# DAYANANDA SAGAR COLLEGE OF ENGINEERING

## DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

DSCE

DEPARTMENT OF INFORMATION SCIENCE

# BLOCKCHAIN
# (19IS7DEBLC )

**Faculty in charge:**

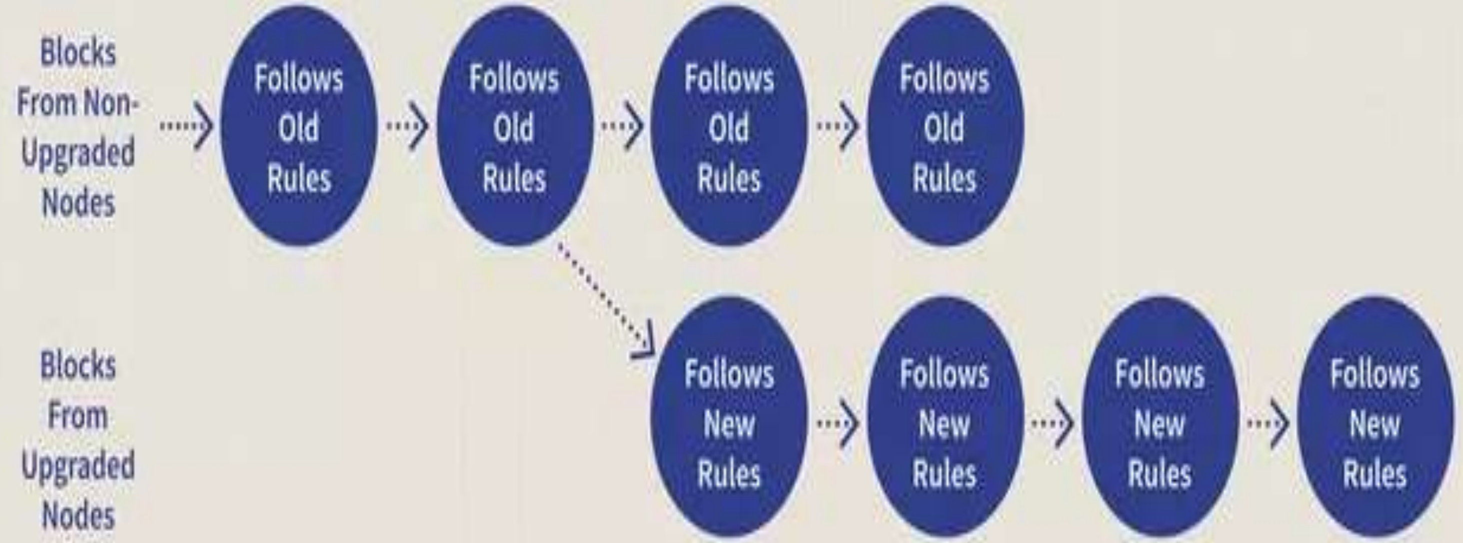
**Bindu Bhargavi S M**
**Asst. Professor, Dept. of ISE, DSCE**

# Module 4

**Ethereum 101:** Introduction, Ethereum bird’s eyeview, The Ethereum network, Components of the Ethereum ecosystem, Ether Cryptocurrency, The Ethereum VM (EVM), Smart Contracts

# Introduction

- Ethereum is the community-run technology powering the cryptocurrency ether (ETH) and thousands of decentralized applications.
- Termed as World’s Programmable Blockchain
- Ethereum is a decentralized, open source blockchain with smart contract functionality.
- Ether is the cryptocurrency used – Vitalik Buterin, 2013
- The platform allows anyone to deploy permanent and immutable decentralized applications onto it, with which users can interact
- A hard fork (or hardfork), as it relates to blockchain technology, is a radical change to a network's protocol that makes previously invalid blocks and transactions valid, or vice-versa. A hard fork requires all nodes or users to upgrade to the latest version of the protocol software



A Hard Fork: Non-Upgraded Nodes Reject The New Rules, Diverging The Chain

Investopedia

- Ethereum is permissionless, non hierarchical network of computers
- Ethereum is a decentralized public ledger for verifying and recording transactions
- The network's users can create, publish, monetize, and use applications on the platform, and use its Ether cryptocurrency as payment – dApps
- Fees – gas
- Ex: Overstock, Shopify, and CheapAir are among the online sites that accept Ether as payment
- Ethereum Business - Bitcoin, Ripple, IBM, IOTA, Microsoft, Blockstream, JP Morgan, and NEO
- Market value of one ETH is $2,236

# How does it Work

- Database of information that is designed to be unhackable
- Ether is the cryptocurrency used
- Every transaction happens using an Ether coin – which is further verified and recorded as an additional block on the coin’s unique blockchain
- ETH trading – trading platforms include Coinbase, Kraken, Bitstamp, Gemini, Binance, and Bitfinex

# Ethereum v/s Bitcoin

- Ether cryptocurrency was created to provide an in-house currency for applications built on the Ethereum blockchain
- Each is a digital currency traded via online exchanges and stored in various types of cryptocurrency wallets
- Both of these tokens are decentralized, meaning that they are not issued or regulated by a central bank or other authority
- Both make use of the distributed ledger technology known as blockchain
- Transactions on the Ethereum network may contain executable code, while data affixed to Bitcoin network transactions are generally only for keeping notes
- Block time (an ether transaction is confirmed in seconds compared to minutes for bitcoin) and the algorithms that they run on (Ethereum uses ethash while Bitcoin uses SHA-256)

- The length of time it takes to mine Ethereum and receive Ether mining rewards depends on the hashrate, power consumption, cost of electricity, and any fees paid to a mining pool and/or hosting service related to the mining operation.
- These factors also directly impact profitability and increases in mining difficulty targets and the overall price performance of the crypto market. Using the default calculations of this popular Ethereum mining calculator, it is estimated that mining one ETH will take 51.8 days

# More About Ethereum

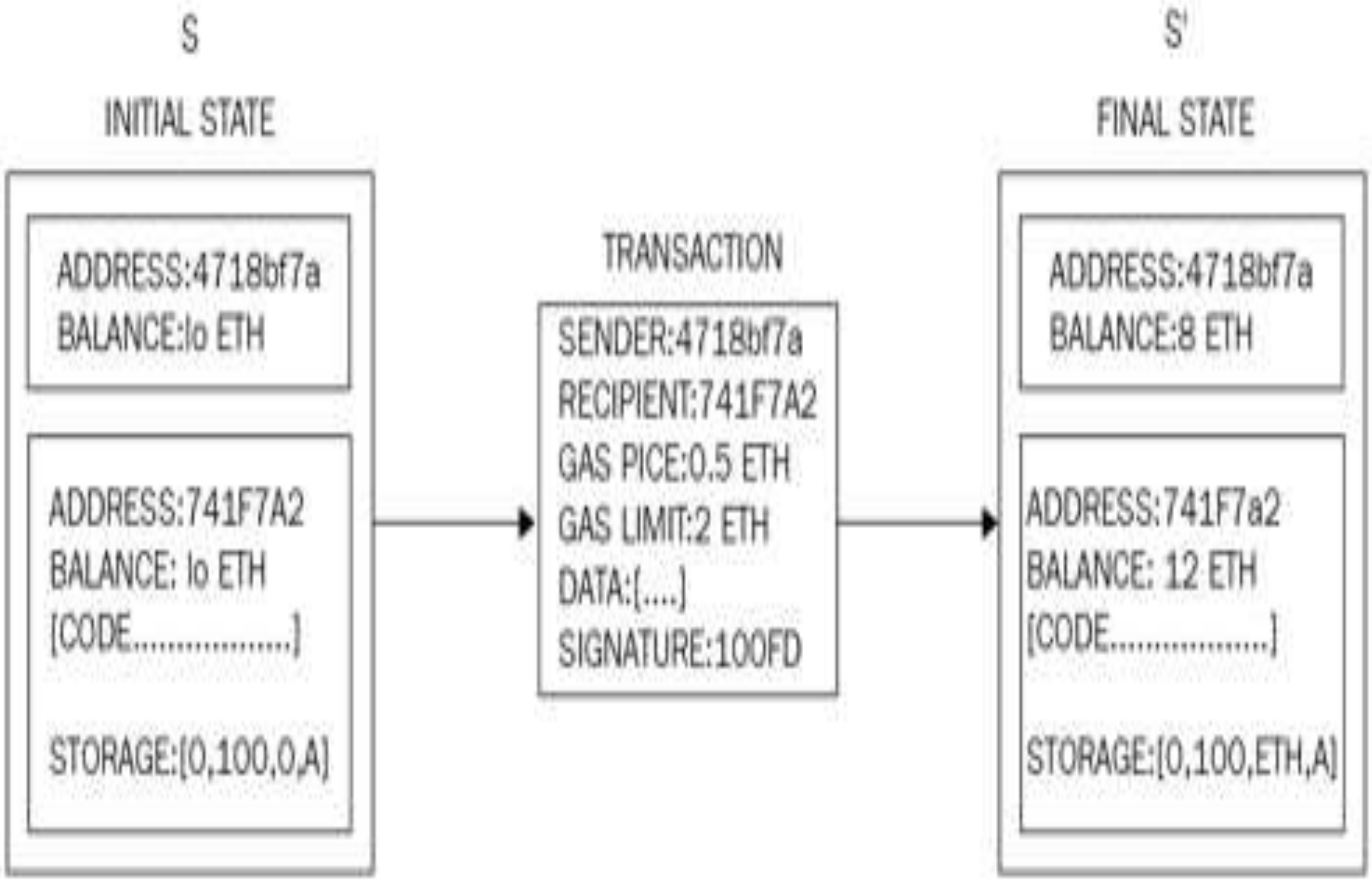
- **Version Release date**
- Olympic - May, 2015
- Frontier - July 30, 2015
- Homestead - March 14, 2016
- Byzantium - (first phase of Metropolis) October 16, 2017
- Constantinople - 2018
- Metropolis - To be released
- Serenity - (final version of Ethereum) To be released

- Establishment of Ethereum foundation – to raise funds to build a blockchain with built in programming language
- Ethereum Frontier – more technical
- Homestead – user friendly, dedicated development community
- Metropolis – Applications fully developed and tested, easy to use applications
- Serenity – moving from proof of work consensus to proof of stake model

- Turing complete programming language - a full-functioning programming language that allows developers to build any type of application
- User friendly interfaces
- Rapid development time, security for small applications and ability of the applications to interact with one another
- Applications – managing digital assets, recording ownership of assets such as land, and decentralized autonomous organizations (DAOs)

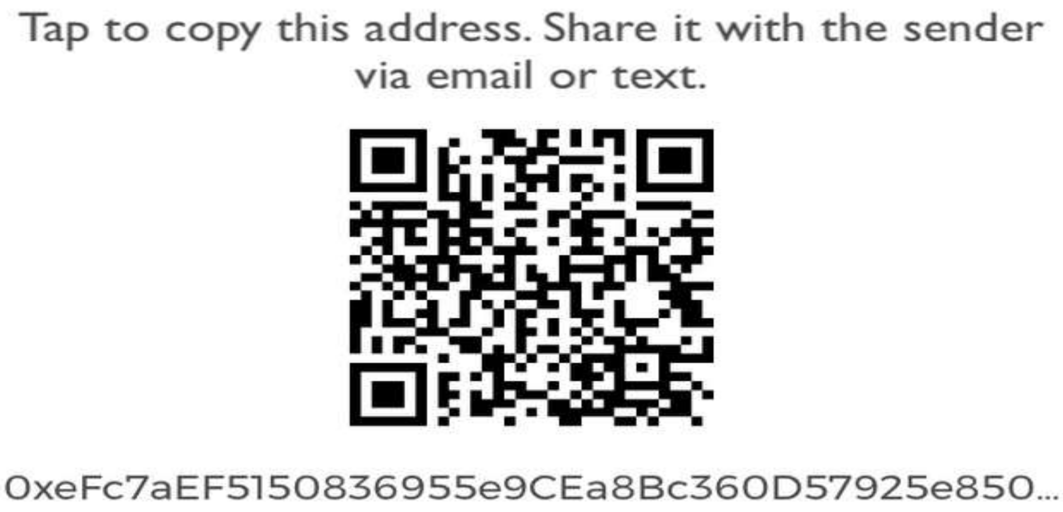
# Ethereum Blockchain

- Genesis state is transformed into a final state by the incremental execution of transactions
- The final transformation is then accepted as the final state of the system
- Each state is stored on the Ethereum network as the world state


S
INITIAL STATE
ADDRESS:4718bf7a
BALANCE:lo ETH
ADDRESS:741F7A2
BALANCE: lo ETH
[CODE.....................]
STORAGE:[0,100,0,A]
TRANSACTION
SENDER:4718bf7a
RECIPIENT:741F7A2
GAS PICE:0.5 ETH
GAS LIMIT:2 ETH
DATA:[....]
SIGNATURE:100FD
S'
FINAL STATE
ADDRESS:4718bf7a
BALANCE:8 ETH
ADDRESS:741F7a2
BALANCE: 12 ETH
[CODE.....................]
STORAGE:[0,100,ETH,A]

# Working of Ethereum

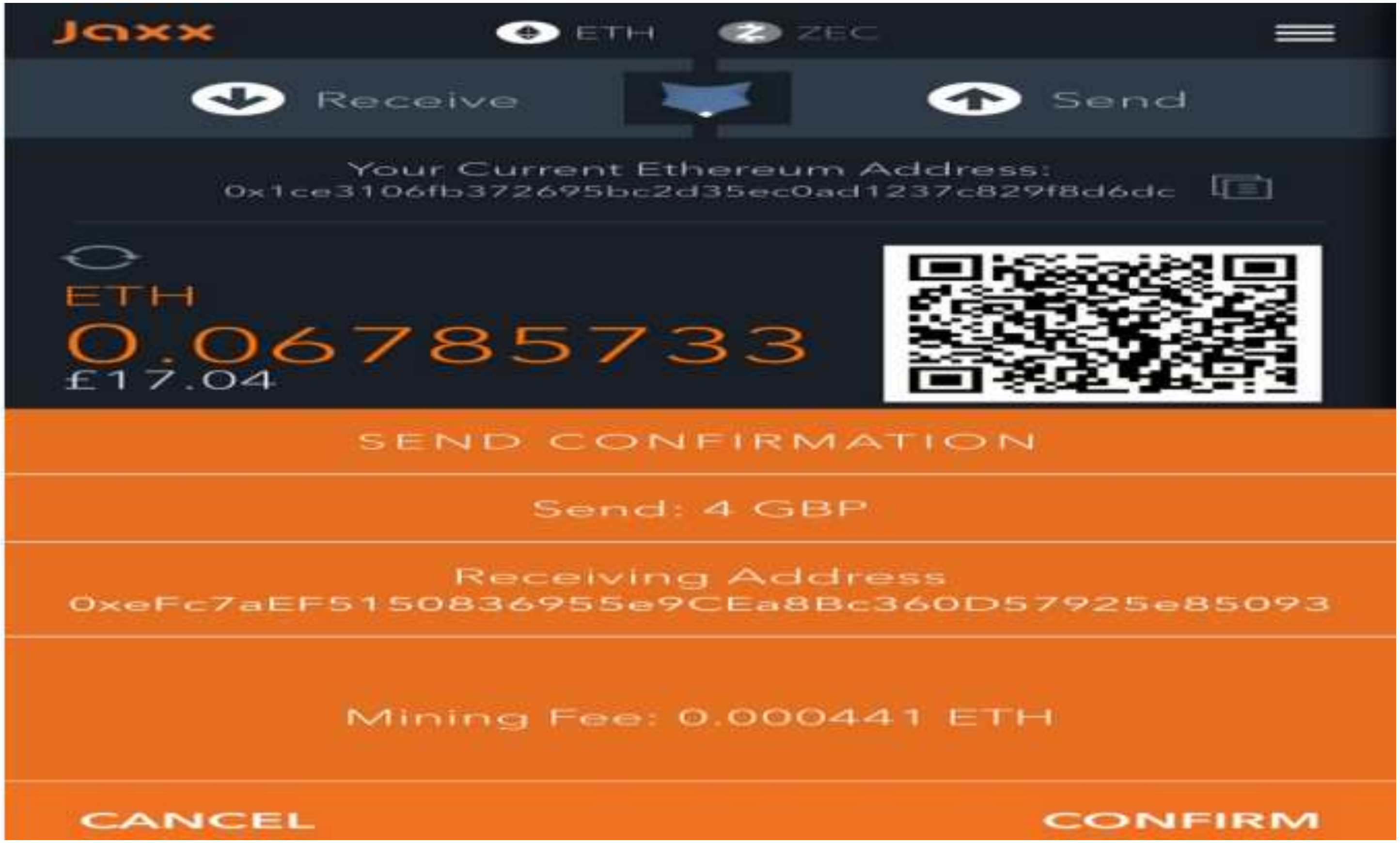
- User requests money by sending a request to the sender or sender decides to send the money to the receiver
- request can be sent by sending the receivers Ethereum address to the sender – scanning the QR code/manually typing the Ethereum address
- QR code transmission  via email, text other communication techniques



Tap to copy this address. Share it with the sender via email or text.

0xeFc7aEF5150836955e9CEa8Bc360D57925e850...

- Once the request is received, either scan the QR code/ copy the Ethereum address into the Ethereum Wallet software to initiate a transaction



Your Current Ethereum Address:
0x1ce3106fb372695bc2d35ec0ad1237c829f8d6dc

ETH
0.06785733
£17.04

SEND CONFIRMATION

Send: 4 GBP

Receiving Address
0xeFc7aEF5150836955e9CEa8Bc360D57925e85093

Mining Fee: 0.000441 ETH

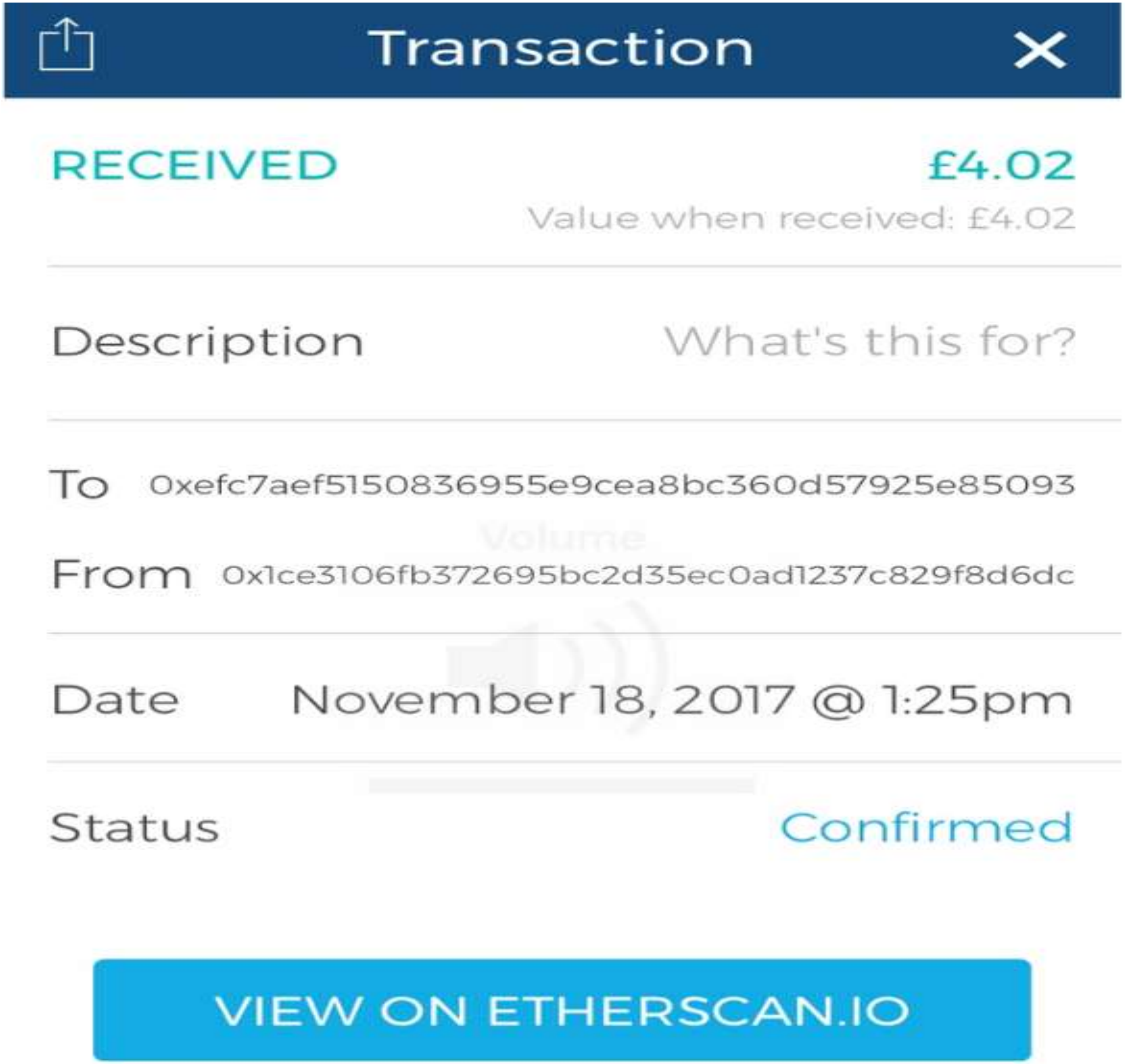
- Once the request (transaction) of sending money is constructed in the wallet software, it is then broadcasted to the Ethereum network. The transaction is digitally signed by the sender as proof that he is the owner of the Ether.

- This transaction is then picked up by nodes called miners on the Ethereum network for verification and inclusion in the block. At this stage, the transaction is still unconfirmed.

- Once it is verified and included in the block, the PoW process starts.
-
- Once a miner finds the answer to the PoW problem, by repeatedly hashing the block with a new nonce, this block is immediately broadcasted to the rest of the nodes which then verifies the block and PoW.

- If all the checks pass then this block is added to the blockchain, and
- miners are paid rewards accordingly.

- Finally, receiver gets the Ether, and it is shown in the respective wallet software.



RECEIVED £4.02

Value when received: £4.02

Description What's this for?

To 0xefc7aef5150836955e9cea8bc360d57925e85093

From 0x1ce3106fb372695bc2d35ec0ad1237c829f8d6dc

Date November 18, 2017 @ 1:25pm

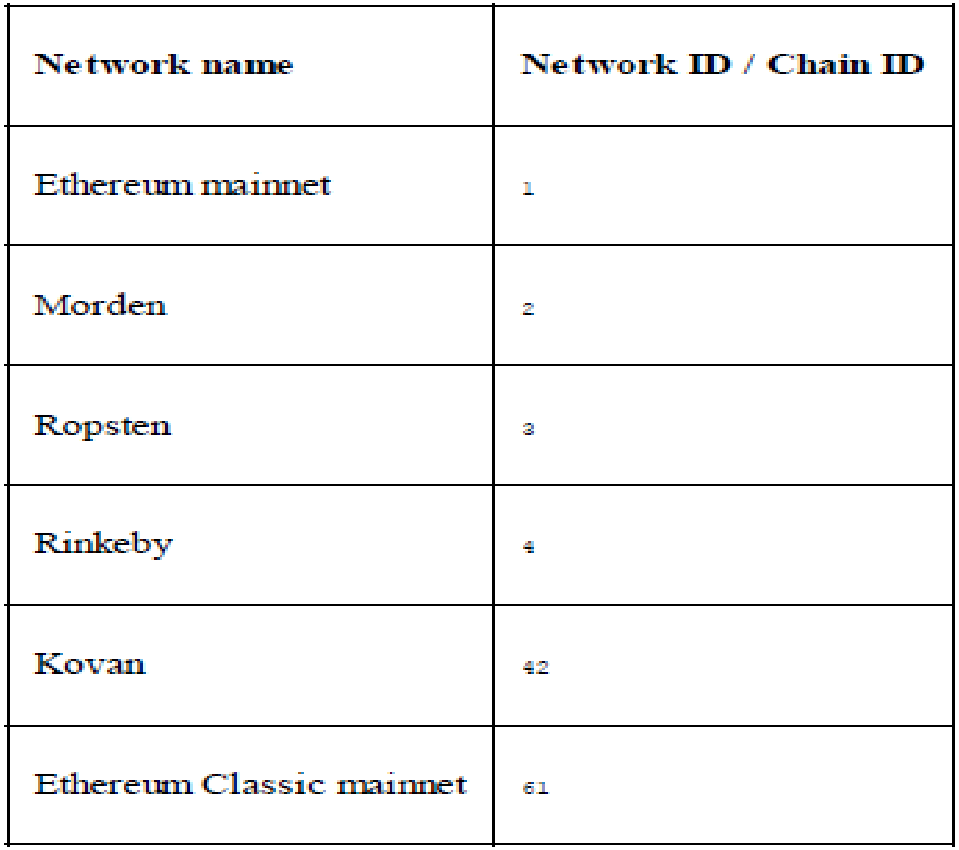
Status Confirmed

# The Ethereum Network

- Peer to Peer Network where the nodes participate and contribute to the consensus mechanism
- Three types of Ethereum network are:
  - Mainnet
  - Testnet
  - Private net

- Mainnet
  - Current live network of Ethereum
  - Current version is Byzantium (Metropolis) with Chain ID as 1 – used to identify the network
- Testnet
  - Also called Ropsten
  - Used to test the smart contracts and Dapps before their deployment
  - Allows for experimentation and research
  - Kovan and Rinkeby – developed for testing Byzantium releases

- Private net
  - Private network created by generating a new genesis block
  - Ex: private group of entities starting a blockchain and using it as a permissioned blockchain



| Network name | Network ID / Chain ID |
|---|---|
| Ethereum mainnet | 1 |
| Morden | 2 |
| Ropsten | 3 |
| Rinkeby | 4 |
| Kovan | 42 |
| Ethereum Classic mainnet | 61 |

# Working of DAO’s

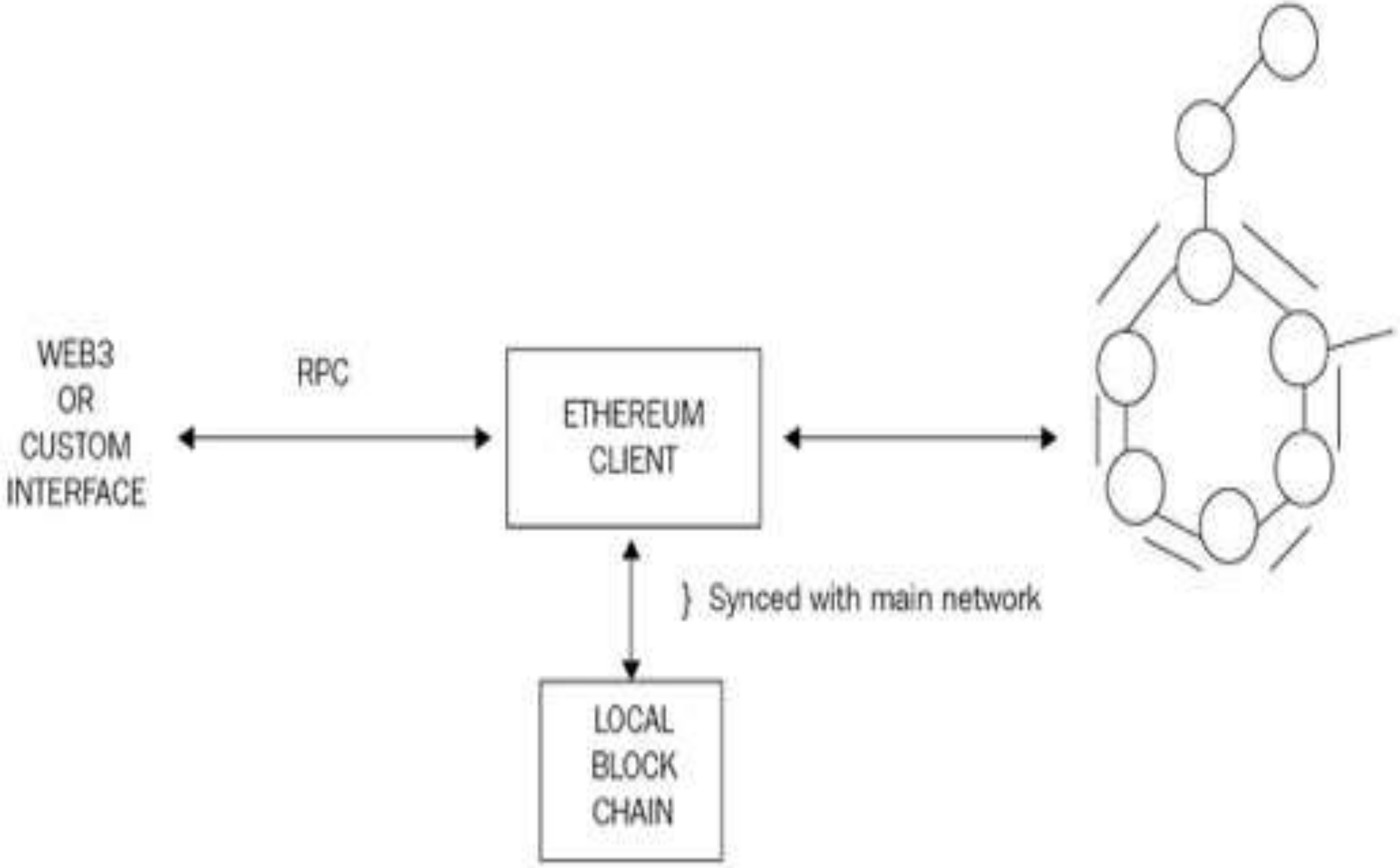
- A group of people writes a smart contract to govern the organization.
- People add funds to the DAO and are given tokens that represent Ownership - members have control of the funds from day one.
- When the funds have been raised, the DAO begins to operate by having members propose how to spend the money.
- The members vote on these proposals.
- When the predetermined time has passed and the predetermined number of votes has accrued, the proposal passes or fails.
- Individuals act as contractors to service the DAO.

# Hacking a blockchain

- Ethereum has never been hacked
- Hack of the DAO - An unexpected code path in The DAO’s contract allowed any sophisticated user to withdraw funds
- Decentralization, immutability, and autonomy meant no central authority could decide what to do quickly
- Use of Hard Fork – two Ethereums: Ethereum and Ethereum classic

# Components of Ethereum Ecosystem

- Ethereum blockchain
  - Running on the peer-to-peer Ethereum network
- Ethereum clients
  - Runs on the nodes and connects to the peer to peer network
  - Blockchain in downloaded and stored locally
- Ethereum network
  - Mining and account management
- Web3.js library
  - allow you to interact with a local or remote ethereum node using HTTP, IPC or WebSocket


WEB3
OR
CUSTOM
INTERFACE
RPC
ETHEREUM
CLIENT
} Synced with main network
LOCAL
BLOCK
CHAIN

# Remote Procedure Call

- **JSON-RPC** is simply a remote procedure call protocol that is used on Ethereum to define different data structures
- It also defines the rules on how data structures are processed in the network
- JSON is a stateless, lightweight data-interchange format that can represent numbers, ordered sequences of values, and collections of value pairs.
- Use of a namespace System
  - Contains **fundamental classes and base classes** that define commonly-used value and reference data types, events and event handlers, interfaces, attributes, and processing exceptions
- Ex: eth_call method
- A method is a function required to execute a Ethereum node.

# Methods and Categories

- Block data  related
- Per-User data
- Reading existing smart contract data

- Block Data Related
  - retrieve static data about a particular block in the blockchain where data is contained in the blockchain headers and include information such as *blocknumber* or transactions mined
  - eth_getBlockByHash: This method retrieves a block’s details by its hash.
  - eth_blockNumber: This retrieves the current block number.
  - eth_getBlockByNumber: It retrieves a block’s details by number.

- **Per-User (address) Data**
  - retrieves the simple state data for addresses regardless of whether that address is a user or a smart contract
  - eth_getTransactionCount: This method retrieves the nonce of an address, i.e., how many transactions have already been mined, per user, for that address.

- **Reading Existing Smart Contract Data**
  - methods used for interacting with smart contracts
  - eth_getStorageAt: retrieves raw state storage for a smart contract.
  - eth_call: retrieves a constant EVM method on a Smart Contract and is the primary way to retrieve already mined data from the blockchain about a particular smart contract.
  - Other methods such as eth_getLogs, eth_getCode can also be used to interact with smart contracts.

# Keys and Addresses

- Used to represent the ownership and for the transfer of Ether
- Used in pairs of private and public key
- Random generation of private key and public key is derived from the private key
- Addresses are derived from public keys – 20 bytes code that is used to identify the account

# Process of key generation and address derivation

- First, a private key is randomly chosen (256 bits positive integer) under the rules defined by elliptic curve secp256k1 specification (in the range [1, secp256k1n − 1]).
- The public key is then derived from this private key using ECDSA recovery function.
- An address is derived from the public key which is the right most 160 bits of the Keccak hash of the public key.
  - versatile cryptographic function
  - Used for authentication, encryption and pseudo random number generation
  - Creating a **deterministic** unique ID from a input.

# Ethereum State Transition function

- Confirm the transaction validity by checking the syntax, signature validity, and nonce
- Calculation of the transaction fee, and the sending address is resolved using the signature.
  - sender's account balance is checked and subtracted and nonce is incremented.
  - An error is returned if the account balance is not enough
- Provide enough Ether to cover the cost of the transaction.
  - charged per byte incrementally proportional to the size of the transaction
  - The flow is from the sender's account to receiver's account.
- In cases of transaction failure due to insufficient account balance or gas, all state changes are rolled back except for fee payment, which is paid to the miners.
- Finally, the remainder (if any) of the fee is sent back to the sender as change and fee are paid to the miners accordingly. At this point, the function returns the resulting state which is also stored on the blockchain.

# Accounts

- Building blocks of Ethereum
- Transaction between the accounts is represented as creation or updation of states
- Types:
  - **Externally Owned Accounts (EOAs)**
    - **Accounts controlled by a private key in Bitcoin**
  - **Contract Accounts (CAs)**
    - **Have codes associated along with the private key of the Bitcoin**

- EOs:
  - EOAs has ether balance
  - They are capable of sending transactions
  - They have no associated code
  - They are controlled by private keys
  - Accounts contain a key-value store
  - They are associated with a human user

- CAs:
  - CAs have Ether balance.
  - Associated code placed in the memory/storage on the blockchain
  - They can get triggered and execute code in response to a transaction or a message from other contracts
  - Code is executed by EVM by each mining node on the network
  - Not intrinsically associated with any user or actor on the blockchain
  - Accounts contain key-value store

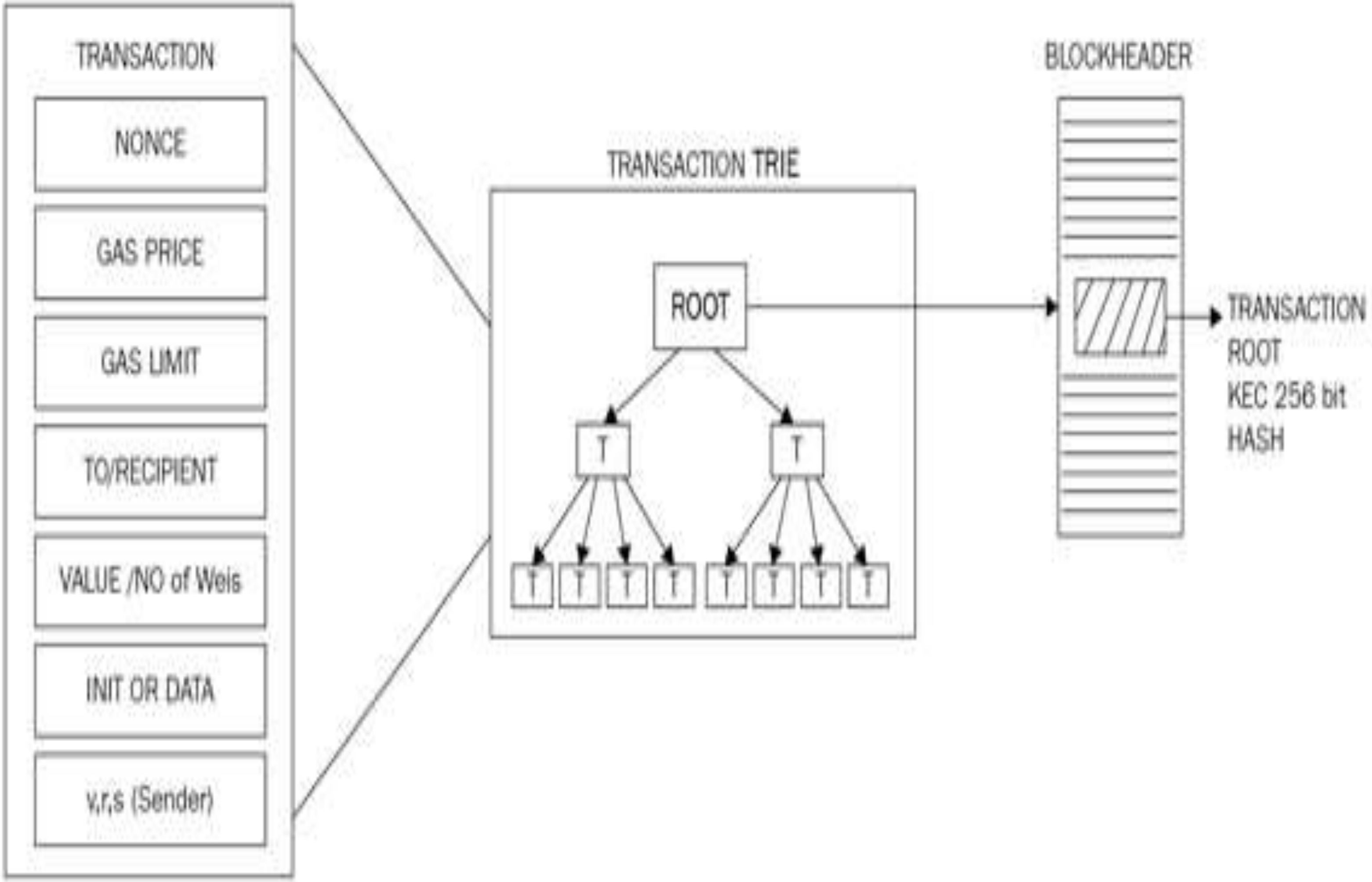
# Transactions in Ethereum

- A transaction is the act of transferring Ethereum-based assets from one address to another that is initiated from your wallet
- A transaction is a cryptographically-signed message that specifies what change is to be made, and it is sent to any node in the network
- A transaction in Ethereum is a digitally signed data packet using a private key that contains the instructions that, when completed, either result in a message call or contract creation
- Types of transaction:
  - Message call transaction
  - Contract creation transaction

- Message call transactions
  - Production of a message call used to pass the messages from
- Contract creation transactions
  - transactions result in the creation of a new contract account.

- Common fields considered for both the type of transactions are:
  - Nonce
  - Gas price
  - Gas limit
  - To
  - Value
  - Signature
  - Init
  - Data

- Nonce
  - incremented by one every time a transaction is sent by the sender
- Gas Price
  - The gas price field represents the amount of Wei required to execute the transaction
- Gas limit
  - The gas limit field contains the value that represents the maximum amount of gas that can be consumed to execute the transaction
- To
  - address of the recipient of the transaction
- Value
  - Value represents the total number of Wei to be transferred to the recipient
- Signature
  - Three fields v, r and s
- Init
  - The Init field is used only in transactions that are intended to create contracts
- 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


TRANSACTION
NONCE
GAS PRICE
GAS LIMIT
TO/RECIPIENT
VALUE /NO of Weis
INIT OR DATA
v,r,s (Sender)
TRANSACTION TRIE
ROOT
T
T
T
T
T
T
T
T
T
T
BLOCKHEADER
TRANSACTION
ROOT
KEC 256 bit
HASH

# Contract Creation Transaction

- Sender
- Original transactor (transaction originator)
- Available gas
- Gas price
- Endowment, which is the amount of ether allocated
- A byte array of an arbitrary length
- Initialization EVM code
- Current depth of the message call/contract-creation stack

- Addresses generated as a result of contract creation transaction are 160 bit in length
- Nonce in the account is initially set to 0
- The balance of the account is set to the value passed to the contract.
- Storage is also set to empty.
- The new account is initialized when the EVM is executed
- If the execution is successful, then the account is created after the payment of appropriate gas costs
- In case of exception the state does not change.

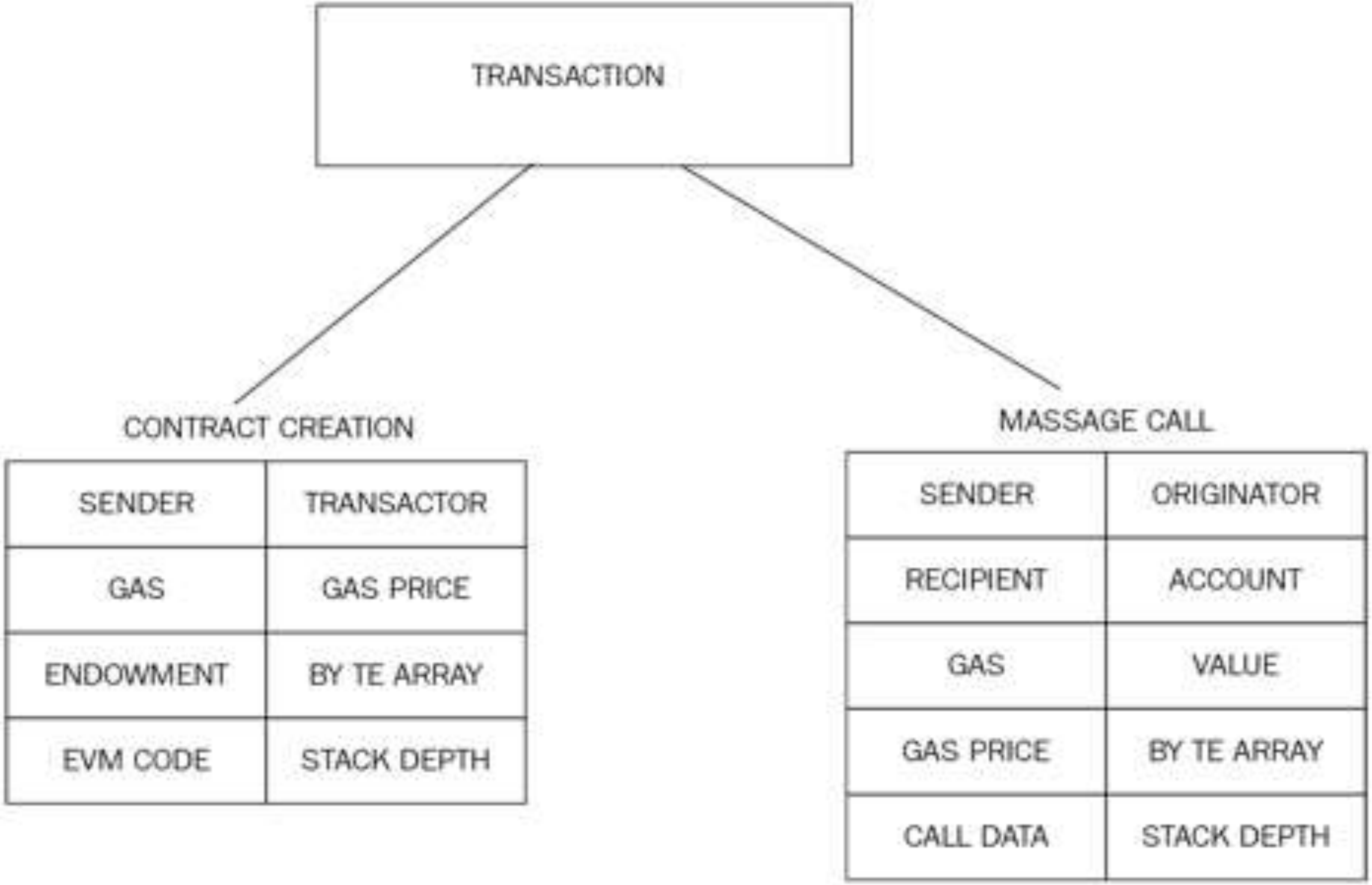
# Message Call Transaction

- The sender
- The transaction originator
- Recipient
- The account whose code is to be executed (usually same as the recipient)
- 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 a state transition.
- Message calls also produce the output data which is not used if the transactions are executed.
- When message calls are triggered by VM code, the output produced by the transaction execution is used.
- If the destination account has an associated EVM code, then the virtual machine will start, upon the receipt of the message to perform the required operations.
- If the message sender is an autonomous object (external actor), then the call passes back any data returned from the EVM operation.
- The state is altered by transactions. These are created by external factors and are signed and then broadcasted to the Ethereum network.
- Messages are passed using message calls.

# Messages

- A **message is a data packet passed between** two accounts – data and value
- Messages can be sent either through a smart contract or from an external actor in the form of transaction which is digitally signed by the sender
- Messages only exist in the execution environment and are never stored
- Messages are similar to transactions but messages are produced by contracts while transactions are produced by the entities of the Ethereum environment
- Message consists of:
  - Sender
  - Recipient
  - Amount of Wei
  - Optional data field
  - Maximum amount of gas that can be consumed
- Messages are generated when CALL or DELEGATECALL opcodes are executed by the code in execution by the contracts



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 |

# Calls

- Runs locally on the node
- Similar to a local function call
- Performs read-only operation
- Does not result in any state change
- Locally executed on the VM of a node

# Transaction validation and execution

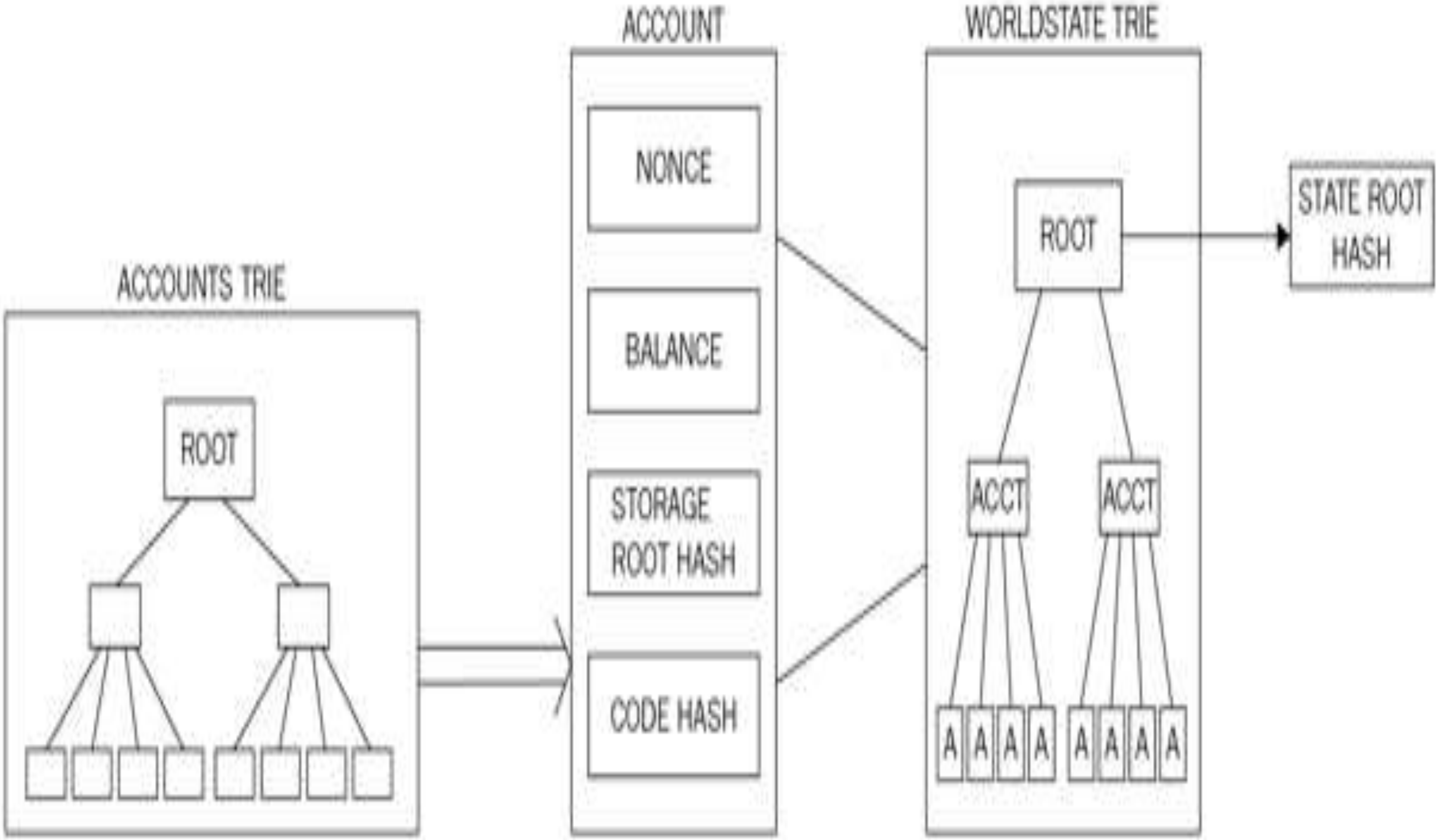
- Transactions are executed after verifying the transactions for validity.
  - transaction must be well-formed and RLP-encoded without any additional trailing bytes
  - The digital signature used to sign the transaction is valid
  - Transaction nonce must be equal to the sender's account's current nonce gas limit must not be less than the gas used by the transaction
  - The sender's account contains enough balance to cover the execution cost.

- A transaction substate is created during the execution of the transaction that is processed immediately after the execution completes:
  - Suicide set or self-destruct set – list of accounts that are disposed of after transaction execution
  - Log series – indexed series of checkpoints
  - Refund balance – total price of gas that initiated the execution
  - Touched accounts – empty accounts are deleted at the end of the transaction

- **State storage in the Ethereum blockchain**
  - transaction and consensus driven state machine
  - state needs to be stored permanently in the blockchain
  - World state, transactions and transaction receipts are stored on the blockchain in blocks

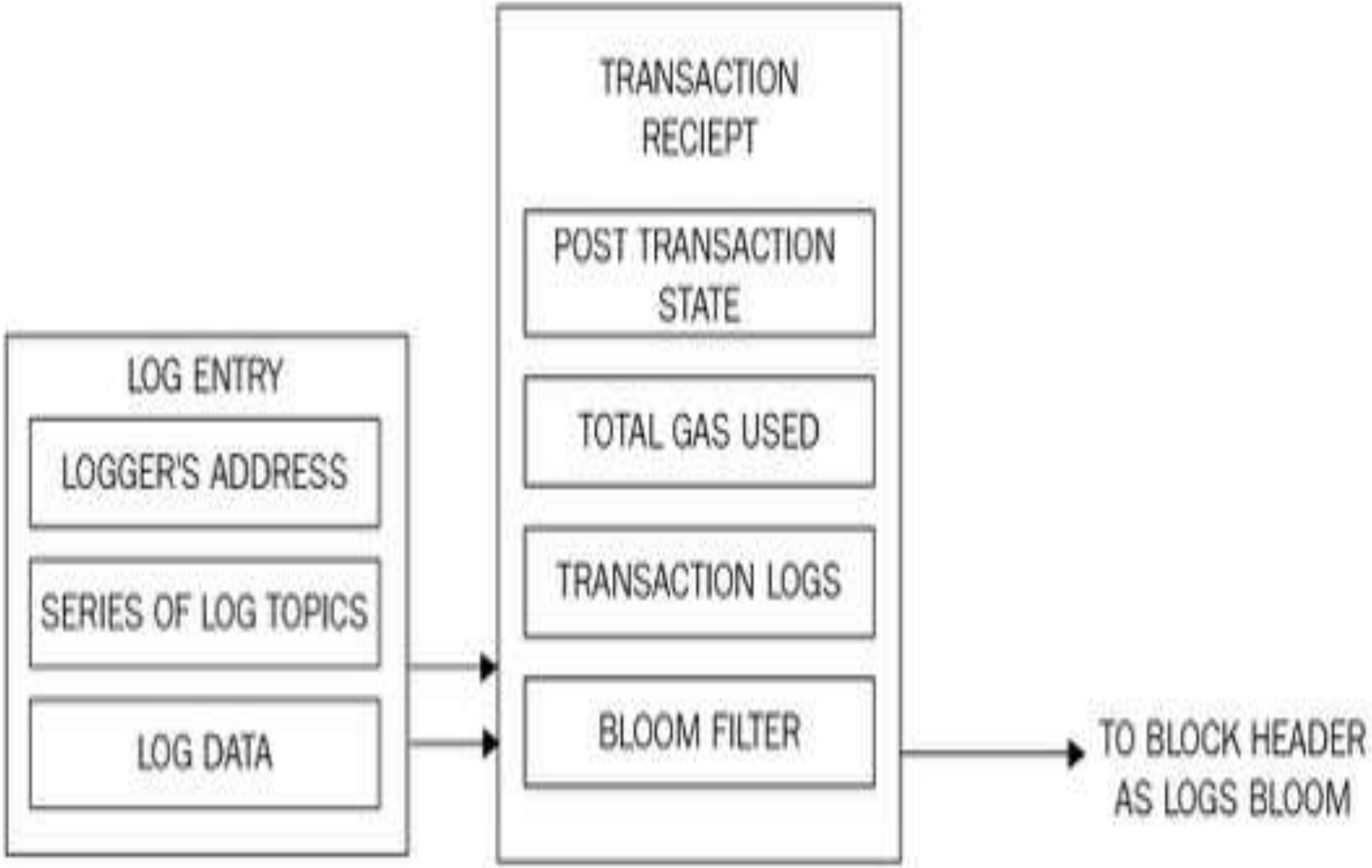
- World state –
  - Mapping between Ethereum address and account states
  - 20 byte addresses are mapped into a data structure that is serialized using Recursive Length Prefix
  - Serializes binary data – Patricia Tree data structure

- Account state:
  - Four fields: Nonce, Balance, Storage root, Code hash
  - Account data structure contains a storage root hash that is derived from the root node of account storage trie
  - Account data structure is used in world state trie which gives the mapping between addresses and account states
  - Accounts trie is a Merkle Patricia tree used to encode the storage contents of an account - mapping between Keccak 256-bit hashes of 256-bit integer keys to the RLP-encoded 256-bit integer values
  - Root node of the worldstate trie is hashed using Keccak 256-bit algorithm there by making it a part of block header data structure


ACCOUNTS TRIE
ROOT
ACCOUNT
NONCE
BALANCE
STORAGE ROOT HASH
CODE HASH
WORLDSTATE TRIE
ROOT
ACCT
ACCT
A
A
A
A
A
A
A
A
STATE ROOT HASH

# Transaction Receipts

- Mechanism to store the state after the transaction execution – enables in recording the outcomes of the transaction
- Receipts are stored as index-keyed trie
- The hash (Keccak 256-bit) of the root of this trie is placed in the block header as the receipts root
- Four elements:
  - Post transaction state
  - Gas used
  - Set of logs – logger's address, series of log topics and log data
  - Bloom filter - **used to test whether an element is a member of a set,** used in **Ethereum** to minimize the number of block queries that clients need to make.


TRANSACTION RECIEPT
POST TRANSACTION STATE
TOTAL GAS USED
TRANSACTION LOGS
BLOOM FILTER
LOG ENTRY
LOGGER'S ADDRESS
SERIES OF LOG TOPICS
LOG DATA
TO BLOCK HEADER AS LOGS BLOOM

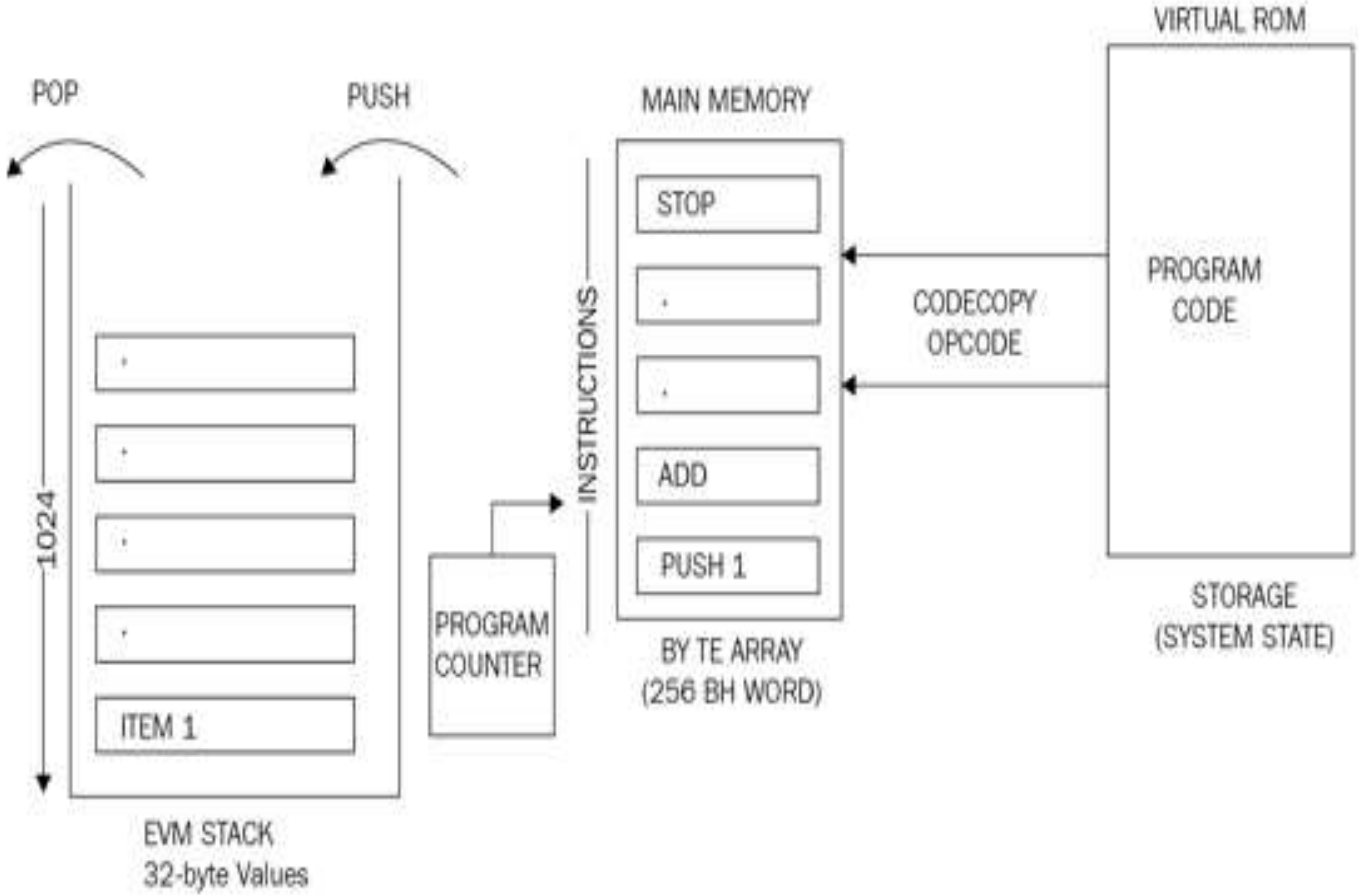
# Ether cryptocurrency/tokens (ETC & ETH)

- Ethereum classic – ETC, hard forked version as ETH
- Ether is used within the Ethereum blockchain to pay for the execution of contracts on the EVM
- Ether is used to purchase gas as *crypto fuel, which is required to perform computation on the Ethereum* blockchain.
- Units: Wei, Kwei, Mwei, Gwei, Micro Ether, Milli Ether, Ether
- 1 Ether = 1^18 Wei = 1,000,000,000,000,000,000

# The Ethereum Virtual Machine

- Simple stack based execution machine
- Runs bytecode instructions to transform the system state
- Word size limit – 256 bit, stack size – 1024 elements
- LIFO queue
- Turing Complete machine, limited by the amount of gas
- Supports exception handling
- Consists of extremely isolated run time environment – no access to any external filesystem or network there by providing increase security, deterministic execution and allows the execution of untrusted code
- Follows a stack based architecture
- Big-endian, using 256 bit wide words thereby supporting Keccak 256-bit hash and ECC computations

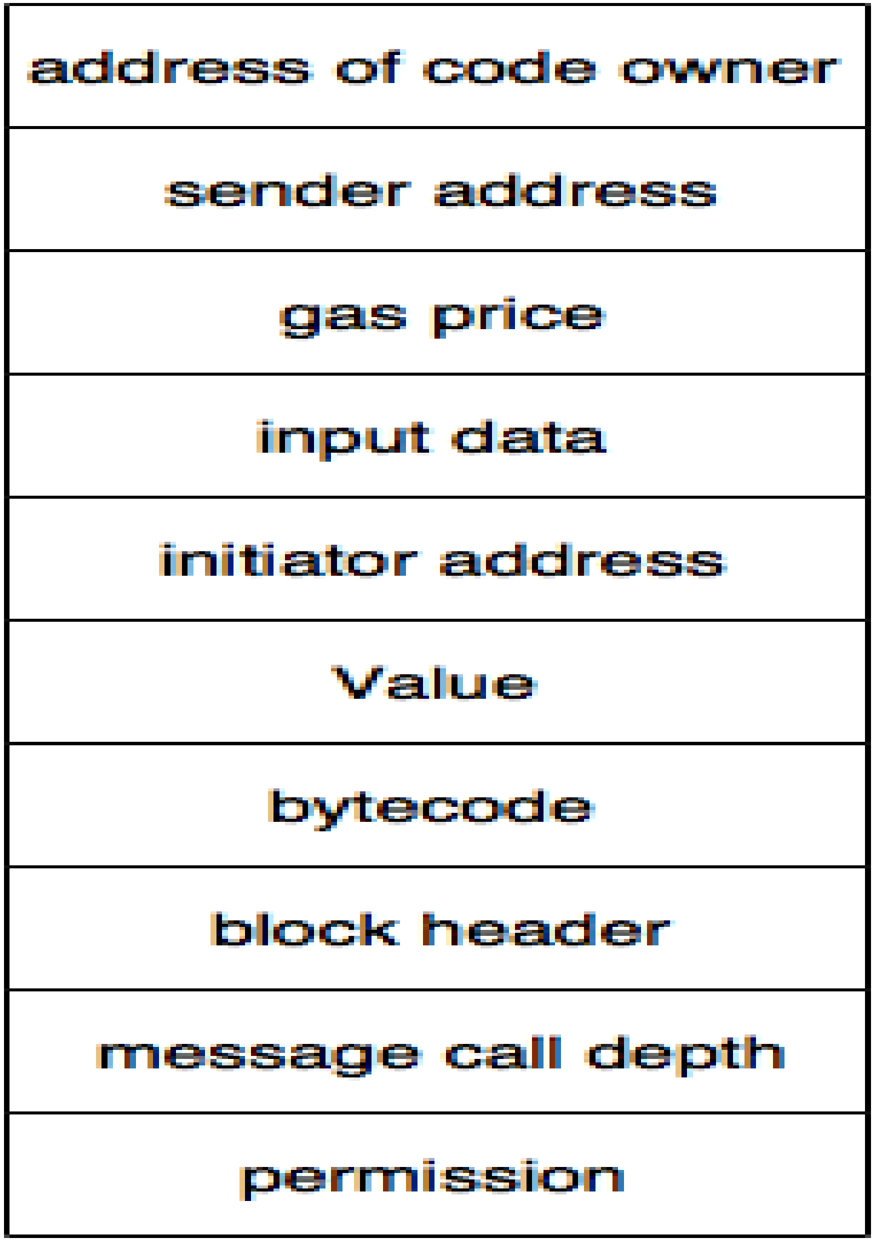
- Two types of storage available to contracts and EVM: memory and storage
- Memory is similar to RAM, while storage is a key value store that can be permanently stored
- Memory is unlimited, but constrained by gas fee requirements
- Storage is nonvolatile, and is maintained as a part of the system
- Keys and value are 32 bytes in size and storage
- Program code is stored in virtual Rom that is accessible by CODECOPY instruction – copies the program code into main memory
- Initially all storage and memory are set to 0 in EVM


POP
PUSH
1024
ITEM 1
EVM STACK
32-byte Values
PROGRAM
COUNTER
INSTRUCTIONS
MAIN MEMORY
STOP
ADD
PUSH 1
BY TE ARRAY
(256 BH WORD)
CODECOPY
OPCODE
VIRTUAL ROM
PROGRAM
CODE
STORAGE
(SYSTEM STATE)

- virtual ROM stores the program code that is copied into main memory using CODECOPY
- main memory is then read by the EVM by referring to the program counter and executes instructions step by step
- Program counter and EVM stack are updated accordingly with each instruction execution
- elements are pushed and popped from the stack
- Main memory gets the program code from virtual ROM / storage via the CODECOPY instruction
- Research area – WASM, EVM 2.1
  - run machine code in the browser that will result in execution at native speed
  - able to run the EVM instruction set (opcodes) natively in CPUs – faster and more efficient

# Execution Environment

- System state.
- Remaining gas for execution.
- The address of the account that owns the executing code.
- The address of the sender of the transaction.
- The originating address of this execution (it can be different from the sender).
- The gas price of the transaction that initiated the execution.
- Input data or transaction data depending on the type of executing agent.
- The address of the account that initiated the code execution or transaction sender.
- The value or transaction value - amount in Wei.
- The code to be executed presented as a byte array that the iterator function picks up in each execution cycle.
- The block header of the current block.
- The number of message calls or contract creation transactions currently in execution.
- Permission to make modifications to the state.



address of code owner

sender address

gas price

input data

initiator address

Value

bytecode

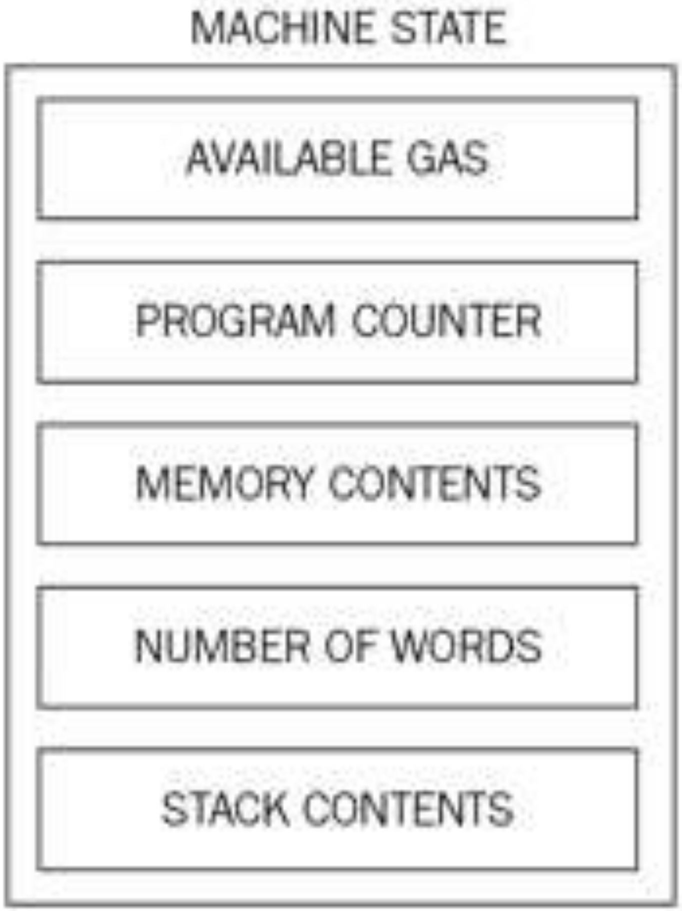
block header

message call depth

permission

# Machine state

- Maintained internally by the EVM
- Updated after each execution cycle of EVM
- Use of iterator function which outputs the result of a single cycle of the state machine
- Machine state is a tuple which consists of:
  - Available gas
  - The program counter - which is a positive integer up to 256 memory contents
  - Contents of the stack
  - Active number of words in memory
  - Contents of the stack



- Exception handling in EVM:
  - Not having enough gas required for execution invalid instructions
  - Insufficient stack items
  - Invalid destination of jump opcodes
  - Invalid stack size (greater than 1024)

# iterator Function

- used to set the next state of the machine
- It fetches the next instruction from a byte array where the machine code is stored in the execution environment.
- It adds/removes (PUSH/POP) items from the stack accordingly.
- Gas is reduced according to the gas cost of the instructions/opcodes. It increments the Program Counter (PC).

# Smart Contracts

- Precompiled contracts
- functions that are available natively on the blockchain to support various computationally intensive tasks
- run on the local node and are coded within the Ethereum client

# Native Contracts

- The elliptic curve public key recovery function
  - Available at adders 0x1
  - Requires 3000 gas fee for execution
  - Use of public key recovery mechanism for deriving the public key
  - Considers 4 inputs - H, V, R and S
- The SHA-256-bit hash function
  - Available at address 0x2
  - Produces SHA256 hash of the input
  - Gas requirements depend on input data size
  - Output is 32 byte value
- The RIPEMD-160-bit hash function
  - Available at address 0x3
  - provide RIPEMD 160-bit hash
  - Output is a 20 byte value
  - Gas requirements depend on input data size

- The identity/datacopy function
  - Available at address 0x4
  - defines output as input
  - The gas requirement is calculated by a simple formula: *15 + 3 [Id/32] where Id is the input data*
- Big mod exponentiation function
  - Available at address 0x5
  - Follows RSA signature verification
  - implements a native big integer exponential modular operation
- Elliptic Curve (EC) point addition function
  - Available at address 0x6
  - Implements EC point addition function
- Elliptic Curve (EC) scalar multiplication
  - Available at address 0x7
- Elliptic Curve (EC) pairing
  - Available at address 0x8