
BLOCKCHAIN TECHNOLOGY

COMPUTER ENGINEERING

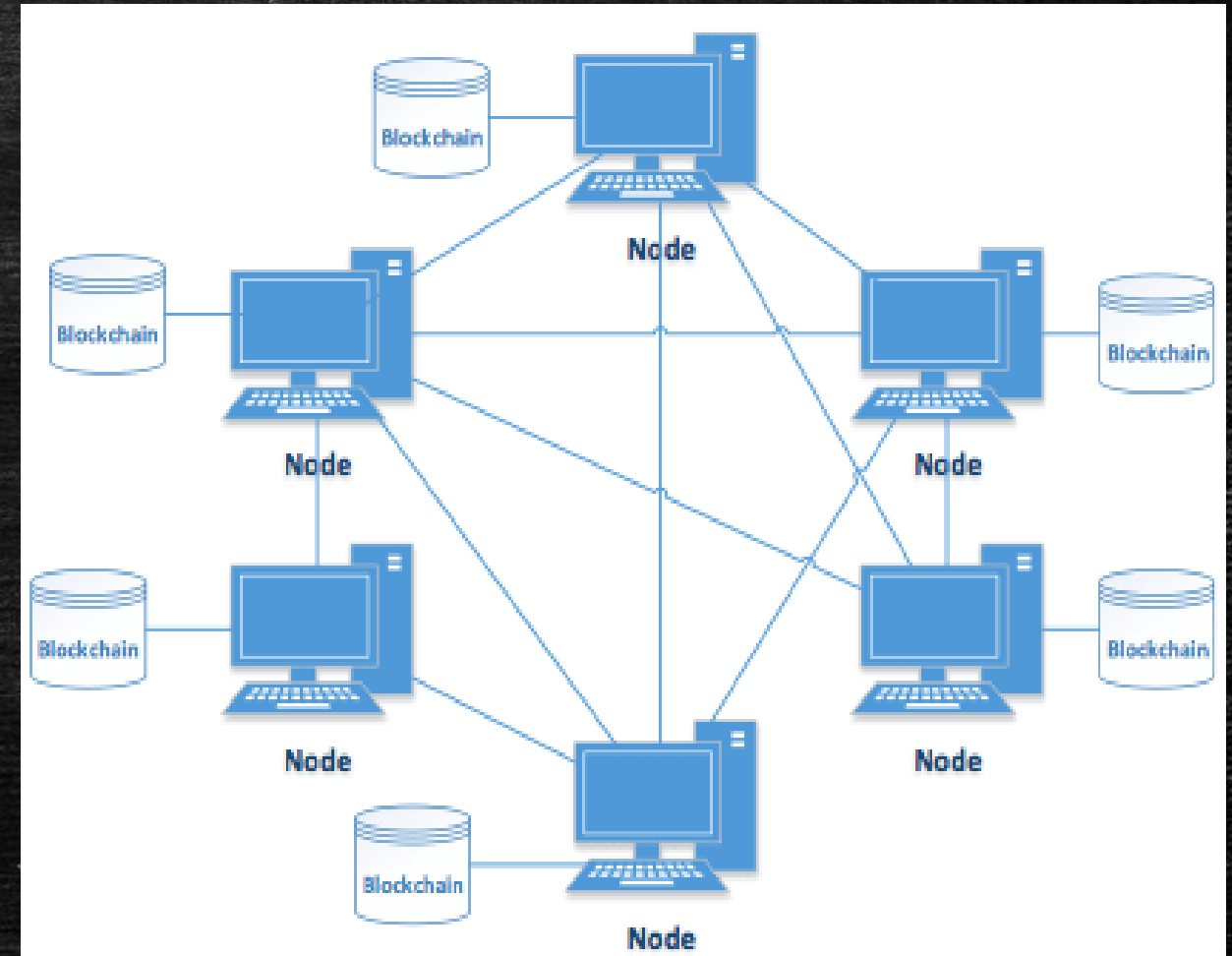
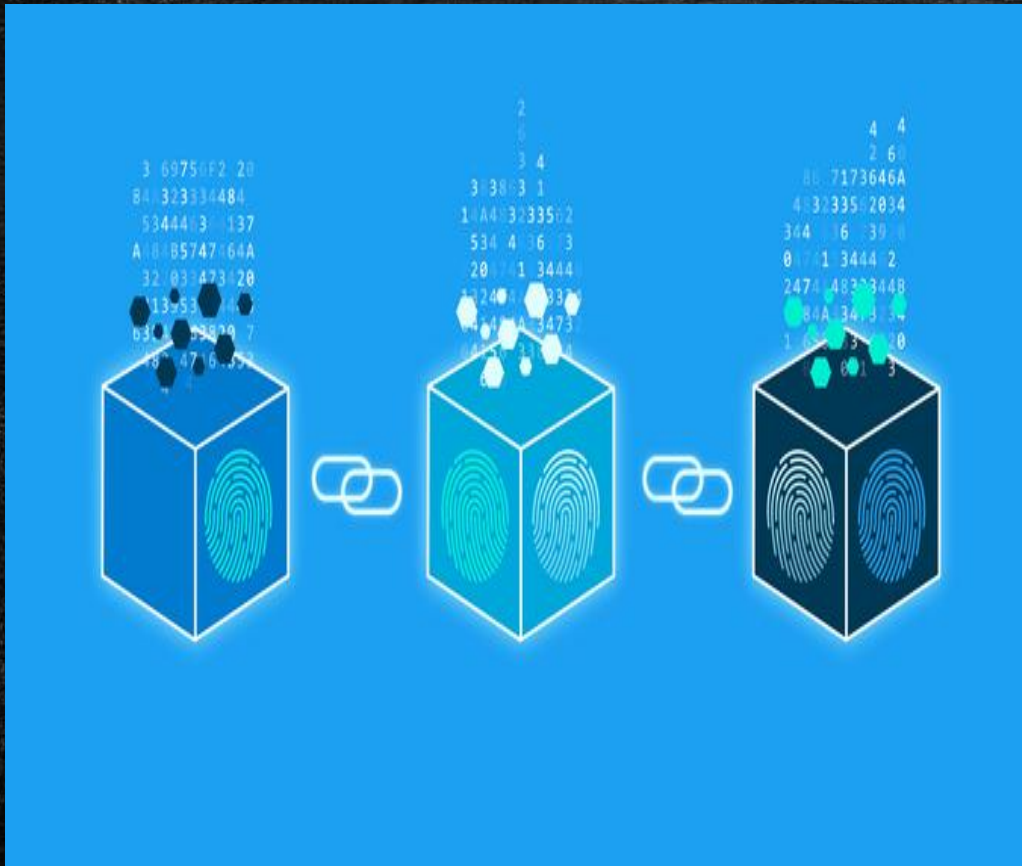
BE SEMESTER VII

PROF. SWAPNIL SONAWANE

MODULE 1:

INTRODUCTION TO BLOCKCHAIN

BLOCKCHAIN STRUCTURE



DEFINITION OF BLOCKCHAIN

Blockchain is defined as a distributed, replicated and peer-to-peer network of databases that allows multiple non-trusting parties to transact without a trusted intermediary and maintains an ever-growing, append-only, temper-resistant list of time-sequenced records.

BASIC OPERATIONS ON BLOCKCHAIN

- ✓ Broadcasting transactions
- ✓ Validation of transactions
- ✓ Gathering transactions for a block
- ✓ Consensus on new block creation
- ✓ Chaining blocks

HOW BLOCKCHAIN TECHNOLOGY WORKS

1. A transaction is initiated
2. A transaction is broadcasted to peer-to-peer network
3. A network of node validate transaction
4. A verified transaction can involve cryptocurrency, contact, record or any transaction
5. A transaction is combined with other transactions and keep in a block
6. A miner uses consensus algorithm to add new block in blockchain
7. A transaction is now called as finished and hence immutable

COMPONENTS OF BLOCKCHAIN

1. Node:

A node is an electronic device (computer or mobile device) that are connected to the internet.

In blockchain, any computer or hardware device that is connected to blockchain network is called as a node.

All the nodes in the network have a copy of blockchain ledger and are interconnected

A node can be:

A. Full Node:

A node maintains a full copy of the transaction history of the blockchain. They also help the network by processing and accepting transactions or blocks, validating those transactions or blocks and then broadcasting them to the network.

B. Partial/ Lightweight/ Light Node:

It maintains only partial copy of a blockchain and normally used when user do not have sufficient disk space for full blockchain. They only download the block headers to see transactions.

2. Ledger:

It is digital database of information that is immutable

Its properties are:

- a. **Ledger is public** as anyone can access the ledger and can read or verify the transactions
- b. **Ledger is distributed** because all the nodes in the blockchain have a copy of blockchain ledger
- c. **Ledger is decentralized** so no node have excessive control over ledger and hence no single point of failure

3. Wallet:

It is a digital wallet that allows user to manage cryptocurrency like bitcoin, ether etc. One can send and receive cryptocurrency using wallet

Its features are:

- a. **Privacy is maintained:** When wallet is created, user's public and private key also generated. We need to share public key to receive funds
- b. **Transactions are secured:** Private key is used to send funds as well as open encrypted message
- c. **Ease of usage:** Wallets can be installed and accessed from web, desktop or mobile device without intermediary like banks
- d. **Currency conversion:** We can transact over various types of cryptocurrencies like BTC, ETH, LTC etc.

4. Nonce:

It is an arbitrary number generated randomly that can be used while block is added. It is a key for creating block in blockchain

5. Hash:

A hash function can take data of any size, perform an operation on it and return a “hash” of that data of fixed size. It has following characteristics:

- a. It creates unique hash for data
- b. It is one dimensional, means we can not recreate data from its hash
- c. A very minute change in data gives different hash
- d. It keeps database small

6. Mining:

It is the mechanism, where miners or forgers validate new transactions and add them to the blockchain ledger.

It is done by solving complex cryptographic hash puzzles to verify blocks of transactions that are updated on the decentralized blockchain ledger.

Solving these puzzles requires powerful computing power and sophisticated equipment. In return, miners are rewarded with bitcoins.

7. Consensus protocol:

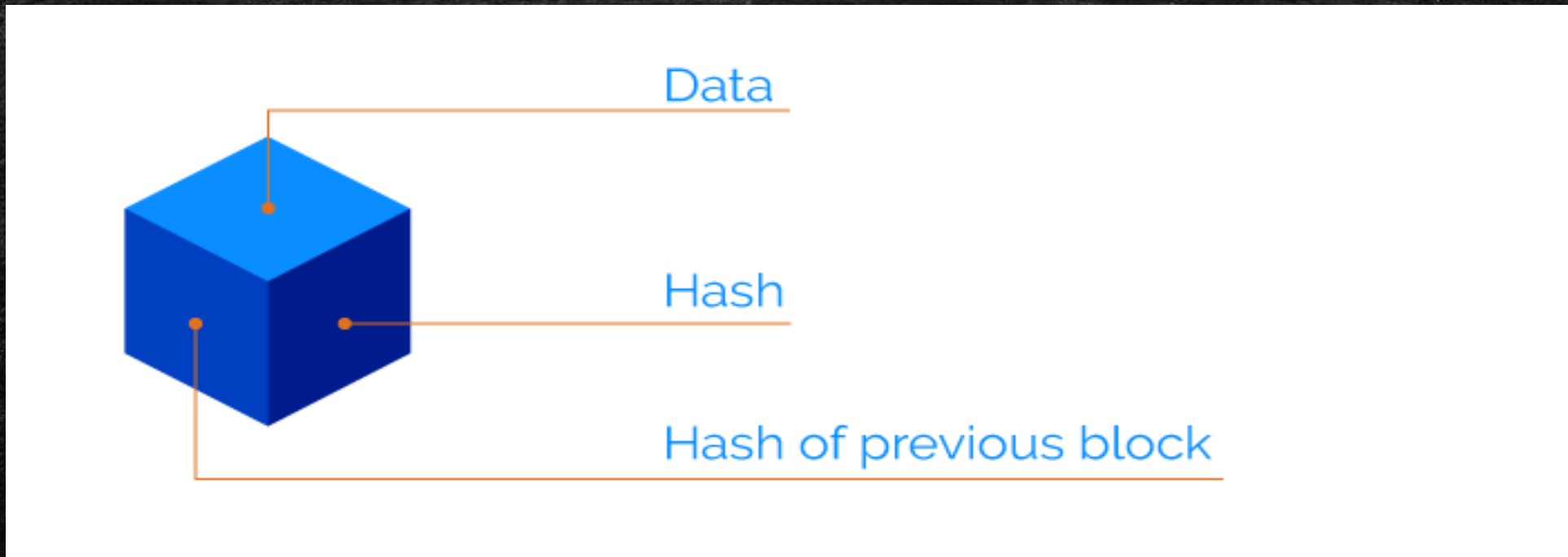
Consensus for blockchain is a procedure in which the peers of a Blockchain network reach agreement about the present state of the data in the network.

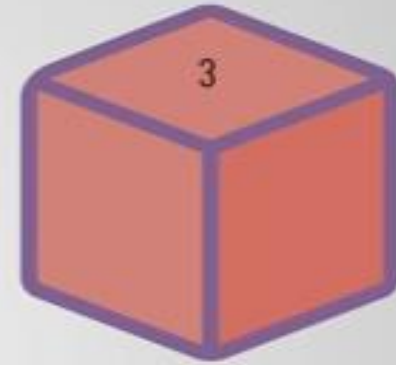
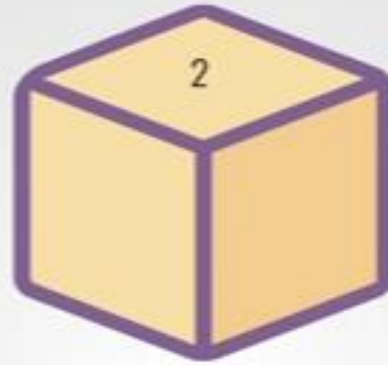
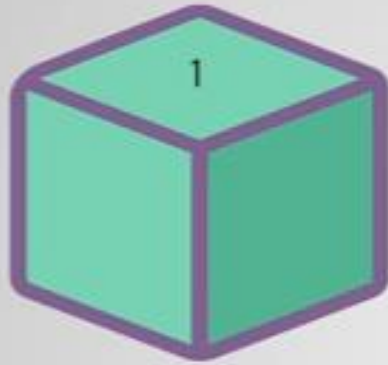
Through this, consensus algorithms establish reliability and trust in the Blockchain network.

Example: Proof-of-work, Proof-of-stake etc.

BLOCK IN BLOCKCHAIN

Each block containing the data, its own hash value and a pointer to the hash of the previous block





Hash: **1Z8F**

Previous hash: **0000**

Hash: **6BQ1**

Previous hash: **1Z8F**

Hash: **3H4Q**

Previous hash: **6BQ1**

A block is a place in a blockchain where information is stored and encrypted.

Blocks are identified by long numbers that include encrypted transaction information from previous blocks and new transaction information.

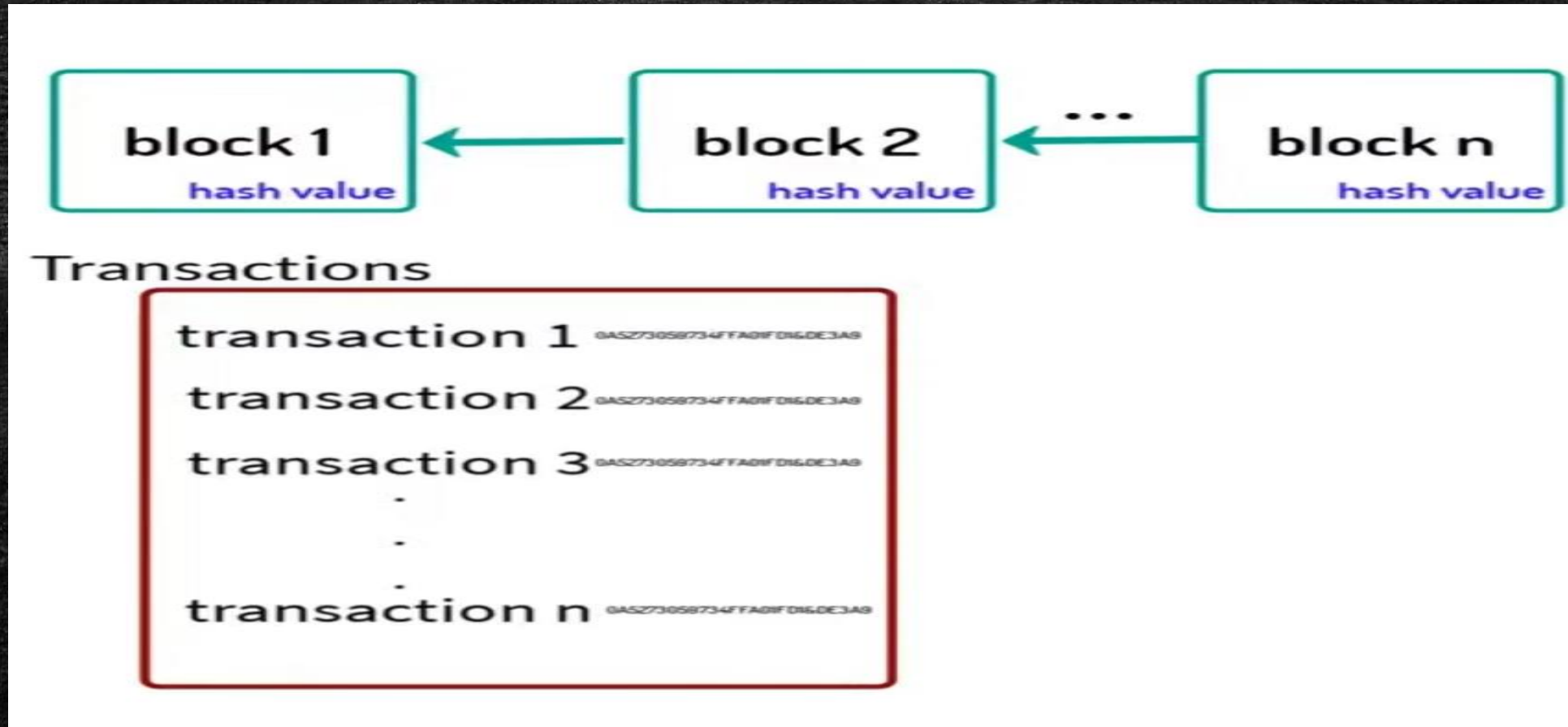
Blocks and the information within them must be verified by a network before new blocks can be created.

Blocks and blockchains are not used solely by cryptocurrencies. They also have many other uses.

The header of the block is divided into six components:

1. The version number of the software
2. The hash of the previous block
3. The root hash of the Merkle tree
4. The time in seconds since 1970–01–01 00:00 UTC
5. The goal of the current difficulty
6. The nonce

MERKLE TREE

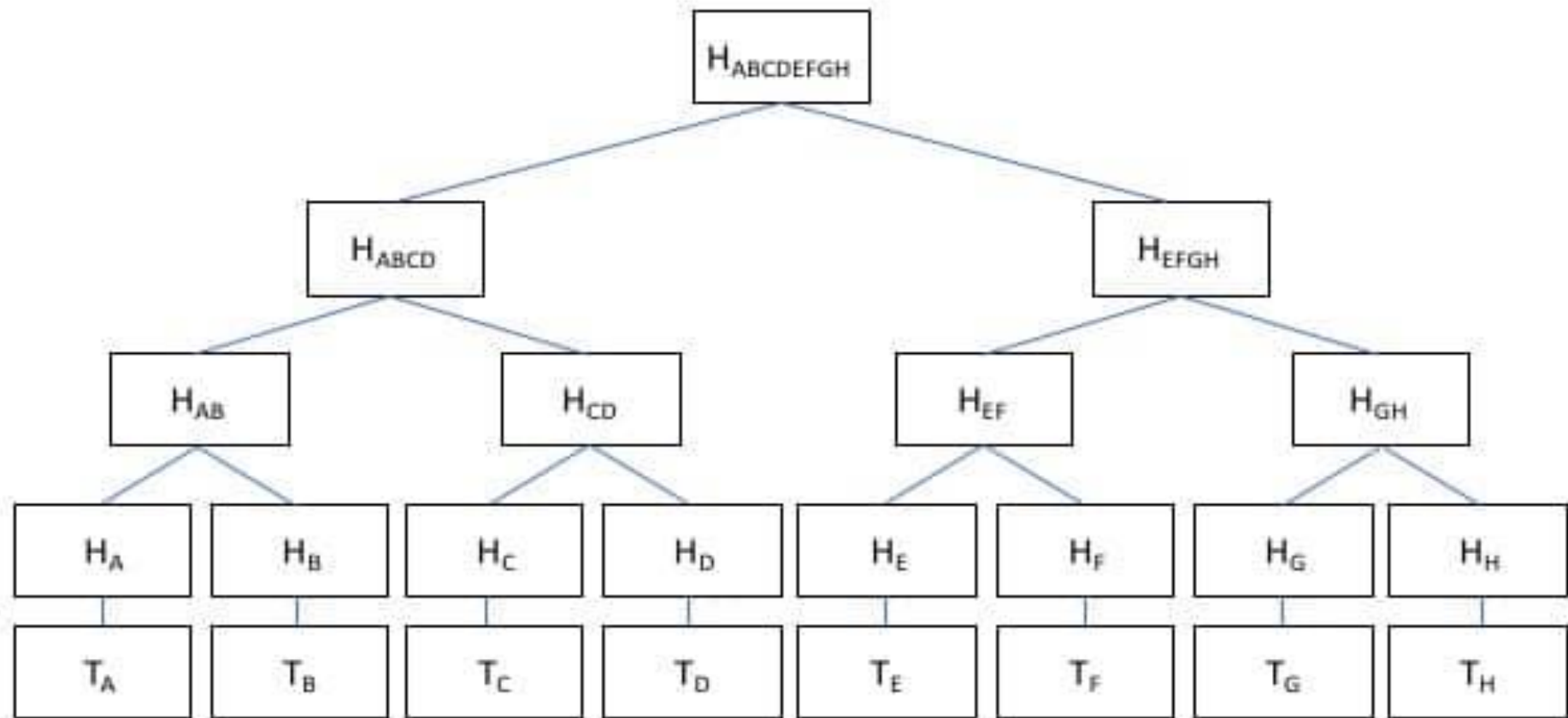


Merkle trees, also known as Binary hash trees, are a prevalent sort of data structure in computer science.

In bitcoin and other cryptocurrencies, they're used to encrypt blockchain data more efficiently and securely.

It's a mathematical data structure made up of hashes of various data blocks that summarize all the transactions in a block.

It also enables quick and secure content verification across big datasets and verifies the consistency and content of the data.



TYPES OF BLOCKCHAIN

Public Blockchain:

It is public permissionless blockchain

In public blockchain, anyone in the world can access the blockchain, download a copy and run the node.

One does not need any permission to read/access a transaction, initiate the transaction or participate in the consensus process to create a block.

Features of public blockchain:

1. Anyone can join the network and be the participant
2. No permissions are required for anyone to read/send transactions
3. The standard consensus algorithm used is Proof-of-Work (PoW) where nodes (miners) solve the hash puzzle and submit their resultant block to the rest of the network participants for consensus
4. High cryptographic methods are used to secure data
5. It has low transaction processing speed
6. Consensus mechanism requires an immense amount of energy and computation power

Private Blockchain:

It is private permissioned blockchain

The network is not open to anyone.

Features of decentralization and openness is lost as all the permissions are controlled by few nodes in organization

Here owner has sole control over who can read, write or validate the data, it stands to reason why many may not consider it to be a real blockchain

Features of private blockchain:

1. It is not open to public
2. All participants are pre-approved by the organization
3. The owner or central authority controls the permission to read, write or audit the ledger
4. It has high transaction processing speed
5. A central authority means single point of failure
6. The organization must agree on who has the highest power to be the central authority.

Consortium Blockchain:




It is also known as Federated blockchain

It is permissioned blockchain and considered to be hybrid between public and private blockchain

It is a distributed ledger that anyone can download, or access and the consensus process is not controlled by one company but by the predetermined consortium of companies or representative individuals.

Features of consortium blockchain:

1. Any member node can initiate and receive transactions but permission to audit the ledger is done by pre-approved individuals only
2. They are faster as compared with public blockchain
3. They are not fully decentralized
4. Agreement on a standard set of rules may get challenging

ORGANIZATION TYPE	PUBLIC BLOCKCHAIN	PRIVATE BLOCKCHAIN	CONSORTIUM BLOCKCHAIN
	 Public	 Single entity or organization	 Multiple organizations or enterprise
COMMON FEATURES	- Chain of blocks - Peer to peer architecture - Public-key cryptography - Immutable - Byzantine fault tolerance - Auditable -		
USERS	Anonymous; but web tracking and cookies pose a risk to privacy	Known & trusted participants	Known & trusted participants
ACCESS	open and transparent to all	Access fully restricted	selectively open; relevant transparency provided
NETWORK TYPE	Decentralized; zero points of failure	Centralized; single point of failure	partially decentralized; multiple points of failure
OPERATION	Anyone can read or initiate or receive transactions	Pre-approved participants can read &/or initiate transactions	Pre-approved participants can read &/or initiate transactions
VERIFICATION	Anyone can be a node and take part in the consensus process to validate transactions and create a block	Single validator node or central authority to create a block	Only privileged members of the consortium can validate and create a block
IMMUTABILITY	Secured by hashing	Secured by distributed consensus	Secured by distributed consensus
CONSENSUS MECHANISM	PoW, PoS, etc.	Voting or variations of PoW/PoS consensus algorithms	Voting or variations of PoW/PoS consensus algorithms
INCENTIVIZATION	Incentivizes miners to grow the network	Users limited to within a company; hence incentivization is not relevant	Limited incentivization
SECURITY	Security based on consensus protocols and hash functions. Higher the security, lower the performance	Security is dependent on the blockchain architecture adopted	Security is dependent on the blockchain architecture adopted
TRUST	Trust-free system; trust is enforced via cryptographic proof	Trusted; central control	Trusted; need to trust the majority

CONSENSUS ALGORITHMS

Proof of Work:

Proof of Work(PoW) is the original consensus algorithm in a blockchain network.

The algorithm is used to confirm the transaction and creates a new block to the chain. In this algorithm, miners (a group of people) compete against each other to complete the transaction on the network.

The process of competing against each other is called mining. As soon as miners successfully created a valid block, he gets rewarded.

The most famous application of Proof of Work(PoW) is Bitcoin.

Proof of Elapsed Time:

Proof of elapsed time (PoET) is a consensus algorithm developed by Intel Corporation that enables permissioned blockchain networks to determine who creates the next block.

PoET follows a lottery system that spreads the chances of winning equally across network participants, giving every node the same chance.

The PoET algorithm generates a random wait time for each node in the blockchain network; each node must sleep for that duration.

The node with the shortest wait time will wake up first and win the block, thus being allowed to commit a new block to the blockchain.

The PoET workflow is similar to Bitcoin's proof of work (PoW) but consumes less power because it allows a node to sleep and switch to other tasks for the specified time, thereby increasing network energy efficiency.

Proof of Stake:

With proof-of-stake (POS), cryptocurrency owners validate block transactions based on the number of coins a validator stakes.

Proof-of-stake (POS) was created as an alternative to Proof-of-work (POW), the original consensus mechanism used to validate a blockchain and add new blocks.

An algorithm chooses from the pool of candidates the node which will validate the new block.

This selection algorithm of Validators or Forgers combines the quantity of stake (amount of cryptocurrency) with other factors (like coin-age based selection, randomization process) to make the selection fair to everyone on the network.

Delegated Proof of Stake:

Delegated Proof of Stake (DPoS) is a variation of the POS consensus mechanism.

Here, the network participants or nodes use their cryptocurrency or tokens to vote for the delegates.

Just as in POS, the delegates are responsible for validating transactions and maintaining the blockchain ledger. These elected delegates are called witnesses. The more the crypto-coins or tokens, the more the voting power.

Any fraudulent activity by the witnesses can be easily detected by the voters and penalized. As it is a democratic system, it is not only the rich, but all users have a chance to be elected as witnesses and earn rewards. This makes DPOS more decentralized than either PoS or PoW

Proof of Authority:

In PoA, rights to generate new blocks are awarded to nodes that have proven their authority to do so.

These nodes are referred to as “Validators” and they run software allowing them to put transactions in blocks.

These validators are pre-approved and trusted by the network, ensuring that the process is fast and efficient. Unlike other algorithms, PoA relies on the reputation and identity of validators rather than computational power or stake, making it suitable for private or consortium blockchains.

PROS AND CONS OF BLOCKCHAIN

Decentralized and Distributed:

Blockchain technology works on the principle of ledger data distributed to all nodes that are non-hierarchical with no single central control.

Pros:

- ✓ Removes any single point of failure
- ✓ Fosters transparency, faster consensus and synching of data
- ✓ More engagement as everyone is involved in the decision-making process

Cons:

- ✓ In some cases, the traditional database may be more suited and do the work a lot faster and cheaper
- ✓ If a time-tested and fully functional database and the operational network are already in place, the benefits of replacing or introducing blockchain may not produce the required return on investment.
- ✓ Stronger players can take control of the network, impacting decentralization

Trust-lessness:

In the blockchain, cryptography completely replaces the need for third parties to ensure trust.

Pros:

- ✓ Allows for multiple entities or key players who do not trust each other to transact directly with one another.
- ✓ Ensures valid and accurate data
- ✓ Disintermediation (removal of the middleman) reduces the overall cost of transacting.

Cons:

- ✓ Every node needs to run the blockchain to verify transactions and maintain consensus.
- ✓ Significant computing power is expended by miners leading to substantial energy consumption and wastage.
- ✓ Nodes may prioritize transactions with higher rewards.

Immutability:

In blockchain technology, one cannot modify data or transactions once they are recorded in the blockchain database.

Pros:

- ✓ Contains a verifiable record of all transactions made that is auditable
- ✓ Consensus algorithms mitigate the risk of double-spending, fraud, and manipulation of data.
- ✓ There is provenance, i.e., ability to track transaction or product movement across accounts.

Cons:

- ✓ Not every node has the capacity to maintain and run a full copy of the blockchain.
- ✓ In smaller blockchains, there is a risk of a 51% attack.

MODULE 2:

CRYPTOCURRENCY

TRANSACTIONS IN BLOCKCHAIN

In a nutshell, the process of blockchain transaction consists of:

1. A node in the blockchain (P2P network) requests a transaction via a wallet.
2. The transaction is broadcasted to all the nodes in the network.
3. The transaction is validated/verified by the network using consensus algorithms, i.e. preset rules set by the specific blockchain.
4. The transaction is either accepted or rejected. If accepted, the transaction is added in a chronological order along with other transactions to create a new block of data that is sealed (hash).
5. The transaction is now part of the blockchain and is permanent and immutable.

DOUBLE SPENDING PROBLEM

The double-spend problem, a flaw that is unique to digital currencies.

Double-spending, is spending the money more than once. Just as one can copy a digital file and send it to several people, it is possible to duplicate crypto-coin or token and reuse it.

If this occurs in the blockchain network, it could not only breakdown the concept of trusted distributed ledger but also lead to inflation with fraudulent, duplicate currencies in the network

The double-spend problem is circumvented in blockchain through its consensus mechanism.

Once a transaction is confirmed, it is nearly impossible to double-spend it. The more confirmed blocks in the chain, the harder it is to double-spend the crypto.

However, it is theoretically possible to double-spend a cryptocurrency. Though rare, this can be done by the 51% attack

How Blockchain handles the Double Spending Problem?

In a blockchain, every transaction is recorded in a public ledger that is maintained by a network of nodes (computers).

When you spend a digital currency (like Bitcoin), the transaction is verified and recorded in a block. This block is then added to the blockchain, and the network ensures that the same digital currency cannot be spent again.

- First Transaction: You send 1 Bitcoin to Alice. This transaction is verified and added to the blockchain.
- Second Transaction: You try to send the same 1 Bitcoin to Bob. The network checks the blockchain and sees that this Bitcoin has already been spent in the transaction to Alice. The second transaction is rejected.

By recording every transaction in an immutable ledger and having multiple nodes verify each transaction, the blockchain effectively prevents double spending.

UTXO

An unspent transaction output (UTXO) is the technical term for the amount of digital currency that remains after a cryptocurrency transaction

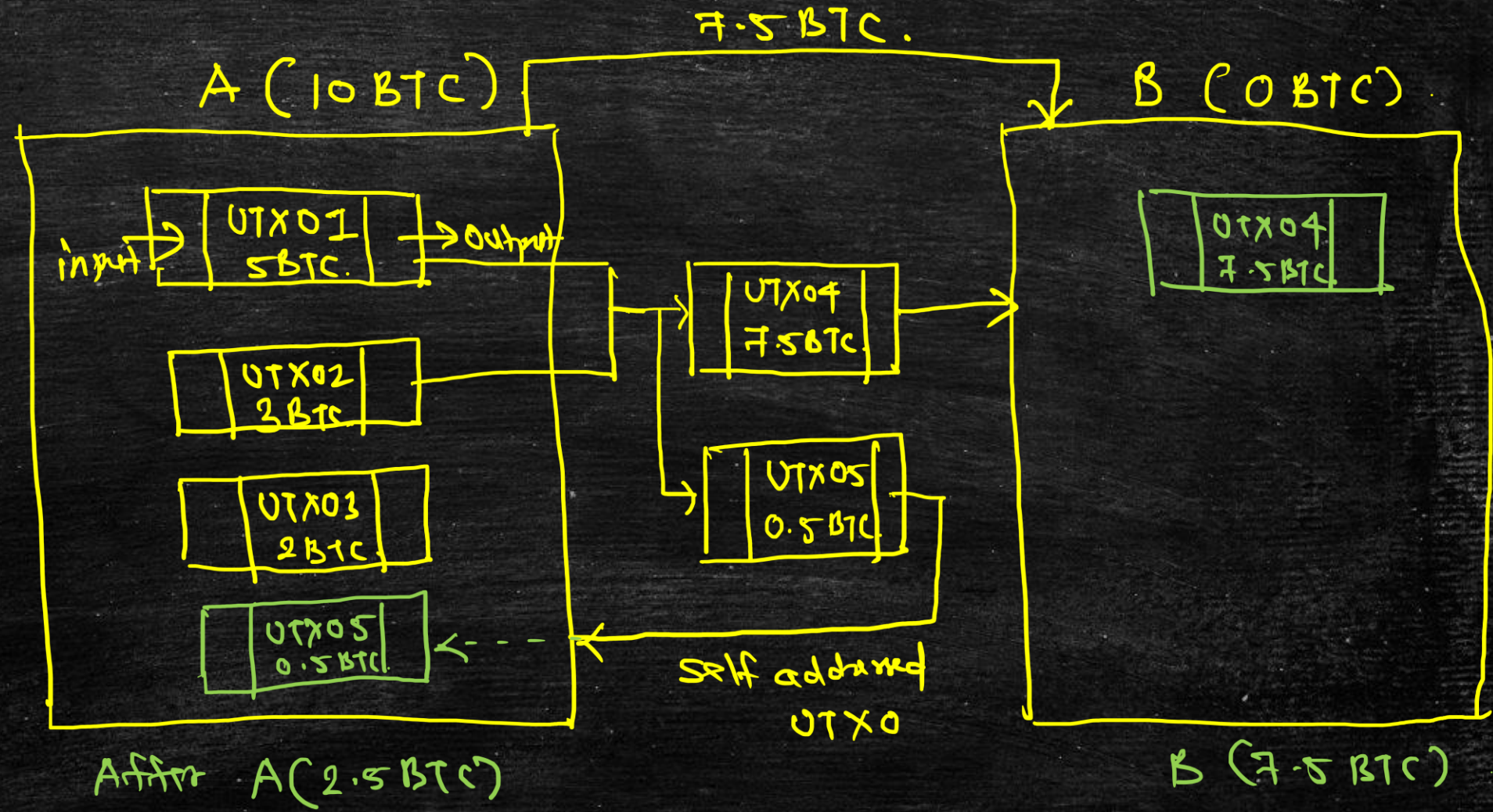
It is the amount of digital currency someone has left remaining after executing a transaction.

When a transaction is completed, the unspent output is deposited back into the database as input which can be used later for another transaction.

UTXOs are created through the consumption of existing UTXOs. Every Bitcoin transaction is composed of inputs and outputs. Inputs consume an existing UTXO, while outputs create a new UTXO.

other node
mining reward
BTC A.

Satoshi
1 BTC = 10^8 satoshi



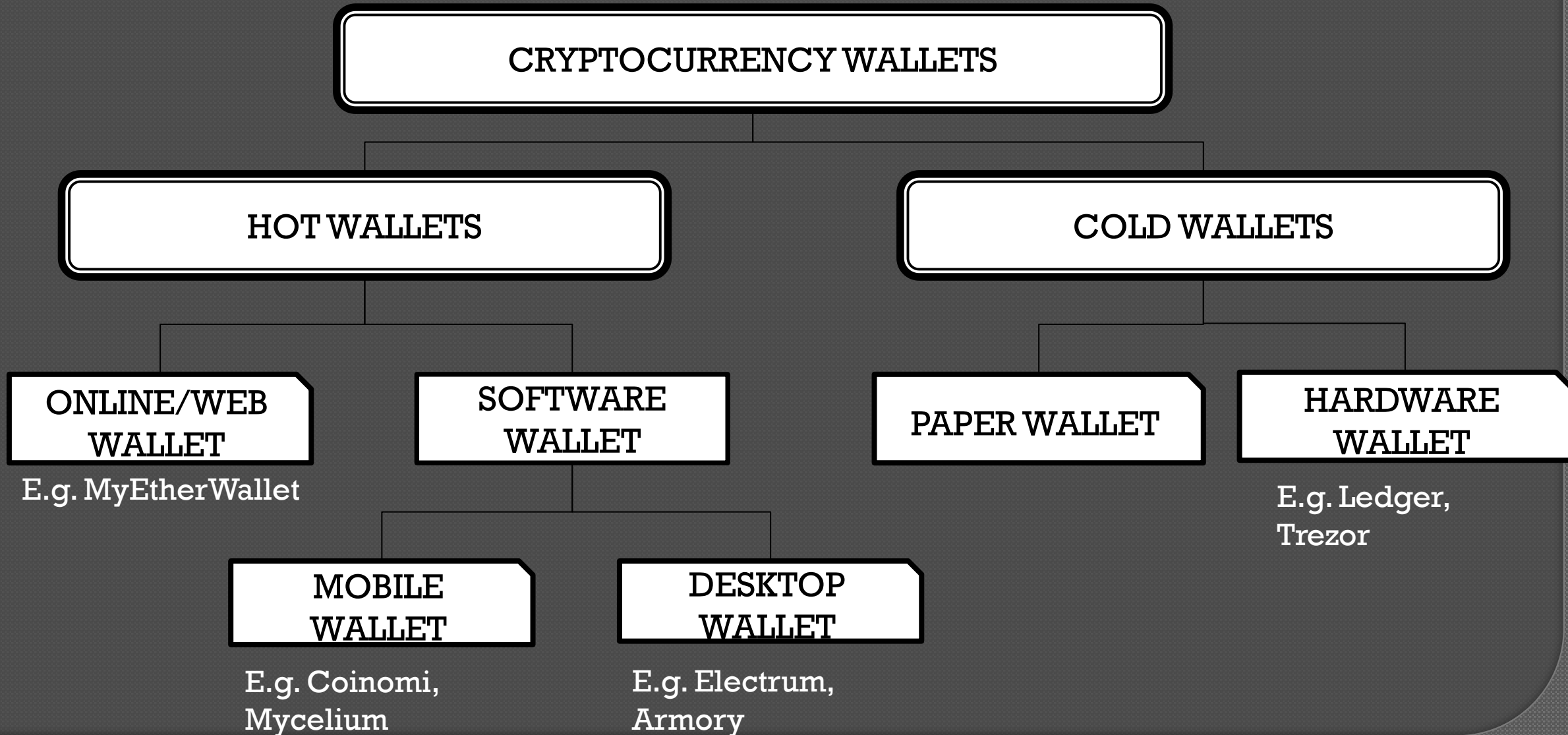
CRYPTOCURRENCY WALLETS

A digital wallet or cryptocurrency wallet is a software program that stores the user's private and public keys enabling the user to transact crypto assets.

It is a management system that interacts with various blockchains to enable users to send and receive digital currency and monitor their balance.

The public key is used by other wallets to send funds to your wallet's address. However, the private key is required if you want to spend cryptocurrency from your address.

Types of wallets



Hot Wallet:

It is designed for online day to day transactions. It is connected to the internet at all times.

It is not advisable to use a hot wallet for long term storage. Hot wallets are typically free, easy to set up and convenient to use

Desktop wallet:

Desktop wallets can be downloaded and installed on our desktop or laptop. As they are locally stored, the user has complete control of the wallet.

The wallet is dependent on the security features installed on your computer.

Mobile wallet:

Mobile wallets are designed to operate on smartphone devices.

you can easily carry it with you wherever you go and make purchases from merchants who accept cryptocurrency payments using QR codes.

However, they are more vulnerable to malicious apps and viruses than desktop wallets.

Online wallet:

Online wallets are also called web wallets. They run on the cloud, and hence the user does not need to download or install any application.

They can be accessed from any computing device via a web browser.

Online wallets are hosted and controlled by a third party and hence are the most vulnerable.

Cold Wallet:

A cold wallet is a digital wallet that is not connected to the internet. Being offline, they are more secure and used for long-term storage of cryptocurrencies.

Hardware wallet:

A hardware wallet is a physical device used to store cryptocurrency private keys securely offline, protecting them from online hacking attempts. To make transactions, users connect the wallet to a computer or mobile device, allowing the wallet to sign transactions without exposing private keys to the internet.

Paper wallet:

A paper wallet involves printing your cryptocurrency private and public keys on a piece of paper, often as QR codes, making it a cost-effective and highly secure method of offline storage. It is generated using trusted software tools and keeps keys completely offline, reducing the risk of online theft. However, paper wallets require careful storage to prevent loss, theft, or damage and need to be imported into a software wallet for transactions.

ALTCOINS AND TOKENS

Altcoins:

The cryptocurrency alternatives to the Bitcoin are referred to as 'Alternative Cryptocurrency Coins,' abbreviated as Altcoins or simply Coins.

Many of the altcoins come from a fork of famous and durable cryptocurrencies like Bitcoin, Litecoin, and Ethereum.

While some altcoins are like their predecessors, most attempt to improve or set themselves apart by bringing in additional features or security like improved block times, different parameters, transaction management, scripting language, consensus mechanism, etc.

The main features of altcoins are:

- They are peer-to-peer digital currencies that involve a mining process.
- They possess their independent blockchain.
- They possess the characteristics of money, i.e., they are fungible, divisible and have limited supply, and typically meant to operate only as a means of payment.

Tokens:

Tokens represent an asset, or a utility created on an existing blockchain. They represent assets that are tradable and fungible.

Tokens are used for fundraising crowd sales; Ethereum is the most popular token platform. All tokens created on the Ethereum platform are called ERC-20 tokens.

Tokens are mostly used with decentralized applications. Developers can decide how many tokens should be created and send the tokens once created. They can pay the blockchain in the blockchain's native currency when creating the token.

Tokens are distributed and sold through Initial Coin Offerings (ICO). ICOs are used to fund the development of the project. Tokens are also used for transactions on the blockchain that they have been created on.

Types of tokens:

Utility tokens:

Utility tokens provide users with access to a product or a service. These tokens are in limited supply, making them rare and valuable.

Security tokens:

Tokens issued by ICOs are mostly security tokens. They are also called as Equity tokens and they represent equity in the company that issues the token.

BITCOIN MINING PROTOCOLS

Proof of Burn:

When coins are destroyed on the blockchain, it is referred to as being burned.

Technically, the coins in circulation are sent to an unspendable address, known as an eater address.

Just like in PoW consensus where the more that is invested in supercomputers and electricity, the more the chances of mining, in Proof of Burn, more the coins one burns, the more chance one gets to mine blocks.

Proof of Burn is used in Counterparty and Slimcoin.

Proof of Work:

Proof of Work(PoW) is the original consensus algorithm in a blockchain network.

The algorithm is used to confirm the transaction and creates a new block to the chain. In this algorithm, miners (a group of people) compete against each other to complete the transaction on the network.

The process of competing against each other is called mining. As soon as miners successfully created a valid block, he gets rewarded.

The most famous application of Proof of Work(PoW) is Bitcoin.

Proof of Elapsed Time:

Proof of elapsed time (PoET) is a consensus algorithm developed by Intel Corporation that enables permissioned blockchain networks to determine who creates the next block.

PoET follows a lottery system that spreads the chances of winning equally across network participants, giving every node the same chance.

The PoET algorithm generates a random wait time for each node in the blockchain network; each node must sleep for that duration.

The node with the shortest wait time will wake up first and win the block, thus being allowed to commit a new block to the blockchain.

The PoET workflow is similar to Bitcoin's proof of work (PoW) but consumes less power because it allows a node to sleep and switch to other tasks for the specified time, thereby increasing network energy efficiency.

Proof of Stake:

With proof-of-stake (POS), cryptocurrency owners validate block transactions based on the number of coins a validator stakes.

Proof-of-stake (POS) was created as an alternative to Proof-of-work (POW), the original consensus mechanism used to validate a blockchain and add new blocks.

An algorithm chooses from the pool of candidates the node which will validate the new block.

This selection algorithm of Validators or Forgers combines the quantity of stake (amount of cryptocurrency) with other factors (like coin-age based selection, randomization process) to make the selection fair to everyone on the network.

BITCOIN MINING TERMINOLOGIES

Creation of bitcoin blocks is called mining.

Mining is the mechanism whereby nodes called "miners" in the Bitcoin world validate the new transactions and add them to the blockchain ledger.

The first miner to create the winning hash will receive rewards in the form of transaction fees or new bitcoins.

This process of computing the hash is called proof-of-work or consensus mechanism and it provides integrity to the blockchain.

All nodes in the network could participate in mining and earn mining rewards.

Block Frequency

Bitcoin transactions are being registered into blockchain once in **ten minutes**. Miners will first check the transaction. After reviewing the transaction, the software will give a complex target hash for the miners to solve. If miners can solve the hash, then the computer will give bitcoins as a reward to the miners.

Industrial Mining

When all the nodes are of uniform size and power, every node gets equal opportunity. It is a fair competition to get a reward. However, there are some nodes, which do industrial-sized mining and **connect to a massive set of computers**, consume enormous power, and use complicated software.

Mining Pool

To counteract the high time and energy consumption in transaction validation, some **miners group together in mining pools** to combine their mining resources for more efficiency and savings. Miners do not work for themselves; they work together in mine pooling. They share the mining power and processing power.

Halving Policy

Block frequency and Halving are the monetary policies of bitcoin. Nowadays, miners get 6.25 bitcoins as a reward to solve the hash. From the year 2009 to 2012, 50 bitcoins were given as rewards. The reward will be reduced to **half for every four years**. At last, in 2140, it will decrease to zero.

Block

Each block (of transactions) has three necessary informations, namely:

1. Block header
2. Hash of previous block header
3. Merkle root

Block header:

In a bitcoin blockchain, the hash function of the previous block header is stored as a reference in the next subsequent block to ensure the blocks are correctly connected.

Hash

A hash function can be considered equivalent of fingerprint of a data, similar to a fingerprint of a person. Using a fingerprint, we can identify a person since it is unique. A hash function converts the data of arbitrary size to data of a fixed size.

Merkle root

A Merkle root is the fingerprint of all transactions in the block. It is created by hashing together pairs of Transaction IDs to give a short and unique fingerprint for all the transactions in the block

Orphaned block

Detached or orphaned blocks are valid blocks, which are not part of the main chain. They occur naturally when **two different miners successfully mine at the same time**.

Timestamp

Timestamp is another field, which indicates UNIX time. It is the seconds passed after the first of January 1970 and is a 10-digit number. This is also part of the data, which changes every second. When the timestamp changes (every second), the corresponding data changes, as do the results.

Mempool

Blocks are added every 10 minutes, but transactions happen all the time. **Mempool is the staging area for the transactions**. A mempool can have around 10,000 transactions at a time. When a miner successfully mines, a set of transactions are added from mempool to the block, and into the blockchain. Once the transactions are added to the blockchain, the same set of transactions is removed from the mempool.

Block Propagation

When a miner announces a block, the block needs to propagate to all the nodes in the network, using **gossip protocol**. A node would perform the following functions:

- a. Validate transaction
- b. Make sure the nonce is valid (nonce is within an acceptable range)
- c. Check if each transaction in the block is valid.

If all of the above three conditions, namely, a, b and c are valid (True), then the block is eligible for further propagation within the network; if the conditions are not met, the node will discard the block. Hence, the block would not be propagated further.

SEGWIT

The size of the signature and public key is so large within the transaction that it occupies more than 60% of the overall size of the transaction. In the bitcoin protocol, this portion (signature and public key) is kept out of the block and is distributed separately. This **segregation of the signature from the block is called as Segregated Witness** (SegWit). Because of SegWit, more (almost double) transactions can be added into the block. Seg Wit will be sent separately in the network.

Nonce

Nonce is a number that can be used just once in the cryptographic communication. Adding a nonce to a transaction's identifier makes it additionally unique, thus reducing the chance of duplicate transactions.

MINING DIFFICULTY

Mining difficulty refers to the difficulty of solving the math puzzle and generating bitcoin.

Mining difficulty influences the rate at which bitcoins are generated.

Mining difficulty changes every 2,016 blocks or approximately every two weeks.

The difficulty level for mining in current bitcoin is 52.39 trillion. That is, the chances of a computer producing a hash below the target is 1 in 52.39 trillion.

Bitcoin mining difficulty is calculated with various formulas. However, the most common one is: $\text{Difficulty Level} = \text{Difficulty Target} / \text{Current Target}$.

The Difficulty Target is a hexadecimal notation.

In contrast, the current target is the target hash of the most recent block of transactions. When the two values are divided, it yields a whole number which is the difficulty level of mining bitcoin.

An adjustment of difficulty upwards or downwards depends on the number of participants in the mining network and their combined hash power.

The mining difficulty of a cryptocurrency such as Bitcoin indicates how difficult and time-consuming it is to find the right hash for each block.

MINING POOL

When all the nodes are of uniform size, power, and every node gets equal opportunity, it is a fair competition to get a reward. However, some nodes indulge in industrial-size mining, by connecting to a massive set of computers, consuming enormous power, and using complex software.

It will be difficult for individual miners to compete with them.

To counteract the huge time and energy consumption involved in transaction validation, some miners group together in mining pools to combine their mining resources for more efficiency and savings. They share the mining power and processing power.

Mining pools provide the service; they share the services so that the individual miners do not repeat the work, thus avoiding double work and waste of time. Nonce values (cryptographic puzzles) are distributed among the individual miners within the pool.

When one of them finds the golden nonce, the corresponding mining pool wins the reward for that block. They share/split the reward based on the computing power (hash rate) of the individual miners.

However, mining pools can go against the basic principle of distributed ledgers as someone or a group gaining control of over 50% of the computing power of the network, usually referred to as a 51% attack, can control the validation process.

51 % attack is not designed to tamper with the blockchain. In order to attack a block within the blockchain, the hacker needs to change all the subsequence blocks of the blockchain

MODULE 3:

PROGRAMMING IN BLOCKCHAIN

INTRODUCTION TO SMART CONTRACT

A Smart Contract is a computer program working on top of a blockchain, which has a set of rules (conditions) based on which, the parties to that Smart Contract, agree to interact with each other.

The agreement will be automatically executed when the pre-defined rules are met.

Real-life agreements to record the purchase and sale of assets can be automated using Smart Contracts.

When the smart contracts are programmed, compiled, and migrated on to the blockchain, they are provided an address depicting the location of the contract on the blockchain, and the application is recorded in a block and distributed across all the nodes.

Hence, the applications are called decentralized applications. Decentralized applications interact with external users through a web browser.

A Smart Contract reduces transaction costs drastically, including cost of Reaching an agreement, Formalizing an agreement and Enforcing an agreement

The benefits of smart contracts are:

- Simple, faster to execute, and availability of updates in real-time
- No requirement of mediators and centralized entities
- Lesser cost because there is no need to pay fees to middlemen
- There is no delay in delivering outcomes.

STRUCTURE OF SMART CONTRACT

How Smart Contract Work

A smart contract works through automated conditional performance. When obligation is met, the corresponding obligation is triggered.

For example, an obligation could be triggered by:

- a specific event ("if X happens, then action Y")
- a specific date or at the expiration of time ("at X date, actionY")

Different platforms can host smart contracts, but generally, smart contracts are implemented on a blockchain where the validation logic sits inside a "block". All the messages needed for a smart contract are bundled within the block.

High-level programming languages like Solidity are used to create applications to automate business processes and record them on the blockchain in the form of smart contracts.

Blockchain technologies use public-key encryption infrastructure (PKI). The initiator who wishes to participate in a smart contract hosted on a permissionless blockchain can download the software from open sources and publish the public key on the system (software) publicly.

When publishing the public key, the blockchain will also generate a corresponding private key for the initiator's address.

If the initiator wants to trigger a smart contract transaction on the relevant ledger, the software will use its address to send an initiating message, encrypted with its private key to the other active participants.

That message is picked up by the nodes; however, it can be signed only by nodes having the private key. Participants with access to the public key received from the software can use it to verify the smart contract transaction and also to authenticate the message contents.

Whenever a sufficient number of other participants or nodes is reached in a permissionless blockchain (typically more than 50%), the consensus protocol determines that message related to the smart contract should be added to the blockchain.

TYPES OF SMART CONTRACT

Smart contracts have the potential to influence many industries. Smart Contracts can be categorized into four types based on the applications

1. Smart Legal Contracts

Imagine a regular legal contract, like an agreement you sign for a job or renting an apartment. Now, think of it as a digital version that automatically enforces the rules. Smart legal contracts are digital agreements that automatically execute actions (like transferring money) when certain conditions are met. For example, if you rent a house, the smart contract can automatically send the rent to the landlord every month without you needing to do anything.

2. DApps (Decentralized Applications)

Think of Dapps like regular apps on your phone, but with a twist—they run on a blockchain instead of a single server. This means they are decentralized, meaning no one person or company controls them. They use smart contracts to run their functions, and once they are launched, they keep working on their own without needing a central authority to manage them.

3. DAO (Distributed Autonomous Organization)

Imagine a company that runs itself without a boss. That's what a DAO is. It's an organization that operates based on rules encoded in smart contracts. People can join the DAO, vote on decisions, and the rules are automatically enforced by the smart contracts. Everything is transparent and happens automatically, so there's no need for a central leader or manager.

4. Smart Contracting Devices (Combined with IoT)

Imagine everyday devices, like a smart thermostat or a car, that can automatically perform actions based on certain conditions. When these devices are connected to a blockchain, they can be programmed with smart contracts to carry out specific tasks autonomously. For example, a smart fridge could automatically reorder groceries when it detects that certain items are running low, and the payment could be made instantly using a smart contract. These devices interact with the blockchain to perform actions without needing human intervention, making them "smart contracting devices."

COMPONENTS OF SMART CONTRACT

Variables

Variables in a smart contract store data that the contract will use or modify. There are different types of variables, each serving a specific purpose:

State Variables:

These are variables that hold data which is stored permanently on the blockchain. Every time a state variable is updated, the change is recorded on the blockchain.

Example: If you have a smart contract for a crowdfunding campaign, a state variable could be used to store the total amount of money raised.

Local Variables:

These are variables used only within a function. They exist temporarily while the function is executing and do not store data permanently.

Example: If a function is calculating the average donation in a crowdfunding contract, it might use a local variable to hold the sum of donations during the calculation.

Global Variables:

Special variables that provide information about the blockchain environment.

Example: Global variables might include the address of the sender of a transaction or the current block's timestamp.

Environment Variables

Environment variables are a special kind of global variable that give the smart contract information about the blockchain context it is operating in:

msg.sender:

What It Is: Represents the address of the person or contract that is calling a function.

Example: In a payment function, msg.sender would be the person sending the payment.

msg.value:

What It Is: Represents the amount of Ether (the cryptocurrency of Ethereum) that was sent with the transaction.

Example: If a user sends 1 Ether to a crowdfunding contract, msg.value would hold that 1 Ether.

block.timestamp:

What It Is: Gives the timestamp of the current block, which can be used to determine when a transaction was mined.

Example: This can be used to set deadlines, like ensuring a payment is made before a specific time.

block.number:

What It Is: Provides the number of the current block.

Example: It can be used to set certain actions that happen at specific blocks.

Functions

Functions in a smart contract define what the contract can do. They can perform actions like transferring money, updating a variable, or interacting with other contracts.

View Functions:

Functions that read data from the blockchain but do not modify it. They are marked as view and don't cost any gas (transaction fee).

Example: A function that returns the current balance of a user's account.

Pure Functions:

Functions that neither read from nor write to the blockchain. They perform calculations and return results without changing the contract's state.

Example: A function that adds two numbers together and returns the sum.

Payable Functions:

Functions that can accept Ether. They are marked as payable, indicating that the function is meant to handle payments.

Example: A function that allows users to send Ether to the contract to purchase a token or service.

Modifier Functions:

Functions that change the behavior of other functions. They can be used to check conditions before a function executes.

Example: A modifier that checks if the caller is the contract owner before allowing certain actions, like withdrawing funds.

Events

Events in smart contracts are like notifications or signals that let people or other programs know when something important happens inside the contract. For example, if someone deposits money, an event can be triggered to announce that this deposit was made. These events help keep a record of actions without storing too much information directly on the blockchain, making it more efficient. They also allow external systems, like websites or apps, to listen for these signals and update information accordingly. In simple terms, events are a way for a smart contract to communicate and log what's going on.

SMART CONTRACT APPROACH

Smart contracts are set of rules and protocols which two parties agree upon and have to follow.

IDE

To write and execute solidity codes, the most common IDE used is an online IDE known as REMIX.

MetaMask

MetaMask is a type of Ethereum wallet that bridges the gap between the user interfaces for Ethereum and the regular web.

MetaMask is mainly used as a plugin in chrome. You can add it from

<https://chrome.google.com/webstore/detail/metamask>

After adding MetaMask as an extension in chrome and creating an account, set up our account as follows –

Step 1: Select Goerli Test Network from a list of available networks

Step 2: Request test ether

Step 3: MetaMask is ready for deployment.

DATA TYPES IN SOLIDITY

Solidity is a statically typed language, which implies that the type of each of the variables should be specified.

Value Types

Value types are data types that store their value directly in the memory. When you assign a value type to another variable, a copy of the original value is created. This means that changing the value of one variable does not affect the other.

Common Value Types in Solidity:

uint (Unsigned Integer):

A non-negative whole number (no decimals). It's called "unsigned" because it doesn't allow negative values.

Example: `uint balance = 100;` creates a variable called balance that holds the value 100.

int (Signed Integer):

A whole number that can be positive or negative.

Example: `int temperature = -10;` creates a variable called `temperature` that holds the value `-10`.

bool (Boolean):

A true/false value.

Example: `bool isOpen = true;` creates a variable called `isOpen` that holds the value `true`.

address:

A type that holds Ethereum addresses, which are used to identify accounts and smart contracts.

Example: `address owner = 0x123...;` stores an Ethereum address in the `owner` variable.

bytes:

A fixed-size array of bytes (e.g., bytes32).

Example: `bytes32 data = "Hello";` creates a variable that stores a 32-byte value, often used for storing hashes.

Reference Type

Reference types store references (or pointers) to the actual data rather than storing the data itself. This means when you assign a reference type to another variable, they both point to the same data. Changing the data through one variable will affect the other.

Common Reference Types in Solidity:

string:

A sequence of characters (text).

Example: `string name = "Alice";` stores the text "Alice" in the variable name.

array:

A collection of values of the same type. Arrays can be fixed-size or dynamic (size can change).

Example: `uint[] numbers = [1, 2, 3];` creates an array of unsigned integers with three elements: 1, 2, and 3.

struct:

A custom data structure that groups different types of variables together.

Example:

```
struct Person
{
    string name;
    uint age;
}
```

```
Person person1 = Person("Alice", 30);
```

This creates a Person struct with a name and age.

mapping:

A collection of key-value pairs, similar to a dictionary or hash map.

Example:

```
mapping(address => uint) balances;
balances[0x123...] = 100;
```

This creates a mapping that associates Ethereum addresses with a balance.

Enums:

An enum in Solidity is a custom data type that lets you define a variable with a limited set of possible values, making your code more readable and safe. For example, if you're tracking the status of an order, you can create an enum with values like Pending, Shipped, and Delivered. This ensures the status can only be one of these valid options, preventing errors and making the code easier to understand. Enums are useful when you want to restrict a variable to a specific list of named choices, enhancing both clarity and reliability in your smart contracts.


```
pragma solidity ^ 0.5.0;
contract Types
{
    bool public boolean = false;
    int32 public int_var = -60313;
    string public str = "Vidyalankar";
    bytes1 public b = "a";
    enum enum1 {apple, orange, mango}
    function Enum() public pure returns(enum1)
    {
        return enum1.mango;
    }
}
```



```
pragma solidity ^0.4.18;
contract example1
{
    uint[5] public array= [1, 2, 3, 4, 5] ;
    struct student
    {
        string name;
        string subject;
        uint8 marks;
    }
    student public std1;
    function structure() public returns(string memory, string memory, uint)
    {
        std1.name = "AAA";
        std1.subject = "Chemistry";
        std1.marks = 88;
        return (std1.name, std1.subject, std1.marks);
    }
}
```


FUNCTIONS IN SOLIDITY

A function is a group of reusable code which can be called anywhere in the program.

Function Definition

Syntax:

```
function function-name(parameter-  
list) scope returns()  
{  
    //statements  
}
```

```
pragma solidity ^0.8.0;  
contract Test  
{  
    function getResult() public view returns(uint)  
    {  
        uint a = 1; // local variable  
        uint b = 2;  
        uint result = a + b;  
        return result;  
    }  
}
```


View functions

A view function is a type of function in a smart contract (like in Solidity) that only reads data from the blockchain but doesn't change anything.

We can think of it like checking your bank balance online. You can see how much money you have, but you're not changing anything in your account.

Example: If you have a smart contract that tracks the number of apples someone owns, a view function might show you how many apples you have. It just looks at the data without altering it.

Callback functions

A callback function is a function that is passed as an argument to another function and gets called after that function is done. It's like giving someone your phone number and asking them to call you back after they finish their work.

In programming, if you ask a smart contract to perform a task (like sending some tokens), you might give it a callback function to execute once the task is complete.

Pure functions

A pure function is a function that doesn't read or change any data outside of itself. It always gives the same result if you provide the same inputs, like a math formula.

Example: If you have a smart contract with a function that just multiplies two numbers together and gives you the result, that's a pure function. It doesn't care about anything else happening in the blockchain.

Fallback Function

A fallback function is like a backup plan in a smart contract. If someone sends Ether to your contract but doesn't specify what to do, or calls a function that doesn't exist, the fallback function automatically runs. It usually just accepts the Ether and adds it to the contract's balance, ensuring the contract doesn't fail.

Example: Imagine you have a smart contract that acts like a donation box. People can send Ether (cryptocurrency) to it, but they don't need to specify any details. The fallback function automatically takes the Ether and adds it to the total donations. It doesn't need any instructions—just send the Ether, and the fallback function handles the rest.

Example:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract CombinedFunctions
{
    uint public totalReceived;
    // Pure function: Multiplies two numbers
    function multiply(uint a, uint b) public pure
returns (uint)
    {
        return a * b;
    }
    // View function: Returns the total Ether
received by the contract
    function getTotalReceived() public view returns
(uint)
    {
        return totalReceived;
    }
}
```

```
// Function that accepts a callback function and
calls it
    function doSomethingAndCallBack(function()
external callback) public {
        // Imagine we do something here...Call the
callback function
        callback();
    }
    // Fallback function: Adds received Ether to
totalReceived
    fallback() external payable {
        totalReceived += msg.value;
    }
}
```


VISIBILITY QUANTIFIERS

Following are various visibility quantifiers for functions/state variables of a contract.

external: External functions are meant to be called by other contracts. They cannot be used for internal call. State variables cannot be marked as external.

public: Public functions/ Variables can be used both externally and internally. For public state variable, Solidity automatically creates a getter function.

internal: Internal functions/ Variables can only be used internally or by derived contracts.

private: Private functions/ Variables can only be used internally and not even by derived contracts.

Example:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract VisibilityExample {
    // Private function: Can only be called within this
    // contract
    function privateFunction() private pure returns (string
memory) {
        return "I am private";
    }
    // Internal function: Can be called within this contract
    // and derived contracts
    function internalFunction() internal pure returns
(string memory) {
        return "I am internal";
    }
    // Public function: Can be called from anywhere,
    // including external calls
    function publicFunction() public pure returns (string
memory) {
        return "I am public";
    }
}
```

```
// External function: Can only be called from outside the
// contract
function externalFunction() external pure returns
(string memory) {
    return "I am external";
}
// Example of using internal and private functions
// within the contract
function testFunctions() public view returns (string
memory, string memory, string memory)
{
    string memory priv = privateFunction(); // Calling
    private function
    string memory intern = internalFunction(); //
    Calling internal function
    string memory pub = publicFunction(); // Calling
    public function
    return (priv, intern, pub);
}
}
```


INHERITANCE

Inheritance is a way to extend functionality of a contract. Solidity supports both single as well as multiple inheritance.

A derived contract can access all non-private members including internal methods and state variables.

Function overriding is allowed provided function signature remains same.

We can call a super contract's function using super keyword or using super contract name.

In case of multiple inheritance, function call using super gives preference to most derived contract.

Example:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

// Parent contract
contract Animal {
    string public species;

    // Constructor to set species
    constructor(string memory _species) {
        species = _species;
    }

    // Public function in parent contract
    function sound() public pure returns (string
memory) {
        return "Some generic animal sound";
    }
}
```

```
// Child contract inheriting from Animal
contract Dog is Animal {

    // Constructor for Dog, passes the species to the
parent constructor
    constructor() Animal("Dog") {}

    // Overriding the sound function in the child
contract
    function sound() public pure override returns
(string memory) {
        return "Woof!";
    }
}
```


ERROR HANDLING

Following are some of the important methods used in error handling:

assert(bool condition): In case condition is not met, this method call causes an invalid opcode and any changes done to state got reverted. This method is to be used for internal errors.

require(bool condition): In case condition is not met, this method call reverts to original state. This method is to be used for errors in inputs or external components.

require(bool condition, string memory_message): In case condition is not met, this method call reverts to original state. This method is to be used for errors in inputs or external components. It provides an option to provide a custom message.

revert(): This method aborts the execution and revert any changes done to the state.

revert(string memory_reason): This method aborts the execution and revert any changes done to the state. It provides an option to provide a custom message.

Example:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract SimpleRequire
{
    uint public balance;
    // Function to deposit Ether with a minimum
    // amount requirement
    function deposit() public payable
    {
        require(msg.value >= 1 ether, "Minimum
deposit is 1 Ether");
        balance += msg.value;
    }
}
```

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract SimpleRevert
{
    uint public balance;
    // Function to withdraw Ether, only if balance is
    // sufficient
    function withdraw(uint amount) public {
        if (amount > balance)
        {
            revert("Insufficient balance for
withdrawal");
        }
        balance -= amount;
        payable(msg.sender).transfer(amount);
    }
}
```

MODULE 4:

PUBLIC BLOCKCHAIN

ETHEREUM COMPONENTS

Miner and Mining node:

Miners are like workers who process transactions and secure the Ethereum network. They use computers (called mining nodes) to solve complex math problems. When they solve one, they get to add a new block of transactions to the blockchain and earn a reward.

Example: Think of miners as people digging for gold. The "gold" in this case is a reward they get for helping keep the Ethereum network running smoothly.

Ethereum Virtual Machine (EVM):

The EVM is like a giant computer that runs all the smart contracts on Ethereum. It makes sure that all contracts work the same way on every computer in the network.

Example: Imagine the EVM as the brain of Ethereum, making sure everything operates correctly no matter where it's being used.

Ether:

Ether (ETH) is the cryptocurrency used on the Ethereum network. It's like digital money that you can send to others, pay for things, or use to run applications on Ethereum.

Example: If Ethereum were a country, Ether would be its currency, like dollars or euros.

Gas:

Gas is a small fee you pay when you do things on Ethereum, like sending Ether or running a smart contract. This fee goes to the miners who process your transaction.

Example: Think of gas like the fuel you put in a car. You need to pay for fuel (gas) to get your transaction moving on the Ethereum "road."

Transactions in Ethereum:

Transactions are the actions you perform on the Ethereum network, like sending Ether or interacting with a smart contract. Every transaction gets recorded on the blockchain.

Example: If you send money to a friend, that's a transaction. In Ethereum, it's like writing down in a public ledger that you sent them some Ether.

Ethereum Accounts:

Accounts in Ethereum are like your digital wallets. They hold your Ether and allow you to interact with the Ethereum network. There are two types: Externally Owned Accounts (controlled by private keys) and Contract Accounts (controlled by code).

An EOA is like a regular Ethereum account that a person controls with a private key. It can send and receive Ether, interact with smart contracts, and initiate transactions.

A CA is an account controlled by a smart contract code, not by a person. It can hold Ether and automatically execute actions when certain conditions are met.

Example: Imagine your Ethereum account as your personal bank account, where you store your digital money and make payments.

Swarm and Whisper:

Swarm: Swarm is a storage service on Ethereum, designed to store large files in a decentralized way, similar to how you store photos in the cloud.

Whisper: Whisper is a messaging service on Ethereum that allows users to send messages securely and privately.

Example: Think of Swarm as a decentralized Google Drive and Whisper as a secure WhatsApp for the Ethereum world.

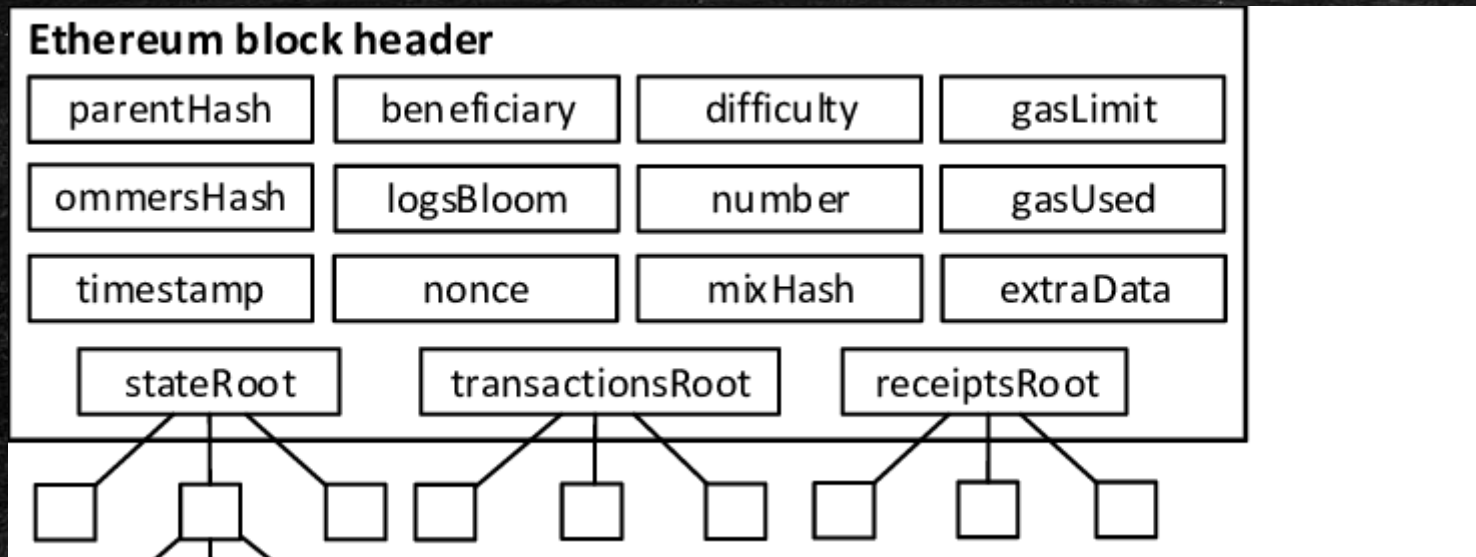
Ethash

Ethash is the algorithm used by Ethereum to secure its network and validate transactions. It's what miners use to solve the complex problems needed to add new blocks to the blockchain.

Example: Ethash is like the rules miners follow when trying to "mine" new blocks, making sure the process is fair and secure.

MINING IN ETHEREUM

The mining principle of Ethereum is different from that of bitcoin. Ethereum miners always look forward to mine new blocks and listen actively to receive new blocks from other miners. The endeavor of miners will always be to build on the block header and perform the subsequent task.



Parent Hash:

The miner need to identify hash of previous block. The hash of previous block or parent block will be added to current block header

LogsBloom:

It is the data structure that consist of log information

Ethereum Accounts:

Ethereum supports two kinds of accounts. Every account comes with a balance that yields the present value saved in it. Once somebody creates an externally owned account on Ethereum, a public-private key is produced.

Transaction, state, and receipt hash:

A transaction hash is a unique identifier (a long string of letters and numbers) that's created when a transaction is made on the Ethereum network. It helps you track and verify the specific transaction on the blockchain. The state in Ethereum refers to the current status of all accounts and smart contracts at any given moment. A receipt hash is a unique identifier that corresponds to the receipt of a transaction. This receipt contains details about the transaction, such as whether it was successful, how much gas was used, and logs of any events triggered.

Nonce:

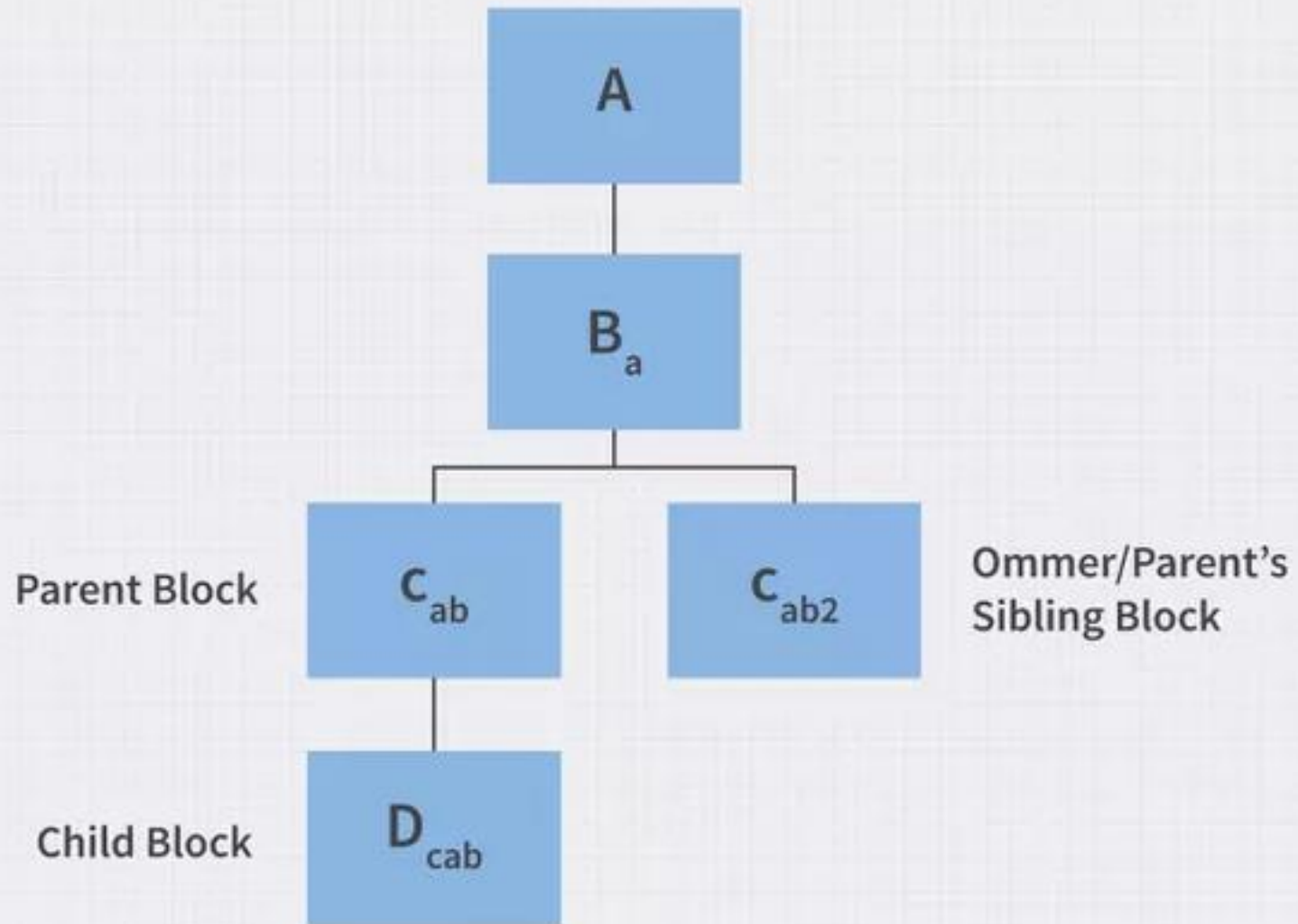
A nonce is a random number that can be used just once in the cryptographic communication.

Timestamp (Current UNIX time):

The timestamp is another field, which indicates the UNIX time. It is the seconds passed after the first of January 1970 and is a 10-digit number.

Ommer block:

Ommer blocks are created in the Ethereum blockchain when two blocks are created and submitted to the ledger at roughly the same time. Only one can enter the ledger.



MixHash:

MixHash can be a hash which, when together with all nonce, demonstrates that a block has enough computation. Miners generate a mixhash until the outcome is the desired target hash.

Difficulty:

It decides the complexity of the puzzle/challenge awarded to miners of a particular block. The gas limit of the block decides the most gas allowed for the block.

Number:

The number is the sequential number of a block on the chain.

ARCHITECTURE OF ETHEREUM

In a decentralized application, we need consensus and the blockchain was that missing ingredient to reach consensus in a decentralized way. So, for consensus (Layer 0) , we need some way to agree upon all of these application level constructs

Layer 5: Dapps
Layer 4: Browsers
Layer 3: Inter-operability
Layer 2: Blockchain services
Layer 1: Economic layer
Layer 0: Consensus layer

Layer 0: Consensus Layer: This is the foundation of Ethereum, where all nodes in the network agree on the state of the blockchain. It ensures that all participants have the same data and that transactions are secure and trustworthy.

Layer 1: Economic Layer: This layer introduces Ether (ETH), the cryptocurrency that powers the Ethereum network. It's used to incentivize participants (like miners) to validate transactions and keep the network running.

Layer 2: Blockchain Services: Here, we have services like smart contracts and decentralized storage. Smart contracts are programs that automatically execute when certain conditions are met, while decentralized storage ensures that data is stored across multiple nodes, making it secure and resilient.

Layer 3: Interoperability: This layer handles how different blockchain networks and their applications communicate and work together. It's like a bridge that allows different systems to interact, ensuring smooth exchanges between various decentralized apps (DApps).

Layer 4: Browsers: Browsers in this context are special interfaces that allow users to interact with DApps on Ethereum. These are not traditional web browsers but are designed to work with decentralized applications, providing access to the Ethereum network.

Layer 5: DApps (Decentralized Applications): The topmost layer is where the actual decentralized applications (DApps) live. These are applications that run on the Ethereum network, offering various services like finance, gaming, and social networks, all without a central authority controlling them.

WORKFLOW OF ETHEREUM

Solidity is the native coding language of Ethereum. It creates a .sol file as an outcome. Solidity Compiler (SOLC) is used to compile the solidity file (sol).

Bytecode invokes Web3js for deployment. Once it is deployed, it returns the Contract and the Application Binary Interface (ABI).

When this contract is initiated, it can invoke contract methods and signs, and pass them to Ether to perform the Operation.

Workflow for Deploying Smart Contracts:

1. Writing the Smart Contract: A developer writes the contract code using Solidity, defining its rules and logic.
2. Compiling the Smart Contract: The code is compiled into bytecode, which can be understood by the Ethereum network.

3. Deploying the Smart Contract: The compiled contract is uploaded to the Ethereum blockchain, making it accessible to users.
4. Interacting with the Contract Using Web3.js: Users interact with the contract via web apps using Web3.js, sending transactions or calling functions.
5. Executing Contract Methods: The network processes these interactions, executing the contract's methods as defined in the code.
6. Updating the Blockchain: Each interaction updates the blockchain, ensuring the contract's state is recorded across the network.
7. Generating Receipts: After processing, a receipt is generated, detailing the transaction's outcome and associated costs.

COMPARISON BETWEEN BITCOIN AND ETHEREUM

	Bitcoin	Ethereum
Founder	Satoshi Nakamoto	Vitalik Buterin and Team
Purpose	Cryptocurrency	Network Software
Release Date	January 2009	July 2015
Coin release method	Early mining	Through ICO
Average block time	Approx. 10 mins	Approx. 15 seconds
Transaction Model	UTXO	Account

	Bitcoin	Ethereum
Coin symbol	BTC	ETH
Tokens	Not available	Available
Monetary policy	Hardcoded	Not hardcoded
Emission rate	Halving policy followed	Occasional
Backward compatibility	Available	Not available
Block limitation	1 MB per block	No limit

TYPES OF TEST NETWORKS

There are three test nets currently in use, and each behaves similarly to the production blockchain (where your real Ether and tokens reside).

1. **Ropsten:** A proof-of-work blockchain that most closely resembles Ethereum; you can easily mine faux-Ether.
2. **Sepolia:** A proof-of-stake blockchain, started by the Ethereum team. Ether can't be mined; it has to be requested.
3. **Rinkeby:** A proof-of-authority blockchain, started by the Geth team. Ether can't be mined; it has to be requested.
4. **Goerli:** A proof-of-authority blockchain, started by the parity team. Ether can't be mined; it has to be requested.

TRANSFERRING ETHERS USING METAMASK

Step 1: Sign Into Your Coinbase Account

To initiate the transfer of ETH, you will need to start by signing into your Coinbase account to access or buy some ETH.

Step 2: Buy Ethereum (ETH)

Select buy, enter the amount of money you want to convert to ETH, double-check that you are buying Ethereum, then choose your payment method and finally click on “Buy Now”

Step 3: Click Send

Once you have some ETH in your Coinbase account, navigate to the top of your account and click the Send/Receive button. Then, choose the amount of ETH you want to send to your MetaMask wallet,

Step 4: Copy/Paste Your MetaMask Address to Coinbase

Before sending your ETH, you need to sign into your MetaMask wallet to copy your wallet address. Next, paste your Metamask wallet address into the “To” section in your Coinbase account.

Step 5: Click Continue/Send Now

After pasting your MetaMask wallet into Coinbase, click Continue to review your transfer. Once you have confirmed the transfer looks good, tap “Send Now”

Step 6: Check Your MetaMask Wallet

To confirm that you have received your ETH, log into your MetaMask wallet where you should see a front preview of your new balance.

ETHEREUM FRAMEWORKS

1. Truffle

Truffle is a popular toolkit for building decentralized applications (DApps) on Ethereum. It simplifies the process of writing, testing, and deploying smart contracts, making it easier for developers to bring their projects to life without needing to handle all the complexities manually.

2. Embark

Embark is a comprehensive framework that allows developers to build both the front-end (user interface) and back-end (smart contracts) of a DApp simultaneously. It streamlines the development process by enabling you to create the entire application in one integrated environment.

3. Hardhat

Hardhat is a professional development environment for building Ethereum smart contracts. It provides tools for compiling, testing, and debugging contracts, ensuring that developers can efficiently build and refine their projects with fewer errors.

4. OpenZeppelin

OpenZeppelin is a library of secure, reusable smart contract templates that developers can use in their projects. It offers pre-built components that save time and enhance security, acting as reliable building blocks for creating robust Ethereum applications.

ETHERSCAN.IO

Etherscan.io is a website that lets you explore and track everything happening on the Ethereum blockchain. It's like a search engine for Ethereum, where you can look up details about transactions, wallet balances, and smart contracts.

Key Features:

Transaction Tracking: You can enter a transaction's unique ID (called a hash) to see all the details, like who sent it, who received it, how much Ether was involved, and whether it was successful.

Wallet Balances: By entering a wallet address, you can see how much Ether and other tokens are held in that wallet, as well as its transaction history.

Gas Fees: Etherscan shows the gas fees (transaction costs) at any given time, helping users decide when to send transactions at a lower cost.

Smart Contracts: You can view, verify, and even interact with smart contracts on Etherscan, seeing the code and how it's used.

Live Updates: It displays live data on all transactions happening on the Ethereum network, so you can see what's going on in real-time.

In short, Etherscan.io is a handy tool for anyone using Ethereum to check transactions, track wallet activity, and explore the blockchain in detail.

MODULE 5:

PRIVATE BLOCKCHAIN

WHAT IS PRIVATE BLOCKCHAIN

Private Blockchains are private, as all permissions for the blockchain are kept centralized. Another name of a private blockchain is Permissioned or Centralized blockchain. They are closed ecosystems where all participants are well-defined. Only pre-approved entities can run nodes.

Characteristics:

Characteristics	Private Blockchain
Organization type	Single entity or organization
Users	Known and trusted participants
Access	Access fully restricted

Network type	Centralized; single point of failure
Operation	Pre-approved participants can read and/or initiate transactions
Verification	Single validator node or central authority to create a block
Consensus mechanism	Voting or variations of PoW/POS consensus algorithms
Security	Security is dependent on the blockchain architecture adopted
Trust	Entrusted; central control
Transaction speed	High; seconds to create a block
Energy consumption	Low
Scalability	Better scalability as high storage and computational power is not required

NEED OF PRIVATE BLOCKCHAIN

A private permissioned blockchain strictly controls who can do what, within the IT system, and there is a well-defined mechanism to revoke and grant access to the nodes.

Hyperledger Fabric is an example of such private permissioned blockchain.

A permissioned blockchain can be different from a private blockchain.

It is possible to create a blockchain network, which is open to the public, i.e., it may allow the public to authenticate itself, grant privilege to nodes and create a mechanism for proper audit trail; such a blockchain is public permissioned blockchain (Ripple blockchain).

Private blockchains are used by individual hobbyists, private enterprises or organizations for a specific purpose. Organizations may prefer to use a private blockchain, if they would like to control:

- Who can use the system
- Who can write to the system
- Who can read the system

Besides, organizations need a solution with a mechanism to ensure users are added via process, and user rights are created, changed, or deleted by an authorized user. These needs gave birth to a need for private blockchain. In addition, the solution needs faster transactions, proper audit trail, and interactions with the organization's existing IT systems.

SMART CONTRACT IN PRIVATE BLOCKCHAIN

In a private blockchain, a smart contract is a self-executing digital agreement where the terms between a buyer and seller are directly written into code. In this environment, only authorized participants within a closed network can create, deploy, and interact with these contracts. Because the blockchain is private, the network is controlled by a single entity or organization, which allows for greater control over who can access the system and perform transactions.

Design Limitations in a Private Environment:

1. Denial-of-Service Attack

If a malicious contract is introduced, it could slow down or block the execution of other contracts by taking up excessive time or resources.

2. Implementation language of Smart contract

Choosing a suitable programming language for smart contracts is crucial, as it needs to balance power with security, and avoid vulnerabilities like infinite loops.

3. State Synchronization in Smart contract

All nodes in the private blockchain must agree on the state of the contract after it's executed. If malicious nodes disrupt this process, it can cause inconsistencies.

Despite these challenges, private blockchains offer faster transaction speeds, lower energy consumption, and better scalability compared to public blockchains, making them suitable for organizations that need a controlled, efficient environment for executing smart contracts.

The CAP Theorem:

The CAP Theorem is a concept in computer science that explains the limitations of distributed systems, like blockchains. It says that any distributed system can only guarantee two out of three of the following properties at the same time:

Consistency

Every node (computer) in the network sees the same data at the same time. If you update the data in one place, all other nodes will immediately reflect that change.

Example: Imagine if every time you made a change to a Google Doc, all your collaborators instantly saw the exact same change. That's consistency.

Availability

Every request to the system gets a response, even if the response is not the most recent data.

Example: Think of a web page that always loads, even if some of the information is slightly outdated because the server couldn't get the latest data right away.

Partition Tolerance

The system continues to operate even if there are communication breakdowns between parts of the system. It can handle the failure of some parts of the network without crashing.

Example: Imagine a messaging app where you can still send messages even if part of the network is down, though some users might receive the messages later.

The CAP Theorem explains that in any distributed system, like a blockchain, you can only have two out of three properties—Consistency, Availability, and Partition Tolerance—at the same time. This means you have to make a trade-off: if you prioritize Consistency and Availability, the system might struggle to handle network failures (Partition Tolerance). If you focus on Consistency and Partition Tolerance, sometimes the system might not be immediately available. And if you choose Availability and Partition Tolerance, the system might not always have the most up-to-date information (Consistency). In other words, you can't have it all, so you have to decide which two are most important for your needs.

The BASE Theory:

The BASE Theory is an approach used in distributed systems, like blockchains, that's more relaxed compared to the strict rules of the CAP Theorem. It stands for:

Basically Available:

The system guarantees that it will always be available to respond to requests, even if it's not always perfectly up-to-date or accurate.

Example: Imagine a store that's always open for customers, even if some shelves might not have the latest products or information.

Soft state:

The system's state (its data) can change over time, even without new inputs, because it's gradually syncing with other parts of the system.

Example: Think of a social media feed that slowly updates with new posts; it's not instantly synchronized but eventually catches up.

Eventual Consistency:

The system doesn't need to be consistent all the time, but it guarantees that if you wait long enough, all parts of the system will eventually agree on the same data.

Example: Imagine your bank account balance might show differently on different devices immediately after a transaction, but eventually, they all show the correct balance.

BASE Theory focuses on making sure the system is always available and that it will eventually be consistent, even if it's not right away. This approach is useful in systems that prioritize availability and performance over immediate accuracy, making it more flexible but less strict than approaches that demand immediate consistency.

STATE MACHINE IN SMART CONTRACT

A state machine in a smart contract is a way to manage and track the different stages or "states" that a contract can be in. It ensures that actions or transactions happen in a specific order, and only when the contract is in the correct state.

How It Works:

States: Think of states as the different steps or phases a contract can go through. For example, a contract might start in a "Pending" state, then move to "Active," and finally to "Completed."

Transitions: These are the rules that determine how and when the contract can move from one state to another. For instance, a contract might only move from "Pending" to "Active" when both parties have signed it.

Execution: The smart contract follows these rules strictly, making sure that all transactions or actions happen in the right order. If a transaction doesn't match the current state or expected transition, it won't be processed.

Imagine a vending machine as a state machine. The machine starts in a "Ready" state. When you insert money (a transition), it moves to the "Money Inserted" state. Then, when you press a button to select a product, it moves to the "Dispensing" state. Finally, after giving you the product, it returns to the "Ready" state. The machine won't dispense a product unless it's in the "Money Inserted" state, ensuring everything happens in the right order.

In smart contracts, this concept ensures that all actions happen in the correct sequence and that everyone follows the rules set in the contract, making transactions secure and orderly.

CONSENSUS ALGORITHMS FOR PRIVATE BLOCKCHAIN

PAXOS Algorithm:

The PAXOS Algorithm is a way to achieve agreement (consensus) among multiple computers (nodes) in a network, even if some of them fail or give wrong information. It's used to ensure that everyone in the network agrees on the same decision, like which transaction to process or which block to add to the blockchain.

Paxos has three entities:

1. **Proposers:** These are the nodes that suggest a value or decision to the network. They propose what they think should be the next step or decision.
2. **Acceptors:** These nodes listen to the proposers and vote on whether to accept the proposed value. They play a key role in reaching an agreement.
3. **Learners:** Once a value is accepted by enough acceptors, the learners announce the agreed-upon decision to everyone in the network.

Handling Failures:

1. Acceptor Fails:

If an acceptor (a node that votes) fails during the process, other acceptors can still continue the process. As long as the majority (more than half) of the acceptors are still working, the system can reach a decision.

2. Proposer Fails:

If a proposer (a node that suggests a value) fails, the process can be restarted by another proposer. The system waits, and if it doesn't hear from the original proposer, another node steps in to make a new proposal.

Multi-PAXOS System:

In real-life systems, we often need to make a series of decisions, not just one. The Multi-PAXOS system extends the basic PAXOS process to handle multiple decisions in sequence, like processing multiple transactions in order.

Multi-PAXOS repeatedly runs the PAXOS process for each decision. It's more complex because each decision needs communication among the nodes, but it ensures that a series of decisions can be made correctly, one after the other.

Imagine you and your friends are trying to decide on a place to eat. One friend (the proposer) suggests a restaurant. The rest of the group (acceptors) vote on whether they agree. If most agree, the decision is made, and everyone goes to that restaurant. If one of your friends (an acceptor) is not available, the rest can still make the decision if the majority agrees. If the friend who suggested the restaurant (the proposer) changes their mind or is unavailable, someone else can suggest a new place, and the process starts again.

In summary, the PAXOS Algorithm is a method for ensuring that all participants in a network agree on the same decision, even if some participants fail or don't respond. The Multi-PAXOS system allows this process to handle multiple decisions, one after the other.

RAFT Algorithm:

The RAFT Algorithm is a method used in distributed systems to ensure that all computers (nodes) in a network agree on the same data or decision, and it does this by electing a leader to manage the process. The leader coordinates the work, making sure everyone is on the same page.

How RAFT Works:

Leader: The main node that makes decisions and tells the other nodes (followers) what to do.

Candidate: A node that wants to become the leader. It asks the other nodes to vote for it.

Follower: A node that listens to the leader and follows its instructions.

When the system starts, all nodes are followers. If no leader is present, a follower can become a candidate and ask for votes from the others to become the leader.

The node that gets the most votes becomes the leader and starts managing the process.

Handling Failures:

1. Leader Fails:

Case 1: Leader Fails and Doesn't Recover:

When the leader fails, the followers notice that they haven't heard from the leader for a while. They hold a new election to choose a new leader. The node that gets the most votes becomes the new leader.

Case 2: Leader Fails and Recovers:

If the old leader recovers after a new leader has been elected, it steps down and becomes a follower because it knows that a new leader is now in charge.

2. Two Candidates Compete:

Case 1: Both Candidates get Votes at the Same Time:

If two nodes become candidates at the same time, they might split the votes. In this case, neither can become the leader right away. The system will hold another round of voting until one candidate gets the majority and becomes the leader.

Case 2: One Candidate Wins:

The candidate that wins the majority vote becomes the leader. It sends out a "heartbeat" message to let all other nodes know that it's in charge, and the other candidate steps down and becomes a follower.

Imagine a classroom where one student is chosen as the class leader. The leader writes the homework on the board, and all students (followers) copy it down. If the leader is absent (fails), the class votes for a new leader. If two students want to be the leader and both get some votes, they have to vote again until one wins. If the old leader comes back, they let the new leader continue in charge.

In summary, the RAFT Algorithm helps a group of computers (nodes) in a network agree on a shared state by electing a leader to manage the process. If the leader fails, the system can recover by electing a new leader, ensuring that the network continues to function smoothly.

Byzantine Fault Tolerance Algorithm:

The Byzantine Fault Tolerance (BFT) Algorithm is designed to help a network of computers (nodes) agree on a decision even if some of the nodes are faulty or acting maliciously. It's named after the "Byzantine Generals Problem," where generals need to agree on a battle plan, but some of them might be traitors trying to confuse the others.

How BFT Works:

The goal is to ensure that the network can still reach a correct decision even if some nodes try to sabotage the process by sending wrong or conflicting information.

The correct decision is made based on the majority of honest nodes. If most nodes are honest, they will outvote the malicious ones, leading to the right decision.

Three Byzantine Problem:

Imagine there's one commander (leader) and two lieutenants (followers). The commander sends orders to the lieutenants, but one of them might be faulty.

Lieutenant is faulty:

If one lieutenant is faulty but the commander is honest, the other honest lieutenant can still follow the correct order because it's clear what the commander wants.

Commander is faulty:

If the commander is faulty and sends conflicting orders to the lieutenants, the honest lieutenants might get confused because they receive different instructions. In this case, they won't be able to agree on the right action, and the system fails to make a correct decision.

Four Byzantine Problem:

Now, imagine there's one commander and three lieutenants. This extra lieutenant helps improve the chances of making the right decision even if some nodes are faulty.

One lieutenant is faulty:

If one of the three lieutenants is faulty and sends different messages to the others, the two honest lieutenants can still compare notes and agree on the correct decision based on the majority rule.

Two lieutenant are faulty:

If two of the three lieutenants are faulty, they can send different messages to the honest lieutenant, making it impossible for the honest lieutenant to figure out the correct decision. In this case, the system fails to make a correct decision.

Commander is faulty:

If the commander is faulty but the majority of lieutenants are honest, the honest lieutenants can compare their orders and reach the correct decision despite the faulty commander.

Handling Failures:

Faulty Leader (Commander):

If the leader sends conflicting information to different followers, the honest followers may get confused, but if the majority are honest, they can still agree on the right decision by comparing their information.

Faulty Followers (Lieutenants):

If some followers are faulty, the honest ones can still reach a correct decision by ignoring the conflicting information from the faulty nodes and following the majority.

Imagine a group of friends trying to decide on a place to eat, but one of them is trying to confuse the others by suggesting different places to different people. If most of the friends are honest and compare their suggestions, they can still agree on the right place to go, even if the confusing friend tries to mess things up.

The Byzantine Fault Tolerance (BFT) Algorithm helps a network reach the right decision even if some participants are faulty or acting maliciously. By relying on the majority of honest nodes, the network can still function correctly despite these faults.

Lamport-Shostak-Pease Algorithm:

The Lamport-Shostak-Pease Algorithm is a method used to ensure that all participants in a network agree on the same information, even if some participants (nodes) are faulty or malicious. This algorithm is an extension of the Byzantine Fault Tolerance (BFT) concept, and it focuses on how to achieve consensus in the presence of these faulty nodes.

Key Concepts:

Broadcasting Messages: The leader (commander) sends a message to all other nodes (lieutenants). Each lieutenant then shares the received message with the others. This way, everyone gets the same information and can compare it.

Pulse 1 Message: When a lieutenant receives a message, it checks if it's directly from the leader (a "pulse 1" message). If it is, the lieutenant considers it reliable. If the message comes indirectly from another lieutenant, the lieutenant has to decide based on what the majority says.

Majority Voting: To reach a final decision, each node compares all the received messages. They apply a majority rule, where the decision supported by most nodes is accepted as the correct one. This helps overcome the influence of faulty nodes.

Example:

Imagine you're in a group chat where one person (the leader) sends out a message about where to meet. Everyone in the group then shares this message with each other to make sure they all received the same information. If most people agree on the location, that's where everyone will go, even if one or two people received or sent wrong info.

Practical Byzantine Fault Tolerance (pBFT) Algorithm:

The Practical Byzantine Fault Tolerance (pBFT) Algorithm is a specific approach used in blockchain networks to achieve consensus efficiently, even if some nodes are faulty or malicious. It's designed to work well in real-world systems with multiple nodes.

Key Concepts:

Leader and Backups: In pBFT, one node is chosen as the leader (primary), and the other nodes are backups (secondary). The leader receives requests from clients and forwards them to the backups for processing.

Agreement on Transactions: All nodes (leader and backups) work together to process and agree on each transaction. They run calculations and share the results with each other to ensure everyone agrees.

Final Decision: The transaction is accepted only if the majority of nodes agree on the result. This means that even if some nodes are faulty, they can't affect the outcome as long as most nodes are honest.

Handling Failures:

Small Group Sizes: pBFT works best in small networks because all nodes need to communicate with each other to agree on every transaction. This keeps the process fast and efficient.

Sybil Attacks: In a Sybil attack, a single bad actor might pretend to be multiple nodes to try and manipulate the network. pBFT guards against this by relying on the honest majority to outvote the malicious nodes.

Imagine you are part of a committee that makes decisions. One person (the leader) proposes a decision, and everyone (backups) discusses and votes on it. The decision is accepted only if most members agree. Even if one or two members disagree or try to sabotage the process, the majority rule ensures the right decision is made.

HYPERLEDGER FRAMEWORK

The Hyperledger Framework is a collection of tools and platforms designed to help businesses create their own private, secure, and scalable blockchain networks. It's like a toolkit that allows companies to build and customize their blockchain solutions for specific needs, without relying on public blockchains like Ethereum.

Key Features:

Permissioned Blockchain: Hyperledger networks are "permissioned," meaning only approved participants can join and interact with the network. This makes it more secure and controlled compared to public blockchains.

Modular Architecture: Hyperledger is built with flexibility in mind. It's modular, which means businesses can pick and choose different components to build a blockchain that fits their specific requirements.

No Cryptocurrency: Unlike public blockchains that use cryptocurrencies like Bitcoin or Ether, Hyperledger doesn't require a native currency to operate. It focuses more on business applications and secure data sharing.

Popular Hyperledger Frameworks:

1. Fabric:

This is one of the most widely used frameworks under the Hyperledger umbrella. It's known for its high security, flexibility, and ability to handle complex business processes. Fabric allows businesses to create their own blockchain networks with customized rules and privacy settings.

2. INDY:

Indy focuses on digital identity. It provides tools to create and manage decentralized identities, which can be used to verify the identity of participants in a network without relying on a central authority.

3. Sawtooth:

Another framework that supports smart contracts and is designed to be highly modular. Sawtooth allows businesses to separate different parts of the blockchain process, making it easier to manage and scale.

4. Grid:

This framework is specifically designed for supply chain management. It helps businesses track and manage the flow of goods and services in a transparent and secure way.

Imagine a large company that needs to securely track shipments across the globe. Using the Hyperledger Framework, they can create a private blockchain where only trusted partners can join and see the shipment details. They can customize the network to their needs, ensuring that everything is secure, transparent, and efficient, without needing to use a public blockchain or cryptocurrency.

HYPERLEDGER TOOLS

The Hyperledger Tools are a set of software tools that help developers and businesses create, manage, and interact with blockchain networks built using Hyperledger frameworks. These tools make it easier to develop blockchain applications, monitor blockchain networks, and manage data on the blockchain.

1. Composer

Composer is a tool that simplifies the process of building blockchain applications. It provides a user-friendly way to model business networks, write smart contracts, and quickly create and test blockchain applications.

Example: Imagine you want to create a prototype of a blockchain application to manage supply chains. Composer helps you design the system, write the rules (smart contracts), and test everything without needing deep blockchain expertise.

2. Explorer:

Explorer is a tool that provides a web-based interface to view and analyze the blockchain network. It allows users to see blocks, transactions, and network status in real-time.

Example: Think of Explorer as a dashboard for your blockchain. It's like a control panel where you can see all the activity on your network, check the history of transactions, and monitor the health of the system.

Imagine you're a business owner wanting to create a blockchain application for tracking product deliveries. Using Hyperledger Composer, you can easily model your business process, write the necessary rules, and test the application. Once your application is running, Hyperledger Explorer lets you monitor all the deliveries and ensure that everything is being tracked correctly in real-time.

Comparison between Hyperledger Fabric & Other Technologies

Characteristics	Ethereum	Hyperledger Fabric	R3 Cords
Platform	Generic Blockchain	Modular Blockchain	Specialized for financial industry
Governance	Ethereum developers	Linux Foundations	R3
Mode of Operation	Public, Permissionless	Private, Permissioned	Private, Permissioned
Consensus	PoW or PoS	Multiple approaches	Multiple approaches
Smart Contract Language	Solidity	Go, JAVA	Kotlin, JAVA
Currency	Ether	None	None

HYPERLEDGER FABRIC ARCHITECTURE

Hyperledger Fabric is a framework used to build private and secure blockchain networks for businesses. Its architecture is designed to be flexible, allowing companies to create customized blockchain solutions that meet their specific needs.

Key Components of Hyperledger Fabric:

Peers:

Peers are the computers (nodes) in the network that maintain the blockchain. They store a copy of the ledger (the record of all transactions) and can execute smart contracts (called chaincode in Fabric).

Example: Imagine peers as the employees in a company who keep records of all business transactions. Each peer has a copy of the records and helps verify new transactions.

Orderer:

The Orderer is responsible for organizing the transactions into blocks and ensuring they are added to the blockchain in the correct order. It ensures that everyone in the network sees the same sequence of transactions.

Example: Think of the Orderer as the manager who organizes all the records in the right order before they are filed away, making sure everything is in sequence.

Channels:

Channels are private pathways within the blockchain network where specific groups of participants can conduct transactions privately, without others in the network seeing them.

Example: Imagine channels as private meeting rooms in a company where only certain departments discuss and finalize transactions that others don't need to know about.

Membership Service Provider (MSP):

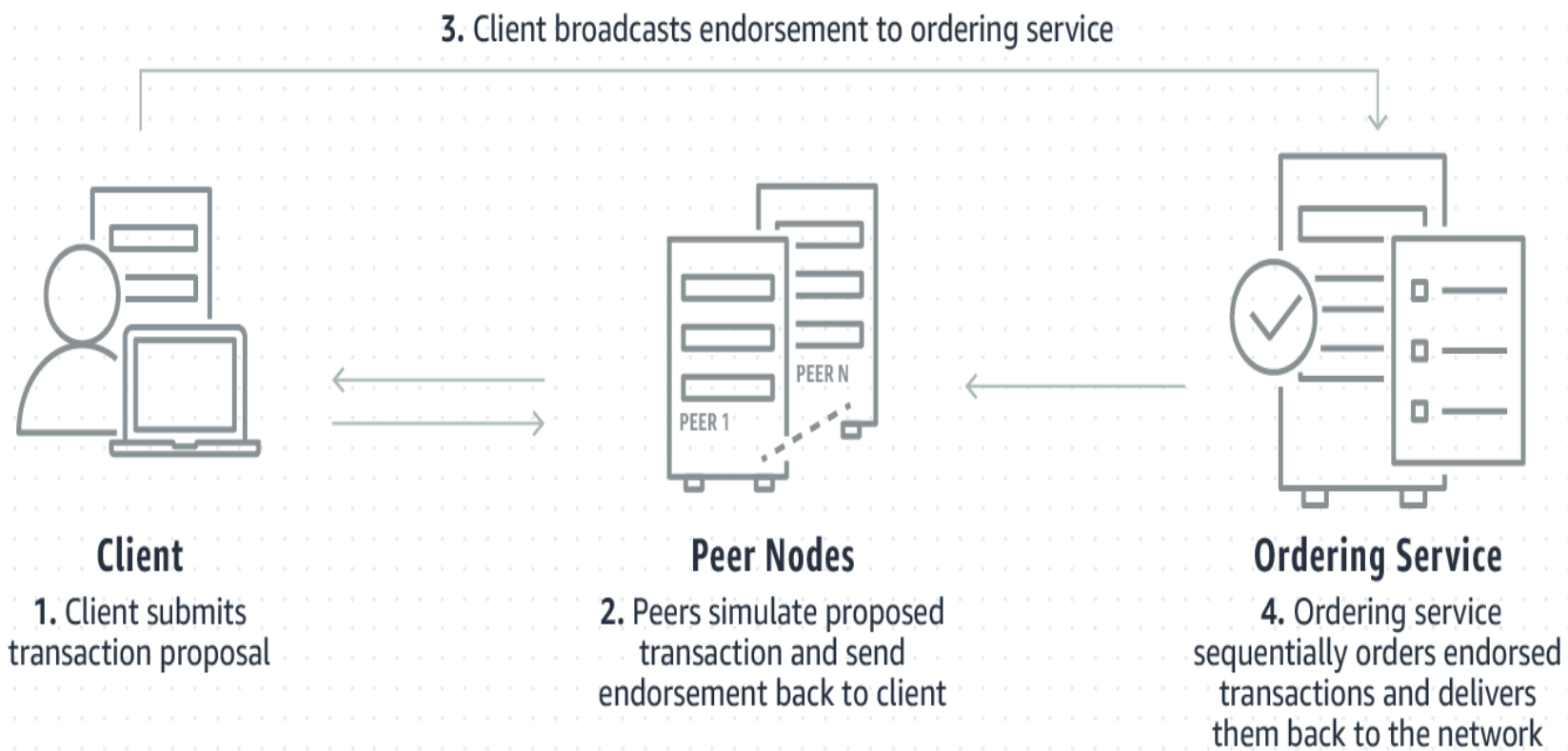
MSP is the component that manages the identities of all participants in the network. It ensures that only authorized members can join and interact with the network.

Example: Think of MSP as the security team that checks IDs and grants access to the building only to employees who are supposed to be there.

Chaincode (Smart Contracts):

Chaincode is the term for smart contracts in Hyperledger Fabric. These are the rules or logic that define what happens when certain transactions occur.

Example: Chaincode is like the company policies that automatically trigger certain actions when specific conditions are met, like approving a payment when an invoice is received.



Workflow:

1. Creation of the proposal:

A participant in the network initiates a transaction, which is sent to specific peers for approval based on the chaincode (smart contract).

2. Endorsement:

The selected peers (endorsers) check the transaction against the rules set in the chaincode. If the transaction is valid, they endorse it.

3. Ordering:

The Orderer collects all the endorsed transactions, organizes them into blocks, and ensures they are added to the blockchain in the correct order.

4. Commitment:

Finally, the blocks of transactions are distributed to all peers in the network, who update their copies of the ledger to reflect the new information.

SUPPLY CHAIN MANAGEMENT USING HYPERLEDGER

Supply Chain Management involves tracking the movement of goods from the manufacturer to the consumer. Using Hyperledger for this process makes it more secure, transparent, and efficient.

How Hyperledger Helps in Supply Chain Management:

Tracking Every Step:

Hyperledger allows every step of the supply chain to be recorded on a blockchain. This includes when goods are made, shipped, and delivered.

Example: Imagine being able to see where your package is at every moment, from the factory to your doorstep. Hyperledger keeps a record of each step in the process.

Transparency and Trust:

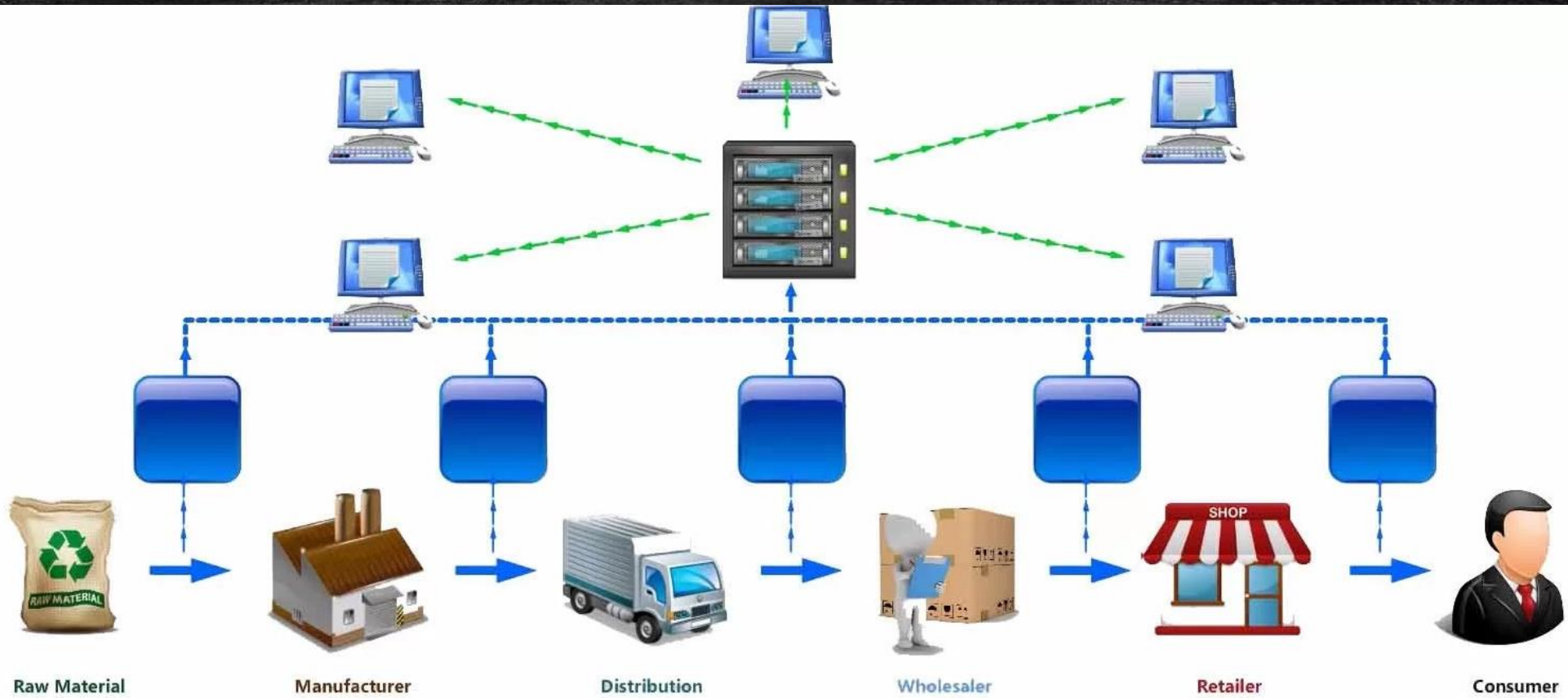
Since the information is recorded on a blockchain, everyone involved can see the history of the product, ensuring that it's authentic and hasn't been tampered with.

Example: Think of buying an expensive watch and being able to see its entire journey from the manufacturer to the store, making sure it's genuine.

Faster and More Efficient:

By using Hyperledger, companies can reduce the time and cost involved in tracking and managing goods, as everything is automated and securely recorded.

Example: Instead of waiting for paperwork to be processed, everything is done instantly on the blockchain, speeding up the delivery process.



Key Benefits:

Provenance Tracking:

Provenance means knowing where something comes from. Hyperledger records every detail about the origin and history of the goods, which is crucial in industries like food, medicine, or luxury goods.

Example: If there's a problem with a product, like a recall on a food item, companies can quickly trace it back to its source and find out where things went wrong.

Cost Reduction:

Hyperledger helps cut down costs by reducing the need for middlemen and manual processes. It allows real-time tracking, which means better planning and fewer delays.

Example: Imagine cutting out all the extra steps that slow down deliveries and cost money—Hyperledger helps streamline the process.

Building Trust:

Since everyone in the supply chain can see the same information, it's easier to build trust among all parties. This reduces the chances of fraud or mistakes.

Example: When all participants—like suppliers, manufacturers, and retailers—can see the same records, they're more likely to trust the information and work together smoothly.

Improved Traceability:

Hyperledger makes it easy to trace products through every step of the supply chain, ensuring that if something goes wrong, it can be quickly identified and fixed.

Example: If a batch of medicines is found to be faulty, the exact location of the issue can be identified quickly, preventing further distribution.

MODULE 6:

TOOLS AND APPLICATIONS OF BLOCKCHAIN

HYPERLEDGER FABRIC

Hyperledger Fabric is a blockchain framework that is permissioned, meaning it is only accessible to authorized users. It was created by The Linux Foundation and is designed for use in businesses where privacy, security, and quick transactions are important. Here's how it works in simple terms:

How does Hyperledger Fabric Work?

1. Transactions: A transaction starts when a user (client) sends a request to the system. This request is sent to different organizations, which check the user's identity and see if they are allowed to make the transaction.
2. Endorsements: These organizations (called peers) then simulate the transaction and, if everything looks right, approve it by giving their signatures. The user collects these approvals.

3. **Ordering:** Once the required approvals are collected, the transaction is sent to the ordering service, which arranges all transactions in order and groups them into blocks.
4. **Validation and Commitment:** The blocks are sent back to the peers, which double-check the transactions to make sure everything is in order. Once validated, the transactions are added to the blockchain, and the ledger is updated.

Key features include:

- **Open Source:** It's free to use and modify.
- **Permissioned:** Only authorized members can access the system, ensuring security and privacy.
- **Access Control:** Members can create private channels where only specific parties can view the transactions, useful for businesses that need to keep certain information confidential.
- **Performance:** Hyperledger Fabric is designed to handle high-speed, enterprise-level transactions efficiently.

CORDA

Corda is a distributed ledger platform designed for handling legal agreements and transactions between two or more parties in a secure way. It focuses on privacy and efficiency, especially for business applications like financial contracts.

Working:

1. **Distributed Ledger:** Like other blockchains, Corda stores data across multiple computers (nodes), which makes it secure because no single party can change the data without others knowing.
2. **Private Transactions:** One key difference with Corda is that only the parties involved in a transaction can see the details. So, if two companies make a deal, only they can access the information, unlike typical blockchains where everyone on the network sees all transactions.

3. Smart Contracts: Corda uses smart contracts, but they are permissioned. This means only the people involved in the contract can execute it, and, if needed, a third party like a legal authority can also review and verify the contract.

Advantages of Corda:

- Data Privacy: You can decide who sees each transaction, making it ideal for sensitive business deals.
- Fast Transactions: Corda doesn't wait for blocks to be filled before verifying transactions. Each transaction is verified individually, which makes it faster.
- Improves Business Efficiency: Corda helps businesses cut costs and improve cooperation by using a trusted and efficient network for exchanging information and contracts

RIPPLE

Ripple is a digital payment network that allows fast and cheap transactions using its own cryptocurrency called XRP. It's designed mainly for banks and financial institutions to transfer money quickly across the globe.

Working of RIPPLE:

1. **Consensus Mechanism:** Unlike other blockchains like Bitcoin, which rely on miners, Ripple uses a group of servers (usually owned by banks) to confirm transactions. This process is faster and uses less energy.
2. **Fast Transactions:** Ripple transactions are completed in seconds, while Bitcoin transactions can take much longer. This makes Ripple ideal for businesses that need quick transfers of large sums of money.
3. **Low Fees:** Transactions on Ripple cost very little, making it an efficient option for cross-border payments.

Key features:

- Primarily for Banks: Ripple is mostly used by financial institutions to settle international payments faster and more cheaply than traditional methods.
- Not Mined: XRP is not mined like Bitcoin. Instead, all the XRP tokens were created at once, and new XRP cannot be created.
- Energy Efficient: Since Ripple doesn't use mining, it consumes much less energy compared to other cryptocurrencies like Bitcoin

QUORUM

Quorum is a blockchain platform that is built for businesses and aims to improve privacy and performance compared to other blockchain systems like Ethereum. It's often used for enterprise applications where security and efficiency are important.

Working of Quorum:

1. Private and Public Transactions: Quorum allows both public and private transactions. For public transactions, everyone can see the details, but for private ones, only the involved parties can view the transaction, which adds a layer of privacy.
2. Faster Transactions: Quorum uses different ways of verifying transactions called RAFT and IBFT (Byzantine fault tolerance), which are faster than traditional methods like Ethereum's proof of work. This makes transactions quicker.

3. No Transaction Fees: Unlike other blockchains, Quorum does not charge fees for transactions (no "gas" cost). This is beneficial for businesses that need to process many transactions.

Key features:

- Hybrid Smart Contracts: Quorum supports both public and private smart contracts, ensuring flexibility for businesses that need different levels of privacy.
- High Performance: It's designed to handle high-speed transactions, especially with private contracts, which are generally faster than public ones.
- Voting-Based Consensus: Quorum uses a system where only selected, trusted nodes vote to verify transactions, making it more efficient and secure.

This makes Quorum an ideal choice for financial services and other industries where speed, privacy, and security are critical

DEFI

DeFi, or Decentralized Finance, refers to financial services that run on blockchain technology without the need for traditional banks or intermediaries. It's like a digital financial system where users can lend, borrow, trade, and save directly through software applications.

Working of DeFi:

1. No Middlemen: In DeFi, you don't need a bank or broker to manage your money. Everything is done through smart contracts—self-executing agreements that automatically handle transactions when certain conditions are met.

2. Dapps: Users interact with decentralized apps (dapps), which are built on platforms like Ethereum. These apps provide services like lending, borrowing, and trading without the need for paperwork or bank accounts.
3. Crypto-Based: DeFi mainly uses cryptocurrencies, so you can earn interest, get loans, or trade assets using digital coins like Bitcoin or Ethereum. You can even save or invest your crypto and earn higher interest rates than traditional banks.

Key activities in DeFi:

- a. Lending and Borrowing: You can lend your crypto and earn interest, or take out a loan without going through a bank.
- b. Trading: DeFi allows for peer-to-peer trading, meaning you can directly buy and sell cryptocurrencies without needing a middleman.
- c. Savings: DeFi platforms offer high-interest savings accounts where you can deposit your crypto.

Advantages of DeFi:

- Open to Everyone: Anyone with an internet connection can access DeFi services without needing to open a bank account.
- Fast and Flexible: Transactions happen quickly, and you can move your money without the delays or fees that come with traditional financial systems.
- Transparent: All transactions are recorded on the blockchain, meaning they are publicly visible, making the system more transparent.

DeFi is transforming how people use financial services by making them more accessible, fast, and open