1 INTRODUCTION TO BLOCKCHAIN

MODULE - I

Syllabus

1.1 What is a blockchain, Origin of blockchain (cryptographically secure hash functions), Foundation of blockchain: Merkle trees

1.2 Components of blockchain, Block in blockchain, Types: Public, Private, and Consortium, Consensus Protocol, Limitations and Challenges of blockchain

1.1 Introduction

Today, over fifty percent of the world is connected via internet. Technology is also progressing with Machine Learning, Artificial Intelligence, Virtual Reality, and Big Data. One of these technologies is blockchain which is the backbone technology for digital cryptocurrency bitcoin. In order to understand a blockchain, understanding the distributed system is necessary. Blockchain is the decentralized distributed system. In distributed computation, two or more nodes work with each other in a coordinated way to achieve a common outcome and nodes share the resources available on other nodes. A computer network is defined as a set of communicating devices that are connected together by communication links. These devices include computers, printers and other devices capable of sending and/or receiving information from other devices on the network. These devices often called as node in the network. So, computer network is interconnected set of autonomous computers. A distributed system is defined as set of autonomous computers that appears to its users as a single coherent system. Users of distributed system feel that, they are working with a single system.

In 2008, Santoshi Nakamoto released a white paper titled "A peer-to-peer Electronic Cash System". The knowledge of electronic cash is important in order to understand the very first application of blockchain which is bitcoin or cryptocurrencies. Consensus algorithms in distributed system offers the basis of the practical implementation of Proof-of-Work algorithms in bitcoin. There are thousands of digital currencies apart from bitcoins. All these currencies are called as altcoins or coins which are used for buying or selling the products or services. It is noticed that, till October 2019, more than 3000 active cryptocurrencies exist. Now days, cryptocurrency concept is applicable in almost all domains like finance, security, supply chain management, ownership, health management, records management and in many cases which requires faster fund raising such as start-up ventures. Now days, Blockchain is acceptable and important technology as no business operates in separation. Multiple institutions can accomplish more together than any single institution can alone.

1.2 Origin of Blockchain

In 2009, Santoshi Nakamoto implemented first application of blockchain called as bitcoin. Blockchain and bitcoin are not same. Bitcoin was built on blockchain technology. Apart from Bitcoin Blockchain, there are so many blockchains such as Ethereum Blockchain and other private blockchains that are known and adapted as per various industry applications.

1.2.1 What is Blockchain?

Blockchain can be defined in many ways depending on how you think about it. If you define it from a business perspective then its definition will be in that context. if you define it from a technical viewpoint one can define it in view of that. Some of the definitions are given as below.

Definitions :

1. Blockchain is a peer-to-peer distributed ledger or distributed database that is cryptographically secure, append-only, immutable (really hard to alter it), and updateable only through consensus or agreement among peers.
2. Blockchain is distributed, replicated databases or ledgers which exists on all the nodes of peer-to-peer networks and these databases permits many non-trusting parties to carry out transactions without a trusted intermediatory and maintains records which are immutable, time sequenced and tamper-resistant.
3. Blockchain is distributed ledger that records transactions and maintains verifiable and permanent record-set of information.
4. Blockchain technology is a software; a protocol for the secure transfer of unique instances of value (eg, money, property, contracts, and identity credentials) via the internet without requiring a third-party intermediary such as bank or government.
5. Blockchain is a decentralized computation and information sharing platform that enables multiple authoritative domains, who do not trust each other, to cooperate , to coordinate and collaborate in a rational decision making process.

1.2.2 Blockchain and Bitcoin

Blockchain is database of records related to various transactions and bitcoin is a cryptocurrency. The aim towards bitcoin creation is to minimise government's control over cross-border transactions and to speed up transaction process without involvement of any third party intermediatory. Blockchain offers secured environment for Bitcoin transactions. In other words, Blockchain maintains all the transactions of the Bitcoin. Bitcoin is related to currency transactions whereas Blockchain is applicable in many domains such as finance, security, supply chain management, ownership, health management, records management and research management. In bitcoin transactions, it is impossible to identify users involved in it. Whereas, Blockchain is known to everyone as it is applicable in many industrial applications.

1.2.3 Blockchain Evolution

- In the 1960s, people were using mainframe computer system as a computing facility. The mainframe computer system would be normally kept in a computer center with a controlled environment. The users used to prepare their program as a deck of punched cards with encoded list of program instructions. In the decade of 1970-80, there was an exceptional growth in bulk storage. It has helped to realize the concept to propose the idea of extended storage.
- Again, the concept of "virtual storage" helped to utilize this extended storage as an enhanced address space. In 1973 public key cryptography was implemented. Concept of hash tree was patented in 1979. In 1997, Proof-of-Work with Hashcash was introduced. In 1996, bit gold as a mechanism for a decentralized digital currency and smart contract was introduced by Nick Szabo.

- In 2000, a general cryptographic theory of secured chains was introduced by Stefan Konst. In 2008, Santoshi Nakamoto released a white paper titled "A peer-to-peer Electronic Cash System". In 2009, mining of Bitcoin Genesis Block was carried out and first application of public Blockchain was launched. In the same year, a Bitcoin was recognized as digital currency.
- The first Bitcoin purchase was carried out in 2010. It was 10,000 BTC for $25 pizza. In 2011, Namecoin and Litecoin was released. The first decentralized social network Diaspora was launched in the year 2012. The Ripple, a permissioned Blockchain and also Bitcoin Foundation was launched in the same year 2012.
- In 2013, Bitcoin market place exceeds $1 billion. The first Bitcoin ATM was unveiled in the same year. Also, Ethereum white paper was released in 2013. In 2014, R3 which is consortium of the above 40 financial institutions had decided to implement the Blockchain Technology.
- In the same year, Microsoft accepted a Bitcoin and Ethereum Blockchain was funded by crowdsale. In 2015, a Genesis block in Ethereum was created. In the same year, Linux Foundation introduced Hyperledger for speeding up the Blockchain development. In 2016, there was a theft of $50 M in ether due to bug identified in Ethereum DAO code. In 2017, a new Blockchain protocol introduced for industry-scale decentralized applications. It was noticed that, in 2018 current Bitcoin market place was in between $10 to $20 billion. TRON platform was introduced for entertainment industry.

1.2.4 Blockchain Generations

- First generation Blockchain is considered in the period between 2008 to 2013. In this generation, concept of Bitcoin was introduced. In 2009, Santoshi Nakamoto implemented first application of blockchain called as bitcoin and he has formed the Genesis block. The first successful bitcoin transaction was executed in the same year. The first generation Blockchain offers transparency, security, immutability and accountability in transactions. The protocols require heavy mining hardware and huge number of resources. Because of this reason, its hard to offer scalability, interoperability and speed of processing.
- Second generation Blockchain is considered in the period between 2013 to 2015. This generation attempted to overcome the limitations of Bitcoin. It was realized that, the same Bitcoin technology was applicable to the all classes of business to customer and other applications. Ethereum was introduced in 2013 and it needs lesser energy to maintain. Although scalability is still an issue for the Ethereum, smart contracts introduced new functionalities.
- Third generation Blockchain is considered in the period between 2015 to 2018. In this generation, Linux Foundation introduced Hyperledger for speeding up the Blockchain development. Decentralized applications (DApps) of Ethereum were also introduced in this generation. Although Hyperledger platform can be plugged in any consensus mechanism, Smart Contracts of Ethereum can be used in Proof-of-Stake consensus mechanism. This generation Blockchain has given more attention towards consensus mechanism in order to accomplish the interoperability and increase the speed. Scalability is still an issue for the Ethereum. As a result, EOS and TRON have taken over the DApps.
- Fourth generation Blockchain is considered from 2018 onwards. Here more focus is on improving the speed along with scaling the blockchain with better security.

1.3 Blockchain Solution

When we deal with transactions with digital currency, the double-spending problem is more common to arise. Double-spending means just spending the money more than once. It happens just by duplicating the token or crypto-coin and reusing it.

1.3.1 Traditional vs. Blockchain Transactions

- Before use of computers for transactions, people were using Paper-based physical ledgers to record the transaction entries. This was more time-consuming recording and maintenance, more error prone. Also, paper-based ledgers are with lack of security and with limited number of copies which cannot fulfil need of large organizations.
- Digital ledgers are digital file or files or database which are manipulated only through computer program written for this purpose and not manually. Consider the example of transactions with different scenarios given below. Consider User a purchase book from User B for $100.

Scenario 1 : Physical Transaction

User A hand over physical currency of $100 to User B and then User B hand over Book to User A.

Scenario 2 : Online Transaction

Above same transaction, both Users A and B want to carry out online as they are unable to meet. In this case bank acts as intermediatory. Bank debits $100 from User A's account and credit it to User B's account. Once User B confirms that his account is credited, he sends book to User A. Banks database or ledger records these credit and debit entries. Only bank has access to this database. In this case, bank works as trusted third party and it may charge transaction fee. Now days, many banks and exchanges for cross-border transactions involve large amount of money. Hence, financial institutions have to perform transaction maintenance and verify ownership. They have to also resolve any dispute if happens. Hence, institutions will charge to the customer for offering services.

Scenario 3 : Blockchain Transaction

In this case, User A and User B both are members of Blockchain network. In this case User A initiate the transaction with $100 equivalent cryptocurrency say bitcoin. Now, Blockchain network validates this transaction. After validation, User B receives $100 equivalent cryptocurrency. There is no any intermediatory and slight amount of fee will be given to miner which very less compared to bank fee. In this way, there is inherent trust involved in Blockchain transaction. This blockchain data is not only money but it can be data in any other domains.

1.3.2 Peer-to-Peer Network

- In peer-to-peer network all the nodes are equal and each node plays role of client and server with other nodes. In this network, multiple copies of same data are stored on different nodes in the network. As data is available on multiple nodes, there is no chances of single point of failure like in traditional client server model.
- Failure of a single node does not affect the data as multiple copies of the same data remain safely and securely available on other nodes. If one copy of data changes or becomes inaccurate, on majority of nodes/peers exact and accurate copies exists which makes false record outdated. Following Fig 1.3.1 shows traditional client-server network. In this network each client request services from centralized server.

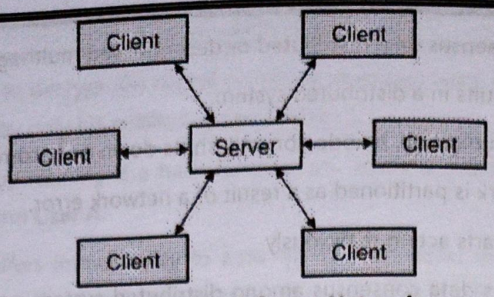


Fig. 1.3.1 : Client-Server Network

- There is no centralized administrative system in peer-to-peer network. The resources available on other nodes are shared by each node or peer. In Blockchain technology peer-to-peer architecture, every node maintains complete updated copy of blockchain ledger or database. It helps to reach on consensus about the actual state of distributed ledger.
- As a result, it assures decentralization, security and resistant to several cyber-attacks. Apart from this, agreement is needed between majorities of nodes to take decision about to add the data block in Blockchain. Therefore, it is almost impossible for attacker to change data in bigger networks such as Bitcoin, Ethereum etc. Hence, consensus in widespread distributed peer-to-peer network assures high degree of resistant to many attackers and malicious activities. Following is the Fig. 1.3.2 that shows peer-to-peer network where each peer or node communicates with other peer as a client or server.

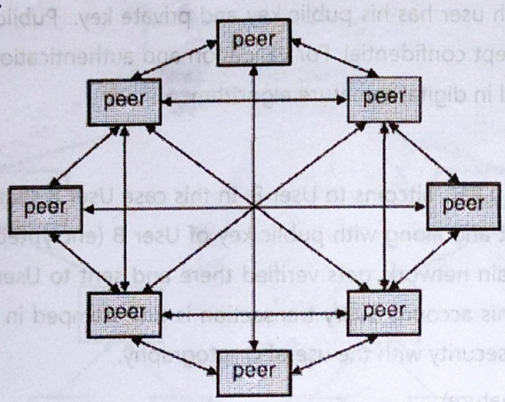


Fig. 1.3.2 : Peer-to-Peer Network

1.3.3 Asymmetric Cryptography

This is also called as public Key cryptography. In this cryptography, two keys are generated which are known as public and private keys. If message is encrypted by using public key, then the same key cannot be used to decrypt the same encrypted message. If message is encrypted by using private key, then same key cannot be used to decrypt the same encrypted message. It means that, both keys need to be used in combination. Hence, confidentiality and integrity are assured with this combination of public and private keys.

1.3.4 Distributed Consensus

- The distributed consensus protocols are more helpful to prevent cyber-attackers. Hashcash which has used secure hash algorithm 1 (SHA-1) served as the motivation in Proof-of-Work consensus mechanism within Bitcoin distributed ledger.

- The process of coming to a consensus on a distributed or decentralised multi-agent platform.
- There can be various types of faults in a distributed system.
  - Crash fault- During a communication, a node abruptly shuts down or becomes inaccessible.
  - Network Faults - The network is partitioned as a result of a network error.
  - Byzantine Faults - A node starts acting maliciously
- A distributed consensus ensures data consensus among distributed system nodes or reaches an agreement on a proposal.
- In together all these technologies : Peer-to-Peer network, public key cryptography and consensus protocols are backbone for the success of new cutting-edge technology called as Blockchain.

Working steps of Blockchain

- Authentication proves identity of the user and authorization provide rights to user to carry out his/her intended work. Both are useful for digital trust in Blockchain. Authentication of the participating member in Blockchain is proved with public-key cryptography in which two keys are generated: public and private keys.
- If user creates private key, then its public key pair is also gets created. Both keys are then stored in wallet which is nothing but software program. This wallet is also used to send and receive the cryptocurrency in Blockchain.
- In public key cryptography, each user has his public key and private key. Public key can be shared with any other user but private key is always kept confidential. For validation and authentication of data, digital signature is used. Following are the steps involved in digital signature algorithm.

Step 1 : Generation of Keys

Suppose User A want to send 1.0 BTC bitcoins to User B. In this case User B shares his public key with User A. User A then takes 1.0 BTC from his wallet and along with public key of User B (encrypted message) sends the request. This transaction passes through blockchain network, gets verified there and sent to User B. User B uses his private key for decryption and receives 1.0 BTC in his account. Every transaction is timestamped in Blockchain network. It ensures the uniqueness of each transaction and security with the use of cryptography.

Step 2 : Signing Steps (Digital Signature)

In this case, to ensure the security of transaction done so that data will not be altered, User A share his public key with User B. User A then creates digital signature. User A performs following steps to create digital signature.

1) Hash of transaction (message) is generated by using public hashing algorithm.
2) This generated hash then is encrypted by using his private key.

User A then appends this signature in message and sends it to User B. User A then use User B's public key to encrypt this digitally signed message or transaction. This message then sends to User B through Blockchain network. After receiving this message, User B use his private key to access the digitally signed message of User A. In this manner the integrity of data is ensured in Blockchain network.

Step 3 : Verification of Received Message or Transaction

Suppose User B want to verify the authenticity and integrity of the message received from User A. It means that User B want to confirm that the sender of the message is User A and message is not altered in transit. Following are the steps performed to verify the message.

1) User B use User A's public key to decrypt the digital signature to obtain hash. In this case, as hashed message was encrypted by User A's private key only his public key can decrypt it.

2) User B again generates hash by running the hash function on received message. If both hash matches then he confirms that the message is from User A.

In this way, digital signature offers immutability to data which is recorded in Blockchain and ensures security in Blockchain. Authorization is addressed by distributed consensus mechanism in on peer-to-peer network of Blockchain where each peer has the copy of database.

1.4 Foundation of Block chain : Merkle Trees

- When we use hash function, it maps input to fixed output. For every input, a unique hash is generated as output. Hence, it is possible to validate huge amounts of data through their hash.
- The concept of Merkel tree was introduced by Ralph Merkle. Merkel tree is a data structure which is also known as binary hash tree. This data structure summarizes and verifies the integrity of large sets of data. In this data structure, each non-leaf node is a hash of its child nodes and all the leaf nodes are at the same depth.
- In Merkel tree, inputs are first placed at the leaves. Both the child nodes are hashed together to generate a value for parent node. This procedure is repeated till the single hash value which is Merkle root is generated. Following Fig. 1.4.1 shows a Merkel tree.

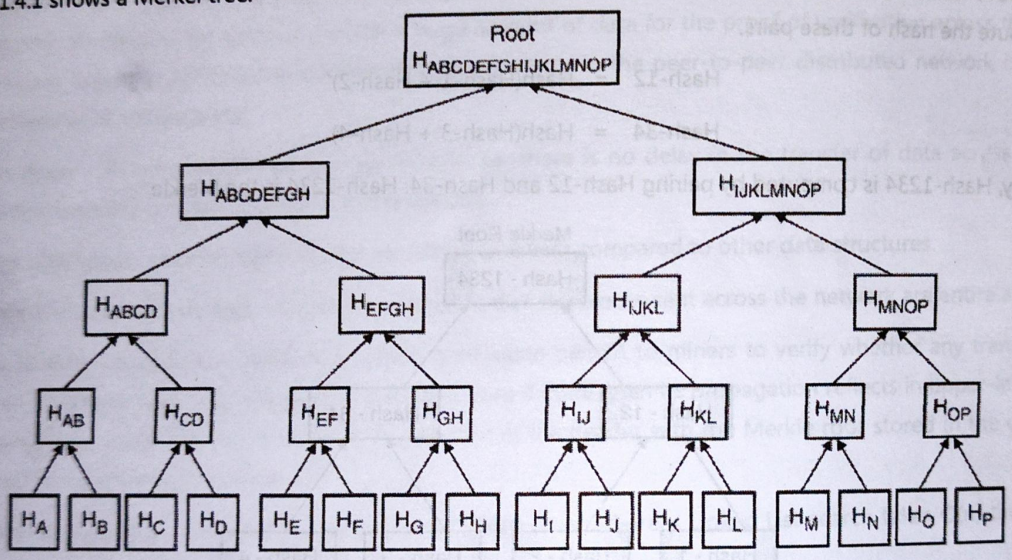


Fig. 1.4.1 : Merkel tree

- The data is always stored in memory at particular address and pointer stores this memory address of data. In hash pointer, pointer to data and cryptographic hash of data is stored. Therefore, hash pointer points to data and also helpful in verification of this data. Hence, data structures such as Merkle Tree or Blockchain can be built with hash pointer.

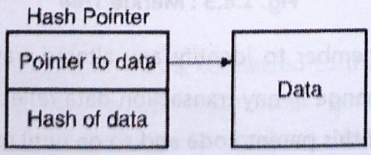


Fig. 1.4.2 : Hash Pointer

- Each block in Blockchain contains certain number of transactions. All these transactions can be stored as linked list of transactions in particular block. This approach of storing linked list of transactions in a block is inefficient or not practical. In order to search particular transaction in some particular block, first we will have to traverse block one by one and then we will have to traverse linked list of transactions in in searched block. This approach will reduce the efficiency Blockchain. Merkel tree solve this problem. Merkle tree of all the transactions for each block is created. Merkle tree permits to generates hash of all the transactions and offers proof of membership in a time efficient manner. A hash-based linked list of blocks is Blockchain. Each block includes header and transactions. These transactions are organized in a tree-like fashion, known as the Merkle tree.

Example :

- Consider block contains four transactions T1, T2, T3, T4. In this case hash of each transaction computed.

Hash-1 = Hash(T1)

Hash-2 = Hash(T2)

Hash-3 = Hash(T3)

Hash-4 = Hash(T4)

- All above hashes are stored in leaf node of Merkle tree. These leaf nodes are paired together from left to right to compute the hash of these pairs.

Hash-12 = Hash(Hash-1 + Hash-2)

Hash-34 = Hash(Hash-3 + Hash-4)

- Finally, Hash-1234 is computed by pairing Hash-12 and Hash-34. Hash-1234 is the Merkle

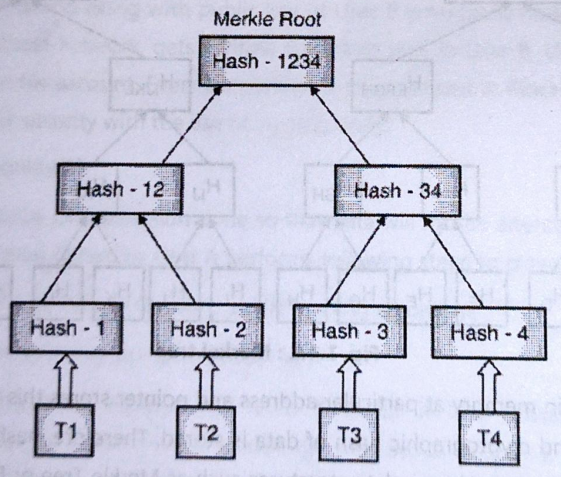


Fig. 1.4.3 : Merkle Tree

- Only root of the tree needs to remember to identify any altered transaction. Transaction can be accessed by traversing through hash pointers. Change in any transaction data reflects in hash stored in parent node. Which in turn reflects in hash of upper level of this parent node and so on until root node. In this way, Merkel root stored in the header of the block ensures transactions tamper-proof and validates the integrity of data.

- Due to Merkel root, Merkle tree can identify and remove duplicate or false transactions in a block. Merkle root in the header of the block is additionally protected by the hash of the block header stored in the next block. Because of Merkel tree, no need to send request for ledger to other nodes in peer-to-peer network for verification of any past transaction. Hence, huge amount of data transfer and processing at node is avoided. Merkle trees makes use of one-way hash function widely and this hashing separates the proof of data from data itself.
- If you have to verify whether particular transaction belong to Merkel tree or not **(Proof of membership)** then the only path from leaf node to root node in upward direction is required. Other nodes in tree are ignored. So, out n nodes in tree only log(n) nodes need to be verified.

Merkle Proofs :

- Merkle proofs are recognized by hashing a hash's corresponding hash together and going upward the tree until you obtain the root hash which is or can be publicly known. In figure 1.5, presence of transaction T1 is verified by hash only. We obtain Hash-12 and further Hash-1234 which is Merkle root.
- The obtained Merkle root and the Merkle root available within the block header are compared and presence of T1 can be verified. All the transactions T1 to T4 does not have to be revealed, only their hash is enough.

1.4.1 Advantages of Merkle Tree

- **Efficient verification** - Integrity and validity of data is verified in efficient manner. Verification requires less amount of memory. No need to transfer a huge amount of data for the proof of verification across the blockchain network. Supports for trustless transfer of cryptocurrency in the peer-to-peer distributed network by the speedy verification of transactions.
- **No delay** - As amount of data transfer is reduced, there is no delay in the transfer of data across the network. Merkle trees are extensively used in computations.
- **Less disk space** - Merkle trees require less disk space with compared to other data structures.
- **Unaltered transfer of data** - Merkle root ensures that the blocks sent across the network are entire and unaltered.
- **Tampering Detection** - Merkle tree offers remarkable benefit to miners to verify whether any transactions have been tampered with. Any change in transaction data if done, then its propagation reflects in upper-level hashes till Merkle root node. The comparison of Merkle root in the header with the Merkle root stored in the data part of a block can detect the tampering.
- **Time Complexity** - Merkle Tree search takes O(logn) time whereas Linked List search takes O(n) time where n is the number of transactions in a block.

1.5 Components of Blockchain

In order to briefly understand the block in Blockchain, we need to understand different terminologies, concepts and components that are used in Blockchain technology.

1.5.1 Node

Any device such as computer, mobile device that is connected to the Blockchain network is called as node. These interconnected nodes have copy of blockchain ledger. In Blockchain technology peer-to-peer architecture, every node maintains complete updated copy of blockchain ledger or database. Nodes maintain copy of Blockchain and preserve integrity of the Blockchain.

- **A Full Node :** This node keeps whole copy of transaction history of the Blockchain. These nodes processes and accepts the blocks or transaction, validates them and then broadcast to Blockchain network. This is known as forging or mining. Nodes run the full node to help synch the Blockchain.
- **Light or Partial Node :** These nodes in Blockchain network may have insufficient disk space for the full Blockchain or they may be early users. Such nodes maintain partial copy of ledger. Light nodes use Simplified Payment Verification scheme to validate the authenticity of transactions. For this purpose, they download only block headers. These nodes are dependent on full nodes for latest headers, account balance and any transactions affecting their wallet. Several full nodes are preferred in Blockchain network so that it can be truly trustless and decentralized distributed system.
- In public permissionless Blockchain such as Bitcoin or Ethereum, anyone is allowed to join Blockchain network in order to validate and create blocks. But, Blockchains that are created as per business requirements such as: Private Permissioned Blockchain or Consortium Blockchain, nodes should fulfil some prerequisite norms to join or participate. Ripple, Hyperledger Fabric, Monero are the some of the examples of this category of Blockchain.

1.5.2 Ledger or Database

Ledger is a digital database of transactions or information which can not be altered by anyone. It means they are immutable. Blockchain is public distributed decentralized ledger because of the following reasons.

- Ledger is public in Blockchain and anyone can have access to it for reading or verification of transactions.
- Blockchain ledger is replicated on all the nodes and hence ledger is distributed.
- Blockchain protocols ensures that, any node or group nodes cannot have extreme control over the ledger or database. All the nodes can validate the records. Blockchain operates with Append-only principle and not with create-read-update-delete (CURD) principle in traditional database.

1.5.3 Wallet

Wallet manages cryptocurrencies. Blockchain participants can send or receive cryptocurrency with blockchain wallet. Suppose User A sends cryptocurrency such as bitcoin, Litecoin, ether to User B's wallet. Here, User A signs off ownership of sent cryptocurrency and User B can spend it only if private key of his wallet matches with public address of cryptocurrency. After this match, User A's wallet amount gets deducted and User B's wallet amount gets increased.

- After creation of wallet, the public and private key associated with it is generated.
- In order to open the encrypted message and to send cryptocurrency. It ensures security of transactions.
- Wallet's usage is simple and easy. Funds can be transferred instantly across any geographical area without any requirement mediators such as banks.
- Transaction across various types of cryptocurrencies such as BTS, ETH, XMR, LTC etc is supported by wallet. No need to convert currency.

1.5.4 Nonce

Nonce is related to cryptographic communication. It is unique random number generated and used only one time in communication. It is added in identifier of transaction to make it unique so that duplicate transactions can be avoided. If miner successfully find a nonce able to a valid block hash receive the right for addition of the next block in the blockchain. At the same time, the miners get rewards for finding the nonce. Nonce is key to create block in blockchain database.

1.5.5 Hash

Hash is fixed size data returned by hash function after processing data of any arbitrary size. Hash function can take data of any size as a input, process it, and outputs fixed size data called as hash. Hash is much smaller than input data. Following are the important features of hashing.

- Hashing creates identifier (hash) which is unique. In blockchain, these identifiers are used for block, transactions and addresses. In bitcoin, hashes are of 256 bits or 64 characters and Blockchain uses Secured Hash Algorithm that generates 256-bit hash (SHA-256).
- It is one directional. It is not possible to generate original input data from generated hash.
- Sligh change in input data generates different hash. Hence, hashing function supports better security.
- As hashes are of small fixed sizes, its database is also of small size with compare to database storing the original data.

1.5.6 Mining

In mining, validation of new transactions is carried out. Miners or Forgers are nodes which carries out mining to validates new transactions and add them to the Blockchain ledger. These nodes are called as miners in bitcoin Blockchain and forgers in Ethereum Blockchain. On average, a block is mined in every 10 minutes. Each Blockchain has its own block size. Bitcoin blockchain has block size of 1 M. Minors and Forgers use cryptographic algorithm. In mining, block is hashed and then nonce is introduced to hashing function. Following steps are involved in mining process.

1. **Broadcasting of transactions :** Once transaction is done, it is advertised by wallet application and minor confirm it on the Blockchain. Initially transaction is the part of pool of unconfirmed transactions where it waits for its turn to pick up by miner.
2. **Selection of transaction to add in block :** A miner creates his own block of transactions. The same transaction can be selected by other miners by verifying the eligibility of transaction. As per history, if wallet balance of sender has sufficient funds, then transaction is eligible and added to the blockchain.
3. **Signature mining and Creating Block :** Miners/Forgers choses the transactions and form a new block to add them. Every new block needs a signature to be added to a blockchain. This signature is called as Proof-of-Work. This signature is the indication to denote that the miner has used many resources and time to solve complex mathematical problem. Every mathematical problem to be solved by different miners is quite complex and needs a large amount of computational work. This mechanism of solving a complex mathematical problem that involves huge computational resources is known as mining. Mining is the efforts carried out to obtain the signature of a block. The resulting hash of new block should be less than target value. Target value is the number that a hashed block header must be less than, for new block to be awarded. If hash is not less than equal to target value, miner increment the nonce and returns hash. This is repeated until needed hash is reached.
4. **Addition of block in to the blockchain :** First miner who obtains the qualified signature for the next block to be added in the blockchain, introduces this block and its signature to other miners.
5. **Verification of signatures legality :** Other miners check the validity of the signature advertised by the miner by hashing the string of data of the broadcasted block and matching the output hash with the signature. If match is found then the miners confirm on its validity. The higher number of confirmations restrict the hackers to work on it.

In this way, the definition 'proof of work' can be clarified. The signature is the 'proof' of the work carried out in the form of the computational calculations. The block is now ready to include in to the blockchain, and is broadcasted to nodes on the Blockchain network. The other nodes will receive the data into their transaction data, depending on the condition that its coherence correctly with the current wallet balances (transaction history) at that point of time.

6. **Confirmation count :** After adding the new block in the blockchain, every succeeding block which is added after that block is a 'confirmation' for that particular block. After addition of new block to Blockchain, the miners are needed to perform step 2 by preparing a fresh block of transaction.

1.5.7 Consensus Protocol

- Nodes in the Blockchain network reaches on agreement about data value or network state with the help of consensus protocols. In peer-to-peer decentralized Blockchain network, it is necessary to have agreement on single source of truth by all participants. Hence, these protocols ensure validation of all transactions before adding them in Blockchain.
- Public (decentralised) blockchains are created as distributed systems, and because they don't rely on a single entity, the dispersed nodes must concur that a transaction is genuine.

1.6 What is Block in Blockchain?

1.6.1 Block in Blockchain

Block in Blockchain contains many transactions and following are the details that block comprises :

1. Hash of block
2. Hash of previous block
3. Timestamp
4. Nonce
5. Merkle root
6. Transaction data, containing details of transactions in block.

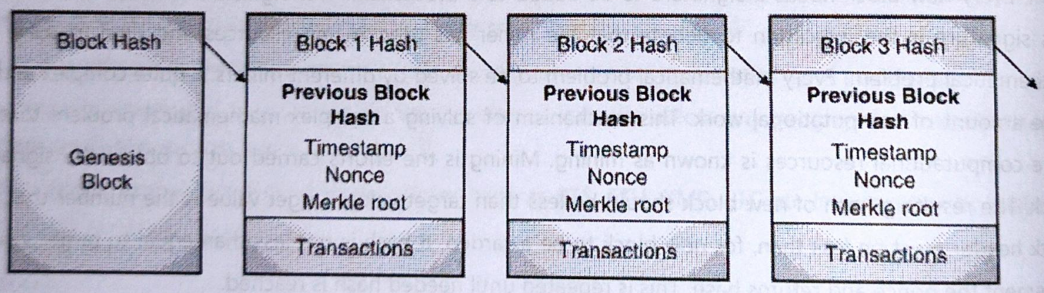


Fig. 1.6.1 : Block representation

- Genesis block is very first block which does not contain data hash of previous block. Hence, previous hash value Genesis block is set to zero. This block includes transactions that are combined and validated to generate unique hash. The hash of genesis block is added to all new transactions in a new block. This combination is used to create its unique hash. This process is repeated until all the new blocks are added to a blockchain.

- For example, In general Block n hash is created with the hash of block n-1 and another set of new transactions. Hence, each block is linked with previous block via its hash. From any block, there is continuous linkage in reverse direction till genesis block. Hence, it is not possible for attacker to alter the data or to insert block between two existing blocks. If many number of blocks are present in blockchain then more blocks needs to be altered for tampering the data.

1.6.2 Steps in Blockchain Transaction

Consider the same example where User A sends 1.0 BTC bitcoins to User B. Following are the steps involved.

1. Proposed transaction is requested by User A and he sends 1.0 BTC from his wallet.
2. This transaction is then broadcasted to Blockchain network.
3. Miner validate this transaction. Its authenticity, balance in wallet of User A. In this way many transactions are validated by miners to include them in Block they plan to mine.
4. Minor solves complex mathematical problem as explained in section 1.5.6.
5. If miner finds correct hash then this block is broadcasted to network anf majority of miners/nodes verify it. In this way consensus is obtained. Wining miner then receives award.
6. Winning minor adds his block in Blockchain after verifying it. User A's transaction added in now in Blockchain along with other transactions.
7. This updated copy then is circulated in the Blockchain network.
8. User B recieves 1.0 BTC in his wallet.

1.6.3 Double Spending

When we deal with transactions with digital currency, the double-spending problem is more common to arise. Double-spending means just spending the money more than once. It happens just by duplicating the token or crypto-coin and reusing it. This double spending leads to increase of duplicate, fraudulent currency in network. Every transaction has a timestamp. The first transaction is time-stamped and linked to previous block and then broadcasted to all the nodes. When the second duplicate (fraudulent) transaction is proposed, it will fail in verification and will not be accepted. It is nearly impossible to double spend the transaction once it confirms.

1.7 Blockchain Layers

Blockchain ecosystem is shown in Fig. 1.7.1

1. Presentation Layer

Throgh presentation layer, user accesses the network and blockchain applications. For any user, mobility and ease of use are the most important factors. Like other system, the success of Blockchain is also depends on user experience and user interface. Customor satisfaction increases with good user experience design which also improves product usability and interaction of user and rest of the Blockchain ecosystem. In the same way user interface also plays important role. Good user experience and user interface design supported by blockchain application supports user for better interaction with Blockchain.

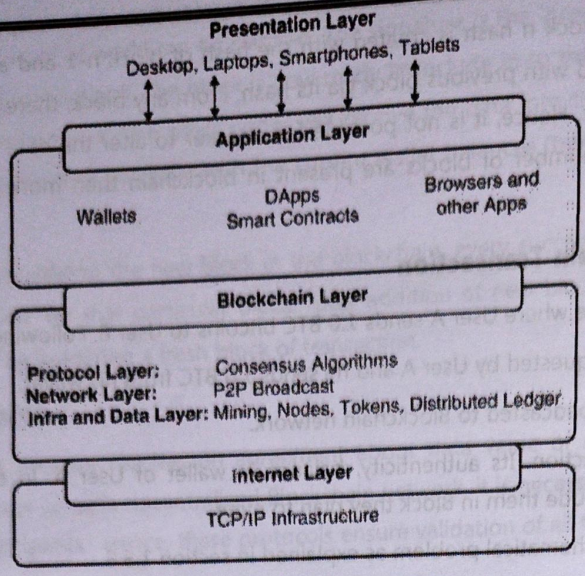


Fig. 1.7.1 : Blockchain Ecosystem

2. Application Layer

This layer combines business logic with interaction of users. DApps running on P2P network are the part of this layer. These Dapps are responsible for communication between presentation and bockchain layers. Websites connect to database through APIs. Distributed applications (DApps) connect to Blockchain throgh Smart Contracts. Smart contract is the program which implements set of rules and criteria. These set of rules/criteria automatically executes provided predefined conditions are fulfilled.

3. Blockchain Layer

Many consensus algorithms are the part of this layer. This layer also contains medium and interface to access Peer-to-Peer network. This involves in packetization of data and its transmission. Mining, different protocols involved to take decision on consensus methods and participation, nodes involved in this activity and distributed database are controlled by this layer.

4. Internet Layer

This layer is responsible for interoperability between different networks that involves computers, mobile devices, IoT devices etc.

1.8 Types of Blockchain

Blockchain network contain many numbers of nodes which are computers or any other hardware devices which are used by people. Hence, node is the user, member or participant in the blockchain network. Blockchains are categorized on the basis of user authentication and authorization.

1. Public and Private Blockchain

- In public Blockchain, anyone is allowed to join the Blockchain network. They can access the copy of ledger and can read, write, participate within the Blockchain. They can initiate, broadcast and mine the blocks. In this blockchain, everyone is anonymous. It is not necessary to use your real name, or real identity. A public blockchain is decentralized without having a single entity for controlling the network.

- A private blockchain is managed by a network administrator. In this Blockchain, membership is controlled and participants should obtain consent to join the network. Users are not anonymous and must fulfil certain criteria to be a part of blockchain network.

2. Permissionless and Permissioned Blockchain

- This classification of Blockchain is based on rights that node has in blockchain network. Rights are defined by central entity or group of entities. In permissionless Blockchain, there is no control entity and all the nodes have equal rights. These are the public blockchains where nodes can read, receive and send transactions and are the part of consensus process.
- Permissioned blockchains are closed networks where earlier approved parties, sometimes members of a group, interact and contribute in consensus process and data validation. In this Blockchain there is central entity or group of entities controls the role which nodes has to play. Some selected members are the part of consensus mechanism. It is private blockchain.
- Following is the classification of Blockchains based on the authentication and authorization rights.
  - Public Blockchain (or Public Permissionless Blockchain)
  - Private Blockchain (or Private Permissioned Blockchain)
  - Consortium Blockchain (or Public /Private Permissioned Blockchain)
  - Hybrid Blockchain (Interconnected Public-Private Blockchain)

1.8.1 Public Blockchain

This is also known as public permissionless Blockchain. Bitcoin is public permissionless Blockchain. This Blockchain is fully decentralized distributed network. In this blockchain, everyone is anonymous. It is not necessary to use your real name, or real identity. Anyone can access the copy of ledger and can read, write, participate within the Blockchain. There is no need to obtain permission for accessing or initiating the transaction. Also, to create a block, permission to participate in consensus process is not necessary. Following are the key features of public Blockchain.

- The public blockchain is open for all and anyone can be the participant in the network.
- It is immutable.
- Anyone can read or send transaction without any permission.
- In Proof-of-Work, standard consensus algorithm is used. Miner obtains hash and sends the block to other members for getting the consensus.
- All the nodes in the blockchain participate in the transaction validation. Hence, no single point of failure in the network.
- Highly secured with the use of cryptographic methods.
- Truly decentralized as no centralized entity present.
- Highly secured due to mining (51% rule)
- It offers transparency as it permits you to see the ledger anytime you want.

Limitations :

- It is having poor scalability.
- Speed of transaction processing is slow. It takes 10 minutes to create a new block.
- Huge amount of energy and computation power is involved in consensus process.
- Examples of public Blockchain: Bitcoin, Litecoin, Ethereum, Monero, Zcash, Dash etc.

1.8.2 Private Blockchain

This is also known as private permissioned Blockchain. Blockchain network is only open to the participants having access permissions. In this Blockchain, permissions are controlled by few nodes in the organization. Hence, private Blockchain is not fully decentralized, but it is more centralized. In this blockchain, members participating in the transaction only know about the transaction carried out whereas others will not able to access it i.e., transactions are private. Following are the key features of private Blockchain.

- It gives assurance about the trust as participants are known to each other.
- All the participants in private blockchain are first approved by organization. Ledger and access are distributed within this network of pre-approved participants.
- Provides better Scalability as supports to add nodes and services on demand can offer a great advantage to the enterprise.
- There is central authority or owner who authorizes the members to read, write or audit the ledger.
- In private Blockchain, consensus process is controlled by central authority.
- Highly secured with the use of cryptographic methods.
- High transaction processing speed. It requires only seconds to create block.
- Low energy consumption and highly efficient.

Limitations :

- Private Blockchain are more centralized.
- Due to presence of central authority, chances of single point of failure.
- Controlled by central authority, hence inherent doubt of immutability and trust of transaction. Central authority is decided by organization
- Examples of private Blockchain: R3 (Banks), EWF (Energy), B3i (Insurance), Multichain and Monax etc.

1.8.3 Consortium Blockchain

This is also known as federated Blockchain which is hybrid between the public and private Blockchain. It is also permissioned Blockchain having security features just like the features in public Blockchain but it maintains fair amount of control over the network. Anyone can access and down load this distributed ledger. Here, consensus process is carried out by previously decided group of organizations or representative individuals. It is not controlled by single organization just like in private Blockchain. Anyone cannot take part in validation process to create the block. This right is only assigned to predetermined group. In case of supply chain, this group can be importer, exporter, bank, customs, shipping company or inspectors.

Depending on business requirements, smart contracts and other protocols together limit the user access. Following are the key features of consortium Blockchain.

- Consortium Blockchain has few numbers of nodes as compared to public blockchain, but it is more secure and scalable in nature.
- It is like semi-decentralized network where any node can initiate and receive the transactions.
- In this Blockchain members are granted to pre-determined group of individuals.
- Permission to audit or write the ledger is determined by group of pre-approved consortium.
- Highly secured with the use of cryptographic methods.

Limitations :

- These Blockchains are not fully decentralized.
- Consensus on standard set of rules is challenging due to varying requirements of different organizations.
- Examples of consortium Blockchain are Hyperledger Fabric, Corda, Ripple etc.

1.8.4 Hybrid Blockchain

This type of Blockchain is mostly used in government organizations and regulated enterprises. It includes best practices and benefits of both private and public Blockchains. The hashed data block created in private network is then shared in public network without compromising the privacy od data. In this Blockchain, public network members are anonymous whereas, private network members are known within private network. Here access is offered through centralized control which ensures privacy and confidentiality.

There is no single point of failure and operations are customizable. It is the role of central authority to make transactions public or private. The verification of block is carried out by public network. Security is ensured with hashing in private network and with consensus process in public network. Transaction speed is high with low energy consumption. It is highly scalable.

1.8.5 Comparison between Public, Private and Consortium Blockchain

Common Features	Public Blockchain	Private Blockchain	Consortium Blockchain
Network	Fully decentralized	Centralized	Partially decentralized
Accessibility	Open to anyone	Central Incharge or Single individual	More than one central Incharge.
Failure	No single point of failure	Single point of failure	Multiple point of failure
Transaction speed	Slower	High	Very high
Participants	Everyone is anonymous	Known and trusted participants	Known and trusted participants
Energy Consumption	Very High	Low	Low

Common Features	Public Blockchain	Private Blockchain	Consortium Blockchain
Scalability	Limited	Better	Better
Computation power required	High	Low	Low
Trust	Trust-free	Trusted	Trusted
Efficiency	Low	High	High
Consensus determination	All miners	Single organization	Selected set of members
Immutability	Tampering is nearly impossible	Tampering can be possible	Tampering can be possible
Consensus Process	Permissionless	Permissioned	Permissioned
Time for Block creation	More than 10 minutes	In seconds	In seconds
Read Permission	Public	Public or restricted	Public or restricted
Security	Based on consensus protocols and hash functions.	Depends on blockchain architecture	Depends on blockchain architecture
Consensus Mechanism	PoW, PoS, etc.	Voting or different PoW/PoS consensus Algorithms	Voting or different PoW/PoS consensus Algorithms

1.9 Consensus Protocol

- In many situations, we need agreement among all the members in a people group. Sometimes, one or more members in a group cannot be trusted to give correct answers. Consensus is a general agreement shared by all the members in a group. Protocols are set of rules defined to exchange the information in the network.
- In Blockchain, consensus protocol is the set of rules and procedures to reach on collective agreement between various participants about the status of the network. In peer-to-peer network all the nodes are equal. In the network, multiple copies of same data are stored on different nodes in the network. As data is available on multiple nodes, there is no chances of single point of failure.
- In Blockchain technology peer-to-peer architecture, every node maintains complete updated copy of blockchain ledger or database. It helps to reach on consensus about the actual state of distributed ledger.
- The rules defined in consensus protocols decides how the different participants in distributed ledger authenticate and validate the transactions added to it. This is done to prevent the creation of different version of ledger or to edit the previously added transactions.

1.9.1 Byzantine Generals Problem

The Byzantine Generals Problem describes the trouble decentralized systems faces in agreeing on a single truth. In this problem, two byzantine armies led by different generals plans to attack a city. Armies are based on either side of city and only can win the battle if both armies launch a coordinated attack simultaneously. Messages between two generals can be exchanged by sending messengers through the city. One general (General-1) sends message through messenger with planned day and time. Other general (General-2, recipient of message) acknowledges this message and sends it sender general. Following can be the reasons for defeat.

- General-1 does not receive acknowledgement and hesitate to attack on given planed time.
- Messenger sent by General-1 is captured by enemy and message is intercepted.
- General-1's message is intercepted and replaced with false information. General-2 hesitate to attack as he is unable to check authenticity of the message.
- Messenger from General-2 can be captured by enemy and message can be intercepted and replaced with false/fake message.
- One of the generals could be traitor.
- Any doubt or lack of confidence between two generals may affect the soldier's confidence. This may lead to defeat even if there is coordinated attack.

Above is the example of dilemma between two participants. There are many participants in Blockchain network. It is necessary that, all the participants should verify and reach agreement (consensus) to nullify the fake parties and spread of false information. The consensus mechanism in Blockchain should verify authenticity of message and consider collective agreement by all nodes about any action to be taken. Few hackers should not be able to compromise the process.

1.9.2 Objectives of Consensus Protocol

A consensus mechanism is used in blockchain for the agreement between nodes in the network about the valid transactions so that they can be updated on to the ledger. This mechanism is fault-tolerant and ensures security in blockchain. Following are the objectives of consensus mechanism.

1. **Unified Agreement :** Protocols in the network should ensure that, there should be a unified agreement amongst all nodes about data is valid and correct.
2. **Fault-tolerant :** In Blockchain network, some of the nodes may become unresponsive. In this case, many other nodes participate in the process of consensus to keep system functional.
3. **Collaborative and Participative :** Consensus mechanism should guarantee that, all the members contribute in the consensus process in the favour of the group rather than in the favour of single member or few members. This mechanism also permits for any member or node to check and verify that the consensus process is fair to all the participants in the network.
4. **Egalitarian :** Every node involved in consensus process should be considered at equal level.
5. **Incentivisation :** In mining, validation of new transactions is carried out. Miners carries out mining to validates new transactions and add them to the Blockchain ledger. Every new block needs a signature to be added to a blockchain. This signature is called as Proof-of-Work. This signature is the indication to denote that the miner has used many resources and time to solve complex mathematical problem. Consensus mechanism contain rules to incentivize the miners to work for system ensuring more security.

6. **Prevent Double-spend :** Double-spending means just spending the money more than once. It happens just by duplicating the token or crypto-coin and reusing it. Consensus mechanism implement rules that prevents double spending.

1.9.3 Consensus Algorithms

Many consensus algorithms exist those functions based on different principles. For public distributed ledger, most preferred mechanisms are Proof-of-Work (PoW) and Proof-of Stake (PoS). In Private Blockchain, Proof-of-Authority and Proof-of-Elapsed-Time (PoET) are more common.

We will discuss consensus algorithms in detail in coming modules. Here just giving overview of algorithms.

1. Proof-of-Work (PoW)

- The distributed consensus protocols are more helpful to prevent cyber-attackers. Hashcash which has used secure hash algorithm 1 (SHA-1) served as the motivation in Proof-of-Work consensus mechanism within Bitcoin distributed ledger. Consensus algorithms in distributed system offers the basis of the practical implementation of Proof-of-Work algorithms in bitcoin.
- This algorithm is used by first Blockchain bitcoin in 2009. In this mechanism, several miners compete to solve complex mathematical problem by using cryptographic hash algorithm. Mining consists of hashing the block and introducing nonce to hash function and running the hash all over again. Miner will win if generated hash is less than or equal to the target hash. The solution then obtained is known as PoW. This restricts the attacker to carry out any malicious activity.
- Mining node broadcast this PoW to other participants for verification in order to get consensus on the same. Network can verify it with low cost and very first miner gets rewarded with digital currency. This reward is given as mining process elapses more computational resources as well as time. Bitcoin, Litecoin, Monero, Ethereum uses PoW mechanism.

2. Proof of Elapsed Time (PoET)

- Proof of elapsed time (PoET) is a consensus algorithm is developed by Intel Corporation in 2016. The algorithm mainly used in permissioned blockchain networks to determine who creates the next block. The algorithm follows a fair lottery system which ensures equal chance for each participant to be a winner. PoET algorithm also improves transparency by confirming lottery results are verifiable by external participants. This is done by running a trusted code within a secure environment.

3. Proof of Stake (PoS)

- Proof-of-stake is a cryptocurrency consensus mechanism developed for Peercoin in 2012. In this algorithm owners of the cryptocurrency can stake their coins, which assigns them the right to validate new blocks transactions and add them to the blockchain. If more stake one has in validating node, then chances corrupting the node reduces. Users with highest stake in cryptocurrency will always pay more attention towards maintaining and securing the network. The reason is, any security attack would be a threat reputation and value of their cryptocurrency. The algorithm randomly selects validator depending on stake collateral he or she has put forward.

4. Delegated Proof of Stake (DPoS)

- DPoS consensus algorithm is an advancement of the basic concepts of the PoS algorithm. It was developed by Daniel Larimer in 2014. DPoS system is maintained by an election system for selecting nodes which validate blocks. These nodes are known as "witnesses" or "block producers". In DPoS consensus mechanism, participants in network use their cryptocurrency or tokens to vote for the delegates (witness). The voting power is directly proportional to cryptocurrency or tokens. More the tokens then more the voting power.
- In DPoS consensus, participants can either give their vote directly or handover their voting power to another participant to vote on their behalf. Chosen delegates or witnesses creates blocks by verifying transactions. After verifying and signing of all transactions in a block, delegates receive a reward, which is generally shared with voters of witness. If a witness fails to verify all transactions in allotted time, block is missed, all transactions in unverified state and no reward is given to that witness. The reward is added up to reward of the next witness responsible to verify that block. All these unverified transactions are collected by the next witness, and such a block is called stolen.
- In DPoS there is better distribution of cryptocurrency and it is more secured. If witness performs fraudulent activity, then voters can detect it and penalty is posed on witness. All users can elect as witness. Hence, it is more decentralized with compared to PoW and PoS.

Limitations of DPoS Consensus Mechanism :

- Few people maintain network. Hence, chances of 51% attack.
- If more validators are there then there is risk of slowing down the network which leads to centralized power in the hands of limited members.

5. Proof of Authority (PoA)

- PoA consensus mechanism was proposed in 2017 and used in permissioned Blockchain. In permissioned blockchain, all the participants are pre-selected which permits to use consensus mechanism with high scalability and bandwidth. Just like in PoS and DPoS, in PoA also pre-selected validators secure the Blockchain and can create new block. Instead of staking coins or tokens like in PoS and DPoS, in PoA validators stake their identity. These identities are public and verifiable by reliable third party like public notary database. The reputation of validators can ruin in case they do not work in the interest of the network. To circumvent spoiling their reputation, validators are inspired to maintain a normal transaction process. Thus, most members value their hard-earned role of a validator. Validators are incentivized if they work in the interest of network. Maximum number of validators are 25 or less to ensure the efficiency and security of the network. Once verified and approved, reverification of the validator is not done unless needed. PoA requires less computational power and consume less time and energy. Speed of validating transactions is very high leading to a higher transaction rate. Due to limited number of validators, it is highly scalable. It also ensures guaranteed protection against 51% attacks on the network. VChainThor blockchain, Ethereums Kovan, Rinkeby testnets uses PoA consensus mechanism. Hyperledger and Ripple uses optimized vertion of PoA.
- In order to identify as a validator, following three basic conditions must be met :
  - Valid identity of the validator should be available in public domain. This identity should be same as per record present in public notary database.
  - The authority needs to be uniform and impartial for all validators.
  - Eligibility criteria for staking identity must be strict to assure the trustworthiness of the validator.

6. Practical Byzantine Fault Tolerance (pBFT)

- Practical Byzantine Fault Tolerance is a consensus algorithm introduced in 1999 by Barbara Liskov and Miguel Castro. It is a blockchain consensus mechanism which is widely used presently, but the confidence of blockchain node in pBFT cannot be guaranteed. It requires large amount of communication resources in the process of reaching consensus.
- Byzantine general problem. In this case, all general nodes are at equal level and gets their work direction from the leader node. The leader node is primary node and all other nodes are considered as secondary or backup nodes. Selection of leader is randomly in round-robin manner. Client node sends transaction request to leader node. This received request is then broadcasted by leader node to all the secondary/backup nodes. Both leader and secondary nodes use the request message with their internal state to run computation and broadcast the decision result to all client nodes. Majority of nodes agreed upon decision is now considered as final decision.
- Suppose f are maximum number of faulty nodes. Practical Byzantine Fault Tolerance consensus algorithm relies on minimum number of back up nodes which are (f+1). Therefore, high hash rate is not required. This ensures less computation leading to less energy consumption.

1.9.4 Other Consensus Mechanisms

Following are some other variations of consensus mechanisms.

1. Proof of Stake Anonymous (PoSA)

It was first introduced in Clockcoin in 2014. It is variation of PoS and truly decentralized. there is no master node present. Hence, it is truly secured network. Nodes are incentivized for cloaking the transactions.

2. Leased Proof of Stake (LPoS)

Leased Proof of Stake (LPoS) is a consensus mechanism which is a variation of PoS. It is used in Waves blockchain in which users lease crypto tokens to a node that wants to act as a network block creator. In PoS, anyone staking cryptocurrency could be selected as a validator, the chances are very low if you stake relatively small amount. Hence in LPoS, smaller stake holders can pool their stakes and increase their probabilities of receiving a share of the network's transaction fees. Hence, small cryptocurrency amount holders can become block creator.

3. Proof of Importance (PoI)

The PoI consensus mechanism was introduced by the NEM (New Economy Movement) blockchain in 2015 to overcome the disadvantages of PoS. Their cryptocurrency coin is called XEM. PoI assigns consensus addresses a importance scores. A higher trust score indicates more network trusts to verify a new block of transactions. Hence higher chances of getting selected as block creator. Therefore, instead of stakes, number of transactions and the quality of the transactions processed in the past is important factor.

4. Proof of Storage

Proof-of-storage consensus mechanism implemented in Storj system. Network uses a block tree and no need to go through all the transactions. The needed transactions only user can see in this system.

5. Proof of Burn (PoB)

This consensus mechanism first proposed by Iain Stewart in 2012. Validators burn coins by sending them to eater address from where they are irretrievable. These coins are now destroyed coins and called as burned coins. Those who burns more coins, their chances to mine the blocks are more.

6. Proof of Activity

The PoA consensus mechanism combines the best features of both the PoW and the PoS consensus mechanism. In PoA, the mining process initiates in similar manner as in a PoW process. Different miners attempt to beat each other with higher computing power to find a new block. Once, a new block is located or mined, the mechanism switches to PoS. Now, this newly found block contains only a header and the miner's reward address. With this header details, a new, random group of validators from the blockchain network is selected.

These validators are needed to validate or sign the new block. The more currency a validator owns, the more probabilities they have for being selected as a signer. After signing of newly-found block by all the validators, block becomes complete block and it gets identified and added to the blockchain network. Now the recording of transactions on it begins. PoA also prevents the chance of a 51% attack. Energy consumption is still the issue in this mechanism.

7. Proof of Capacity (PoC)

- Proof of capacity consensus mechanism is used by Burstcoin cryptocurrency. Storj, Chia, and SpaceMintare the other coins that used this mechanism. Proof of capacity mechanism is developed to prevent the problem of high energy consumption in proof of work (PoW) mechanism and cryptocurrency hoarding in proof of stake (PoS) consensus mechanism. PoS permits the mining nodes on the blockchain network to use empty storage on their hard drive to mine the available cryptocurrencies.
- The proof-of-capacity protocol comprises a two-step process: plotting and mining. Initially, miners plot their hard drive. They create all possible nonce values with repeated hashing of data, together with a miner's account. In this case, miners find solutions and store them ahead of time. When actual mining gets started, miner with fastest solution wins the block. Mining data can be easily wiped-off and the drive can be reused for any other data storage purpose.

8. Directed Acyclic Graph (DAG)

In PoW consensus mechanism, it takes 10 minutes or more to create a block. It is also not allowed to create blocks at the same time. DAG allows multiple transactions to run on different chains simultaneously. IoT chain built on DAG processes 10,000 transactions per second.

1.10 Limitations and Challenges of Blockchain

Following are the **limitations and Challenges** of blockchain.

1. Complexity of Technology and Lack of Awareness

- Blockchain technology is increasing its popularity day by day but there is a lack of awareness and knowledge about it. There is lack of specialized expertise in blockchain technology. Hence, the lack of developers is a difficulty for developing anything on the blockchain in different situations. Overall, this is a very complicated industry that will not be soaked in and applied overnight.

- Blockchain technology understanding involves learning of an entirely new vocabulary or terminologies. Singl blockchain application cannot be easily duplicated across operations and use-cases. Each application needs deeper knowledge of the business needs and blockchain can be drastically different for applications such insurance contracts and for land records. The traditional IT systems are benefited from decades of researc The blockchain is still a new technology and needs a lot of research to attain maturity.

2. Limited Privacy

Privacy of transactions is a very important feature of blockchains. However, especially in public blockchain everything is transparent, thus constraining its usage in various industries where privacy is of main importanc such as finance, health, and many others. Several protocols and techniques address privacy issue buy they hav their advantages and disadvantages. There are more chances of 51% attack. Anyone can create the new walle anonymously and transact through that wallet.

3. Limited Scalability

Consensus mechanisms in bitcoin requires every participant to verify the transaction. It bounds on the number transactions a blockchain network can process. The financial networks are currently more scalable than blockcha network. At present, if we compare the largest centralised payments system (for example, Visa) and largest cryp payments system (for example, Bitcoin) then there is a huge difference. Visa processes 65,000 transactions second with compare to 7 transactions per second by bitcoin. This is because the of the control of centraliz authority in Visa. In block chain process, the validation involves several minutes as majority of nodes has authorise the transaction.

4. Security Concerns and Flaws

If more than 50 percent people in the network are corrupt and they agree on manipulated information then tr will be considered as lie. This will be leading to failure of whole network. There are more chances of 51% atta when network size increases. Following are the security challenges of blockchain.

- **Spoofing of payment information :** Malware can replace crypto wallet address with another one at time money transfer.
- **Phishing :** Exchange users can be falsely directed to phishing sites through which their crypto wallet cred can be stolen.
- **Crypto Wallet Theft :** As users store their wallet on their computer, which can be vulnerable to malwa attack.

5. High Energy Consumption

Blockchain requires constant computational power in many locations leading to the on-going extensive electric consumption worldwide. In 2009, Blockchain technology got introduced with bitcoin. Bitcoin blockchain uses th Proof-of-Work (PoS) consensus mechanism that relied on the miners to solve complex mathematical problem f which they get incentivized. Solving complex mathematical problem consumes lot of energy. Every time the ledg is updated with a new transaction, the miners need to solve the problems which means spending a lot of energ Some other consensus mechanisms address this problem and attempts for the low energy consumption.

6. Increased transaction cost

In the beginning, blockchain transaction cost was almost free. With growing network, this cost is now changing. There is no any fixed regulation to decide transaction fee and this is up to the transacting parties to decide. Hence, miners may choose to process the transactions based on who are highest paying members. This will either mean higher transaction cost, or slowly processing of your transaction.

7. Consensus Mechanism

In the blockchain, a block can be created in either 10 minutes or in seconds as per type of blockchain network. The transaction performed must ensure that, every block in the blockchain network must reach a common consensus. Depending on the network size and the number of blocks or nodes involved in a blockchain, the huge communication involved to reach a consensus can consume a significant amount of time and resources.

8. Immutability

With this feature in blockchain, we can not modify or change any record. It helps to keep integrity of the record. It is not possible to tamper the records. The drawback of the immutability is that, we cannot make revisions, go back and make any reversals. If payment is already processed and need to go back and make an alteration to change that payment then it is not possible.

9. Adoption

The blockchain is seen as an emerging technology. Although this viewpoint is rapidly changing, there is still a long way to go before the mass adoption of this technology. The challenge here is to permit blockchain networks easy to use so that adoption can increase. Furthermore, several other challenges such as scalability exist, which must be solved in order to increase adoption.

10. Regulation

- The regulation is almost impossible on the blockchain. This is due to its decentralized nature. This can be the hurdle in its adoption. When regulatory authorities exist, customers have a certain level of confidence that, they can hold someone accountable in case of any problem.
- However, in blockchain networks no such regulatory authority and control exists, which is an constraining factor for many customers.

1.11 The Benefits of Blockchain Technology

1. Decentralization

Decentralization is important and key concept in Blockchain technology. There are many consensus mechanisms are used to agree on the validity of transactions. The agreement among majority is considered to take decision. There is no need for a trusted third party or intermediary to validate transactions.

2. Transparency and trust

As blockchain system is transparent and different blockchains are shared. All the members or participants can see what is on the blockchain. Thus, trust is established. It is more relevant in situations such as the payment of funds or benefits where personal discretion in relation to choosing beneficiaries needs to be restricted.

3. Immutability

With this feature in blockchain, it is not possible to alter any record. Altering the data is so challenging and near impossible, this is a benefit to maintaining an immutable ledger of transactions.

4. High availability

There are many numbers of nodes in a peer-to-peer Blockchain network. The data is replicated and updated on every node. Hence, failure of some nodes does not affect the system and it becomes highly available. Although few nodes leave the network or become inaccessible, the network as a whole continues the functioning. This redundancy leads to the high availability.

5. Highly secure

The transactions in a blockchain are secured with cryptographic algorithms and thus offer network integrity. The transactions initiated from the nodes on the blockchain are verified using protocols. Hence, valid transactions are only included in a block. The privacy and confidentiality of transactions on the blockchain are being researched and studied to offer guaranteed security.

6. Simplification of current paradigms

In current model of many industries, multiple entities maintain their own databases and data sharing becomes very difficult due to the dissimilar nature of the systems. Single shared ledger between all the participants in blockchain simplifies this model by minimizing the complexity of managing the separate systems maintained by each entity.

7. Cost-saving

Absence of intermediatory or third party in the blockchain model hugely remove overhead costs in the form of fees, which are paid to such parties.

8. Faster dealings

In the financial business, particularly in post-trade settlement functions, blockchain can play a key role in permitting the faster settlement of trades as it does not need a long process of verification, reconciliation, and clearance because a single version of agreed upon data is already available on a shared ledger between financial organizations.

9. Platform for smart contracts

Blockchain platform runs business logic on behalf of the users. Smart contracts is a very beneficial and desirable feature supported on newer blockchain platforms such as Ethereum and MultiChain, but not on Bitcoin.

10. Smart property

Linking of digital or physical asset to the blockchain is possible in a secure and precise way and hence, it cannot be claimed by anyone else.

Review Questions

Q. 1 Define blockchain.

Q. 2 Differentiate between blockchain and bitcoin.

Q. 3 Explain generations of blockchain

Q. 4 What is distributed consensus?

Q. 5 How blockchain transaction differs from traditional transaction? Explain with example.

Q. 6 What is peer to peer distributed network?

Q. 7 Explain working steps of blockchain.

Q. 8 What is Merkle root? Explain with example.

Q. 9 How public key cryptography plays role in security of blockchain? Explain with example.

Q. 10 What are the advantages of Merkle tree?

Q. 11 Explain various components of blockchain.

Q. 12 How Merkel tree ensures security in blockchain? Explain in detail with example.

Q. 13 What is nonce?

Q. 14 What is mining?

Q. 15 Explain different steps involved in mining.

Q. 16 What is wallet in blockchain?

Q. 17 What is hash and how it provides security in blockchain? Explain.

Q. 18 Explain different types of blockchain.

Q. 19 Differentiate between public, private and consortium blockchain.

Q. 20 What is full node and partial node in blockchain?

Q. 21 What is distributed ledger in blockchain?

Q. 22 Explain Byzantine Generals Problem.

Q. 23 What are the objectives of consensus protocols or mechanisms? Explain.

Q. 24 Explain with advantages and disadvantages the working of proof-of-work (PoW) algorithm.

Q. 25 Explain in short Proof-of-stake (PoS) and delegated proof of stake (DPoS) mechanism.

Q. 26 Explain with advantages and disadvantages the working of Proof of Authority (PoA) algorithm.

Q. 27 Explain with advantages and disadvantages the working of Practical Byzantine Fault Tolerance (pBFT) algorithm.

Q. 28 Explain in short various consensus algorithms.

Q. 29 Explain in short Proof of capacity consensus mechanism.

Q. 30 Explain working of RAFT algorithm.

Q. 31 What are the Limitations and Challenges of Blockchain? Explain in detail.

Q. 32 What are the benefits of Blockchain technology? Explain in detail.

Q. 33 What is block in blockchain? Explain

Q. 34 Explain blockchain ecosystem in detail.

□□□

2

MODULE - 2

CRYPTOCURRENCY

Syllabus

2.1 Cryptocurrency : Bitcoin, Altcoin, and Tokens (Utility and Security), Cryptocurrency wallets: Hot and cold wallets, Cryptocurrency usage, Transactions in Blockchain, UTXO and double spending problem.

2.2 Bitcoin blockchain : Consensus in Bitcoin, Proof-of-Work (PoW), Proof-of-Burn (PoB), Proof-of-Stake (PoS), and Proof-of-Elapsed Time (PoET), Life of a miner, Mining difficulty, Mining pool and its methods

2.1 Introduction

- The meaning of two words crypto and currency is: "crypto" means concealed or secret and "currency" is system o money in general use in a particular country. The very first cryptocurrency is bitcoin. The main aim behind these cryptocurrencies or digital currencies to have a faster, cheaper and reliable currency compared to money Cryptocurrency is an open- source encrypted database/ledger of transactions which is built on Blockchai technology. Mining process is used to create the cryptocurrency in Blockchain network. The members o participants can directly transact with each other over online platforms.
- In cryptocurrency transactions, no need to create money by government and third party such as banks are n involved in money transactions. This removal of intermediaries makes faster transactions with reduced cost. User involved in cryptocurrency transactions can record and verify each other's transactions. Unless the network accep the transaction, it cannot be committed to distributed ledger.
- This also ensures prevention from any fraudulent activities by intruders. This ledger containing the transactions a replicated on all the nodes and hence, they are public and secure. The trust is offered by Blockchain netwo without any trusted intermediatory or third party. Cryptocurrency is still in its early stages and considered more a an investment or commodity instead of currency. Now days, cryptocurrencies are gaining popularity. Apart fro Bitcoin, other cryptocurrencies that are gaining popularity are Litecoin, Ether and tether.

2.2 Bitcoin

- Now days, people more prefer to use credit/debit cards, mobile payments or net banking. Hence, use of currency notes is decreasing day by day. This encourages use of secured decentralized digital currency by anyone i Blockchain network. Banking system plays important role in economy. All the transactions that involve online payment, involve the bank as an intermediatory to offer the trust.
- People needs to have bank accounts and they have to pay service fee for the transaction purpose. The financial institutions incur the significant cost due to several expenses that are involved in maintenance activities. In many situations, customer's money does not remain safe in banks due to some fraud cases. Government issues restrictions on withdrawal of money. Hence, there was a need for financial system that could :
  - Allow to transact with other person directly without involvement of third party like bank and allow verification and validation of the transactions, and thus saving the cost of third party.

- Operate without control of central authority ensuring the value of money is maintained
- Preserve the transparency in transactions with maintaining user's privacy.

- The first cryptocurrency bitcoin as open-source software was introduced in 2009. It allowed to directly transact with other person without involvement of third party by maintaining trust with the use of cryptography. Hashing creates identifier (hash) which is unique. In blockchain, these identifiers are used for block, transactions and addresses. In bitcoin, hashes are of 256 bits or 64 characters and Blockchain uses Secured Hash Algorithm that generates 256-bit hash (SHA-256). There is bitcoin wallet provided for making transactions through it. There is a fixed number of bitcoins that can always be in circulation, which is fixed at 21 million. This is in contrast to government currency, which reduces value of money already in circulation if more currency is printed and brought in circulation. The value of bitcoin is dependant only on market supply and demand.
- The validation and verification of the transaction is done by Blockchain network. The transaction is broadcasted to whole network and only accepted if majority of nodes are agreed upon it. This ensures security and prevents duplication of bitcoins. Santoshi Nakamoto released a white paper titled "A peer-to-peer Electronic Cash System". In 2009, mining of Bitcoin Genesis Block was carried out and first application of public Blockchain was launched. In the same year, a Bitcoin was recognized as digital currency. After this, several developers developed their variation of cryptocurrency by forking to improve upon. Examples are Bitcoin XT, Segwit, Bitcoin Cash etc. As of January 2020, over 2.8 million bitcoins are yet to be mined and the value of bitcoin is 8,000 USD.

2.3 Cryptocurrency Basics

There are thousands of digital currencies apart from bitcoins. All these currencies are called as altcoins (alternate coins) or coins which are used for buying or selling the products or services. It is noticed that, till October 2019, more than 3000 active cryptocurrencies exist. Bitcoin is related to currency transactions whereas Cryptocurrency has now found many user cases in many domains such as finance, security, supply chain management, ownership, health management, records management and research management.

2.3.1 Forking

Cryptocurrencies like Bitcoin and Ethereum are powered by decentralized, open software that anyone can contribute to called a blockchain. These blockchain is an open source, hence anyone can maintain and develop their underlying code. If anyone updates the blockchain's protocol or alters basic set of rules then we say that fork is happened.

What is fork and why fork is used?

- Fork is used to copy one type of cryptocurrency to create an entire new type of cryptocurrency. We can download the Bitcoin software in order to run our own code and we don't need anyone's permission to do so. The software is open source and available to anybody who wants a copy. As long as you are able to complete the necessary technical tasks involved and recruit enough miners to form network of nodes in order to process your transactions, nobody can stop you to create your own version of Bitcoin under a different name.
- Cloning an existing blockchain based cryptocurrency in this way and releasing it is called Forking.
- There are number of forks of existing cryptocurrencies in existence to date, such as Bitcoin Cash, Bitcoin Gold, and Ethereum Classic.

- Forks usually try to differentiate themselves from the original crypto application that they are copy of. Usually forks occur when different parts of the development community, which support the original cryptocurrency disagree on future technical developments.
- It's fair to say that most Bitcoin and Ethereum forks have failed to rival the success of the original projects they were based on.

2.3.2 Characteristics of Cryptocurrencies

All kinds of money that are made legal tender by a government decree or fiat is called fiat currency. Both the fiat currency and cryptocurrency are money used to buy and sell the goods, services etc. Both form of currency is governed by supply and demands along with other economic factors. Following are some important characteristics of cryptocurrencies.

1. Decentralized

Fiat currency value can be influenced by government policies or decisions. This currency is regulated by central authority such as central bank. Cryptocurrency is not controlled by any bank or central government. All the participants in network are involved in processing of transactions. Hence, value of cryptocurrency cannot be influenced by government policies or decisions.

2. Digital form

Cryptocurrency exists only in digital form. It is implemented with software code and secured algorithms. Fiat currency exists in both form: physical and digital.

3. Limited Supply

The maximum supply of crypto that can ever be generated at the time of creation of genesis block.

4. Global Access

In Blockchain distributed network, anyone can transact and access the cryptocurrency from any location. The participants across any geographical area involves in transaction processing. It results in quicker transaction processing without paying any transaction fee to third party.

5. Anonymity and Transparency

The personal details of the members are not required in cryptocurrency transaction. Whereas, in bank transaction, all these details are stored in bank database. In cryptocurrency, transactions are linked to cryptocurrency address and not to the personal details. Every encrypted record is maintained in distributed ledger which can be trustworthy, transparent and verifiable. Hence, both anonymity and transparency is provided in Blockchain network.

6. Impossible to duplicate

Use of cryptographic algorithms and consensus protocols makes duplication of cryptocurrency impossible in distributed network.

7. Immutable

With this feature in blockchain, it is not possible to alter any recorded transaction. Altering the data is so challenging and nearly impossible, this is a benefit to maintaining an immutable ledger of transactions. Hence any transaction is irreversible.

2.4 Cryptocurrency Wallets

- Wallet manages cryptocurrencies. Blockchain participants can send or receive cryptocurrency with blockchain wallet. If user creates private key, then its public key pair is also gets created. Both keys are then stored in wallet which is nothing but software program. This wallet is also used to send and receive the cryptocurrency in Blockchain.
- Following is the classification of cryptocurrency wallets. Cryptocurrency wallets are mainly classified as **Hot wallets and Cold wallets**. Hot wallets are further classified as Online/web wallets and software wallets, which are of two types : Mobile wallet and Desktop wallet. Two types of Cold wallets are Paper wallet and Hardware wallet. Following figure shows the classification of wallets.

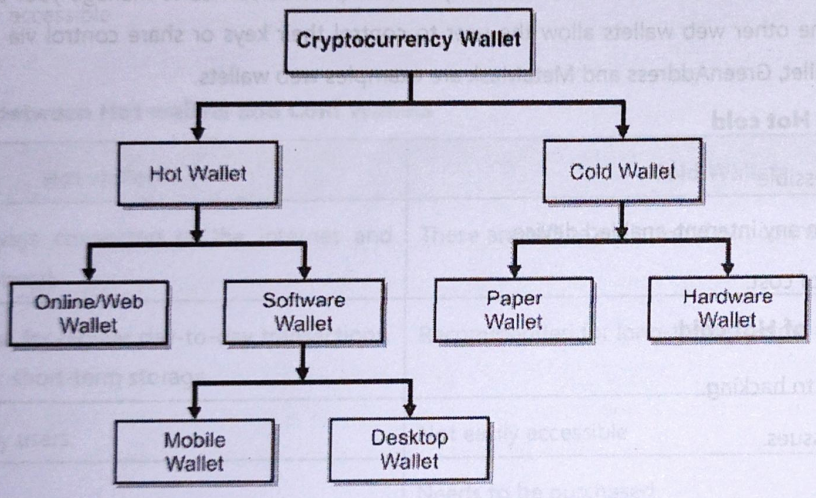


Fig. 2.4.1 : Types of wallets

1. Hot Wallet

A hot wallet is a cryptocurrency wallet that is developed for online daily transactions to be carried out. This wallet is always connected to the internet and cryptocurrency network. Hence, it is more prone to attacks by attackers or strong candidate for hackers in network.

Hot wallets are used to send and receive cryptocurrency, and they permit you to view how many tokens you have available to use. These are free wallets, easy to use and more convenient for users. These wallets have different installation characteristics. On this basis, these are categorized as: online/web wallets, mobile wallet and desktop wallet.

- **Desktop Wallet :** These wallets you can download and install on your desktop and laptops. Users have complete control on this wallet as they are locally stored. One advantage to desktop crypto wallets is that you are in control of your private keys and hence on your funds. User do not have to depend on a third party to hold cryptocurrency. Some better desktop wallets are password protected and supports a recovery phrase to facilitate the regeneration of your crypto wallet if your desktop or laptop is stolen or destroyed. These wallets rely on security features that are available on your desktop or laptop. Bitcoin Core, Electrum and Exodus are examples of desktop wallets.

- **Mobile Wallets :** Mobile crypto wallets are downloaded on your phone or tablet. Some mobile crypto wallets support Android and others iOS, and some support both operating systems. These wallets permit you to trade while in control of your private keys. Other software wallets can have both mobile and desktop versions. You can carry mobile devices easily at any location. Hence, it provides easy accessibility and make purchases from merchants who accepts cryptocurrency payments. However, they are more vulnerable to security threats with compare to desktop wallets. Hence, security features should be available on mobile devices. Coinbase wallet, Mycelium and TrustWallet are some of the examples of mobile wallets.

- **Online or Web Wallets :** These wallets run on the cloud and requires web browser to access them from any device. These are more preferred by new cryptocurrency users. As the control of these wallets are with third party, they are more vulnerable. Some service providers provide service to manage your private keys on your behalf. Some other web wallets allow the user to control their keys or share control via multiple signatures. MyEtherWallet, GreenAddress and MetaMask are examples web wallets.

Advantages of Hot cold

- Highly accessible
- Available on any interent-enabled device.
- Often free of cost.

Disadvantages of Hot cold

- Vulnerable to hacking.
- Technical issues.

2. Cold Wallet

Usually, cold storage wallets are fairly secure. Stealing the currency from cold wallet usually would need physical control of or access to the cold wallet, in addition to any associated PINs or passwords that must be used to access the funds. These wallets are not connected to internet and used for long-term storage of cryptocurrencies. Two types of Cold wallets are Hardware wallets and Paper wallets.

- **Hardware wallets :** Hardware wallets are designed to be safe to hacking. In case if a hardware wallet is plugged into your machine or connected via Bluetooth, depending on the storage method, the cryptocurrency stored on the drive are hard or even impossible to steal. Though technically connected to the internet, the signing of transactions is done "in-device," and only then broadcasted to the network using internet connection of your machine.

- Hardware wallets are less convenient to use with compare to hot wallets as they must be powered on and then connected to the internet to send or receive payments. Hardware wallet is the device that use random number generator to generate public/private key that is stored in device. Trezor and Ledger wallets are the examples of hardware wallets.
- **Paper Wallets :** These are not online wallets. A paper wallet is a printed piece of paper containing keys and QR codes used to facilitate your cryptocurrency transactions. Because they are not connected to the internet, at one point, paper wallets were considered to be more secure than other forms of cryptocurrency storage. Some paper wallets contain a scannable barcode generated by application. With this, you can store and take access to your cryptocurrency offline. When you print your keys, they are removed from the cryptocurrency network, but the tokens remain; however, they are inaccessible without your keys. Paper wallets contain single public/private key pair which can be used single time for whole amount in one transaction. If you spend partial amount then remaining balance you have to lose.

Advanatges of Cold Wallets

- Extremely secure
- No issues involving internet.

Disadvanatges of Cold Wallets

- Not easily accessible
- Expensive

Comparison between Hot wallets and Cold Wallets

Hot wallets	Cold Wallets
This wallet is always connected to the internet and cryptocurrency network.	These are offline wallets and not connected to internet
These better to use for regular day-to-day transactions. Recommended for short-term storage.	Recommended for long-term storage.
Easily accessible by users.	Not easily accessible
Usage is free	Needs to be purchased
Susceptible to malware or cyber attacks	Stealing the currency from cold wallet usually would need physical control of or access to the cold wallet. No risk of hack or malware.
Software wallets and online wallets are the examples.	Hardware wallets and paper wallets are the examples.

2.5 Types of Cryptocurrency

Bitcoin, Altcoin and Tokens are the three types of cryptocurrencies. Following diagram shows its classification.

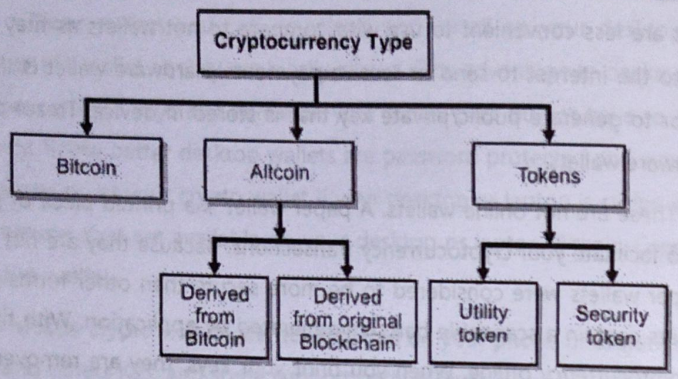


Fig. 2.5.1 : Types of cryptocurrency

2.5.1 Altcoins

- Bitcoin was first introduced in 2009 by Satoshi Nakamoto. It has since become the most well-known and widely used cryptocurrency in the world. Its acceptance has inspired the development of many other cryptocurrencies. Since the introduction of bitcoin in 2009, its many alternates have created using its open-source protocol introducing new currency coins having different set of features that works on its blockchain. Some of the examples of Bitcoin derived blockchains are : Namecoin, Peercoin, and Litecoin.
- Some currencies are created as native currency of an original blockchain and protocol. These are not created from bitcoin. The examples are Monero, XRP (Ripple), NxtCoin etc. All these alternative cryptocurrency coins called as Altcoins. These are alternative to bitcoins.
- Hence, an Altcoin is an alternative digital currency to Bitcoin and the word Altcoin is a combination of "alternative" and "coin", to form "altcoin". Some of the most well-known Altcoins are Ethereum, Ripple, Tether, Bitcoin Cash, Bitcoin SV, and Litecoin. Altcoins are peer-to-peer digital currencies that involves mining process. Each coin has its own independent blockchain. All are having characteristics just like money and used for payments.

2.5.2 Tokens

- Tokens or crypto tokens are units of value that blockchain-based organizations or projects develop on top of existing blockchain networks. These tokens share deep compatibility with the cryptocurrencies of the underlying network, but they are a completely different digital asset class. Token facilitates the creation of decentralized applications (DApps).
- Tokens and their underlying applications benefit from the technology and security of the native blockchain. Cryptocurrencies are the native asset of a particular blockchain protocol, while the tokens are created by platforms that build on top of those blockchains. In order to create the tokens, standard templates from blockchain platforms are used. The Ethereum blockchain's native token is ether (ETH).
- While ether is the cryptocurrency native to the Ethereum blockchain, there are many other different tokens that also utilize the Ethereum blockchain. Crypto tokens built using Ethereum include DAI, LINK, COMP, and CryptoKitties, among others. The functional variance is possible through use of the smart contracts. These tokens can serve a multitude of functions on the platforms for which they are built, including participating in decentralized finance (DeFi) mechanisms, accessing platform-specific services, and even playing games.

- There are many broadly used token standards to create the crypto tokens. The majority of these have been built on top of Ethereum. The most widely used token standards are ERC-20, which allows the creation of tokens that can interoperate within Ethereum's ecosystem of decentralized apps. Status (SNT) and OmiseGO (OMG) are the widely known ERC-20 tokens.
- Other standard ERC-721, was developed to enable non-fungible tokens that are individually unique and cannot be interchanged with another similar token. The main purpose of tokens is its use as utility or asset to give the holder specific rights to participate in network. They may represent the physical objects such as house or painting. WPR token represents electricity and permits public to buy and sell electricity using smart contracts. In the following ways, altcoin differs from tokens.
  - Tokens are not operational like cryptocurrencies. They are used for specified service only. For example, you cannot use cinema ticket as a coupon for dinner.
  - You can purchase tokens with coins but cannot purchase coins with tokens.
  - Tokens are easier to create than coins.
- Tokens are created and distributed to public through crowd funding mechanism known as Initial Coin Offering (ICO). Following are the two types of tokens.

1. Utility Tokens

- These tokens provide the rights to its holder to use a service. A utility token can serve just about any purpose a developer wants it to. In general, utility tokens offer access to a specific service or product with a blockchain ecosystem. Certain utility token may be used to perform actions on an altcoin's network.
- Users with utility tokens receive exclusive benefits like discounted transaction fees and early access to products and services on the platform. These tokens are given out during crowd sales as a project executes an Initial Coin Offering (ICO). Creating utility token is just like a creating form of a digital coupon which can be exchanged in the future for discounted fees or special access to a product or service.
- These tokens are not used as investments Some of the Examples of utility tokens are Filecoin, Siacoin, Civic, etc. ERC20 is another type of utility token which can be used as technical standard for all smart contracts on the Ethereum blockchain for token implementation.

2. Security Tokens

- Security tokens does not have a utility. A security token signifies some kind of ownership, most commonly a share of the company that issues the token. As the concept is just like purchasing shares of stock on a traditional stock exchange, the security tokens are sometimes called as equity tokens. Security tokens are viewed as securities by financial regulatory authorities; hence, they are subject to regulations similar to ordinary stocks and bonds. Investors from the world of traditional finance might prefer the fact that, security tokens offer all the legal protections and regulatory clarity they are familiar to.
- Security tokens are offered during Security Token Offering (STO). STO is a variation of ICO and offers more regulations and due diligence controls encouraging investors confidence. In USA, anything that fulfilling the definition of a "security" falls under the regulatory purview of the Securities and Exchange Commission (SEC). Many countries have established regulations around security tokens.

- Securities and Exchange Commission (SEC) makes use of "Howey Test" which involves four criteria to determine whether or not something qualifies as being a security.
  - Someone has invested in goods or services.
  - Investment is in common enterprise.
  - There is an expectation of profits either from fixed returns or capital appreciation.
  - If any profit involved stems from the efforts of the promoters of the investment.
- It is time consuming to launch the security tokens because of regulations. On the contrary, regulations avoid the fraud and scams which may happen with investors. Examples of security tokens are BCap of Blockchain Capital, 22X Fund, and Spice VC. Bitcoin and Ether are not yet considered as securities. But many others such as Ripple's XRP is under review.

Comparison between Altcoin, Utility Tokens and Security tokens

Altcoin	Utility Tokens	Security tokens
These currencies are created using open-source software of an existing currency or as native currency of an original blockchain and protocol.	In order to create the tokens, standard templates from blockchain platform are used. They are created via smart contracts.	In order to create the tokens, standard templates from blockchain platform are used. They are created via smart contracts.
Altcoins are used for payment purpose.	Utility tokens are created for funding blockchain projects or companies. These tokens are given out during crowd sales as a project executes an Initial Coin Offering (ICO).	Utility tokens are created for funding blockchain projects or companies. Security tokens are offered during Security Token Offering (STO).
It is currency or cash	It represents right to use service or product in future	A security token signifies some kind of ownership, most commonly a share of the company that issues the token.
Value depends on market supply and demand	Used for utility. Users with utility tokens receive exclusive benefits like discounted transaction fees and early access to products and services on the platform. Users also gets voting rights etc.	Token holders gets part ownership. Value is directly linked to the valuation of company.
Security depends on strength of blockchain protocols.	Susceptible to ICO scams and frauds	Regulations prevents from scam and frauds.
These are not regulated or controlled by any authorities.	Typically unregulated.	Strictly Regulated by financial market authority.

2.5.3 Popular Coins and Tokens

As per CoinMarket, Bitcoin is still outperforming in key financial assets such as equities, commodities and bonds. Following are coins/tokens which are more preferred today in cryptocurrency world along with bitcoins.

1. Ethereum

- Ethereum is now to be the second most popular cryptocurrency, beaten now only by Bitcoin. As said by, The Motley Fool, the Enterprise Ethereum Alliance (EEA) has some big-name founding members, including Microsoft, Intel, and JPMorgan Chase. CNBC has confirmed its adoption in news, means it could eventually be bigger than its early-stage competent.
- Hence, it is important to know Ethereum platform along with its features and applications, and how it differs from Bitcoin. Ethereum platform supports smart contracts which encourages developers to develop DApps without any centralized development platform. It uses PoW as a consensus mechanism. Currently running Ethereum is hard fork of traditional Ethereum.
- It was forked due to hacking in decentralized Autonomous Organization (DAO) in 2016. This hacking was resulted in loss of 3.6 million Ether. Original Ethereum is now referred as Ethereum Classic and its crypto ether as ETC.

2. Ripple XRP

- Ripple plays both role as a cryptocurrency and a digital payment network for financial transactions. It was first released in 2012 and was co-founded by Chris Larsen and Jed McCaleb. This blockchain was developed by Ripple Lab Inc. XRP is the inborn cryptocurrency for products implemented by Ripple Labs.
- Its products are used for payment settlement, asset exchange, and remittance systems that work similar to SWIFT, a service for international money and security transfers used by a network of banks and financial intermediaries. XRP is pre-mined and uses a less complicated method of mining as compared to Bitcoin.
- Total 100 billion XRP coins are available today out of which 43 billion are in circulation. Remaining coins are released time to time as set by in built smart contract. Ripple network uses Practical Byzantine Fault Tolerance (pBFT) consensus mechanism to verify and agree on authenticity of the transaction.
- Consensus is reached if 66% of validators agree on it. Three to five seconds are required to confirm the transaction. Ripple is the corporation that is behind XRP, the cryptocurrency. It seamlessly transfers both fiat and cryptocurrencies. RippleNet is the association of more than 300 financial institutions worldwide which facilitate low-cost cross border payments.

3. Bitcoin Cash (BCH)

- Some Bitcoin users and developers desired to circumvent the upgrades that SegWit brought about. Hence, they created a hard fork called Bitcoin Cash. This fork was then split from the main blockchain in August of 2017. Bitcoin Cash uses the same PoW protocol as Bitcoin.
- It is one of the most effective forks, assisted by many prominent figures. Today, Bitcoin Cash is the 25th largest cryptocurrency by market cap, trading above $300. The main aim behind the forking was to increase block size from 1 MB to 8 MB to improve transaction times and scalability issues.
- This increased block size resulted in higher cost in running the full node on a blockchain-based network. This is the drawback as few groups or associations have resources to run full node. Again, hard forking is carried out on Bitcoin Cash in 2018 that resulted in Bitcoin Cash ABC or BCHA and Bitcoin SV. BCHA offers a speedy, more scalable, global cryptocurrency in a way that is workable in the long term.

4. Litecoin (LTC)

- Litecoin cryptocurrency was introduced in 2011 by Charlie Lee, a former Google engineer. It was created from fork in Bitcoin blockchain.
- The main aim behind creation of Litecoin was to improve on Bitcoin. For one, Lee introduced a new memory intensive hashing algorithm for Litecoin called as "Scrypt". This algorithm improved Litecoin's transaction speeds. The transaction processing speed of Bitcoin is roughly five transactions per second.
- Creation of new blocks on the Bitcoin blockchain requires nearly 10 minutes or more. Whereas, the transaction processing speed of
- Litecoin's is 54 transactions per second. Creation of new block on the Litecoin blockchain takes nearly 2.5 minutes. The supply of Litecoin is capped at 84 million coins. Whereas, Bitcoin has a supply capped at 21 million.

5. Eos (EOS)

- Electro-Optical System (EOS) was launched in June 2018. This was launched by Block which is software company specialized in Blockchain Technology. EOS support decentralized applications (DApps). EOS crypto tokens are earned in a different manner with compared to Bitcoin mining.
- The needed number of blocks users can create and gets reward with fresh EOS coins for each block created. The EOS blockchain is considered to be main competitor to Ethereum. It combines positive features of both Bitcoin and Ethereum. It is also smart contracts Blockchain development platform.

6. Monero (XMR)

Cryptocurrency like Bitcoin and Ethereum are transparent

- Monero (XMR) is an open-source digital token. It was introduced in 2014 and offers anonymous transacting features. Bitcoin, Ethereum and other many cryptocurrencies, have transparent blockchains in which transactions can be verified and/or traced by anonymous.
- Hence, it can be possible to link sending and receiving addresses of the transactions to real-world identities of the anonymous. Monero, in contrast, implements several privacy-enhancing technologies to guarantee the anonymity of its users by default. Monero is in the topmost 30 coins having market cap of over $2 billion and uses a proof-of-work (PoW) consensus algorithm.
- Monero eases privacy issues using the concepts of ring signatures and stealth addresses. Ring signatures allow a sender to hide their identity from other members in a group. Ring signatures belongs to only one member of the group and remains anonymous digital signatures.
- The identity of the member signing a transaction remains unknown. This ring signature is generated by combining the sender's account keys and then it is clubbed with public keys on the blockchain. Therefore, ring signature is unique and private.
- The identity of the sender remains hidden because it is computationally impossible to determine which of the group members' keys was used to create the complex signature. There is no predefined block size in monero. The block rewards get reduced to prevent the malicious miner.

7. Stellar (XLM)

- Stellar is an open-source, cross-border payments network system which works with both fiat and cryptocurrency. Stellar was introduced by two business partners Jed McCaleb and Joyce Kim in 2014. Stellar maintains a ledger to track the accounts and crypto tokens ownership on its network.
- It processes transactions by using its own unique mechanism, the Stellar Consensus Protocol (SCP). SCP supports for fast transaction processing. After every 5 seconds, all the balances and operations are broadcasted to whole network. The nodes with core Stellar software validates the transactions.
- If the transaction is pending then nodes verify it to assess if correct amount is transferred or not. XLM token is the native token of the Stellar network.
- SCP permits secure Federated Byzantine Agreement (FBA) blockchain protocol which is an implementation of a consensus mechanism by Ripple. The native currency of stellar is Lumens (XLM).

2.6 Cryptocurrency Usage

2.6.1 Ecosystem Players

There are different players in blockchain ecosystem which plays different roles directly or indirectly. These are :

1. Programmers or Developers

Many blockchain projects are under development. Its success of any project depends on team working in that project including programmers and developers It is the job of the core blockchain developers to develop and maintain the architecture of blockchain systems. They design protocols, develop security patterns, and manage the network as a whole. Blockchain software developers implements applications onto existing blockchain platforms. They are also responsible to manage front-end and back-end development, design, and maintenance. The developers of DApps works with decentralized applications and smart contracts. Developers also develops the consensus mechanism required for blockchain platform.

2. Miners

In mining process, new transactions are added to the large distributed public ledger of existing transactions called as block chain. The person who carries out mining is called a miner. Miners validate new transactions and record them on the blockchain. Miners compete to solve a complex mathematical problem based on a cryptographic hash algorithm. Miner also participates in cryptocurrency transactions, and plays important role in creating new cryptocurrencies and in verifying transactions on the blockchain. Miners who mine the block is either predefined or obtains the privilege. They also get rewarded with block or transaction fee.

3. Users

Blockchain user uses blockchain or cryptocurrency. User can be any participant or member in the network. User can be miner, validator or investor.

4. Merchants

These are business entities that accept cryptocurrency as a form of payment for their product or services.

5. Traders

Trading of the cryptocurrency can be carried out using CFD or via an exchange. Traders in cryptocurrency take risks, buy or sell the crypto based on price movement via a decentralized network.

2.6.2 Cryptomining

Crypto miners receive reward in terms of cryptocurrency for mining a new block on the blockchain. The miners have to solve complex mathematical problem based on a cryptographic hash algorithm. It is the competitive process that verifies and adds new transactions to the blockchain for a cryptocurrency which uses PoW consensus mechanism. The winner miner is rewarded with some amount of the currency and/or transaction fee. Although, the mining process varies in different Blockchain, they use consensus mechanism and incentive system.

In 2009, bitcoin was introduced and each block reward was worth 50 BTC at that time. In May 2020, the block reward was shared a third time to 6.25 BTC. In case of Ethereum, it is 2 ETH along with some other fee. Since May 2021, there were already 18.7 million bitcoins in existence. In order to reward the miner, a native cryptocurrency of the Blockchain is used. Fir example BTC in Bitcoin, LTC in Litecoin etc. Two main types of mining are solo mining and pool mining. These are based on whether miners mine individually and independently (solo mining) or they collaborate with each other (pool mining).

1. Solo Mining

- Crypto mining can be great source of income although it is somewhat confusing process. The different hardware, software, and hash rates needed for the same is overwhelming in both solo and pool mining. In solo mining, you mine cryptocurrency on your own without involving third party. The responsibility of an entire mining process lies with individual miner. Initially, Bitcoin mining was carried out by the mysterious person Satoshi Nakamoto.
- In solo mining, miner is in network with his/her wallet. Once block hash is approved by all the members and its inclusion in block is confirmed, miners get rewarded with cryptocurrency. The benefit is that, the whole reward value of the full block gets credited to miner's account. With compare to pool mining, individual miner may take long time to find a block due to shortage of resources.

2. Pool Mining

- Pool mining means mining along with other people in which group of participants pool their cryptocurrency together for a better chance of winning new blocks. To nullify energy consumption and high time in the validation of transactions, some miners combine mining pools together to share the resources for efficiency.
- In the pool mining process, every miner combines their computing power to increase the possibilities of mining a block. The reward is distributed among all the miners involved in mining on the basis of their contribution in processing power or as per agreed upon terms.
- Hashes from all the miners are submitted to pool and anyone's hash wins to create a block. Antpool, BTC.com and F2Pool are the some of the examples of popular mining pools.

3. Multipool mining

- Multipools shift between different altcoins and continually compute which coin is at present is the most profitable to mine. Algorithm calculates profitability by considering the block time and the price on the exchanges.

- Instead of keeping the different wallets for all likely minable coins, multipools may instantly exchange the mined coin to a more recognized coin such as bitcoin. In this scheme, as the most profitable coins are being mined and then sold for the intended coin, it is possible to obtain more coins in the intended currency than by mining that currency alone.
- This method rises the demand on the intended coin, having the unexpected result of increasing or stabilizing the value of the intended coin. Companies selling the hash power may carry out this by combining the work of many small miners (for example, NiceHash), paying them proportionally by share like a pool would.
- Some of these companies run their own pools. These are multipools, as they generally use a similar scheme of work switching, though the work they assign is determined by client demand instead of "raw" profitability.

There are other mining pools based on the resources used by miners.

- **CPU Mining :** CPU mining is a mining process that makes use of central processing unit cores to check blockchain transactions, solve mathematical problems, verify transaction blocks. In this mining process, miners use CPU to mine the cryptocurrency. When the bitcoin was introduced, it was feasible to use CPU for mining. As mining complexity increased over the period of time, the process was becoming slow, consuming more electricity and other resources. Hence, today this mining is not financially viable solution.
- **GPU Mining :** In GPU mining, miners use Graphics processing unit to carry out block validation. GPUs have been used in the mining process as they are more efficient than the CPU. It offers more computing power and cost effective with compare to CPU. GPU, for example, Radeon HD 5970, executes 3,200 32-bit instructions per clock, which is 800 times more than the speed of a CPU that executes 4 32-bit instructions per clock. GPU hardware can be used to mine the coins like Ethereum, Zcash etc.
- **ASIC Mining :** An application-specific integrated circuit (ASIC) is used by miner to validate the block. ASIC miners are nodes designed only for the purpose of mining cryptocurrencies. The miners are designed for one hashing algorithm, and hence, it decides which cryptocurrencies you can mine. For instance, Bitcoin is mined through the SHA-256 algorithm and Ethereum is mined through ETHASH. AISC miners are faster, efficient and hence requires less time for mining the coins with compare to other processors. It is having smaller physical size than CPU and GPU and also consumes less power. their resale value is low as they are not upgradable as per new versions.
- **Cloud Mining :** In this mining process of cryptocurrency, any mining equipment or hardware is not used directly. In cloud mining, you can rent the computing power. Any miner or cloud company rent their mining services to other miners. In this case, miners no need to buy the equipment. Some of the well-known models for cloud mining are hosted mining and leased hash power.

2.6.3 Airdrop

It is very much important in business to attract consumers in order to improve the market base of the business. This is also applicable to the concept of cryptocurrencies. Airdrop is marketing strategy or promotional activity to attract the blockchain users towards new projects in order to stand in the market place. In this activity, native cryptocurrency is distributed with no charge (free) among current or potential users. Sometimes, users are asked to fulfil simple promotional activities before their claim. For example, following the project's social media account and sharing their posts. These cryptocurrencies are distributed to the user's wallet address so that market for the project launched can be created. In some cases, manual claim is also required.

Benefits of Airdrop

As there is a risky competition among cryptocurrency startups, an airdrop helps to stand out from the crowd. Some businesses are also involved in crypto airdrops offer alerts to users and listing services to startups, in addition to a range of marketing services to fine-tune their airdrops. Following are the benefits of Airdrops.

- **Marketing and Publicity :** Airdrop is the way to people interested in cryptocurrency to create publicity around ICOs and projects. As people receives free coins in wallets, they advertise about the project through various medium such as mouth publicity, social media, forums and many more. This helps to increase the participants/members of the new project.
- **Rewards to loyalty supporters and Investors :** Some cryptocurrency companies use the airdrop as an exchange and trading platform in order to encourage the users by allotting the incentives for continuing buying and use on their platform. If user have good amount tokens in wallets, then they are awarded with free tokens. These tokens are allowed to use in further promotional activities through various medium.
- **Equal distribution of Tokens :** This activity of airdrop, evenly distribute tokens among existing token holders, participants in ICO, and project. Hence, it supports for wide promotion.
- **Database Creation :** As claim of the free coins/token requires filling of the form, database containing various users' information remains available with blockchain company. This information can be used for further promotional activities.

Types of Airdrops

There are two ways to carry out the airdrops. These are **surprised airdrop** and **planed airdrop**. In surprised airdrop, company takes snapshot of the blockchain on particular date and distributes free coins to the wallets based on balance in each blockchain address. New enterprises carry out the planned airdrop. In this case, blockchain start-up declares date and time of the airdrop. If users carry out some of the actions before airdrop, then they become eligible to receive coins. These actions include KYC form filling, bringing the new member, social media promotion etc. There are several types of airdrops. These are:

Standard Airdrop :

In this airdrop, members receive free tokens just for signing up for a newsletter or something similar. Small amounts of coins are deposited for free to wallet users if, the wallet address or signing up with the new crypto platform would be sufficient for standard airdrops.

- **Bounty Airdrops :** Here, user require to carry out simple tasks to receive coins or tokens. The wallet user has to involve in some promotional activity in exchange for the airdrops. Fir example, social media promotion.
- **Exclusive Airdrops :** These airdrops are exclusively for people having an established history with a particular project, website, or community. For example, Uniswap allotted its loyal users 2500 UNI tokens in September 2020.
- **Hard Fork Airdrop :** A new coin is created after hard fork. Users holding the original coin will receive an equal amount of the new tokens in their wallets. The most famous example of this would be the Bitcoin Cash (BCH) hard fork that happened in 2017: Bitcoin users who were holding BTC received an equal amount of BCH automatically.
- **Holder Airdrop :** These airdrops are close to hard forks in that user holding certain tokens will receive new ones. EOS and Ethereum have sometimes given users free tokens after creation of new project on one of their blockchains. These are totally new projects created on top of the EOS or Ethereum protocol.

Disadvantages

- Cryptocurrency airdrops have some limitations. For example, crypto airdrops can be pump-and-dump schemes. That is, cryptocurrency owners could be insincerely inflating its value in order to make a fast profit. Sometimes, fraudsters may lure people via airdrops to phish their MyEtherWallet's private keys and steal their funds. User should not handover their wallet's private keys to anyone.
- Also, make sure the URL of your wallet before inputting your private keys to log in, because some fraudsters will use sites that look exactly like your wallet's login page to phish your private keys.
- ICO founders follow the trend and get attracted to raise their social media numbers and telegram channel to raise funds, which really not happens. Hence, using airdrop cannot be good idea for marketing at this stage when conducting an ICO. For example, the Ubex Token Sale are still struggling to hit the hard cap although their Telegram channel hit 100K people, 60K followers on Twitter and Facebook.

2.6.4 Tokens or Coin Burning

- In cryptocurrency burning process users can remove tokens/coins from circulation, which reduces the number of coins in circulation. In this process, the tokens are sent to a wallet address that only receives the coins and cannot be used for transactions. This wallet is not reachable as it is outside of the network, and the tokens can no longer be used.
- The airdrops are carried out to increase awareness about crypto coins and on the contrary, burning is for limiting the existing supply of coins/tokens in order to increase their value. The wallet address at which coins are sent to burn is called as eater address or burner address. This address is without the private key and members are aware about this address as transparency is maintained in blockchain. Members know about transfer of coins to this address and at the same time that, address does not belong to anyone. This burning of coins is transparent to users and all the members knows that; coins are legally burned. This is called as proof of burn. Burning process is carried out to increase the value of remaining coins in circulation. Following are the most important reasons to carry out burning.
  - Coin holders gets benefit of their held coins/tokens. Enterprises use this approach to avoid the regulatory bodies so that tokens will not be considered as security tokens.
  - ICOs with excess (unsold) tokens, burn their tokens. The value of burned tokens/coins is then distributed to token holders. This ensures the promise of project team towards long-term achievement.
  - In proof of burn (PoB) consensus mechanism, miners get rights to mine and validate transactions. These miners are selected on the basis of coins/tokens they burned. More burning for benefits of network here increases chances of becoming validator for next block. This ensures equal chance for every member to become validator. PoB can leads to miner centralization problem. This is due to the fact that, the members with sufficient resources can success in more burning. It is also not guarantee in PoB to recover full value of burned coins.

2.6.5 Investing and Trading

Public can earn more cryptocurrencies if they follow the principal of buying at low price and selling at high price. Investors focuses on long-term profits. On the other hand, traders are more focused towards the short-term goals. As

cryptocurrency market is more volatile compared to stock market, either huge profit or huge loss can be the outcome of investment in short period of time.

Investing in Cryptocurrency

Investors study the company's financial statement, research reports and many more other details to take decision for investment. Investors aim is to hold on invested cryptocurrency for long duration to earn the profit. Investors also regularly keeps updates about potential ICOs, hard forks, and airdrops in order to invest the cryptocurrency.

Trading in Cryptocurrency

Traders takes more risk than investors and trading is for short period. Traders may observe the underlying cost of the assets they trade, but it doesn't matter to them if a cryptocurrency they are trading will grow or not since trading can be money-making regardless of whether an asset is flourishing or floundering.

This can be benefit in some situations. For example, it means traders can make a profit in both bull and bear markets, however long-term investors have to wait out bear markets. Trading, particularly in cryptocurrency, requires nerves of steel, cautious planning, discipline, and observing the markets closely and regularly. Therefore, many successful traders are experts who commit most of their waking hours to trading.

2.6.6 Cryptocurrency Safety

In case of fiat currency, bank takes the responsibility to track down the culprit for any fraudulent activities that happens with customer. In these cases, sometimes transaction is reversed provided fraud is verified or customer is repaid for the same. In cryptocurrency transactions, customer is solely responsible for all the transactions. He/She is the only owner of the cryptocurrency they hold. The executed transactions can not be reversed in cryptocurrency world. The verification about transaction is possible only with private keys. If they are compromised then it is impossible to discover the actual root cause of the fraud. Hence, following are some safety measures to consider while transacting,

Use of Exchanges

- Choose regulated exchanges that are more secure. Some of the crypto exchanges are Binance, Bittrex, and Coinbase.
- Avoid to store your cryptocurrencies on exchanges for more than a day or two as they store your private keys on the behalf of you. The accounts are also encrypted with these private keys. In 2019, $170 million were hacked by hacking ten crypto exchanges.
- Do not execute one large trade. Keep conducting smaller trades.
- It is preferred to have separate email address for different exchanges.

Storing Cryptocurrency

- For short-term, desktop or mobile wallets are best options. But, for long-term always prefer paper and hardware wallets.
- Do not use wallets having no control of your private keys.
- Do not keep large amount in mobile or online wallet.
- Keep the asset on several wallets or devices so that risk of loss can be minimized.

- Keep back up of each transaction executed.
- Always generate seed phrase. Reputable hard wallets generate 12 words. This is the method to back up and store your coins. You can recover your coins by downloading same wallet software and performing the seed phrase.
- Take all the back up regularly in hard wallet and do not keep it in any online medium.

Safety of Transactions

- Always keep studying the requirements that different blockchain transactions needs. In Ripple, address constitutes two parts: wallet address and destination tag.
- Avoid to access phishing sites and type manually URL to be accessed. Do not respond to links received through emails or through any means. Keep your password secret and do not share it with anyone.
- Do not publish your interest in cryptocurrency or investment done in it on any media
- Always remain watchful about your wallet address.

Security Measures to be taken

- Protect wallets and backups with strong passwords having combinations of all types of letters including capital letters, small letters, special symbols and digits etc. Password length should be at least 15 characters.
- Do not use same password for all accounts.
- Use two factor authentication to ensure security of your transactions.
- Use secured network for transaction purpose.
- Use your dedicated devices to perform transactions.
- Use all the software which are up-to-date and if not, update them.

Regulations Around Cryptocurrency

Most of the activities related to policies and regulations for cryptocurrencies are happened in the year 2019-2020. The Fintech bank also could not ignore the cryptocurrencies as demand was increasing. There are several standard-setting bodies and regulatory that have done important work related to cryptocurrency regulation and given recommendations. These bodies and regulatory are Financial Stability Board (FSB), International Organization of Securities Commission (IOSCO), Financial Action Task Force (FATF) etc. They suggested global regulatory framework for cryptocurrencies and virtual asset industry to better manage the benefits and challenges thereof.

2.7 Transactions in Blockchain

Transactions are the key component in bitcoin ecosystem. Transactions involves sending some bitcoins by one member to another member at bitcoin address, or it can be quite complex depending on the necessities. Each transaction comprises at least one input and output. Inputs are coins that are sent to other address and were created in previous transaction. On the contrary, outputs as coins being created. If a transaction is generating the new coins, then input is not required. Hence, signature is not needed. If a transaction is to transfer coins to some other user's wallet address, then the sender should sign it with their private key. For such transaction, it is necessary to have a reference to previous transaction in order to confirm the origin of the coins. Transactions are not encrypted and are publicly seen in the blockchain network. Blocks includes the transactions and anyone can view them using any online blockchain explorer.

Blockchain Transaction Life Cycle

Transactions are included in blocks which becomes part of the blockchain later on. Let's consider the single transaction as it passes through each stage in the process of joining the blockchain. Transaction involves sending of money by the sender which are received by receivers. The Blockchain transaction is a digital transaction. Following are the steps involved in bitcoin transaction.

1. Sender transfers a transaction through wallet which is software program or by using some other interface.
2. The wallet software uses sender's private key to sign the transaction.
3. This transaction is then broadcasted to the Bitcoin network. For this broadcasting, flooding algorithm is used.
4. Mining nodes consider this transaction to add in next block to be mined.
5. The winner miner in proof of work (PoW) consensus mechanism after solving complex mathematical problem, broadcast newly mined block to network.
6. The nodes which are participants in network, verify the block and propagate the block further, and confirmation starts to generate.
7. At the end of these activities, the approvals start to receive in the receiver's wallet and after almost six confirmations, the transaction is considered as finalized and confirmed. Though, six is just a suggested number; the transaction can be considered final even after the first confirmation. The reason behind waiting for six confirmations is to eliminate chances of double spending virtually after six confirmations.

Structure of Transaction

At high level, transaction comprises metadata, inputs, and outputs. Transactions are combined to create a block.

- **MetaData :** Metadata of the transaction includes the size of the transaction, the number of inputs and outputs, the hash of the transaction, and a lock time field. Every transaction has a prefix specifying the version number.
- **Inputs :** Usually, each input to the current transaction spends a previous output. Each output is an Unspent Transaction Output (UTXO) until an input consumes it.
- **Outputs :** Two fields in output includes instructions for the sending of bitcoins. The first field is about amount of Satoshis: the smallest denomination of bitcoin, equivalent to 100 millionth of a bitcoin, other second field is a locking script that includes the conditions that need to be fulfilled in order for the output to be expended.
- **Verification :** Verification is carried out using bitcoin's scripting language.

Other concepts related to transactions in bitcoin are described below.

Transaction fee

Miner charges the transaction fees according to the size of the transaction. Transaction fees are computed by subtracting the sum of the inputs and the sum of the outputs. These fees are considered as an incentive for miners to inspire them to add a user transaction in the block which miners are creating. All transactions finish in the memory pool, from where miners choose transactions on the basis of their priority to add them in the proposed block. In this case, transaction with a higher fee will be chosen preferably by the miners. Transaction fees are calculated with different criteria: for example, sending transactions, addition in blocks, and transmitting by nodes. Bitcoin protocol do not implement the rules for calculation of fees. This transaction fees are not compulsory. Even, transaction without fee will be processed in due course but may consume a longer time.

Contracts

Contracts are transactions that use the bitcoin system to execute a financial agreement. It permits users to design complex contracts that can be applied in many real-world scenarios. Contracts permit the development of a totally decentralized, independent, and reduced risk platform. Example of contracts are escrow, arbitration, and micropayment channels, and can be built using the bitcoin scripting language.

Transaction malleability

Transaction malleability in bitcoin was presented because of a bug in the bitcoin implementation. This bug allows an adversary to change the Transaction ID of a transaction, hence, it would look that a certain transaction has not been executed. This can permit the situations where double deposits or withdrawals can occur. In other words, this bug permits the altering of the unique ID of a bitcoin transaction before it is confirmed. If the ID of the transaction is altered before confirmation, it would seem that the transaction did not happen at all, which can then permit double deposits or withdrawal attacks.

Transaction pools

Transaction pools are also called as as memory pools. The nodes create these pools in local memory for maintaining a temporary list of transactions that are not yet confirmed in a block. Transactions are added in a block after passing verification and on the basis of their priority.

2.8 UTXO and Double Spending

- UTXO is an unspent transaction output which can be spent an input to a new transaction. A transaction in Bitcoin comprises of inputs and outputs. The input in Bitcoin terms an unspent transaction output. When you carry out a transaction, you will always spend an unspent transaction output (UTXO) as a total. It means, finished paying the entire amount. Then, you receive the remaining amount in another address called as change address. This change address is your personal address where you collect the change. This collected change is also an unspent transaction output (UTXO). For example, you received 1.0 BTC from someone. Now you want to send 0.5 BTC to another person. In this case, it is not possible to break the 1.0 BTC. Here, you will spend the complete 1.0 BTC in a transaction and receive the change in your change address.
- Consider you sends a 0.1 BTC to bitcoin user and you have 1.0 BTC with you. Actually, you send a 1.0 BTC as it is not possible to break. Now, amount is sent to two addresses. To first address 0.1 BTC and to other 0.89432145 BTC. The amount in inputs and outputs should be balanced. It means,
- 1.0 BTC - (0.1 BTC + 0.89432145 BTC) should be equal to zero. But here the difference we get is 0.00567855. This difference is the transaction fee that is paid to the miner. In this case, this fee is paid to miner as he is involved in validating the transaction, adding it into a block, mining the block, and broadcasting to the network. This is the mining fee for the work miner has carried out.
- In this transaction, two outputs are "unspent". In this manner bitcoin keeps track of balances. Finally, your balance is the sum of all unspent transaction outputs. Bitcoin network supports UTXO for account balance purpose. The Bitcoin addresses only can receive payments. A transaction in Bitcoin only contains a reference to a previous unspent transaction output. It does not hold any types of "from-address". Hence, the input of a Bitcoin address is in fact a previous unspent output. Bitcoin refers to a previous unspent transaction output using a combination of transaction ID/hash and an index. Once you spent an unspent transaction it is not possible to spend it again and hence avoid the double spending.

2.9 Bitcoin Blockchain

Following are some information details related to bitcoin.

Bitcoin as a currency

- In 2009, Santoshi Nakamoto implemented first application of blockchain called as bitcoin. Anybody can create the bitcoin using this application. Miners creates the bitcoins. In 2020, The value of bitcoin is not fixed and changes in every second. The smallest unit of bitcoin cryptocurrency is Satoshi. One BTC is equivalent to ten crore Satoshi. Hence,
  - 1 Satoshi = 0.00000001 BTC
  - 10 Satoshi = 0.00000010 BTC
  - 100 Satoshi = 0.00000100 BTC
  - 1,000 Satoshi = 0.00001000 BTC
  - 10,000 Satoshi = 0.00010000 BTC
  - 100,000 Satoshi = 0.00100000 BTC
  - 1,000,000 Satoshi = 0.01000000 BTC
  - 10,000,000 Satoshi = 0.10000000 BTC
  - 100,000,000 Satoshi = 1.00000000 BTC
- Security is supported in bitcoin and hacking bitcoin network is not easy. There is no regulatory body and any governance to control bitcoin currency. Following are some of the important terminologies.

Mining

In mining process, bitcoin blocks are created. Miners validate the new transactions and include them in the blockchain ledger. In bitcoin blockchain, miners compete to solve complex mathematical problem. The winner miner come with 256-bit hash that is less than or equal to target hash (given by software after verification of transaction). In this computation, huge computation power, energy and time is required. The miner who creates the winning hash first, gets rewarded with transaction fee or new bitcoins. This process of mining in bitcoin is called as Proof-of-Work (PoW) consensus mechanism.

Block Frequency

Speed of transaction processing is slow. It takes 10 minutes to create a new block.

Industrial Mining

Its not possible that, all the nodes are of equal power. Some nodes carry out industrial-sized mining. They connect to heavy set of nodes, use complex software and consume high energy.

Mining Pools

To nullify energy consumption and high time in the validation of transactions, some miners group together in mining pools to share the resources and services for efficiency.

Halving Policy

- Bitcoin halving is a key occurrence in the cryptocurrency's lifecycle that lessens the number of bitcoins in circulation by one half. It reduces the reward for mining bitcoin in half and reduces the inflation rate by the same amount. Halving is carried out roughly every four years according to planned schedule.

- Experts expect that this will continue until year 2140, when the whole supply of 21 million coins has been mined. Compared to reward of 50 BTC in 2009, nearly after four years (in 2012) the reward for mining dropped to just 25 BTC. In 2016, it was only 12.5 BTC per coin and in 2020, rewards dropped to 6.25 BTC.

Block in Bitcoin

- Block containing transactions has three important information associated with it. These are header of block, hash of previous block header and Merkle root.
  - **Block header :** Always previous block header's hash is stored as a reference in next block header.
  - **Hash of previous block :** Secured Hash Algorithm produces 256-bit hash.
  - **Merkle Root :** it contains the hash related to all the transactions in block. Merkle tree of all the transactions is stored in block. Due to Merkel root, Merkle tree can identify and remove duplicate or false transactions in a block. Merkle root ensures the fingerprint of all the transactions in the block.

Orphaned Block

It is valid block that are not the part of main chain. Such block is created if two miner succeeds to mine the block at the same time.

Timestamp

It is the 10-digit number denoting UNIX time noted since January 1, 1970. Since then, when after each second time changes, related data also changes.

Mempool

Each node has memory pool containing nearly 10,000 unvalidated (unconfirmed) transactions. Transaction continuously occurs in network. Whereas, blocks are added after every 10 minutes. When miner wins to add block in chain, transactions from the pool are added in block. After this, included transactions in block are removed from memory pool.

Block Propagation

- To propagate a newly announced and mined block, gossip protocol is used. Block becomes eligible to propagate if following conditions are fulfilled by node.
  - All the transactions within block are validated.
  - Nonce is valid.
  - Node validates the transactions.
- On failure of any one of the above conditions, block will be discarded by node.

SEGWIT

Signature and public key size occupy nearly 60% size of the entire transaction. Bitcoin protocol does not keep it within the block and distribute it separately. This separation is called as Segregated Witness (SegWit).

Nonce

This unique randomly generated number is added to identifier of transaction to avoid duplication of it.

Monetary policies

Block frequency and Halving are two monetary policies of bitcoin.

2.10 Consensus in Bitcoin

Bitcoin Mining

Miners validates the transactions using mining process. Nodes need to have high processing power and graphic cards for mining. It requires huge amount of energy for mining.

1. Types of Bitcoin Mining

Mining consists of hashing the block and introducing nonce to hash function and running the hash all over again. Miner will win if generated hash is less than or equal to the target hash. Hash is 64-digit hexadecimal number in which each digit can have number ranging from 0 to 9 and A to F. Hence,

Total possible hash set = $16^{64} = 1.16 \times 10^{77}$

There are two types of bitcoin mining. These are software mining and Hardware mining.

Software Mining

- Software mining is further categorized as pool mining and cloud mining. Pool mining is carried out by network of people from different places. They share bitcoin value on the basis of pre-specified agreed upon ratio. Cloud mining allows the miner to mine bitcoins over the Internet through mining contracts signed with cloud mining companies. There is no need to buy mining equipment. These are managed by cloud providers. Cloud mining is used by organizations where miner uses hardware setup and software that is provided by cloud providers. Cloud providers offers this service by charging some cost.

Hardware Mining

- Following are the four types of hardware mining
  - CPU Mining (Central Processing Unit)
  - GPU Mining (Central Processing Unit)
  - FPGA Mining (Field Programmable Gate Array)
  - ASIC Mining (Application Specific Integrated Circuit)
- CPU Mining, GPU mining and ASIC mining is already explained in section 2.6.2. The first implementation of Bitcoin mining was in Verilog which is a hardware design language used to program FPGAs. The general motivation behind FPGAs is to achieve performance close to the performance of custom hardware and permitting the owner of the card to customize it or reconfigure it "in the field" as well. On the contrary, custom hardware chips are designed in a factory to carry out the same work forever. FPGAs gives better performance with compare to graphics cards, mainly on "bit fiddling" operations which are insignificant to specify on an FPGA. Cooling is also easier with FPGAs and it is possible to use almost all of the transistors on the card for mining.
- Earlier, miners used to use CPU mining. CPU is capable to generate only less than 10 million hashes per second. Hence, a smaller number of bitcoins were created. GPU mining then replaced the CPU mining. It is 100 times faster than CPU and hence, a greater number of bitcoins were created with compare to CPU mining. Energy consumption of GPU is high with compare to CPU. FPGA is dedicated circuit board connected to USB port for mining purpose. Today many participants use ASIC machine for bitcoin mining. Mining speed is very high.

2. Proof of Work (PoW) and Hashcash in Bitcoin

- Proof of Work (PoW) consensus mechanism is discussed in chapter 1. In this mechanism, several miners compete to solve complex mathematical problem by using cryptographic hash algorithm. The solution then obtained is known as PoW. This restricts the attacker to carry out any malicious activity.
- Mining consists of hashing the block and introducing nonce to hash function and running the hash all over again. Miner will win if generated hash is less than or equal to the target hash. Mining node broadcast this PoW to other participants for verification in order to get consensus on the same.
- Network can verify it with low cost and very first miner gets rewarded with bitcoin block reward. In order to maliciously add block, miner's node should be 51% more strong than other nodes. To nullify energy consumption and high time in the validation of transactions, some miners group together in mining pools to share the resources and services for efficiency. In PoW process, nodes validate transactions in block, find nonce for hashcash and suggest a block.
- In second part of hashcash nodes find the nonce to add into block. This improves the security. The bitcoin block of transactions includes following three important types of information.
  - Block header
  - Hash of previous block header
  - Merkle Root

Block header

- In bitcoin block, block header size is 80 bytes which is divided in different fields as below.
  - Bitcoin version number (4 Byte)
  - Previous block hash (32 Byte)
  - Merkle root (32 Byte)
  - Time stamp (4 Byte)
  - Difficulty Target (4 Byte)
  - Nonce used by miners (4 Byte)

Hash Function used to create a hash of previous block header

- Hash function can take input data in any format such as words, text, PDFs, letters, audio files, video files, executable files etc. Secured Hash Algorithm (SHA-256) is core building block of blockchain. This hash function takes input with large arbitrary size and generates short fixed sized output of 256-bit as hash. Bitcoin uses SHA3-256 algorithm to generate hash. Following are the characteristics of hash function.
  - This hash function is one directional as it is not possible to generate data from output hash.
  - Hash function supports for fast computation.
  - Small changes in input generates the drastic change in output hash.
  - Hash produced for any given input produces the unique output. It means, duplicate hash for different inputs is not possible.
- In bitcoin blockchain, previous block header's hash is stored in next subsequent block as a reference to previous block.

Merkle Root

- The concept of Merkle root is already explained in section 1.4 of chapter 1. Due to Merkel root, Merkle tree can identify and remove duplicate or false transactions in a block. Merkle root in the header of the block is additionally protected by the hash of the block header stored in the next block.

3. Hashcash

- Hashcash is cryptographic hash-based algorithm introduced by Adam Back in 1997 which was initially used to identify email spammers. The target hash contains leading zeros. As per the mining difficulty, it is really hard to generate the hash with leading zeros. Hence, to generate the hash with leading zeros, a miner needs to add a nonce into the header of a block. As hashes are unpredictable, the miner has to continue process by trying different nonce to hash the whole block. Hence, it is not easy to generate hash with leading zeros which is less than target hash.
- As of 2019, it was needed to identify hash value with 18 leading zeros. More leading zeros increases complexity. In early days, few zeros were sufficient for valid block due to availability of less computing power and equipment. Over the period of time, more and more miners come onboard having powerful equipment. Without increasing difficulty, new blocks would be created too fast and the average block time would be far below 10 minutes.
- Bitcoin uses SHA-256 hash function to generate the hash. Bitcoin accomplishes proof-of-work using hash puzzles. To create a block, the node has to find nonce (a random number), such that when you concatenate the nonce, the previous hash, and the list of transactions that contain in that block and generate the hash of this whole string, then that generated hash output should be a number that falls into a target space that is quite small compared to the much larger output space of that hash function. This target space is any value falling below a certain target value. In this case, the nonce will have to fulfil the following inequality:

$$H(\text{nonce} \,||\, \text{prev_hash} \,||\, tx1 \,||\, tx\,2 \,||\, \ldots \,||\, txn) < \text{target}$$

- Where, nonce is a random number, prev_hash is previous block hash and tx1to txn are 1 to n transactions.

Timestamp

- It is the 10-digit number denoting UNIX time noted since January 1, 1970. Since then, when after each second time changes, related data also changes.

4. Block of Transactions

- Transaction data in block is used in hashing. Unconfirmed transactions first are added in Mempool (memory pool). Transactions happens continuously but block is added every 10 minutes. Once validated, transactions are added in block. Every transaction has ID and fee associated with it for miner. Miner gets this fee to add the transaction in block. Out of many transactions, miner always choose the transaction with higher fee to add in block.
- It is possible for miner to add 1MB of the transactions at a time which is equivalent to 2000 transactions. Practically, 1000 to 3500 transactions are added in block in one second. Sometime this figure can reach up to 7000 transactions. Consider 5 transactions gets selected to add in block on the basis of their higher fee. UNIX timestamp is with all the transactions. Nonce values range from zero to 4 billion.

- The miners change this nonce and combines with transaction data to create wining hash. If very first hash does not create the wining hash, then transaction with next highest fee is selected. If this transaction succeeds in generating wining hash, then it is added in block by removing it from memory pool.

Miner Pool working

- In case of miner pool, the allocation will be done by pool to different miners. Mempool contains 10,000 transactions and block contains 2000 transactions. If one member sends transaction to another member then the transaction is added in memory pool of receiver. After validation, transaction also gets duplicated in memory pool of other members (nodes). Once the validation overs and validated transaction are included in blockchain, they are removed from the memory pool.

5. Block Propagation and Relay

- Gossip protocol is used to propagate the announced block by miner. This protocol is just like the gossiping in our real life.

Nodes

- Node submit the transaction to other nodes and validate them from these nodes. Node also maintains the full block of the bitcoin network. They come with a nonce for the block of transactions and publish your block to network so that other nodes know it. It is to get acceptance of the block by other miners. All nodes are expected to participate in mining process and earn reward. Due to lack of resources, it is not possible for all the nodes to play role of miner. Nodes are of three types: Full node, Mining node, and SPV node.
  - **Full node :** Full nodes perform all the functions of nodes explained above. All the resources needed for mining are available with full nodes. Full node holds entire blockchain.
  - **Mining nodes :** Eligible for mining but do not hold entire blockchain.
  - **SPV nodes :** Today, Simplified payment verification (SPV) nodes are more in numbers but they do not involve in mining. They have to compulsorily validate the block using script.
- As of May 2019, total 82,640 nodes were in bitcoin network. Out of these, 9,384 were full nodes and 2,312 were SPV nodes. This figure of number of nodes is varying and full nodes varies in number ranging roughly from 9000 to 10,000. The newly created wining block by node is broadcasted to all other nodes in blockchain. Receiving nodes of the block validate the overall block, nonce and each transaction within the block. It is resource intensive task. Once validation is done, this block is further broadcasted to other nodes in blockchain. If validation fails, then block is discarded. It takes roughly 30 seconds to reach the validated block to all the nodes.

Orphaned Block

These blocks are not part of the main chain. These blocks are created when mining by miners happens simultaneously. Sometimes hackers may duplicate the transactions.

2.11 Proof-of-Burn (PoB)

- Proof-of-Burn (PoB) mechanism prevents from investing in expensive hardware equipment like PoW. This consensus mechanism first proposed by Iain Stewart in 2012. Validators burn coins by sending them to eater address from where they are irretrievable.

- These coins are now destroyed coins and called as burned coins. Those who burns more coins, their chances to mine the blocks are more. The validators in this mechanism follow the following approach:
  - In order to burn the coins, validators send them to eater address from where they are irretrievable.
  - Once, validators commit the coins to unreachable address, they earn a privilege to mine on the node based on a random selection process.
  - It shows a validator's long-term commitment in exchange although this is a short-term loss for them.
  - As per implementation of PoB, miners may burn the native currency of the Blockchain application or the currency of an alternative chain, like bitcoin.
  - The chances of selection for mining the next block increases if validators burn more coins.
- Although, PoB saves more resources, time, energy and computation power as an alternative to PoW, the protocol still wastes resources unnecessarily. In PoB, mining power simply goes to members willing to burn more money. The sent coins to eater addresses cannot be used again.
- Coins to be burned are sent to a public verifiable address from where it cannot be accessed. This address or account is publicly visible and therefore transparent for all participants of the network. Due to burning, availability of coin decreases which then increases the value of available coins. Destroying the coins shows a big commitment of the member to the currency by foregoing a narrow profit in exchange for a long-term profit. To avoid any excessive benefits for early adopters, the PoB has invented a method that permits for the periodic burning of crypto coins in order to maintain mining capacity. If the new block is mined, the energy of burned coins decreases a little. When quantity of the coin increases its value decreases. To guarantee that miners do not suffer losses from willingly burning their coins, many PoB-based systems gives a reward for each new block. In the long run, this reward at least counterbalances the initial investment.

Advantages

- PoB process is inexpensive in energy and hardware compared to other mechanisms. Proof of Work (PoW) are known to require elaborate computations.
- miners have to burn a share of their coins to create new blocks, the security of the system increases. The reason is, a moderately significant investment must be done prior to participant receives the appropriate authorization, like in the proof of work.
- Used for long-term commitment

Limitations

- In PoB, mining power simply goes to members willing to burn more money.
- It is a slower mechanism and takes more time to validate the block of transactions.

2.12 Proof-of-Stake (PoS)

- Proof-of-stake is a cryptocurrency consensus mechanism developed for Peercoin in 2012. In this algorithm, owners of the cryptocurrency can stake their coins, which assigns them the right to validate new blocks of transactions and add them to the blockchain. If more stake one has in validating node, then chances of corrupting the node reduces.

- Users with highest stake in cryptocurrency will always pay more attention towards maintaining and securing the network. The reason is, any security attack would be a threat to reputation and value of their cryptocurrency. The algorithm randomly selects validator depending on stake or collateral he or she has put forward.
- In PoS mechanism, miners are known as forgers or validators or delegates. Anyone staking cryptocurrency could be selected as a validator, the chances are very low if you stake relatively small amount. If node has 5% stake in network, then it can validate 5% of transactions. Therefore, most participants join staking pools. The owner of the pool selects the validator node, and a group of participants pool their cryptocurrency together for a better chance of winning new blocks. Rewards are also divided among the pool's participants. The pool owner may also charge a small fee.
- The energy consumption in this mechanism is less with compared to PoW mechanism. It is efficient mechanism with reduced threat of 51% attack. Peercoin, NXT and Blackcoin use PoS consensus mechanism. The biggest proof-of-stake blockchains by market capitalization in 2021 were Cardano, Avalanche, Polkadot and Solana. Ethereum uses hybrid PoW/PoS system.

Advantages

- It offers fast and inexpensive transaction processing.
- It is energy efficient mechanism.

Limitations of PoS Consensus Mechanism :

- More prone to attack as attacker only need to spend some cryptocurrency and no need to invest in other resources.
- More stake is proportional to more chances of winning. It indicates richest validator can control PoS consensus mechanism and earn more rewards frequently.

2.13 Proof of Elapsed Time (PoET)

- Proof of elapsed time (PoET) is a consensus algorithm is developed by Intel Corporation in 2016. This algorithm mainly used in permissioned blockchain networks to determine who creates the next block. This algorithm follows a fair lottery system which ensures equal chance for each participant to be a winner. PoET algorithm also improves transparency by confirming lottery results are verifiable by external participants. This is done by running a trusted code within a secure environment. PoET algorithm makes use of randomly generated elapsed time to decide mining rights and block winners on a blockchain network. This consensus algorithm prevents high resource utilization. It also allows miner's processor to sleep and switch to other tasks for specified time. Hence, its efficiency increases and high-power consumption is prevented. In this algorithm, miner's identity is known and he is not anonymous like in PoW.
- Each miner gets a randomized timer object from a trusted code which generates wait time. The main goal of this randomization mechanism is to avoid any attempt by miner to get the timer with shorter period. This miner who finishes the designated waiting time commits a new block to Blockchain. The related information is then broadcasted across the network. This procedure is again repeated to discover the new block. Hyperledger Sawtooth Architecture from Intel uses PoET consensus mechanism.

Limitations of PoET Consensus Mechanism :

- PoET is a permissioned and closed network. The mechanism highly depends on tools by Intel technology. This may lead to compatibility issues with other tools later.
- It is heavily dependent on use of trusted execution environment: Intel SGX-enabled CPU. The protocol is susceptible to security attack like "Foreshadow" which attacks secure enclave of SGX.

2.14 Life of a Miner

- ASIC miners are expensive machines. Hence, careful and regular maintenance is necessary to avoid spending a lot on replacements. Fortunately, there are some practices needs to be followed to get more benefits out of it. ASICs are the most effective and powerful mining hardware today. These machines are manufactured only for mining a specific crypto algorithm. As mining is an intensive and competitive process, miners make heavy use of them continuously. The average lifetime of a well-maintained machine can be around 3 to 5 years. However, if you operate the ASICs in harsh or poor conditions, they can weaken in few months. The careful handling of an ASIC miner can extend their lifespan for more than 5 years.
- Many miners are still using their Antminer S9s which is a model from 2016 and at a profit. These miners should be operated in well-ventilated location to avoid excess heating. ASICs operations should be avoided in environment with humidity that damages electronic circuits. In search of profit, crypto miners will run multiple cards on a single GPU motherboard, and exceed the power limit on individual cards. Mining cryptocurrency means running a GPU under full load for a continued time. It is the reason they do not last for longer period of time.

2.15 Mining Difficulty

Mining difficulty decides the exact difficulty to find a valid block. As of March 2015, the mining difficulty target (in hexadecimal) was :

0000000000000000172EC0000000000000000000000000000000000000000000

The Bitcoin difficulty algorithm offers the stability to the entire system by maintaining a 10-minute duration for finding new blocks. Nearly it requires 10 minutes for one miner out of the whole network to create a winning code and to propose a new block of bitcoin transactions to be included to the blockchain. Algorithm plays an important role to keep this frequency. It increases or decreases the difficulty of mining bitcoin. Whenever many miners or mining rigs participation increases in network at the same time, algorithm increases the difficulty of mining bitcoin. In reverse case of this, when there is drop in the number of miners competing to discover new blocks, it reduces the mining difficulty. In this case, algorithm/protocol works in favour of remaining miners to help them to discover blocks. The mining difficulty of the bitcoin network is changed by adding or reducing the leading zeros in target hash. If anyone among the competing miners succeeds to generate a wining hash having equal or higher number of leading zeros compared to target hash, is selected as the winner. In absence of such protocols, blocks would likely be generated faster and faster as more miners joined the network with more and more sophisticated equipment. This would be reason for new bitcoins in circulation at an unpredictable rate and would probably have the knock-on effect of hindering it's rise in value.

Determining the difficulty

- The mining difficulty is altered every 2016 blocks or roughly every 2 weeks. It is adjusted on the basis of miner's efficiency over the period of the previous 2016 blocks according to the following formula :

  next_difficulty = (previous_difficulty X 2016 X 10 minutes)/ (time to mine last 2016 blocks)

- When we multiply 2016 by 10 minutes (2016*10), it is exactly two weeks. If a block were created precisely every 10 minutes, then 2016 blocks would be mined in two weeks. This formula balances the difficulty to keep the property that blocks should be found by the network on average about once every ten minutes. If the period were much shorter, the difficulty might vary due to random variations in the number of blocks found in each period. If the period were much higher, the network's hash power might get too far out of balance with the difficulty. This difficulty is computed by each bitcoin miner independently. The block meeting the computed difficulty is then accepted. Any two miners mining on top of the same block will settle on what the difficulty should be. This permits consensus to be reached although miners with different branches compute different difficulty.

2.16 Mining Pool and its Methods

- It is not possible to have all nodes uniform size, power and resources. Hence, chances to get reward of the mining are not same. Some nodes carry out industrial-sized mining. They connect to heavy set of nodes, use complex software and consume high energy.
- These nodes themselves act as network. Hence, there are less chances of getting the equal opportunity to all the nodes to validate the transactions. Individual miners will find it difficult to get nonce that produces wining hash. Huge time and energy consumption is involved in validation of transaction.
- Hence, some miners group together into mining pools in order to combine their mining resources to save energy and improve efficiency. In other words, they work together in mine pooling. Mining pools offer the services that are shared by pool members to avoid the repetition of work and waste of time. The complex puzzle of creating the hash is distributed among each miner in pool.
- When anyone in the pool gets nonce that produces wining hash then mining pool is rewarded for that block. If pool itself gets 50% control of computing power of the network, then that pool will control the validation process. This is the against basic principle of decentralized distributed ledger. This is also known as 51% attack.

Mining Pool Methods

- Share is the key idea of the mining pool operation. Share may or may not be the potential block solution. Suppose a block solution is a number ending with 10 zeros and, a share may be a number ending with 5 zeros. During the process, there can be the shares that will not be ending with 5 zeros, but ending with 10 zeros.
- In this case, this will be a block solution. Mining pools uses shares to compute the miner's contribution to the work achieved by the pool to find a block. There are many miner reward systems: PPS, PROP, PPLNS, PPLNT, and many more.
- There can be hundreds or thousands of miners in mining pools that uses specialized protocols. In all these methods, let B is the block reward minus pool fee and P is a probability of finding a block in a share attempt. Let D is the block difficulty. A pool can support "variable share difficulty" feature. It means that, a miner can choose the share target (the lower bound of share difficulty) on their own and alter P accordingly.

  Probability P = 1/D

Following are the different mining pool methods.

1. Pay-per-Share (PPS)

Pay-per-share pools is a direct method where each miner gets shares for their contribution. However, these pools offer instant payouts irrespective of when the block is found. A miner contributing in this pool can exchange shares for a proportional payout at any time. Miner's payment is done from the pool's current balance and can withdraw their payout immediately. This model permits for the slightest possible change in payment for miners while also transferring much of the risk to the pool's operator. Each share costs precisely the probable value of each hash attempt.

R = B * P (Where * indicates multiplication)

2. Proportional

Proportional mining pools are the most common in which miners contributing to the pool's computation power receive shares up until the point at which the pool finds a block (the end of the mining round). After that, miners receive rewards proportional to the number of shares they hold. Let n is the amount of own shares N is amount of all the shares in this round. Then, reward R = B * n/N. It means, all shares are equal, but its value is computed only at the end of each round.

3. Pooled mining

Pooled mining is also known as bitcoin pooled mining (BPM). It is also called as "slush's system", as it was initially used on "slush's pool', which uses a scheme where older shares from the beginning of a block round are given less weightage than more recent shares. This is for reducing the chances of cheating by cheaters by switching between pools within a round. A new round starts the moment the pool finds a block and miners are rewarded **Proportional** to the shares submitted. This lessens the capability to cheat the mining pool system by switching pools during a round, to maximize profit.

4. Pay-per-last-N-shares

Pay-per-last-N-shares (PPLNS) method is similar to Proportional method. But in PPLNS, the miner's reward is computed based on the N last shares, rather than considering all shares for the last round. Hence, after finding the block, the reward of each miner is computed based on the miner contribution to the last N pool shares. As a result, if the round was short enough all miners receive more profit and vice versa.

5. Solo Mining Pool

Solo pools function similar to usual pools. In this pool, block reward is not distributed among all miners. The miner who finds the block gets the entire reward.

6. Peer-to-Peer Mining Pool (P2Pool)

In this Peer-to-peer mining pool (P2Pool), the pool server's responsibilities are distributed. This decentralization of the responsibilities removes the chance of the pool server being a single point of failure. It also prevents the chances of cheating from pool operator. Miners work on a share chain which is a side blockchain, Mining is carried out at a lower difficulty at a rate of one share block per 30 seconds. After the network target is reached by share block, it is transmitted and merged onto the blockchain. Miners are rewarded when this happens proportional to the shares submitted prior to the target block. A P2Pool needs the miners to run a full node, bearing the cost of hardware expenses and network bandwidth.

7. Geometric Method

Geometric Method (GM) was introduced by Meni Rosenfeld. In this method, the score is awarded for each new share, relatively to already existing score and the score of future shares, is always the same. It means that, there is no advantage even9 though mining is finished early or late in the round. In this method, fixed (f) and variable (c) fee is decided.

- At the start of every round the parameter s = 1.
- For every work k, S_k be the workers score for this round
- Set $S_k = 0$
- Ser r = 1 – P + P/c where P = 1/D
- Update r if difficulty changes during the round.
- When worker k submits share then set $S_k = S_k + s*P*B$ and the s = s*r
- If the share is a valid block, end the round. For every worker k, pay $((1-f)\ (r-1)\ S_k)/sP$

8. Double Geometric method

Double geometric method is a general version of Geometric and PPLNS methods. It contains new parameter: o ("cross-round leakage"). When o = Zero, this becomes the Geometric method. When o = 1, this becomes a variant of PPLNS, with exponential decay rather than a step function.

- Choose the parameters f, c and o
- When the pool first starts running, initialize s = 1.
- For every work k, S_k be the workers score and set $S_k = 0$
- Set r = 1+1/c * P * (1-c) * (1-o).
- If at any point the difficulty or the parameters change, r should be recalculated.
- When k submits a share, set $S_k = S_k + (1-f) * (1-c) * s * P * B$. In this case, B is the block reward at the time it was submitted and then s = s * r.
- If the share is a valid block, then also do the following for each worker k.
- Give the worker a payout of 1/cs * (1-o) * S_k and then set $S_k = S_k * o$

Review Questions

Q. 1 What is cryptocurrency? Explain.

Q. 2 Write short note on bitcoin.

Q. 3 What is forking? Explain different types of forks.

Q. 4 What are the different characteristics of cryptocurrencies? Explain.

Q. 5 What are the cryptocurrency wallets? Explain.

Q. 6 What are the different types of cryptocurrency wallets? Explain.

Q. 7 Differentiate between Hot wallets and cold wallets.

Q. 8 What is Altcoins? Explain in detail.

Q. 9 What is token? Explain different types of tokens.

Q. 10 Explain utility tokens in detail.

Q. 11 Explain security tokens in detail.

Q. 12 Differentiate between Altcoins, utility tokens and security tokens.

Q. 13 Explain in short different popular coins and tokens.

Q. 14 Write short note on Ethereum.

Q. 15 Write short note on Ripple.

Q. 16 Write short note on Bitcoin Cash.

Q. 17 Write short note on Litecoin.

Q. 18 Write short note on EOS.

Q. 19 Write short note on Monero.

Q. 21 Write short note on Stellar.

Q. 22 What are the different players in cryptocurrency ecosystem? Explain.

Q. 23 What is cryptomining? Explain.

Q. 24 What are the different types of cryptomining? Explain.

Q. 25 What is solo mining?

Q. 26 What is pool mining?

Q. 27 What is Airdrop? Explain its advantages and disadvantages.

Q. 28 What are the benefits of airdrop? Explain.

Q. 29 Explain different types of airdrops.

Q. 30 What is token or coin burning?

Q. 31 Why coin burning is necessary? Explain.

Q. 32 What are the reasons behind burning the coins?

Q. 33 Write short note on investing and trading.

Q. 34 Explain different cryptocurrency safety measures.

Q. 35 What are the different regulations around cryptocurrencies?

Q. 36 Explain block chain transaction life cycle.

Q. 37 Explain the structure of transaction in short.

Q. 38 What is transaction fee? Explain.

Q. 39 What is UTXO? Explain how it prevents double spending.

Q. 40 Explain in short bitcoin blockchain.

Q. 41 Explain bitcoin as cryptocurrency.

Q. 42 Explain software mining in bitcoin blockchain.

Q. 43 Explain hardware mining in bitcoin blockchain.

Q. 44 Explain Proof of Work (PoW) and Hashcash in Bitcoin.

Q. 45 What is the halving policy in bitcoin blockchain?

Q. 46 Explain bitcoin block header.

Q. 47 Explain Block Propagation and Relay in bitcoin blockchain.

Q. 48 Explain Proof-of-Burn (PoB) consensus algorithm with its advantages and disadvantages.

Q. 49 Explain Proof-of-Stake (Pos) consensus algorithm with its advantages and disadvantages.

Q. 50 Explain Proof-of-Elapsed Time (PoET) consensus algorithm with its advantages and disadvantages.

Q. 51 Write notes on life of miner.

Q. 52 What is mean by mining difficulty? Explain.

Q. 53 What is mining pool? Explain different types of mining pools.

❑❑❑

3 PROGRAMMING FOR BLOCKCHAIN

MODULE - 3

Syllabus

3.1 Introduction to Smart Contracts, Types of Smart Contracts, Structure of a Smart Contract, Smart Contract Approaches, Limitations of Smart Contracts.

3.2 Introduction to Programming: Solidity Programming – Basics, functions, Visibility and Activity Qualifiers, Address and Address Payable, Bytes and Enums, Arrays-Fixed and Dynamic Arrays, Special Arrays-Bytes and strings, Struct, Mapping, Inheritance, Error handling.

3.3 Case Study – Voting Contract App, Preparing for smart contract development.

3.1 Introduction to Smart Contracts

- A Smart Contract or cryptocontract is a code that automatically controls the transfer of digital assets between the different parties under certain conditions. The functioning of the smart contract is similar to the traditional contract while also automatically enforcing the contract. Smart contracts code performs exactly specified work in their implementation. Hence, smart contracts are enforceable by code. On the contrary, traditional contract is enforceable by law.
- In the year 1994, Nick Szabo, a legal scholar and a cryptographer, known the application of decentralized ledger for smart contracts. He realized that, these contracts could be programmed and code can be stored and replicated on the nodes and administered by the network of computers that constitute the blockchain. These smart contracts could also offer the transfer of digital assets between the parties under certain conditions.

1. Definition of Smart Contract

The smart contract is the software program implementing a set of rules or conditions and operates on the top of a blockchain. These set of rules are used by different parties to that smart contract to transfer the digital assets between them.

2. Smart Contract

- Based on set of rules implemented in smart contracts, different parties to the smart contracts agree to interact with each other. This agreement automatically gets executed after fulfilling the pre-defined rules. The smart contract code ease, validate and ensure the transaction or agreement. Hence, they are fully automated and it is possible to automate the records of real-life agreements related to purchase and sale of assets.
- A smart contract implements security coding of the blockchain and include details and permissions that require a precise sequence of events to happen to trigger the agreement of the terms stated in the smart contract. It can also comprise the time constraints that can enforce the deadlines in the contract. This contract is embedded in the blockchain and hence, it is transparent, immutable, inexpensive and decentralized.

- All the smart contracts are with their address in the blockchain. The contract address is used to interact with the contract by presuming that it has been broadcasted to all the participant nodes.
- The set of rules of the smart contract is coded at protocol level or application level. This ensures to reduce transaction cost related to the following.
  - Reaching the agreement as rules on which agreement is based are pre-defined.
  - New agreements among the people, institutions and about asset they hold can be formalized by implementing them in the form of software code.
  - Agreements are enforced automatically as per rule/conditions implemented are met.
- Today, there are so many organizations that are offering services to their consumers. These organizations are with centralized control. Smart contracts now provide the opportunities for innovation to be carried out in business where interactions are directly between the different business entities or end users of the business. There is no need to have third party involvement in this peer-to-peer interaction. There are several critical areas where smart contract can play the key role to ensure the better trust compared to its traditional counterpart related to predefined agreement.
- Following are the benefits of smart contracts.
  - Real-time updates are supported and they are simple and faster.
  - They eliminate the mediator and other centralized entities.
  - Cost effective by saving cost of paying to mediator.
  - As they execute automatically, there is no delay in producing expected and agreed outcome.

3. Smart Contract Example

- If John wants to purchase piece of lands from other Smith, then there are so many government agencies works as trusted third parties in this purchase deal. These agencies are land revenue office, government registrar's office, local authorities in combination with lawyer's office and land brokers etc. These third parties play role in verification and authentication of land document.
- If all these concerned authorities and government agencies are on the blockchain, then all the rules or conditions of valid sale could be implemented in smart contract. If the John want to buy land in 40000 USD using smart contract in blockchain, then the transaction would be validated/verified by all the members in blockchain. In this verification, nodes in network verifies whether the Smith is real owner of the land and all the government taxes are clear, encumbrance is valid and the John has a credentials to buy the land.
- If the blockchain network confirms that all the conditions are met, John automatically gets access code to the land document. Land gets transferred to his name and Smith's account is added with 40000 USD without involvement of any intermediatory.
- All the nodes in blockchain network, could check whether person is real owner of the land. They also check the seller's and buyer's identity and security records.

4. How Smart Contract Works

- All the smart contracts are with their address in the blockchain. The contract address is used to interact with the contract by presuming that it has been broadcasted to all the participant nodes. The smart contracts are implemented on blockchain where logic for validation is kept inside the block. Also, block contains the messages needed for smart contract.

- Participant in permissionless blockchain who want to participate in smart contract can download open-source software and publish its public key. Blockchain also generates private key for the participants address. If participant wants to carry out smart contract transaction on the related ledger, then software sends initiating message (encrypted with its private key) to other active participants.
- This message is signed by nodes having private key. Other participants with public key can verify the transaction and authenticate the message content. If more than 50% nodes in public blockchain reaches to consensus the message added in blockchain. Most of the time smart contracts has to access the external data or applications through web browser.

5. Characteristics of a Smart Contract

- Real-time tracking of performance and transactions is carried out by smart contracts. It also contributes in saving the overall cost. For external information, "information oracles" are used which provides external information to smart contracts through web browser interaction. Following are some of the important characteristics of the smart contracts.
  - **Independence :** Smart contracts can verify and execute itself and are tamper resistant. As it is developed by individual node, authorization is not dependent on an agent, notary or other third parties. Hence tampering by any intermediator not possible. They automate legal transactions that defines rules and conditions and also offers high security.
  - **Cost Effective :** There is no need to have trusted intermediatory in any sell or purchase deal and hence having lower transaction costs.
  - **Trustfulness :** As transactions are encoded and encrypted on a distributed replicated ledger, No one can announce that it has been lost.
  - **Recovery and Backup :** It is possible as same copies are available on multiple nodes.
  - **Protection :** Addresses are encrypted and hence contracts are well-preserved and prevented from any malicious activities.
  - **Agility:** All the business-related activities to be carried out are done/executed automatically.
  - **Truth based Accuracy:** There is no intervention of human and smart contracts gets executed automatically. Hence, operation is error free.

3.2 Types of Smart Contracts

Following are the categories of smart contracts using natural language for encoding.

- The contracts which are entirely in code.
- Contracts embedded in code using distinct version of the natural language.
- "Split" natural language contract with encoded performance.
- The contracts with encoded payment scheme. Encoding is in natural language. Here, the automation is offered.

Smart contracts can be used in many industries with several benefits. The different domains where they are used are : banking, health, finance, insurance, energy, e-governance, media, telecommunication, education, manufacturing and many other domains. The smart contracts can be classified into 4 types on the basis of applications.

1. Smart legal Contracts	2. Distributed Applications (DApps)
3. Decentralized Autonomous Organizations (DAO)	4. Smart Contracting Devices (SCD)

1. Smart Legal Contracts

- These are the most common type of smart contracts. They are with legal contract templates. They involve legal requirements such as mutual agreement which expresses a valid offer and acceptance, acceptable consideration, capacity and legality. These requirements are similar to their traditional counterpart.
- These are created to hold concerned parties responsible for fulfilling their end of an agreement. When set up correctly, a smart contract is lawfully enforceable and requires the parties to fulfil their duties. If parties fail to fulfill obligations in the contract then a legal action is triggered by the smart contract automatically against the party that is in breach.
- The laws and its contexts are different in different countries. Hence, they lack adequate support for smart and automated contracts on the blockchain and their legal status is unclear. However, once the laws are created, smart contracts can be created to ease processes which presently involve strict regulatory oversight such as transactions in the financial and real estate market, government subsidies, international trade etc.

2. Decentralized Applications (DApps)

- Smart contracts and DApps are closely related with each other. DApps interacts with the blockchain and controls the state of all network members. Smart contracts demonstrate the core logic of a decentralized app. DApps have unlimited number of members or participants in market.
- The user interface (UI) of DApps is similar to a typical website or mobile application. DApps front end represent the user's interaction, and its back end is implemented for all business logic. The business logic of DApps is depicted by one or more smart contracts communicating with the underlying technology of blockchain.

3. Decentralized Autonomous Organizations (DAO)

- DAO are the communities that exist on the blockchain. These communities can be defined by a set of stipulatory rules which are implemented via smart contracts. Every participant and their actions are bound by the community's rules with the task of imposing these rules. These rules are constituted from many smart contracts and collaborate to watch over activities in the community.
- The Genesis DAO was created by Ethereum members in May 2016. The main aim of community was for crowdfunding and as a venture capital platform. In an unexpectedly short time, they succeeded to raise an astonishing $150 million. However, hackers got some loopholes in the system and succeeded to steal about $50 million dollars' worth of Ethers from the crowdfund investors. The hack and its consequences resulted in a fork of the Ethereum blockchain into Ethereum and Ethereum Classic.
- So, DAO is the organization that is represented by logic implemented in transparent manner that is controlled by shareholder community without any central authority. DAOs are sophisticated form of smart contracts. There are some activities that needs manual intervention to execute certain tasks which cannot be executed automatically. These activities are :
  - **Tokens :** Tokens are used to reward certain activities which are funded directly upon creation of organizations.
  - **Autonomous :** The DAO code after deployment, cannot be controlled by its creator or any outsiders. Hence, it is transparent and incorruptible.

- **Agreement :** DAO's stakeholder should agree to withdraw or move funds. The number of stakeholders agree is already mentioned in DAO code.
- **Contractor :** To elect contractor for carrying out product and hardware building, and coding as well voting of token holders is necessary. This is because, the DAO cannot perform these tasks and contractor is appointed to achieve the gaols defined.
- **Proposals :** Used only for decision making in DAO. DAO need monetary deposit to avoid network overloading because of many proposals from many people. This discourages people from spamming the network.
- **Voting :** Once the proposal is submitted, voting takes place. DAO permits participants to exchange economic value with any person in global network without any intermediatory.

4. **Smart Contracting Devices (SCD)**

- Internet of things (IoT) in combination with the blockchain, talks about Application logic contacts (ALC) which contain an application-based code that remains in-step with other blockchain contracts. They allow communication across various devices and enable the merging of the Internet of Things (IoT) with blockchain technology.
- ALCs are an essential piece of multi-function smart contract and typically work under a managing program. Such smart contracts comprise application specific code that function in combination with other smart contracts and programs on the blockchain. They assist in communicating with and validating communication between IoT devices.

3.3 Structure of a Smart Contract

- In Ethereum blockchain, transactions are implemented via smart contract code, which is activated by the data and inputs from external account automatically. When external account triggers the smart contract, the business logic within the contract gets executed and status of ledger balances of all the accounts gets updated. This is also get updated on all the nodes in blockchain.
- The data gets written on all the nodes while writing but during reading, it is only in read only from the nearest or adjacent node. Hence, more efforts and resources are involved in writing and updating the information in Ethereum blockchain. To pay the reward for this work, Gas concept is used. It is a conversion unit of "ether" which is native currency of Ethereum blockchain.

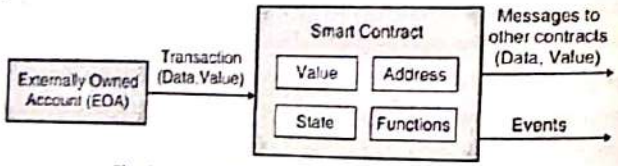


Fig. 3.3.1 : Basic Structure of Smart Contract

- A smart contract contains the value, address, functions, and state. The input to the smart contract is transaction. With this input it executes the corresponding code and triggers the output events. A smart contract is a software program that resides on a Blockchain and it has its unique address.

- Smart contract is identified by this unique address. A smart contract contains a set of executable functions and state variables. The execution of the function happens when transactions are made to these functions. The transactions contain input parameters which are needed by the functions in the contract. When function executes, the state variables in the contract changes depending on the logic implemented in the function.
- Contracts are written in various high-level languages like Solidity or Python. Language-specific compilers are used to compile the smart contract code into byte code. The compiled code of contracts is uploaded to the Blockchain network which assigns unique addresses to the contracts. Any participant or member on the Blockchain network can trigger the functions in the contract by sending transactions to the contract. The contract code runs on each node participating in the network as part of the verification of new blocks. Figure 3.1 shows the structure of a smart contract.

Structure of a Smart Contract in Ethereum

As stated above, a contract is like a class which contains state variables, functions, function modifiers, events, structures, and enums. The inheritance and polymorphism are also supported by contract. Following is a very basic smart contract structure overview written in Solidity language. The details about Solidity language is explained in section 3.6

```
pragma solidity ^0.8.2;
contract Customer {
  string id;
  function setId(string serial) public {
    id = serial;
  }
  function getId() public constant returns (string) {
    return id;
  }
}
```

Here pragma keyword, which is used to enable certain compiler features or checks; pragma indicates the version of the solidity compiler that should be used to compile the current file. This way, an incompatible change introduced by a future compiler version will be rejected. New versions of solidity are frequently released and some of them introduce incompatible changes.

If the compilation fails, you need to check the compiler version, you are using.

Next is contract keyword and name of the contract. The contract source code is placed between a pair of curly braces. The name of the contract will not be saved on the blockchain. At deployment, each contractor receives a unique address that will uniquely identify the contract.

Each contract can contain declaration of state variables that will be saved on blockchain, functions, events,function modifiers, structs, and enum types.

- **State Variables :** State variables are stored in contract storage. These can be set as constants. Expensive to use, they cost gas.

```
pragma solidity ^0.8.2;
contract myContract {
  uint myData; // State variable - unsigned integer type// ...
}
```

Function: Functions are like a few lines of code that can perform something within the contract.

```
contract Customer {
  string id = 34;

  function getId() public constant returns (string) { // function
    return id;
  }
}
```

State variables values are permanently stored in contract storage.

Function getID() is a getter function, as it creates a call that returns the value of the state variable.

➢ **Function Modifiers :** Function modifiers are similar to a function that checks the validation rules prior to executing the main function.

```
contract Purchase {
  address public seller;

  modifier onlySeller() { // Modifier
    require(msg.sender == seller);
    _;
  }
  function abort() public onlySeller { // Modifier usage
    // ...
  }
}
```

Here function modifier onlySeller() which changes the behavior of the function to which it is applied. In example, it is applies to abort().

➢ **Struct Types :** Struct is custom defined data types that consists of several variables declared in it.

```
pragma solidity ^0.8.2;
contract Node {
  struct caller { // Struct
    uint id;
    bool isEXists;
```



```
    address delegate;
    uint age;
  }
}
```

➢ **Events :** Events are functions that are bubbled up after compilation of function execution. If event code written there, it provide EVM logging functionality.

```
pragma solidity ^0.8.2;
contract Auction {
  event HighestBid(address user, uint amount); // Event

  function bid() public payable {
    // ...
    emit HighestBid(msg.sender, msg.value); // Triggering event
  }
}
```

➢ **Enum Types :** Enum is used to create custom types and it has fixed the set of values.

```
pragma solidity ^0.8.2;
contract shoping {
  enum KIT { cricket, hokey, table-tennis } // Enum
}
```

Structure of a Smart Contract in Ethereum

```
pragma solidity ^0.8.2;
import "browser/ERC20.sol";
contract MyFirstToken is ERC20 {
  string public constant symbol = "MFT";
  uint8 public constant decimals = 18;
  uint private constant _totalSupply = 1000;
  mapping (address => mapping (address => uint)) private _allowances;
  function MyFirstToken() {
    _balanceOf[msg.sender] = _totalSupply;
  }

  function totalSupply() constant returns (uint _totalSupply) {
    _totalSupply = _totalSupply;
```

```
}

function balanceOf(address _addr) constant returns (uint balance) {
  return _balanceOf[_addr];
}

function transfer(address _to, uint _value) returns (bool success) {
  if (_value > 0 && _value <= balanceOf(msg.sender)) {
    _balanceOf[msg.sender] -= _value;
    _balanceOf[_to] += _value;
    return true;
  }
  return false;
}

function transferFrom(address _from, address _to, uint _value) returns (bool success) {
  if (_allowances[_from][msg.sender] > 0 &&
    _value > 0 &&
    _allowances[_from][msg.sender] >= _value &&
    _balanceOf[_from] >= _value) {
    _balanceOf[_from] -= _value;
    _balanceOf[_to] += _value;
    // Missed from the video
    _allowances[_from][msg.sender] -= _value;
    return true;
}
  return false;
}

function approve(address _spender, uint _value) returns (bool success) {
  _allowances[msg.sender][_spender] = _value;
  return true;
}
```


3.4 Types of Oracles

Oracle is an agent that verifies off-chain occurrences and submit facts to a blockchain. For example, trustee agree to release fund if certain conditions are fulfilled. Then, to release any fund, oracle has to sign smart contract as well. Following are the different types of oracles.

- **Software Oracles :** These oracles deal with information that is available over the internet. For example, Temperature in city, delay in train arrival etc. Such data originates from online sources.
- **Hardware Oracles :** In this case, some smart contract directly receive input from physical devices such as sensors. It can be used for example, to track details of car passing particular cross junction or to track movement of trucks carrying the goods.
- **Inbound and Outbound Oracles :** Here, inbound oracles provide data from external world to smart contract and outbound oracles permits the smart contract to send data to outside world.
- **Consensus-based Oracles :** Prediction markets use more than one oracle to get consensus about future outcomes. Majority of oracles will determine the outcome of the event.

3.5 Benefits and Limitations of Smart Contracts

Following are the benefits and limitations of the smart contracts.

Benefits of Smart Contract

Organizations can make use of smart contracts to automate the contract lifecycle, improve collaboration, and digitize their assets or transactions.

1. **Security :** Smart contracts support data encryption and its tampering is not possible. Hence, they are highly secure alternative to paper-based contracts. Smart contracts can be used in financial institutions and government offices due to high security.
2. **Speed :** Because of the automation of contract lifecycle process and the self-enforcement capability, Smart contracts speed the process significantly. Automation of the contract lifecycle management (CLM) process makes it less prone to human error.
3. **Fewer Intermediaries :** There's a reduced dependence on intermedianies for trust services.
4. **Transparency :** Smart contracts guarantee higher transparency and hence a lower execution risk. Any alteration to the contract needs the mutual consent of the involved parties. There is no scope of manipulation or non-performance due to any individual.
5. **Cost-Effective :** Smart contracts are very cost-effective because of reduced dependence on intermedianies and ease of storage.
6. **Digital backup :** As there are multiple times copying of the documents in blockchain, it ensures a secure digital backup and easy accessibility to the contract in the event of a loss.
7. **Trust :** There is no any third-party involvement. The data record is also distributed across the entire chain. Hence, changing the information for personal profit is challenging. This ensures trust among all members of the contract.
8. **Clear Communication :** Smart contracts can drastically cut down on efficiency lost to gaps in communication.

Limitations of Smart Contracts

While there are many reasons to adopt smart contracts, there are a few reasons to not. Following are the few of the challenges and limitations of smart contracts.

1. **Flexibility :** If we consider the law firms, the logic-based execution is often challenging for them. Contract creation has a subjective element in it and has deliberate usage of terms like 'good faith' or 'reasonable' to have room for flexibility. This type of contract is obligatory when the parties want to create a relational contract rather than a transactional one. In this situation, smart contracts fail to offer utility.
2. **Legal Difficulties :** Smart contracts reduce dependence on intermediaries and lawyers, it's imperative for involved parties to be aware of the legal consequences of public, private, criminal, and commercial law.
3. **Delayed Transactions :** There are cases where technology proves to be a nuisance rather than a benefit. The congestion in a blockchain network can lead to a delayed transaction and finally push transactional cost above the cost incurred in traditional contracts.
4. **Inflexibility :** Practically, smart contracts are a fully self-operating mechanism that works without human intervention after being deployed on a blockchain. However, the most of the commercial relationships are very complex in nature. Accounting for all likely contingencies and nuances in a contract is not possible both in written and digital agreements. Conventionally, parties believe on judicial assistance to help them resolve contractual issues that are often imposed by unexpected circumstances. While this problem can't be totally eased regardless of the type of contract, smart contracts can automatically reference an external arbitrager when unanticipated factors come into play.
5. **Link to the real world :** Many contractual agreements need some sort of a connection to the real world, which hinders the promise of smart contracts to be fully autonomous.
6. **Speed and scalability :** Smart contracts are deployed on blockchain platforms such as Ethereum. Hence transaction processing speed depends on the blockchain. For example, Ethereum processes 13 transactions per second. This severely limits the number of transactions that the network can process in any given second.
7. **Immutable :** AS smart Contracts are tamper-proof, it can be difficult or impossible to fix the problems that occur.
8. **Technical challenges :** To carry out the audit of smart contract, technical proficiency is necessary which falls short in common investors and end-users. This lack of knowledge leads to foolish investment, which might lead to insecure smart contracts, permitting hackers to steal millions of dollars.
9. **Legal enforcement :** There are no any intermediatory involved in smart contracts. If a contract works as per agreement, then the users do not have to go to court for settlement. In case of disagreements, these contracts are not legally enforceable in all countries.

3.6 Solidity Programming

3.6.1 Basics

- Solidity is a statically-typed curly-braces programming language used for developing smart contracts that run on the Ethereum Virtual Machine (EVM) which is the runtime environment for smart contracts in Ethereum. Solidity is vastly influenced by C++, Python and JavaScript and has been designed to target the Ethereum Virtual Machine (EVM). Solidity supports inheritance and libraries.

- Solidity language can be used to create contracts for voting, crowdfunding, blind auctions, and multi-signature wallets. Smart contract programs execute inside a peer-to-peer network where execution is not controlled by any authority, and thus they permit to implement tokens of value, ownership, voting and other kinds of logics. For deploying the contracts, the latest released version of Solidity is the requirement.
- The reason is, breaking changes along with the new features and bug fixes are introduced regularly. Presently, 0.x version numbers are being used to specify the fast pace of change. Solidity was publicly introduced for the first time in November 2014 at Devcon0. Solidity v0.1.0 turned 5 years old on July 8th 2020.
- The Solidity programming language is an open-source, community project governed by a core team that is sponsored by the Ethereum Foundation. The Ethereum blockchain platform runs smart contracts which are the applications that run precisely as they are programmed.
- The key "product" of the Solidity project is the Solidity compiler (solc). Solidity compilers convert solidity programs to EVM bytecode. The project also manages the important application binary interface (ABI) standard for Ethereum smart contracts.
- Solidity is case sensitive.
- Every statement must end with semicolon.
- Most of the control structures are available : if, else, for, while, continue, break, return.
- // represents a single line comment.
- /* ---- */ represents a multi-line comment.

3.6.2 Data Types

Following are the basic data types in Solidity.

Boolean (bool)

Boolean value is either true or false. Following are the logical operators used.

Operator	Logical Meaning
!	not
&&	and
||	or
==	equal
!=	not equal

Signed (int) and unsigned (uint) integers

These integers declared in increments of 8 bits from int8 to uint256. Without a size suffix, 256-bit quantities are used, to match the word size of the EVM.

Fixed point (fixed, ufixed)

Fixed-point numbers, declared with (u)fixedMxN where M is the size in bits (increments of 8 up to 256) and N is the number of decimals after the point (up to 18); e.g., ufixed32x2.

Address

Address contains the 20-byte value that represents the size of an Ethereum address. An address object has member functions. For example, to get the balance, balance method and to transfer balance to another address transfer method is used.

Byte array (fixed)

Fixed-size arrays of bytes, declared with bytes1 up to bytes32.

Byte array (dynamic)

Variable-sized arrays of bytes, declared with bytes or string.

Enum

User-defined type for enumerating discrete values: enum NAME {LABEL1, LABEL 2, ...}.

Arrays

An array of any type, either fixed or dynamic: uint32[][5] is a fixed-size array of five dynamic arrays of unsigned integers.

Struct

User-defined data containers for grouping variables: struct NAME {TYPE1 VARIABLE1; TYPE2 VARIABLE2; ...}.

Mapping

Hash lookup tables for key ⇒ value pairs: mapping(KEY_TYPE ⇒ VALUE_TYPE) NAME.

In addition to these data types, Solidity also offers a variety of value literals that can be used to calculate different units:

Time units

The units seconds, minutes, hours, and days can be used as suffixes, converting to multiples of the base unit seconds.

Ether units

The units wei, finney, szabo, and ether can be used as suffixes, converting to multiples of the base unit wei.

3.6.3 Variables

Solidity supports three types of variables.

- **State Variables :** These variable's values are permanently stored in a contract storage.
- **Local Variables :** Variables whose values exist till function is executing.
- **Global Variables :** Special variables present in the global namespace and used to receive information about the blockchain.

The state or local variable type needs to be specified during declaration. The declared variable always has a default value based on its type. There is no concept of "undefined" or "null". Remix IDE is used to compile and run the solidity program. Consider the following solidity program. The first line is a pragma directive which tells that the source code is written for Solidity version 0.5.0


```
pragma solidity ^0.8.2;
contract SolidityTest {
 constructor() public{
 }
 function getResult() public view returns(uint){
  uint a = 1;
  uint b = 2;
  uint result = a + b;
  return result;
 }
}
```

The output of the above code is 0: uint256: 3

The comments are written with // symbol. We can also write multiline comments. For example.

```
// This is a comment. (Single line)
/* This is a comment written in more than one line
   This a comment written in more than one line
*/
```

State Variable

Following is the example of state variables.

```
pragma solidity ^0.8.2;
contract SolidityTest {
 uint storedData;    // State variable
 constructor() public {
  storedData = 10;  // Using State variable
 }
}
```

Local Variable

Following is the example of local variable.

```
pragma solidity ^0.8.2;
contract SolidityTest {
 uint storedData; // State variable
 constructor() public {
  storedData = 10;
 }
```

```
function getResult() public view returns(uint){
  uint a = 1; // local variable
  uint b = 2;
  uint result = a + b;
  return result; //access the local variable
  return storedData; //access the state variable
 }
}
```

Global Variables

These are special variables which present in global workspace and offer information about the blockchain and transaction properties. When a contract is executed in the EVM, it has access to a small set of global objects which includes block, msg, and tx objects.

message call context

- **msg.sender(address payable) :** Sender of the message. It represents the address that initiated this contract call, not necessarily the originating EOA that sent the transaction. If our contract was called directly by an EOA transaction, then this is the address that signed the transaction, but otherwise it will be a contract address.
- **msg.value :** Number of wei sent with this call.
- **msg.gas :** The amount of gas left in the gas supply of this execution environment. This was deprecated in Solidity v0.4.21 and replaced by the gasleft function.
- **msg.data :** The data payload of this call into our contract.
- **msg.sig(bytes4) :** The first four bytes of the data payload, which is the function selector.

Transaction context

The transaction-related information is accessed through tx object.

- **tx.gasprice (uint) :** Gas price of the transaction
- **tx.origin (address payable) :** Sender of the transaction

Block context

The block object contains information about the current block:

- **block.blockhash(blockNumber) :** The block hash of the specified block number, up to 256 blocks in the past.
- **block.coinbase :** Current block miner's address.
- **block.difficulty :** The difficulty (PoW) of the current block.
- **block.gaslimit :** Current block gaslimit.
- **block.number :** The current block number (blockchain height).
- **block.timestamp :** The timestamp placed in the current block. (Number of seconds since the Unix epoch).

Solidity Variable Names

While naming your variables in Solidity, keep the following rules in mind.

- Solidity reserved keywords cannot be used as a variable name. For example, break or boolean variable names are not valid.
- Variable names should not start with a numeral (0-9). They should begin with a letter or an underscore character. For example, _123test is a valid and 123test is an invalid variable name.
- Solidity variable names are case-sensitive.

Address Object

Any address, either passed as an input or cast from a contract object, has a number of attributes and methods:

- **address.balance :** It denotes the balance of the address in wei. For example, the current contract balance is address(this).balance.
- **address.transfer(amount) :** Transfers the amount (in wei) to this address. It throws an exception on any error.
- **address.send(amount) :** It is similar to transfer. It returns false on error and does not throw any exception. Always check the return value of send.
- **address.call(payload) :** This is a low-level CALL function—can construct an arbitrary message call with a data payload. Returns false on error.
- WARNING: unsafe—recipient can accidentally or maliciously use up all your gas, causing your contract to halt with an OOG exception; always check the return value of call.
- **address.callcode(payload) :**Low-level CALLCODE function, like address(this).call(...) but with this contract's code replaced with that of address. Returns false on error.
- WARNING: advanced use only!
- **address.delegatecall()** Low-level DELEGATECALL function, like callcode(...) but with the full msg context seen by the current contract. Returns false on error. WARNING: advanced use only!

3.6.4 Variable Scope

Scope of local variables is remains within the function in which they are defined but State variables can have three types of scopes.

- **Public :** Public state variables can be accessed internally as well as via messages. For a public state variable, an automatic getter function is generated.
- **Internal :** Internal state variables can be accessed only internally from the current contract or contract deriving from it without using this.
- **Private :** Private state variables can be accessed only internally from the current contract they are defined not in the derived contract from it.

Example

```
pragma solidity ^0.8.2;
contract C {
```

```
  uint public data = 30;
  uint internal iData= 10;

  function x() public returns (uint) {
    data = 3; // internal access
    return data;
  }
}
contract Caller {
  C c = new C();
  function f() public view returns (uint) {
    return c.data(); //external access
  }
}
contract D is C {
  function y() public returns (uint) {
    iData = 3; // internal access
    return iData;
  }
  function getResult() public view returns(uint){
    uint a = 1; // local variable
    uint b = 2;
    uint result = a + b;
    return storedData; //access the state variable
  }
}
```

Arithmetic Operators

Following are the arithmetic operators used.

Operator	Meaning
+	Addition
-	Subtraction
*	Multiplication
/	Division

Operator	Meaning
%	Modulus
++	Increment
--	Decrement

Comparison Operators

Following are the comparison operators used.

Operator	Meaning
>	Greater than
<	Less than
>=	Greater than and Equal to
<=	Less than and Equal to

Bitwise Operators

Operator	Meaning
&	Boolean AND
|	Boolean OR
^	Boolean exclusive OR
~	Bitwise NOT
<<	Left Shift
>>	Right Shift
>>>	Right shift with Zero

Assignment Operators

Operator	Meaning
=	Simple Assignment
+=	Add and Assignment
- =	Subtract and Assignment
*=	Multiply and Assignment
/=	Divide and Assignment
%=	Modules and Assignment

String

String literal using are used with both double quote (") and single quote ('). It offers string as a data type to declare a variable of type String.

```
pragma solidity ^0.8.2;
contract SolidityTest {
  string data = "test";
}
```

In above code, "test" is a string literal and data is a string variable. If byte types are used instead of String as string operation, then less gas is required compared to string operation. Solidity supports inbuilt conversion between bytes to string and vice versa. It is easily possible to assign String literal to a byte32 type variable. Solidity considers it as a byte32 literal.

```
pragma solidity ^0.8.2;
contract SolidityTest {
  bytes32 data = "test";
}
```

3.6.5 Arrays : Fixed and Dynamic Arrays

Array data structure stores a fixed-size sequential collection of elements of the same type. It is more useful to think array as a collection of same type variables. There is no need to declare individual variables, for example, number0, number1, ..., and number99. Instead of this, it is sufficient to declare one variable such as numbers and use numbers[0], numbers[1], and ..., numbers[99] to represent individual variables. A particular element in an array is directly accessed by an index. In Solidity, two types of arrays are supported. One is compile-time fixed size and another is of dynamic size. In case of storage array, different types of elements can be stored. For memory array, element type cannot be mapping and in case it is to be used as function parameter then element type should be an ABI type. All arrays comprise contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element of an array.

Fixed Size Array

In Solidity, to declare fixed size array, programmer has to specify the type of the elements and the number of elements in an array.

```
type arrayName [ arraySize ];
```

This is a single-dimension array. The arraySize must be an integer constant greater than zero and type is any valid Solidity data type. For example, to declare a 20-element array called account of type uint, use this statement –

```
uint account[20];
```

Following program demonstrate the use of fixed size array.

```
// Solidity program to demonstrate creating a fixed-size array
pragma solidity ^0.8.2;
// Creating a contract
```



```
contract Types {
  // Declaring state variables of type array
  uint[6] data1;
  // Defining function to add values to an array
  function array_example() public returns (
  int[5] memory, uint[6] memory){
    int[5] memory data = [int(50), -63, 77, -28, 90],
    data1 = [uint(10), 20, 30, 40, 50, 60];
    return (data, data1);
}
```

Output:

```
"0": " int256[5] : 50, -63, 77, -28, 90",
"1": "uint256[6] : 10, 20, 30, 40, 50, 60"
```

Dynamic Arrays

- In Solidity, to declare an array of dynamic size, the programmer has to specify the type of the elements as follows.

  type[] arrayName;

- Dynamic memory arrays are created using new keyword.

  uint size = 3;
  uint account[] = new uint[](size);

- Following single statement initializes Solidity array elements.

  uint account[3] = [10, 20, 30];

- Following is an example to assign a single element of the array :

  account[2] = 5;

- The above statement assigns element number 3rd in the array a value of 5.

  uint salary = account[2];

- The above statement assigns the value of third element in array to salary variable.

Example:

```
// Solidity program to demonstrate creating a dynamic array
pragma solidity ^0.8.2;
// Creating a contract
contract Types {
  // Declaring state variable of type array. One is fixed-size and the other is dynamic array
  uint[] data
  = [10, 20, 30, 40, 50];
  int[] data1;
```

```
// Defining function to assign values to dynamic array
function dynamic_array() public returns(
 uint[] memory, int[] memory){
  data1 = [int(-60), 70, -80, 90, -100, -120, 140];
  return (data, data1);
 }
}
```

Output :

"0": " int256[] : 10, 20, 30, 40, 50",

"1": "int256[] : -60, 70, -80, 90, -100, -120, 140"

Members

- **Length :** It returns the size of the array. It can be can be also used to change the size of dynamic array.
- **Push :** push is used to append an element to a dynamic storage array at the end. It returns the new length of the array.
- Declaration, assignment and accessing the arrays is demonstrated in following example.

Example

```
pragma solidity ^0.8.2;
contract test {
 function testArray() public pure{
  uint len = 7;
  //dynamic array
  uint[] memory a = new uint[](7);
  //bytes is same as byte[]
  bytes memory b = new bytes(len);
  assert(a.length == 7);
  assert(b.length == len);
  //access array variable
  a[6] = 8;
//test array variable
  assert(a[6] == 8);
  //static array
  uint[3] memory c = [uint(1) , 2, 3];
  assert(c.length == 3);
 }
}
```


3.6.6 Special Arrays: Bytes and Strings

Byte Arrays

In Solidity language, byte array is a dynamic array and hold any number of bytes. It is different from the bytes value type, which takes 32B for each variable. Following are the ways to use the byte array.

- It allows to declare a state element with initial length size
- It is possible divide into code lines that are similar to the arrays.
- Can assign values directly and have values pushed into it.
- Offer a read/write length property

Example :

```
pragma solidity ^0.8.2;
// code illustration of byte array in solidity
Contract SampleBytesArray  {
// as a state variable
 bytes simBytes = riBytes (2)
// as divided code lines
 bytes simBytes ;
   simBytes = riBytes (8)
// with directly assigned values
   simBytes = " Lorem Modi ";
// with values pushed into it
 simBytes. push ( byte(10) )
// as read/write length property
 simBytes. length = 5; // setting bytes length to 5 bytes
```

String arrays

String arrays are also dynamic arrays similar to byte arrays. The string does not have an index and hence, array members such as length, push, and pop does not present. If needed then we must convert string variables to bytes first, and then convert back to strings after the operation. Strings are composed of characters within a single or double quote. In following example, strings are declared and the values are assigned directly :

```
pragma solidity ^0.8.2;
// code illustration of a string array in solidity
Contract SampleString
 String name = "Solidity " ;
// convert back to bytes
 Bytes ByteName = bytes ( name ) ;
```

3.6.7 Enums

Enums allows for variable to have one of only a few predefined values. Enums are values in this enumerated list. Enums helps to reduce the number of bugs in your program. Consider the example of a fresh juice shop. It would be possible to limit the glass size to small, medium, and large. This would ensure to order only size that is small, medium, or large.

```
pragma solidity ^0.8.2;
contract test {
  enum FreshJuiceSize{ SMALL, MEDIUM, LARGE }
  FreshJuiceSize choice;
  FreshJuiceSize constant defaultChoice = FreshJuiceSize.MEDIUM;
  function setLarge() public {
    choice = FreshJuiceSize.LARGE;
  }
  function getChoice() public view returns (FreshJuiceSize) {
    return choice;
  }
  function getDefaultChoice() public pure returns (uint) {
    return uint(defaultChoice);
  }
}
```

3.6.8 Structs

Struct types represents a record. Following are the attributes to keep track of your books in a library.

- Book-Title
- Author
- Subject
- Book ID

The struct keyword is used to define structure. The struct keyword defines a new data type, with more than one member. Following is the syntax of struct statement.

```
struct struct_name {
  type1 type_name_1;
  type2 type_name_2;
  type3 type_name_3;
}
```


Example

```
struct Book {
  string title;
  string author;
  uint book_Id;
}
```

The member access operator (.) is used to access any member of the structure. The member access operator is written as a period between the structure variable name and the structure member that we wish to access. Following program demonstrate the use of struct.

```
pragma solidity ^0.5.0;
contract test {
  struct Book {
    string title;
    string author;
    uint book_Id;
  }
  Book book;
  function setBook() public {
    book = Book('Learn Java', 'TP', 1);
  }
  function getBookId() public view returns (uint) {
    return book.book_id;
  }
}
```

First Click setBook Button to set the value as LARGE then click getBookId to get the selected book id.

Output :

```
5uint256: 1
```

3.6.9 Mapping

Mapping in Solidity are used to store the data in the form of key-value pairs. A key is any of the built-in data types but reference types are not allowed while the value can be of any type. Mappings are typically used to associate the unique Ethereum address with the associated value type. Following is the syntax to declare a mapping type. All keys must have the same type and all values must have the same type. Mapping is always stored in storage, it's a state variable. The mapping advantage is that lookup time is constant no matter of size.

```
mapping(_KeyType => _ValueType)
```

```
//Function adding values to the mapping
function adding_values() public {
  var student
   = result[0xDEE7796E89C82C36BAdd1375076f39D69FafE252];
  student.name = "John";
  student.subject = "Chemistry";
  student.marks = 88;
  student_result.push(
    0xDEE7796E89C82C36BAdd1375076f39D69FafE252) -1;
}
// Function to retrieve values from the mapping
function get_student_result(
) view public returns (address[]) {
  return student_result;
}
// Function to count number of values in a mapping
function count_students(
) view public returns (uint) {
  return student_result.length;
}
}
```

3.6.10 Functions

Solidity supports all the important and necessary features to write modular code using functions. First, we have to define and then use the function In Solidity, function is defined by using the function keyword, followed by a unique function name, a list of parameters (that might be empty), and a statement block surrounded by curly braces. The function defined in the contract can be called by an EOA transaction or another contract. Following is the basic syntax.

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

In following example, function getResult is used and that takes no parameters.

Example :

```
pragma solidity ^0.8.2;
contract Test {
  function getResult() public view returns(uint){
    uint a = 1; // local variable
```



```
    uint b = 2;
    uint result = a + b;
    return result;
  }
}
```

3.6.11 Visibility and Activity Qualifiers

The following keywords specify the function visibility. Visibility to a function specifies how the function is visible to the caller either from internal or external.

- **Public** : The public is the default and these public functions are can be called by other contracts or EOA transactions, or from within the contract. They are visible from internal and external contracts. It means, they can be called inside a contract or outside contract. As this function is visible to public, security concern is there.
- **Private :** private functions are visible inside the same contract are like internal functions. They can not be called by derived contract.
- **External :** External functions are similar to public functions. These functions cannot be called from within the contract unless explicitly prefixed with the keyword this. External functions are visible to public functions and can be called outside contract only.
- **Internal :** Internal functions are similar to private functions and only accessible from within the contract. These functions cannot be called by another contract or EOA transaction. Internal functions are visible to contract hierarchy. It means, they can be called inside a contract and Contracts that inherit from it.

Function qualifiers

Modifiers or qualifiers permits to define the functions with behavior for changing the state of contract data.

1. **Constant or view :** If the view is used then it does not allow to modify any state. The term constant is an alias for view that will be deprecated in a future release. View is expected to become an enforced keyword in v0.5 of Solidity. The view functions display the data and do not modify the state variable If following statements are present in the function then it is considered modifying the state and compiler will throw warning in such cases. Getter method are by default view functions.
   - Modifying state variables.
   - Emitting events.
   - Creating other contracts.
   - Using selfdestruct.
   - Sending Ether via calls.
   - Calling any function which is not marked view or pure.
   - Using low-level calls.
   - Using inline assembly containing certain opcodes.

Example

```
pragma solidity ^0.8.2;
contract Test {
```

```
function getResult() public view returns(uint product, uint sum){
  uint a = 1; // local variable
  uint b = 2;
  product = a * b;
  sum = a + b;
 }
}
```

Output

0: uint256: product 2

1: uint256: sum 3

2. **pure :** pure functions do not access data from blockchain and also not permit read or write operations. They do not allow access to state data or transaction data. They can only work on arguments and return data, without reference to any stored data. Following statements existing in the function are considered reading the state and compiler will throw warning.
   - That reads state variables.
   - That accesses address(this).balance or <address>.balance.
   - Those accesses any of the special variable of block, tx, msg (msg.sig and msg.data can be read).
   - That calls any function not marked pure.
   - Using inline assembly that contains certain opcodes.

   Pure functions can use the revert() and require() functions to revert possible state changes if an error occurs.

Example :

```
pragma solidity ^0.8.2;
contract Test {
 function getResult() public pure returns(uint product, uint sum){
   uint a = 1;
   uint b = 2;
   product = a * b;
   sum = a + b;
  }
}
```

Output

0: uint256: product 2

1: uint256: sum 3

3. **Payable :** A payable function accepts the incoming payments from caller. Those function which are not declared as payable rejects incoming payments. Hence, payable function can accept Ether sent to the contract. if it is not specified, the function automatically rejects all Ether sent to it.

Example:

```
pragma solidity ^0.8.2;
contract Sample {
  uint amount =0;
  function payme() payable{
    amount += msg.value;
  }
}
```

Above example shows that, payme function is marked with payable keyword, which translates to that you can send ethers to payme function. As all the ethers sent to payable functions are owned by contract, Sample contract owns all of the ethers.

Payable fallback functions :

A function having no name and marked with payable keyword is called payable fallback function. There could be single such function in contract.

Example :

```
contract Sample {
function() payable {
//sample fallback
}
}
```

Address and Address Payable

Address and address payable difference introduced in Solidity version 0.5.0. The intention was to separate addresses that can receive money, and those who can't (used for other purposes). Hence, an address payable can receive Ether, while a plain address cannot. Hence, in Solidity, Ether can be sent to a variable defined as address payable but cannot be sent to a variable defined as address. Use the keyword payable before the variable name of an address type to enable it to receive ether.

For Example :

```
contract Vaults {
// United States gold reserves (cannot receive ether)
  address FortKnox;
// deposit ether here
  address payable etherVault;
}
```

The type returned by msg.sender is of address payable type. Therefore, the following methods become available for an address defined as payable in Solidity: transfer() , .send() , .call() , .delegatecall() and .staticcall()

3.6.12 Solidity Contracts and Constructors

Contract contains Constructor, state variables and functions. Constructor is a special function declared with constructor keyword which will be executed once per contract and is invoked when a contract is created. State variables and functions are already explained. The visibility quantifiers which are public, private, internal and external are explained in above sections.

Example :

```
pragma solidity ^0.8.2;
contract C {
  //private state variable uint private data;
  //public state variable
  uint public info;
  //constructor
  constructor() public {
    info = 10;
  }
  //private function
  function increment(uint a) private pure returns(uint) { return a + 1; }
  //public function
  function updateData(uint a) public { data = a; }
  function getData() public view returns(uint) { return data; }
  function compute(uint a, uint b) internal pure returns (uint) { return a + b; }
}
//External Contract
contract D {
  function readData() public returns(uint) {
    C c = new C();
    c.updateData(7);
    return c.getData();
  }
}
//Derived Contract
contract E is C {
```



```
  uint private result;
  C private c;
  constructor() public {
    c = new C();
  }
  function getComputedResult() public {
    result = compute(3, 5);
  }
  function getResult() public view returns(uint) { return result; }
  function getData() public view returns(uint) { return c.info(); }
}
```

Output

0: uint256: 8

Abstract Contract

This contract includes at least one function that is short of any implementation and is used as a base contract. Generally, an abstract contract includes both implemented together with abstract functions. Derived contract will implement the abstract function and use the existing functions as and when essential. If, a derived contract is not implementing the abstract function then this derived contract will be marked as abstract.

Example :

```
pragma solidity ^0.8.2;
contract Calculator {
  function getResult() public view returns(uint);
}
contract Test is Calculator {
  function getResult() public view returns(uint) {
    uint a = 1;
    uint b = 2;
    uint result = a + b;
    return result;
  }
}
```

Output

0: uint256: 3

Interfaces

Interfaces are just like as abstract contracts and are created using interface keyword. Interface are without any function with implementation. Functions of an interface are only of type external. Interface is without constructor and state variables. Interface can have enum, structs that are accessed using interface name dot notation.

Example

```
pragma solidity ^0.8.2;
interface Calculator {
  function getResult() external view returns(uint);
}
contract Test is Calculator {
  constructor() public {}
  function getResult() external view returns(uint){
    uint a = 1;
    uint b = 2;
    uint result = a + b;
    return result;
  }
}
```

Output

0: uint256: 3

3.6.13 Inheritance in Solidity

Functionality of a contract can be extended with inheritance. In Solidity both single and multiple inheritance is supported. Following are the important highlights related to inheritance.

- A derived contract can access all non-private members together with internal methods and state variables. But using this is not permissible.
- Function overriding is permissible if function signature remains similar. If difference output parameters differ the compilation fails.
- It is allowed to call a super contract's function using super keyword or using super contract name.
- If multiple inheritance is used then function call using super gives preference to most derived contract.

Example :

```
pragma solidity ^0.8.2;
contract C {
  //private state variable
  uint private data;
```



```
  //public state variable
  uint public info;
  //constructor
  constructor() public {
    info = 10;
  }
  //private function
  function increment(uint a) private pure returns(uint) { return a + 1; }
  //public function
  function updateData(uint a) public { data = a; }
  function getData() public view returns(uint) { return data; }
  function compute(uint a, uint b) internal pure returns (uint) { return a + b; }
}
//Derived Contract
contract E is C {
  uint private result;
  C private c;
  constructor() public {
    c = new C();
  }
  function getComputedResult() public {
    result = compute(3, 5);
  }
  function getResult() public view returns(uint) { return result; }
  function getData() public view returns(uint) { return c.info(); }
}
```

Output

0: uint256: 8

3.6.14 Solidity Error Handling

Solidity offers several functions for error handling. Usually, on occurring of an error, the state is reverted back to its original state. Other verifications are to avoid unauthorized code access. Following important methods are used in error handling of Solidity.

- **assert(bool condition):** If condition fails to meet, this method call causes an invalid opcode and any altered state gets reverted. This method is often used for internal errors.

- **require(bool condition) :** If condition fails to meet then this method call reverts to original state. This method is used for errors in inputs or external components.
- **require(bool condition, string memory message) :** If condition fails to meet then this method call reverts to original state. This method is used for errors in inputs or external components. It offers an option to provide a custom message.
- revert() – This method aborts the execution and revert any alternations carried out to the state.
- **revert(string memory reason) :** This method aborts the execution and revert any alternations carried out to the state. It provides an option to offer a custom message.

Example of error handling in Solidity

```
pragma solidity ^0.8.2;
contract Vendor {
  address public seller;
  modifier onlySeller() {
    require(
      msg.sender == seller,
      "Only seller can call this."
    );
    _;
  }
  function sell(uint amount) public payable onlySeller {
    if (amount > msg.value / 2 ether)
      revert("Not enough Ether provided.");
    // Perform the sell operation.
  }
}
```

Calling the revert return the hexadecimal data shown below.

Output

```
0x08c379a0 // Function selector for Error(string)
0x0000000000000000000000000000000000000000000000000000000000000020 //Data offset
0x000000000000000000000000000000000000000000000000000000000000001a //String length
0x4e6f7420656e6f756768204574686572207072696465642e000000000000 //String data
```

3.7 Case Study: Voting Contract App

- The main goal of an electronic voting systems is to reduce the cost of running an election, while ensuring the election integrity by fulfilling the security, privacy and compliance requirements. Blockchain based election system has the potential to limit fraud while making the voting process traceable and verifiable.

- Conventional voting methods incurs more expenditure and are more time consuming. Electronic voting systems has enhanced the efficiency and the integrity of the election process. They are more flexible, simple to use, and affordable costs compared to general elections.
- Apart from these benefits of e-voting systems, it is found that the, existing electronic voting methods face the risks of running into over-authority and manipulated details, limiting basic fairness, privacy, confidentiality, anonymity, and transparency in the voting process. E-voting procedures are not decentralized and approved by the critical authority that controls, measures, and monitors the process in an electronic voting system. Hence, there are more chances of obstructing a transparent voting process. It is important to guarantee that assurance in voting does not diminish. Blockchain can overcome the shortcomings in the voting method used in modern elections and make the polling process transparent and accessible to stop illegal voting. Blockchain can also strengthen the data protection to guarantee a transparent outcome of the polling process. As blockchain is decentralized and replicated ledger, a smart contract-based electronic voting system will surely minimize the risks that are with electronic voting and permits for the process to be tamper-proof.

Design Considerations for Blockchain based Voting App

- In order to design any application, we have to first consider its design requirements. For implementing an e-voting system, it is necessary to think about its design requirements. This can be done by evaluating the existing e-voting systems and considering the requirements for such systems to be effectively used in a national election. Following are the general requirements for blockchain based voting system.
  - Forced voting should be prevented.
  - The proposed blockchain based voting system should permit a method of secure authentication via an identity verification service.
  - It should not permit traceability from votes to respective voters.
  - It should offer transparency, in the form of a verifiable guarantee to each voter that their vote was counted, correctly, and without risking the voter's privacy.
  - Tampering of vote given should be prevented.
  - Counting of votes should not be centralized with single authority.
  - Only eligible voter should be allowed to vote in an election.
- Considering above requirements, if we consider the election system in India, then following are the essential requirements.
  - Voter should have valid national ID to be eligible to vote in election. This can be Aadhaar number or Permanent account number (PAN) to recognize him/her as a citizen of India.
  - All the citizens having age 18 and above can vote in any election if they are already registered for voting.
  - More than one vote by single voter is not allowed.

Conditions for Denying the Right to Vote

- A voter can vote only at the recognized centre in constituency where they are registered. The voter-related information such as age, state, and constituency code are associated with the valid national ID number of the voter.

- Election commission's (EC) contract list should include the national ID number of the voter. If this ID does not exist in contract list of EC, then voter should not be allowed to cast the vote.
- Election is ongoing in already declared constituency and voter's constituency code is not matching with this constituency code.
- If voter has already casted the vote in his/her constituency.

Process of Voting

- Voter can login to the election commission website/portal by using national ID. This will provide you one time password (OTP).
- If the national ID registered does not exist in the contract (voter list) and the voter's constituency code match with the constituency where election is ongoing then EC portal will show it. Otherwise, the voters are not permitted to vote. If the election process is still going on, it will prompt you with the option to vote in the following manner.
  - First click Vote and then Select Candidate. After that finish the voting by submitting the vote.
  - Voters are not permitted to cast vote or update vote to different candidates again.
  - Transparency is ensured by displaying the final candidates list with the vote count after the election voting process is completed.

Smart Contract

As per defined assumptions, requirements, processes and constraints, following are the steps illustrating the process of voting.

1. Voter logs in using website or portal of EC.
   - If voter is valid, already registered and having age >= 18 years then candidate list will be displayed based on constituency code.
   - If voter is not valid then voting will be denied.
2. Valid voter initiates the voting process and will be allowed to vote if following conditions holds.
   - If age >= 18 years
   - If election is ongoing in declared constituency.
   - If voting is already not done with this ID, then will be allowed to cast the vote.
   - If all of the above conditions fulfil, vote will be casted to eligible/selected candidate in list.
3. If voting is yet not started in constituency, then reject the vote.

Entities in Voting Process

Following are the main participants in this voting process.

- Voters who cast the vote.
- Candidates to whom vote to be casted.
- Election Commission (EC).

A contract named **Ballot** can be created to define key entities in this process. Solidity structure can be used to define own data types as per user's requirement. The election commission will be the creator of the contract.

Voter

Each voter has a unique national ID number. Other details associated with the voter are name, age, state, and constituency code. Structure with name **Voter** can be created which includes the fields that define the national identification number of the voter, name of the voter, age, code of the state, constituency code, is alive and candidate to whom vote is casted.

Candidate Contesting the Election

- Candidates belong to different political parties or can be independent. They are the key entities in the process of the election. Each political party has its symbol and also EC allocates symbol to independent candidate. Each candidate contesting the election has to file for the election. Therefore, each candidate has a unique nomination number.
- Structure with name **Candidate** can be created that defines the attributes related to candidate. These attributes are name of the candidate, name of the party, symbol of the party, nomination number, number of votes received, whether independent or not, constituency code and state code.

Election Commission (EC)

- The election commission is responsible to create the contracts and has authority to update the voters and candidates list. EC will also set the start and end times for the voting. Instead of considering the several addresses to manage the entire election process, a single address with admin authorities is considered.
- This address is a state variable which that contains the Ballot contract. The voter and candidate will be declared as mapping to maintain the list of records of voter and candidates as the key value pair, where the key is the national ID number and the value is the voter object. Hence, for each key value (national ID number), only one voter record will be there and the same goes for candidates with a unique nomination ID.
- Following are the steps involved in interaction of voter with EC website/portal.
  1. f election process is already completed the website displays the result. Voter can view the result (step 8). If election process is ongoing and voter has to cast the vote then he/she will carry out login process by using his/her ID number. In this case, result of election can not be accessed (step 7). If user is valid registered user, then step 2 is carried out. Voter's ID number should be valid, age should be greater than or equal to 18 and he must be alive. All these conditions should be met for voter to a valid voter.
  2. The ID of the voter will be presented based on their constituency along with the list of candidates to vote for. The voter now will select the candidate to whom the vote will be casted.
  3. The vote is initiated by voter and validation of step 4 and step 5 is done. After validation of these steps, vote gets registered against candidate and his/her vote count gets incremented.
  4. The contract verifies if voting lines for the election are open. If yes then flow moves to step 5 else it moves to Step 6.
  5. In this step, eligibility conditions for voting are verified for the voter. These conditions are, age is greater than or equal to 18, this voting is first time, and does the candidate the voter has initiated the vote for belongs to the same constituency as the voter. These verifications are carried out through a functional modifier.
  6. Voting access to voter is denied.
  7. Election is ongoing and result cannot be accessed.
  8. Election process is over and anyone can access the result of election.

Benefits of Blockchain based E-voting

- Enhanced security.
- Accurate and faster election process.
- Incurs less expenditure in for election process.
- Faster and accurate counting of votes.
- As human intervention is less, fraud is prevented.

Limitations of Blockchain based E-voting

- Voter should have internet facility available.
- Voter should be aware about how to use and access the web.
- Voters have to provide personal information.

3.8 Preparing for Smart Contract Development

- Smart contracts are one of the most capable and appealing ideas in blockchain technology. Their self-enforcing and event-driven features facilitates to carry out online activities without the involvement of trusted third parties. The security, privacy, and efficiency are the key problems for universal adoption of the smart contracts. To develop the smart contracts, a developer should be aware about the following three types of information.
  - The state-of-the-art design paradigms, design tools, and alternative systems available to create or execute smart contracts securely.
  - The current challenges involved in efficient development of secure smart contracts and their adoption by various applications.
  - The methods and research to overcome these challenges and limitations.
- There are various forms of smart contracts depending on the platforms on which they are running. Therefore, design schemes of smart contracts are platform dependent, especially the language they support. Smart contracts programmed in scripts are generally used to describe financial transactions. Whereas, smart contracts written in Turing-complete languages could theoretically describe any deterministic protocol as a computer program.

1. **Smart Contract Construction using Scripts**

   - Script-based blockchains typically offer simple stack-based opcodes to ease a more flexible circulation of cryptocurrencies. For example, it is possible for payer to specify a condition under which the payee receives his payment. The key purpose of such script languages is to facilitate simple financial affairs or demands. Hence, smart contracts written in script-based blockchains are relatively simple and limited compared with those in Turing-complete blockchains.
   - There are two categories of contract construction schemes. These are design paradigms and tools. Design paradigms denotes the modular patterns in functionalities, applicable to different scenarios and widely considered secure. Such schemes facilitate to develop secure smart contracts efficiently. Design tools denotes the solutions intended at assuring the security of smart contracts. Bitcoin is the example of script-based Blockchain.
   - Because of difficulty in script programming, many new languages for bitcoin relevant blockchains to develop smart contracts are proposed to improve the expressivity, readability, and verifiability of the Bitcoin scripts. These languages are further divided as high-level languages and intermediate representation languages based on their expressivity.

   - High-level languages are more expressive and can be directly used by ordinary developers with ease. On the other hand, intermediate representation languages are suitable during the compilation and security analysis.

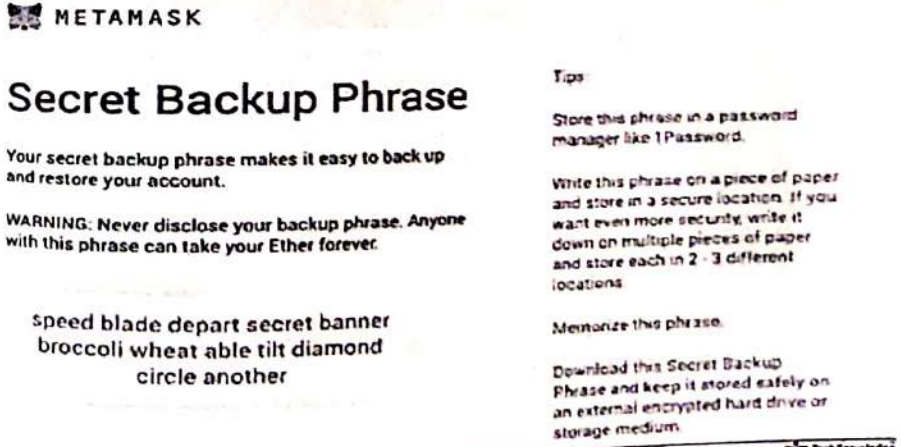
2. **Smart Contract Construction using Turing-Complete Languages**

   - Bitcoin's script language supports limited operations. Ethereum extends it by introducing a new virtual machine structure to support Turing-complete programming languages, significantly extending the application scenarios of smart contracts.
   - The smart contracts run in Ethereum Virtual Machine in the form of EVM bytecode. In fact, to ease the definition of the execution rules in smart contracts, several high-level programming languages are introduced, which can be converted into bytecode by compilers, such as Solidity, LLL (Lisp-Like Language), and Serpent.
   - These high-level languages provide high expressivity and can lessen the difficulty of contract construction. The execution mechanisms of Ethereum derivatives are almost the same as EVM and support Turing complete languages. These systems are Turing-complete blockchains.
   - To minimize the errors in contract programming, developers must refer contract design paradigms, which are cautiously designed against common attacks and recognized as safe. These paradigms are for specific applications and general purposes.
   - To be a experienced blockchain technologist, you must have both a fundamental understanding of smart contracts and the skills to build them on at least one of the available blockchain platforms.

Steps to Develop an Ethereum Smart Contract

Step 1 : Create a wallet at meta-mask

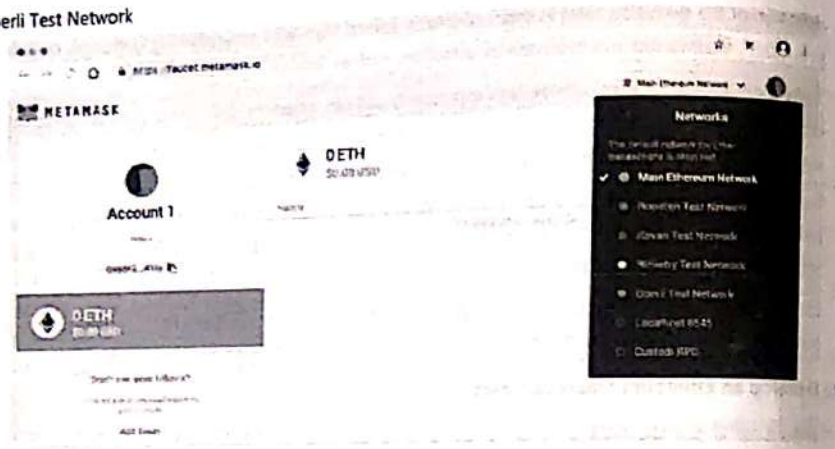
Install MetaMask in Chrome browser and enable it. After installation, click on its icon on the top right of the browser page. It will open in a new tab of the browser. Click on "Create Wallet" and click "I agree" to proceed further. Now you are agreeing to the terms and conditions. Create a password and then it will send you a secret backup phrase used for backing up and restoring the account. It should not be disclosed or shared with anyone, as this phrase can steal your Ethers. Ensure that you are in the "Main Ethereum Network." If you find a checkmark next to "Main Ethereum Network", you are in the right place.



Step 2 : Select any one test network

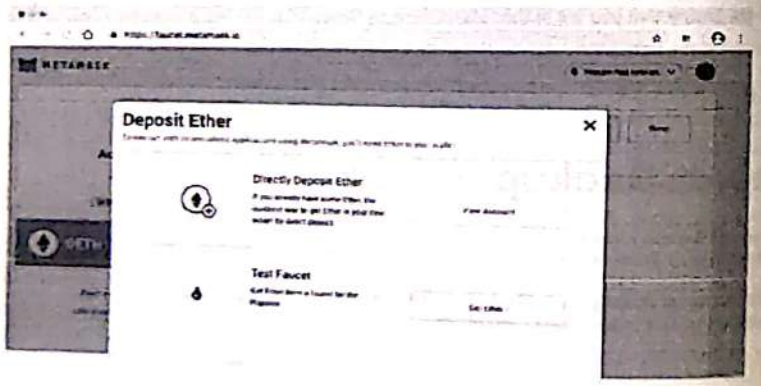
Following test networks can be seen in your MetaMask wallet. These networks are only for the purpose of testing and ethers involved in it have no value.

- Robsten Test Network
- Kovan Test Network
- Rinkeby Test Network
- Goerli Test Network

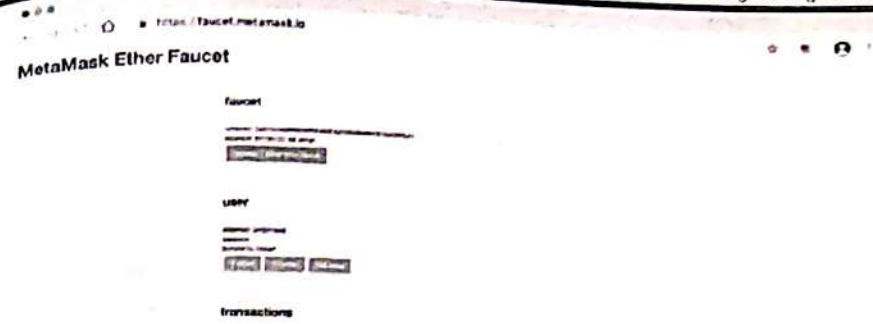


Step 3 : Add some dummy Ethers to your wallet

To test the smart contract, your MetaMask wallet should contain some dummy ethers. For example, if you test a contract using the Robsten test network, select it and you will find 0 ETH as the initial balance in your account. Click on the "Deposit" and "Get Ether" buttons under Test Faucet in order to add dummy ethers.



To proceed, you need to click "request one ether from the faucet," and 1 ETH will be added to your wallet. You can add as many Ethers you want to the test network.



Once the dummy ethers are added to the wallet, you can start writing smart contracts on the Remix Browser IDE in the Solidity programming language. Following figure shows 1 ETH is added.

Step 4 : Use editor remix to write the smart contract in Solidity

Use Remix Browser IDE to write our Solidity code. The remix is the best choice for writing smart contracts as it has a handful of features and offers a comprehensive development experience. It is usually used for writing smaller-sized contracts. Remix's features include :

- Warnings like gas cost, unsafe code, checks for overlapping variable names, whether functions can be constant or not
- Syntax and error highlighting
- Functions with injected Web3 objects
- Static analysis
- Integrated debugger
- Integrated testing and deployment environment
- Deploy directly to Mist or MetaMask

Step 5 : Create a .sol extension file

Open Remix Browser and click on the plus icon on the top left side, next to the browser to create a .sol extension file.

Step 6 : A sample smart contract code to create ERC20 tokens

ERC20.sol is a standard template for ERC20 tokens.

Code :

```
pragma solidity ^0.5.0;
import "./ERC20.sol";
contract myToken is ERC20{
mapping(address =&gt;uint256) public amount;
uint256 totalAmount;
string tokenName;
string tokenSymbol;
uint256 decimal;
constructor() public{
totalAmount = 10000 * 10**18;
amount[msg.sender]=totalAmount;
tokenName="Mytoken";
tokenSymbol="Mytoken";
decimal=18;
}
function totalSupply() public view returns(uint256){
return totalAmount;
}
function balanceOf(address to_who) public view
returns(uint256){
return amount[to_who];
}
function transfer(address to_a,uint256 _value) public
returns(bool){
require(_value&lt;=amount[msg.sender]);
amount[msg.sender]=amount[msg.sender]-_value;
amount[to_a]=amount[to_a]+_value;
return true;
}
}
```


Select a version of the compiler from Remix to compile the solidity Ethereum smart contract code.

Step 7 : Deploy your contract

- Deploy the smart contract at the Ethereum test network. Use deploy button at the Remix window's right-hand side and until the transaction is complete.
- Once the transaction commits successfully, the address of the smart contract would be visible at the right-hand side of the remix window.
- At first, all the ERC20 tokens will be stored in the wallet of a user who is deploying the smart contract.
- To check the tokens in your wallet, go to the metamask window, click add tokens, enter the smart contract address and click ok. You would be able to see the number of tokens there.

Steps to test an Ethereum smart contract

- Execute all smart contract methods like transfer, total supply, and balance. These methods appear at the right-hand side of the remix window and you can run all the processes from there itself.
- Transfer some tokens to other ethereum wallet addresses and then check the balance of that address by calling the balance method.
- Get total supply by running the total supply method.

Steps to deploy Ethereum Smart Contracts

- To make your smart contract live, switch to the main ethereum network at metamask
- Add some real ethers.
- Now again, deploy your smart contract using remix as mentioned in the above steps.
- When a smart contract is deployed successfully, visit http://www.etherscan.io and search your smart contract address there. Select your smart contract.
- Now you need to verify your smart contract here, click "verify the contract."
- Copy your smart contract code and paste it at Etherscan. Select the same compiler version that you selected at remix to compile your code.
- Check "optimization" to Yes, if you had selected optimization at remix; otherwise, select No.
- Click Verify.
- It will take a few minutes and your smart contract will be live if no issue occurs.
- You can now run your smart contract methods at Etherscan.

Review Questions

Q. 1 Define the smart contract.

Q. 2 What is smart contract? Explain in detail.

Q. 3 What are benefits of smart contract? Explain

Q. 4 Explain working of smart contract with example.

Q. 5 Explain various characteristics of smart contract.

Q. 6 Explain different types of smart contracts.

Q. 7 Write note on smart legal contracts.

Q. 8 Write short note on decentralized applications (DApps).

Q. 9 Write short note on Decentralized Autonomous Organizations (DAO).

Q. 10 Write short note on Smart Contracting Devices (SCD)

Q. 11 Explain basic structure of smart contract.

Q. 12 What is oracle? Explain its types.

Q. 13 Explain various benefits of smart contracts.

Q. 14 Explain limitations of smart contracts.

Q. 15 Explain in short various data types supported in Solidity language.

Q. 16 What are the different types of variables in Solidity? Explain with example.

Q. 17 Explain the various global variables in Solidity language.

Q. 18 Explain with example scope of the local and state variables in Solidity language.

Q. 19 What are the various operators supported in Solidity language?

Q. 20 Explain string data type in Solidity language.

Q. 21 Explain with example different types of arrays supported in Solidity language.

Q. 22 Explain Enums and Struct data type in Solidity with example.

Q. 22 What is mappings and how it is created in Solidity language?

Q. 23 What are different function visibility in Solidity language?

Q. 24 Explain with example different function qualifiers in Solidity language.

Q. 25 Write notes on Address and Address Payable in Solidity language.

Q. 26 What is constructor? Explain with example.

Q. 27 What is Abstract contract in Solidity language?

Q. 28 Explain with example how inheritance is supported in Solidity language.

Q. 29 Explain Error handling in Solidity language.

Q. 30 Explain various steps to develop voting contract application.

Q. 31 What are benefits and limitations of blockchain based electronic voting application?

Q. 32 Explain various steps to develop an Ethereum smart contract.

□□□

4 PUBLIC BLOCKCHAIN

MODULE - 4

Syllabus

Introduction to Public Blockchain, Ethereum and its Components, Mining in Ethereum, Ethereum Virtual Machine(EVM), Transaction, Accounts, Architecture and workflow, Comparison between Bitcoin and Ethereum. Types of test-networks used in Ethereum, Transferring Ethers using Metamask, Mist Wallet, Ethereum frameworks. Case study of Ganache for Ethereum Blockchain, Exploring etherscan.io, and ether block structure.

4.1 Introduction to Public Blockchain

- In this module we will learn the Ethereum blockchain and discuss how it works. Ethereum is a public blockchain.
- It is open to everyone, no matter your location, background, skin, education or age.
- Blockchain can be divided into two main categories according to accessing and managing permissions.
  - Public Blockchain – Permissionless
  - Private Blockchain – Permissioned
- In a public blockchain anyone is able to join or leave the network at any time, meaning that any individual may access the information, submit and receive transactions and also get involved in the process of verifying transactions.
- A private blockchain operates on the basis of access rules that limit the number of people who can join the network. We will discuss private blockchain in detail in the next module.
- The public blockchain model, which is used by Bitcoin, Ethereum, and Litecoin, is largely viewed as the original distributed ledger structure.
- Because it is completely open, anyone may join and participate in this type of blockchain.
- It can accept and send transactions from anywhere on the globe and is auditable by everyone in the system.
- Public blockchain is not only decentralized, but also fully distributed, because each node connected to the network has the same amount of transmission and power as all the others.
- In order to be considered legitimate, a transaction should be authorized by each of its constituent nodes via the consensus process.
- When a record is authorized, it is recorded on the chain.
- Incentives are commonly used in public blockchains to incentivize individuals to join the network and to authenticate transactions.
- Data kept on a public blockchain is secure because it cannot be changed or edited after it has been validated by the blockchain network.

- In the public blockchain, each network node possesses a complete copy of the ledger. Each user of the permissionless ledger keeps a copy of the ledger on their local nodes, and all users adopt a distributed consensus algorithm to decide on the ledger's final state.
- One of the most significant drawbacks of a public blockchain is its complete transparency. This level of transparency implies minimal to no transaction privacy and supports a poor security paradigm.
- Another disadvantage is the large amount of computational power required to keep the ledger up to date.

4.2 Ethereum

- Ethereum, founded in 2015 by Vitalik Buterin, is a unique blockchain with a unique token known as Ether.
- While the ownership of digital currency is verified by Bitcoin via the blockchain, the Ethereum platform enables developers to create fully distributed apps and does away with the "centralized" client-server internet architecture.
- The key feature of Ethereum is its programmability, which enables the development of applications.
- Ethereum is a blockchain-based open software platform that allows developers to create and deploy decentralized applications.
- It is more than simply a platform for cryptocurrencies; it has the ability to perform tasks and develop distributed applications.
- Ethereum is a DIY platform for decentralized applications known as Dapps (decentralized apps). It is the global infrastructure for operating Dapps.
- The Ethereum network is completely decentralized since it is run by thousands of separate computers.
- Once the program is deployed to the Ethereum network, these computers also identified as nodes will, be assured that it execute as written.
- It was the first technology to implement a smart contract. It permits participants to run decentralized blockchain applications, which are called Dapps.
- Ethereum blockchain is an immutable public database that stores a record of every transactions that has ever taken place.
- Ethereum is a Turing complete system.
- In its virtual machine, Ethereum can run programmes and read and write data to memory. Any programme that a Turing complete machine can compute may be done by this machine.
- Ethereum's major point of differentiation is how it mixes blockchain technology and computing power. When the programmability feature is paired with all of the blockchain's distinguishing characteristics, Ethereum becomes a practical technology.
- It is referred to as ETH and has its own currency called Ether.

What is Ether?

- Ether (ETH) is the native cryptocurrency of Ethereum.
- It is the second largest cryptocurrency around.
- The goal of ether is to create a market for computing.
- The Ethereum network and smart contracts are powered by ether, which is used as fuel (or gas).
- The Ethereum network's users utilize ether to pay for services rendered by the network.
- Ethereum allows us to send ether which is equivalent to Bitcoin.

Fig. 4.2.1 : Ether Cryptocurrency

4.3 Ethereum Components

Ethereum's components are as follows :

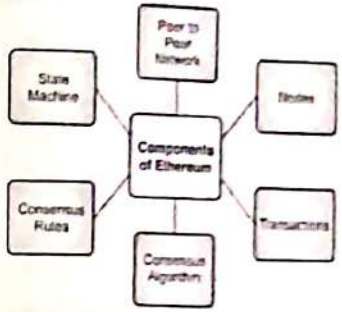


Fig. 4.3.1 : Components of Ethereum

- Ethereum is a peer-to-peer network of computers named Ethereum nodes or clients, each of which runs a compatible software. Anyone can run an Ethereum node and there are no special prerequisites or requirements.
- At the moment, there are thousands of Ethereum nodes that communicate with each other. In Fig. 4.3.2 and 4.3.3 shows, statistics about the Ethereum mainnet.

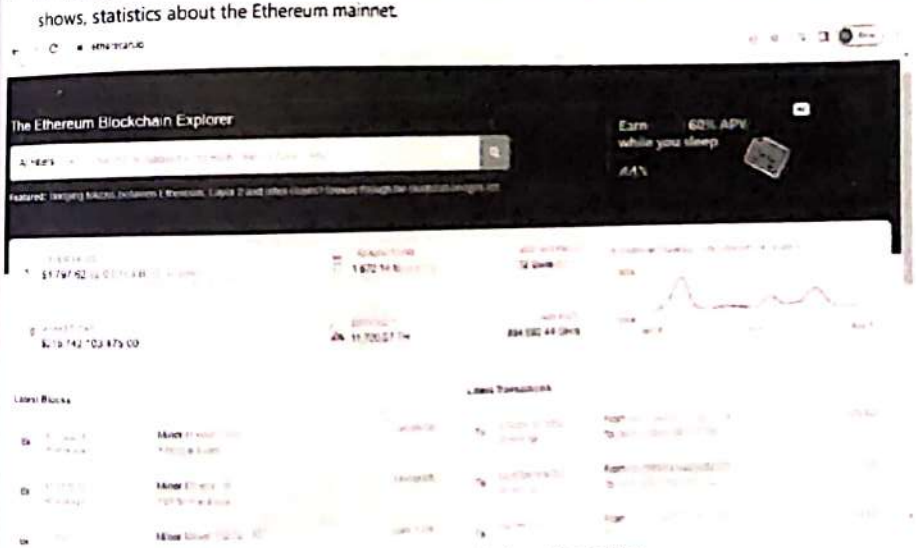


Fig. 4.3.2 : Ethereum latest blocks and transaction

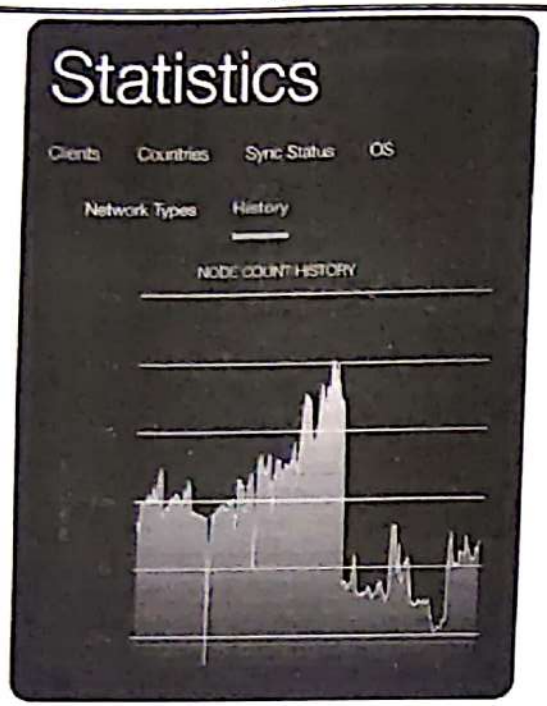


Fig. 4.3.3 : Ethereum Mainnet Statistics

- There are more ways for a user or application to communicate with the network. Web 3.js is a collection of libraries that allows web applications to interact with an Ethereum node. using HTTP , IPC or WebSocket. A normal website cannot communicate with a Ethereum blockchain so we will need Web3.js
- Now let us discuss components of Ethereum.

1) **Peer-To-Peer (P2P) network :** Every computer or node on the Ethereum blockchain is linked to every other node. Ethereum operates on the Ethereum main network, which is accessible through TCP port 30303, and employs the ĐΞVp2p protocol.
2) **Consensus rules :** The set of rules used to determine if a transaction is legitimate
3) **Transactions :** On a Ethereum network, transactions are the messages that are sent and received. A value, a sender, a receiver, and a payload of data are all components of a transaction.
4) **State machine :** The state machine known as the Ethereum Virtual Machine, or EVM for short, processes Ethereum transactions and state changes. The Ethereum virtual machine executes the program in a machine language (bytecodes). These programmes are created in a high level language, such as Solidity, and are then converted into bytecode via a compiler. The program written in a high level language is called a smart contract.
5) **Consensus algorithm :** The distributed Ethereum network's consensus algorithm is the process used to reach consensus on the longest chain. Proof of work and proof of stake are the two consensus methods for Ethereum that are most often utilized.

6) **Data structures :** The state of Ethereum is kept locally on each node in the form of a database, often Google's LevelDB, which keeps the system information and transactions in a serialized hashed data structure known as a Merkle Patricia Tree.

4.4 Mining in Ethereum

- Since there is no one owner of a distributed system, computers are free to join the Ethereum network and start verifying transactions whenever they choose. This process is known as mining.
- Mining nodes consult one another (consensus) to agree on the sequence of transactions across the system, which is required to instantly calculate everyone's account balances despite the fact that many transactions are going through the network.

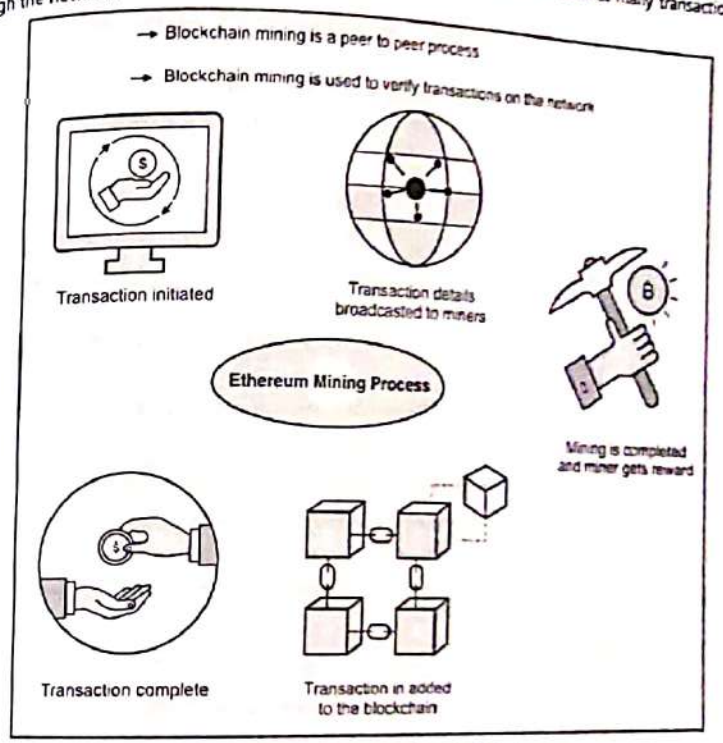


Fig. 4.4.1 : Mining Process Overview In Ethereum

- Because this process uses electricity, which is an expensive, miners receive a reward of around 5 ethers for each block they mine.
- Mining is a core component of both Bitcoin and Ethereum, and it functions much the same way in both networks, with a few exceptions.
- Mining also secures the network by creating, verifying, publishing and propagating blocks in the Blockchain.
- New blocks are added to the blockchain through mining, a resource-intensive process.

- Blocks are made up of transactions that are added to the blockchain after being verified by the mining process by mining nodes.
- Mining protects the network by ensuring that only legitimate blocks containing valid transactions are permitted on the Blockchain.
- After a block is accepted into the chain, it cannot be modified, and miners safeguard the network by permitting only legitimate transactions in their proposed block.
- This method uses a lot of energy to make sure that miners have used all the resources necessary for a block to be approved.
- One new block is produced (mined) roughly every ten minutes.

The Ethereum Mining Algorithm

- The Ethereum mining algorithm is no longer based on the proof of work that was employed in Bitcoin. Ethash is the algorithm used in Proof-of-Work mining.
- Instead, Ethereum employs the Casper proof of stake process, which consumes significantly less computational power while producing the identical results as the more expensive proof of work approach.
- Ethereum developers are working on a consensus algorithm known as "Casper."
- Casper is an economic consensus protocol based on security deposits.
- Bonded validators put up a security deposit in order to contribute to consensus by proposing and creating blocks.
- A group of bonded validators exists. A validator is a user who helps with the mining process by verifying transactions and producing blocks.
- By staking a quantity of ether, a user can participate and become a validator in the set. The security deposit must be a small amount of ether.
- Bonding is the deposit of a stake. After the deposit, the user becomes a bound validator.
- A list of validators is constructed from the set of bonded validators for each block.
- The list is chosen pseudo-randomly, with the randomness weighted by the initial stake put by a validator.
- Thus, if a validator deposits 10% of the total stake, he has a 10% chance of being picked to propose the next block.
- The first validator is permitted to propose a block from this priority list. If the first validator does not react or does not respond within a certain length of time, the next validator on the list may propose a block, and so on.

4.5 Ethereum Virtual Machine (EVM)

- Before Ethereum, blockchain apps could only be created to do a small number of tasks.
- The only purpose for which crypto currencies were intended was to function as peer-to-peer digital currencies.
- This presented a challenge for programmers since they either had to increase the scope of operations that these crypto currencies could do, which would be difficult and time-consuming, or they could create a new class of blockchain-based platform and programmes.
- The EVM is a cutting-edge strategy created by the Ethereum creator.
- The Ethereum virtual machine is the engine in which transaction code gets executed.
- The Ethereum Virtual Machine (EVM) is a global computer that anybody may utilize for a nominal fee paid in cryptocurrency.

- Ethereum smart contracts are sets of computer instructions that are executed on all nodes that have a full Ethereum client installed. The portion of Ethereum that executes smart contract instructions is known as the EVM.
- EVM may simply be viewed as a system created to work as a runtime environment for **smart contracts** based on Ethereum. This is where all your smart contracts are executed.

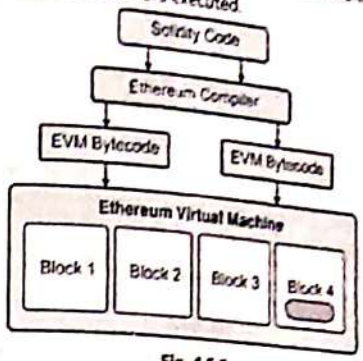


Fig. 4.5.1

- Every Ethereum node runs EVM.
- It is a virtual machine similar to the JVM used by Java.
- An Ethereum node runs EVM and runs applications based on a global consensus mechanism. This EVM has its microkernel, stack, memory and storage.
- Smart contracts are represented at a basic level by Ethereum Bytecode, which is read by the EVM.
- The bytecode for Ethereum is an assembly language made up of several opcodes. On the Ethereum blockchain, each opcode carries out a specific operation.
- It is Turing-complete software that runs on the Ethereum network and enables anybody, given enough memory and time, to run any program they choose, independent of the programming language it was created in.
- The EVM streamlines and improves the process of developing blockchain-based apps.
- The EVM makes it possible to construct hundreds of distinct apps on a single platform rather than having to create a unique blockchain for each application.
- It is completely isolated and has no access to file systems or processes. So, each node in the system runs the EVM. That way it can maintain a consensus across the network.
- When we install and launch the geth, parity, or any other client, the EVM is launched and begins synchronizing, verifying, and executing transactions.

4.6 Transaction and Accounts

4.6.1 Transaction

What Is a Transaction ?

- A blockchain's basic building block is a transaction. Currency is transferred from one address to another during a transaction.
- A transaction is a piece of cryptographically signed instructions issued by an externally owned account, serialized, and then uploaded to a blockchain.

- On Ethereum, a transaction is a data packet that has been digitally signed using a private key and contains the instructions that, when followed, either lead to a message call or the establishment of a contract.
- Based on the output they generate, transactions may be classified into two types :
  - Message call transactions: Simply, this transaction creates a message call that may be used to transfer messages across accounts.
  - Contract creation transactions: As the name implies, a new contract is made as a result of these transactions. This indicates that when this transaction is completed successfully, an account is created with the corresponding code.
- The way the outside world interacts with the Ethereum network is through transactions. As a result, they alter and update Ethereum state. For example, if Nishad sends Swara 2 ETH, Nishad's account must be debited and Swara's must be credited. This change of state occurs within a transaction.

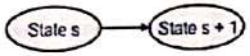


Fig. 4.6.1

- Any activity, including transferring ether or invoking smart contract functionality, can be considered a transaction. As a result, the state of the EVM is updated.
- Transactions that alter the EVM's state must be broadcast to everyone on the network. Any node may broadcast a request to conduct a transaction on the EVM;
- A miner will then carry out the transaction and communicate the ensuing state change to the rest of the network when this has occurred.
- In order for a transaction to be acceptable, it must be mined and cost a fee. A gas limit and a gas charge are applied to each transaction. For confirming (or mining) that transaction, miners are compensated with this fee.
- Now we have a basic understanding of what a transaction is, what it is used for. Let's understand in detail the fields that are included in these transactions.

Transaction Components :

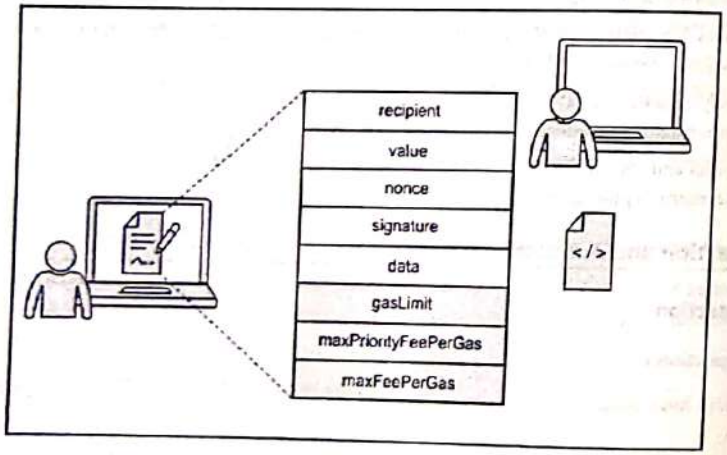


Fig. 4.6.2

- **Nonce :** A nonce is a unique number that is required for every single transaction. It is a number showing the sequence of a transaction. For an externally owned account, this number is the number of transactions sent from the account. For a contract account, it is the number of contracts created by the account. It is used to prevent message replay.
- **Recipient :** The recipient of the transaction can either be an externally owned account (individual human) or a smart contract. Whenever the recipient is another externally owned account, then the transaction is simply a transfer of value from one account to another account. When the recipient is a contract account, that transaction is going to trigger the code in the contract to run or execute.
- **GasPrice :** representing the fee the sender is willing to pay for gas (in wei)
- **GasLimit :** The maximum amount of gas the sender is willing to buy for this transaction
- **Signature :** Digital signature. A signature identifying the sender
- **INIT :** Init is used only in transactions that are intended to create contracts. This is an unlimited byte array that defines the EVM code to be utilized in the account initialization procedure.
- **The main "payload" of a transaction is contained in two fields :** value and data. Data and value can both be present in a transaction, but they can also be absent. A payment is a single-value transaction. A data-only transaction is referred to as an invocation. Both a payment and an invocation are parts of a transaction that includes value and data.
- **Value :** The amount of ether that you want to send to the recipient.(in ETH or in WEI)
- **DATA :** Data that can be sent. This field is optional and can be left blank.

4.6.2 Accounts

- A crucial part of the Ethereum system is an account.
- It is the primary component and has an ether (ETH) balance.
- A user's account can transmit transactions through the Ethereum network.
- Transactions are produced by the interactions between several accounts and are kept in the distributed database.
- Sender, recipient, and contract accounts are all involved in every transaction.
- An account can be managed by a user or by a programme running on the Ethereum network.
- Ethereum accounts are divided into two types:
  - Externally owned accounts (EOA)
  - Contract accounts

Externally owned accounts (EOA) :

- EOA is controlled via private keys from a user and the contract accounts are controlled by their contract code.
- The term "Externally Owned Account" is referred to as EOA. This is the type of account that is owned by people on the Ethereum network.
- A public-private key pair is produced for an account once the user creates one on Ethereum. The name or any other personal information of the user does not identify the account.

What Is Externally Owned Account ?

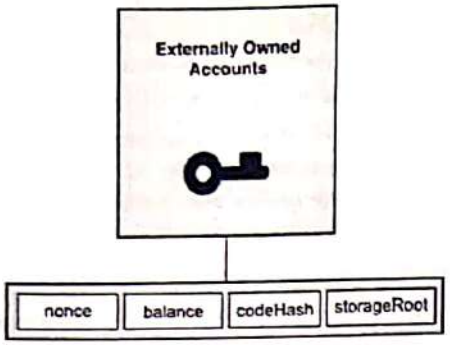


Fig. 4.6.3 : Externally Owned Account

- Externally owned accounts do not contain code; it has balance in ether and can send messages by generating and signing a transaction.
- It is used for holding, sending and receiving ETH and for interacting with smart contracts like deployment and calling functions etc.
- Access and control over one's contracts and assets are provided through the EOA private key. The user maintains security for the private key.
- Although, as the name indicates, the account's public key is accessible. The identity of the account is represented by this key. Through the use of a one-way cryptographic function, the public key is created from the private key.
- If you register for an Ethereum account, you will keep the private key to yourself while sharing the public key.
- Public keys are used for account-to-account transactions. An Ethereum account's public key is what creates the account's address.
- Other externally owned accounts and contract accounts can communicate with this kind of account.

Contract Accounts :

- A contract account has a code for a smart contract.
- This account doesn't have a public or private key; the public address of the contract account is used to identify it. Private keys are not owned by this account.

What is Contract Account ?

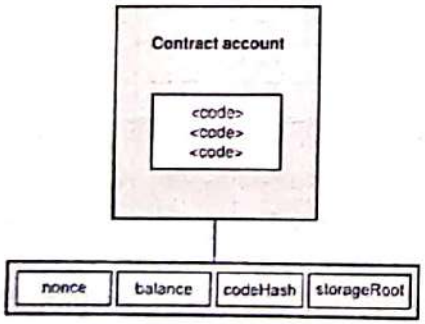


Fig. 4.6.4 : Contract Account

- Functions to run and state variables are present in this code.
- This account is managed by the smart contract's logic. The owner of the contract's account is the smart contract that is implemented on the Ethereum blockchain.
- The code for the contract is executed on the Ethereum virtual machine when an externally owned account begins a transaction with a contract account.
- When a contract account receives a message, it activates its code. This might imply that it reads and writes to its storage, transmits messages, or generates new contracts.
- In addition to ether, the transaction also contains information on which functions should be executed and what arguments should be sent to those functions.
- There are four fields in the account state: nonce, balance, storageroot, and codehash
  - **Nonce** - The nonce serves as a counter, ensuring that each transaction is only completed once.
  - **The current ether balance** - The amount of ether the user presently holds. (in wei)
  - **Codehash or Contract code** - If the account is EOA, this field is empty. This field includes the code that will be done when a message is sent to the account if the account is a contract account. This is immutable EVM bytrecode.
  - **Storage** - The storage is initially empty, but it may be filled and read from through transactions and contract codes. (only for contract account)

Externally Owned Account	Contract Account
Owned by some external entity (person, corporation etc)	Owned by contract
This account does not have a code.	A code is assigned to each contract account.
A transaction can be initiated by an externally owned account with another externally owned account or a contract account.	But contract account cannot initiate transactions.
Externally owned accounts have a private key associated with them	Contract accounts do not have a private key
It is free to open an externally owned account.	It costs money to set up a contract account.
The only type of transaction that may occur between two EOAs is the transfer of ether.	Any type of transaction with a contract account is permitted, including deploying or invoking the functionality of other contracts.

4.7 Architecture and Workflow

- In the decentralized world, Ethereum is sometimes referred to as the world computer.
- A Dapp is a blockchain-based application or service that allows direct contact between users and suppliers, such as linking buyers and sellers in a decentralized marketplace.
- A DApp, like a centralized application, typically consists of a decentralized backend that operates on the blockchain network and a centralized frontend that allows end users to access their wallets and perform transactions.

- The diagram below shows decentralized applications.

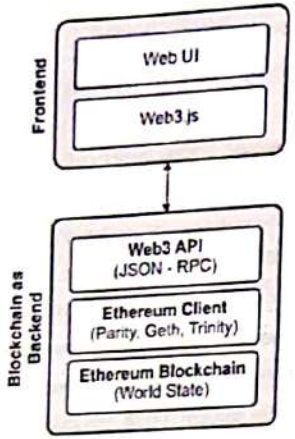


Fig. 4.7.1 : Decentralized Application Architecture with Ethereum

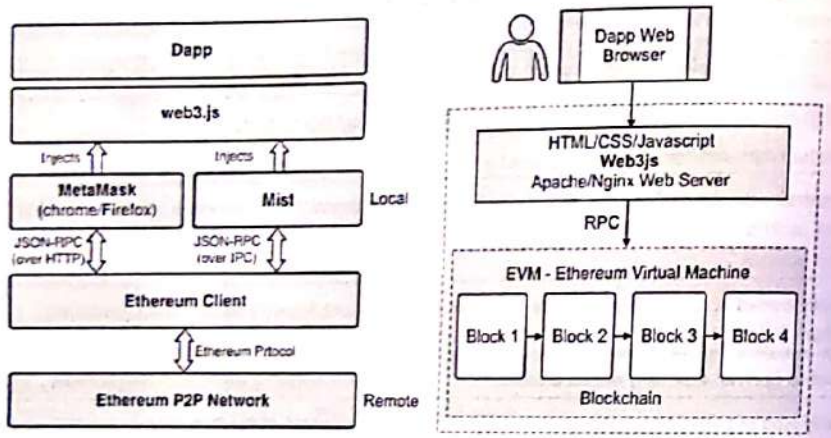


Fig. 4.7.2 : A Common Architecture for an Ethereum Dapp.

- In the above diagram, every client (browser) talks with its own application for instance. There is no single server to which all clients must connect.
- This implies that in an ideal decentralized world, everybody who wishes to engage with a dapp (Decentralized Application) will need to have a full copy of the blockchain running on their computer/phone/etc.
- That is, before you can use an application, you must first download the complete blockchain and then start using it.
- To utilize these apps, it is unreasonable to ask everyone to run a blockchain server.
- However, the goal of decentralization is to avoid relying on a single/centralized server.

- As a result, the community has devised methods (hosted blockchain servers, metamask, etc.) that allow you to avoid wasting a lot of hard drive and RAM downloading and operating a full copy of the blockchain while maintaining the decentralized component.
- So, in essence, the blockchain stores your data, stores the code, and executes the code in the EVM.
- To create web-based Dapps, Ethereum has a useful JavaScript library called web3.js that connects to your blockchain node.
- Now, the next question is where the smart contract actually fit into the Ethereum blockchain architecture?
- If we are using Remix and writing some solidity contracts, and then we deploy those contracts to the blockchain. Where do they actually go? Let's discuss this.

Workflow :

1. Write smart contract in solidity
2. Use Interactive Development Environment (IDE), Remix to create smart contract code.
3. This smart contract is going to get processed by miners included in a block and then distributed to the Ethereum network.
4. In Ethereum, node has to do with mining, so miners execute proof-of-work in order to find a hash and miners are also going to publish any new blocks.
5. Once a miner finds a new block, all those transactions get published and that's going to be fed into the state transition.

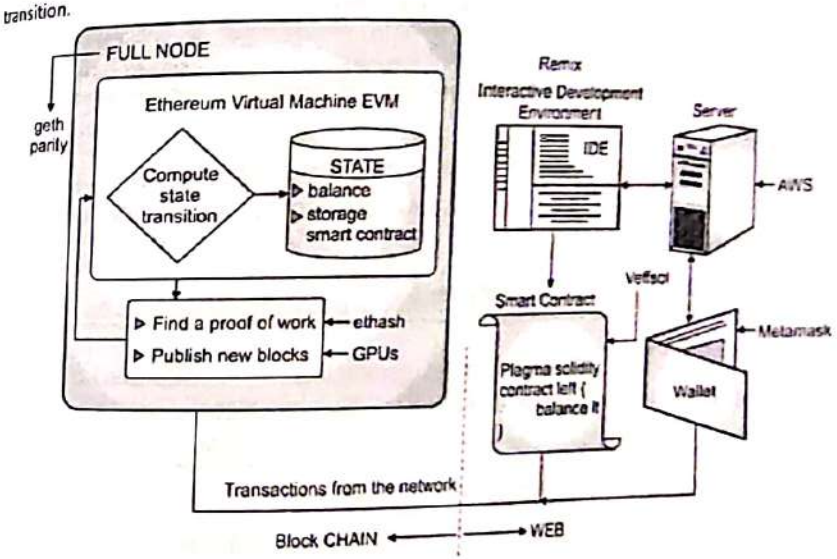


Fig. 4.7.3 : Workflow of Ethereum

6. The first thing inside Ethereum is going to be called the state and the state just represents to date what is happening on the Ethereum network.
7. Inside the state we have a balance of all the user accounts and, storage.

8. Next is how we decide on what those contracts do, how do we decide on all of the state updates. So compute state transition.
9. So, at this stage, smart contract is in the state-transition and it will actually end up in storage.
10. The state inside of Ethereum gets a special name, Ethereum Virtual Machine.
11. So EVM processes the computation and then also needs to update the state which involves the storage.
12. Now Metamask come in. So, if we are interacting through remix with Metamask, then it is a wallet to store your private keys.
13. So, both the contracts that are created and the transactions from any wallets come in and are processed by the miners.
14. The last step is server, wallets are usually apps either on a smartphone or on the desktop. And the remix was also hosted by somebody.
15. So you can see in the diagram, the server is hosting Remix and also hosting Metamask.
16. In the diagram, the full node from Ethereum is the blockchain part and then IDE, wallet, server. These are all on the web. And the smart contract is both, i.e. are going to be deployed on the web in order to get into the virtual machine.

Summary of workflow

- Use IDE to program a solidity smart contract.
- Smart contract gets deployed and deployment gets picked up by miners.
- Miners are using PoW in order to find a new block.
- Once this happens, transactions get packaged into a new block and transactions get computed by the state machine transition.
- If necessary, the contract gets stored and the account balances get updated by the state.
- Once all the transactions in the block have been processed, then we need to contact the miners and pick up any new transactions from new blocks.

4.8 Comparison between Bitcoin and Ethereum

Bitcoin	Ethereum
In Bitcoin SHA-256 is used in the algorithm of the Blockchain	In Ethereum, Keccak-256 or Keccak-512 are used. Keccak is a variant of SHA3.
Assets : Bitcoin	Assets : Ether
Bitcoin is exclusively used as an online currency.	Ethereum is often used as an online currency, however its blockchain is also utilized for decentralized applications such as voting systems, file storage, and so on.
Data related to Bitcoin network transactions is frequently utilized simply to keep track of transactions.	On the Ethereum network, for example, transactions may include executable code.

Bitcoin	Ethereum
Simple and robust	Complex and feature-rich
Bitcoin consumes much more energy than Ethereum.	Ethereum is certainly consuming an increasing amount of energy, but this will likely decrease significantly after it has transitioned to Proof-of-Stake.
Stack based primitive scripting language, not Turing-complete	Turing-complete scripting language
Key lengths Public key = 512 bits (compressed: 257 bits) Private key = 256 bits	Key lengths Public key = 512 bits Private key = 256 bits Address = 160 bits
Mining Proof-of-work	Mining Proof-of-Work to Proof-of-Stake
Signature size 512 bits (64 bytes)	Signature size 520 bits (65 bytes)
Energy consumption ≈ 3, 2 million households	Energy consumption ≈ 1, 1 million households
Transaction time Bitcoin is known for its slow and expensive transactions. It takes around 10 mins to complete a Bitcoin transactions.	Transaction time Ethereum transaction takes only 12 seconds.
Block time 10 minutes	Block time 12 seconds
Mining reward 12,5 Bitcoin per block	Mining reward 3 Ether plus some other rewards
Consensus time ≈ 1 hour	Consensus time ≈ 3 minutes

4.9 Types of test-networks used in Ethereum

- A version of a project is deployed to an Ethereum Test Network ("testnet"), which replicates Ethereum, before it is launched on the Ethereum blockchain (or before modifications are made to the blockchain itself).
- This allows developers and the community to test the project before actual assets are engaged.
- On a testnet, ether and tokens are simple to get and have no real-world value.
- There are presently three testnets in use, and each performs identically to the main blockchain where you're real Ether and tokens reside.

Ropsten

- The Ropsten test network is an Ethereum Proof-of-Work testnet. One can mine on the Ropsten network to obtain ETH.

Kovan

- The Parity team created the Kovan test network as a Proof-of-Authority testnet for Ethereum.

Rinkeby

- The Rinkeby test network is an Ethereum Proof-of-Authority testnet that was founded by the Geth team. At Rinkeby, one may obtain ETH by requesting it through a faucet.

Making a connection to a testnet

- Ethereum addresses and private keys that operate on Ethereum work on each testnet.
- Create a new wallet specifically for use on testnets.

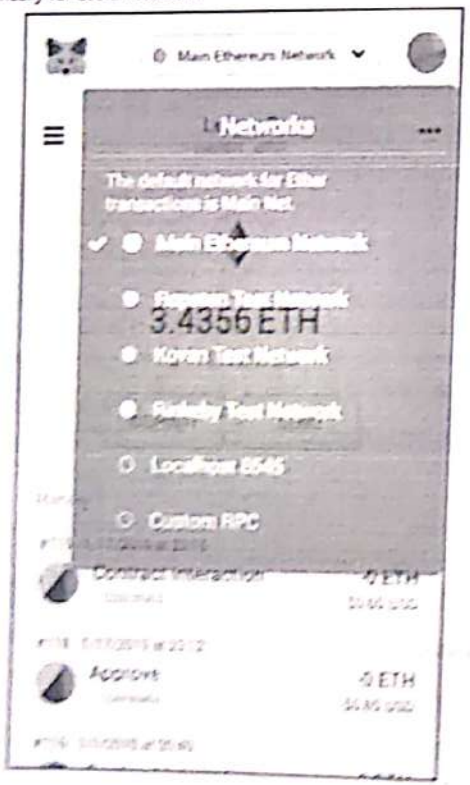
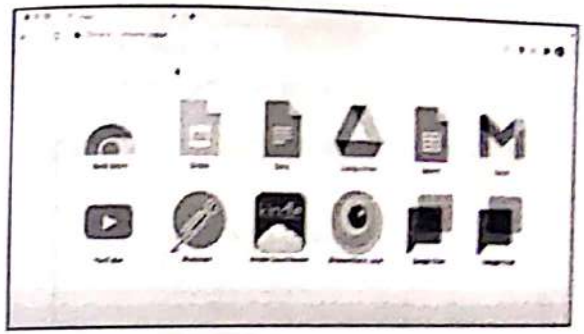


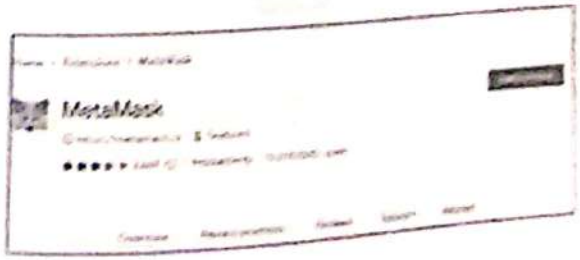
Fig. 4.9.1 : Metamask Plugin

- Using MetaMask to transmit Ether and tokens on a testnet is simple; simply choose an Ethereum network in the top-left corner of MetaMask.
- When you switch from the Main Ethereum Network to Rinkeby or another testnet, your balance and transaction history should change to reflect the network you've chosen.
- When you make a transaction in MetaMask, it will now be sent to the network you specify.

4.10 Transferring Ethers using Metamask

- Wallets are basically software that allows users to connect to the Ethereum.
- A wallet is a lightweight node that connects to a blockchain to conduct basic activities like transferring and receiving bitcoin.
- Let's discuss how to transfer ethers using Metamask.
- Download metamask first.
- Go to chrome apps -> Web Store -> Metamask

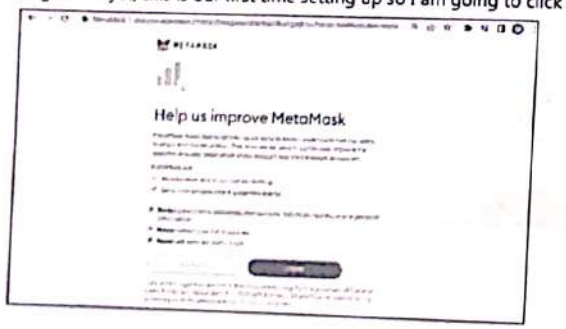




So now Ethereum wallet is in your browser.

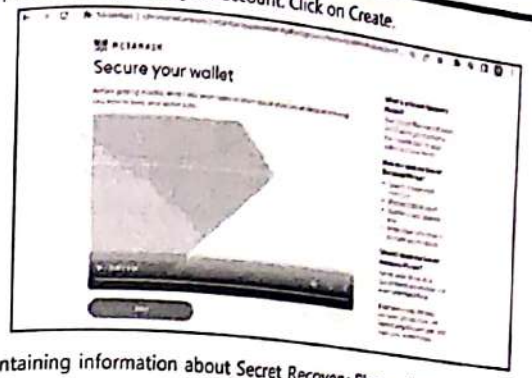
We can see Welcome to Metamask on browser connecting us to Ethereum and the decentralized web. Lets go ahead and get started

We don't have one existing wallet yet, this is our first time setting up so I am going to click Create a Wallet.

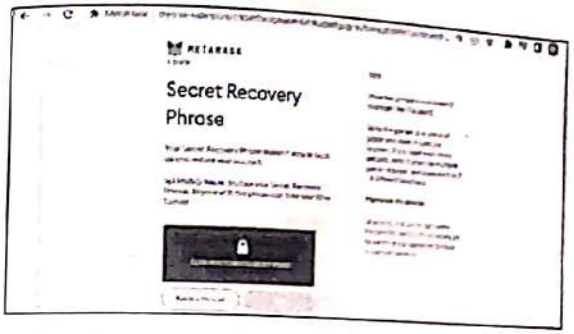


Click on I Agree.

Asking you to set a password to protect your account. Click on Create.

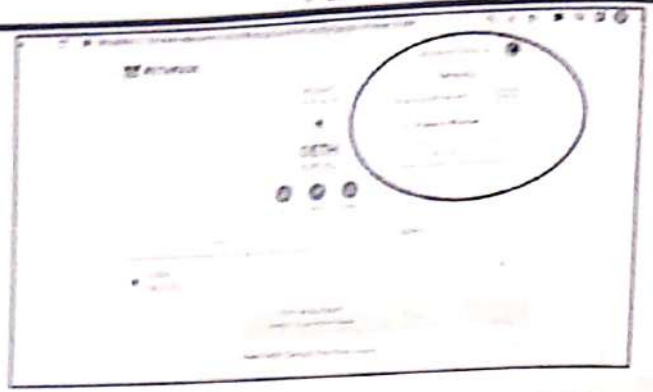


Watch this video containing information about Secret Recovery Phrase, how to save secret recovery phrase. Click on Next



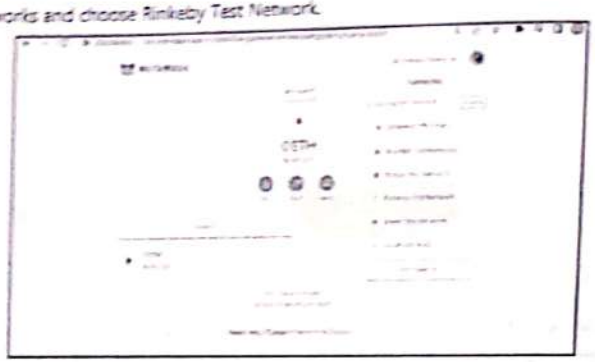
It explains Your Secret Recovery Phrase makes it easy to back up and restore your account. Click Next.

And done. We can see Account no. We can copy this and send it to friend and family.



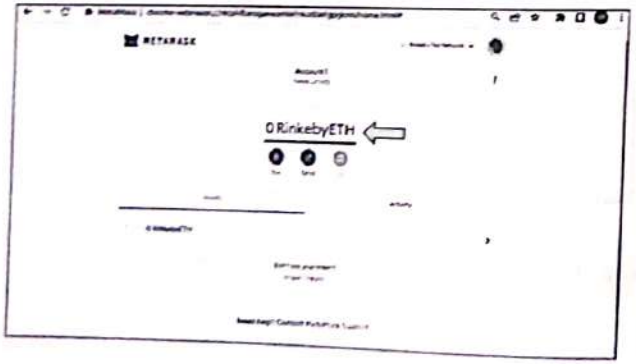
If you click on Ethereum Mainnet, we can see that we are connected to the main Ethereum network. Ethereum Mainnet is like the network, it's the actual Ethereum network where our ether is worth real money.

Click on show networks and choose Rinkeby Test Network.



Reason for choosing Rinkeby Test Network is, if we do test and deploy of smart contract on Mainnet , transaction fees are pretty high in main network and its very expensive for the development so we are using the test network where we will get ether free more or less.

If we switch between networks, our account no will be same across all the networks.

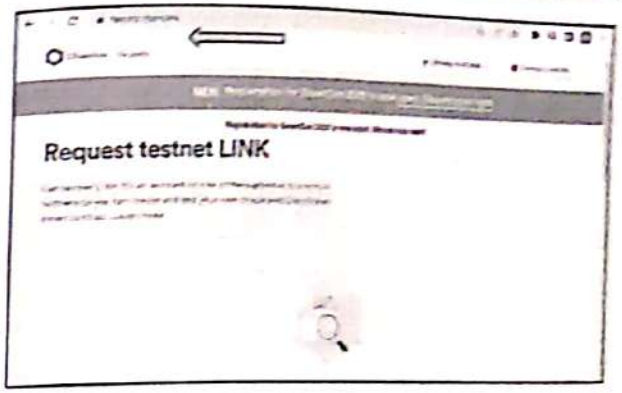


Now we need to get some ether in our account to send transactions, deploy smart contracts and so on. Write now it's showing 0 ETH.

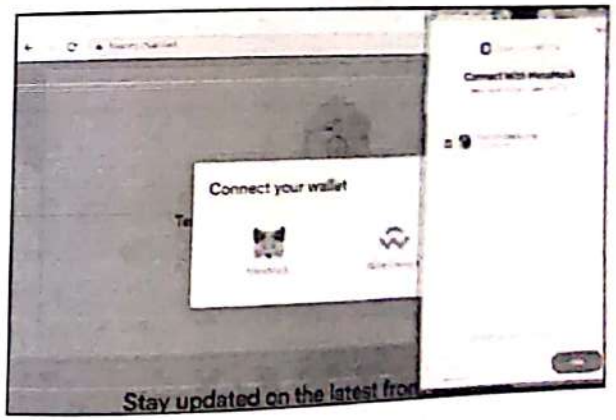
We will utilise a service called faucets, which is an Ethereum faucet and a developer tool for obtaining testnet Ether (ETH) in order to test and debug your decentralised application or protocol before going live on the Ethereum mainnet, where actual Ether must be used.

Most faucets require social authentication (e.g. Twitter post or login confirming you are a real human) or place you in a queue to wait for a testnet token through the faucet.

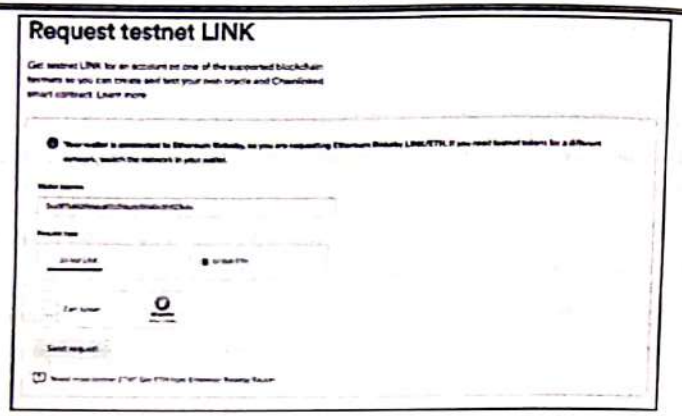
Go to Google and type rinkeby faucet and choose Faucets | Chainlink https://faucets.chain.link



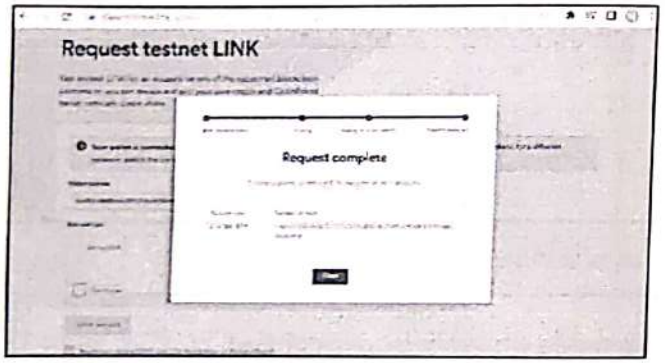
Now we need account no, So click on Connect Wallet and choose Metamask.



Click on Next and will get following screen

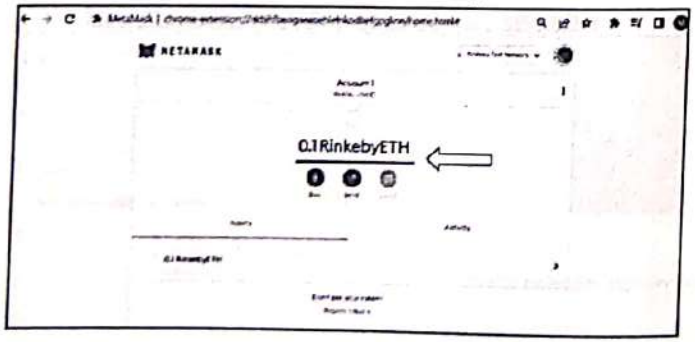


Uncheck 20 test Link and click on Send request. It will take 10-15 seconds for Token transferred.

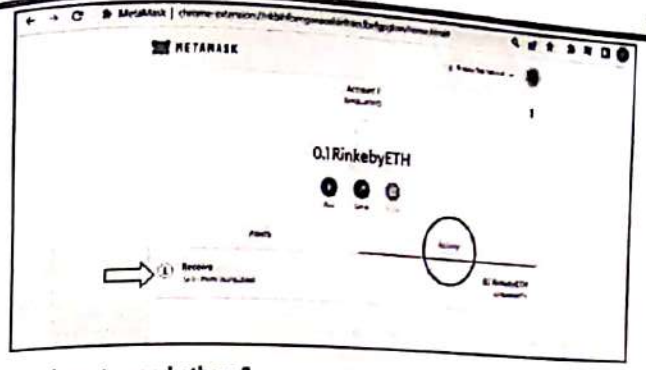


And we are getting message Request complete. Click on Close.

Now will go to Metamask account and we can see, I have 0.1 ETH in my account.



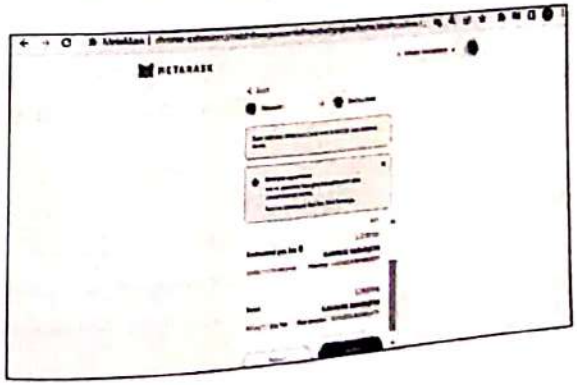
If we on click Activity tab, it will show our transaction history.



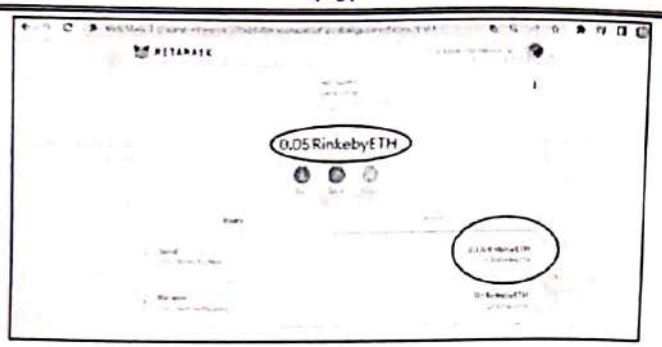
Now we will discuss how to send ethers ?

Click on Send button.

Put sender public address and the amount of ether to transfer and click on Next.



It will show gas fee and click on Confirm.



We can see 0.05 ETH transferred to the mentioned address.

4.11 Mist Wallet

- Mist is sometimes known as a wallet, a phrase derived from Bitcoin jargon.
- The Ethereum Mist wallet is Ethereum's official blockchain wallet where we can hold our Ethereum assets.
- The Ethereum mist wallet was launched in 2017.
- This wallet was created by the team behind the Ethereum foundation so that it can run decentralized applications and blockchain applications.
- The mist is a blockchain-based browser that serves as a platform for operating blockchain networks and decentralized applications.
- Mist wallet is compatible with Linux, Windows, and macOS.
- This wallet saves your private key on your computer and not on third party servers like some wallets.
- The wallet can also be used to hold other tokens issued on the Ethereum platform. In addition, it supports over 16 languages. This makes it a go-to-wallet for users from non-English speaking places
- Currently the Ethereum wallet is only available for desktop.
- Unlike many other wallets, the Ethereum Mist Wallet can be a bit challenging to use. Users can choose from alternatives such as Trust Wallet. These wallets are close competitors that are even available on mobile. Other competitors include Metamask and MyCrypto.
- This wallet has been discontinued from March 2019.

4.12 Ethereum Frameworks

- Frameworks may be used to make the creation of Ethereum smart contracts easier.
- The frameworks discussed in this section can automate some processes and simplify development.

1. Truffle

- For blockchains that use the Ethereum Virtual Machine (EVM), Truffle provides a top-tier development environment, testing framework, and asset pipeline.
- It's a platform for developing decentralized apps on the Ethereum blockchain.

- It is the most well-known and innovative framework.
- Due to its configuration being user-friendly for beginners, Truffle is likely to be your first contact if you are new to DApp development.
- The majority of other frameworks adopted Truffle's approach.
- Additionally, it is interoperable with other blockchains that support earned value management and compatible EVM blockchains, like Binance Smart Chain, Hyperledger, Polygon, and others.
- The languages Solidity and Vyper are also supported.
  - Website: https://truffleframework.com
  - Documentation: https://truffleframework.com/docs
  - npm package repository: https://www.npmjs.com/package/truffle

Truffle installation :

- Node.js must be set up on your computer before you can start. Installing is simple; just enter the following command in your project :

```
$ npm install -g truffle
```

- Initialize Truffle in your project after that:

```
$ truffle init
```

- If everything goes well, you should observe the following:

```
Starting init...
==============

> Copying project files to
/Users/Desktop/Projects/dapp-truffle

Init successful, sweet!

Try our scaffold commands to get started:
$ truffle create contract YourContractName # scaffold a contract
$ truffle create test YourTestName    # scaffold a test
```

- The project structure is as follows:

```
$ dapp-truffle ls -1
contracts
migrations
test
truffle-config.js
```

The Truffle network

- In addition to acting as a local development blockchain for testing, Truffle provides a development network where DApps may operate and be tested without having to be deployed to the mainnet.
- In the truffle-config.js file, the network object comes with development and advance network configurations:

```
networks: {
 development: {
  host: "127.0.0.1",
  port: 8545,
  network_id: "*", // match any network
  websockets: true
 },
 live: {
  host: "178.25.19.88", // Random IP for example purposes (do not use)
  port: 80,
  network_id: 1,     // Ethereum public network
 }
}
```

- The development configuration is the default network where you run the following command:

```
$ truffle migrate --network
```

- To compile your smart contracts, use the command below:

```
truffle compile
```

2. **Embark**
   - Another framework for the EVM blockchain, known as a full-stack framework, is Embark.
   - This indicates that the framework provides a method for simultaneously developing the front end and back end of a decentralized application.
   - Your frontend (HTML and JavaScript) code and Solidity smart contracts are monitored by Embark for updates and they are then redeployed to the blockchain network.
     - GitHub: https://github.com/embark-framework/embark/
     - Documentation: https://embark.status.im/docs/
     - npm package repository: https://www.npmjs.com/package/embark

Embark installation

- Ensure that Node.js is set up on your machine before installing Embark.
- You must also have the following installed in order to use Embark:
- The Ethereum Client, which offers a blockchain network to test our application on, and

- IPFS, which enables decentralized file storage. Even if they are optional, installing them is still advised.
- It's really simple to install Embark :

```
$ npm -g install embark
```

- Before using Embark, we must first perform the aforementioned command:

```
$ embark demo
$ cd embark_demo
```

- By doing this, a scaffold application is created with every necessary folder
- With that in mind, you launch the programme to launch the IPFS and Ethereum nodes.
- Keep in mind that Embark monitors your code for changes before redeploying:

```
$ embark run
```

3. **OpenZeppelin**
   - A toolkit called OpenZepplin has plugins that make the creation of smart contracts faster
     - **Website :** https://openzeppelin.org/
     - **Documentation :** https://openzeppelin.org/api/docs/open-zeppelin.html
- Any of the OpenZeppelin smart contracts must be imported into your own smart contract before you may utilize them.
- OpenZeppelin is an open framework for safe, reusable smart contracts written in the Solidity programming language.
- Node.js has to be installed first.

OpenZeppelin installation

- To install OpenZeppelin, use the following command :

```
$ npm install @openzeppelin/cli
```

- To set up an OpenZeppelin project, use the following :

```
$ npx openzeppelin init
```

- We must build the following code using an example smart contract from OpenZeppelin's documentation, which is shown below:

```
// contracts/Counter.sol
pragma solidity ^0.5.0;

contract Counter {
 uint256 public value;

 function increase() public {
  value++;
 }
}
```

- To compile, run the following :

```
$ oz compile
```

The OpenZeppelin network

- Our smart contract's deployment is as simple as their compilation;
- Ganache is a network used by OpenZeppelin where a local blockchain may be created.
- If you don't already have it, you should install Ganache:

```
$ npm install ganache-cli
```

- Run the following commands to launch a new process:

```
$ npx ganache-cli --deterministic
```

- Then, deploy the smart contracts :

```
$ npx oz deploy
✓ Compiled contracts with solc 0.5.9 (commit.e560f70d)
? Choose the kind of deployment: upgradeable
? Pick a network: development
? Pick a contract to instantiate: Counter
✓ Added contract Counter
✓ Contract Counter deployed
? Call a function to initialize the instance after creating it?: No
✓ Setting everything up to create contract instances
✓ Instance created at 0xCfEB869F69431e42cdB54A4F4f105C19C080A601
```

- The best framework to choose for your DApp project is to understand what you want to build and what necessary toolkit you need.
  1. **Brownie** - Python-based development environment and testing framework.
     Documentation - https://eth-brownie.readthedocs.io/en/latest/

4.13 Case study of Ganache for Ethereum Blockchain

Ganache : A Local Test Blockchain

- Ganache is a public Ethereum blockchain that you can use to perform tests, send orders, and check the chain's state while keeping complete control over it.
- Ganache is a local test blockchain on which we may deploy contracts, create apps, and perform tests.
- It is offered for Windows, macOS, and Linux as a desktop programme (with a graphical user interface).
- The ganache-cli command-line tool is another way to access it.
  - Ganache desktop application, see https://trufflesuite.com/ganache.
  - The ganache-cli code can be found at https://github.com/trufflesuite/ganache-cli/.

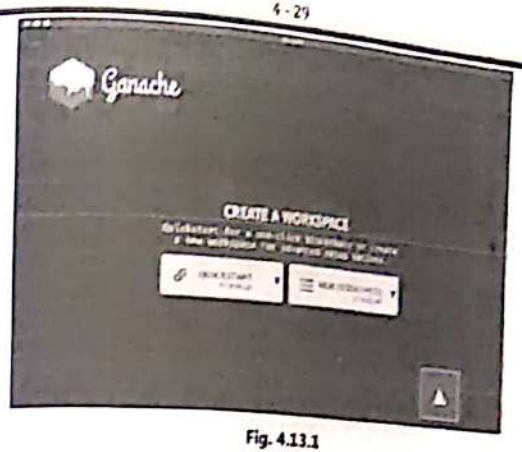


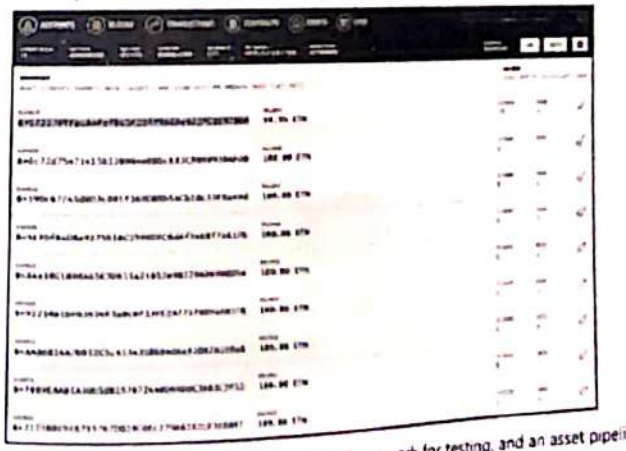
Fig. 4.13.1

- After compilation, launch GANACHE from your desktop, which you got from the link provided above. The screen will appear as seen below. Select QUICKSTART Ethereum-

Case study – To develop a smart contract for Pizza Business

Following Frameworks, Technologies, and Tools will be required to develop an Ethereum smart contract to manage a pizza store.

- Solidity - An object-oriented programming language called Solidity is used to create smart contracts.
- Ganache - A personal blockchain called Ganache is used to create distributed applications for Ethereum and Corda quickly.
- Ganache may be used throughout the whole development cycle, allowing you to create, distribute, and test your DApps in a secure and predictable setting.



- Truffle- Truffle combines a programming environment, a framework for testing, and an asset pipeline.
- It is built on the Ethereum Blockchain and intended to make the building of DApps simple and straightforward.
- Web3.js - Web3.js is a library collection that allows us to interface with a local or distant Ethereum node through HTTP.

The Pizza Store

- There are following components in this application.
  - **Smart contract** - Provides immutable contracts that are secure and strong.
  - **Customers** - Can orders pizza
  - **Pizza store** - deploys and owns the contract
- There are several functions implemented.
  - Send_order, Check_order, Send_Invoice, Order_delivered and the events triggered are Invoice_sent, Order_sent, Order_delivered
- First and foremost, let us define variables and structs. Essentially, we need pizza store and customer addresses, as well as order and invoice structs.
- We require order and invoice lists, thus we utilise mapping in Solidity.
- Write the events are triggered for the activities.
- Write the functions to provide order delivery, safe payment and get invoice.
- After creating the project and launching ganache, which offers a local Ethereum blockchain network, smart contract can be deployed using truffle with the following command :
  - truffle migrate –reset
  - truffle test

4.14 Exploring Etherscan.io, and Ether Block Structure

4.14.1 Etherscan.io

- As we discovered in earlier lessons, blockchain transactions are transparent, anybody can view them and/or confirm them, and everyone may check trade history.
- But how does one really go about doing all of this?
- Thanks to Etherscan, any Ethereum network users may quickly learn all of this information and more.

What is Etherscan and how does it work ?

- For Ethereum, a decentralised platform for smart contracts, Etherscan is the top Blockchain Explorer, Search, API, and Analytics Platform.

ETHERSCAN
The Ethereum Block Explorer

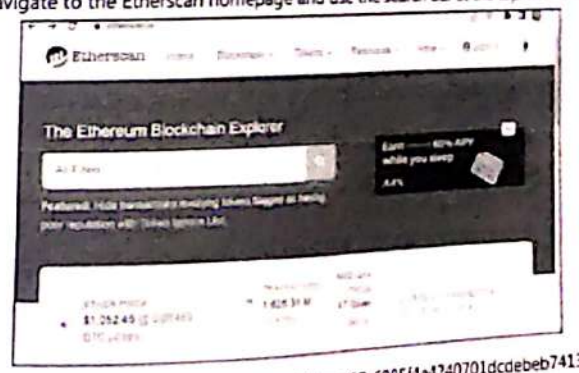
- It is built and launched in 2015
- We may search through transactions, blocks, wallet addresses, smart contracts, and other on-chain data using the etherscan.io website.
- To identify and verify Ethereum blockchain transactions, it effectively serves as a search engine.

- Additionally, it can help in the communication with smart contracts and the identification of any suspicious conduct.
- In short, Etherscan.io monitors transactions on the Ethereum network, indexes them, and makes them available on the website.
- Several people came together to develop Etherscan.io as a totally free resource to help the general public learn more about Ethereum.
- It is independent of the Ethereum Foundation and is run independently.
- "Our objective is to enable Blockchain transparency by indexing and making searchable all Ethereum Blockchain transactions in the most transparent and accessible way possible," the team explains.
- It's important to understand that Etherscan is a data source rather than a wallet.
- One of the most well-liked tools for learning how to use the fundamental blockchain features is Etherscan. Most of the features of this website can be used without registering or connecting to our wallet.
- For instance, using a hash, we can confirm any activity associated with this transaction, whether it involves smart contracts, tokens, or anything else.
- We don't need to register for this, but it could be required to do so if a user wants to access certain further features.

Then, how does it manage to do all of this ?

This tool has several functions.

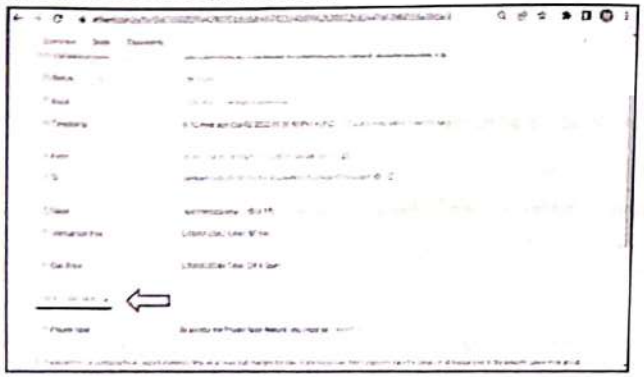
1. **Finding a transaction or wallet is one of Etherscan's primary purposes.**
   - Think about sending 0.0315950222 ETH ($33.28) to the public address 0x3f1d77a0d99e1ea4f3c5330fd2ac1449af27e132. Obviously, we have also paid a transaction charge of 0.0005132052 Ether ($0.54) in addition to paying.
   - Once the transaction is done, our wallet will show us the hash 0x87c6085f4e4240701dcdebeb74133404996282887 2b82ee7a63bf801da38dac3
   - Let's now say that we want to check how many confirmations our transaction has received in order to determine whether it was completed successfully.
   - To begin, navigate to the Etherscan homepage and use the search bar at the top of the screen.



- We enter the transaction's hash there, which would be x87c6085f4e4240701dcdebeb74133404996282887 2b82ee7a63bf801da38dac3 and then click on the search button.

- After completing these few steps, we will be able to see all the specifics of the transaction we just made.

- We'll observe that there is a "see more" option available.

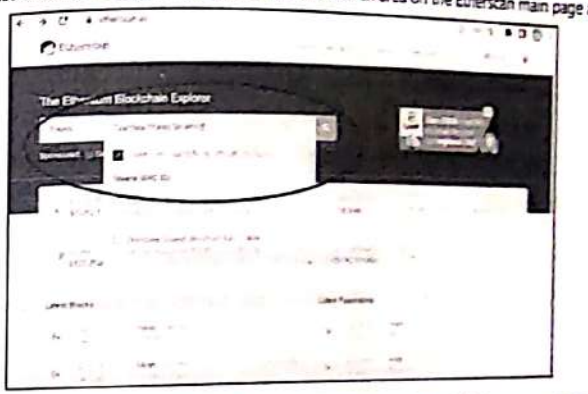


- By choosing that button, we can view further information about the transaction, including its hash, status, block, timestamp, wallet from which it was produced, wallet to which it was sent, transaction value, commission paid, gas price, and kind of TXN.

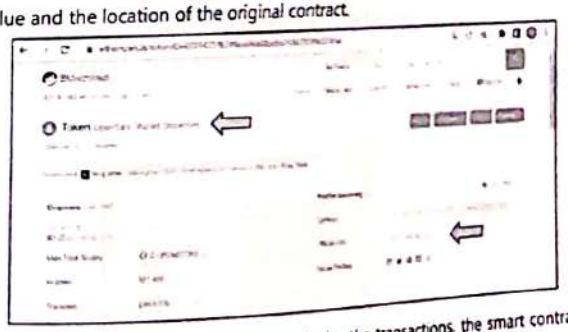
- It will be quite simple to confirm the transaction's status with the use of this data.
- If we still want to look into something more, the receiving address may be used to examine all transactions connected to that address.

How can I use Etherscan to look for smart contracts ?

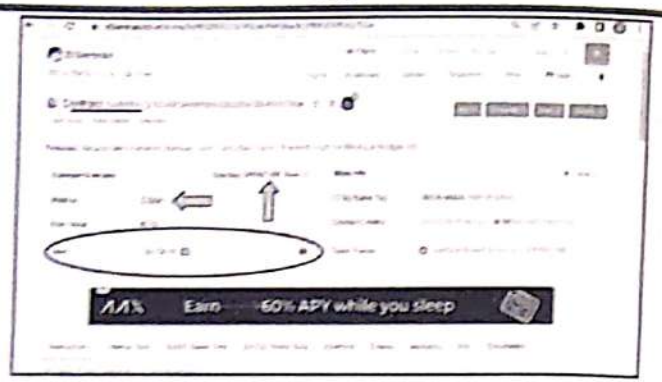
- Knowing that Etherscan is a useful tool to analyse smart contracts is important if you are a user who often engages with decentralised applications.
- In this manner, you may ensure that your money is sent to the appropriate smart contract.
- Suppose you need to transfer tokens to your wallet and you need to locate the token's contract address.
- Token logic in smart contract addresses involves, among other things, how transfers are carried out.
- Finding the address of the token in question is the first step. You may find this address at https://etherscan.io/tokens or on the project's official website.
- The token address should be copied and pasted into the search area on the Etherscan main page as a first step.



- After clicking the search button, we will see the contract's specifics, such as the balance, as well as more basic facts, including the value and the location of the original contract.



- The "see more" option is available as usual, and it includes the transactions, the smart contract, and a comments area. The "contract" option may be seen if we check up above.
- There are choices for interacting with the contract and its regulations. The general details of the contract are available in the "see contract" box.

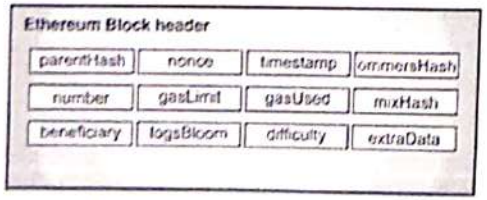


- We may access choices like checking the balance of particular wallets that enable us to utilize that token.
- This is a highly comprehensive tool to analyze the Ethereum blockchain, as we can see.
- In addition to the aforementioned features, the network allows us to search for airdrops, check gas prices, and do countless other operations.
- Also available is token analytics, if that's what you're looking for. Etherscan displays the most popular tokens for each standard, since Ethereum supports both ERC-20 and ERC-721. In addition, all Ethereum smart contracts with validated source code are included in the Verified Contracts section.

4.14.2 Ether Block Structure

- The Block is the basic building block of a blockchain that stores transactions.
- Every block in a blockchain will have a blog header and multiple transactions which we have already covered in previous modules.
- Ethereum is progressively switching from its current Proof-of-Work consensus process to Proof-of-Stake.
- Since Proof-of-Work consumes too much power, it is not environmentally beneficial.
- Proof-of-Stake is quicker than Proof-of-Work and may mine blocks according to how much money the miner has.
- Each Ethereum block contains a number of significant pieces of information, including the difficulty, timestamp, preceding block hash, and Merkle tree-based root hash.

What's In A Block ?



- **Timestamp :** Timestamp represents the time when block mining ends.
- **Block Number :** counting how many blocks there are. The number increases in steps. A genesis block is zero.
- **Difficulty :** The value of difficulty indicates how challenging it is to locate a hash. The effort required to mine the block.

- **mixHash :** a unique identifier for that block./ A 256-bit hash that, when paired with the nonce, confirms that enough computational work has been completed.
- **ParentHash :** Block hash stored in the previous block.
- **Beneficiary :** Block reward and transaction fee destination address
- **ExtraData :** A 32-byte array containing data related to this block. Can be left empty
- **GasUsed :** Indicates the total quantity of gas utilized in all transactions in this block.
- **GasLimit :** The current maximum amount of gas that may be spent per block.

Review Questions

Q.1 Explain Ethereum and components of Ethereum.

Q.2 What is mining in Ethereum? Explain mining process in detail.

Q.3 Short note on Ethereum Virtual Machine (EVM).

Q.4 Short note on Ethereum transactions.

Q.5 Explain accounts in Ethereum and its types.

Q.6 Differentiate between EOA and Contract accounts.

Q.7 Explain Ethereum architecture.

Q.8 Compare Bitcoin and Ethereum.

Q.9 Short note on test networks in Ethereum.

Q.10 Explain how to transfer Ethers using Metamask.

Q.11 Short note on Mist wallet.

Q.12 Explain different Ethereum frameworks.

Q.13 Explain any one application using Ganache.

□□□

5 PRIVATE BLOCKCHAIN

MODULE - 5

Syllabus

Introduction, Key characteristics, Need of Private Blockchain, Smart Contract in a Private Environment, State Machine Replication, Consensus Algorithms for Private Blockchain –PAXOS and RAFT, Byzantine Faults: Byzantine Fault Tolerant(BFT) and Practical BFT

Introduction to Hyperledger, Tools and Frameworks, Hyperledger Fabric, Comparison between Hyperledger Fabric & Other Technologies

Hyperledger Fabric Architecture, Components of Hyperledger Fabric: MSP, Chain Codes, Transaction Flow, Working of Hyperledger Fabric, Creating Hyperledger Network, Case Study of Supply Chain Management using Hyperledger

5.1 Introduction

- Private blockchains are permissioned.
- Access controls—or more specifically, how you access the blockchain—are the primary distinction between public and private blockchain.
- Because private blockchains are permissioned, a third party or central administrator must approve access before a user can access the blockchain.
- Private blockchains restrict read access and the ability to create new transactions to a limited number of users or nodes.
- In private blockchains, a single organization will be in charge of the network and be able to control who can access and join it.
- It is, in essence, a partially decentralized system.
- Bitcoin, Ethereum, or Litecoin are common examples of native currencies on public blockchains. It acts as a vital component of the system that rewards individuals for giving their approval. Private blockchains, on the other hand, usually do not.
- In comparison to public blockchain, there are more transactions per second.

5.2 Key Characteristics

- Complete privacy: Private blockchain concentrates on privacy issues.
- Private Blockchains tend to be more centralized.
- **Improved Scalability** - The flexibility to add nodes and services on demand can be a huge benefit to the organization.

Scanned with CamScanner

- **High performance and quick transactions** - When the nodes are distributed locally but there are fewer nodes participating in the ledger, the performance is quicker.
- Private blockchains are easily scalable and can process thousands of transactions per second.
- **Robust architecture** - In practice, private networks currently have one of the most stable network structures. They are designed to withstand any problems. They therefore include a high level of security mechanisms that aid in preventing malicious activity.

5.3 Need of Private Blockchain

- Private Blockchains are intended for businesses that want greater control over their data and greater privacy through fine-grained permission control.
- Furthermore, because they are designed differently, they do not have the concept of gas fees.
- Businesses interested in these solutions, which we can refer to as network participants, frequently agree to host Blockchain on their own private infrastructure.
- If we want to establish channels of communication between a few parties who do not trust one another and who need to verify one another's transactions, we need private blockchain concepts.
- When a lot of untrusted parties need to work together to solve a problem, private blockchain is very helpful.
- Private blockchains are undoubtedly the technology for you if you're interested in a field where confidentiality is of utmost importance.
- It saves a lot of resources.A private system actually requires very little attention and is relatively easy to maintain. As a result, they only consume a small portion of the resources available on public blockchain platforms. You can therefore reduce the amount of resources (both financial and human) required for these solutions.
- It empowers enterprise companies.
- It offers very low transactional fees.In most cases, a fee may not even be necessary at all. In actuality, this is a superb justification for beginning with private blockchains. Furthermore, in many cases, these blockchains do not have a native token for the network.
- It provides network regulation. Actually, without a regulatory system, enterprise companies cannot function.Therefore, there should be a proper way to carry out specific tasks, and breaking the rules will have negative effects. As a result, it guarantees the participants greater security.

5.4 Smart Contract in a Private Environment

- With a smart contract, the terms of the agreement between the parties are expressed directly in code or programmable logic using a general-purpose, high-level programming language rather than on paper.
- Similar to how bitcoin scripts are executed, these types of contracts are executed more easily in a blockchain system.
- Nevertheless, the bitcoin script was easy to understand in comparison to the smart contract scripts.

5.5 State Machine Replication

- Before discussing state machine replication, understand why it is neede ?
- We know single server architecture.

Scanned with CamScanner

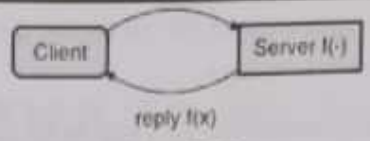


Fig. 5.5.1

- But there can be a single point failure.
- Here comes the need of state machine replication.

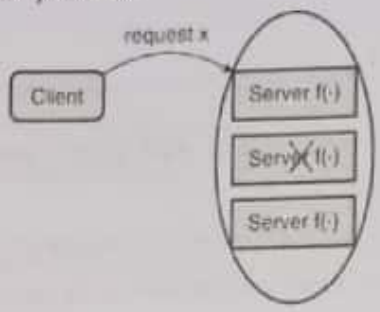


Fig. 5.5.2

- State machine replication gives safety and liveness by creating replicas of state of servers.
- It is an interactive protocol among servers.
- Replicas maintain the same state
  - replicas start in the same state.
  - Replicas execute operations in the same order.
- Replicas send replies to clients.
- Clients vote on replica replies.
- **The replication of state machines assists us in reaching a consensus in a permission model.**
- A smart contract does not have to be executed on each node.
- Instead, the chosen subset of the contract executor executes it and propagates it with further nodes to make sure that all nodes in the network are informed of the contract's state consistently and in real time.
- Consensus is ensured in a permission blockchain context via the distributed state machine replication technique.

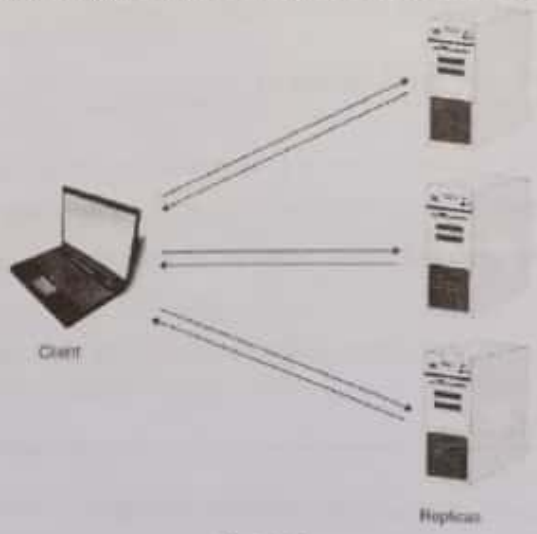


Fig. 5.5.3

Scanned with CamScanner

- What is a state machine ?
  - A state machine executes a sequence of commands.
  - Transforms its state and may produce some output.
- Outputs of the state machines are solely determined by the initial state and by sequence of commands that it has executed.

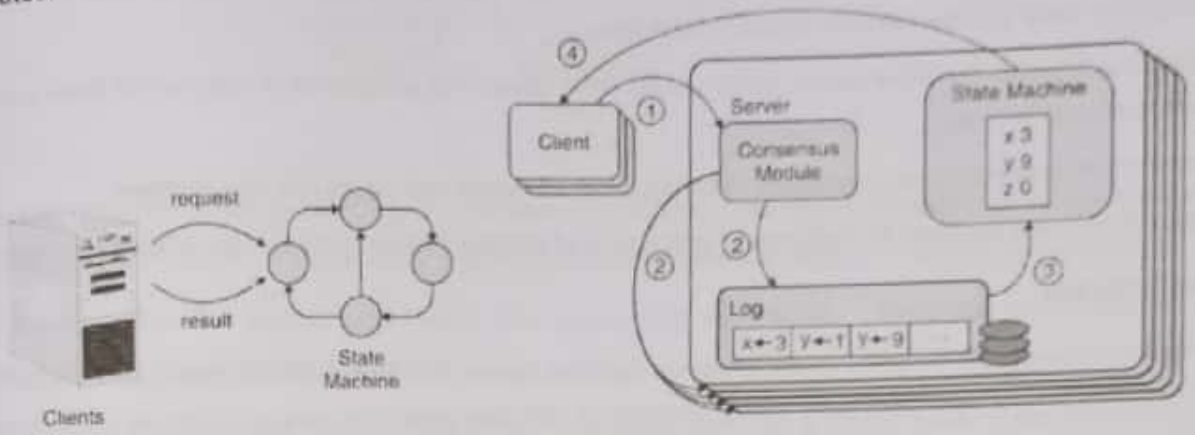


Fig. 5.5.4

- Typically, replicated logs are used to create replicated state machines.
- Upon receiving commands from clients, the server's consensus module adds them to its log.
- Whenever commands have been correctly replicated, the state machine on each server executes them in log order.
- Let us look at a scenario of a replicated state machine.
- We have a number of clients, and services are replicated into three machines. So the replication degree is three.
- We have a consensus module, and when a client wants to issue a command, it sends a command to one of the state machines. The Consensus Module makes an attempt to add these commands to a log.

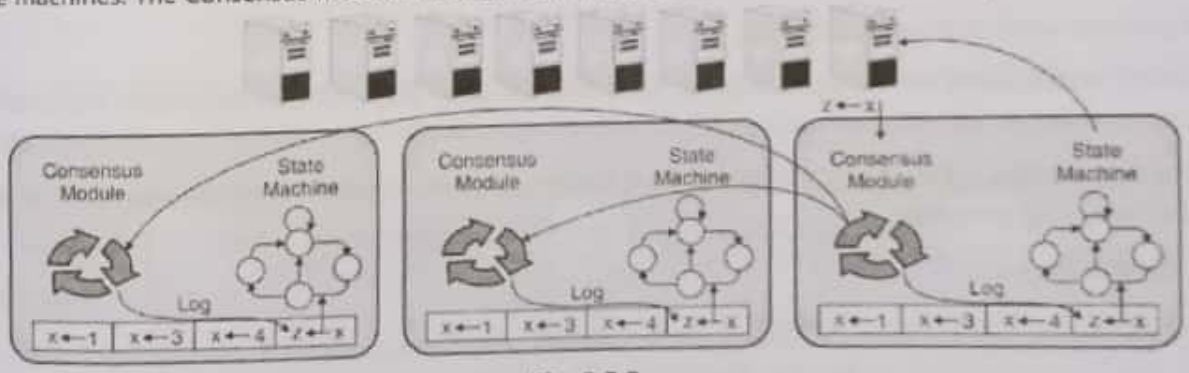


Fig. 5.5.5

- Replicated log ensures state machines execute the same commands in the same order.
- Consensus module guarantees agreement on command sequence in the replicated log.

Summary :

a) Implement service as a deterministic state machine

b) Replicate

c) Provide all replicas with the same input

Scanned with CamScanner

5.6 Consensus Algorithms for Private Blockchain –PAXOS and RAFT

5.6.1 PAXOS Consensus Algorithm

- L. Lamport introduced it as the first consensus algorithm in 1989.
- The goal was to select a single value under the network or crash problems. Paxos lets all nodes agree on the same value despite node failures, network failures and delays.
- By delivering them to the other nodes, certain nodes offer values. The acceptance or rejection of those values must be decided by each node.
- But concurrent processes and uncertainty of timing, order of events and inputs can be a problem
- And also failure and recovery of machines or processors, of communication channels will be a major concern.

Consensus Problem :

- Consensus is impossible to solve in asynchronous systems means it is impossible to distinguish a failed process from one that is just very very slow. Hence the rest of the alive processes may stay unsure when it comes to deciding.
- But consensus is important since it maps to many important distributed computing problems. So can we solve the consensus problem ?
- Paxos does solve the consensus problem. It is the most popular consensus solving algorithm.
- The only entirely safe and mostly live agreement protocol known.
- Paxos is a family of distributed algorithms used to reach consensus.
- We know consensus is agreeing on one result. Once a majority agrees on a proposal, that is the consensus. The reached consensus can be eventually known by everyone.
- Paxos provides safety and eventual liveness.
  - Safety- consensus is not violated. Only the value that has been proposed may be chosen. And also no two correct nodes choose different values.
  - Eventual liveness - If things go well sometime in the future, there is a good chance consensus will be reached. But there is no guarantee.
- Paxos General approach
  - A node or more nodes select a leader.
  - An idea is put out by the leader, who then seeks support from others.
  - Leader either declares the outcome or tries again.

Paxos : Types of nodes

- Paxos defines three roles: proposers, acceptors, and learners.
- Proposer, who proposes values to reach consensus.

Scanned with CamScanner

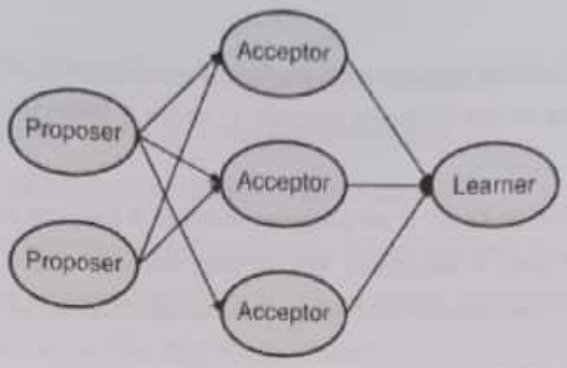


Fig. 5.6.1

- Acceptors, who contribute to reaching the consensus itself; they form the consensus and accept the values. The acceptors have the option of accepting or rejecting a proposal whenever they hear it from the proposer.
- Learners, who learn the agreed upon value. This learner will identify which value has been selected by each acceptor and agree to accept it as a group.
- In practice, Paxos nodes can take multiple roles, even all of them. So a single node, a single server can send proposals to other nodes, they can contribute to reaching consensus, and they learn the final agreed upon value so that you can query them for it later.
- Example, we are at the college, and following courses, we plan to hang out as a group. Pizzahut and Starbucks are our two options for meeting up with classmates after courses. So, depending on the group vote, we can either go to starbucks or pizza hut after classes, but everyone will go there. There is no centralized leader in this situation. The only method to make a group choice is for a couple of them; in this situation, the maximum number may be the two students to present an alternative i.e Starbucks or Subway. The consensus will ultimately be determined by how many people accept or reject that suggested choice, with the outcome being determined by the final majority count. (two vlu example m, & m2)
- Paxos nodes must know how many acceptors a majority is. Paxos nodes must be persistent; they can't forget what they accepted, even though the communication channel may be faulty.
- A Paxos run aims at reaching a single consensus. Once a consensus is reached, it cannot progress to another consensus. In order to reach another consensus, a different Paxos run must happen.

Paxos Algorithm

- We have a Proposer and three acceptors, so the majority of the acceptors would be two. Each node runs as a proposer, acceptor, and learner.
- Paxos has rounds and each round has a unique ballot ID.
- These rounds are asynchronous. If anybody is in round 1, and hear a message from round 2, abort everything and go to round 2.
- Each round has 3 phases

Phase 1 : A leader is elected (election) propose

Phase 2 : Leader proposes a value (bill), processes acknowledgements

Phase 3 : Leaders multicast final value (based on consensus)

Scanned with CamScanner

Phase 1 :

- Proposer proposes a value and solicits acceptance from acceptors. A specific value is what the proposer intends to propose. It notifies all Acceptors of the proposal number n, which is PREPARE ID. The proposal number must be unique.
- Acceptors respond to highest PREPARE Id. If an acceptor receives a prepare request with number n greater than that of any prepare request it saw, it responses YES to that request with a promise not to accept any more proposals numbered less than n and include the highest-numbered proposal (if any) that it has accepted.

  <prepare ,n>

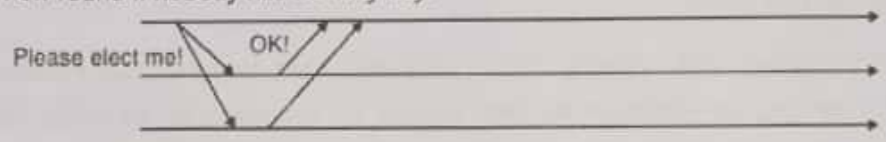
  if n< np

  reply prepare reject (as already saw a higher proposal number), where np is previous proposal

  else

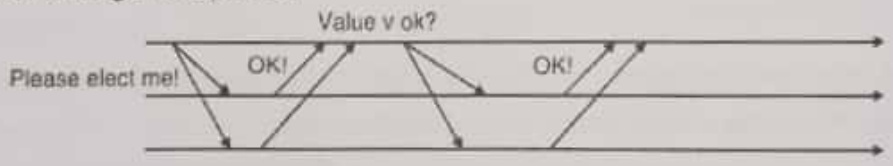
  np = n

  reply prepare accept , (this node will not accept any lower than n proposal.
- Acceptors record the PREPARE ID they got on disc as well. Node is the leader if the majority of responses are positive. Start a new round if nobody has the majority.



Phase 2 :

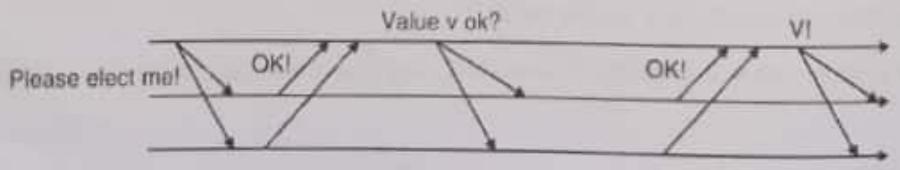
- If the majority of the acceptors respond YES to the proposer's prepare requests, the proposer will issue accept requests for each of those acceptors for proposals with numbers n and v, where v is the value of the proposal with the largest number among the responses.



- Recipient log on disk and responds OK.
- Delay the procedure and restart Paxos if the leader is unable to get the support of the majority.

Phase 3 :

- If leader hears OK from majority, it lets everyone know of the decision (learning).
  - send (decide, v) to all nodes



- Recipient receive decisions and log it on disk.
- Delay the procedure and restart Paxos if the leader is unable to get the support of the majority.

Scanned with CamScanner

5.6.2 RAFT

- Raft is a consensus algorithm that is designed to be easy to understand.
- It is designed to be an alternative to Paxos. It is easier to implement.
- Raft need to run on 5 servers to tolerate the failure of two of them

Raft algorithm was developed by Diego Ongaro and John Ousterhout at Stanford University. Raft was designed to have a better understanding about Consensus to be achieved considering that its predecessor, the Paxos Algorithm, developed by Lesli Lamport is very tough to understand and implement. The RAFT algorithm is similar to pBFT but in RAFT only leader node can establish communication with other nodes and takes decision about the state of the transaction. Raft algorithm divides time into small terms of random length. Each term is identified by a monotonically increasing number, called term number. Each node exists in one of the three states: leader, follower and candidate. Raft divides consensus into three phases :

- **Leader Election :** If existing leader fails then a new leader needs to be elected.
- **Log replication :** The leader replicates log to other nodes so that all the copies will be up-to-date.
- **Safety :** If one of the nodes has committed a log entry at a particular index, no other node can apply a different log entry for that index.

- Initially, there is just one leader and all of the other nodes are followers. Followers are passive and only reply to requests from leaders and candidates. The leader handles all client requests. If a client communicates with follower, the follower redirects it to the leader. Candidate state is used to elect a new leader.
- Randomized timer is used to elect the leader in each term. If leader is not elected in a term, candidates will time out and initiates the election for the next term. Leader, candidates log should be most up-to-date compared to log at followers. If candidate's log is less up-to-date compared to potential follower, then follower rejects the candidate.
- Node begins as a follower and if it does not receive heartbeat message from leader within election time then it assumes that the leader is dead. It then takes candidate state to send out a "RequestVote". If candidate node gets majority approvals from follower nodes, then it becomes leader.
- Only leader can append log entries as per client requests. After receiving request from client, leader appends entry to its log. This new entry then leader sends to all the follower nodes. When majority of followers confirms this message, leader commits the message and sends heartbeat message to client and followers. This consensus mechanism is used in Quorum for consortium setting.

How RAFT works ?

- A RAFT cluster has one and only one elected leader
  - which communicates with client directly
  - which responsible for managing log replication on the other servers of the cluster.
  - which leads until it fails or disconnects, in which case a new leader is elected.

Scanned with CamScanner

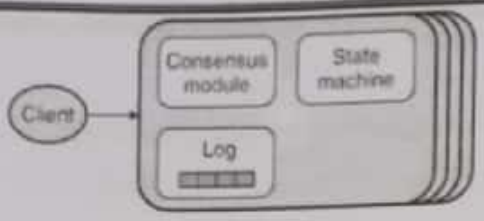


Fig. 5.6.2 : A Raft Cluster

- One leader and several followers and all client requests go through the leader.
- A client sends a request

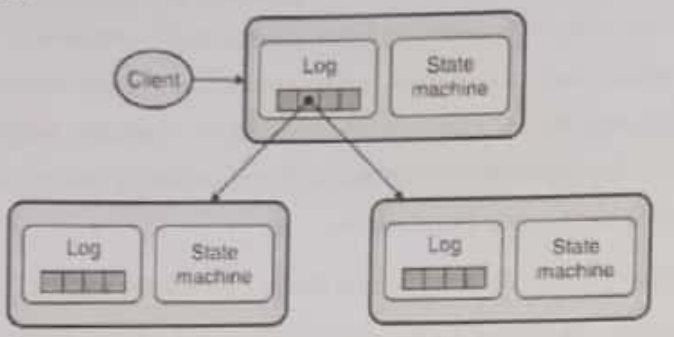


Fig. 5.6.3

- Leader stores requests on its log and forwards it to its followers.
- Followers store the request on their logs and acknowledge its receipt.
- Leader Election
  1. Leader sends "heartbeats" to other nodes saying that it is online : they update their state machines

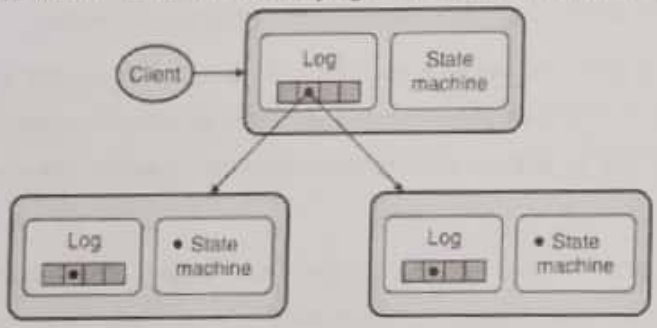


Fig. 5.6.4

  2. If other nodes no longer receive 'heartbeat', they start an election cycle.
  3. Followers decide to elect a new leader.
  4. First candidate to timeout becomes new leader.
- Log Replication
  1. Leader accepts client requests and appends them to its log.
  2. Leader ensures all other nodes have followed that request i.e leader replicates its log to other servers.

5.7 Byzantine Faults : Byzantine Fault Tolerant (BFT) and Practical BFT

5.7.1 Byzantine Fault Tolerant (BFT)

- Consensus achievement is the process by which network participants regularly agree on the current state of a specific blockchain.

Scanned with CamScanner

- Over distributed networks, it is possible to reach consensus, but it is not an easy task. This issue gave rise to the concept of Byzantine fault tolerance.
- Lamport, Pease, and Shostak proposed the Byzantine Generals Problem in 1982 to guarantee that malfunctioning components providing conflict information are managed inside a system.

What is the Byzantine Fault Problem / Byzantine Generals Problem?

- This problem represents an issue of computer networking over an unreliable link.
- A theoretical problem known as The Byzantine Generals' Problem illustrates how several Byzantine generals might find it difficult to communicate while deciding on their next course of action.
- The Byzantine Generals problem demonstrates the value of blockchains and explains why they are important. It will aid in understanding why a blockchain operates as it does.
- First we will discuss the unsolvable Two Generals Problem. Following that, we'll expand on that to the Byzantine Generals' Problem and talk about Byzantine Fault Tolerance in distributed and decentralized systems.

Two Generals Problem

- This problem was first published in 1975.
- In this problem we will call these two nodes as generals. In this situation the generals in their armies have come upon a city that they wish to attack.
- Each general's army on its own is not enough to defeat the enemy's army successfully, thus they need to cooperate with an attack at the same time the problem arises because two generals are separated by a significant distance.
- So think of this as if they are like on opposite sides of the city.
- General 1 is considered as a Leader and the other is considered the Follower. In order for them to communicate and decide on a time of attack, General 1 has to send a messenger across the enemy's camp that will deliver the time of attack to General 2.
- But this could mean that the messenger could get captured and the message would be lost forever.
- This will result in General 1 attacking while General 2 and his army hold their grounds assuming the first message goes through.
- General 2 must still acknowledge that he received a message so he sends a messenger back.
- Thus repeating the previous scenario where the messenger can get caught.
- This opens up an infinite loop where both parties are sending acknowledgements back and forth with neither General knowing whether or not the message was received.
- If General 1 receives a message how General 2 will know that the message was received without an additional acknowledgement message.
- General 1 presents a repeating problem because General's problem is currently unsolvable.
- And still a newer problem was iterated off of this one, which is Byzantine Generals Problem.

Byzantine Generals Problem

- We have already learned about this problem in Module 1.

Scanned with CamScanner

- This problem was introduced in 1982 in a paper written by Leslie Lamport, Robert Shostak and Marshall Pease which were a group of researchers.
- It's a special case of the Two Generals Problem that has been generalized.
- Similar to the Two Generals problem, the Byzantine Generals problem deals with reliable computer systems handling malfunctioning components that give conflicting information to different parts of the system to ensure consensus within the distributed computing system.
- The difference now is that more than two generals need to agree on a time to attack their common enemy.
- The same situation is described, but this time, more than two generals are required to agree on when to launch an attack against their common enemies.
- One or more of the generals may be a traitor in this situation, which would allow them to lie about their decision, adding another layer of complexity (e.g. they say that they agree to attack at 08.00 but instead they do not).
- An arrangement of commander and lieutenant replaces the leader-follower paradigm described in the Two Generals Problem.
- The commander and each lieutenant must reach consensus here in order for the action to be taken.
- The issues that could develop when communicating a decision from one general to another are while conveying the message, the messenger could be arrested; What if the message was altered by a forger? ; How can a general be certain that he received the message from the expecting general? ; What if other generals turn traitor and tell their troops to strike but then they flee?
- How can the system be confident that every general will launch a simultaneous attack from their predetermined location? Is there any other option but to have complete trust in one another?
- With its Byzantine fault tolerance (BFT) consensus method, blockchain appears to solve this issue.

Byzantine Fault Tolerance (BFT) Algorithm

- BFT algorithm is based on state machine replication.
- In distributed systems, ensuring that computers can establish consensus on specific facts in a reliable manner is a very real difficulty that byzantine fault tolerance seeks to address.
- An method that resists a system to enter the Byzantine Generals' problem is known as Byzantine Fault Tolerance (BFT).
- BFT system is able to continue operating even if some of the nodes fails communicate who acts maliciously.
- The general's team needs an algorithm that can follow the following guidelines in order to guarantee the success of the generals' team: The next action of the plan must be approved by all troop generals ; the generals must be dependable and committed to the establishment ; the impact of network traitors on generals must be avoided ; they must adhere to the system's algorithm ; no matter what the traitors do, the generals must agree on something or make a choice ; at no point during the action, the system or network should result in a 51 percent attack.
- The basic operation of a BFT algorithm is :
  - **Phase 1 :** Client sends a request to the primary. The primary can then validate the message and propose a sequence number for it.

Scanned with CamScanner

- **Phase 2 :** Primary node sends pre-prepare message to all backup nodes. This allows the backup nodes to validate the message and receive the sequence number.
- **Phase 3 :** All functioning backup nodes sends prepare message to all other backup nodes. This allows replicas to agree on a total ordering.

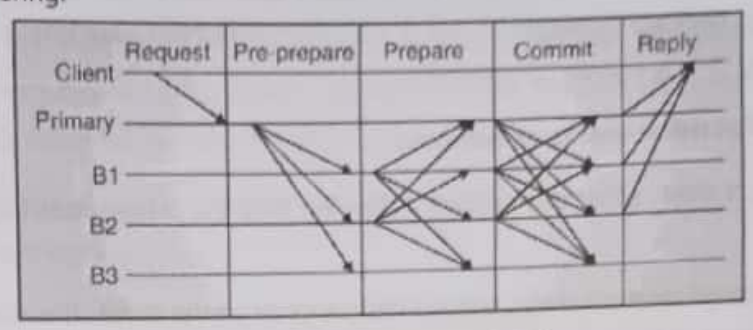


Fig. 5.7.1 : Working of BFT

- **Phase 4 :** All replicas multicast a commit. The replicas have agreed on an ordering and have acknowledged the receipt of the request.
- **Phase 5 :** Each functioning replica sends a reply directly to the client. This bypasses the case where the primary fails between the request and reply.

- The amount of network communications among participants in the BFT-based consensus method grows as the number of participants rises. This is so because each transaction's processing requires participation from every participant in the BFT-based consensus method as shown in Fig. 5.7.1.1, and the process itself is made up of a variety of steps. Due to the fact that the completion of the consensus process slows down as more participants join, this feature of the BFT-based consensus method may cause performance and scalability issues.
- Limitations of BFT :
  - If malicious clients have the necessary access, they may corrupt the system state.
  - No consideration is given to fault-tolerant privacy.

5.7.2 Practical BFT

- Another famous consensus algorithm is Practical Byzantine Fault Tolerance or PBFT.
- It was published by Miguel Castro and Barbara Liskov in 1999, which is the solution to the Byzantine General problems.
- Paxos and Raft are not by default Byzantine fault tolerant, though there are variants of them like BFT-Paxos,
- It is a blockchain consensus mechanism which is widely used presently, but the confidence of blockchain node in PBFT cannot be guaranteed. It requires large amount of communication resources in the process of reaching consensus.
- Byzantine general problem. In this case, all general nodes are at equal level and gets their work directions from the leader node. The leader node is primary node and all other nodes are considered as secondary or backup nodes. Selection of leader is randomly in round-robin manner. Client node sends transaction request to leader node. This received request is then broadcasted by leader node to all the secondary/backup nodes.

Scanned with CamScanner

- Both leader and secondary nodes use the request message with their internal state to run computation and broadcast the decision result to all client nodes. Majority of nodes agreed upon decision is now considered as final decision.
- Suppose f are maximum number of faulty nodes. Practical Byzantine Fault Tolerance consensus algorithm relies on minimum number of back up nodes which are (f + 1). Therefore, high hash rate is not required. This ensures less computation leading to less energy consumption.

The PBFT consensus algorithm works as follows :

- To the leader node, a client submits a request. Following that, the leader node broadcasts requests to all backup nodes.
- The request is processed by each node, which then responds to the client. The client watches for responses with the same outcome from all nodes in the amount of (f + 1). Here, f equals the number of potential defective nodes.
- The main PBFT algorithm consists of 3 phases
  - pre-prepare
  - prepare
  - commit
- Pre-prepare phase : The leader node notifies each backup node by sending a pre-written message.
- Prepare phase : The backup nodes the prepared message as a reply to all other nodes, including the leader, after receiving the message that has been pre-prepared from the leader. **Only after receiving the leader's pre-prepared messages and viewing (2f + 1) number of other nodes' prepared messages can a node be said to b e prepared**
- Commit phase : The nodes send a commit message if they are ready. A node executes the client's request if it receives (f + 1) commit messages.

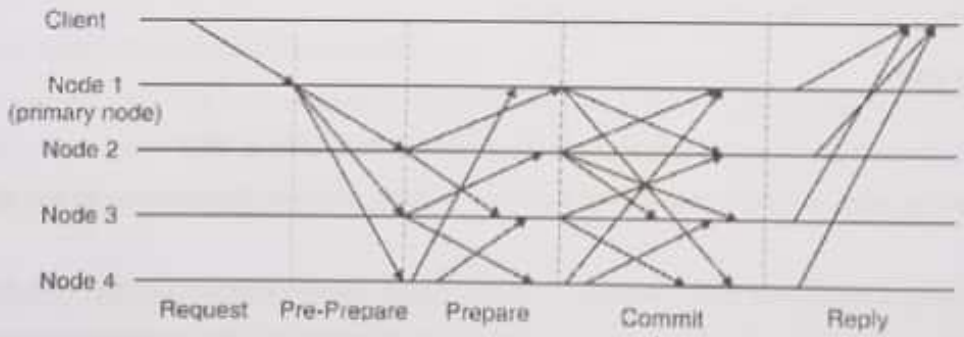


Fig. 5.7.2 : PBFT Algorithm's Operation

Limitations of PBFT Consensus Mechanism :

- PBFT mechanism has complex communication and poor scalability. When the number of consensus nodes in the Blockchain network is large enough, the functionality will be degraded. When there's a great deal of network partition, consensus efficiency will be compromised which leads to a huge delay.
- Better response time can be achieved only with small group sizes,
- Single party can manipulate the network by assuming several identities or nodes.

Scanned with CamScanner

5.8 Introduction to Hyperledger

- In order to improve cross-industry blockchain technology, the LF produced the open source initiative known as Hyperledger.
- "Hyperledger is an open sourced community of communities to benefit an ecosystem of Hyperledger based solution providers and users focused on blockchain related use cases that will work across a variety of industrial sectors." – Brian Behlendorf, Executive Director of Hyperledger
- Hyperledger is neither a Blockchain nor a company, nor is it a cryptocurrency. It is a piece of software used to build one's own custom blockchain service.
- A cross-industry open source collaborative project called Hyperledger was developed to advance blockchain technologies.
- Global leaders in banking, supply chains, technology, IOT, and finance are participating in the collaboration with Linux Foundation, which is also being hosted by the Linux Foundation.
- According to the Hyperledger philosophy, multiple private chains will manage multiple markets around the world.
- Since each business is distinct in its own right, applications that cater to these businesses should be created using individualized rules.
- Unlike Ethereum, which frequently forces programmers to create their applications using generic protocols.
- In late 2015, a small group of developers launched the Hyperledger project.
- These developers were from a range of industries, including data science, manufacturing, banking, etc., and they shared the objective of making blockchain technology more usable to developers and enterprises.

Why is Hyperledger necessary ?

- Public blockchains do not support private and confidential transactions.
- Public blockchain is having issues with scalability.
- The developers discovered during extensive testing that blockchain networks, where each peer must simultaneously run consensus and validate each and every transaction, suffer greatly in terms of scalability.
- On top of that, due to the extensive measures that are taken to ensure a transaction's integrity, transactions with a degree of confidentiality and privacy attached to them cannot be executed on public blockchains.

5.9 Tools and Frameworks

5.9.1 Hyperledger Frameworks

There are five blockchain frameworks.

1. **Hyperledger Iroha :** Based on Hyperledger Fabric, Iroha was developed for mobile development projects and contributed by Soramitsu, Hitachi, NTT Data, and Colu. Both a novel chain-based Byzantine fault tolerant consensus method called Sumeragi and cutting-edge, domain-driven C++ architecture are featured in it.

Scanned with CamScanner

2. **Hyperledger Fabric (HLF) :** Developed by IBM, Hyperledger Fabric is intended to serve as a base for the creation of applications or other solutions with a modular architecture. It supports plug-and-play components like consensus and membership services and uses containers to host smart contracts known as chaincode, which represent the system's application logic.
3. **Hyperledger Burrow :** Originally developed as a client-built modular blockchain to the specifications of the Ethereum Virtual Machine, Hyperledger Burrow was first made available by Monax and Intel (EVM).
4. **Indy Hyperledger :** Indy is a Hyperledger project created to provide independent identification on distributed ledgers and was first contributed by the Sovrin Foundation. In order to create digital identities based on blockchains or other distributed ledgers, Hyperledger Indy offers the following tools, libraries, and reusable components:
5. **Hyperledger Sawtooth :** Sawtooth was developed by Intel and incorporates a revolutionary consensus mechanism known as Proof of Elapsed Time that Intel developed (PoET). The distributed consensus goal of PoET is to be attained as quickly as feasible. With support for both permissioned and permissionless deployments and acknowledgement of varied requirements, Hyperledger Sawtooth offers a lot of potential in a lot of domains. Sawtooth is made to be adaptable.

5.9.2 Hyperledger Tools

Currently, the Hyperledger project includes five tools, all of which are hosted by the Linux Foundation.

1. **Hyperledger Composer :** is a collection of collaborative tools for constructing blockchain business networks that accelerates the creation of smart contracts and blockchain applications as well as their deployment over a distributed ledger. Hyperledger composer was developed by IBM and Oxchains.
2. **Hyperledger Caliper :** Currently under incubation, caliper is a blockchain benchmarking tool that enables users to assess the effectiveness of a particular implementation using specified use cases. This one was developed by engineers from various organizations.
3. **Hyperledger Cello :** Hyperledger Cello was first provided by IBM and sponsored by Intel, Huawei, and Soramitsu. For the blockchain ecosystem, it is a widely used blockchain module kit. The effort required to create, maintain, and terminate blockchains is greatly reduced by Cello. Furthermore, because it provides a multi-tenant chain service, this tool can work on top of a variety of infrastructures like virtual machines, bare metal, and container platforms.
4. **Hyperledger Explorer :** Hyperledger Explorer was created to help programmers create interactive web applications. It was first made available by Intel, IBM, and DTCC. Using the tool, users can view, deploy, invoke, and query blocks, transaction data, chaincodes, and other data that is kept on a Blockchain ledger.
5. **Hyperledger Quilt :** A blockchain tool for businesses called Hyperledger Quilt was first made available by NTT Data and Ripple. Ledger system interoperability is made possible by Quilt. The Interledger Protocol (ILP), a payment protocol for transferring value between distributed and non-distributed ledgers, enables it to accomplish this.

5.10 Hyperledger Fabric

- Hyper ledger fabric is a private and permissioned blockchain.

Scanned with CamScanner

- It is a framework to create, deploy and maintain blockchains for businesses that ensure accountability, transparency and trust among business partners as a result.
- It runs smart contracts called "chaincode" within Docker containers.
- The members of Hyperledger fabric network enroll through a trusted MSP.
- Hyper ledger fabric also offers the ability to create channels, allowing a group of participants to create a separate ledger of transmissions.
- Shared ledger : HF has a ledger subsystem comprising two components,

  The world state and transaction log.
- World state component describes the state of the ledger at a given point in time. It is a database of ledgers..
- Transaction log component records all transactions which have resulted in current value of the world state.

Advantages of Hyperledger Fabric

- Identity management via PKI (public key infrastructure)
- Efficient processing through segregation of consensus and chaincode execution
- Privacy and confidentiality via 'channels'

Fig. 5.10.1

5.11 Comparison between Hyperledger Fabric & Other Technologies

Platforms	Ethereum	Hyperledger Fabric	R3 Corda	Ripple	Quorum
Public/Private/ Permissioned	Public Blockchain- No permission is needed to access network content.	Private Blockchain- Network is limited to people with permission.	Private Blockchain- Permission is needed to access network content.	Public Blockchain- Permissionless network content	Both Public, Private permissioned network content
Smart Contract	Solidity programming language	Golang programming language	Kotlin programming language	C++ programming language	Solidity and Vyper
Governance	Carried out by developers (DAO)	Linux foundation incharge	R3 company in charge	Ripple Labs	Etherum developers and JP Morgan Chase

Scanned with CamScanner

Platforms	Ethereum	Hyperledger Fabric	R3 Corda	Ripple	Quorum
Consensus	Relies on Proof of Stake for decision making	The consensus process does not require participation from every node in a network.	Making decisions is limited to those parties engaged in a transaction.	Participating nodes verify the authenticity of a transaction by conducting a poll.	Multiple voting based consensus mechanisms which do away with the current proof of work consensus mechanism used today by the public Ethereum blockchain among many others
Consensus Protocol	PoW/ PoS	Puggable	Notary-based	Probabilistic voting	Pluggable
Currency	Ether	No native currency	No native cryptocurrency	XRP	ETH
Use Case	Popular with generalized applications and mostly used for P2P and B2C operations.	the preferred platform for B2B operations, mainly used in enterprise.	Runs on a specialized distributed platform for the financial industry needs.	Runs on a specialized distributed platform for the financial industry needs.	Popular with generalized applications and mostly used for P2P and B2C operations.

5.12 Hyperledger Fabric Architecture

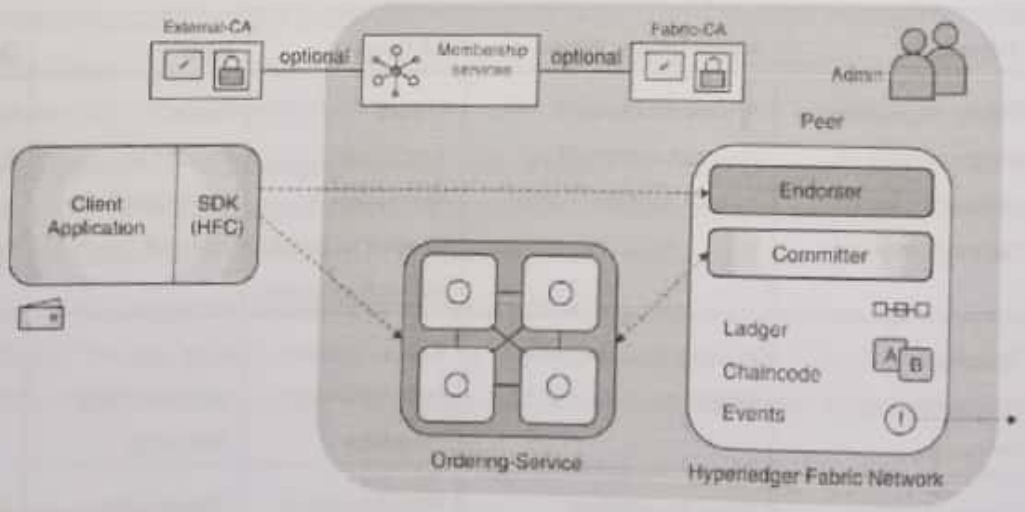


Fig. 5.12.1 : Architecture of Hyperledger Fabric

- The organizations that contribute to the development of the Hyperledger Fabric network are referred to as "members."

Scanned with CamScanner

- Components of Hyperledger Fabric
- Nodes
  - Client
  - Peer
  - Ordering Service
- Fabric CA
- Channel
- Membership Service Provider (MSP)
- Chaincode

Client

- Represents the end user or client applications which trigger blockchain events.
- Clients are software programmes that propose transactions on a network on behalf of a person.
- Clients communicate with both peers and the ordering service.
- The client interacts with the network using a Fabric SDK.
- When reading or writing data from a Fabric blockchain the client interacts with the SDK. To ensure that a legitimate client has started the network transaction, the CA authority issues a certificate to the client as well.

Peer

- nodes participating in the network
- Peer is the place where the ledger and the blockchain data is stored.
- You must have more than one peer in production
- One peer may be part of many channels
- Every single channel is inside the peer
- It endorse any update of the ledger
- You can create a backup of the ledger from the peer.
- All nodes and peers within the blockchain network are equal in public networks like Ethereum.
- All nodes/peers in the Blockchain network are not equal in Hyperledger Fabric.
- The role, behaviour, and tasks carried out by the peers may differ even though the underlying binary used for each peer may be the same (version differences are acceptable).
- The following list includes the typical categories of "roles" that peers are given :

  Endorsing peer, anchor peer, and Ordering peer

Endorsing Peers :

- All peers have the capability to act as an endorsing peer, and will validate transactions.

Scanned with CamScanner

- A unique category of committing peers called endorsing peers has the additional responsibility of endorsing a transaction.
- They give their approval to the client's request for a transaction.
- Each endorsing peer has a ledger and a copy of the smart contract installed.
- The transaction simulation is the primary purpose of the endorser.
- It is carried out in accordance with the smart contract on the client's personal copy of the ledger, and it produces the Read/Write sets that are sent to the client.
- Although the transaction is not recorded in the ledger during simulation.

Anchor Peers :

- We require some peers in order to have communication across an organization because the Fabric network can span multiple ones.
- Not all peers have this ability, but there are some unique peers—the Anchor peer—who are the only ones with this permission.
- Anchor peers are well-known outside of the organization.
- The anchor peers for the participating organizations are encoded in the genesis block at the time the network is first created.
- As a result, let's say Organization-A peers will have access to Organization-B anchor peers' gossip.
- However, because regular peers are unknown to those outside the organization, the non-anchor peer in Organization-B cannot connect with the peer in Organization-A.
- Through peer update transactions, anchor peers can be added to a Live network.

Ordering Peers :

- It approves the inclusion of transaction blocks into the ledger and communicates with other peer nodes.
- It does not hold smart contracts.
- It does not hold a ledger.

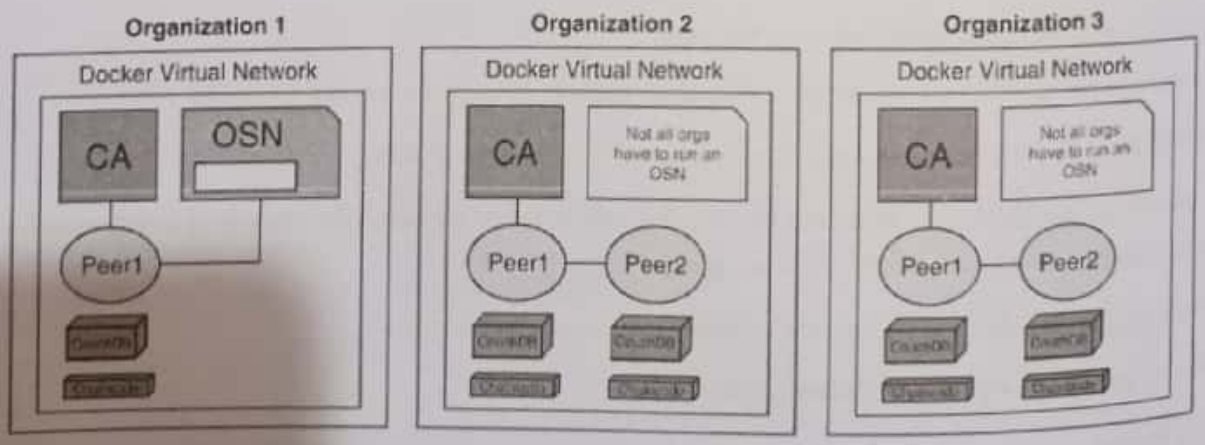


Fig. 5.12.2

Scanned with CamScanner

- Nodes in the network run in Docker Containers on Virtual Machines in the Cloud or locally.
- Each organization will join 1 or more channels.
- Each organization will designate 1 or more peers as 'Anchor Peers'.

Ordering Service Nodes

- Responsible for maintaining consensus among peers.
- Ordering service is actually the heart of consensus algorithm and the heart of Hyperledger fabric.
- Main role is to provide the order of operations.
- Before committing anything to the ledger it must pass through the ordering service.
- It is responsible for verification, security, policy , verification etc.

Hyperledger Fabric CA

- The Hyperledger fabric CA is a Certificate Authority (CA) for Hyperledger Fabric.
- It provides issuance of Enrollment Certificates
- It does certificate renewal and revocation
- It consists of both server and client components.
- It provides registration of identities or connects to LDAP as the user registry.

Channels

- Channel is a private 'subnet' of communication between two or more specific network members.
- A channel is defined by members of the organization, anchor peers per member, the shared ledger, chaincode application, and the ordering service nodes.
- Each peer that joins a channel, has its own identity given by a membership service provider(MSP).
- Channels are completely isolated. They are completely isolated instances of hyper ledger fabric. They have different ledgers, different height of blocks, policies, rules.
- We can make a policy about who can see the data in the channel and who can make an operation.
- MSP (Membership Service Provider) :
- Making digital IDs for team peers and organization users is the responsibility of the membership service provider (MSP).
- An existing network must have its peers' identities defined before a new entity may join the channel.
- An MSP implementation called Fabric CA offers a method for enrolling users from a network participant and granting them with digital IDs (X.509 certificates).
- Typically, a Docker container is used to execute Fabric CA. Each Fabric CA is set up with a backend database that holds the registered identities and their X.509 certificates. The default is SQLite, but there are additional alternatives, including PostgreSQL or MySQL. Users' private keys are not stored by Fabric CA.

Scanned with CamScanner

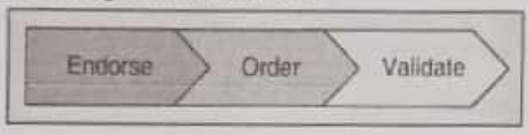
- Identity is going to be a digital certificate and users are going to be using this digital certificate to signed that transactions and submit them to the blockchain and the advantage of signing is that they authenticate with the blockchain that they are a legitimate user and it also ensure that they get the right access privileges on the blockchain for the transactions they are performing.

Chaincodes

- A chaincode typically handles business logic agreed to by members of the network, so it is similar to a smart contract.
- all our business logic is inside the chaincode.
- It is written in Go. Implementation of Java and javascript are on the way.
- Chaincode may be installed in every peer and channel.

Transaction Flows

- Consensus is achieved using the following transaction flow :



Endorsement :

- The client application is going to submit the transaction to a few peers and these peers are going to execute these transactions and agree that the output is the same across all of them.
- So they are all going to find the output of the transaction and they will add their signature to it.
- The client application has to collect endorsement from multiple peers in the networks to say that this transaction is a valid transaction and all outputs are the same.
- Once you have collected sufficient signatures, we will talk then you can submit the transactions for ordering.

Order :

- Multiple applications might be submitting, multiple users might be submitting these transactions to order.
- So now the ordering service will ensure that all of them are fully ordered and totally ordered across all the nodes.
- In the ordering service, you order the transactions and once you determine the order of all the transactions then the resolution of validation.

Validate :

- For instance you should not be performing two transactions simultaneously that modify the same state.
- So this is equivalent to the double spending problem that we would have seen in bitcoin. If you have just a 20 rupees balance in a bank account, you should not be transferring 20 rupees to two different people trying to double spend that money.
- So validation is if two transactions are going simultaneously only one of them can succeed and which one succeeds will depend on the ordering service.
- The first transactions that modify the data element succeed, every other transaction in the same block that is trying to modify the same transaction will get in validated, that is the validation step.

Scanned with CamScanner

5.14 Working of Hyperledger Fabric

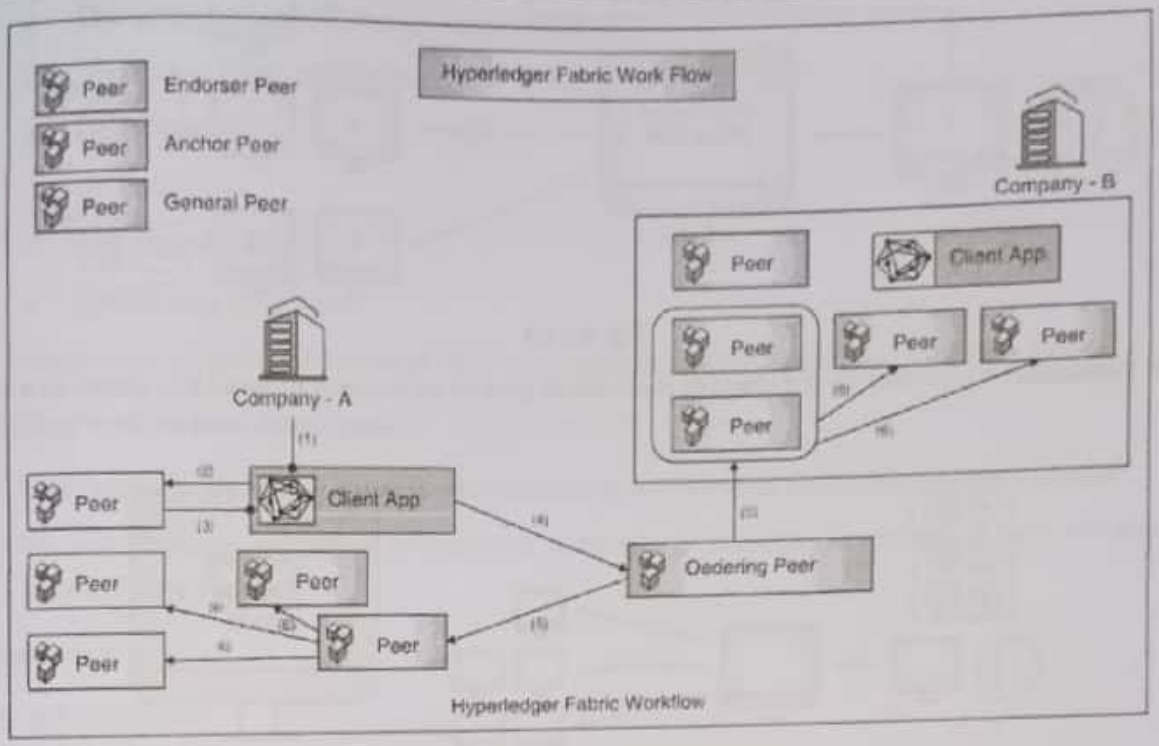


Fig. 5.14.1 : Hyperledger Fabric WorkFlow

Step 1 : Through the client application, a participant in the member Organization initiates a transaction request.

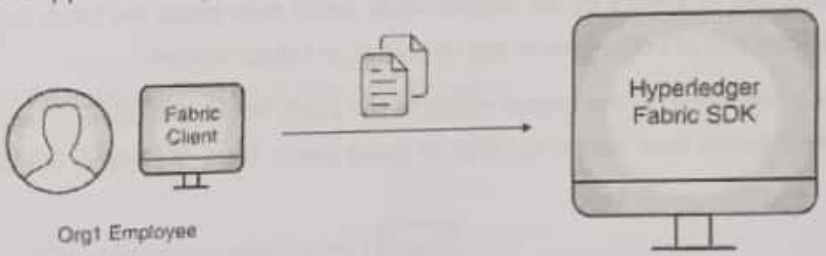


Fig. 5.14.2

Step 2 : The transaction invocation request is broadcast to the Endorser peer by the client application.

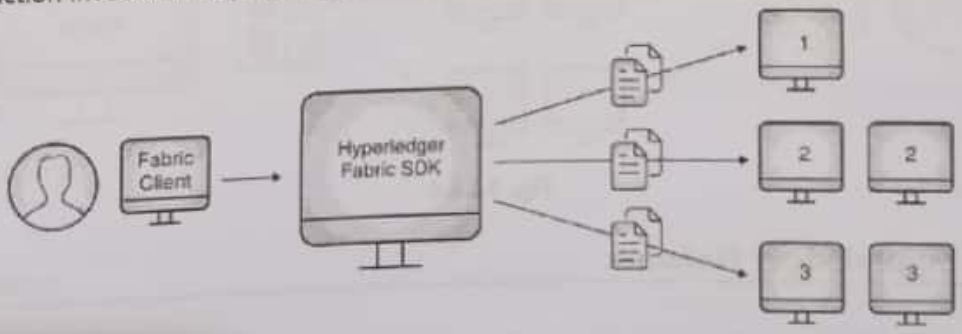


Fig. 5.14.3

Step 3 : The transaction is validated by the Endorser peer reviewing the Certificate details and others. Then it runs the Chaincode (i.e., Smart Contract) and gives the Client the Endorsement responses. As part of the endorsement response, the endorser peer sends a message approving or rejecting the transaction.

Scanned with CamScanner

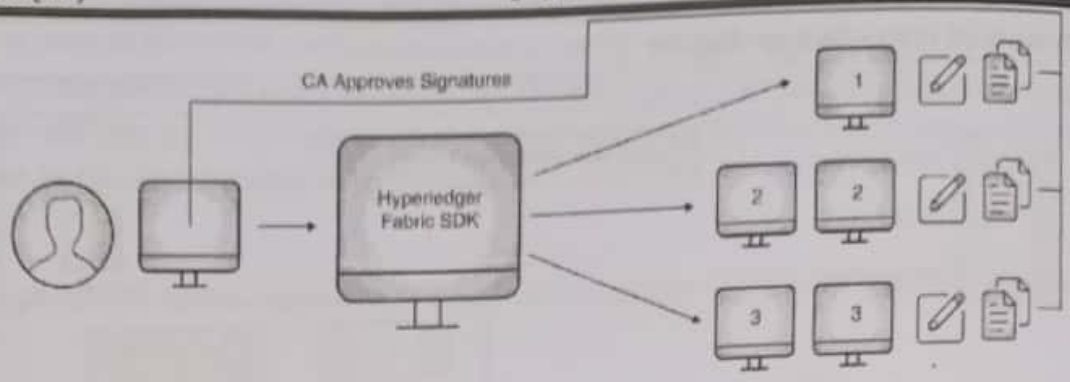


Fig. 5.14.4

Step 4 : The transaction is now sent by the client to the Orderer peer to be properly ordered and added to a block.

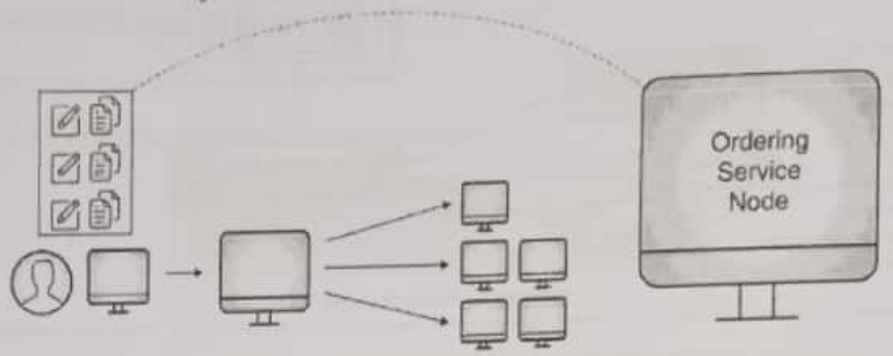


Fig. 5.14.5

Step 5 : The transaction is added to a block by the orderer node, which then sends the block on to the anchor nodes of various organizations that are members of the Hyperledger Fabric network.

Step 6 : The block was then broadcast by the anchor nodes to the peers within their own organization. The most recent block is then added to the local ledger by each of these peers. Thus, the ledger is updated across the entire network.

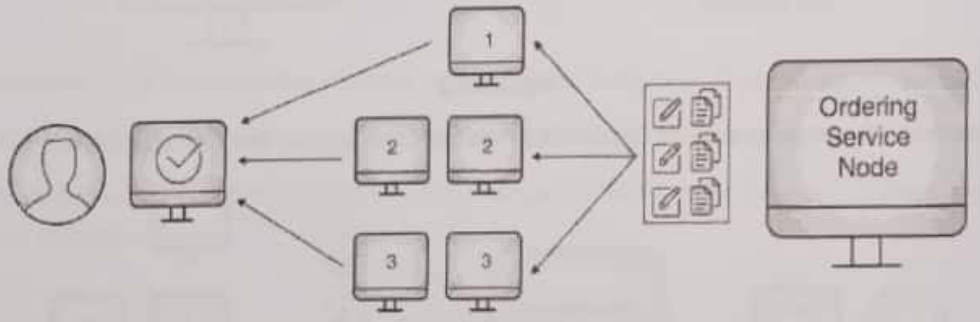


Fig. 5.14.6

5.15 Creating Hyperledger Network

- Installing Hyperledger Fabric and Composer is a good way to create a Hyperledger network.

1. Prerequisites

- You should check the Hyperledger Fabric prerequisites page at https://hyperledger.github.io/composer/v0.19/installing/installing-prereqs.html

Scanned with CamScanner

- because the versions and specifications may have changed.
  - Docker Engine and Docker Compose installation
  - Nodejs and NPM installation
  - Go installation
  - Git installation
  - CURL installation
  - Python 2.7.x installation
- Hyperledger provides a bash script for Ubuntu users to make this operation incredibly simple.

2. **Installing tools to ease development**

- Make sure you are NOT using sudo when executing the following instructions in your Terminal.
- The only necessary package is composer-cli. Although not essential, the remaining parts will eventually prove to be quite helpful.

Curl Installation

sudo apt-get install curl

After installation verify version

curl –version

Node Js Installation

curl –sL https://deb.nodesource.com/setup_10.x | sudo -E bash –

- Then run the following command

sudo apt-get update

- Run below command to start the installation for NodeJs

sudo apt-get install nodejs

- To check NodeJs is installed successfully or not run this command

node –version

Git Installation

sudo apt-get install git

git --version

Python Installation

sudo apt-get install python

python --version

Docker CE(Community Edition) Installation

First download and then install

Scanned with CamScanner

https://download.docker.com/linux/ubuntu/dists/xenial/pool/stable/amd64/docker-ce_18.06.3~ce~3-0~ubuntu_amd64.deb

sudo dpkg -i docker-ce_18.06.3~ce~3-0~ubuntu_amd64.deb

docker -version

Docker Composer Installation

Run below commands to setup Docker compose.

sudo apt-get install python-pip

pip --version

sudo pip install docker-compose

Verify the installation and check the version from below command.

docker-compose version

Hyperledger Installation

Step 1 : Run below command to download and setup Fabric.

- curl -sSL https://bit.ly/2ysbOFE | bash -s

When we run the command above, we can run into the following problem.

failed to get default registry endpoint from daemon (Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock:

Run the command below to solve this.

sudo chmod 666 /var/run/docker.sock

Now Hyperledger is installed in our system.

To create test network

Step 1 : Go to fabric-samples folder by using below command.

cd fabric-samples

Step 2 : Go to test-network folder by using below command.

cd test-network

Step 3 : Run below command to start your test-network

sudo ./network.sh up

This start the network, you can run below command to check docker containers.

sudo docker ps

- This shows you three docker containers

Scanned with CamScanner

1. One for Org1 peer node
2. One for Org2 peer node
3. One for Orderer

- When we start the network, we will also not get any channel by default. We can check the channel by using below command.

```
sudo docker exec peer0.org1.example.com peer channel list
```

- This command shows that, we don't have any channel created.

```
user@user-VirtualBox:~/fabric-samples/test-network$ sudo docker exec peer0.org1.example.com pee
r channel list
                                                    Endorser and orderer connec
tions initialized
channels peers has joined:
user@user-VirtualBox:~/fabric-samples/test-network$
```

Step 4 : Create new channel by using below command.

```
sudo ./network.sh createChannel -c testchannel
```

- This will create a new channel with the name testchannel.
- To verify this channel creation, run below command on both the peers.

```
sudo docker exec peer0.org1.example.com peer channel list
sudo docker exec peer0.org2.example.com peer channel list
```

```
user@user-VirtualBox:~/fabric-samples/test-network$ sudo docker exec peer0.org1.example.com pee
r channel list
                                                    Endorser and orderer connec
tions initialized
Channels peers has joined:
testchannel
```

```
user@user-VirtualBox:~/fabric-samples/test-network$ sudo docker exec peer0.org2.example.com pee
r channel list
                                                    Endorser and orderer connec
tions initialized
Channels peers has joined:
testchannel
```

Step 5 : To stop the network, you need to run below command.

```
sudo ./network.sh down
```

CouchDB Installation (State Database)

Step 1 : Go to fabric-samples folder by using below command.

- cd fabric-samples

Step 2 : Go to test-network folder by using below command.

- cd test-network

Step 3 : Run below command to start the network and create couchDB containers as well.

- sudo ./network.sh up -s couchdb

Scanned with CamScanner

- This command starts your network and create couchdb container for each peer as well

Step 4 : Create new channel by using below command.

- sudo ./network.sh createChannel -c testchannel1

  This will create a new channel with the name testchannel1.

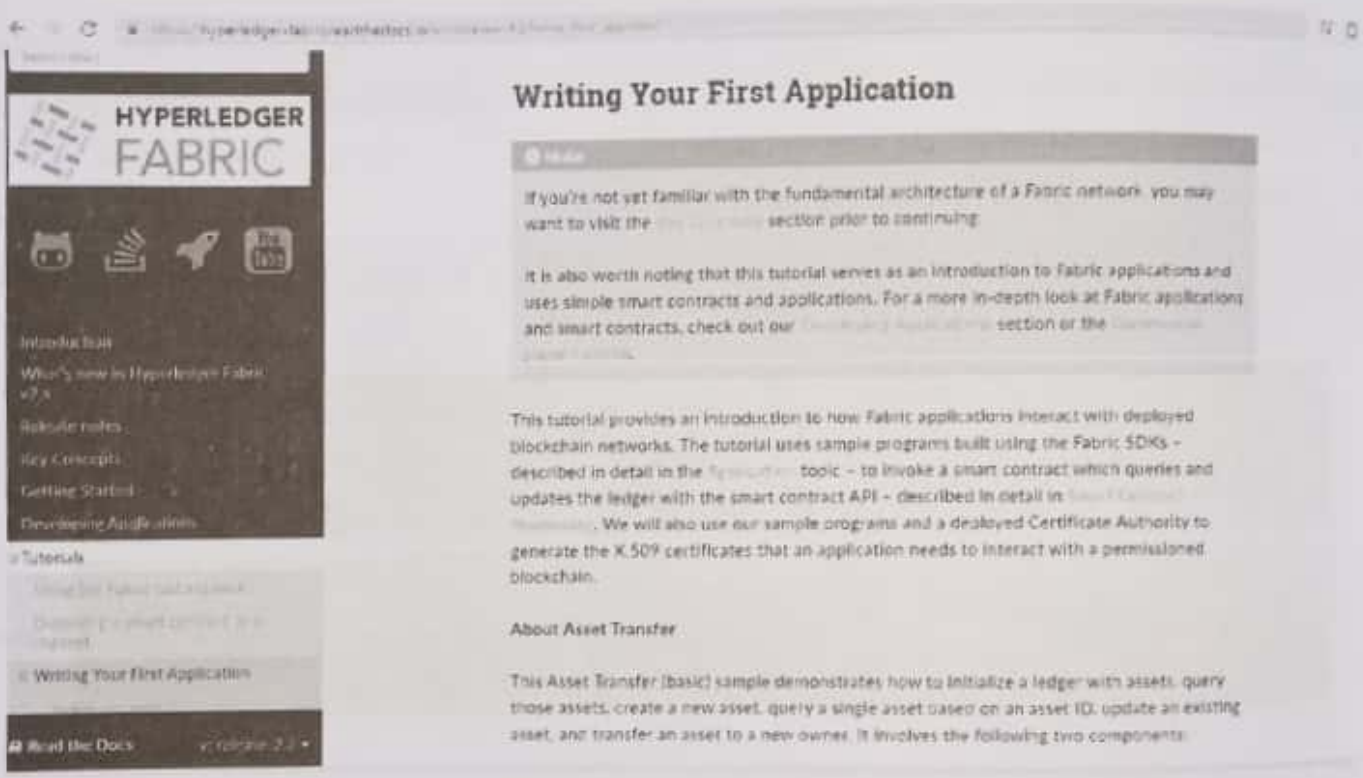
Step 5 : To stop the network, you need to run below command.

- sudo ./network.sh down

Write Smart Contract and Deploy it (Writing First Application)

We will take smart contract code from

https://hyperledger-fabric.readthedocs.io/en/release-2.2/write_first_app.html



And follow step by step instructions given on website for creating smart contract.

5.16 Case Study of Supply Chain Management using Hyperledger

- To utilize the effective qualities offered by blockchain technology, various projects have been launched in the recent years. The building and administration of permissioned networks are both tasks that Hyperledger is suitable for in the enterprise. Performance, trust levels, and scaling are all improved by the permissioned network.
- Most of the supply chain projects are implemented using Hyperledger frameworks, especially Hyperledger Fabric.

Scanned with CamScanner

Supply Chain Management- Before Blockchain

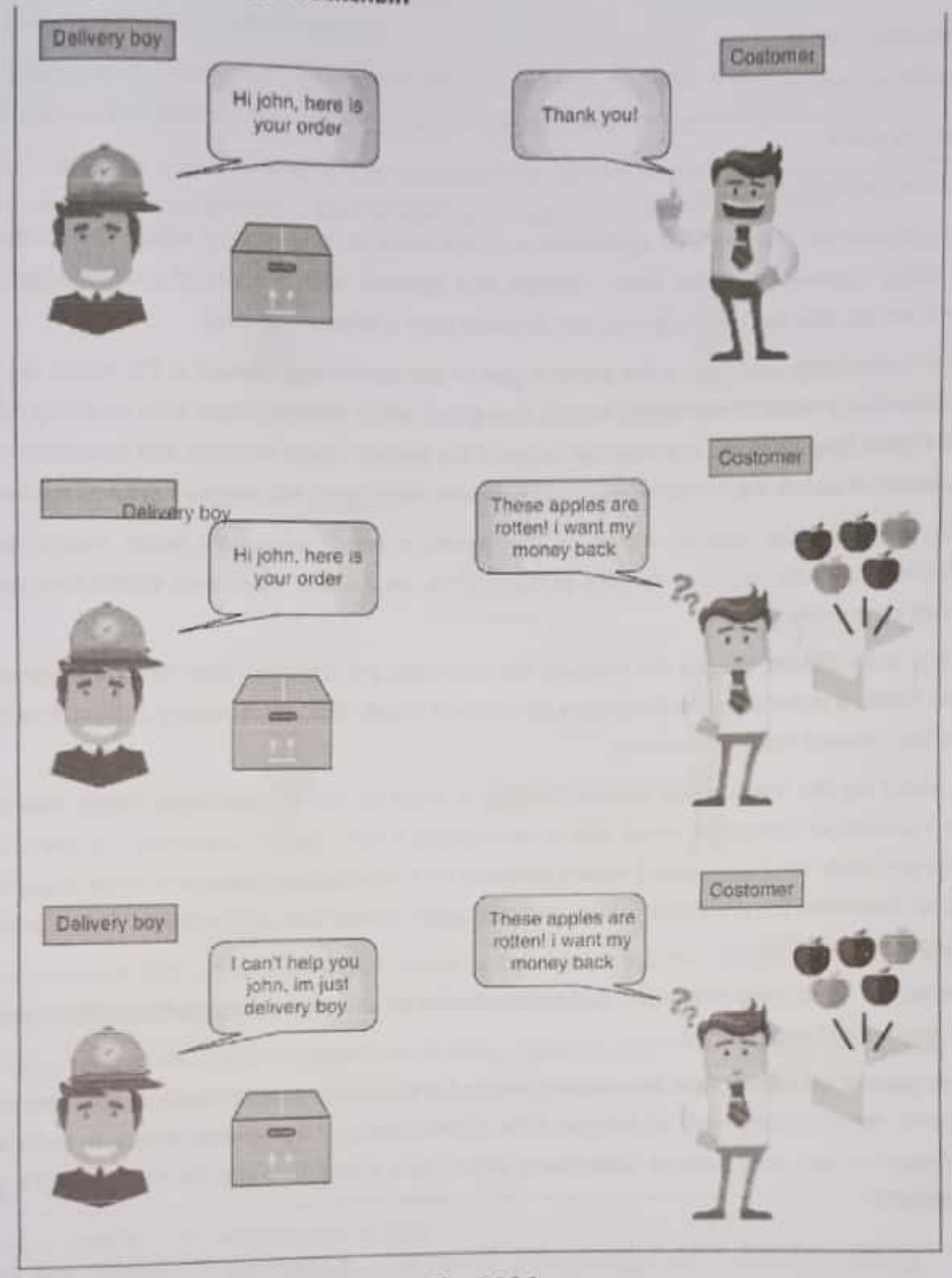


Fig. 5.16.1

- **Current issues in supply chain and logistic industries are :**
  - Risk of using centralized database
  - Good frauds
  - Traceability
  - Incompatible database with various supply chain entities.
  - Blockchain Technology can be used to track all types of transactions in a very secured and transparent manner.

Scanned with CamScanner

- **What Blockchain offers?**
  - Decentralized, high security
  - Immune or tampering
  - Traceability exists
  - Streamline transaction process
- Here we will discuss the most famous application and case study of food product supply chain industry that was developed using Hyperledger Fabric which Walmart also adopted. With the use of a Hyperledger Fabric-based system, Walmart can now track the origin of over 25 items from 5 different vendors.
- Building and maintaining user trust is the primary goal of the blockchain. Farmers in the nation are dealing with serious problems as a result of the supply chain's corruption, which prevents them from receiving the fair market value for the items they produce. The financial status of the farmers would improve, and corruption in the system will be decreased, if we use the Blockchain to make contracts transparent and payment systems automatic.
- Less product margin prices have an impact on the farmers in India's agriculture sector. Numerous parties are involved in the food chain, including farmers, shipping firms, wholesalers, merchants, distributors, grocery stores, and ultimately consumers.
- Currently, it is quite difficult to trace the products that are made and delivered from farmers to consumers. Using Hyperledger Fabric, a technology like Blockchain can increase supply chain transparency and improve the efficiency of a system that involves many transactions.
- One must enroll via the Membership Service Provider in order to join a Hyperledger Fabric network (MSP). In contrast to Hyperledger Composer, which aids in constructing a fairly generic chaincode, the use of a framework like Hyperledger Fabric for the proposed system proves to be a wise decision because it makes it easier to develop a full-fledged chaincode for protection based architecture. In comparison to Composer, Fabric offers improved Transport Layer Security (TLS).
- Using Hyperledger fabric, only authorized personnel are able to upload the records for a certain shipment, after which they are checked and released.
- This protects against tampering with the material released and guarantees that the facts are shared only among the participants, which in turn helps to safeguard the confidentiality of agreement secrets. In order to achieve a transparent supply chain, only network participants would have access to a specific consignment's data in their individual ledgers.
- Actors like Farmers, Logistics, Price_Authority, and Enterprises – a corporate for buying food products can represent most supply chains.
- Although the client is a component of the supply chain, it is not a full participant in the system because it just uses what the supply chain has to offer.
- When designing this application, we will make the assumption that any company can connect directly with any other organization, enabling a circular economy. In the traditional supply chain, the flow of goods would begin at the producer and finish at the customer.
- Farmer's organization creates a blockchain network and creates user credentials, such as a username and password, for the registered entities.

Scanned with CamScanner

- By entering a username and password to access the website, Price Authority publishes the price information for a specific product onto the blockchain network.
- Farmers Organization enters user information for the website, such as a username and password, and the information about the shipment it is sending is validated and posted on the blockchain network.
- Using user credentials such a username and password, the logistics organization checks on to the website. The information about the consignment is then verified and posted to the blockchain network.

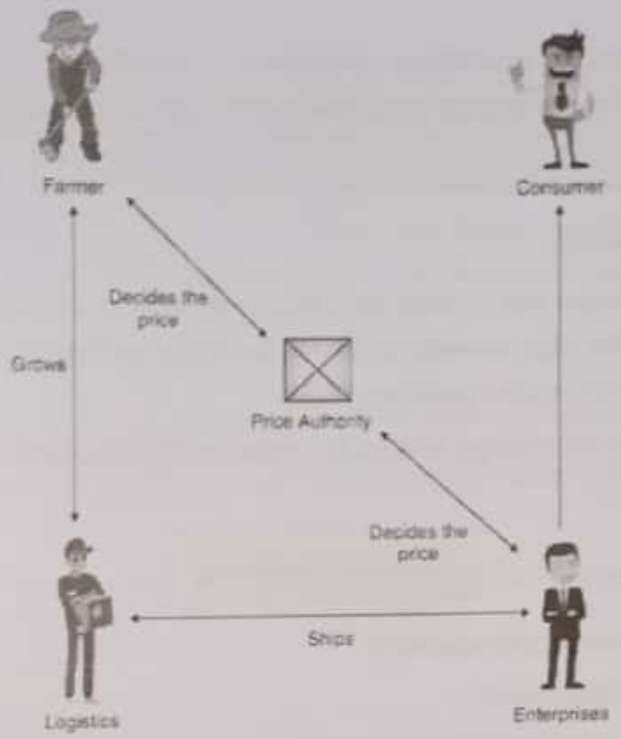


Fig. 5.16.2

- Enterprises enters in with a username and password to the website, and the information about the shipment from Enterprises is validated and published on the blockchain network.
- If the product price is reduced or increased due to various reasons, then Price Authority will update on distributed ledger which will be verified by other actors and famers will also come to know about the actual price he is getting for his goods.

Prerequisites

- We need to install the following third-party tools :
  - Virtual box available at https://www.virtualbox.org/wiki/Downloads or native Linux (Ubuntu)
  - Git (https://git-scm.com/)
  - Go Language (https://golang.org/dl/)
  - Docker engine available at https://docs.docker.com/install/linux/docker- ce/ubuntu/ (version +17.03)
  - Docker Compose available https://docs.docker.com/compose/install/
  - (version +1.8)
  - Curl (https://curl.haxx.se/)
  - Node available at https://nodejs.org/en/ (version 8.x < 9) NPM (version 5.x)
  - - Python 2.7.x and Pip

Scanned with CamScanner

Installing Hyperledger Fabric

- The next step is to build a local Hyperledger Fabric runtime to deploy your business networks to.
- Make sure you have installed all the dependencies required for us to make the Fabric environment run. Then, create a folder with the name food-supply-chain/, which will host our Fabric project and network configurations files.
- Enter the project's directory and install the platform-specific binaries :

```
curl -sSL
ttps://raw.githubusercontent.com/hyperledger/fabric/release-1.2/scripts/bo otstrap.sh | bash
```

As a result, you'll preload all of the requisite Docker images for Hyperledger Fabric 1.2 in one shot.

Fabric's Chaincode

- Now need to write smart contracts.
- The smart contract that manages the business logic chosen by network participants is called Chaincode. It is a piece of code that executes the logic operation within the blockchain network and may be created in any common programming language and run inside a container.
- Next, we need to build a local Hyperledger network to deploy the Chaincode and start tracking the food along the supply chain.

Review Questions

Q. 1 What is private blockchain with its characteristics?

Q. 2 What is the need of private blockchain?

Q. 3 Explain state machine replication?

Q. 4 Explain PAXOS consensus algorithm.

Q. 5 Explain RAFT consensus algorithm.

Q. 6 Explain Byzantine Fault Tolerant algorithm.

Q. 7 What is the need of Practical BFT?

Q. 8 Short note on Hyperledger frameworks.

Q. 9 Short note on Hyperledger tools.

Q. 10 Explain Hyperledger fabric with advantages.

Q. 11 Differentiate between Hyperledger fabric and other technologies.

Q. 12 Explain Hyperledger fabric architecture.

Q. 13 Explain working of Hyperledger fabric.

Q. 14 How to create Hyperledger network.

Q. 15 What are the issues in current supply chain management system? Explain how blockchain with Hyperledger fabric can help to overcome these issues.

□□□

Scanned with CamScanner

6 TOOLS AND APPLICATIONS IN BLOCKCHAIN

MODULE - VI

Syllabus

Corda, Ripple, Quorum and other Emerging Blockchain Platforms, Blockchain in DeFi : Case Study on any of the Blockchain Platforms.

Enterprise blockchains include Quorum, Hyperledger Fabric, and Corda, respectively. These are private networks with functions created for business enterprises, such as knowing the identities of network participants, and access to them is restricted to certain parties.

6.1 Corda

- It is a blockchain platform that is open-source and uses smart contracts to let companies deal **privately** and **directly.**
- Corda is a distributed ledger technology (DLT) that R3 created specifically for enterprise use, with privacy as the guiding principle.

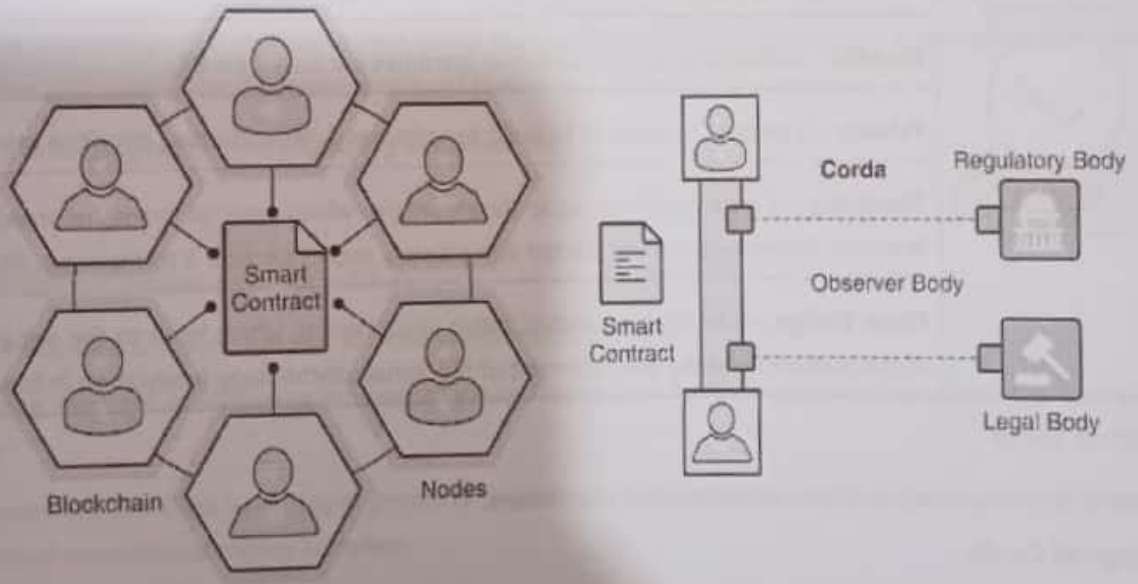


Fig. 6.1.1

- The corda protocol is built on a strong identity model, where every node's identity must be proven to have been properly on boarded by using an x.509 certificate with a validated trust chain.
- Corda is custom-designed for the financial industry.
- It reduces record-keeping expenses and offers development services like Corda App Consulting, User Interfaces, Regulated Tokens, and others.
- The distributed apps created with Corda are known as CorDapps.

Scanned with CamScanner

Public/Private/Permissioned – Public Permissioned, Private, Permissioned

Version: Corda OS 4.4

Consensus: Single Notary, Raft, BFT-SMaRt

Transaction Privacy : Yes

Transaction Visibility : Private transactions between participants are hidden from all other participants.

Smart Contract Language : Java, Kotlin

- Signing identities are only available on Corda nodes.
- Corda enables the creation of immutable financial event records. However, unlike other blockchains, transactions in Corda are private.
- And, most importantly, Corda is not bound to any particular consensus algorithm or cryptocurrency.
- It makes use of the Notary infrastructure for transaction sequencing and validation. A corda network may contain multiple notaries who use different algorithms to validate transactions.

Features of R3 Corda

Node Permissions - On each node, a trusted network map service is supplemented by file-based configurations. Corda networks are divided into compatibility zones, each of which is governed by a different Certificate Authority.
Identity - Individual and organizational identities are based on PKI.
Privacy - It enables businesses to build permissioned networks that prioritize privacy.
Flexibility - Corda facilitates agile development, which gets to meet the needs of the business. Businesses can get started immediately and customize it to their specific needs.
Open Design - It facilitates a collaborative environment, which leads to the enhancement of the platform and the advancement of blockchain technology integration in businesses.

Advantages of Corda

- No mining, cryptocurrency or token- no dependence on miners.

Disadvantage of Corda

- Because the Corda framework is tailored to the financial sector, the project's target audience is limited to financial institutions.

Enterprise Use Cases :

- Corda has seen widespread industry adoption, particularly among financial and insurance firms.
- B3i - reinsurance network
- Marco Polo - global trade finance network

6.2 Ripple

- A blockchain platform called Ripple concentrates on the financial aspect of the digital revolution.
- This real-time technology enables banks and other financial institutions to transmit payment transactions across the globe in real time.
- Chris Larsen and Jed McCaleb co-founded it, and it was first released in 2012.
- Ripple is an open source person-to-person payment network that allows anyone in the world to send money to anyone else for free.
- For banks and payment processors, Ripple Labs aims to make it simple and affordable to send money abroad.
- There is a big problem when you want to send money from one country to another country. Sometimes it takes days and it is usually expensive, and in some countries you even have to do illegal things to make it happen.
- Ripple Lab is trying to solve a big problem. Nowadays, most international transfers use Swift, which is a protocol that makes up half of all international wires and moves around six trillion dollars each day, but it's very slow and takes a lot of fees.
- Ripple's primary goal is to eliminate the need for older systems such as Western Union or Swift.
- Ripple is a distributed payment protocol.
- Ripple is a financial technology that functions as both a cryptocurrency and a digital payment network.
- It also comes with solutions for minimizing liquidity expenses and sending rapid payments.
- Currently, more than 300 companies are using the platform to create and deploy financial assets. With Ripple, businesses can create global payments or build their own custom financial solutions.
- Ripple is emerging as the next promising virtual currency.

XRP is Ripple's token

- Ripple's XRP can settle a payment in 3.5 seconds.
- XRP is a fast, direct, and secure way to send payments on the web.
- In essence, through its RippleNet Internet of Value, Ripple provides businesses with an alternative to the banking system.
- Companies are provided with Ripple products like xRapid, xVia, and xCurrent in order to improve their current methods of international money transfers.
- Ripple operates on an open-source, peer-to-peer decentralized platform that enables the seamless transfer of money in any currency, including dollars, yen, euros, and cryptocurrencies such as litecoin and bitcoin.
- Rather than using blockchain mining, Ripple confirms transactions using a consensus mechanism via a group of bank-owned servers.
- It is a decentralized system using a consensus model called the Unique Node List

Scanned with CamScanner

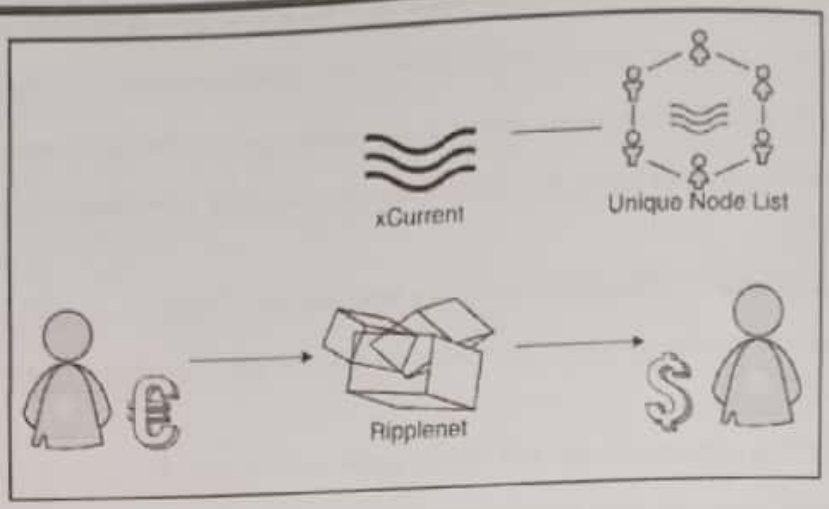


Fig. 6.2.1 : Ripplenet Use

Need of Ripple

- Ripple is a decentralized but unified system.
- We can transfer money from one system to another, or we can keep things simple in Ripple.
- It is also simple to transfer funds into and out of Ripple.

Bitcoin Vs Ripple

- Ripple is consensus oriented. Ripple's token, unlike Bitcoin, Ethereum, and many other cryptocurrencies, is not mined.
- Bitcoin can handle 3-6 transactions per second, whereas Ripple can handle 1500 transactions per second.

Features of Ripple

- Consensus and Validation - The order and validity of the XRP transactions are agreed upon by independent validator nodes. There is no mining process involved here because the validators have already been chosen and the total XRPs already exist. As a result, no time, effort, or money is expended on mining.
- It is an open source protocol that allows for quick and inexpensive transactions. It is a currency as well as a platform.
- It is a trusted platform with increased visibility, which eliminates payment failures.
- It offers digital security to protect against counterfeit coins.
- Ripple allows for the exchange of any currency or valuable asset with a low unified commission.

Advantages of Ripple

- Transactions are less expensive and faster than with Bitcoin.
- Its primary purpose is to be used by banks.

Scanned with CamScanner

- Ripple is capable of being converted into any currency.
- People must wait days for cross-border payments to be completed. RippleNet assists in resolving such concerns, as well as improving transaction speed and lowering associated costs.
- On the Ripple network, transactions are settled in seconds. In contrast, banks may take days or weeks to complete a wire transfer.
- Use less computing.

Disadvantages of Ripple

- Ripple is primarily expected to target larger players in the financial market. As a result, it is only of limited use to retail users.
- The Ripple Company owns the majority of the XRPs. This indicates that the company may have control over the network's operations.

6.3 Quorum

What is Quorum?

- A distributed ledger protocol called Quorum is based on Ethereum.
- As Quorum only modifies Ethereum's core slightly, it is intended to grow and change alongside Ethereum..
- It is a JP Morgan's Enterprise focused blockchain which is open source (LGPL -3.0 License).
- Quorum is a free and open source blockchain protocol designed specifically for use in a private blockchain network, in which only one member owns all nodes, or a consortium blockchain network, in which multiple members each own a portion of the network.
- It is a soft-fork of the very well-known public Ethereum blockchain.
- Quorum is an Ethereum blockchain aimed specifically at the finance sector.
- It is created solely to provide privacy for private transactions between nodes.
- The main feature of Quorum is privacy. Transactions and smart contracts on the blockchain can be private.
- Quorum features multiple voting based consensus mechanisms that do away with the current proof- of- work consensus mechanism used today by the public Ethereum blockchain.

Public/Private/Permissioned - Private, Permissioned

Version - Peer:2.7.0, Transaction manager:Tessera 0.10.6

Consensus - IBFT, Raft

Transaction Privacy - Yes

Transaction Visibility - Private transactions between participants are hidden from all others.

Smart Contract Language - Solidity, Vyper

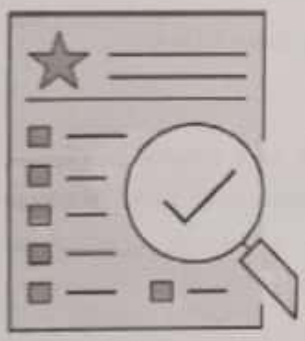
Scanned with CamScanner

- Quorum can be run with two different privacy options, Tessara and Constellation, which are used by Quorum to encrypt and share data among nodes.
- Transactions on the Quorum network don't cost Ether. Voting is the primary method of transaction verification, a mechanism for agreement.

Need for Quorum

- Even if there are several organizations handling the information in the banking industry at the moment, there is still a lack of transparency, information management, and security.
- Even if standard blockchain technology offers traceability and immutability, it nevertheless falls short of meeting the criteria of the banking industry.
- A solution that offers private control on the blockchain through automation and is adaptable to demands is required.
- Public Ethereum had some challenges in adoption with an enterprise environment that quorum addresses, which includes privacy, speed, and throughput.

Features of Quorum



- **Privacy** - Transactions and smart contracts on the blockchain can be private, allowing access only to participants that are involved in a specific transaction or access to entities such as regulatory bodies.
- **Secure** - It uses advanced encryption techniques and ensures only authorised entities can access the network.
- **Voting** - based consensus mechanisms (Flexible consensus) - Quorum features multiple voting based consensus mechanisms which do away with the current proof of work consensus mechanism used today by the public Ethereum blockchain, among many others. It utilizes raft-based and Istanbul Byzantine Fault Tolerance (BTF) consensus mechanisms.
- **Peer permissioning using smart contracts** - Quorum allows peer or node permissioning using smart contracts, which ensures that only known parties can join the network.
- Increased scalability and network performance.
- **Community** - It leverages the world's largest pool of blockchain developers.
- **Performance** - There is a high TPS due to the private and permissioned environments of known peers only.
- **Finality** - The pluggable consensus prevents chain forking and transaction reversal.

Scanned with CamScanner

Quorum Architecture

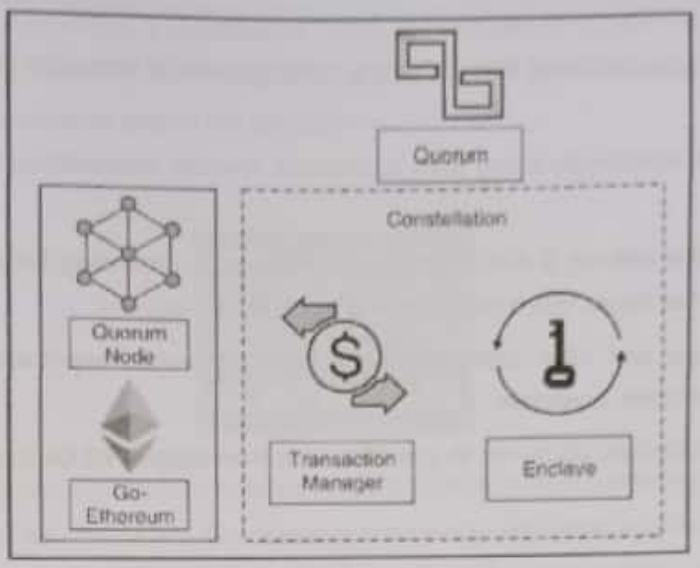


Fig. 6.3.1 : Quorum Architecture

- In architecture, there is a Quorum node, which is the Go Ethereum client.
- They have a notion of constellation, which is really a general purpose peer to peer encrypted message exchange system. This allows peers to exchange messages in a privacy persevering manner.
- This constellation is also responsible for transaction execution.
- Transaction manager is responsible for transaction privacy and it doesn't hold any secret information.
- Enclave is a secure processing environment that acts as a black box for processing commands and data. All cryptographic work is separated out into an enclave. Enclave ensures that the key remains private and encryption, decryption all happens within this secure enclave.

Advantages of Quorum

- Quorum offers faster transaction speeds because private contracts are typically used, and private contracts perform better than public contracts.
- Transaction-level privacy : Quorum is the very first distributed ledger platform to include a security layer with zero knowledge (ZSL). This feature is intended to protect users' privacy by using shielded transactions.
- Smart contracts can be private or public, and they are programmed using Solidity. A smart contract that has been set to private cannot be converted to public. Accordingly, public smart contracts cannot be converted to private smart contracts, making them more secure.

Disadvantages of Quorum

- Quorum does not require a built-in cryptocurrency because consensus is not achieved through mining. With Quorum, it is not possible to create a native currency or a digital token.
- Scalability : As use cases become more complex, Quorum's channel-based approach to privacy poses privacy and scalability challenges.

Enterprise Use- Cases : Starbucks, Louis Vuitton, JPM Coin, Microsoft & EY

Scanned with CamScanner

6.4 Emerging Blockchain Platforms

- Blockchain platforms are new platforms that are nearly indistinguishable from core blockchain technology in some cases.
- Blockchain platforms are increasingly being used to improve financial transactions, trade, traceability, and supply chain efficiency.
- A significant portion of this interest is due to the cyclical upswing surrounding Bitcoin, which is based on an old blockchain platform that has issues with speed and energy consumption.
- To help with these issues and offer useful value for other enterprise applications and uses, contemporary blockchain platforms have been developed.
- Here are the emerging blockchain platforms to consider other than **Ripple, R3 Corda, and Quorum**
  - Ethereum
  - IBM Blockchain
  - Hyperledger Fabric
  - IOTA
  - Microsoft Azure Blockchain
  - Oracle Blockchain Cloud Service
  - ConsenSys Quorum
  - Stellar

Ethereum

- Ethereum is the most secure block chain-based crypto currency platform. It contributes to the development of secure and transparent crypto currency solutions for customers. Its primary strength is that it supports true decentralization via smart contracts. It has its own cryptocurrency, ether, in addition to its role as a blockchain platform that forms the basis for enterprise applications.

Hyperledger Fabric

Scanned with CamScanner

- The Linux Foundation hosts the Hyperledger Fabric platform, which is an open source blockchain framework. One of the most permissioned blockchains is Hyperledger Fabric. Smart Contract (Chaincode) and Pluggable Consensus functionality is provided. It makes it easy to enable digital records without central authorities' approval and provides numerous opportunities to extend the functionality.

IBM Blockchain

- IBM Blockchain is the company's blockchain division that offers enterprise blockchain services. In contrast to public blockchains such as Bitcoin and Ethereum, IBM develops private and permissioned blockchains.IBM has invested in developing a user-friendly interface to make critical tasks like configuring, testing, and rapidly deploying smart contracts easier. IBM Corp. (IBM) has launched the USD Anchor cryptocurrency in collaboration with Stellar, a blockchain that shares technology with Ripple, and Stronghold, a startup. USD Anchor coins are backed by an equivalent amount of US dollars.

IOTA

- Despite being one of the younger blockchain offerings on the market today, IOTA is one of the most exciting platforms. IOTA is a cryptocurrency and open-source distributed ledger created specifically for the Internet of Things. It stores transactions on its ledger using a directed acyclic graph, which is encouraged by the potential for greater scalability over blockchain-based distributed ledgers. IOTA does not use any chains or blocks and is instead powered by its own unique technology.

Microsoft Azure Blockchain

ConsenSys Quorum

Scanned with CamScanner

- ConsenSys acquired JPMorgan's intellectual property assets for the Quorum platform in late 2021 and incorporated them into its own work to build the ConsenSys Quorum open source protocol layer. Enterprise customers such as JPMorgan and the South African Reserve Bank use the firm's development services for the combined platform.

Stellar

- Stellar is a decentralized, open network that connects the world's financial infrastructure. Stellar is used for asset tokenization, holding and diversifying value, and sending and exchanging assets. Stellar is a newly developed blockchain platform that has been optimized for a variety of DeFi applications. It implements the Stellar Consensus Protocol, which is said to shorten the time it takes to process and finalize transactions on a public blockchain network.

Difference between popular and emerging blockchain platforms

Platforms	Ethereum	Hyperledger Fabric	R3 Corda	Ripple	Quorum
Public/Private /Permissioned	Public Blockchain- No permission is needed to access network content.	Private Blockchain- Network is limited to people with permission.	Private Blockchain- Permission is needed to access network content.	Public Blockchain- Permissionless network content	Both Public, Private permissioned network content
Smart Contract	Solidity programming language	Golang programming language	Kotlin programming language	C++ programming language	Solidity and Vyper
Governance	Carried out by developers (DAO)	Linux foundation incharge	R3 company in charge	Ripple Labs	Etherum developers and JP Morgan Chase
Consensus	Relies on Proof of Stake for decision making	The consensus process does not require participation from every node in a network.	Making decisions is limited to those parties engaged in a transaction.	Participating nodes verify the authenticity of a transaction by conducting a poll.	Multiple voting based consensus mechanisms which do away with the current proof of work consensus mechanism used today by the public Ethereum blockchain among many others

Scanned with CamScanner

Platforms	Ethereum	Hyperledger Fabric	R3 Corda	Ripple	Quorum
Consensus Protocol	PoW/ PoS	Puggable	Notary-based	Probabilistic voting	Pluggable
Currency	Ether	No native currency	No native cryptocurrency	XRP	ETH
Use Case	Popular with generalized applications and mostly used for P2P and B2C operations.	the preferred platform for B2B operations, mainly used in enterprise.	Runs on a specialized distributed platform for the financial industry needs.	Runs on a specialized distributed platform for the financial industry needs.	Popular with generalized applications and mostly used for P2P and B2C operations.

6.5 Blockchain in DeFi : Case Study on any of the Blockchain Platforms

- Enterprise DeFi is a term used to describe major global corporations that incorporate Decentralized Finance (DeFi) technology into their services, products, and operations.
- It enables institutional investors to easily access new lending markets based on blockchain-powered financial infrastructure, which was previously impossible.
- The financial sector is trying to take lead in blockchain adoption by investing heavily in the technology.
- We need advanced solutions that incorporate existing legacy technologies, workflows, and regulatory systems.
- There are three types of networks : public, private, and federated.
- A public blockchain network can be joined and used by anyone. Everyone has access to the ledger and can be involved in the consensus mechanism.
- In a private blockchain, only one organization has authority. In other words, the authorities decide who can participate in and access the network. Consider it a "centralized-decentralized" network.
- When you use a federated blockchain, you will gain access to a decentralized private environment. Federated blockchain is central to private blockchain. It is used in financial services, insurance payments, multi - party aggression, supply chains, and many other industries.
- Over the last three years, enormous efforts have been made to establish federated networks, with over a dozen projects, the majority of which use R3 Corda or Hyperledger.
- This is appropriate for businesses who want to use blockchain without going public. These authority nodes have already been chosen from among all the organizations connected by the network. A block is validated by this selected group of nodes. Furthermore, this suggests that the selected nodes have access to information and areas that are restricted.

Scanned with CamScanner

A Case Study of the Trade Finance System using R3 Corda

- The method by which financial firms grant credit to ensure the exchange of goods is known as "trade financing", and it has existed for hundreds of years with little change as the volume of international trade has increased.
- The financial services and equipment used by companies to promote domestic and international trade are referred to as "trade finance".
- The environment is not ideal right now. For instance, at present, transactions are carried out using third-party applications.
- Furthermore, it may necessitate a significant amount of documentation, including multiple copies and the involvement of multiple parties.
- Ultimately, simple tasks can take weeks to complete.
- The players in the environment of trade finance include importers, exporters, financial institutions, trade finance entities, export credit companies, and insurers.

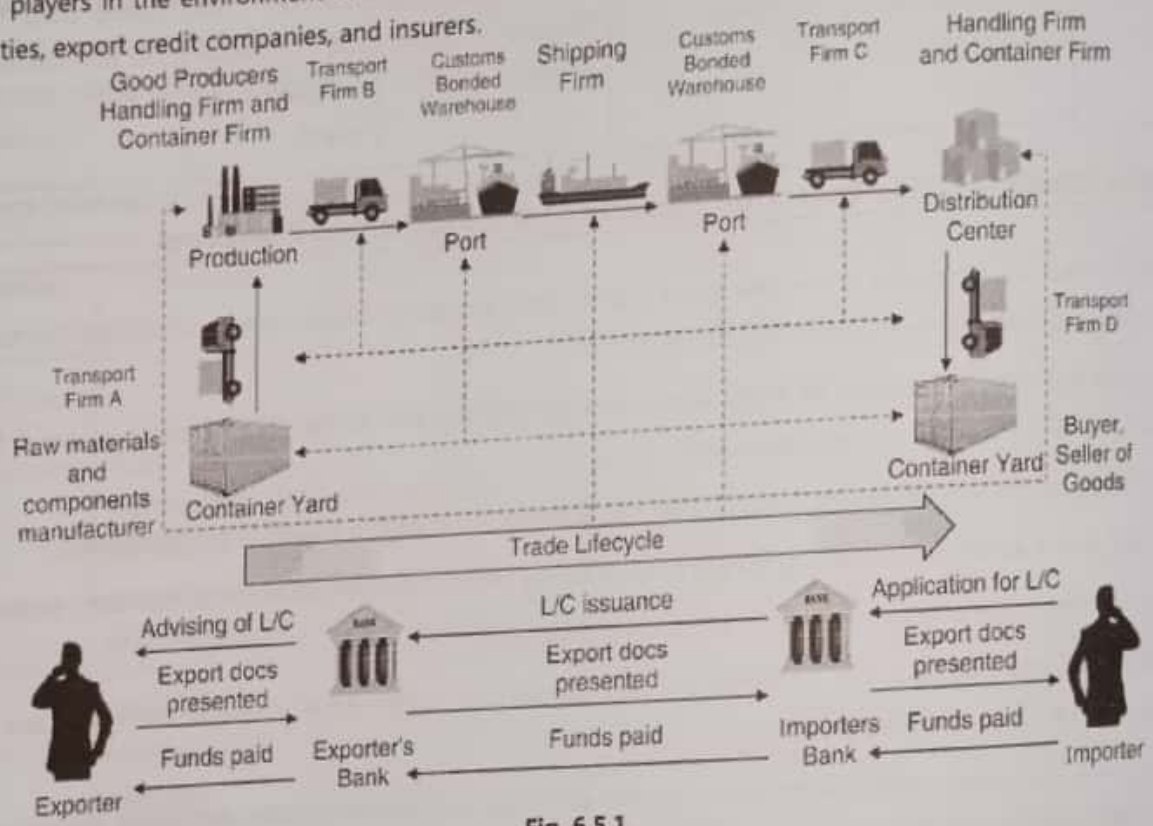


Fig. 6.5.1

- Current trade finance issues are :
  - Every good in the supply chain must be checked for quality on a regular basis.
  - Throughout the supply chain journey, each good must interact with a variety of players and people.
  - A product can take up to 30-35 days to travel from farm to retailer. Furthermore, the documents may take up to ten days to complete.
  - This leads to incorrect party tracking and fraud, such as fake deliveries.
  - Costly throughout, especially when it comes to writing the letter of credit, the process involves banks and clients. Additionally, disputes are managed improperly and take a long time to settle.
  - They face a significant regulatory burden because they must manage things, including fraud prevention, KYC protocols, geopolitical risks, etc.

Scanned with CamScanner

- Furthermore, the seller finds it difficult to maintain product conditions, introducing product risks.
- Currency risks arise as a result of fluctuations in foreign exchange currency rates.
- The transport risk, on the other hand, arises as a result of cargo insurance, which raises the overall cost of the transaction.

How Can Blockchain (R3 Corda) be used in Trade Finance?

- Blockchain has the potential to completely transform trade finance.
- It can ensure that there are no duplicate records.
- Additionally, it eliminates any chance of fraud or alteration.
- Additionally, it ensures a constant level of trust while increasing visibility.

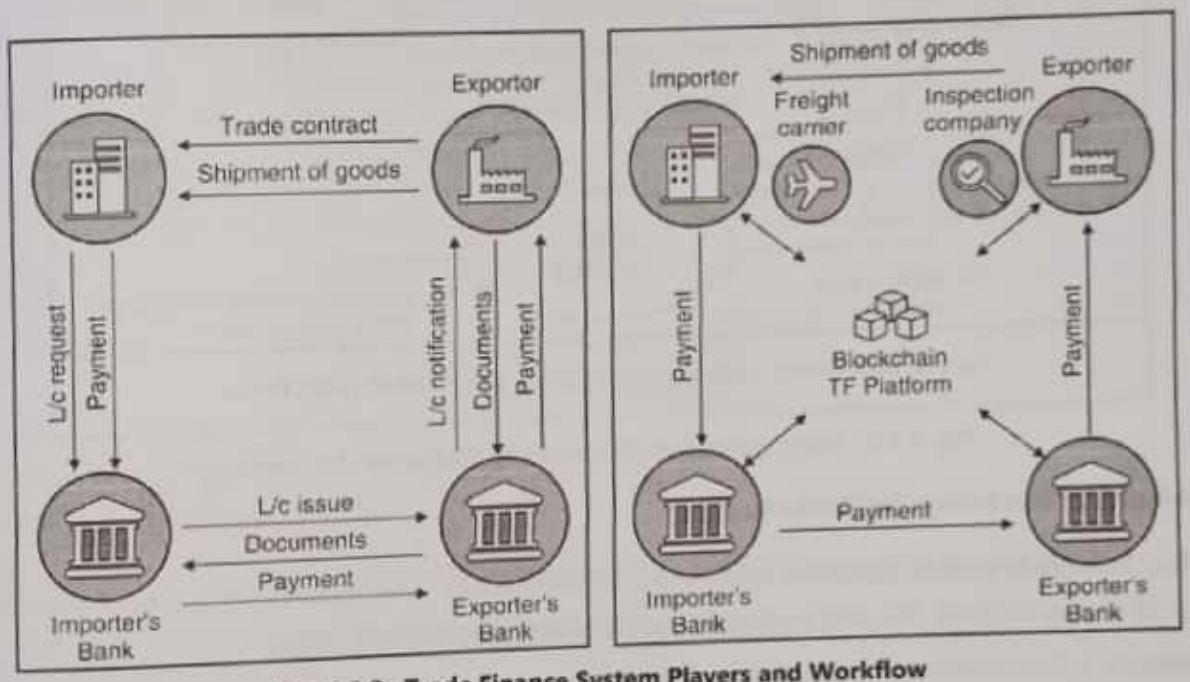


Fig. 6.5.2 : Trade Finance System Players and Workflow

How?

- A trustworthy trading partner can be found by buyers and sellers using blockchain.
- The terms are then agreed upon by both parties, and the smart contracts are initiated. The order is then initiated after the smart contract is created.
- In the event that the smart contract requirements are satisfied, the buyer bank assumes control at the following location and will automatically settle the amount.
- To expedite the settlement process, the seller could request financing from his bank.
- The seller is now shipping the goods. Both parties are keeping an eye on the package.
- The buyer confirms the trade, indicating that the condition on the smart contract has been met.
- Finally, the seller is compensated.
- The procedure is straightforward, and as you can see, the majority of the steps are automated in order to maintain transparency and trust.

Scanned with CamScanner

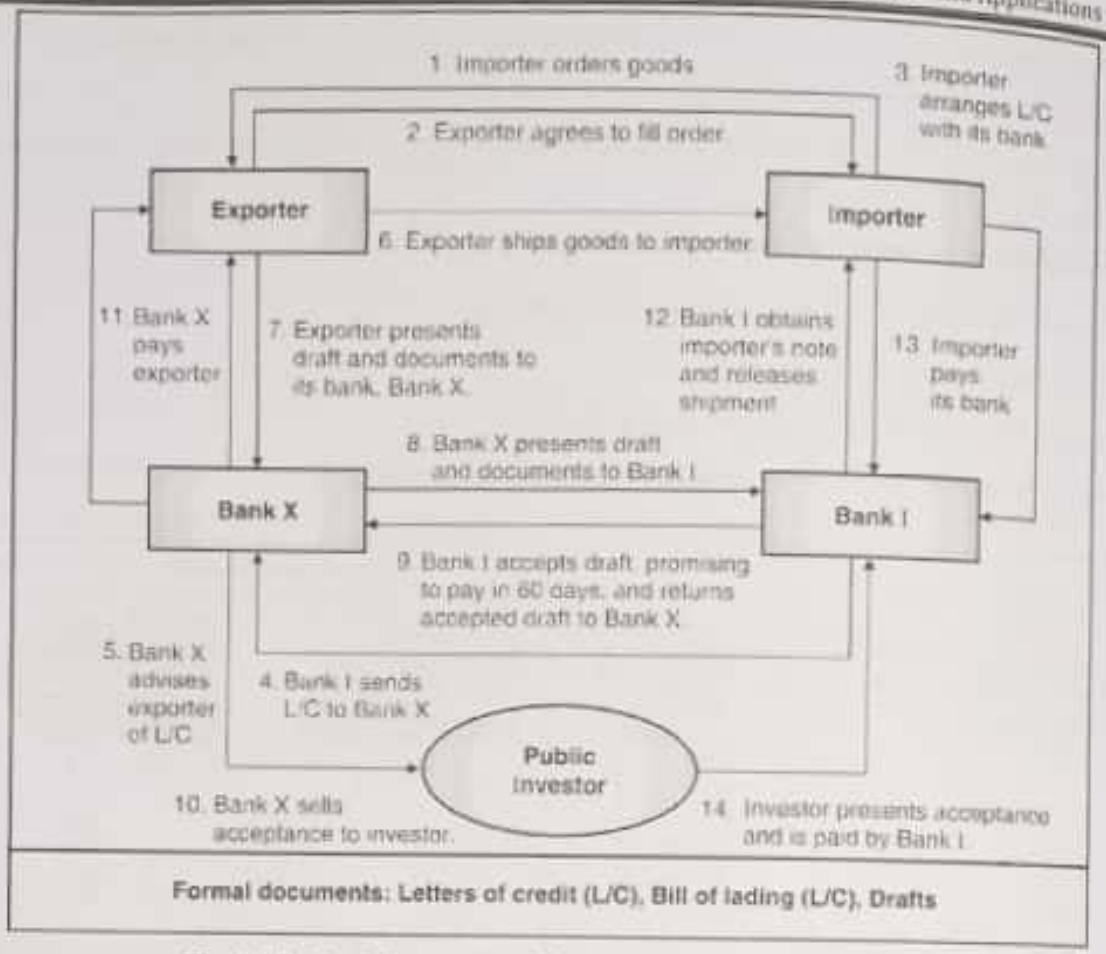


Fig. 6.5.3 : Trade Process Workflow using Blockchain R3 Corda

The Advantages of Blockchain for Trade Finance

- The blockchain trade finance platform's documents can be checked for authenticity at any time! It helps to save time for all parties involved. This also implies that no fraudulent activities take place.
- Transparency is also provided by blockchain in trade finance.
- There are no middlemen, which improves the overall system by reducing fraud. This increases the banks' ability to do trade financing without risk or dispute.
- Blockchain in trade finance can prevent double spending.
- Smart contracts could indeed help to automate and eliminate time spent on paperwork. Before smart contracts can be executed on the network, they must be finalized. Before the smart contract agreement is drafted, both partners must meet and finalize the details.
- Furthermore, the smart contract is only executed when a predefined condition or set of rules is met.
- Blockchain enables buyers and sellers to have proof of ownership, ensuring transparency and tracking the location of the shipment.
- Regulations could be managed in a single location. KYC/AML solutions are included. Additionally, it facilitates trade between nations by allowing for the management of regulatory differences between them.
- Besides this basic use case, blockchain is used in tons of trade finance use cases like, international trade-Letter of Credit, Maritime Trade, etc.

Scanned with CamScanner

- R3 Corda real-world use-cases are TMX, Payments Canada, Maersk, NatWest, CrowellMoring, and Swift.

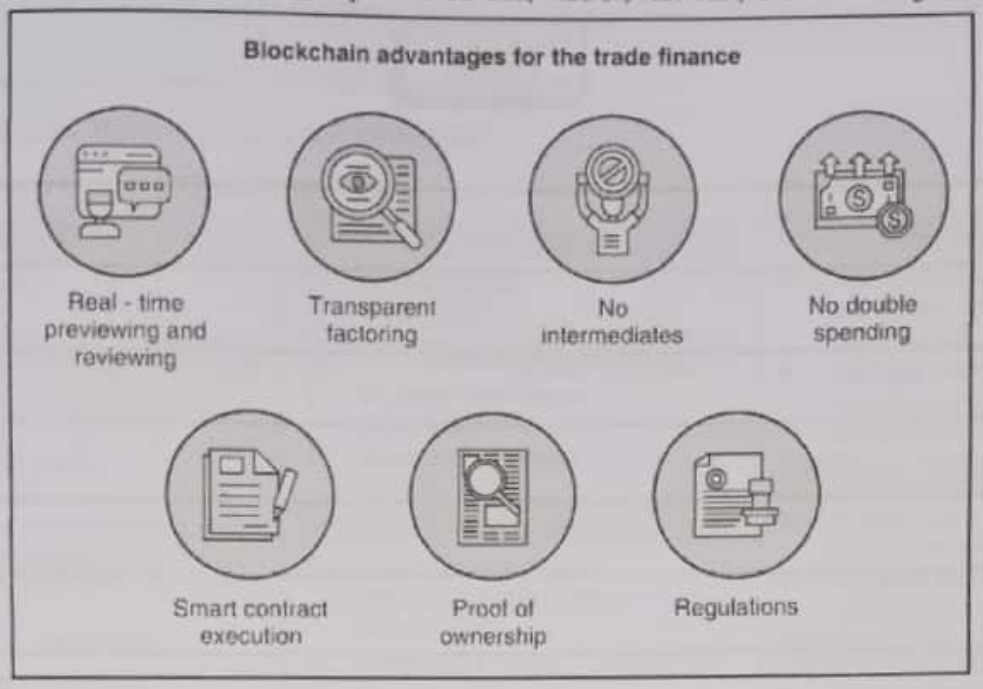


Fig. 6.5.4

Review Questions

Q. 1 Explain R3 Corda blockchain platform with advantages and disadvantages.

Q. 2 Explain Ripple blockchain platform.

Q. 3 Which are the different emerging blockchain platforms are available? Differentiate all the platforms?

Q. 4 How blockchain can be used in decentralized financial application?

Q. 5 Explain Quorum blockchain platform with advantages and disadvantages.

❑❑❑

Scanned with CamScanner