If this fee is too low, the transaction may never be picked up; the more the fee, the higher are the chances that the transactions will be picked up by the miners for inclusion in the block.
- If the transaction that has an appropriate fee paid is included in the block by miners but has too many complex operations to perform, it can result in an out-of-gas exception if the gas cost is not enough.
- In this case, the transaction will fail but will still be made part of the block, and the transaction originator will not get any refund.

Ethereum Blockchain Elements | Blocks and blockchain

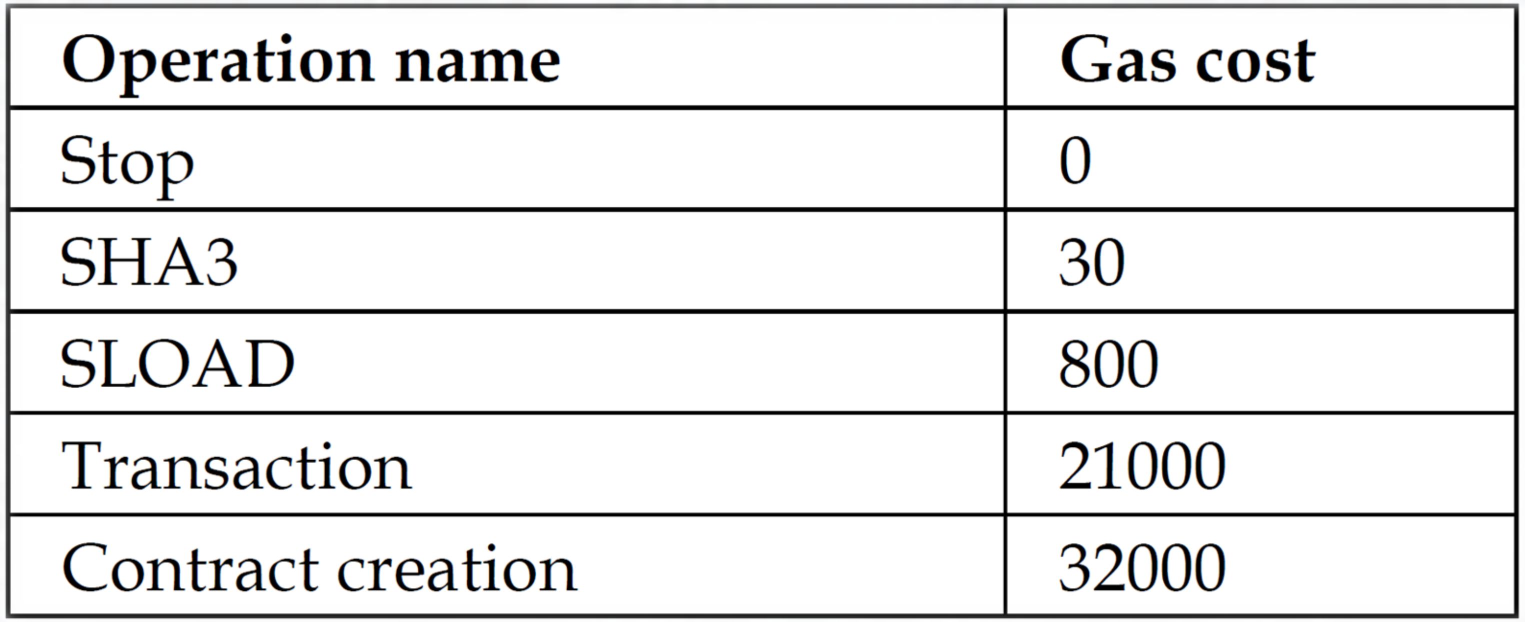
Gas

- Transaction costs can be estimated using the following formula:
  - ***Total cost = gasUsed * gasPrice***
- *gasUsed* is the total gas that is supposed to be used by the transaction during the execution.
- *gasPrice* is specified by the transaction originator as an incentive to the miners to include the transaction in the next block.
- Each EVM opcode has a fee assigned to it. It is an estimate because the gas used can be more or less than the value specified by the transaction originator originally

Ethereum Blockchain Elements | Blocks and blockchain

Gas

- Each operation costs some gas; a high-level fee schedule of a few operations is shown as an example here:



Operation name	**Gas cost**
Stop	0
SHA3	30
SLOAD	800
Transaction	21000
Contract creation	32000

Ethereum Blockchain Elements | Blocks and blockchain

Gas

- Based on the preceding fee schedule and the formula discussed earlier, an example calculation of the SHA-3 operation can be calculated as follows:
  - SHA-3 costs 30 gas.
  - Assume that the current gas price is 25 GWei, and convert it into ETH, which is 0.000000025 ETH. After multiplying both, 0.000000025 * 30, we get 0.00000075 ETH.
  - In total, 0.00000075 ETH is the total gas that will be charged.

Ethereum Blockchain Elements | Blocks and blockchain

Fee schedule

- Gas is charged in three scenarios as a prerequisite to the execution of an operation:
  - The computation of an operation.
  - For contract creation or message calls.
  - An increase in the use of memory.

Introduction to Blockchain | Summary

►**In this topic, we discussed:**

- Ethereum blocks and blockchain
  - Block header
  - Genesis Block
  - Block validation mechanism
  - Block finalization.
  - Block difficulty mechanism
  - Gas and fee structure

THANK YOU