UNIT- II

Block chain Concepts: Introduction, Chaining of Blocks, Hashing, Merkle-Tree, Consensus, Miningand Finalizing Blocks, Currency aka tokens, security on block chain, data storage on block chain,wallets, coding on block chain: smart contracts, peer-to-peer network, types of block chain nodes,risk associated with block chain solutions, life cycle of block chain transaction.

Chaining of Blocks

Introduction: -

We are getting some data that we do not want to change once confirmed. Let us keep capturing the data into a fixed size sets (say 1 KB each) that we will call **blocks**.

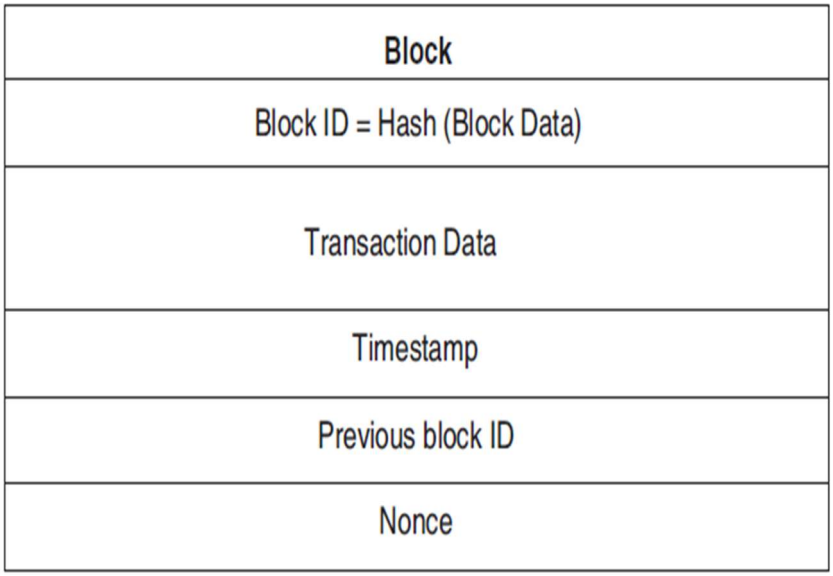
These blocks will get a unique identifier based on the contents of the data. These identifiers are created using something called **hashing**.

Sample Structure of a Block: -

Each block shall have four sections:

-identifier,
-data,
-timestamp, and
-identifier of the previous block.

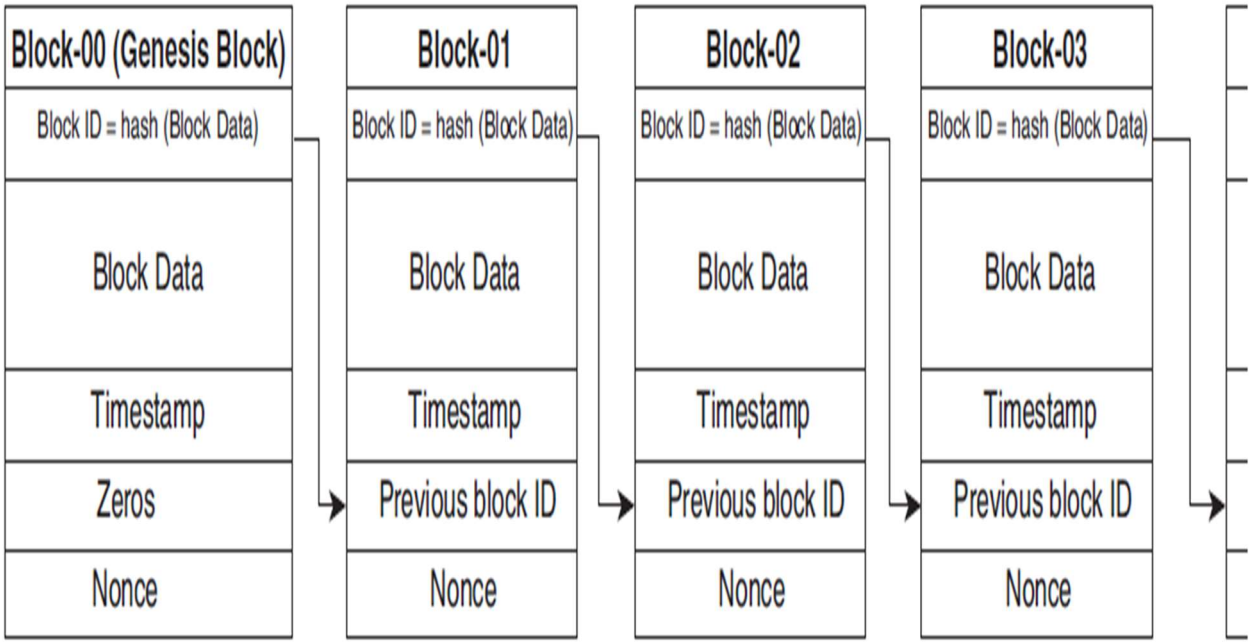
Sample structure of one block as shown below:



For now, let us assume that there is a tool that can provide an identifier for blocks based on the contents of the blocks.

Chaining of Blocks: -

Below diagram shows how blocks are chained together.



In the beginning. we will create the first block: let us call it "BLOCK-00". For this block, we will initialize the "identifier for the previous block" to zeros, and the "timestamp" is set to the current timestamp.

The first block in the chain is usually called **genesis block**. This block does not get confirmed till it gets enough data (1 KB) that it can hold. Whenever 1 KB of data is confirmed, we update "data" of BLOCK-00 to received data. The identifier for BLOCK-00 will be created using contents of the block including "data", "timestamp", and "previous block identifier" of BLOCK-00.

As BLOCK-00 is confirmed, the second block (ie., BLOCK-01) is created with "previous block identifier". The process keeps repeating as many times as required.

Benefit of Creating Block ID using hash function:

Let us look at how this method makes the storage more difficult to change. Let us assume that 10 blocks of data have been created with the above-mentioned method.

Now, if someone intends to update data that is part of BLOCK-07, this change will impact the identifier of BLOCK-07.

This in turn will require changes to BLOCK-08's identifier. This is because BLOCK-08's identifier is derived using creation time- stamp of BLOCK-08, contents of BLOCK-08, and previous block identifier of BLOCK-08 which is BLOCK-07's identifier.

Since BLOCK-07's identifier is changed, identifiers of BLOCK-08, BLOCK-09, and BLOCK-10 will also be changed.

So, deeper the block is in this chain, more changes are required to update to it. All this is possible because of a technique that makes it possible to create identifiers based on the contents of data. This technique is called **cryptographic hash functions**.

It should be noted that the structure of blocks mentioned here is for understanding the concepts. In an actual chain, the number of data items and information stored would be much higher and much more detailed.

Hashing

Introduction: -

Cryptographic hash functions are one of the key components of blockchain. These functions are part of some of the building block functions that help implementation of features such as security, privacy, and consensus on blockchain platforms.

These functions are mathematical algorithms that are used to perform required conversions.

There are different types of cryptographic hash algorithms. **Message Digest Algorithm (MD5)**, **Secure Hash Algorithm (SHA)** are few popular examples, each having their own advantages based on context.

Cryptographic hash functions characteristics: -

Here, we will go through a few characteristics of cryptographic hash functions as shown below, which are useful in the context of blockchain.

-Fixed Output Size
-Deterministic
-Compute Easy
-Pre-Image Resistance
-Second Pre-Image Resistance
-Collision Resistance

1)Fixed Output Size:

In cryptographic hash functions, output string size remains same irrespective of input string size.

Usually, when functions or algorithms run on data, the size of output changes if the size of input changes.

If we take example of popular utility of compressing documents, these utilities have standard underlying algorithms to compress files. If you compress a bigger document, the compressed file size is usually larger compared with compressing a relatively smaller document.

2)Deterministic:

If the input is same, the output will also always be same. This means that if the function is applied on the same input any number of times, the resultant answer will always be same.

That is, any number of times SHA-1 function is executed on string "BLOCKCHAIN HOLDS A GREAT PROMISE", the hash generated will always be same, which is hex string x'82FF9EBDDEF5D1299B0B5C61DE0E245ACB2603160'.

This characteristic helps them to be a very good candidate to be part of the proof-of-work consensus mechanism in the blockchain.

3)Compute Easy:

Creating hash value is not compute intensive; it does not need special hardware and can be completed reasonably fast and not hold up the end user.

This has made these functions popular for signature validation, consensus scenarios for the blockchain.

4)Pre-Image Resistance:

It is not computationally easy to derive input for a given output, say, H1 = hash(string1). In this case, if H1 is given, it is not computationally easy or practical to find string1.

It should be noted that although it is possible, it is just not easy. The stronger the algorithm, the more computation it may require, and it may have better pre-image resistance.

This is the basis of various security mechanisms, where it is not very easy to find source based on data, thus making this a kind of one-way functions.

5)Second Pre-Image Resistance:

It is possible to find another input that can give the same hash value as a given input.
Let a string be given as **string1** and **H1 = hash(string1)**. It is not easy to find another string (i.e., **string2**) such that **string1 ≠ string2** and also **H1 = hash(string2).**

You can try and find any other string that has same hash. To be clear, it may not be impossible, but it will definitely not be easy to do.

6)Collision Resistance:

It is not easy to derive two input values for a hash function such that output value is same for both the input values.

If two inputs map to same output, then the function is said to have **collision**. It is not that collision might not exist, just that probability of that occurring is very less.

So, hash algorithm HASH1 is said to not have collision resistance if you can find two strings, say, string1 and string2, such that string1≠ string2 and HASH1(string1) = HASH1(string2).

This is because it is not that there is no collision in the algorithms, it is just that the probability is very low or it is not easy to crack.

As a matter fact, Google's research has proved that SHA-1 collision is reality.

Merkle Tree

Introduction: -

Merkle tree or **hash tree** is an interesting concept that is used at multiple instances in the verification scenarios in blockchain.

To create Merkle trees, we split source data into smaller chunks, say, Data 1, Data 2, Data 3, Data 4, etc.

After this, the hash value for each of chunk is calculated. We should remember that the size of hash value stays constant and relatively small even if the data chunk is larger in size.

After this, we pair hash values of adjacent data chunks, concatenate them, and create hash value of these hash values.

We keep repeating this process till we have a single hash value left. This single hash value is called **Merkle root.**

Data Verification using Merkle Tree: -

When there is a data to be shared between parties, data is shared from regular channels and Merkle root is shared from/by secured channel.

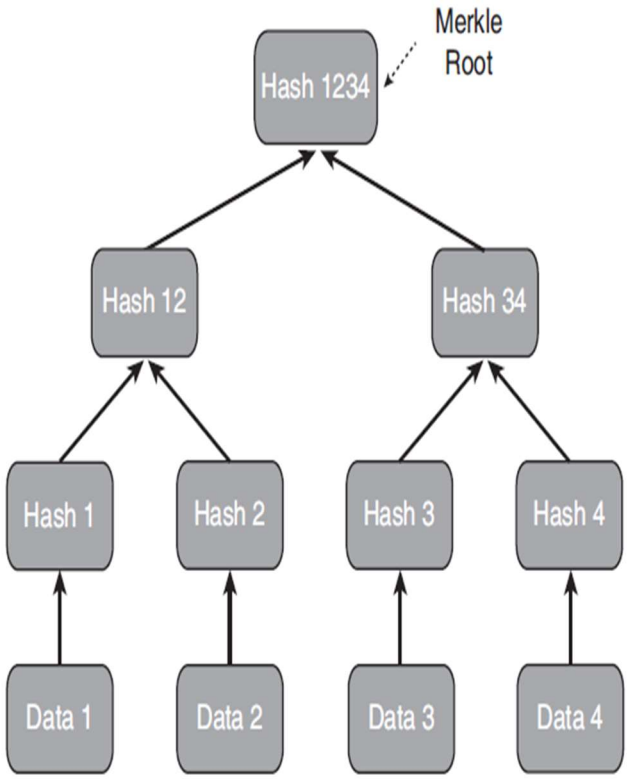
Intermediate hash values can be shared from secure or regular channel as per requirements.

As data is coming in, the verifier can verify if data is valid by calculating hash value of the data received and comparing that with the hash value that has been received for the segment.

After the entire document value of Merkle root is compared with the value of Merkle root received from a secure Source.

If Data is manipulated in the process, value of hash and in turn value of Merkle root would change and this will help the verifier if data received has been authentic or not.

The detailed steps are shown below:



- Split source data into smaller chunks [Data 1, Data 2, Data 3, Data 4]
- Create Hash of data chunks. [Hash 1 = Hash (Data 1)]
- Create Hash of Hash values [Hash 12 = Hash (Hash 1 + Hash 2)]
- Repeat till we just have one hash at top. Call the topmost hash Merkle root
- Share data chunks and hash values through regular source
- Share Merkle root through trusted source
- Verifier can verify data validity using intermediate hash values and documents authenticity using Merkle root by recalculating hash values

This concept of data verification is utilized in blockchain for verification in different use cases in context of blockchain as well as data Security.

Consensus

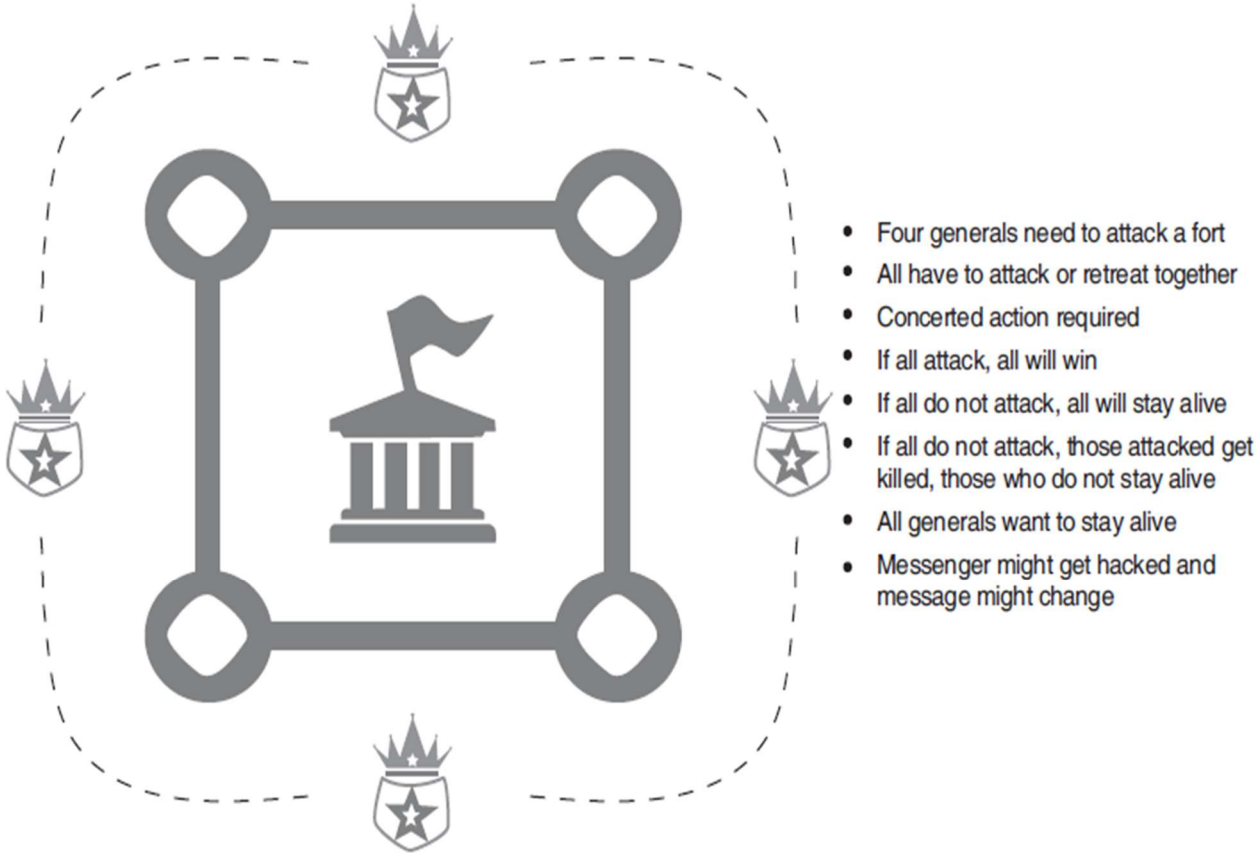
Introduction: -

In blockchain, all participants are ideally supposed to be peers and data on all nodes is supposed to be same. So, there needs to be a mechanism of reaching an agreement where all participants get equal voice while decision-making is done.

Decision-making process where participants agree on decisions of creating a block is called **consensus**.

Byzantine General's problem: -

For the sake of analogy consider the Byzantine General's problem. The Byzantine General's problems discuss a scenario where participants mount decide in union to agree or disagree on a decision, otherwise it is collective loss for all participants



The challenge to be solved is that if one or more participants are not reliable, then how can the entire collection protect themselves from catastrophic failure.

Four generals plan to attack a fort. The conquest requires all general to put on a concerted effort. If all attack together, all will win; if all stay put, status quo is maintained.

If all do not attack together, then only those who attack will be killed. Generals can send messages through messengers but messengers might get hacked or messages might get changed.

Consensus Algorithms:

Blockchain Technology has become a popular means of distributed Ledger System that allows secure and Transparent Transaction Recording System.

To ensure the integrity of these Transactions, Blockchain Networks relay on **Consensus Algorithms** whose job is to ensure that Transactions added to the Blockchain are legitimate and the Network remains secure.

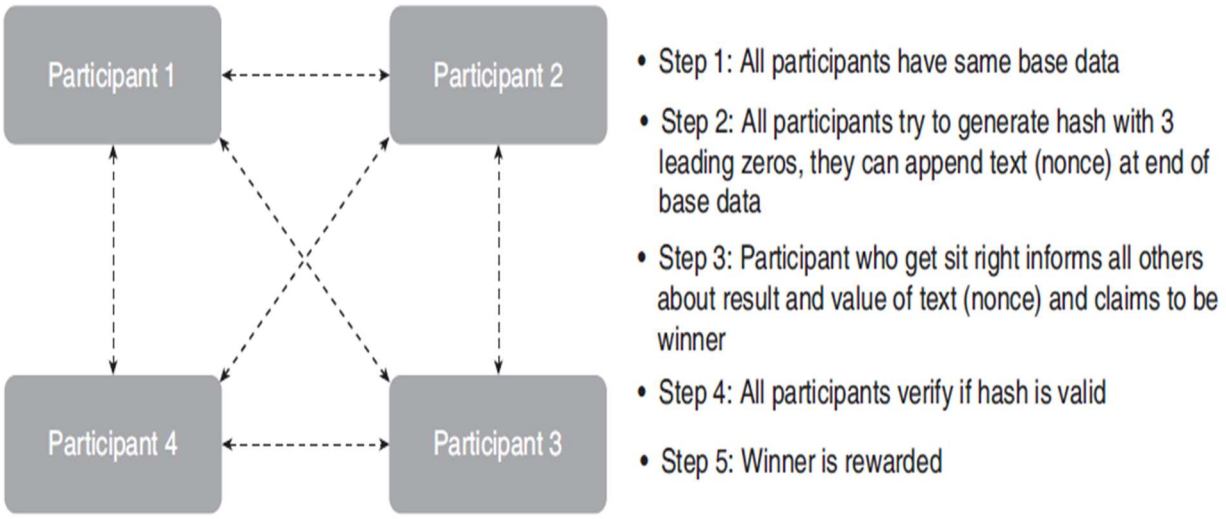
The major Blockchain Consensus Algorithms that are used across the Globe are listed below

1)Proof-of-Work Mechanism
2)Proof-of-Stake Mechanism
3)Delegated Proof-of-Stake Mechanism
4)Byzantine Fault Tolerance Mechanism
5)Sample Directed Acyclic Graphs
6)Proof-Of-Capacity
7)Other Consensus Mechanisms

1)Proof-of-work (PoW) :

Proof-of-work (PoW) is one of the consensus mechanisms in early cryptocurrency blockchain network such as Bitcoin.

The idea here is participants to be joined **should do certain activity and show a proof** that it is done. All other participants shall be able to verify easily that the activity has happened based on some evidence, as shown below:



Proof-of-work mechanism.

Mathematically, we need a **puzzle** that is not easy to crack without brute force. In of PoW consensus, when a block is to be created, all participants are required to add extra bytes to the block called nonce such that hash of the block begins with zeros.

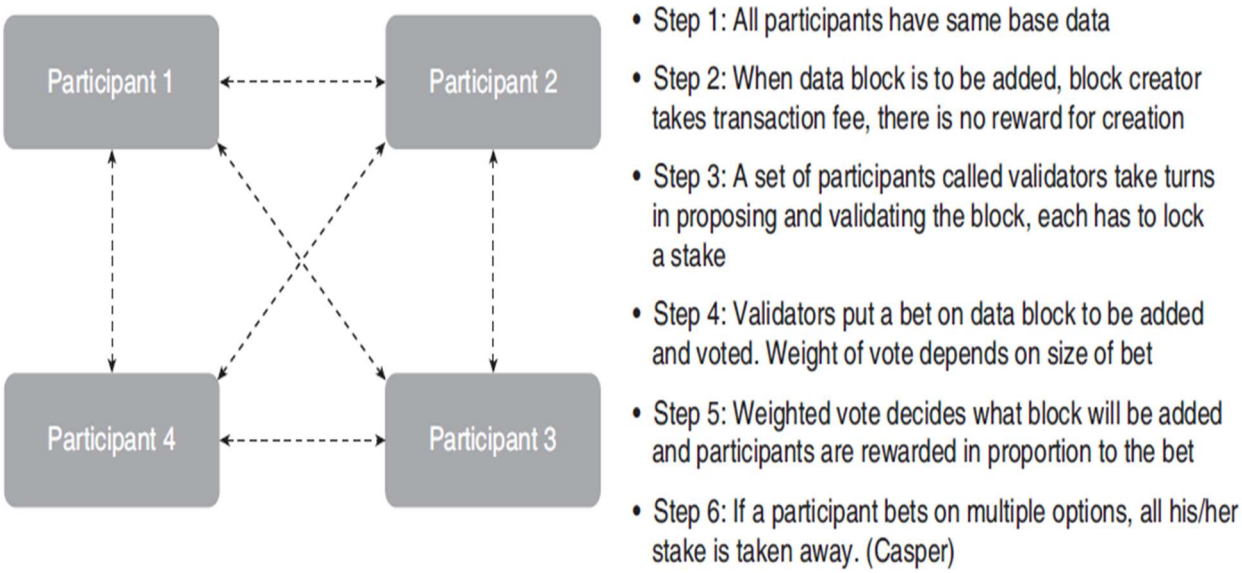
PoW gives equal opportunity for all participants; every participant gets equal chance to prove that they have done the work as long as they have invested some effort (also referred to as compute power). Blockchain networks that use PoW try to incentivize by rewarding those who put more efforts and able to solve the puzzle.

The challenge with PoW is that it is extremely resource intensive, as hash puzzle can be solved only through brute force.

2)Proof-of-Stake Mechanism: -

Instead of making participants solve a computationally costly puzzle, it makes participants place a bet on data block that is to be added to the chain.

PoS makes participants invest whenever they intent to create a block. However, unlike PoW where participants would invest in computing resources, in PoS participants need to invest through cryptocurrency**. PoS is used in Ethereum.**



Proof-of-stake mechanism.

The creator of the block does not get any incentive, but they can charge a fee from data providers.

Block creators publish the block to all participants. Participants, who are also called validators, would bet on next block that it proposed to be added. This bet is considered vote for the block and weight of the block is proportionate to the size of bet they have taken.

Thus, the bigger the bet, the higher the chance of a block getting mined. Participants then bet for the next block that is to be added to the chain.

Once the chosen block is added all participants get rewarded in proportion to their bets. This approach solves the challenge related to the need of extensive compute need.

3)Delegated Proof-of-Stake Mechanism: -

With both PoW and POS, the rate of transaction is less. Also, while both try to give equal opportunity all stakeholders, those who have more resources have a better chance of success rate. In this way, they are not or do not stay as democratic systems.

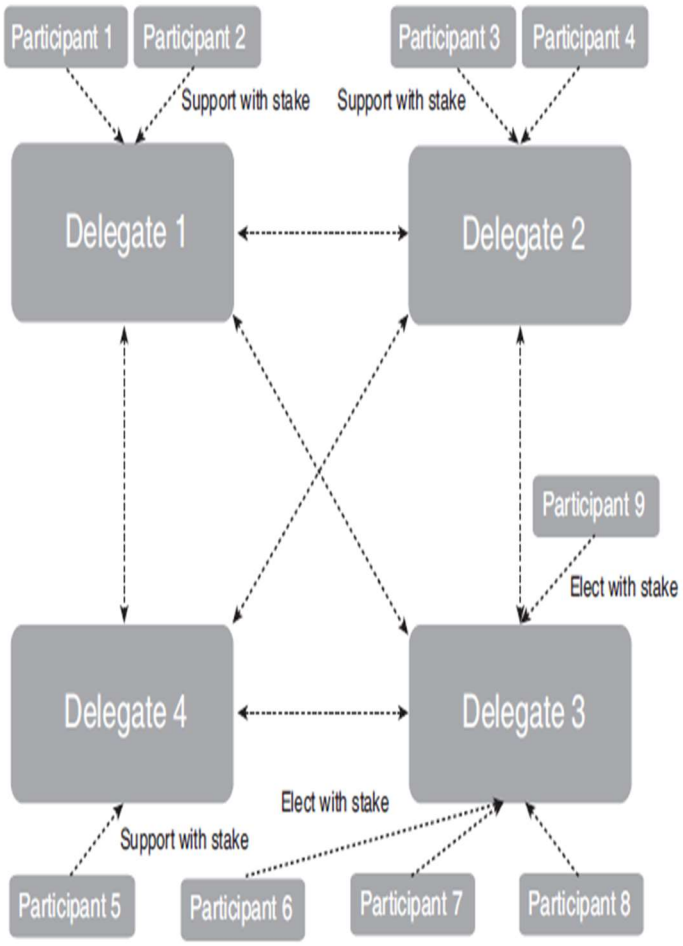
Delegated POS (DPOS) implemented in EOS tries to resolve the issue by randomly selecting and electing a smaller set of delegates.

Participants vote for some of the **delegates** that are randomly selected with a stake. Voting with stakes can also be done to elect a delegate.

Delegates make decisions with 2/3rd majority. With smaller set of delegates reach consensus, the blocks can be created at a much faster rate. Decisions by delegates are bound on all participants. All the participants get rewarded proportionate to the stake they put in. As part of delegates change randomly, this avoids creation of cartels.

The key benefit of DPOS is its performance. With the smaller set to reach consensus, DPOS can have significantly faster transaction rates as compared to PoS and PoW.

The main steps in the Delegated Proof-of-Stake Mechanism are shown below:



- Step 1: All participants have same base data
- Step 2: Participants support smaller set of selected delegates or elect smaller set of elected delegates with stake.
- Step 3: A smaller set of delegates take turns in proposing and validating the block each has to lock a stake
- Step 4: Delegates put a bet on data block to be added and vote. Weight of vote depends on size of bet and support delegates have
- Step 5: Weighted vote decides what block will be added and participants are rewarded in propotional to bet

Delegated proof-of-stake mechanism.

4)Byzantine Fault Tolerance Mechanism: -

BFT mechanism is based on **delegates** that are participants having special hardware. The system appoints a **speaker** among delegates. The speaker is supposed to be different in each transaction cycle, usually, it is created in a round-robin fashion.

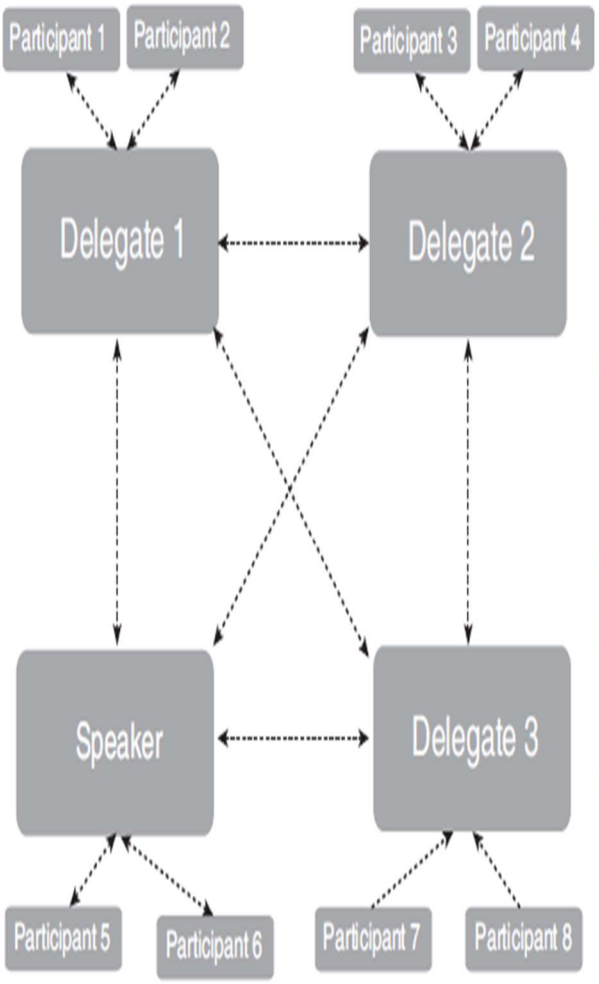
From the data that is received from new participants, block is formed for that by the speaker. The speaker then publishes the new block to all delegates.

The delegates validate the block and when 2/3rd of delegates send confirmation to the speaker, the new block is finalized.

If the speaker is compromised, then 2/3rd of delegates will not agree on the proposal of the speaker. If some of the delegates are compromised and they send invalid confirmation, they will be ignored as 2/3rd of the delegates who are not compromised will not send confirmation

There are multiple variants in Byzantine Fault Tolerance mechanism, three prominent variants are: (1) Practical Byzantine Fault Tolerance (pBFT),
(2) Delegated Byzantine Fault Tolerance (dBFT), and
(3) Federated Byzantine Agreement (FBA).

The main steps in the Byzantine Fault Tolerance mechanism are shown below:



- Step 1: All participants have same base data. Participants send request to network
- Step 2: One of the delegates is chosen as speaker. Different in each round, delegates are participants with better compute power
- Step 3: Speaker creates a block and proposes it to delegates
- Step 4: Delegates validate the block. If 2/3rd delegates accept the proposed change, then it is accepted
- Step 5: All participants get information as it gets confirmed

Byzantine Fault Tolerance mechanism.

(1) Practical Byzantine Fault Tolerance (pBFT):

In pBFT, **membership list** needs to be confirmed beforehand. This makes it challenging to use in public networks where the membership list is dynamic and always changing.

Since the agreements requires multiple communication rounds, pBFT is communication heavy. **This approach works in the private and permissioned blockchain** much better and this is the reason why **PBFT is supported by Hyperledger**. which is a private permission blockchain.

(2) Delegated Byzantine Fault Tolerance (dBFT):

Special nodes in dBFT delegates are known as **consensus nodes**, while regular nodes in dBFT are known as **candidate nodes**. dBFT has preselected consensus nodes, this helps dBFT achieve very high transaction rates. Network **NEO** that implemented dBFT claims that with dBFT, it is possible to reach 10,000 TPS. On the other hand, FBA does

(3) Federated Byzantine Agreement (FBA):

It operates similar to dBFT in a manner that it has a set of delegates that take care of consensus. **This limited set is in turn supported by the rest of the participants.**

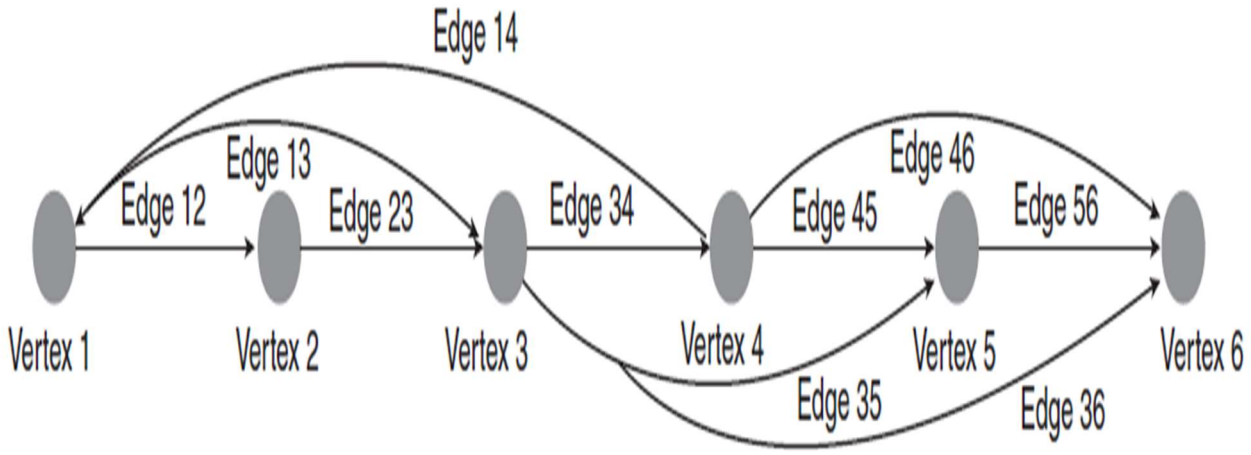
The main difference is in the approach with which delegates reach consensus. Each delegate makes a choice for quorum slice, which is a rule about a smaller group that the delegate believes enough for it to make a decision.

BFT does not require cryptocurrency available out-of-the-box to function, and this makes it very useful in blockchain networks that aim to log information onto ledger than be mechanism to trade cryptocurrencies.

5)Sample Directed Acyclic Graphs: -

The Graph Data structure has nodes or vertices connected by edges or links. In the case of directed graphs, the edges have a direction specified that can be considered as a path from one vertex to another.

In Acyclic Graph, there is no path from a vertex to itself that is going through another vertex/vertices. Basically, in case of DAG, there should not be cyclic loops as shown below:



Sample directed acyclic graphs.

Hashgraph share information through internode gossip. The information about transactions is captured as events and trace of gossip become edges. While hashgraphs are fast with possibilities more than 2,50,000 transactions per second, it can be prone to Sybil attacks some times.

In "block-lattice" protocol of Hashgraph, every account gets a version of chain that it can write to. Vertices are formed by the sender and receiver's accounts and transactions form edges. DAG is a totally different way of doing blockchain and the space is under significant flux

6)Proof-Of-Capacity: -

The proof-of-capacity tries to make participants prove that there is enough space for storage by making them reserve a significant amount of space on a physical disk.

It is also known by different names such as proof-of-space, proof-of-storage, etc. This approach is greener as compared to PoW as it tries to use physical disk space than brute force compute power.

The idea is that the **prover** is supposed to block a significant amount of space and send "proof" for **verifiers** to verify.

The simplest method is that the verifier can send a large file to the prover to store. The verifier can then request the prover to send back information at random positions in the file.

Since the file that is shared is very large, it is impossible for the prover to send back right values unless it physically stores the data.

While this is simple, it will have significant network overhead. In proof-of-capacity, mathematics comes to the rescue with a concept called hard to **pebble graphs,** which is used in different **pebble games played around the world.**

7)Other Consensus Mechanisms

There are numerous other consensus mechanisms as listed below:

a) Proof-of-Activity: The proof-of-activity allows both PoW and PoS to work together on a single chain.

With the fact that some of the blockchain networks have a limited set of cryptocurrency coins that can be created, a hybrid protocol like this can be very useful eventually.

b)Proof-of-Importance: The proof-of-importance aims at prioritizing the transactions from accounts that are more important.

An account is considered important based on how much currency it owns, how many transactions have happened on the account, and for how much time the account has been there on the network. This ensures that importance is given to accounts that transact than those that just hold up currency.

c)Proof-of-Elapsed-Time: The proof-of-elapsed-time was introduced by Intel with a blockchain named IntelLedger.

It is currently supported on the Hyperledger platform. This Consensus algorithm operates in a trusted execution environment and blocking is decided based on the Lottery.

Trusted execution environments such as the Intel's Software Guard Extensions provide guaranteed waiting time and much higher transaction throughput along with scalability.
Having to rely on a third party for processing and functioning comes as downside to all this.

d)Proof-of-Burn: In case of proof-of-burn, the funds are transfer to an address that does not have a valid private key, which means that those funds can never be recovered.

If the currency is burned down, it can be considered as an investment that will encourage network participants to make efforts for survival for the network.

The proof-of-burn looks at the fact that physical resources at the end of day are procured using currency.

e)Proof-of-Authority: The proof-of-authority is very similar to PoS and BFT algorithms. In this case, a smaller predefined set of participants are allowed to decide on consensus.

There might be a predefined order or precedence with the authored group also. To be part of the authorized group, there is supposed to be policy so that all participants feel the standards are equal for all of them.

This works very well for permissioned blockchain networks, where participants are confirmed entities and there is a predefined agreed order.

It should be noted that proof-of-authority has also been utilized in public blockchain, just that the rules to decide on authority are based on the reputation of other factors.

Proof-of-authority also has significant performance benefit as authorized participants are usually not very large in number.

The first downside of this comes mechanism is that the approach is less of decentralization but more of improved centralization as the decision about authority is really managed by the central authority. With lesser no. of participants calling the shots, the mechanism is more prone to security and manipulation risks that are intrinsic to centralization systems.

f)Proof-of-Brain: The proof-of-brain aims to give authority of consensus and rewards for contribution to end users.

The idea is to build consensus and rewards based on how many reviews or likes end users give to the Content. The basic principle of here of crowdsourcing or crowd wisdom.
Proof-of-brain is useful in case there is no firm information source and when the probability of the crowd being right is high. Tokens are generated and distributed to participants. Actors include creators of social content, verifiers of social content, and curators for these tokens.

The downside is that cases need to incentivize truthfulness and there is also some probability that the information captured is invalid or manipulated by the crowd with hidden intentions.

g)Proof-of-Contribution: The proof-of-contribution aims to incentivize donating compute time for large data research programs.

End users who donate compute time are rewarded with tokens. These are very useful where compute can be divided into large number of parallel functions.

These mechanisms are not for regular use cases and in cases where problems cannot really be split into parallel computation tasks.

h)Paxos and Raft: Paxos and raft and two distributed consensus mechanisms that help implement BFT consensus even if some of the nodes are not alive and connected.

In case of Paxos, the nodes are classified into **three roles**. The **proposer nodes** propose a value for acceptance. The **acceptor nodes** accept proposed value from proposers based on rules that are based on indexes of values that are accepted and index the accepted values.

Once the values are accepted, **learners** help propagate the value that is accepted across the network. In this way, Paxos is a relatively complicated mechanism.

Raft tries to simplify the overall mechanism by keeping **only two roles** - leaders and followers. **Leaders** can be chosen round-robin, lottery, or other mechanism and participate in building consensus.

One can observe similarities between Paxos and Raft and other mechanisms mentioned such as pBFT and FBT.

Mining and Finalizing Blocks

Introduction: -

In the blockchain network, **the transaction data is received by network participants**, and each of the nodes tries to ensure that the transaction data they have received is included in the blocks.

While **participant nodes try to compose the block**, they also communicate the transactions to each other. In this manner, each of the participants have their own block composition that they want to all participants to agree on.

Various **consensus mechanisms** help in decision-making, which a participant's block is to be accepted by the network.

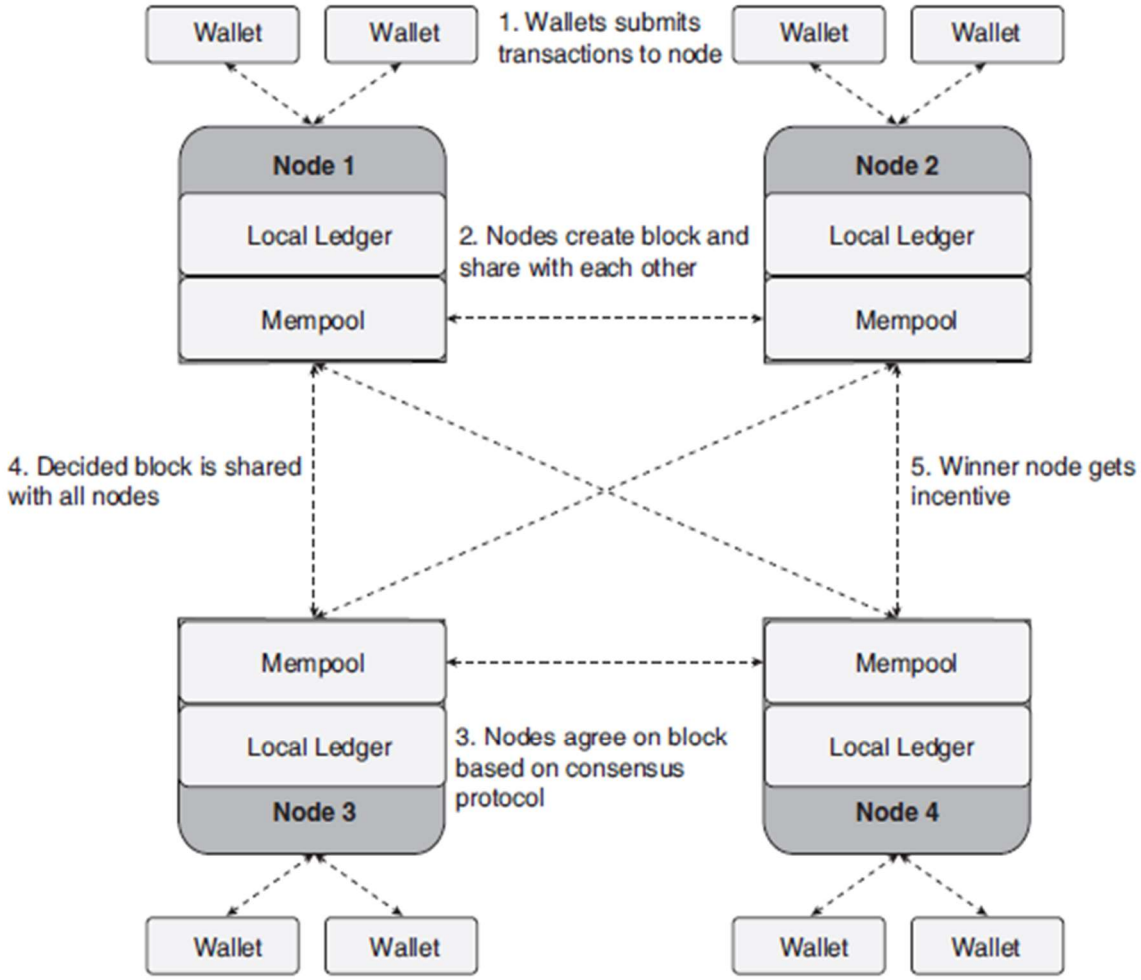
In case of the public blockchain network that supports cryptocurrencies, transactions that perform exchange of currencies form transactional data that is to be put in the block, where each transaction is assigned to a Number called **token.**

The process of creation of the new token and allocating it to the participant whose block is accepted by all participants is called **mining**.

Participant nodes that participate in mining are called **miners**. **Mining** helps the network sustain itself by rewarding participants that put efforts, defines a consistent way in which new tokens are added to the pool, and establishes immutability of the transactions.

Mining Process: -

Let us understand the Concepts and Complications in the Mining Process.



The **mining process** is much more complicated than just reaching consensus and sharing blocks. All the participant nodes are getting transactions in parallel; each wants transactions they have received to be included in the next block, each one is sharing transactions they have received with everyone and at the same time, each participant is also trying to create a block.

To implement this, transactions are first placed in something called **mempool**. When a transaction is received by the network but is not yet confirmed are called **unconfirmed transactions** or **pending transactions**.

Mempool is a transient place where transactions are placed before they are confirmed. In a way, mempool is the location of unconfirmed transactions. Once a transaction is placed in a block that is accepted on the ledger, it is called **confirmed transaction**.

Unconfirmed transactions can stay in mempool for a long time. Transactions that offer a better fee are picked up by miners earlier as compared to transactions that offer a lesser fee.

Blockchain protocols implement checks, incentives, and mechanisms to ensure that Node are working efficiently.

Finalizing Blocks: -

Unconfirmed transactions can stay in mempool till those are confirmed.
The confirmation about transaction submitted by participant to be accepted on immutable ledger chain is called **finality**.

Finality helps ascertain the rate at which transactions become final on the network. Finality is not a property of the consensus mechanism.

Finality is of two types-probabilistic finality or absolute finality.

In the case of **probabilistic finality,** there is a chance or probability associated with the transaction being added to the ledger post as it is received by the minor. Within probabilistic finality, if a sufficient number of validator participants (participants that are verifying transactions) have agreed to include the transaction on the Bock, then it is confirmed.

In case of **absolute finality** scenario, there should be sufficient number of validators to confirm the validity of the block within a specified period of Time.

Probabilistic finality is a property of several consensus mechanisms such as PoW, POS, and DPoS. Similarly, dBFT and PBFT support absolute immediate finality.

Currency Aka Tokens

Introduction: -

It all started with bitcoin, and cryptocurrency still forms a large portion of implemented use cases of blockchain. It is true that there are numerous other use cases where what gets exchanged cannot really be known as a currency.

However, if we consider blockchain as a ledger to transact assets, then blockchain networks that support cryptocurrency are just special cases were the assets are cryptocurrencies that are created and managed on the networks.

When assets get transacted on blockchain networks, they are given digital identities called **tokens**.

A token that is interchangeable is called fungible token. Usually, currency tokens are **fungible**.

A token that is not interchangeable (i.e., in this case we need to individually identify assets and one asset cannot be replaced by another even if it is of same value) is called **non-fungible token**.

The presence of cryptocurrency makes different use cases realizable as well as makes sure that sustainability of those on the network is through reward mechanism.

Rewards can be given to participants in the form of currencies that are managed and exchanged based on the mechanism that is either agreed upon or known to participants.

Cryptocurrency rewards become incentive for participants to ensure that the network survives, becomes popular, and grows.

Security on Blockchain

Introduction: -

While blockchain in principle encourages transparency, it also implements significant technology innovations to implement security at the same time. **Security** can be implemented at various stages- at storage,

-at transport, and

-at end compute level.

Blockchain frameworks utilize public key infrastructure (PKI) in various forms to implement security.

Confidentiality Service is implemented by encryption or encoding functions in cryptography. These functions take two inputs, data that is to be encrypted and a key or passcode; the output will be an encrypted message.

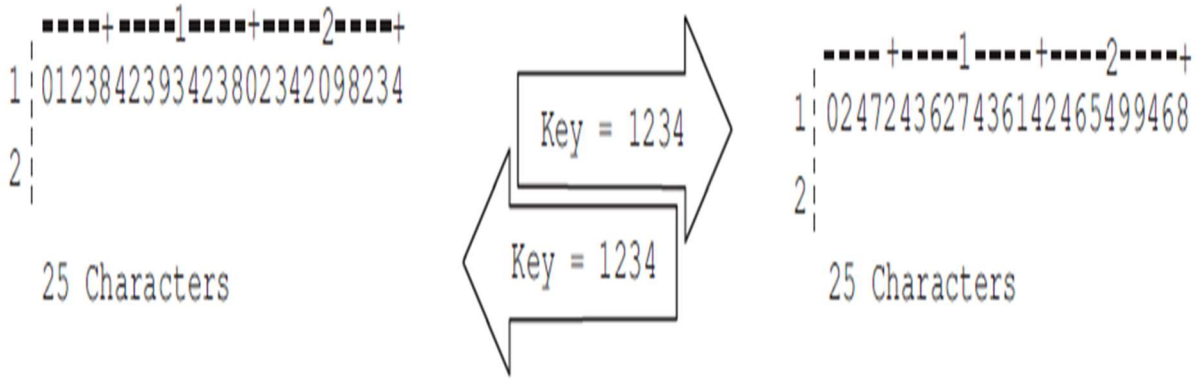
There two types of Encryption Techniques:

- *symmetric cryptography,*
- *Asymmetric cryptography*

Integrity Service is implemented by Hash Functions, Merkle Trees and Digital Signatures.

Symmetric cryptography: -

- In symmetric cryptography, the same passcode is to be used for encryption and decryption as shown below:



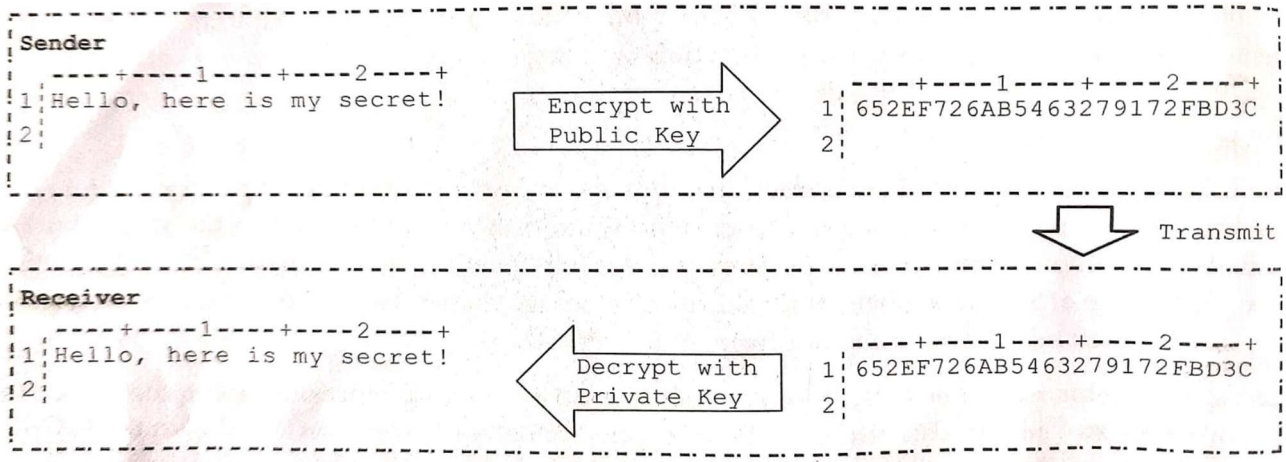
Simple Symmetric Encryption

Example Symmetric Encryption Algorithms include AES, DES etc.

Asymmetric cryptography: -

- Two separate but related keys for encryption and decryption.
- These keys are not the same and they cannot be easily derived from each other. This mechanism of working with a pair of keys is called **asymmetric encryption**.
- One key can be known to everyone, while the other key is known only to a single party. The key known only to the single party is called **private key** and the key that is known to everyone is called **public key**.
- There are various algorithms to create a public key from the private key, examples include **Diffie Hellman Key Exchange Algorithm**.
- Example Asymmetric Encryption includes RSA, ECC etc.

- The Encryption Process is shown below:



Asymmetric cryptography: Securing message through encryption.

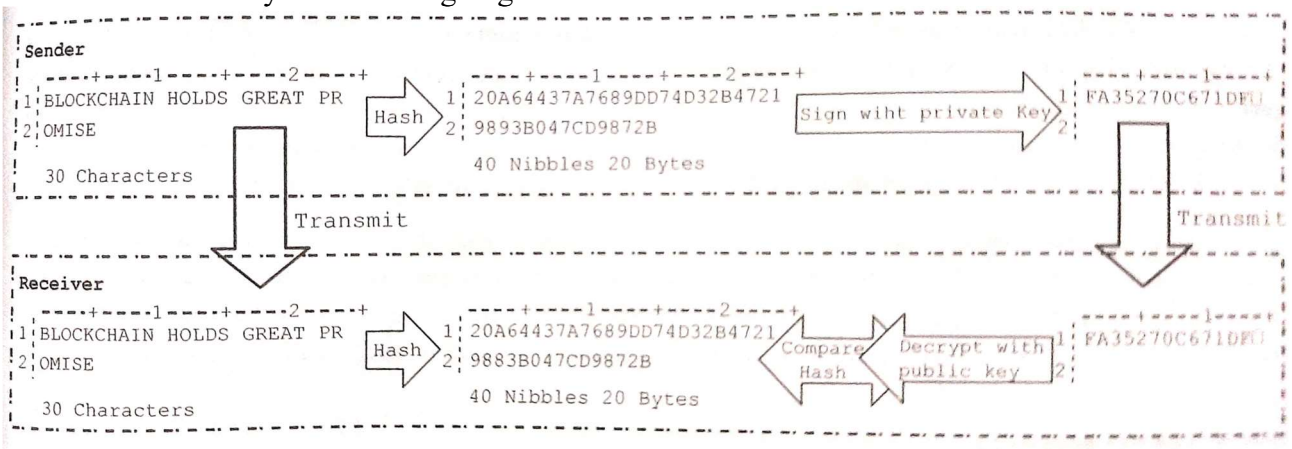
Message Verification: -

when someone wants to share something secure with multiple individuals and the receivers want to verify that the message they have received has been from a valid source and has not been compromised.

In such cases, the sender signs the message using a private key that is known only to him/her. Verifiers decrypt message using a public key that is known to a given set of verifiers and then create hash value of the decrypted message.

The idea is to verify if the message is from a genuine source and that the contents have not been compromised. If the message content gets changed, the verification using public key will fail.

One of the ways in which signing works is as follows:



Asymmetric cryptography: Message verification.

At first, the hash of the message that is being signed is created. Then, the hash value of the message is encrypted using private key of the signer. When the document is shared, both the encrypted hash value as well as the document contents are shared verifier.

The verifier has to read the content of the message and create hash value of the received message. After that the receiver decrypts the encrypted hash value to find the hash value of original message. If the message contents are tampered with, the hash values of original message and shared message not match.

This way, the verifier would know that it does not need to believe the content to be authentic. If the document contents are unchanged, the verification process will see matching hash values. This method of signing a document provides the verifier satisfactory comfort to believe that the message that is shared is valid.

Data Storage on Blockchain

Introduction: -

Blockchain platforms may or may not support cryptocurrencies that are out of the box. Storage of trans actions needs to be optimized for consensus, information sharing, processing, and storage.

With all the participants receiving copy of the data, privacy of information stored along with it is also an important factor.

Generally, there are two models in which the transaction data is stored and processed as shown below:

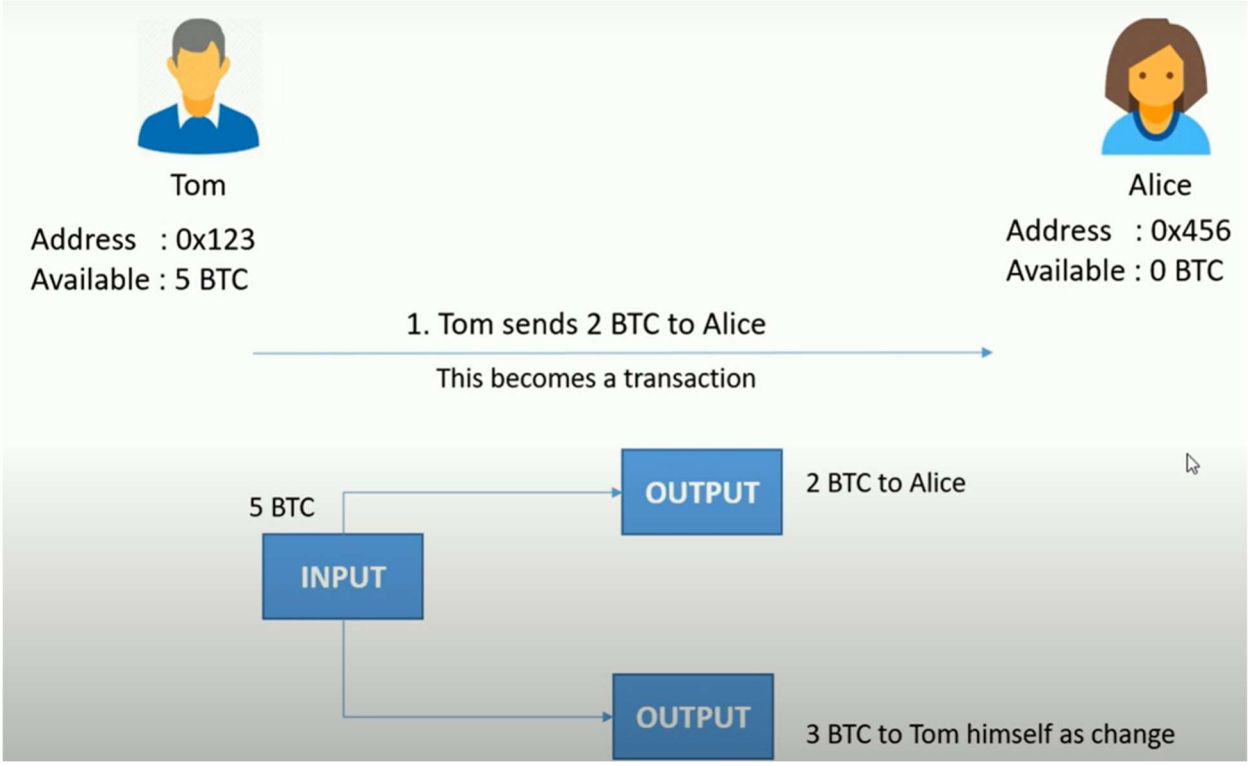
1)UTXO Models
2)Global State Models

1)UTXO Models: -

UTXO stands for Unspent Transaction Output. The idea of this model is to store all transactions on the blocks.

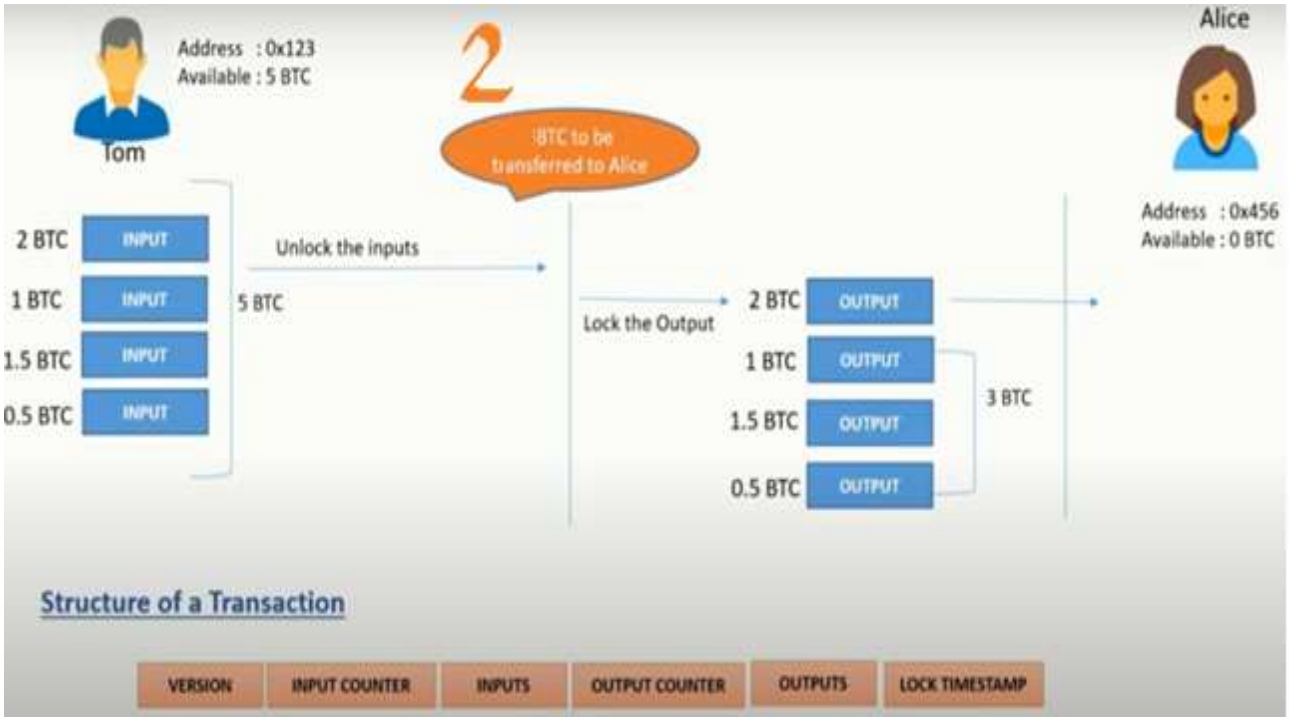
In this method, the account balance is not stored explicitly on the chain. Each time a transaction is to be created, all the UTXOs from all addresses need to be consolidated to feed as input to subsequent transactions. This is done by off chain utilities called *wallets.*

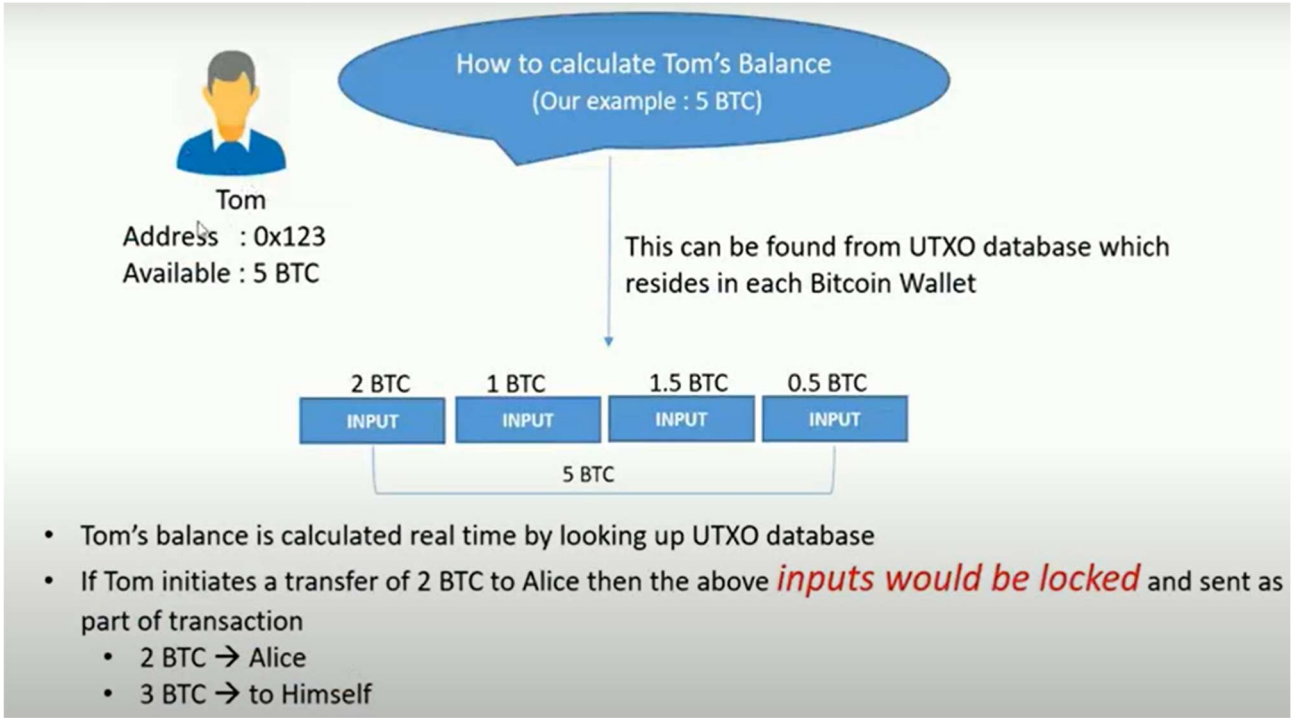
For understanding purpose, let us consider the following example.



Here, Ender User Tom having 5 BTC balance in his Wallet Account want to transfer 2 BTC to Alic Wallet Account.

This Transaction requires a collection of Inputs and a collection Outputs as shown below. An the Transaction Format also shown.





The biggest advantage of the UTXO model is that transactions cannot be replayed on chain as each transaction needs output from a previous transaction as input. This model was used in initial version of **Bitcoin Cryptocurrency**.

2)Global State Models: -

The global state model records transactions and stores the impact of the same outside the transaction. Each Transaction is recorded between accounts and the result is aggregated.

For platforms that support currency platform level such as **Ethereum**, aggregation is done at user accounts.

In the global state model, both Transactions and state of accounts need to be stored at each node.

Wallets

Introduction: -

End users need a way to interact with blockchain Crypto Currencies, which is done by **Wallets**.

It is important that individual identity information is not captured by any of the nodes to avoid privacy breach. Information that is captured also needs to be secured on transit and at rest.

Wallets help establish this by storing three data items - public key,
-private key, and
-wallet address.

Wallet address establishes identity of the wallet with blockchain network. Unlike physical wallets, blockchain wallets do not store cryptocurrency.

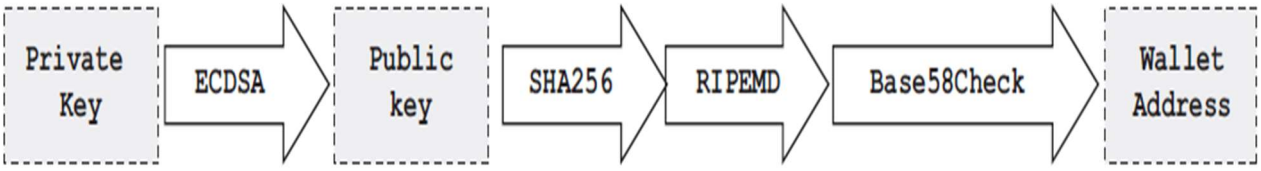
A **private key** must be kept secret and one should be very careful in storing such a key because it is used to perform transactions on block- chain.

A **public key** is associated with a private key and can be shared with everyone; however, it cannot be used to perform transactions on blockchain by individuals. A public key can be shared with others to tell them to share cryptocurrency to the address. A public key is created from a private key using algorithms such as ECDSA.

Wallet addresses: -

Wallet addresses are created from public keys using hashing algorithms such as secure-hashing algorithms, specifically SHA256. As an additional layer of security, Bitcoin creates 166-byte hash from SHA256 hash using RIPEMD-160 algorithm.

This 160-byte address is not easy to read and to solve this, 160 byte output is passed through base58check to convert this number into a readable and smaller number, as shown below:



Wallet address generation in Bitcoin.

The scheme mentioned here is used by Bitcoin and there may be different variant of achieving the same result.

The first step in this scheme is the generation of a private key, and it should be taken care of very seriously. The generation of private keys must be done using secure random algorithms and must not be done in a random way. Websites such as **https://www.bitaddress.org/** can help in the creation of a Bitcoin address and the corresponding private key.

Wallet Types: -

Bed on how private keys and other keys/addresses are generated and managed, wallets can be of different types.

Non-Deterministic Wallets: In the case of non-deterministic wallets, **randomly generated private keys are used and managed**. When multiple private keys to be managed, it becomes very challenging to do so.

Non-deterministic Wallets were introduced in the first version of blockchain. For privacy reasons, new wallet addresses equally generated for transactions. Private Keys are managed from backend using Special hardware devices participating in Trading. Later, non-deterministic wallets are declined because of difficulties to use.

Deterministic Wallets: In case of deterministic wallets, **private keys are created** based on **seed words**. This way, seed words can be utilized to recover private addresses that are part of the wallet in case there is need. This is useful when there is a need to recover the wallet because of failure of the desktop or machine.

It is also easier to member words than remember a string of numbers and letters. While generating keys based on seed words, first a key is generated and then that key is incremented to identify base numbers that are utilized generate the private keys.

When keys are generated based on the master key, which is in turn based on seed word and other keys are based on sequence of keys, which is in turn are based on master key, such wallets are called sequential deterministic wallets.

When the number of keys increase, it becomes challenging manage these also. Thus, there is a need for better organization of keys. For use cases or individuals that want to manage a significant number of keys, the use of sequential wallets become challenging.

Hierarchical deterministic wallets: HD wallets, help organize keys into parent-child relationships of keys. At first, mater public-private key pair is generated using a mnemonic that the end user needs to remember. By applying a counter or similar mathematical mechanism on master private key, child key pairs are generated.

Public keys generated for these child private keys are again organized in similar parent-child relationships. All child private keys can be regenerated using mathematical mechanism. This also allows haring a branch of keys without compromising other branches.

This helps enterprises manage keys based on the organization structure. Transaction signing happens with individual child private keys and not master or parent private keys.

Based on how wallets are used by end users, wallets can be classified into various types.

-Hardware Wallets
- Desktop Wallets
-Browser Extension Wallets

Hardware wallets issue special hardware for an end user to perform transactions. These wallets need end users to physically possess the device to perform transactions. Hardware wallets can store multiple types of cryptocurrencies and tokens in them. **Trezor** is an example of a hardware wallet. If addresses are printed on physical paper, it becomes a paper wallet.

Desktop Wallets are wallets that get installed on desktops and laptops. **Electrum** that introduced deterministic wallets is an example of desktop wallet. Wallets can also provide mobile version of wallets; Electrum also provides a mobile version.

Browser Extension Wallets the wallets are provided as **browser extensions** which are non-hosted and allow end users to transact through any web browser. **Metamask** is one of the examples of browser-based wallet.

Coding on Blockchain: Smart Contracts

Introduction: -

- *Smart contracts* provide a way to store certain data and/or execute certain code on blockchain platforms, which can be initiated based on certain conditions. On Hyperledger platform, this is called chaincode.

- Smart contracts execute on all nodes of a blockchain network. They differ from regular programs in various ways.

- When a smart contract is deployed, similar transactions get propagated on all nodes of the blockchain and smart contracts also get propagated.

- The nature of smart contracts also needs specialized constructs on nodes to consolidate information from the outside blockchain and connect with blockchain smart contracts.

- One of such constructs is ***oracles*****,** which is a routine that executes in the blockchain, consolidates information from multiple sources, and pushes that information to smart contracts.

Peer-to-Peer Networks

Introduction: -

- Based on the responsibility of network participants to support distributed applications, networks can be classified as - Client–Server Networks Model
  - Peer-to-Peer Networks Model.

- In **Client–Server architecture,** there are a set of servers tasked with certain responsibility such as providing data or performing certain functionality, other systems can act as clients to consume "services" provided by the servers.

- In P**eer-to-Peer Networks**, all participating systems are of equal authority or status. All participants can act as clients as well as servers for each other.

- Blockchain networks are peer-to-peer networks.

- Blockchain conceptually considers participants to be peers with equal privileges.

- There is no centralized server or authority to approve or reject any operation.

- This makes peer-to-peer networks an obvious choice for blockchain networks.

Types of Blockchain Nodes

Introduction: -

Blockchain is built based on peer-to-peer network, not all participating nodes have capability or intent to perform all the functionality, to perform all the activities and transactions on nodes.

Based on the capability of nodes to perform transactions or activities on blockchain nodes, these can be classified in different ways:

-Miner Nodes
-Full Nodes
-Administrator / Super Nodes
-Light-Weight Node

Types of Blockchain Nodes Diagram is shown below:

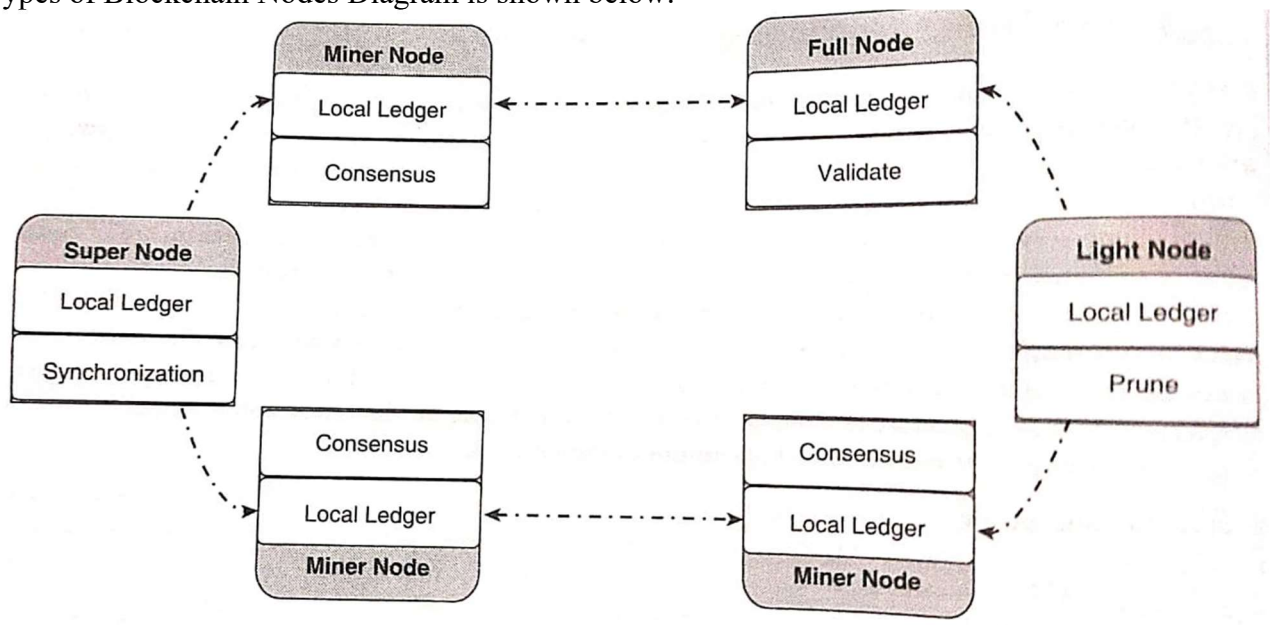


Figure 3.9 Types of blockchain nodes.

1)Miner Nodes: These are nodes that can perform almost all the activities on blockchain networks. Miner nodes have the full copy of ledger with them.

These nodes participate to capture transactions, create or propose blocks, participate in building consensus, finalize transactions, and synchronize their ledger copies.

2)Full Nodes: Full nodes store full copies of ledger and validate the newly added blocks. Full nodes are primarily responsible for verification of transactions that are created by miners.

Full nodes do not perform responsibility of the miner nodes.

3) Administrator / Super Nodes: Administrator nodes perform administration activities and support other nodes for certain activities.

These archival nodes depending upon the role they play in the network. While capabilities such as adding a node are coded in smart contracts, these nodes still end up having different responsibilities and capabilities than other nodes.

4)Light-Weight Node: Light-weight nodes do not store full ledger but only block headers. These nodes verify validity of transactions.

As these nodes do not store the entire ledger, they need lesser resources and are relatively easier to run.

Usually, these nodes support end user applications such as wallets.

Risk Associated with Blockchain Solutions

Introduction: -

Blockchain provides an alternate way of doing business. Moreover, it creates immense opportunities in tracking and data sharing transaction management, all in a secure manner.

Blockchain-based solutions are exposed to a unique set of risks that traditional solutions might or might not be exposed.

All the network players should be aware of these risks and prepare for right mitigation plans for the same. **The risks have been categorized into four broad groups as listed below:**

1)Business Impact-Related Risks
2)Project Management-Related Risks
3)Technology Usage-Related Risks
4)Regularity and Compliance Risks

1)Business Impact-Related Risks: -

Blockchain introduces technology that provides alternate ways of doing business. This also means that solutions using blockchain are not just automation or optimization of business processes but can mean creation of new processes.

While this disruption of the status quo creates new opportunities, it also introduces business to unknown territory, thus creating new and unforeseen risks.

While blockchain provides redundancy by design for larger networks, for early movers with smaller networks, business continuity risks are higher because of the evolving nature of technology as well as possible recovery processes.

Lack of awareness or overeager passion to use blockchain can also make an organization choose a use case that is incorrect for the blockchain or irrelevant for the market.

Mitigation strategy for business impact-related risks is to evaluate business value through smaller scale implementation that would provide additional validation of the business idea. This needs to be accompanied with the initial scenario planning market research, and a process to look for possible disruption coming in to the market.

Some blockchain organizations have also taken the route of raising money through ICOs to attract investors early on with the possible higher risk-higher reward scenario. ICOs aim to raise funds for blockchain solutions with the hope that with an increased acceptance of the use case by market, solutions will be in demand and a cryptocurrency required to participate in solutions will increase its market value.

2)Project Management-Related Risks: -

Blockchain is one of the emerging technologies that is evolving very fast. While adoption of such technologies gives organizations early mover advantage as well as to reap benefits of less competition, **there are significant chances of cost and effort overruns.**

In short, it can be a longer journey in unknown territory for organizations. This is because estimation models are evolving, and the **amount of efforts required is changing due to introduction of tool set on one side and best practices on the other**.

This dynamism creates challenges for established project management practices where most of the cost of effort is known upfront. The journey will also expose programs to unforeseen functional and technical changes.

The project manager needs to actively engage with stakeholders and sponsors throughout the journey because without their support, long-haul programs would typically die on a critical juncture. Apart from stakeholder engagement, mitigation requires management to keep smaller and at the same time, complete release cycles that introduce risks early without significant financial loss.

3)Technology Usage-Related Risks: -

Most of the business impact as well as project management related risks have a root in uncertainty with regards to the technology capability of blockchain at certain points of time. **While conceptually and theoretically, things are supposed to be in a certain manner, technology is just not there as yet.**

For example, debugging issues in smart contracts is a tedious task, with no system output visible to developers. This is also making developers come up with newer and better versions of tool set as well as platform features.

This fast technology evolution also exposes organizations to risk that as by the time a use case is implemented, the technology becomes outdated.

The challenge in debugging issues and asynchronous mode of operations also increase possibility of **security vulnerabilities** through a seemingly insignificant code in smart contracts.

Mitigation strategies for technology risk is to develop solutions in such a way that it shall be easy to migrate if there is a need.

The strategy of right-sizing release cycles is also helpful. Another way is to utilize frameworks and best practices (e.g., OpenZeppelin, Mocha, etc.) in the software development life cycle.

4)Regularity and Compliance Risks: -

Shared ledger means data sharing, and not all the data sharing scenarios are legal in current regulatory scenarios.

With governments wanting to have data residency assured within their jurisdiction, certain international use cases become invalid or illegal.

While blockchain gives control to end users, end customers might be more vulnerable to theft or credential compromise as compared to most organizations.

Regulators are continuously learning about the ecosystem and making changes to protect the interest of masses. This has an indirect impact on investments in blockchain-based solutions.

For **example**, **cryptocurrency is not a legal tender in lot of countries.** Real estate trading on blockchain does not fit the legal framework in every country in the current state, and securities can only be traded by licensed agencies in some jurisdictions.

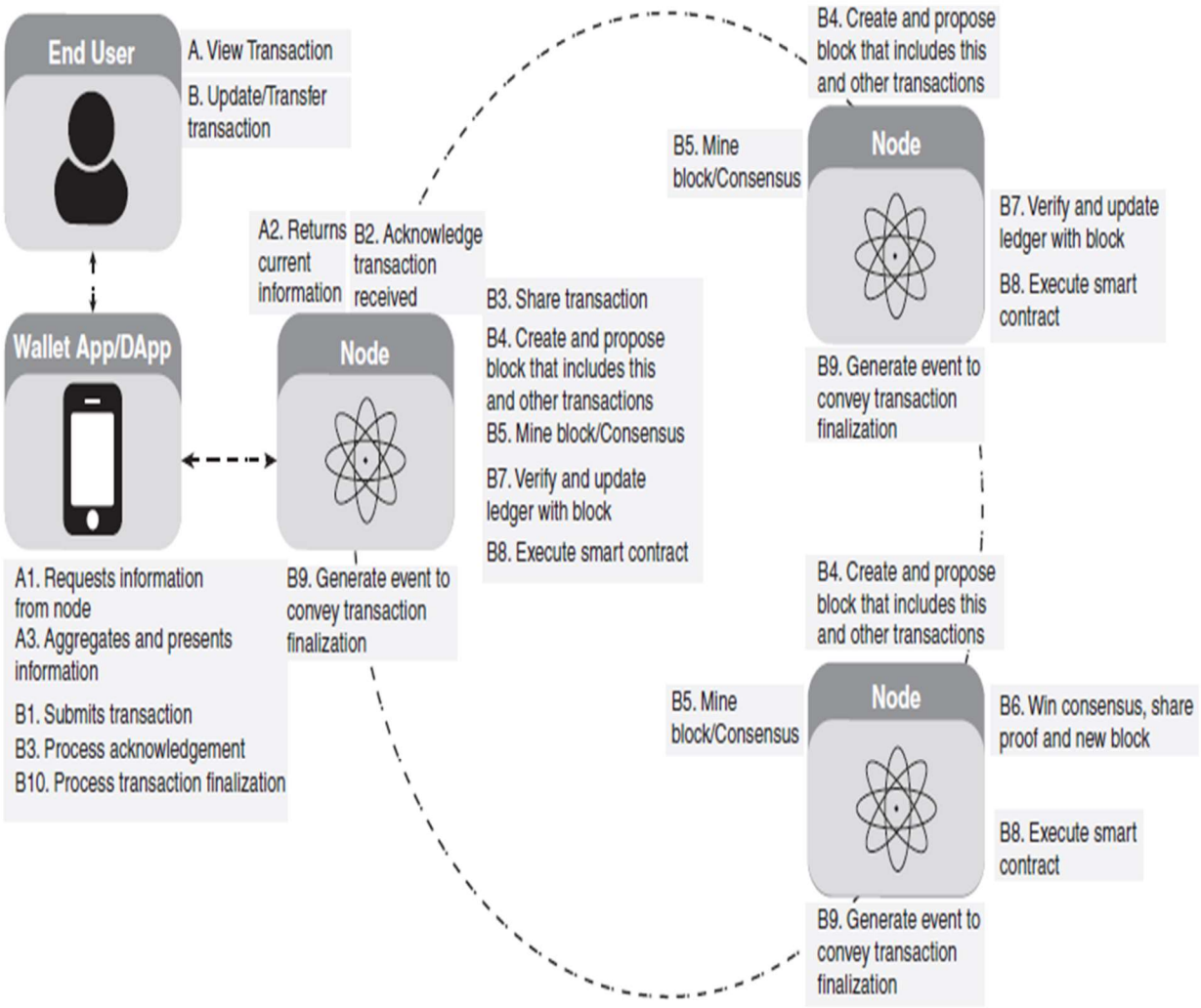
Mitigation of regulatory risks can happen in this space through entrepreneurs engaging with regulatory agencies, respecting regulatory concerns, and coming back with solutions that fit in the legal framework of a government.

It is important that organizations should be in continuous lookout for what is going on in the market and take decisions about participation through a well-devised evaluation model that matches their culture.

Life Cycle of Blockchain Transaction

Introduction: -

Let us discuss how a transaction works. Two transactions namely Information view transactions and Update transactions are explained using the diagram shown below to give a view on how blockchain solutions differ from traditional software systems.



Information view transactions: -

For the information view transactions, wallet application or DApp issues request to a blockchain node to receive information.

The blockchain node returns the requested information. DApp interface may aggregate information coming out from blockchain node to give a consolidated view.

An example of this is a wallet summing up UTXOs for an address to provide end user view of how much balance the individual has to spend.

For networks such as Ethereum, the interface can query the information directly from the network. The interfaces also look for events generated to prepare a better view for end users.

Update transactions: -

Update transactions such as fund transfer work in a slightly complicated manner. Transaction from end user is captured by the interface, signed using private keys of the end user, and submitted to a blockchain node.

The blockchain network acknowledges receipt of transaction, and then the transaction gets shared with all the nodes. All the nodes try to create and propose blocks and for Proposals based on the type of consensus that is trying to finalize the block.

It is only after finalization of the blocks that it can be considered that the transaction is complete. When transactions are agreed upon the network through consensus mechanism defined, smart contract codes can execute based on the trigger or rule that is defined.

The smart contract code can generate events that can communicate back to the DApp client, which can then take note of the fact that transaction has been finalized.

The reason this whole process is elaborated here is because this has significant impact on how applications are architected and designed.

Systems need to be aware of the asynchronous nature of functioning, and they should be looking for and be responding to events and have user interface that keeps users informed about what is happening without creating inconvenience.

----------*********---------