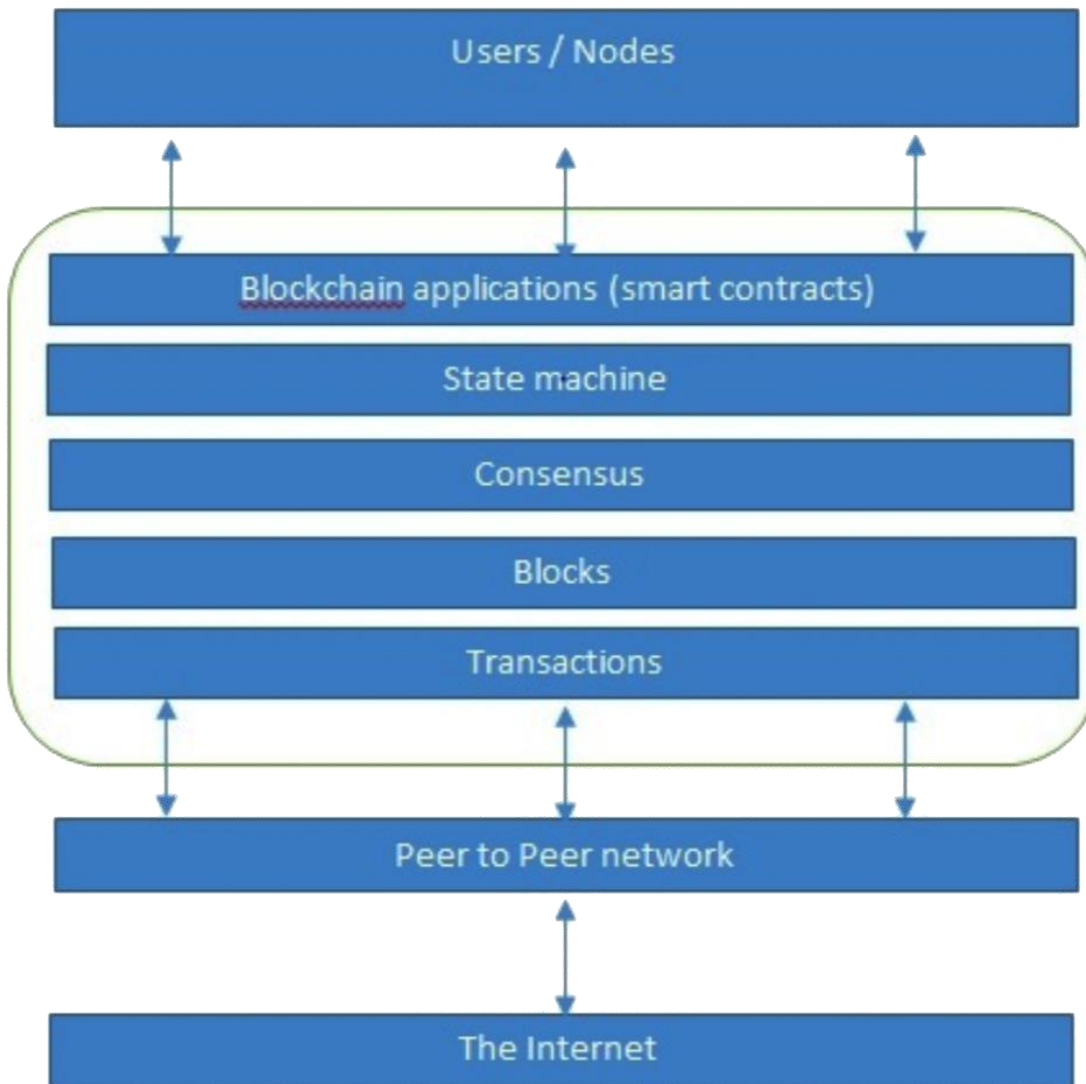Blockchain

BLUEPRINT FOR A NEW ECONOMY

Ethereum Architecture

# Foundation of Blockchain Architecture

Users / Nodes

Blockchain applications (smart contracts)
State machine
Consensus
Blocks
Transactions

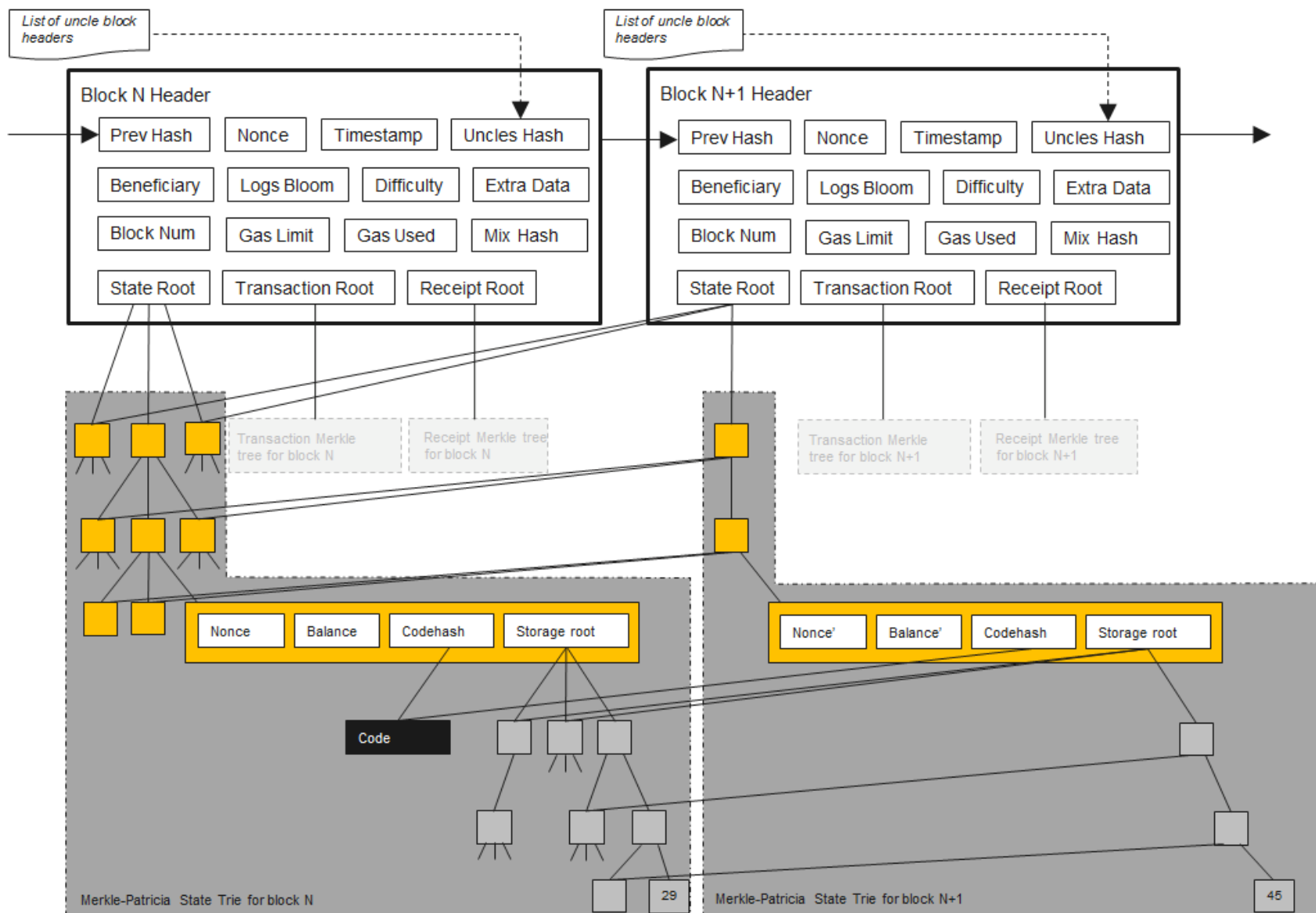| Peer to Peer network |
| --- |

The Internet

Blockchain at its core is a peer-to-peer distributed ledger that is cryptographically secure, append-only, immutable (extremely hard to change), and updateable only via consensus or agreement among peers.
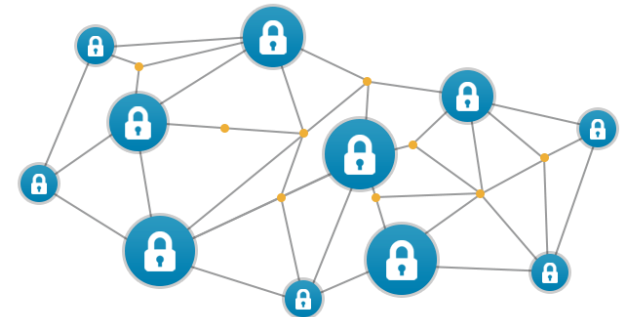
Bitcoin block structure
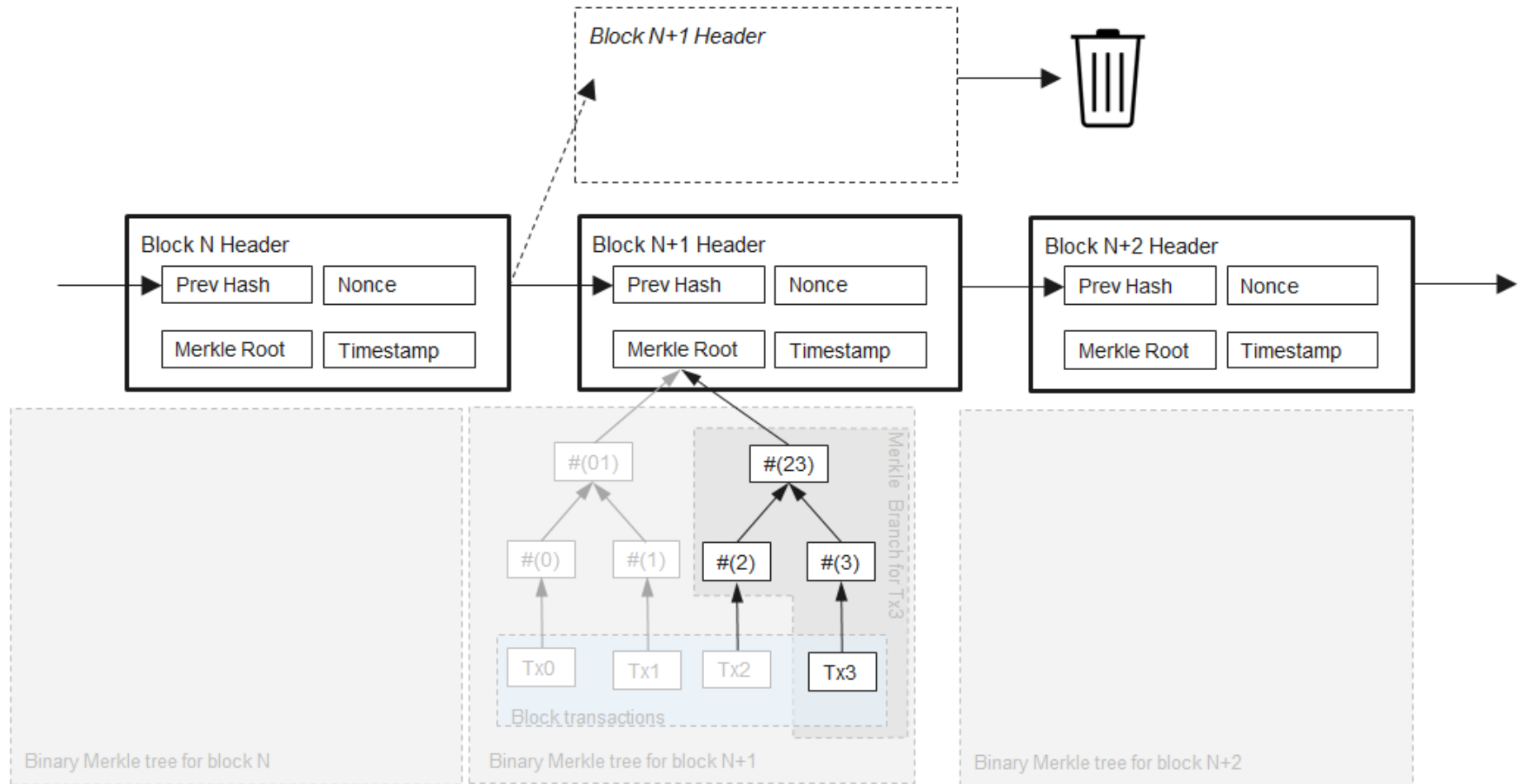
Ethereum Block Structure

Concensus mechanism in Ethereum

BITCOIN, LONGEST CHAIN RULE

ETHEREUM, HEAVIEST CHAIN RULE
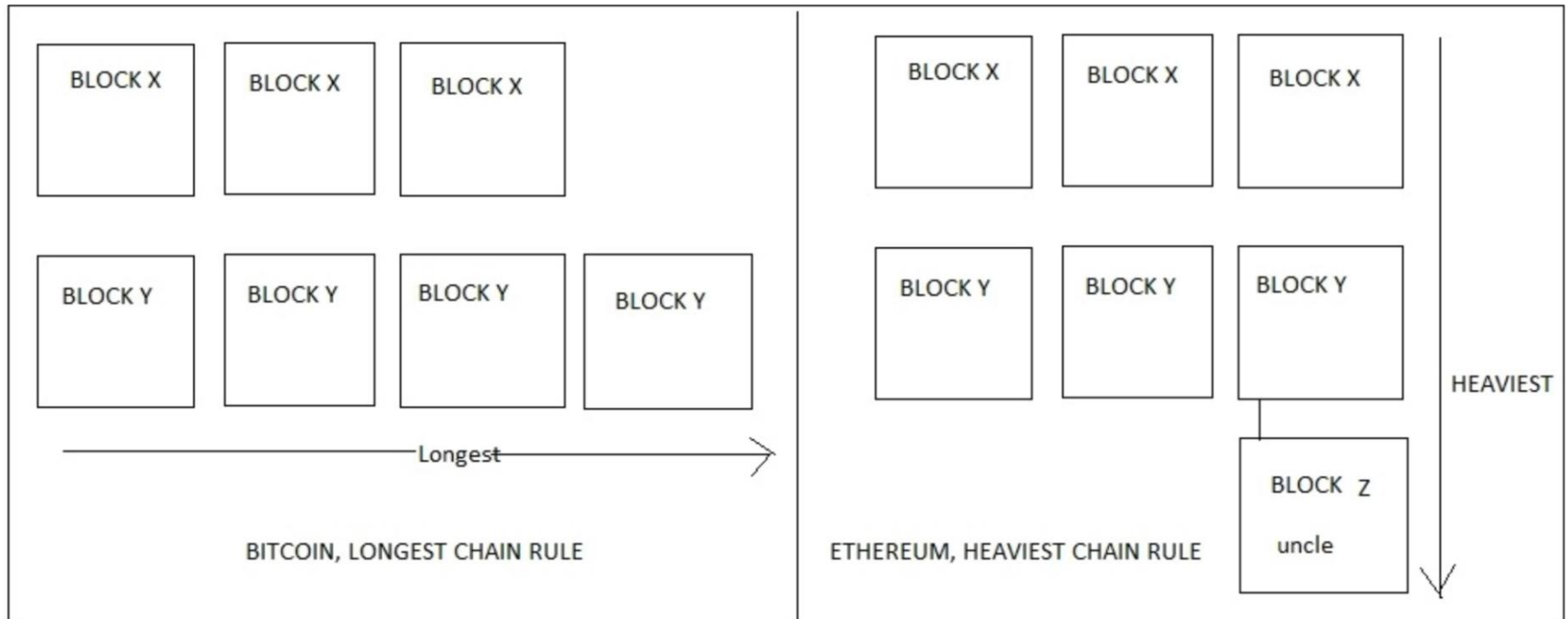
- o **Greedy Heaviest Observed Subtree (GHOST)** was first introduced as a mechanism to alleviate the issues arising out of fast block generation times that led to stale or orphan blocks. In GHOST, The stale blocks are added in calculations to figure out the longest and heaviest chain of blocks. Stale blocks are called Uncles in Ethereum.

Ethereum Blockchain Archtecture

Red indicates KECCAK256 Hash

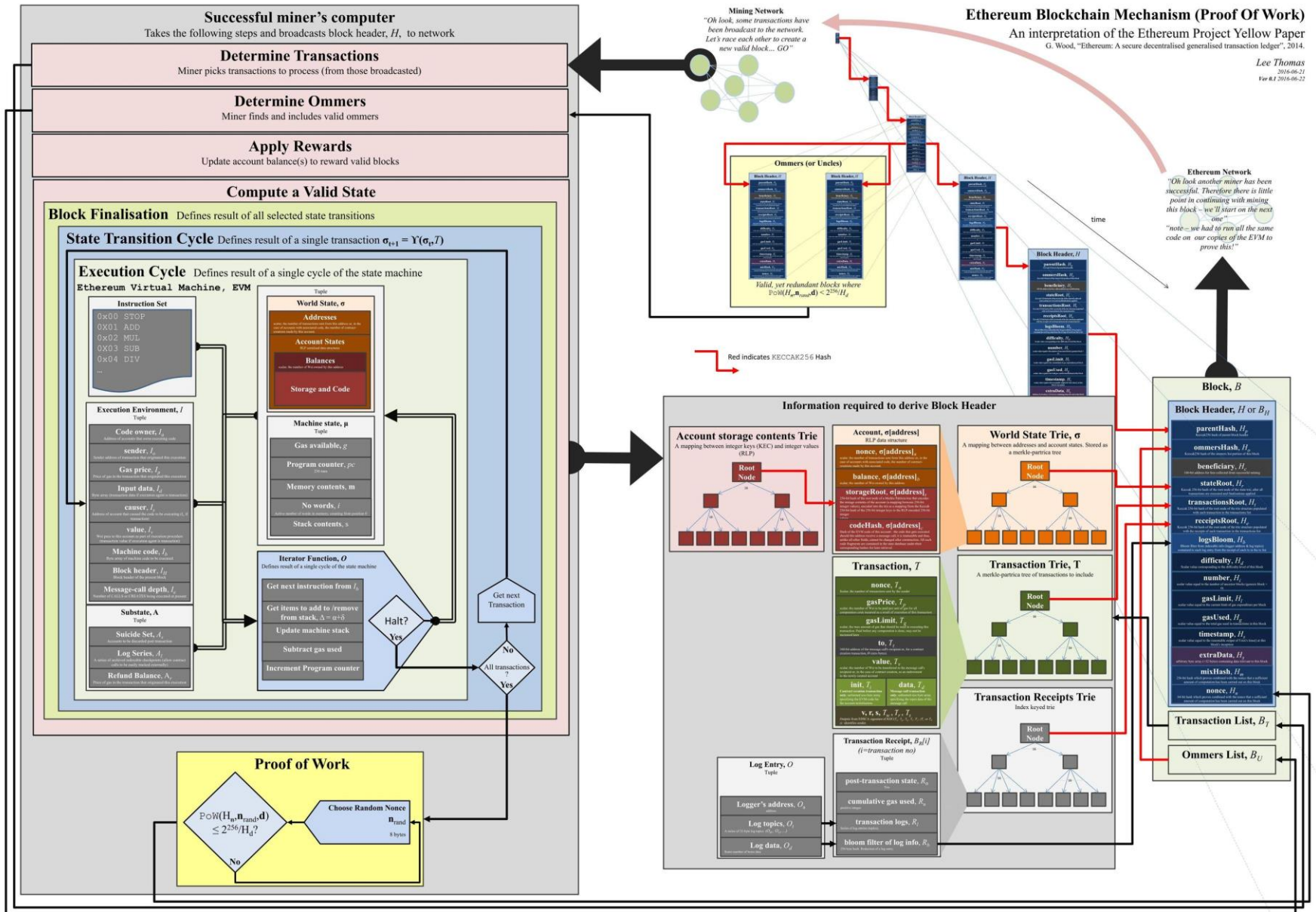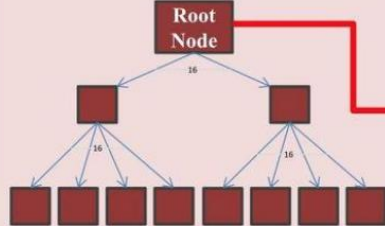## Information required to derive Block Header

Account storage contents Trie
A mapping between integer keys (KEC) and integer values (RLP)

Root Node

16

16   16

Account, σ[address]
RLP data structure

nonce, σ[address]$_n$

scalar; the number of transactions sent from this address or, in the case of accounts with associated code, the number of contract-creations made by this account.

balance, σ[address]$_b$

scalar; the number of Wei owned by this address

storageRoot, σ[address]$_s$

256-bit hash of the root node of a Merkle Patricia tree that encodes the storage contents of the account (a mapping between 256-bit integer values), encoded into the trie as a mapping from the Keccak 256-bit hash of the 256-bit integer keys to the RLP-encoded 256-bit integer values.

codeHash, σ[address]$_c$

Hash of the EVM code of this account - the code that gets executed should this address receive a message call; it is immutable and thus, unlike all other fields, cannot be changed after construction. All such code fragments are contained in the state database under their corresponding hashes for later retrieval.

World State Trie, σ
A mapping between addresses and account states. Stored as a merkle-patrica tree.

Root Node

16

16   16

Transaction, T

nonce, T_n

Scalar; the number of transactions sent by the sender

gasPrice, T_p

scalar; the number of Wei to be paid per unit of gas for all computation costs incurred as a result of execution of this transaction

gasLimit, T_g

scalar; the max amount of gas that should be used in executing this transaction. Paid before any computation is done, may not be increased later.

to, T_t

160-bit address of the message call's recipient or, for a contract creation transaction, Ø (zero bytes)

value, T_v

scalar; the number of Wei to be transferred to the message call's recipient or, in the case of contract creation, as an endowment to the newly created account

init, T_i

Contract creation transaction only; unlimited size byte array specifying the EVM-code for the account initialisation

data, T_d

Message call transaction only; unlimited size byte array specifying the input data of the message call

v, r, s, T_w, T_r, T_s

Outputs from EDSCA signature of KEC(T_n, T_p, T_g, T_t, T_v, (T_i or T_d)). It; identifies sender.

Transaction Trie, T
A merkle-partrica tree of transactions to include

Root Node

16

16   16

Transaction Receipts Trie
Index keyed trie

Root Node

16

16   16

Log Entry, O
Tuple

Logger's address, O_a

address

Log topics, O_t

A series of 32-byte log topics (O_{t0}, O_{t1},...)

Log data, O_d

Some number of bytes data

Transaction Receipt, $B_R[i]$
(i=transaction no)
Tuple

post-transaction state, R_σ

Trie

cumulative gas used, R_u

positive integer

transaction logs, R_l

Series of log entries (tuples),

bloom filter of log info, R_b
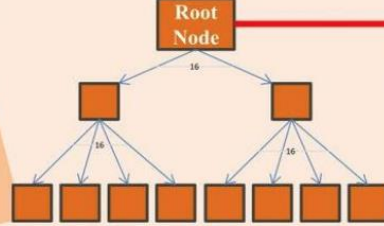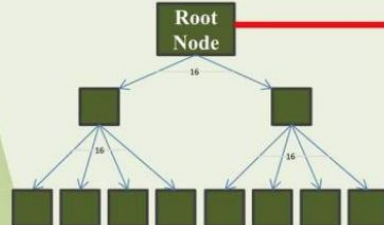
256 byte hash. Reduction of a log entry.

Block, B

Block Header, H or B_H

parentHash, H_p

Keccak256 hash of parent block header

ommersHash, H_o

Keccak256 hash of the ommers list portion of this block

beneficiary, H_c

160-bit address for fees collected from successful mining

stateRoot, H_r

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

transactionsRoot, H_t

Keccak 256-bit hash of the root node of the trie structure populated with each transaction in the transactions list

receiptsRoot, H_e

Keccak 256-bit hash of the root node of the trie structure populated with the receipts of each transaction in the transactions list

logsBloom, H_b

Bloom filter from indexable info (logger address & log topics) contained in each log entry from the receipt of each tx in the tx list

difficulty, H_d

Scalar value corresponding to the difficulty level of this block

number, H_i

scalar value equal to the number of ancestor blocks (genesis block = 0)

gasLimit, H_l

scalar value equal to the current limit of gas expenditure per block

gasUsed, H_g

scalar value equal to the total gas used in transactions in this block

timestamp, H_s

scalar value equal to the reasonable output of Unix's time() at this block's inception

extraData, H_x

arbitrary byte array (<32 bytes) containing data relevant to this block

mixHash, H_m

256-bit hash which proves combined with the nonce that a sufficient amount of computation has been carried out on this block

nonce, H_n

64-bit hash which proves combined with the nonce that a sufficient amount of computation has been carried out on this block

Transaction List, B_T

Ommers List, B_U

number, H_i

scalar value equal to the number of ancestor blocks (genesis block =

gasLimit, H_l

scalar value equal to the current limit of gas expenditure per block

gasUsed, H_g

scalar value equal to the total gas used in transactions in this block

timestamp, H_s

scalar value equal to the reasonable output of Unix's time() at this block's inception

extraData, H_x

arbitrary byte array (<32 bytes) containing data relevant to this block

Account State

The account state consists of four fields: nonce, balance, storageroot and codehash

Account, σ[address]
RLP data structure

nonce, σ[address]$_n$
scalar; the number of transactions sent from this address or, in the case of accounts with associated code, the number of contract-creations made by this account.

balance, σ[address]$_b$
scalar; the number of Wei owned by this address

storageRoot, σ[address]$_s$
256-bit hash of the root node of a Merkle Patricia tree that encodes the storage contents of the account (a mapping between 256-bit integer values), encoded into the trie as a mapping from the Keccak 256-bit hash of the 256-bit integer keys to the RLP-encoded 256-bit integer values

codeHash, σ[address]$_c$
Hash of the EVM code of this account - the code that gets executed should this address receive a message call; it is immutable and thus, unlike all other fields, cannot be changed after construction. All such code fragments are contained in the state database under their corresponding hashes for later retrieval.

o **Nonce,** This is a value that is incremented every time a transaction is sent from the address. In case of <u>contract accounts</u>, it represents the number of contracts created by the account.

o **Balance ,** This value represents the number of Weis which is the smallest unit of the currency (Ether) in Ethereum held by the address.

o **Storageroot ,**This field represents the root node of a Merkle Patricia tree that encodes the storage contents of the account.

o **Code Hash,** This is an immutable field that contains the hash of the smart contract code that is associated with the account.

Transaction

Transaction, T

nonce, T_n
Scalar; the number of transactions sent by the sender

gasPrice, T_p
scalar; the number of Wei to be paid per unit of gas for all computation costs incurred as a result of execution of this transaction

gasLimit, T_g
scalar; the max amount of gas that should be used in executing this transaction. Paid before any computation is done, may not be increased later.

to, T_t
160-bit address of the message call's recipient or, for a contract creation transaction, Ø (zero bytes).

value, T_v
scalar; the number of Wei to be transferred to the message call's recipient or, in the case of contract creation, as an endowment to the newly created account

init, T_i
Contract creation transaction only; unlimited size byte array specifying the EVM-code for the account initialisation

data, T_d
Message call transaction only; unlimited size byte array specifying the input data of the message call

v, r, s, T_w, T_r, T_s
Outputs from EDSCA signature of KEC(T_n, T_p, T_g, T_t, T_v, (T_i or T_d)); identifies sender.

○ **Nonce** is a number that is incremented by one every time a transaction is sent by the sender.

○ **gasPrice** represents the amount of Wei required in order to execute the transaction. gasLimit

○ **gasLimit** contains the value that represents the maximum amount of gas that can be consumed in order to execute the transaction. It is sufficient to say that this is the amount of fee in Ether that a user is willing to pay for computation.

○ **To** is a value that represents the address of the recipient of the transaction. Value

○ **value** represents the total number of Wei to be transferred to the recipient; in the case of a contract account, this represents the balance that the contract will hold.

○ **Init** is used only in transactions that are intended to create contracts. This represents a byte array of unlimited length that specifies the EVM code to be used in the account initialization process.

○ **Data** , If the transaction is a message call, then the data field is used instead of init, which represents the input data of the message call. It is also unlimited in size and is organized as a byte array.

○ **Sender Signature,** These values represent the digital signature (R, S) and some information that can be used to recover the public key (V).

Type of Transaction

TRANSACTION

CONTRACT CREATION

SENDER	TRANSACTOR
GAS	GAS PRICE
ENDOWMENT	BY TE ARRAY
EVM CODE	STACK DEPTH

MASSAGE CALL

SENDER	ORIGINATOR
RECIPIENT	ACCOUNT
GAS	VALUE
GAS PRICE	BY TE ARRAY
CALL DATA	STACK DEPTH

Contract Creation Transaction

- Sender
- Original transactor
- Available gas
- Gas price
- Endowment, which is the amount of ether allocated initially
- A byte array of arbitrary length
- Initialization EVM code
- Current depth of the message call/contract-creation stack (current depth means the number of items that are already there in the stack)

- The account is initialized when the EVM code (Initialization EVM code) is executed. In the case of any exception during code execution, such as not having enough gas, the state does not change.

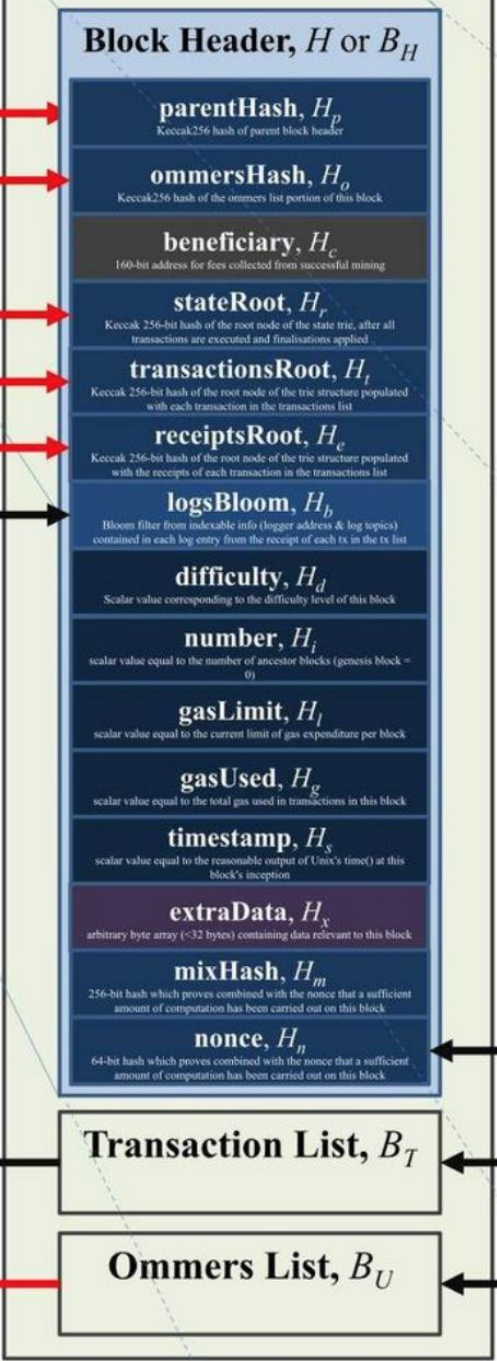| CONTRACT CREATION	
SENDER	TRANSACTOR
GAS	GAS PRICE
ENDOWMENT	BY TE ARRAY
EVM CODE	STACK DEPTH

Message Call Transaction

- A message call requires several parameters for execution, which are listed as follows:
  - Sender
  - The transaction originator
  - Recipient
  - The account whose code is to be executed
  - Available gas
  - Value
  - Gas price
  - Arbitrary length byte array
  - Input data of the call
  - Current depth of the message call/contract creation stack

- Message calls result in state transition. Message calls also produce output data, which is not used if transactions are executed. In cases where message calls are triggered by VM code, the output produced by the transaction execution is used.

MASSAGE CALL	
SENDER	ORIGINATOR
RECIPIENT	ACCOUNT
GAS	VALUE
GAS PRICE	BY TE ARRAY
CALL DATA	STACK DEPTH

Type of Account

- **Externally owned accounts (EOAs)** are similar to accounts that are controlled by a private key in bitcoin. An EOA has ether balance, is able to send transactions, and has no associated code

- **Contract Account (CA)** has ether balance, associated code, and the ability to get triggered and execute code in response to a transaction or a message.

Block Structure

Block, B

Block Header, H or B_H

- **parentHash, H_p**
  Keccak256 hash of parent block header
- **ommersHash, H_o**
  Keccak256 hash of the ommers list portion of this block
- **beneficiary, H_c**
  160-bit address for fees collected from successful mining
- **stateRoot, H_r**
  Keccak 256-bit hash of the root node of the state trie, after all transactions are executed and finalisations applied
- **transactionsRoot, H_t**
  Keccak 256-bit hash of the root node of the trie structure populated with each transaction in the transactions list
- **receiptsRoot, H_e**
  Keccak 256-bit hash of the root node of the trie structure populated with the receipts of each transaction in the transactions list
- **logsBloom, H_b**
  Bloom filter from indexable info (logger address & log topics) contained in each log entry from the receipt of each tx in the tx list
- **difficulty, H_d**
  Scalar value corresponding to the difficulty level of this block
- **number, H_i**
  scalar value equal to the number of ancestor blocks (genesis block = 0)
- **gasLimit, H_l**
  scalar value equal to the current limit of gas expenditure per block
- **gasUsed, H_g**
  scalar value equal to the total gas used in transactions in this block
- **timestamp, H_s**
  scalar value equal to the reasonable output of Unix's time() at this block's inception
- **extraData, H_x**
  arbitrary byte array (<32 bytes) containing data relevant to this block
- **mixHash, H_m**
  256-bit hash which proves combined with the nonce that a sufficient amount of computation has been carried out on this block
- **nonce, H_n**
  64-bit hash which proves combined with the nonce that a sufficient amount of computation has been carried out on this block

Transaction List, B_T

Ommers List, B_U

- **Parent hash,** is the Keccak 256-bit hash of the parent (previous) block's header.
- **Ommers hash,** is the Keccak 256-bit hash of the list of Ommers (Uncles) blocks included in the block.
- **Beneficiary** contains the 160-bit address of the recipient that will receive the mining reward once the block is successfully mined.
- **State root** contains the Keccak 256-bit hash of the root node of the state trie. It is calculated after all transactions have been processed and finalized.
- **Transaction root** is the Keccak 256-bit hash of the root node of the transaction trie. Transaction trie represents the list of transactions included in the block.
- **Receipts root** is the keccak 256 bit hash of the root node of the transaction receipt trie. This trie is composed of receipts of all transactions included in the block. Transaction receipts are generated after each transaction is processed and contain useful post-transaction information. Logs bloom
- **logs bloom** is a bloom filter that is composed of the logger address and log topics from the log entry of each transaction receipt of the included transaction list in the block.

Block Structure

- **Difficulty**, The difficulty level of the current block.
- **Number**, The total number of all previous blocks; the genesis block is block zero.
- **Gas limit**, The field contains the value that represents the limit set on the gas consumption per block.
- **Gas used**, The field contains the total gas consumed by the transactions included in the block
- **Timestamp** is the epoch Unix time of the time of block initialization.
- **Extra data** field can be used to store arbitrary data related to the block.
- **Mixhash** field contains a 256-bit hash that once combined with the nonce is used to prove that adequate computational effort has been spent in order to create this block.
- **Nonce** is a 64-bit hash (a number) that is used to prove, in combination with the mixhash field, that adequate computational effort has been spent in order to create this block.

Ether

Unit	Wei Value	Weis
Wei	1 Wei	1
Babbage	1e3 Wei	1,000
Lovelace	1e6 Wei	1,000,000
Shannon	1e9 Wei	1,000,000,000
Szabo	1e12 Wei	1,000,000,000,000
Finney	1e15 Wei	1,000,000,000,000,000
Ether	1e18 Wei	1,000,000,000,000,000,000

Gas Cost

Operation Name	Gas Cost
step	1
stop	0
suicide	0
sha3	30
sload	20
txdata	5
transaction	500
contract creation	53000

- **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.
- A **transaction fee** is charged as some amount of Ether and is taken from the account balance of the transaction originator.
- 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.
- Transaction cost can be estimated using the following formula:
- Total cost = gas Used * **gasPrice**