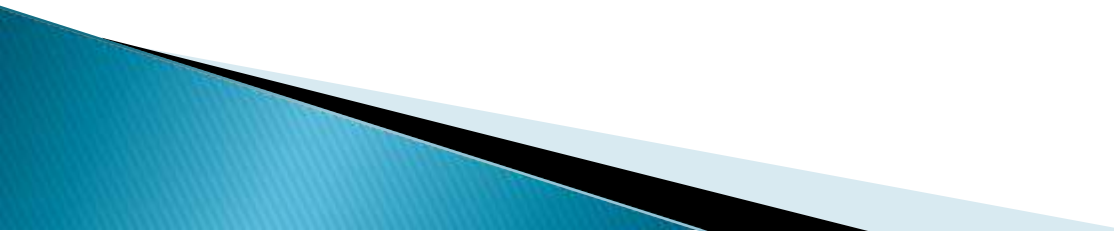


## MODULE 3- CHAPTER 1:


# CONSENSUS

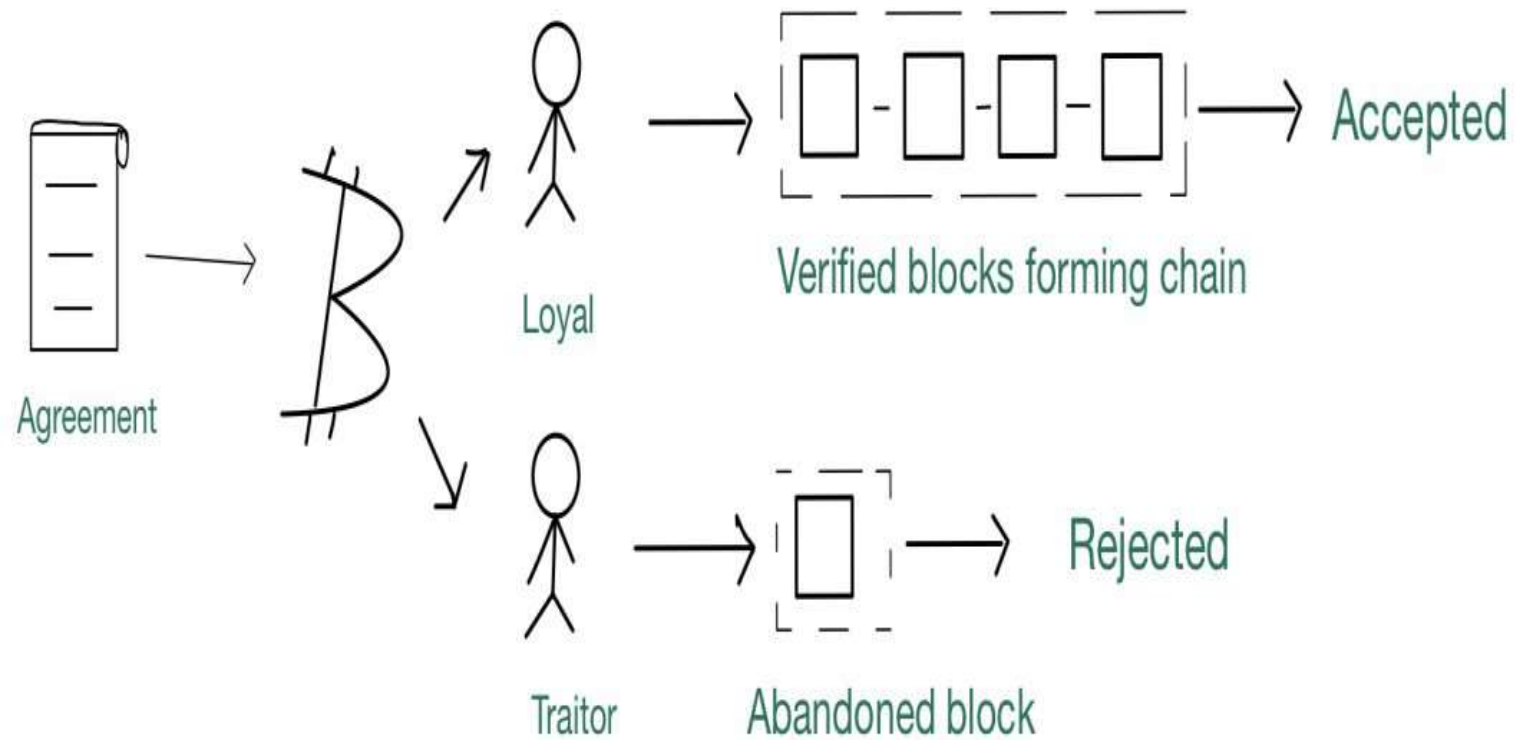


## 3.1.1 Consensus in Trust building Exercise

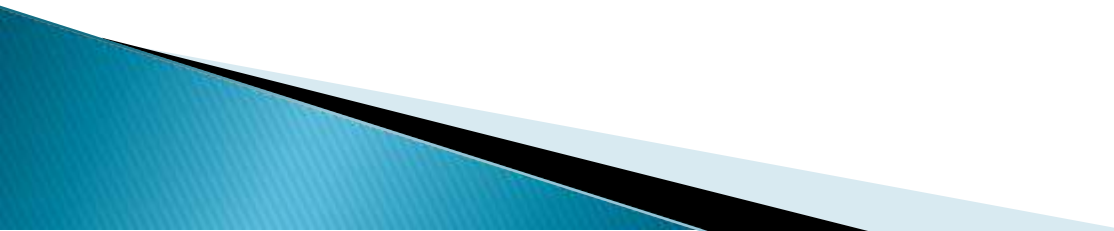
- ▶ A **consensus mechanism** is a procedure through which all the peers of the Blockchain network reach a common agreement about the present state of the distributed ledger.
  - ▶ In simpler terms, consensus is a dynamic way of reaching agreement in a group.
  - ▶ A method by which consensus decision-making is achieved is called “consensus mechanism”.
  - ▶ It confirms that each newly appended block is authentic.
- 

# How consensus works

- ▶ The nodes that validate transactions and propose new blocks are called miners.
  - ▶ Miners compete to generate a random number to unlock the next block on the chain.
  - ▶ The quickest miner to reach that number adds the next block and in exchange for the effort it receives a block reward.
  - ▶ The only way to win is to generate random numbers as quickly as possible (the "work" in the name) and get lucky. That is a contest of computing power, which in turn requires hardware and electricity.
- 



# Trust- through Consensus

- ▶ Why consensus?
    - **Centralized** system
    - Only **admins/central authority** of that server can **add/delete/update** the information
    - While **decentralized** networks are self-regulating systems with nodes that **don't trust each other** and that work on a global scale **without** any single authority
    - They involve contributions from hundreds of thousands of participants who work on verification and authentication of transactions occurring on the blockchain, and on the block mining activities
- 

- ▶ The consensus in blockchain is known as atomic broadcast among the nodes.
- ▶ This provides total order on the messages delivered to all the correct nodes.
- ▶ 2 asynchronous events take place:

### Broadcast and deliver

- ▶ Every node in the system may broadcast some messages(transactions)  $m$  by invoking the function  $\text{broadcast}(m)$ , and the broadcast protocol outputs  $m$  to local application on the node through  $\text{deliver}(m)$ .
- ▶ It also provides validity and integrity of the message delivered.
- ▶ Protocols such as *Paxos* is used.
- ▶ Real world systems like google's PageRank, smart grids, drone control use consensus algorithms.

# Types of consensus mechanisms

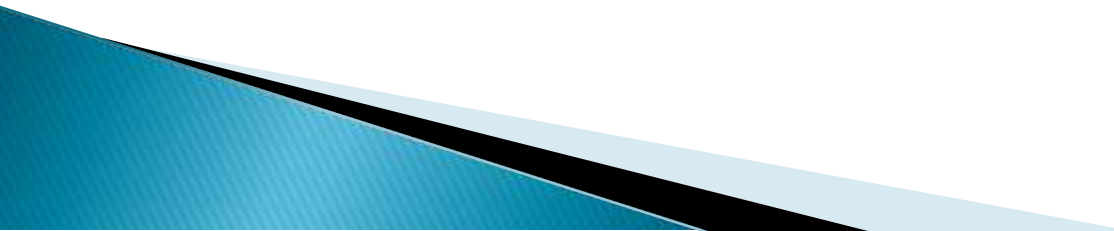
- ▶ There are two general categories of **consensus mechanisms**.
- ▶ These categories deal with all types of faults (fail stop type or arbitrary).  
These common types of consensus mechanisms are as follows:

- ▶ **Traditional Byzantine Fault Tolerance (BFT)-based:**

This method relies on a simple scheme of nodes that are publisher-signed messages. Eventually, when a certain number of messages are received, then an agreement is reached.

- ▶ **Leader election-based consensus mechanisms:**

This arrangement requires nodes to compete in a leader election lottery, and the node that wins proposes a final value. For example, the PoW used in Bitcoin falls into this category.



# Consensus Mechanism

- ▶ A set of **rules** that **decides** on the contributions by the various participants of the blockchain
- ▶ Different consensus mechanism algorithms work on **different** principles:
  - **PoW**: Proof of Work
  - **PoS** : Proof of Stake
  - **PoC** : Proof of Capacity
  - **PoB** : Proof of Burn
  - Proof of Elapsed Time
  - Proof of Activity
  - Proof of Proof

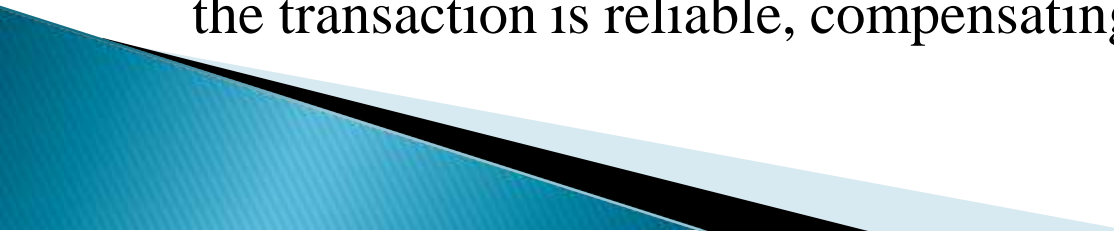


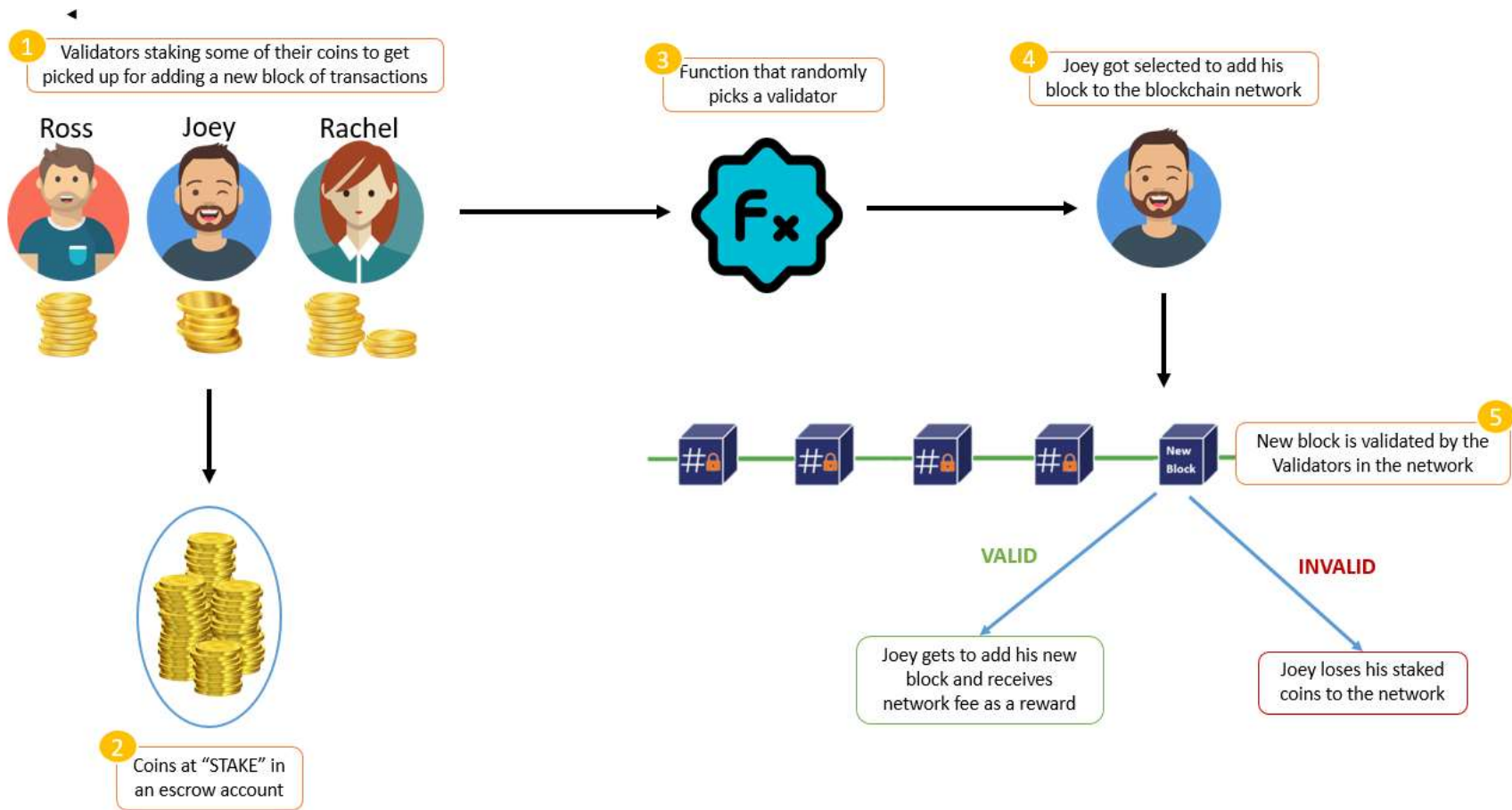
# PoW : Proof of Work

- ▶ Is a **common** consensus algorithm used by the **most** popular **cryptocurrency** networks like bitcoin and litecoin
- ▶ It requires a participant node to **prove** that the **work done** and **submitted** by them qualifies them to receive the right to **add** new transactions to the blockchain
- ▶ Proof-of-work is a type of consensus mechanism that requires network users called “miners” to devote computing power to complete a task.
- ▶ However, this whole mining mechanism of bitcoin needs **high energy consumption** and **longer processing time**



# PoS : Proof of Stake

- ▶ A **low-cost, low-energy consuming, alternative** to PoW algorithm
  - ▶ Proof of work is a competition between miners to solve cryptographic puzzles and validate transaction in order to earn block rewards.
  - ▶ Under proof-of-stake (POS), validators are chosen based on the number of staked coins they have.
  - ▶ To become a validator, a coin owner must "stake" a specific amount of coins.
  - ▶ Proof of stake implements randomly chosen validators to make sure the transaction is reliable, compensating them in return with crypto.
- 



# **Mining**

## **Proof of Work**



# **Staking**

## **Proof of Stake**



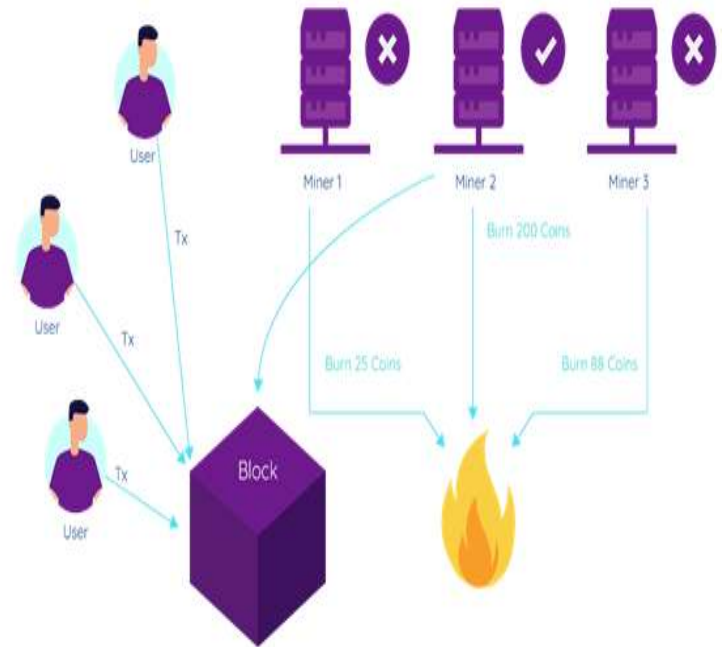
# PoC : Proof of Capacity

- ▶ PoC allows miners to generate a list of all the possible hashes beforehand. Then store them on local hard drives.
- ▶ Hence, the larger the hard drive, the more possible solutions. It increases the chances of getting the correct hash.
- ▶ It allows for mining devices in the network to use their available hard drive space to decide mining rights and validate transactions.



# PoB : Proof of Burn

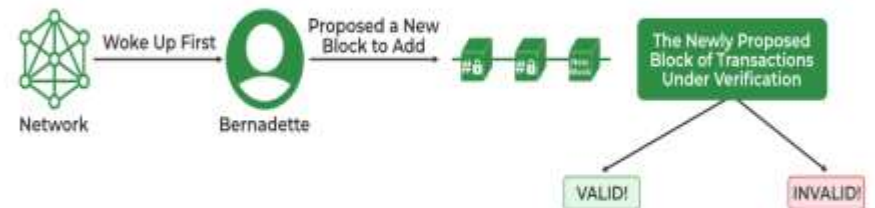
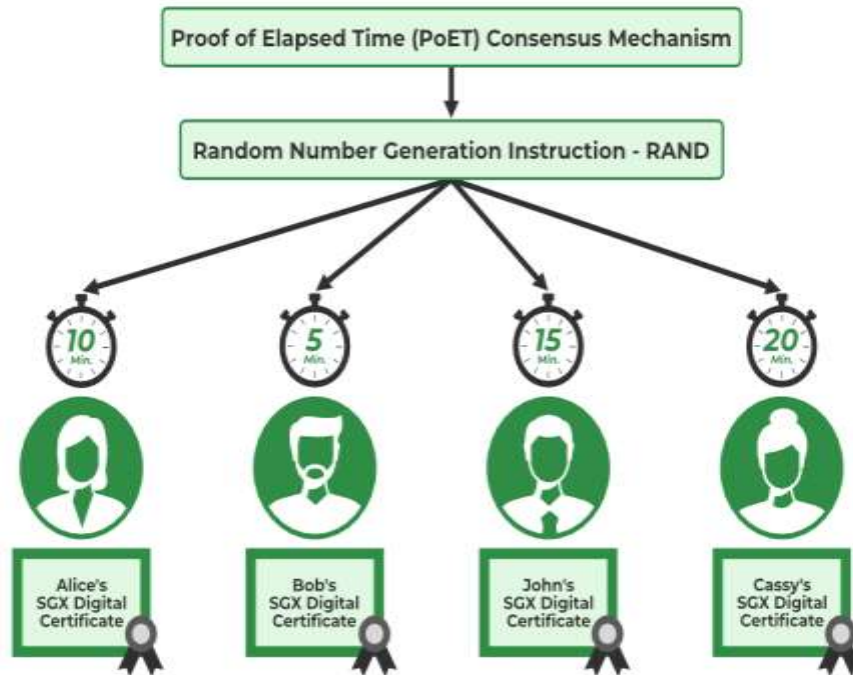
- ▶ Instead of investing into expensive hardware equipment, validators **'burn'** coins by sending them to an address from where they are irretrievable
- ▶ Validators earn a privilege to mine on the system based on a random selection process
- ▶ The **more** coins they burn, the better are their chances of being selected to mine the next block.
- ▶ burning coins means that validators have a long-term commitment in exchange for their short-term loss.
- ▶ Validators have a long-term commitment in exchange for their short-term loss



# Proof of Elapsed Time

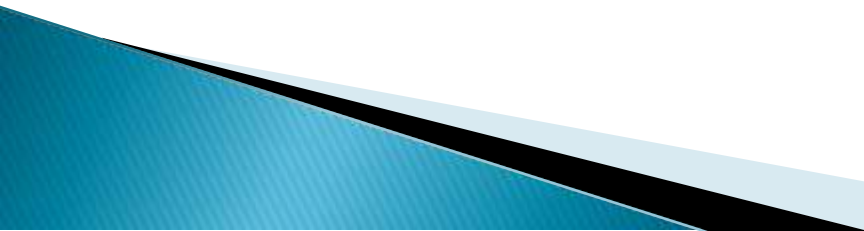
- ▶ Every node in the blockchain network will generate a random wait time and then sleeps for that specified duration.
- ▶ Now the first one to wake up, that is, the first node in the network to have its timer expire after completing its specified waiting time wins the block round and it now becomes the block leader which produces the new block and signs the new block with its private key address before adding it to the blockchain network.
- ▶ Winner(elected leader) has to convince that
  - it had the shortest wait time.
  - did not broadcast the block until the wait time was over



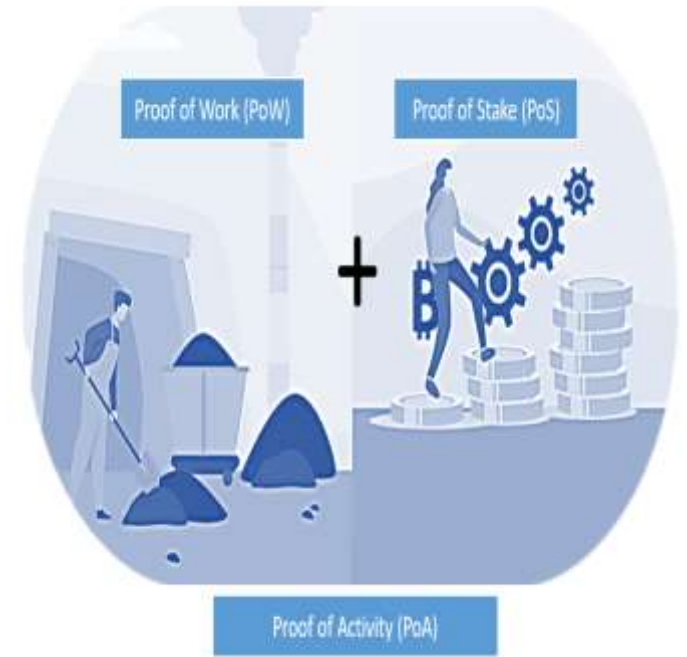




# Proof of Activity

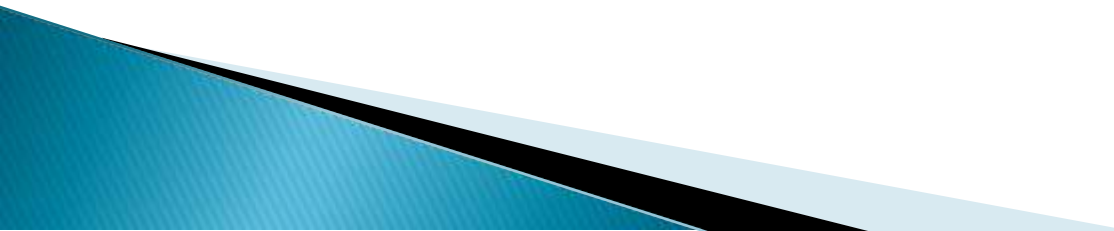
- ▶ Proof of Activity (PoA) is a hybrid consensus algorithm that combines elements of Proof of Work (PoW) and Proof of Stake (PoS) to achieve a balance between security and efficiency.
  - ▶ **First, PoA uses the mining concept by proof of work**, where miners have to do complex mathematical computations to prove their efforts and sincerity to the network.
  - ▶ Instead of mining a block of transactions, **PoA allows miners to mine (or add) a blank template block with header information and miner's address.**
  - ▶ **Once this almost empty block gets mined, the mechanism switches to Proof of Stake (PoS).**
- 

- Then a group of validators is picked at random. They are responsible for validating or signing the new block.
- Validators analyze the header information of the block and then mark it as valid or invalid.
- This process continues until a block receives the required number of validators (or signers). The higher coins the network participant holds, the stronger are their chances of being picked up as a validator (or signer).
- Once the group of validators signs the new block by the miners, it's marked as a valid, complete block and gets added to the blockchain network. Finally, the transactions will be recorded on the newly added block.



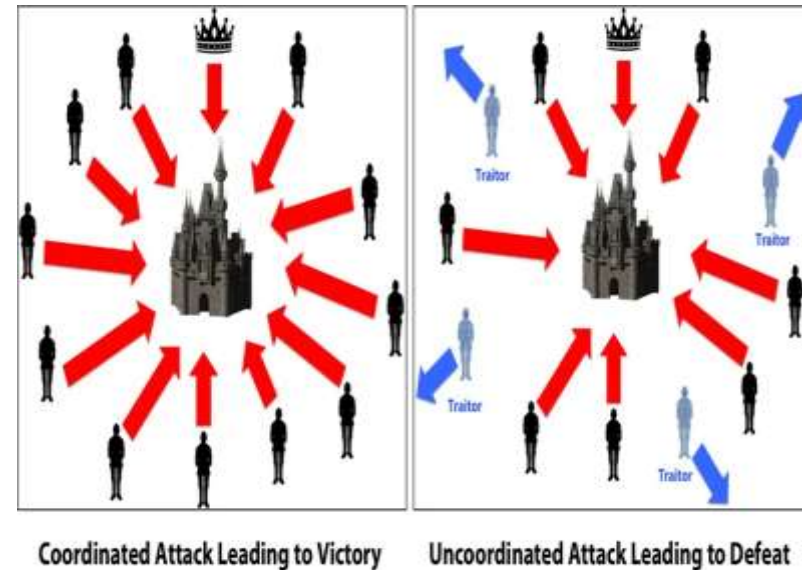
**Decred (D-Cred)** is the best-known cryptocurrency that utilizes proof of activity.

# Proof of Proof

- ▶ New concept compared to other consensus mechanism.
  - ▶ New type of miner is introduced, which performs periodic publications of one blockchain's current state to another.
  - ▶ It ensures that existing blockchain has some means of creating new blocks such as Proof of Work, Proof of Stake etc
- 

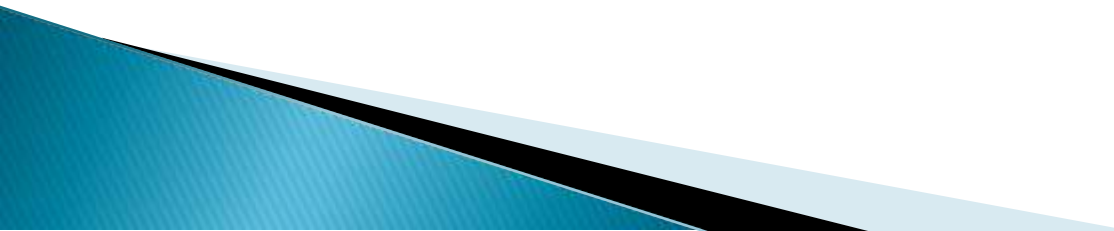
## 3.1.2 Byzantine Agreement Methods

- ▶ The most traditional way to reach consensus is via a Byzantine Agreement (a foundational concept in blockchain technology).
- ▶ Nodes on a blockchain validate blocks of data by reaching consensus on the solution to a given problem.
- ▶ A Byzantine Agreement is reached when a certain minimum number of nodes (known as a quorum) agrees that the solution presented is correct, thereby validating a block and allowing its inclusion on the blockchain.




# Byzantine Agreement based Consensus Methods

a) **Practical Byzantine Fault Tolerance (pBFT)** is a consensus algorithm that seeks to tolerate Byzantine faults (node failures).

- ▶ Based on voting Process.
  - ▶ In order to reach a consensus and add a next block, more than two thirds of all the nodes should agree upon the block.
  - ▶ In other words, **pBFT works on the assumption that the number of malicious nodes cannot be equal to or more than one-third of the total number of nodes in the network.**
  - ▶ It does not require owning assets like PoS.
  - ▶ pBFT has high throughput, low latency, low computational overhead.
- 

## **b) Delegated Byzantine Fault Tolerance (dBFT)**

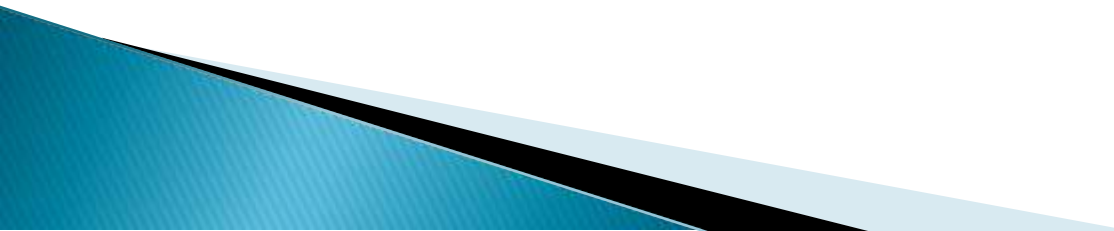
- ▶ It functions by selecting a limited group of trusted nodes, called delegates or consensus nodes, to validate and confirm transactions.
  - ▶ By using a smaller group, dBFT reduces energy consumption, and enhances transaction processing speed in comparison to traditional Byzantine Fault Tolerance mechanisms.
  - ▶ NEO is a popular blockchain platform that focuses on building a smart economy system.
  - ▶ It was the first to implement dBFT consensus in its blockchain network.
  - ▶ NEO's dBFT algorithm ensures that the system operates efficiently, guarantees immediate transaction finality, and is resistant to various attacks
- 

## c) Stellar Consensus Protocol (SCP)

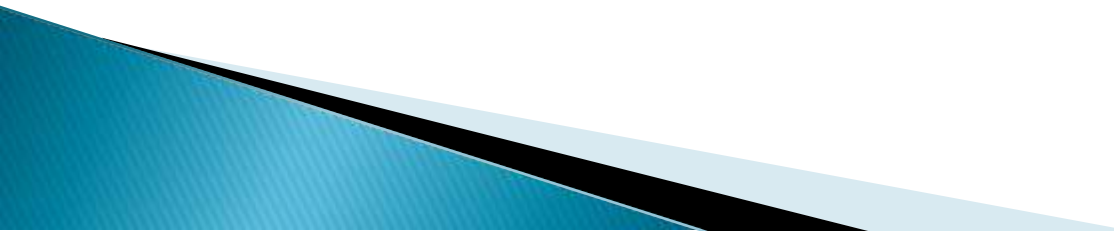
- ▶ It is a variant of pBFT.
- ▶ Uses Federated Byzantine Fault Tolerance as the Backbone.
- ▶ As it is public and decentralized, it allows everyone to participate in the consensus protocol.
- ▶ A set of nodes that participate in the consensus protocol forms a Quorum.

### Two steps in the SCP Consensus:

#### **Nomination Protocol:**

- ▶ For each consensus slot, the nomination protocol produces candidate values.
  - ▶ Every node in the quorum will vote for a single value among the candidate values.
  - ▶ process ends by unanimously choosing the values for that slot.
- 

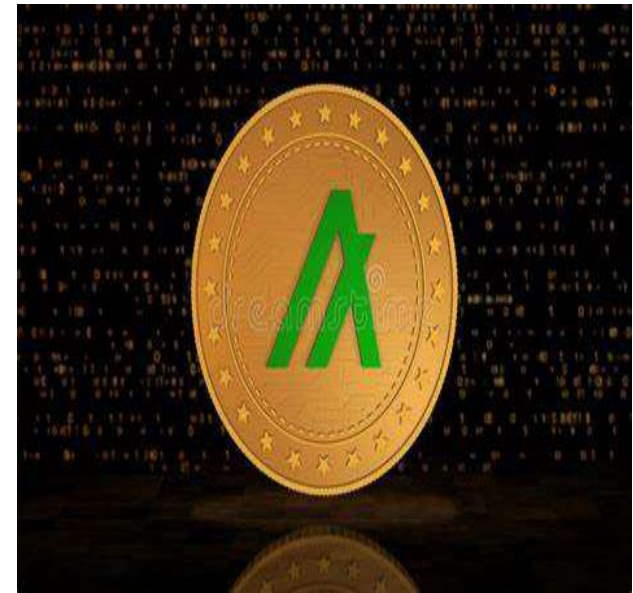
## Ballot protocol

- ▶ It is initiated after nomination protocol.
  - ▶ it involves federated voting to either accept or reject the obtained values in the nomination protocol.
  - ▶ In the ballot protocol, a ballot is tied to the candidate value, and a node must commit or abort the value tied to that ballot. To avoid agreement blocking, nodes can abort certain votes and move on to another.
  - ▶ higher valued ballot is considered as a new ballot protocol execution.
- 



## d) Algorand

- ▶ Algorand is a decentralized permissionless blockchain protocol that anyone can use to develop applications and transfer value.
- ▶ The Algorand protocol is powered by a novel consensus algorithm that enables fast, secure and scalable transactions.
- ▶ The native currency of the Algorand network is called ALGO. ALGO tokens are used to pay for transaction fees and reward users who participate in the network's consensus process.

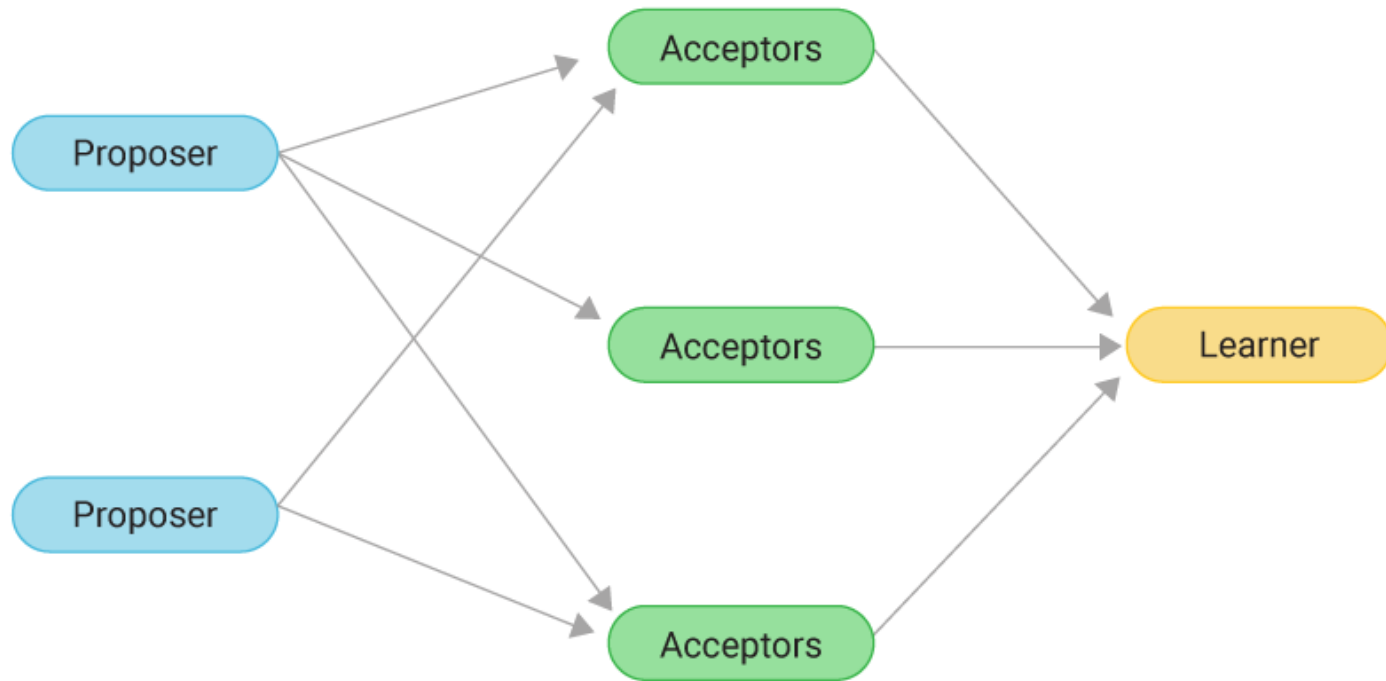


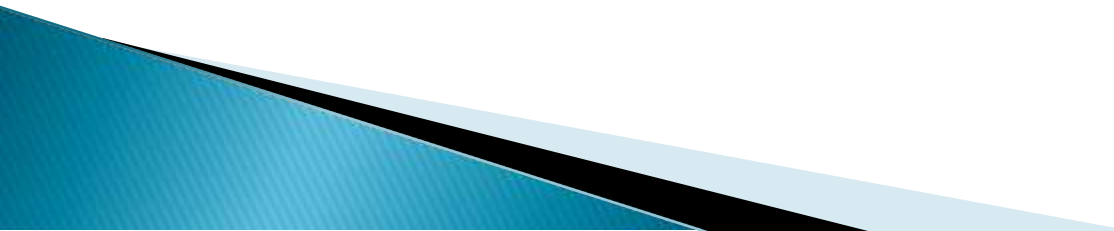
- ▶ In this approach, each block is approved by a unique committee of users that are randomly selected using Verifiable Random Function (VRF) using users' private key and public information from the blockchain.
- ▶ Selection is based on the weights of the users assigned according to the amount of money in their account.



## e) Paxos and Raft Consensus Protocols


- Paxos is the oldest consensus protocol published way back in 1989 by Leslie Lamport.
- Paxos and Raft are two prominent consensus algorithms used to achieve data consistency and fault tolerance in distributed systems.
- Both algorithms aim to ensure that a group of nodes in a distributed network agree on a single value or sequence of operations, even in the presence of failures and network partitions.
- There are three roles in Paxos consensus, known as agents:
  - Proposers
  - Acceptors
  - Learners



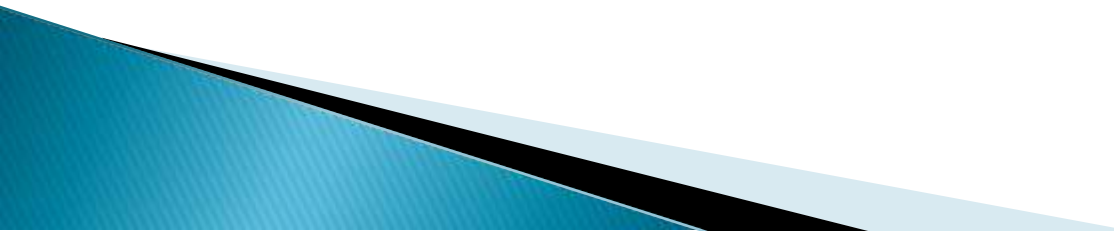
- ▶ Round of consensus begins when a proposer sends a proposed value to a group of acceptors.
  - ▶ Acceptors may accept the proposed value by a given proposer, and once a certain threshold is met, then that value is approved by the network.
  - ▶ Paxos uniquely indexes each proposed value that an acceptor receives which allows them to accept more than one proposal.
  - ▶ A unique number defines each proposal, and the network selects a value once a specific proposed value is accepted by the majority of acceptors, known as the chosen value.
- 

Breaking down the roles of the proposer and acceptor in the protocol is as follows:

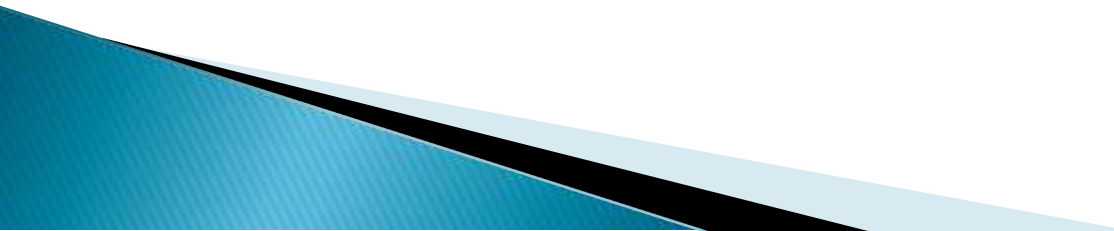
## Proposer

- ▶ Submit proposal  $n$  to acceptors along with prepare request, wait for a majority to reply.
  - ▶ If the majority of acceptor reply they agree, they will reply with the agreed value. If majority reject, then abandon and restart the process.
  - ▶ Proposer sends a commit message with  $n$  and value if the majority accepts.
  - ▶ If the majority of acceptors accept the commit message, protocol round completes.
- 

## Acceptor

- ▶ Receive proposal and compare it to the highest numbered proposal already agreed to.
  - ▶ If  $n$  is higher then accept the proposal, if  $n$  is lower then reject the proposal.
  - ▶ Accept subsequent commit message if its value is the same as a previously accepted proposal and its sequence number is the highest number agreed to.
- 

## learners

- ▶ Finally, the role of the learners is to discover that a majority of acceptors have accepted a proposal from the proposers.
  - ▶ A distinguished learner is selected that propagates the chosen value to the other learners in the network.
  - ▶ Variations of this process can be used where either all acceptors inform corresponding learners of their decisions or acceptors respond to a distinct set of learners who then propagate the message to the rest of the learners.
- 

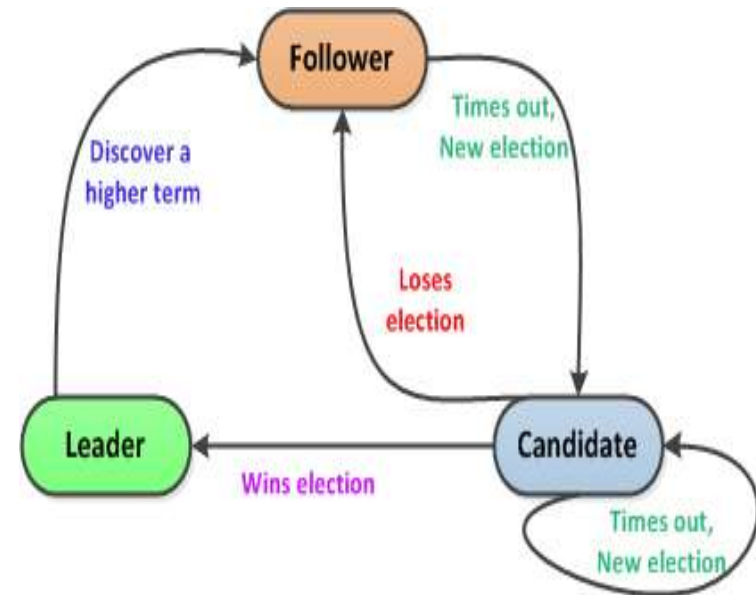


# Raft Consensus Mechanism

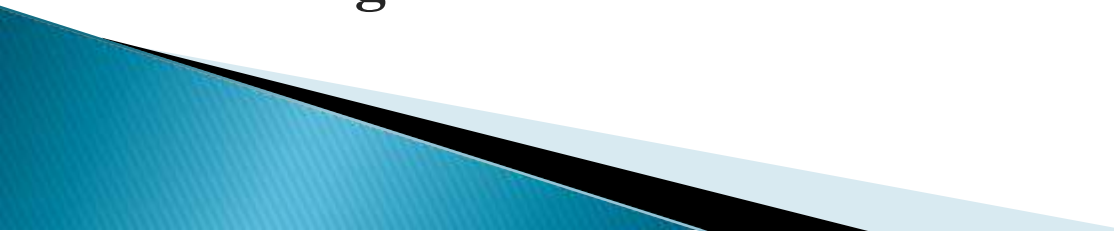
Raft can be described as a simpler version of Paxos.

- ▶ There are 3 possible states for a node in Raft consensus protocol:
  1. Leader
  2. Follower
  3. Candidate
- ▶ All the nodes start off with a Follower state. If followers don't hear from a leader then they can become a candidate.
- ▶ The candidate then requests votes from other nodes.
- ▶ The candidate becomes the leader if it gets votes from a majority of nodes. This process is called a **Leader election**.

- ▶ All changes to the system now go through the leader.
- ▶ Each change is added as an entry in the node's log. Values remain in uncommitted state and get written to the log first, and then these logs are replicated to other nodes.
- ▶ Once a majority of nodes respond, the value gets committed to the leader node as well. The leader then notifies the followers that the entry is committed, so that the followers can commit the value as well. This process is called **Log replication**.



## Leader election

- ▶ Every follower waits for a certain amount of time before becoming a candidate. This time is called the **election timeout**.
  - ▶ After the election timeout the follower becomes a candidate and starts a new **election term**. It Votes for itself and sends out Request Vote messages to other nodes. If the receiving node hasn't voted yet in this term then it votes for the candidate and the node resets its election timeout.
  - ▶ Once a candidate has a majority of votes it becomes leader. Leader keeps sending regular messages to followers to keep it's status as leader. This message is called a **heartbeat message**.
- 

## Lamport-Shostak-Pease Algorithm

Lamport et al.'s algorithm, referred to as the Oral Message algorithm  $OM(m)$ ,  $m > 0$ , solves the Byzantine agreement problem for  $3m+1$  or more processors in the presence of at most  $m$  faulty processors.

Let  $n$  denote the total number of processors (clearly,  $n \geq 3m+1$ ).

### Algorithm $OM(0)$

1. The source processor sends its value to every processor.
2. Each processor uses the value it receives from the source. (If it receives no value, then it uses a default value of 0)

## Algorithm OM(m), $m > 0$

1. The source processor sends its value to every processor.
2. For each  $i$ , let  $v_i$  be the value processor  $i$  receives from the source. (If it receives no value, then it uses a default value of 0.). Processor  $i$  acts as the new source and initiates Algorithm OM( $m-1$ ) wherein it sends the value  $v_i$  to each of the  $n-2$  other processors.
3. For each  $i$  and each  $j$  ( $j \neq i$ ), let  $v_j$  be the value processor  $i$  received from processor  $j$  in Step 2. using Algorithm OM( $m-1$ ). Processor  $i$  uses the value majority( $v_1, v_2, \dots, v_{n-1}$ ).

The message complexity of the algorithm is  $O(n m)$

## **MODULE 3- CHAPTER 2:**

# **ETHEREUM**



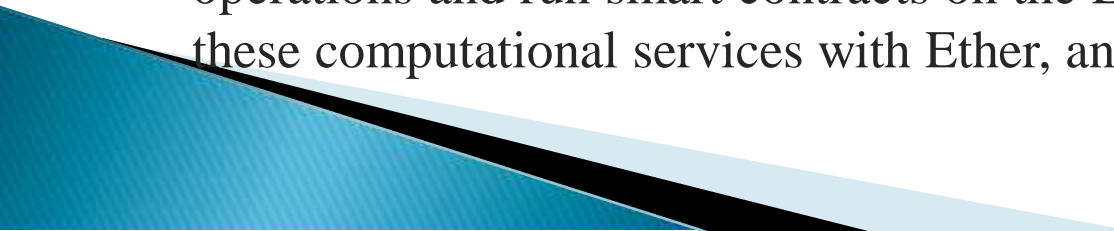
## 3.2.1 Ethereum

- Ethereum is a blockchain-based platform best known for its cryptocurrency, Ether (ETH).
- The blockchain technology that powers Ethereum enables secure digital ledgers to be publicly created and maintained.
- ▶ Ethereum is a blockchain-based network that enables developers to build and deploy decentralized applications and smart contracts.
- ▶ You can create an Ethereum account from anywhere, at any time.
- ▶ Ethereum Blockchain was launched in

2015



Ethereum consists of several **key components**:

- **Smart contracts**: Smart Contracts are self-executing rules/codes. Contracts execute themselves when transactions happen.
  - **The Ethereum blockchain**: A record of Ethereum's entire history – every transaction and smart contract call is stored in its blockchain.
  - **Consensus mechanism**: The method for validating and recording data on the blockchain.
  - **The Ethereum Virtual Machine (EVM)**: The part of Ethereum that executes the rules of Ethereum and makes sure a submitted transaction or smart contract follows the rules.
  - **Ether**: Ethereum's token, which is required to make transactions and execute smart contracts on Ethereum.
  - **Gas** is the unit used to measure the computational effort required to execute operations and run smart contracts on the Ethereum network. Users pay for these computational services with Ether, and this is known as gas.
- 

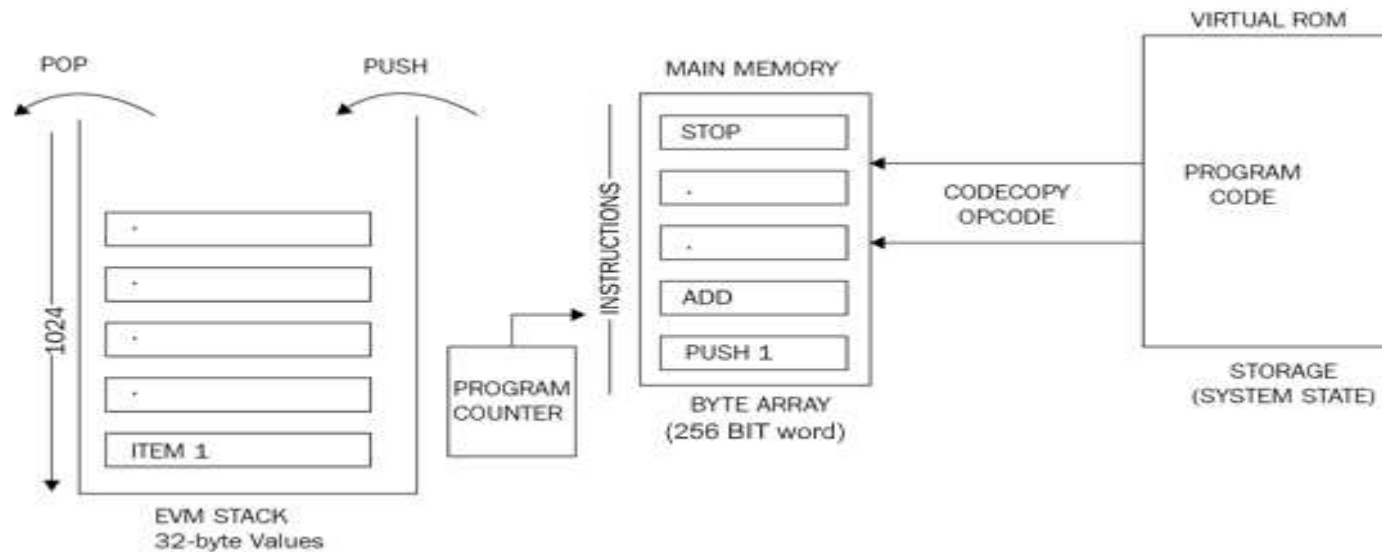


## 3.2.2 Ethereum Virtual Machine

- Ethereum contains its own Turing-complete scripting language, called **Solidity**, and with this comes a need to execute this code.
- A program called the Ethereum Virtual Machine (EVM) can do this task.
- It runs on top of the Ethereum network, meaning that all nodes reach a consensus about what code should be executed at every given time.

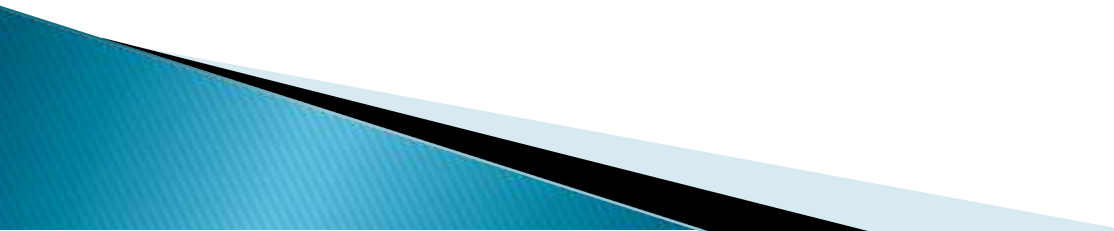
EVM is best thought of as a virtual computer on the blockchain that turns your ideas into code, and runs it on the global Ethereum network.

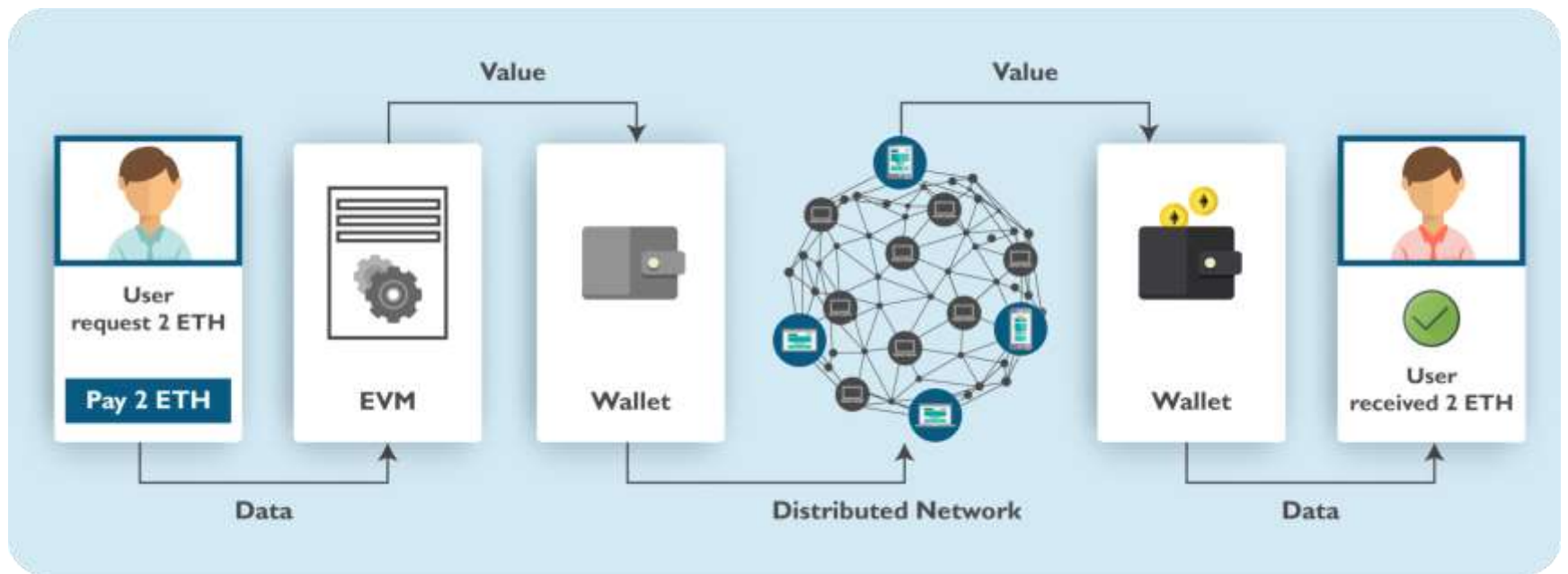
- An EVM is the runtime environment that executes Ethereum smart contracts.



- The EVM executes as a stack machine with a depth of 1024 items.
- Each item is a 256-bit word, which was chosen for the ease of use with 256-bit cryptography (such as Keccak-256 hashes and Elliptic Curve Cryptography).

## Two types of storage available:

- **Memory(byte array):** when a smart contract finishes the code execution, the memory is cleared.
  - **Storage(Virtual ROM):** Key value(32 bytes) store permanently stored on the blockchain.
  - Program code stored in VROM is accessible using **COPYCODE** instruction to the main memory.
  - Main memory is read by the EVM by referring the program counter and executes instructions step by step.
  - Program counter and EVM stack are updated accordingly with each instruction execution.
- 

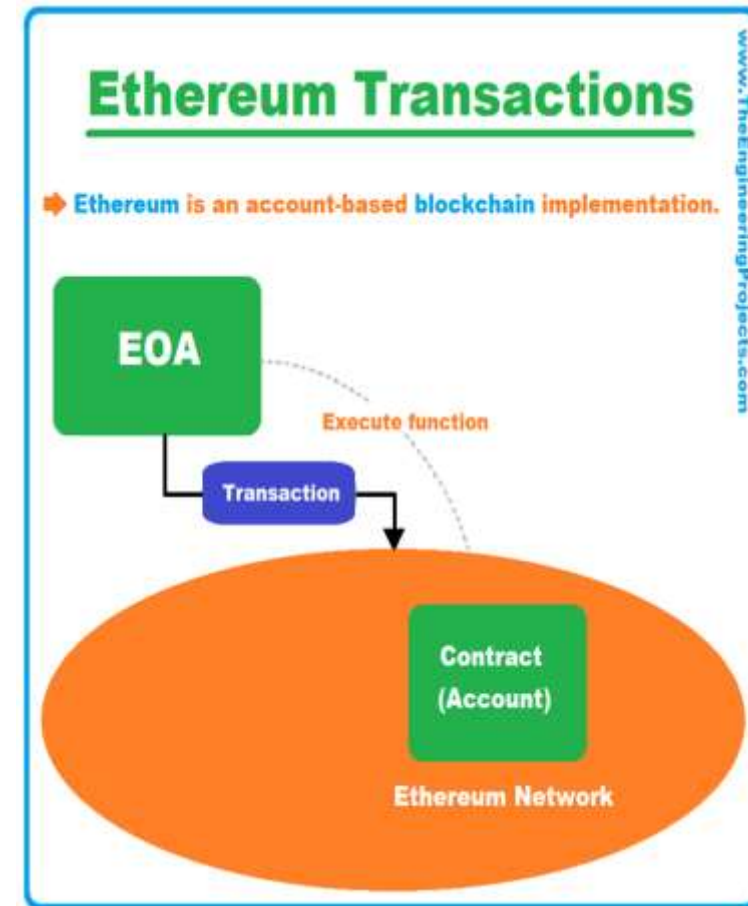


## 3.2.3 Working of Ethereum

- Ethereum uses accounts to store the **ether**, analogous to bank accounts.

There are **two types of accounts** to know:


- **Externally owned accounts (EOAs):** The accounts that normal users use for holding and sending **ether**.
- **Contract Accounts(CA):** These separate accounts are the ones that hold smart contracts(contract codes), which can be triggered by ether transactions from EOAs or other events.



## EOAs

- ▶ Have Ether Balance
- ▶ Capable of sending transactions
- ▶ Have No associated code
- ▶ Contain key-value store
- ▶ Controlled by private keys

## CAs

- ▶ Have Ether Balance
  - ▶ Associated with code that is kept in Memory/storage on the blockchain.
  - ▶ Trigger and Execute code in response to a transaction.
  - ▶ Contain key-value store
- 

## Steps involved in working of Ethereum:

1. Firstly either user requests money by sending request to the sender, or the sender decides to send money to the receiver.

For example, If User Alice requests money from John then she can send a request to John using QR code/Ethereum address which can be shared via email or any communication methods.

Once John receives the request, he will either scan the QR code or manually type Alice's Ethereum address and send ether to Alice's address.



2. Once John receives the request he will either scan this QR code or copy the Ethereum address in Ethereum wallet software and initiate a transaction.

3. Once the transaction request of sending money is constructed in the wallet software, it is broadcasted to the Ethereum network.

Transaction is digitally signed by the sender as the proof that he's the owner of the Ether,

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



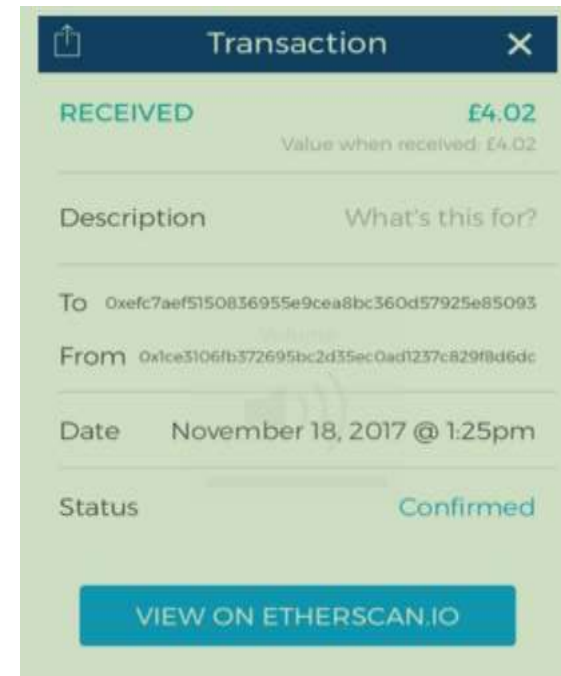
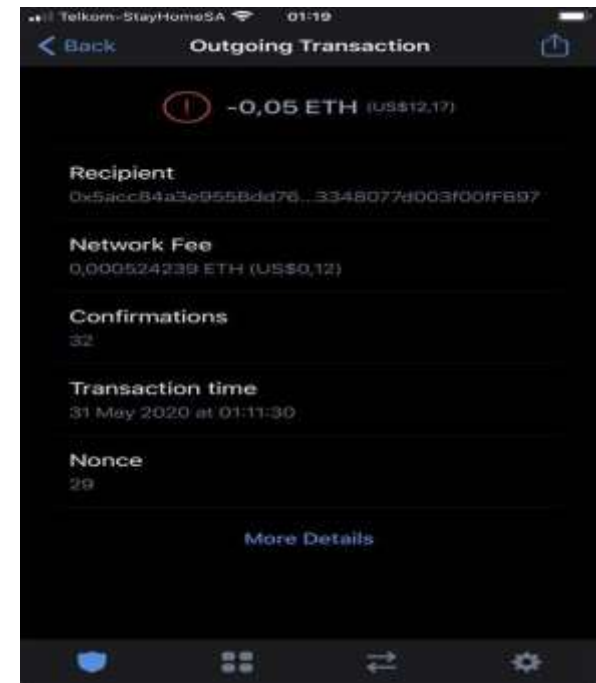


5. Once it is verified and included in the block, POW process starts.

6. 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.

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

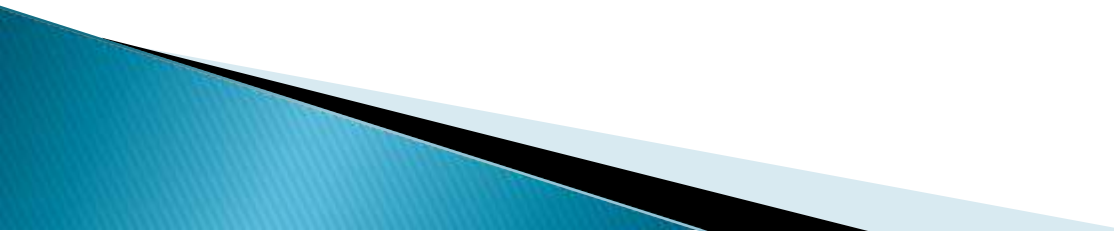
8. Finally John gets Ether and it is shown in his wallet software.



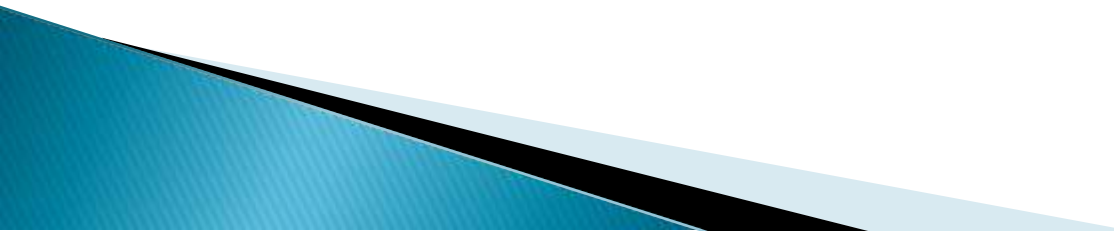
## 3.2.4 Ethereum Clients

Ethereum Client is a software application that implements Ethereum specification and communicates over the peer to peer network with other Ethereum clients.

Some of the Ethereum Clients:

- ▶ Geth (Go)
  - ▶ OpenEthereum (Rust)
  - ▶ Nethermind (C#, .NET)
  - ▶ Besu (Java)
  - ▶ Erigon (Go/Multi)
- 

## 3.2.5 Ethereum Key Pairs

- ▶ Ethereum utilizes key pairs as a fundamental component of its cryptographic security.
  - ▶ A key pair consists of a **private key** and a **public key**.
  - ▶ The private key is kept secret and is used to sign transactions, while the public key is shared with others to verify the authenticity of the transactions ensuring that the transaction hasn't been tampered with.
  - ▶ Ethereum employs various cryptographic techniques such as hash functions and signature schemes to ensure the integrity and security of transactions.
- 

- ▶ Private key is generated randomly or through 256 bit hash algorithms such as Keccak 256 or SHA-256.

- ▶ For calculating public key, private key is used along with the **Elliptic curve multiplication**:  $K = Pr * G$ ,

where  $Pr$  is a Private key,  $G$  is a generator point to get public Key  $K$ ,  $*$  is an Elliptic curve multiplication.

$G + G + G + \dots + G$  added  $Pr$  times

- ▶ Elliptic curve multiplication is a "one-way" function used in cryptography.
- ▶ The owner of the private key can easily create the shareable public key, knowing that no one can reverse it to calculate the private key.

## Private key:

Pr=b51928c22782e97cca95c490eb958b06fab7a70b9512c38c3  
6974f47b954ffc4

K=

**b51928c22782e97cca95c490eb958b06fab7a70b9512c38c3697  
4f47b954ffc4** \*G(x, y)

x=

3aa5b8eefd12bdc2d26f1ae348e5f383480877bda6f9e1a47f6a  
4afb35cf998ab

y=847f1e3948b1173622dafc6b4ac198c97b18fe1d79f90c9093  
ab2ff9ad99260

**Resulting public Key, K=**

04+x-coordinate(32bytes/64hex)+y-coordinate(32bytes/64hex)

## Public key:

043aa5b8eefd12bdc2d26f1ae348e5f383480877bda6f9e1a47f  
6a4afb35cf998ab847f1e3948b1173622dafc6b4ac198c97b18f  
e1d79f90c9093ab2ff9ad99260

## 3.2.6 Ethereum Addresses

- ▶ Address on the Ethereum platform are unique identifiers.
- ▶ An Ethereum address is a **42-character hexadecimal address** derived from the last 20 bytes ( $20 \times \text{bytes} = 160 \text{ bytes}$ ) of the Keccak hash of the public key with 0x appended in front.

**Address: 0x77b4b5699827c5c49f73bd16fd5ce3d828c36f32**

- ▶ The Inter Exchange Client Address Protocol (ICAP) is a method for encoding Ethereum addresses that offers convenience, error-checking, and compatibility with the International Bank Account Number (IBAN) format.
- ▶ Similar to IBAN, ICAP uses alphanumeric characters (letters and numbers), with a maximum length of 34 characters.
- ▶ It introduces a special country code, "XE," which stands for "Ethereum." ICAP includes a two-character checksum and three variations of an account identifier.

## ICAP Variations of an account identifier:

1. **Direct:** This encoding represents the 155 least significant bits of an Ethereum address. It can handle addresses starting with zero bytes

Example: XE60HAMICDXSV5QXVJA7TJW47Q9CHWKJD (33 characters).

2. **Basic:** Similar to the Direct encoding, except it is 31 characters long. but allows representation of any Ethereum address. Example: XE18CHDJBPLTBCJ03FE9O2NS0BPOJVQCU2P (35 characters).

3. **Indirect:** This variation encodes an identifier that links to an Ethereum address through a name registry provider. It includes an asset identifier, a name service, and a 9 character human-readable name. Example: XE##ETHXREGKITTYCATS (20 characters), where the "##" should be replaced by the computed checksum characters.

## 3.2.5 Ethereum Wallets

- ▶ Ethereum wallets are applications that give you control over your account.
- ▶ Wallets hold keys(Public & Private key pairs) while Ether or other tokens are recorded on the Ethereum block chain.
- ▶ By signing transactions with private key present in their wallets, users can control tokens on the network.

There are two primary types of wallets:

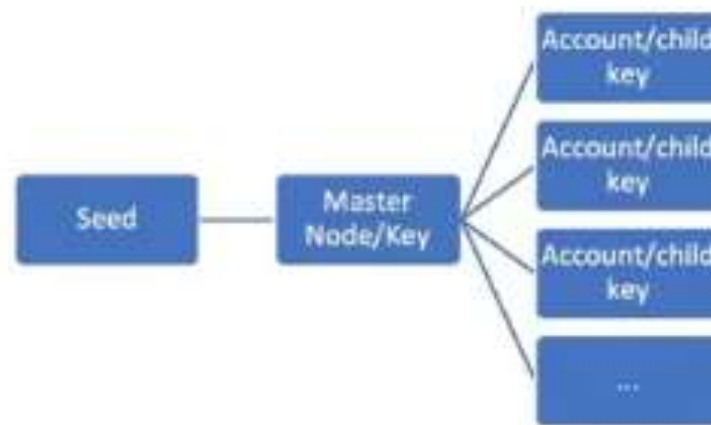
- ▶ Non deterministic wallet
  - ▶ Deterministic wallet
- 



- ▶ A **non-deterministic wallet** generates private keys that are random and independent of each other.
- ▶ There is no particular pattern as to how the keys are derived and hence we need to create a backup of the keys each time there is a new one.
- ▶ They are called **Just a Bunch of Keys (JBOK)** wallet.



- ▶ In a **deterministic wallet**, on the contrary, the private keys are related because they originate from the same key called seed.
- ▶ Just backing up the seed once will be enough to regenerate all the keys.
- ▶ As only the seed is needed to gain access to the entire wallet.



- ▶ Ethereum clients use a keystore file that is a JSON encoded file containing a randomly generated private key and encrypted by a passphrase for extra security.

<https://www.myetherwallet.com/>

MyEtherWallet 3.31.1 English Gas Price: 41 Gwei Network: ETH (myetherwallet.com)

New Wallet Send Ether & Tokens Swap Send Offline Contracts ENS DomainSale Check TX Status View Wallet Info Help

## Create New Wallet

Enter a password

Make-it-strong

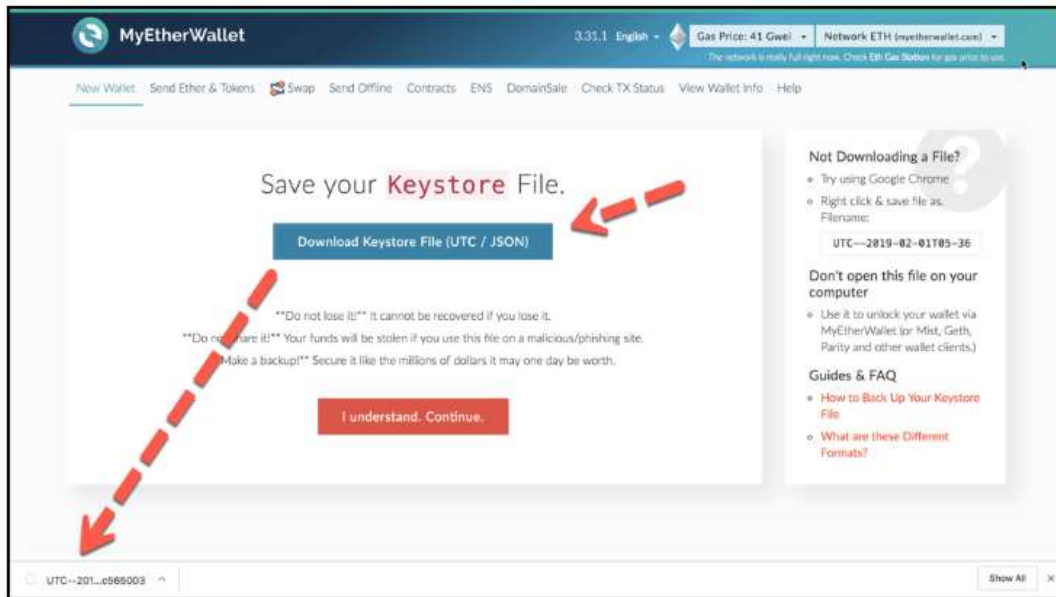
Create New Wallet

This password encrypts your private key. This does not act as a seed to generate your keys. You will need this password + your private key to unlock your wallet.

[How to Create a Wallet](#) • [Getting Started](#)

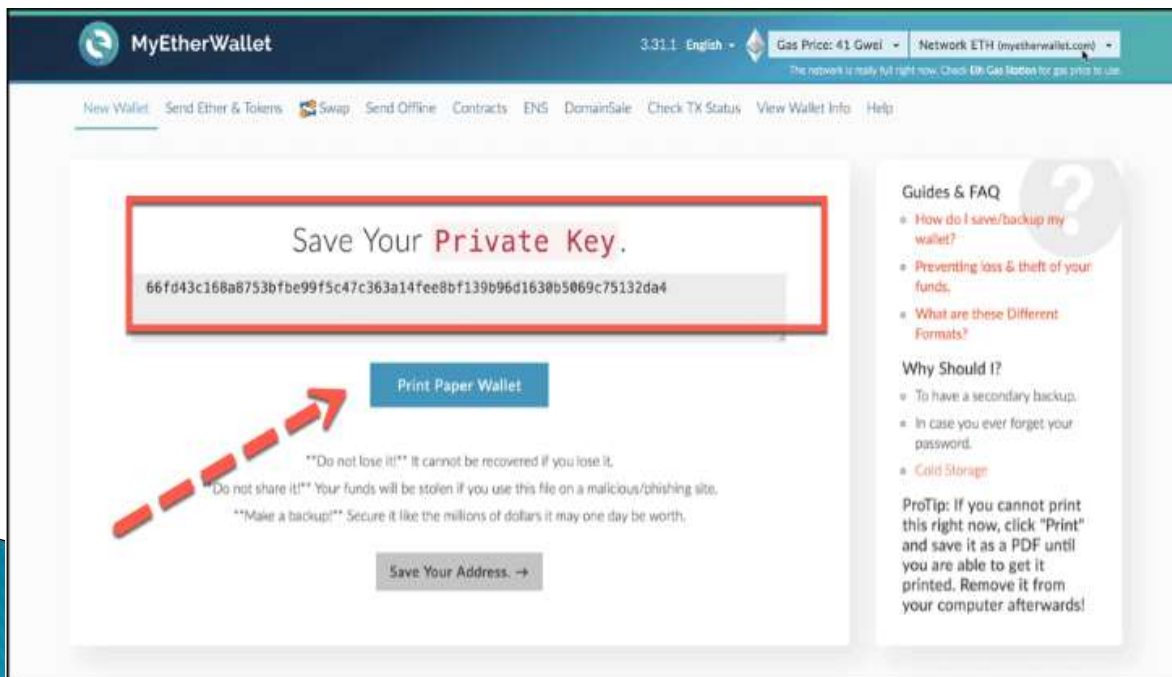
Already have a wallet somewhere?

- Ledger / TREZOR / BitBox / Secalot : Use your **hardware wallet** . Your device is your wallet.
- MetaMask Connect via your **MetaMask Extension** . So easy! Keys stay in MetaMask, not on a phishing site! Try it today.
- Jaxx / ImToken Use your **Mnemonic Phrase** to access your account.
- Mist / Geth / Parity: Use your **Keystore File (UTC / JSON)** to access your account.



➤ A keystore file (sometimes called a UTC file) in Ethereum is an encrypted version of your private key.

➤ They are generated using your private key and a password that you use to encrypt it.





My Ether Wallet

www.MyEtherWallet.com



YOUR ADDRESS



AMOUNT / NOTES



YOUR PRIVATE KEY

Your Address:

0x97289e0C5c0027BC6D8031A8494063e65c565003



Always look for this icon  
when sending to this wallet.

Your Private Key:

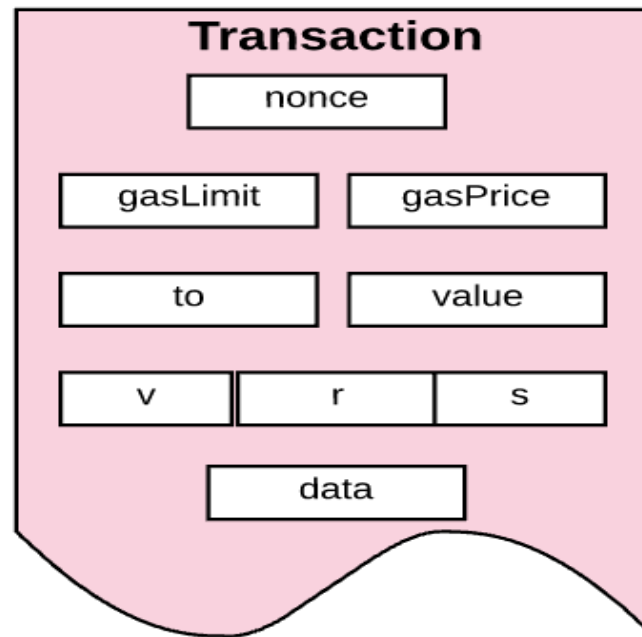
66fd43c168a8753bfbe99f5c47c363a14fee8bf139b96d1630b5069c75132da4

POST /v2/eth/wallets/{walletId}/transfer

```
1 curl localhost:3000/v2/eth/wallets/{walletId}/transfer \
2   -X POST \
3   -H 'X-Henesis-Secret: API_SECRET' \
4   -H 'Authorization: Bearer ACCESS_TOKEN' \
5   -H 'Content-Type: application/json' \
6   -d '{
7     "ticker": "ETH",
8     "to": "0x1f9840a85d5af5bf1d1762f925bdaddc4201f984",
9     "amount": "0xDBE16A831",
10    "passphrase": "PASSWORD"
11  }'
```

## 3.2.6 Ethereum Transactions

- Transaction is an activity that can trigger a change of state or causes a contract to execute in the EVM.
- After receiving a transaction, all the applications and client will store it in-memory using its own internal data structure.



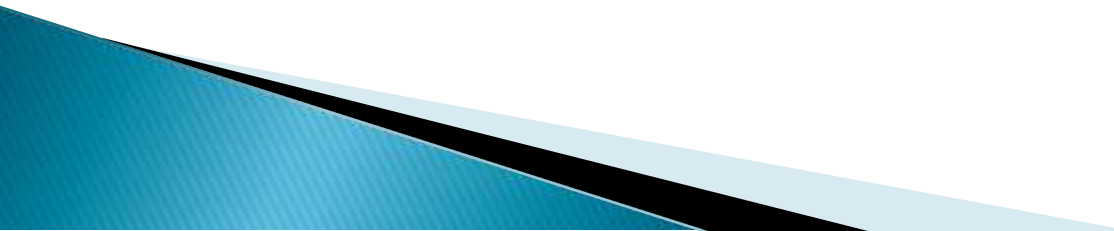
**Structure of the Transaction**

- ▶ **An Ethereum transaction consists of following fields:**
- ▶ **Nonce:** sequentially incrementing counter which indicates the transaction number from the account preventing replay attacks. Counted dynamically based on confirmed transactions.
- ▶ **gas Price** – price of the gas, the originator is willing to pay.
- ▶ **gas Limit**– Maximum amount of gas units the originator is willing to buy for a transaction.
- ▶ **Value:** amount of ether to send to the receiver.
- ▶ **To:** it is 20 byte recipient Ethereum address
- ▶ **sender's signature** (with 3 components)
- ▶ An optional **data** field, includes additional instructions (invocation)

Transaction can have

- only value or
- only data or
- both value and data or
- neither value nor data : valid but waste of gas.

# What is Gas in Ethereum?

- ▶ Gas is the fee required to successfully conduct a transaction or execute a contract on the Ethereum blockchain platform.
  - ▶ Fees are priced in tiny fractions of the cryptocurrency ether (ETH)—denominations called wei ( $10^{-8}$  ETH).
  - ▶ Wei is the smallest denomination of ether, named after Wei Dai, a Computer Scientist.
  - ▶ A gas fee is a blockchain transaction fee, paid to network validators for their services to the blockchain.
- 



## Example of transaction to send 1000 wei (1 ether= $10^8$ wei)

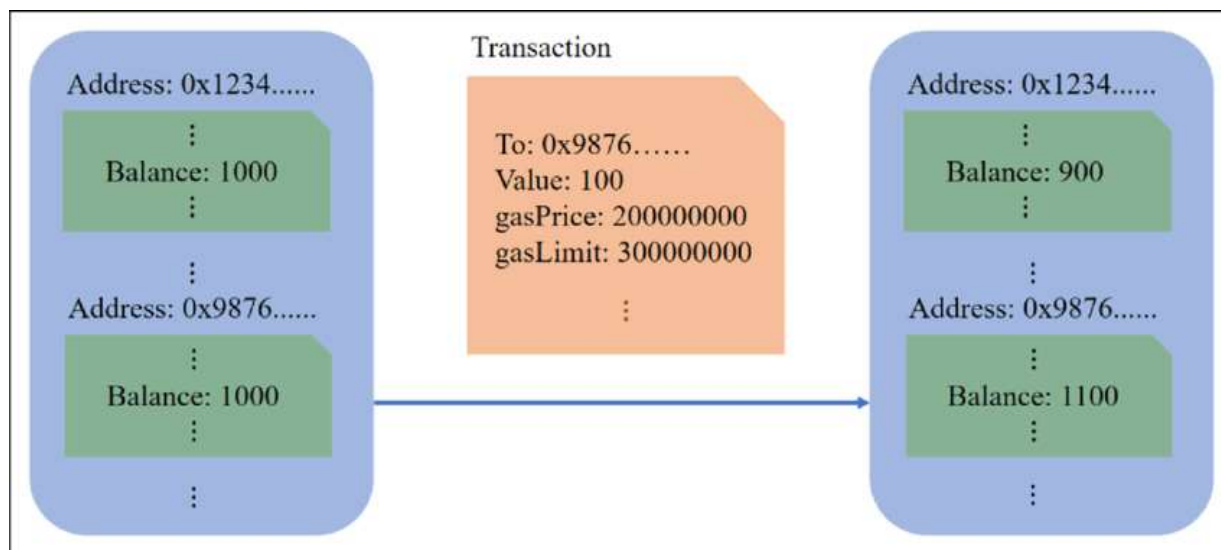
```
signedTransaction = {  
  nonce: web3.toHex(0),  
  gasPrice: web3.toHex(200000000000),  
  gasLimit: web3.toHex(100000),  
  to: '0x687422eEA2cB73B5d3e242bA5456b782919AFc85',  
  value: web3.toHex(1000),  
  data: '0xc0de',  
  v: '0x1c',  
  r: '0x668ed6500efd75df7cb9c9b9d8152292a75453ec2d11030b0eec42f6a7ace602',  
  s: '0x3efcbbf4d53e0dfa4fde5c6d9a73221418652abc66dff7fddd78b81cc28b9fbf'  
};
```

- Gas is the fuel of Ethereum as a separate virtual currency , with its own exchange rate against ether.
- Ethereum uses gas to control the amount of resources a transaction can use.
- gasLimit provides the maximum number of units of gas that the originator of transaction is willing to buy inorder to complete the transaction.

# Transaction Lifecycle

The transaction lifecycle can be simplified to:

1. An external account creates a transaction object.
2. The account sending the transaction verifies it by signing which creates a transaction hash
3. The transaction is broadcasted across the network using an Ethereum node.
4. The transaction execution is idle until the transaction is mined and added to the block or replaced/canceled.



## 3.2.7 Ethereum Languages

Two primary languages used to write smart contracts are **Solidity** and **Serpent**; both allow developing smart contracts and compiling into EVM byte code.

- ▶ **Solidity** is similar to high level object oriented language with similarities to Java and C while Serpent is similar to Python.
- ▶ Solidity recently released the 0.8.x version that introduced a lot of breaking changes.
- ▶ Solidity is a curly-bracket language designed to target the Ethereum Virtual Machine (EVM).

```
pragma solidity >=0.4.16 <0.9.0;
contract SimpleStorage
{
    uint storedData;
    function set(uint x) public
    {
        storedData = x;
    }
    function get() public view returns (uint)
    {
        return storedData;
    }
}
```

Second line specifies that the source code is written for Solidity version 0.4.16, or a newer version of the language up to, but not including version 0.9.0. This is to ensure that the contract is not compilable with a new (breaking) compiler version, where it could behave differently

- ▶ **Serpent** is a programming language inspired by Python. Like Python, it has a simple, minimal syntax, dynamic typing, and support for object-oriented programming.
- ▶ But it is no longer supported by Blockchain community currently.



## 3.2.8 Ethereum Development Tools

Ethereum development tools are software applications that help developers create and deploy decentralized applications (dApps) on the Ethereum blockchain.

### Five Categories:

#### 1. Integrated Development Environment(IDE):

It Provides various functionalities such as editing compiling and debugging at our fingertips, which is carried out in IDE.

#### **Examples:**

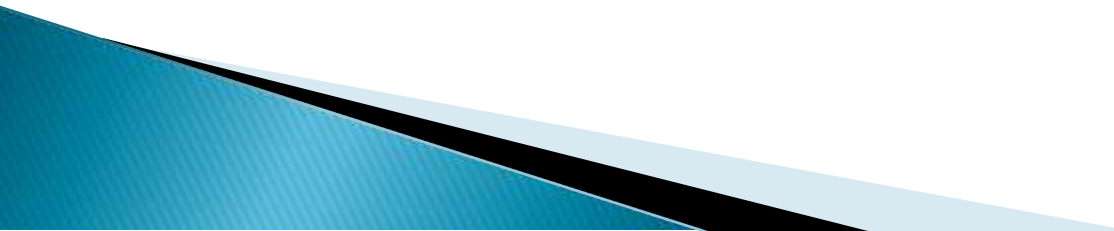
##### **1.a)Remix IDE:**

- ▶ Remix is considered the industry standard for IDEs by many blockchain developers.
- ▶ Remix IDE is written in JavaScript and you can use it from any browser, although you can also run it locally on your computer (as a desktop application).
- ▶ It offers a comprehensive suite of libraries, plugins, and other features to supercharge smart contract development.

## 1.b) Ethfiddle:

- ▶ EthFiddle is a browser-based IDE for writing and debugging Solidity code.
- ▶ It offers well-designed testing and prototyping capabilities, making it a good tool for any blockchain developer.


## 2. Test Nodes

- ▶ Ethereum development tools provide test network nodes to be used as local test nodes to test the interaction of the contracts.
  - ▶ Therefore smart contracts can not be tested on the main blockchain network.
  - ▶ Therefore one needs to have a environment where prior testing can be done.
- 

## **2.a) Ganache**

- ▶ Ganache is a local blockchain tool used to develop and test Ethereum applications.
- ▶ It is an in-memory blockchain server that emulates a full Ethereum client and allows developers to test and debug smart contracts without requiring a real blockchain.
- ▶ Ganache includes an interactive graphical user interface that allows developers to quickly create and deploy new contracts, test their code, and inspect state and transaction data.

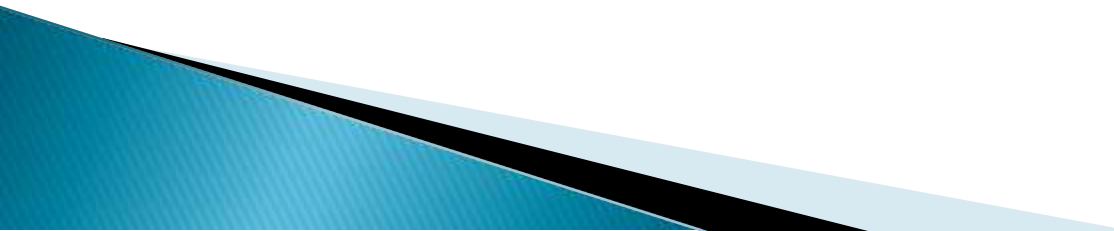
## **2.b) Pythereum**

- ▶ Its a lightweight Ethereum RPC library for Python; not too rich in features.
  - ▶ It can send a transaction using the key private key to the given address with the given value and data.
- 

### 3. Command Line Tool

- ▶ Also known as Command Line Interface (CLI) is a type of human computer Interface that relies on textual request and response transaction process.
- ▶ It can be used in the form of batch files or lists.

#### 3.a) Truffle


- ▶ It provides developers with tools, including a smart contract compiler, automated testing, and deployment scripts.
  - ▶ With Truffle, developers can efficiently write and manage smart contracts using the Solidity programming language.
  - ▶ It simplifies the development process by offering features like contract migration, network management, and debugging.
- 



### 3.b) Embark

- ▶ Embark is a framework used for developing and deploying DApps (Decentralized Apps) using one or more decentralized technologies.
- ▶ Embark currently integrates with Ethereum, decentralized storages like IPFS, and decentralized communication platforms like Whisper and Orbit.
- ▶ The dashboard of Embark will provide the tracking information like the current state of the contracts, the environment it is using and what Embark is doing currently etc.

### 4. Code Analysis Tools


- Code analysis tools for Solidity, on the Ethereum blockchain, play a crucial role in identifying potential security flaws, bugs, and design flaws in smart contract code.
  - These tools help developers ensure the security, reliability, and overall quality of their smart contracts, mitigating risks and protecting user funds.
- 

OPenZeppelin is a solidity Framework for writing secure smart contracts to build applications, protocols. It integrates with Truffle.

## 5. Browsers

Blockchain browsers are web browsers that support web 3.0 technologies and allow users to connect freely with the decentralized economic and non-economic spheres.

### 5.a) Mist

- ▶ The Mist browser, also referred to as the Ethereum dApp Browser, served as an interface designed to facilitate user access to the diverse dApps available on the Ethereum network.
  - ▶ The Mist Ethereum wallet was a notable feature that operated locally on a user's computer. This marked the first instance of a desktop crypto wallet with a GUI, making cryptocurrency management more accessible than ever before.
- 

## 5.b) MetaMask

- ▶ It turns Google Chrome browser into Ethereum browser to fetch data from Blockchain and allow users to securely send and receive signed transactions.
  - ▶ The extension injects the Ethereum web3 API into every website's Javascript context, so that Dapps can read from the blockchain.
  - ▶ MetaMask also lets the user create and manage their own identities (via private keys, local client wallet and hardware wallets like Trezor), so when a Dapp wants to perform a transaction and write to the blockchain, the user gets a secure interface to review the transaction, before approving or rejecting it.
- 