Public Blockchain

Rupesh Mishra

Introduction to Blockchain

1. Public Blockchain
2. Ethereum and its Components
3. Ethereum Virtual Machine
4. Transaction, Accounts
5. Architecture and Workflow
6. Bitcoin and Ethereum

1. Test-networks
2. Metamask, Mist Wallet,
3. Ethereum frameworks
4. Study of
   a. Ganache
   b. etherscan.io
   c. ether
5. block structure

Public blockchain

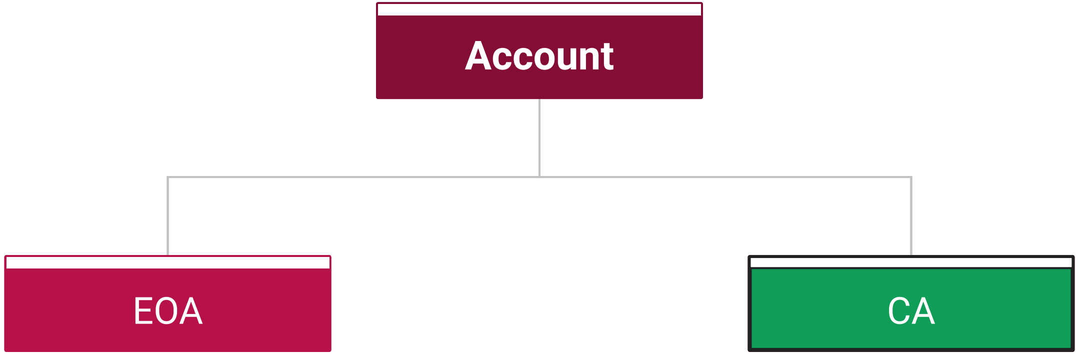
- **Permissionless**
  - ✓ Transection
  - ✓ Mining
- **Anonymity**
- **Transparency**
- **Auditable**
- **Immutable**
- **Neutral**
- **Low Scalability**
- **Decentralised**

- **Ethereum - The World Computer**
  - ✓ Virtual Machine
    - ■ Globally accessible singleton state
  - ✓ Executes Smart Contract
  - ✓ Blockchain
    - ■ Store and Synchronise the change of state
  - ✓ Execution resource utilisation cost is measured with Ether
  - ✓ Build DApp
    - ■ Censorship Resistance
    - ■ Counterparty Risk

Ethereum Component

- **P2P Network**
  - ✓ Ethereum Main Network
  - ✓ TCP port 3030
  - ✓ DEVp2p Protocol
- **Consensus Rule**
- **Transection (3)**
  - ✓ Sender
  - ✓ Receiver
  - ✓ Value
  - ✓ Data

- **State Machine**
  - ✓ Virtual Machine
  - ✓ Stack Based
  - ✓ Smart contract in HLL
  - ✓ Executes Bytecode
- **Data Structure**
  - ✓ LevelDB
  - ✓ Merkle Tree
- **Client**
  - ✓ Geth



Account

EOA

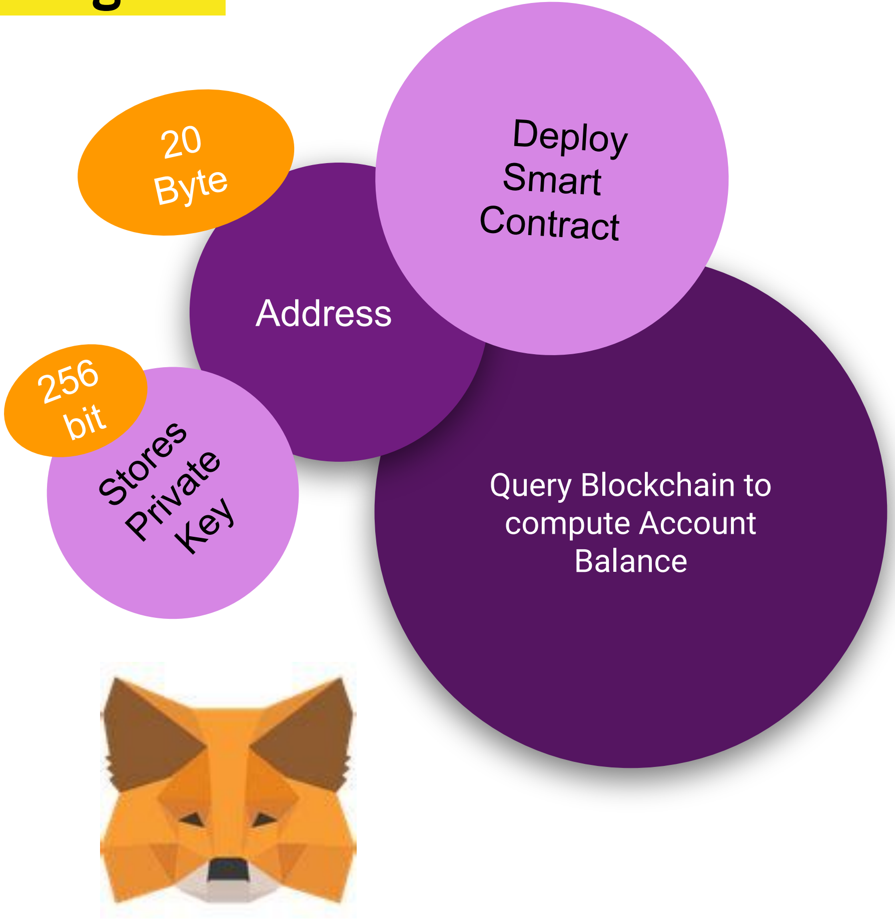
CA

- Owned by User
  Private Key
- Free
- Tx activated by user
- Call EOA and CA
- Private Key requires

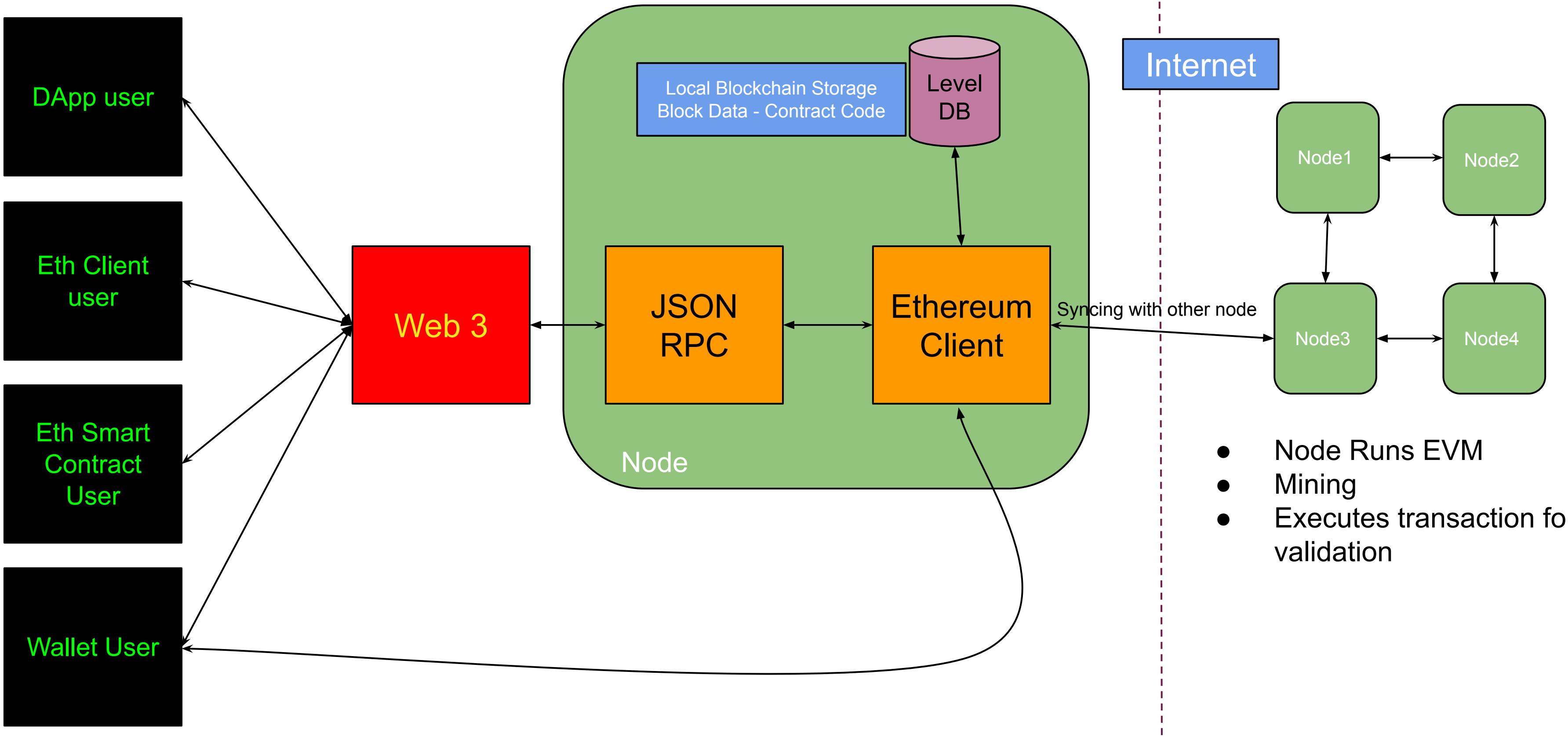
- Owned by Code
  Stored
- Not Free
- Tx activated by EOA
- Call other CA
- Private Key not used

- Two types
- Same for EVM
- Account Balance
  - ✓ Modified by Transections
- Storage
  - ✓ key-value store mapping
  - ✓ 256-bit words to 256-bit words

Wallet - A Generic Program

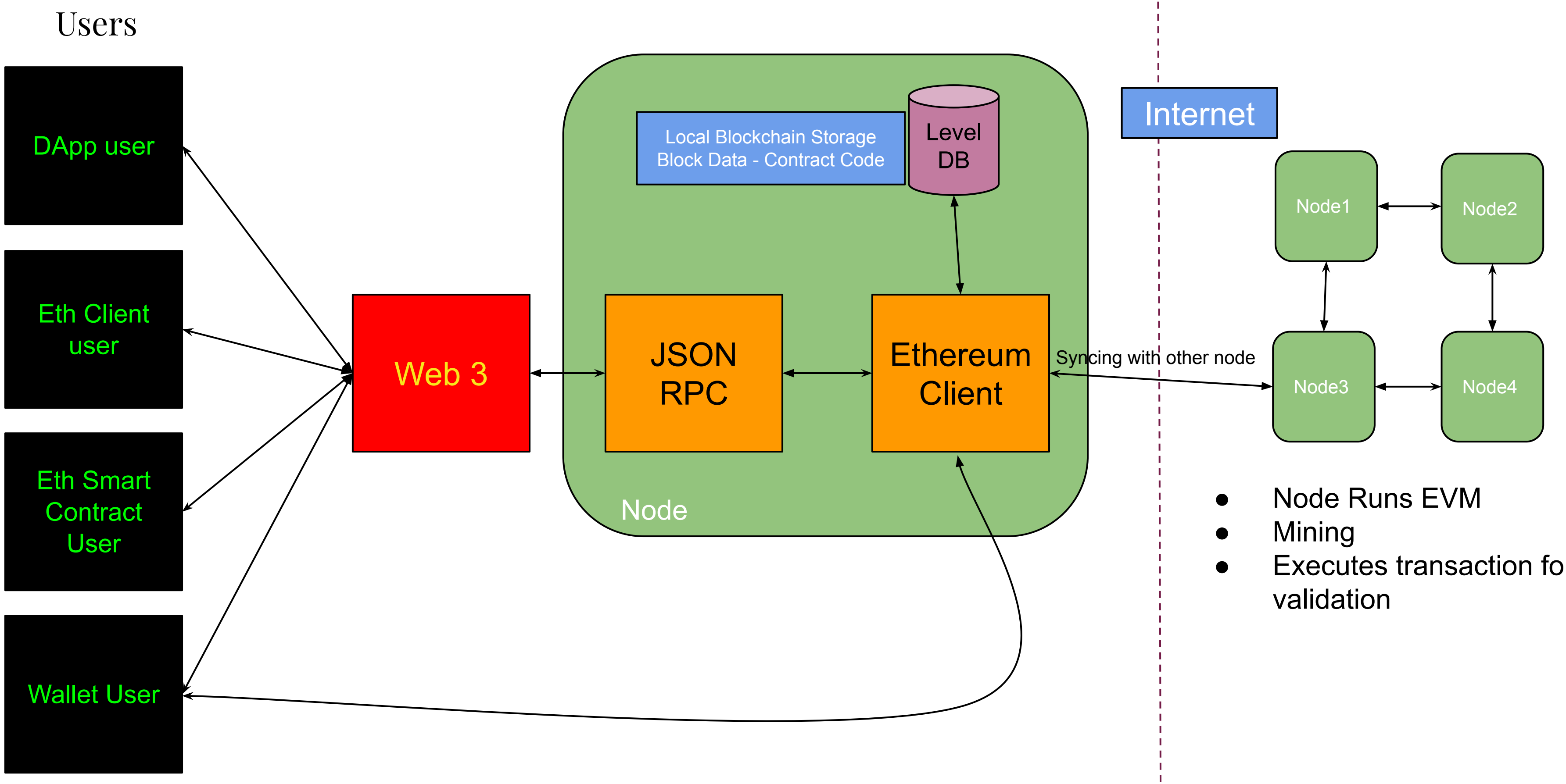


Ethereum Architecture and Ecosystem



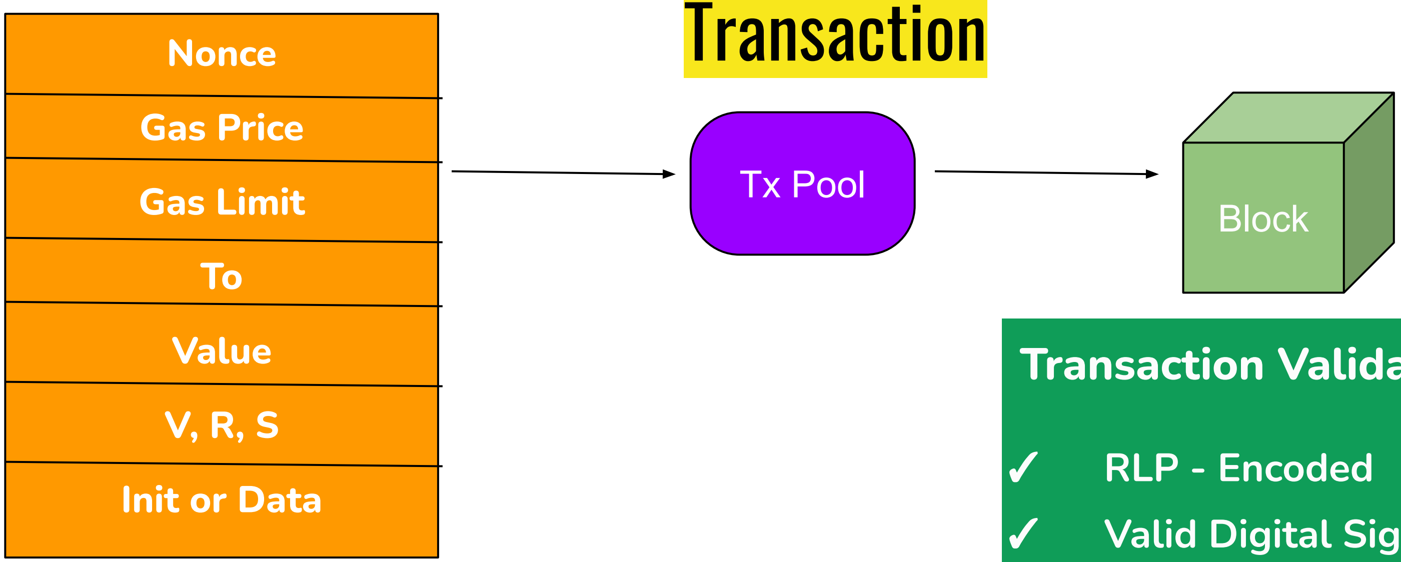
- Node Runs EVM
- Mining
- Executes transaction for validation

Ethereum Architecture



- Node Runs EVM
- Mining
- Executes transaction for validation

Transaction



Nonce

Gas Price

Gas Limit

To

Value

V, R, S

Init or Data

Tx Pool

Block

1. Tx Created by the user and sent to tx Pool
2. Waiting for Verification in Tx Pool
3. Mining Node pick highest paying tx from the pool and execute
4. After successful verification add into block and mine new block
5. Broadcast it to the network
6. Verified and Accepted by network

Transaction Validation

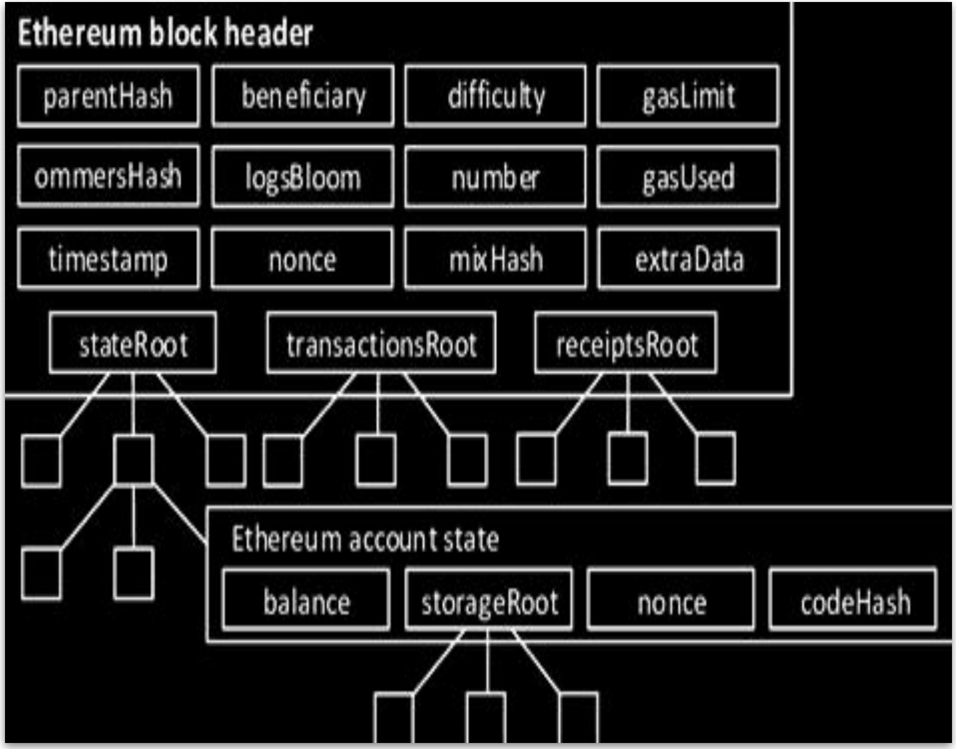
✓ RLP - Encoded

✓ Valid Digital Signature

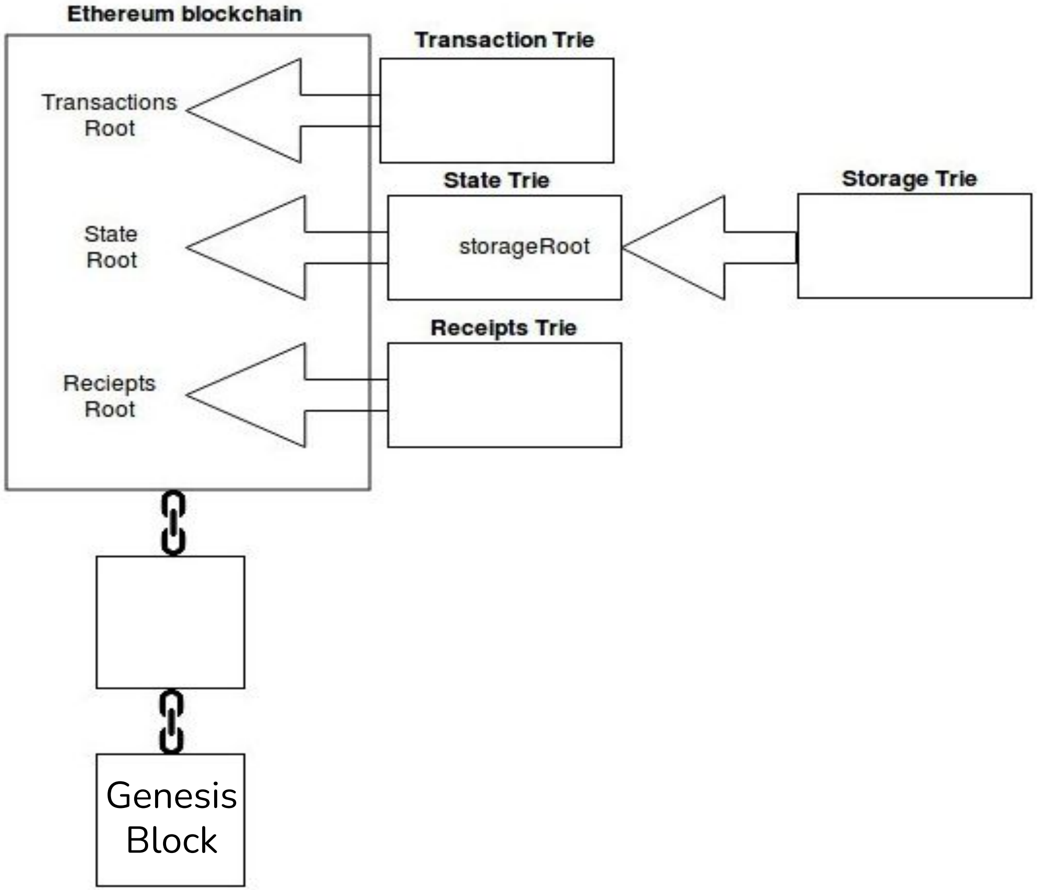
✓ Tx Nonce = sender's account current nonce

✓ Gas limit > gas consumed by the transection

✓ Sufficient balance (sender's account) to cover the cost



- **Block Validation**
  - ✓ Timestamp older than parent
  - ✓ Too many uncle (max 2)
  - ✓ Duplicate uncle
  - ✓ Uncle is ancestor
  - ✓ Uncles parent is not an ancestor
  - ✓ Non positive difficulty
  - ✓ Invalid mix digest
  - ✓ Invalid PoW
- **Block Finalization**
  - ✓ Ommers Validation
  - ✓ Transaction Validation
  - ✓ Reward Application
  - ✓ State and Nonce validation



1. **New Block Received by node**
2. **New MPT Constructed**
    a. All Transaction from the block is executed
    b. New Tx receipt generated and organised in MPT
    c. Global state is modified accordingly
3. **If New MPT roots match with the received Block is Valid**
    a. New Tx, Receipt, State tries are included in the local blockchain

- A message that is sent from one account to another account
- Include data and Ether
- Signature
  - ✓ secp256k1 curve
  - ✓ ECDSA Sign(message, Private Key) = (V,R,S)
  - ✓ V : To recover Pub Key from Pr Key, Depict the size and sign of the elliptic curve point
  - ✓ Signature (R,S) : S is calculated by multiplying R with the Pr Key and adding it into the hash and divide using random number selected to calculate R

- Target Contract Account
  - ✓ Code is executed
  - ✓ Payload provided as input data
- New contract
  - ✓ Target Account null
  - ✓ The payload of a contract creation transaction executed by EVM
  - ✓ The output data of execution is permanently stored as the contract code
- Modified Merkle-Patricia Tree (MPT)
  - ✓ Transaction Tries
- Keccak256 to compute Tx Root Hash

Gas

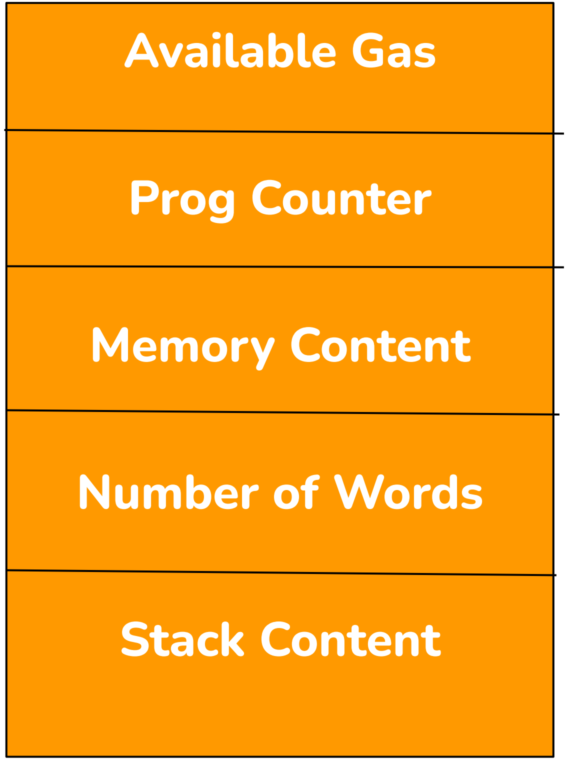
- **Contract creation transaction is charged with a amount of gas**
  - ✓ To limit the amount of work that is needed to execute the transaction
  - ✓ To pay for execution
- **EVM executes the transaction → the gas is gradually depleted**
- **The gas price value set by the creator of the transaction**
- **Sending account to pay → gas_price * gas**
- **Gas Limit**
  - ✓ An out-of-gas exception
  - ✓ Reverts all modifications made to the state in the current call frame
- **Remaining gas after execution is refunded to the creator**

EVM

- **Ethereum Virtual Machine**
  - ✓ Runtime Environment for Ethereum Smart Contract
  - ✓ sandboxed and completely isolated
  - ✓ Limited access to another Smart Contracts
  - ✓ RLP Encoding
- **Stack(256*1024)**
  - ✓ Stack not Register Machine
  - ✓ Computation performed on stack data
    - ■ Copy one of top 16 element
    - ■ Swap top with bellow 16
    - ■ Operation top 2 (2nd Optional)

- **Storage**
  - ✓ Persistent between function call
  - ✓ Key Value Pair
  - ✓ Costly (Read, Write, Initialize)
  - ✓ Contract access for R/W only its own storage
- **Memory**
  - ✓ Fresh instance each call
  - ✓ Linear
  - ✓ Addressed at Byte Level
  - ✓ Read(256) Write(8 to 256) bits
  - ✓ Expanded by Word(256)
  - ✓ Cost increases with increase in size (scales quadratically)

Machine State



Available Gas

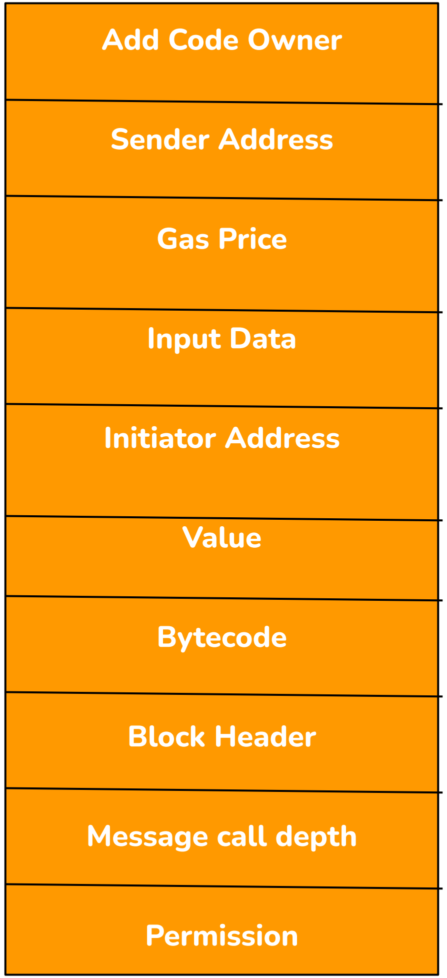
Prog Counter

Memory Content

Number of Words

Stack Content

Execution Environment



Add Code Owner

Sender Address

Gas Price

Input Data

Initiator Address

Value

Bytecode

Block Header

Message call depth

Permission

- **Instruction Set**
  - ✓ **Minimal set of Instruction to avoid inconsistency which may lead to consensus problem**
  - ✓ **Operates on data type, 256 bit words and memory slice**
  - ✓ **Arithmetic, bit, logical, comparison operations**
  - ✓ **Contracts can access relevant properties of the current block**
    - ■ Block number
    - ■ Timestamp, etc
  - ✓ **Conditional and Unconditional Jump**
- **Message Calls**
  - ✓ **Message calls are similar to transactions**
    - ■ Sender
    - ■ A target,
    - ■ Data payload,
    - ■ Ether
    - ■ gas price and limit
    - ■ return data.

- **Message Calls**
  - ✓ Contracts can call other contracts
  - ✓ Send Ether to non-contract accounts
  - ✓ A contract can decide its remaining gas should be sent with the inner message call and how much to retain
  - ✓ Exception in the inner call signaled by an error value from the stack.
  - ✓ Only the gas sent with the call is used up
  - ✓ The called contract receive a freshly cleared instance of memory
  - ✓ It has access to the call payload from calldata
  - ✓ It can return data to the caller
  - ✓ Fully synchronous.
  - ✓ Calls are limited to a depth of 1024,
    - ■ loops should be preferred over recursive calls

- **Delegatecall**
  - ✓ **Special Message Call**
  - ✓ **Code executed at the target address in the context of the calling contract**
  - ✓ **No change in the value of** msg.sender **and** msg.value
  - ✓ **Contract can dynamically load code from a different address at runtime**
  - ✓ **Storage, current address and balance still refer to the calling contract,**
  - ✓ **Only the code is taken from the called address**
  - ✓ **Library Feature**
  - ✓ **Reusable library code that can be applied to a contract’s storage**

- **Log**
  - ✓ Indexed data structure
  - ✓ Store data maps up to the block level
  - ✓ Used to implement events
  - ✓ Contracts cannot access log data
  - ✓ Accessed from outside the blockchain.
  - ✓ Some part of the log data is stored in bloom filters, for efficient and cryptographically secure searching
  - ✓ light clients" find these logs.

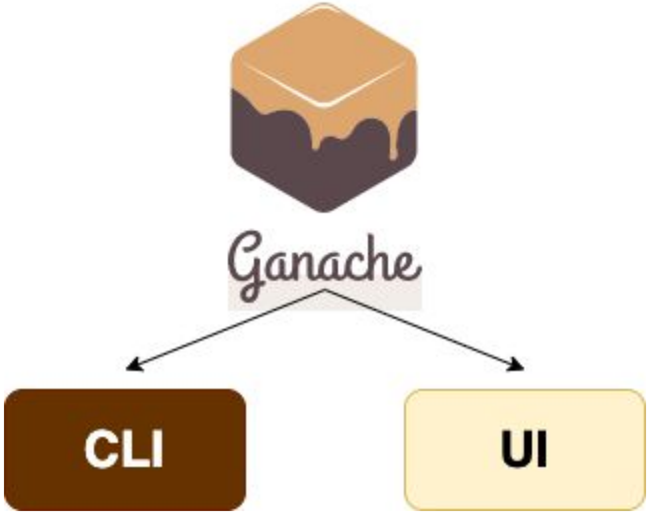
- **Create**
  - ✓ Contracts can create other contracts
  - ✓ The payload data is executed and the result stored as code
  - ✓ The caller / creator receives the address of the new contract
- **Deactivate and Self-destruct**
  - ✓ To remove code from the blockchain
  - ✓ self destruct operation
    - ▪ Ether stored at that address is sent to a designated target
    - ▪ The storage and code is removed from the state
  - ✓ Ether sent to removed contracts is forever lost.

Nodes and Miners

- **Nodes**
  - ✓ Wallet
  - ✓ Light Client
  - ✓ Full Node
- **Mining Node**
  - ✓ Ether in reward for validation and verification of blocks made up of transactions
  - ✓ Determine ommer block and include them in the chain
- **PoW Algorithm → Ethash**

- **PoS Algorithm → Casper with the release of Serenity**
  - ✓ Node → Bonding Validator
- **Consensus Mechanism**
  - ✓ Greedy Heaviest Observed Subtree (GHOST) Protocol
  - ✓ Heaviest Chain
- **Supporting Protocol**
  - ✓ Whisper
    - ■ Decentralised Messaging
  - ✓ Swarm
    - ■ Decentralised Storage

Ganache



Ganache

CLI

UI

- Ganache is available in two varieties:
  - UI : A desktop application
  - CLI : Command-line tool (previously known as the TestRPC)
- Work with Ethereum.
- Available for Windows, Linux, and Mac.

Introduction

- Personal Blockchain
- Development of Ethereum applications
- To build, test and deploy dApps in a secure and predictable environment
- Ganache is used to make a personal Ethereum Blockchain for testing Solidity contracts

- Features
  - ✓ Transactions are “mined” instantly.
  - ✓ No transaction cost
  - ✓ Accounts can be recycled, reset and instantiated with a fixed amount of Ether
  - ✓ no need for faucets or mining
  - ✓ Gas price and mining speed can be modified.
  - ✓ A convenient GUI gives overview of test chain events

Ethereum 2.0

Introduction

- **Ethereum 1.0**
- **World Computer**
  - ✓ Decentralised
  - ✓ Censorship resistant
- **Problem for Mass Adoption**
  - ✓ Scalability
  - ✓ Security
  - ✓ Performance

- **Trilemma**
  - ✓ Decentralisation
    - ■ Anyone can join, access and use the network
  - ✓ Scalability
    - ■ High Performance
    - ■ Efficiency
    - ■ Flexibility
  - ✓ Security
    - ■ Consistency
    - ■ Resilience to nw partition

Ethereum 2.0

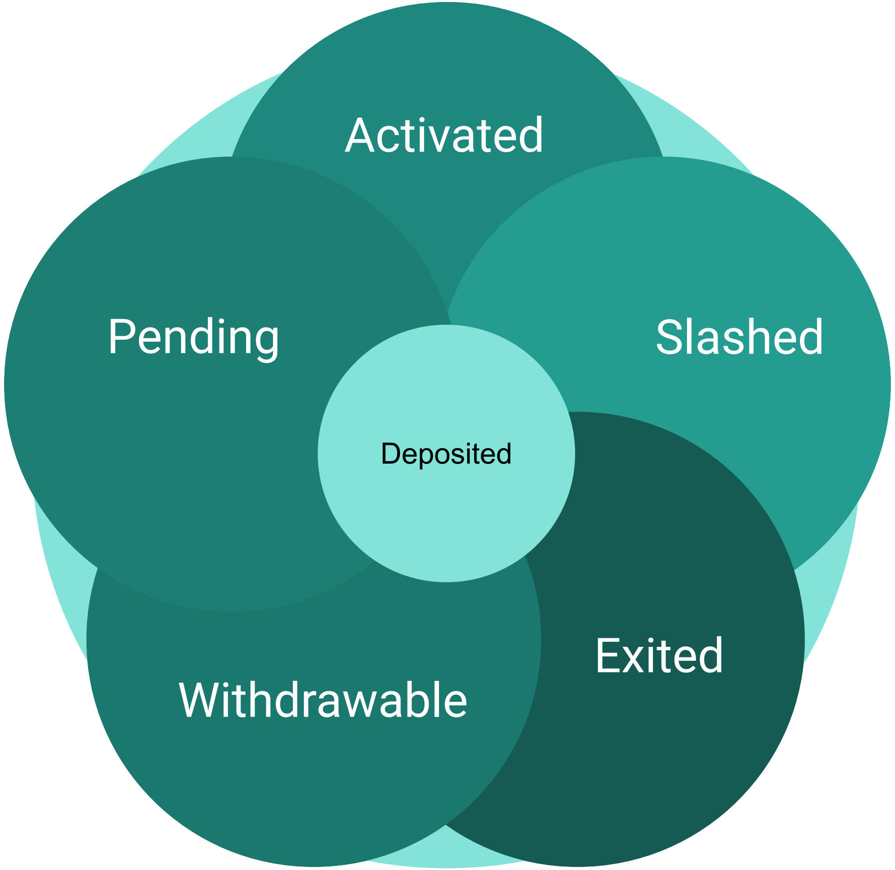
- **Solution**
  - ✓ **Proof of Stake**
    - ■ Validator and Staking
  - ✓ **Sharding**
    - ■ Side chain
    - ■ Privacy
    - ■ Shard chain - Beacon node - Beacon chain
  - ✓ **Ewasm**
    - ■ Efficient EVM
    - ■ Improved performance
    - ■ Fast Execution

- **Beacon Chain**
  - ✓ Core System chain
  - ✓ Backbone of entire Ethereum 2.0 chain system
- **Operate with main chain in parallel before merge**
- **Casper PoS consensus**
- **Includes Registry of Validators**
- **Shard write their states to enable cross shard transaction**

- **Functions**
  - ✓ Select Block Proposer and Attestation committee
  - ✓ Rules and Provision of Attestation
    - ▪ Availability of votes in a shard chain
    - ▪ Adequate number of attestation for a shard block will create cross link
    - ▪ Provide confirmation for shard block in the beacon chain
  - ✓ Validator and stake management
  - ✓ Provision of validator set for voting on proposed block
  - ✓ Provision of consensus, reward and penalty

- Beacon Node
- Synchronise beacon chain with other peers
- Listen to Deposit Event
- Random validator assignment
- Create and manage validator set
- Attestation of Block
  - ✓ Provide critical information to validator regarding attestation of assigned block

- Validator Node
  - ✓ Participate in consensus mechanism
  - ✓ 32 Ether stake
    - ■ Receive reward if attestation accepted
    - ■ Penalise for inactivity and slashed for dishonest act
  - ✓ Function
    - ■ Connect with beacon chain
    - ■ Propose new block
    - ■ Block attestation
    - ■ Attestation aggregation
    - ■ Shard chain synchronisation with beacon chain


Activated
Pending
Slashed
Deposited
Exited
Withdrawable

Whistleblowing Validator

- **Reports for penalization and slashing**
- **Slashin**
  - ✓ Validator attest two different version of chain
  - ✓ Confusion in the system that which chain is actually supported by validator
    - ■ Sign two different blocks in same epoch
    - ■ Sign two different conflicting attestation
    - ■ Sign attestation surrounds another attestation
- **Penalization**
  - ✓ Inactivity in the network

- **Shard Chain**
    - ✓ Scalability
    - ✓ 64 chain
- **Crosslink**
    - ✓ Set of attestation signatures from set of validator of block on shard chain
    - ✓ State of shard chain is periodically written in beacon chain
    - ✓ State is combined data merkle root of shard chain
    - ✓ Block on shard chain is also considered as final for cross shard transaction

- **Deposit contract created in Ethereum 1.0 used to deposit ether on beacon chain**
- **Event emitted every time deposit is done**
- **LMD GHOST for fork handling**
    - ✓ Chain with most attestation and stake
- **Casper FFG**
    - ✓ Maintain chain integrity
- **Merging**
    - ✓ Ethereum 1.0 chain will be merged into Ethereum 2.0