Unit III

3 Blockchain Platforms and Consensus in Blockchain

Syllabus

Types of Blockchain Platforms : Public, Private and Consortium, Bitcoin, Ethereum, Hyperledger, IoTA, Corda, R3.

Consensus in Blockchain : Consensus Approach, Consensus Elements, Consensus Algorithms, Proof of Work, Byzantine General problem, Proof of Stake, Proof of Elapsed Time, Proof of Activity, Proof of Burn.

Contents

3.1 Types of Blockchains

- Four different kinds of blockchain architecture exist :

1. Public blockchains

- Public blockchains are completely decentralised, do not need any authorization to access and are available to anybody. Public blockchains provide every node the same level of access to the chain of blocks of data, the capacity to contribute new blocks of data and the ability to validate the blocks of data that have already been added.
- Public blockchains are being used extensively in the mining and trading of bitcoins in the modern day. You may be acquainted with some of the most well-known public blockchains, such as Bitcoin, Ethereum and Litecoin. The process of generating blocks for network-requested transactions and solving cryptographic puzzles is referred to as "mining" for bitcoin on open blockchains and it is performed by nodes. As compensation for the difficult work that they do, the miner nodes are given a little amount of bitcoin. Miners, in this sense, have a job similar to that of modern-day bank tellers in that they facilitate transactions and are subsequently compensated (or "mined") for their efforts.

2. Private (or Managed) blockchains

- Permissioned blockchains that are administered by a single organisation are known as private blockchains or managed blockchains. Private blockchains also go by the name managed blockchains. The central authority of a private blockchain determines who is allowed to participate as nodes on that blockchain. In addition, the central authority could not always provide each node the same equal right to carry out certain obligations. Private blockchains are only partially decentralised since there are constraints placed on who may access them. Two examples of private blockchains include the business-to-business virtual currency exchange network known as Ripple and the open-source blockchain application platform known as Hyperledger.

Private blockchains are more vulnerable to fraudulent activity and malicious actors, while public blockchains often provide faster validation times for newly added data. Both types of blockchains have their drawbacks, though. These problems were ultimately resolved via the development of consortium and hybrid blockchains.

3. Consortium blockchains

- Consortia blockchains, also known as permissioned blockchains, differ from private blockchains in that, unlike private blockchains, they are not owned by a single entity. Consortia blockchains, in contrast to private blockchains, provide a higher degree of decentralisation, which contributes to an improvement in their level of safety. However, the formation of consortiums may be a challenging task since it requires cooperation between a large number of organisations. This not only creates logistical challenges but also offers a potential antitrust risk (which we will examine in an upcoming article). In addition, some participants in the supply chain may not have the necessary infrastructure and technology to use blockchain technologies. Even those who do have the necessary infrastructure and technology may believe that the initial costs involved in digitising their data and linking it to the data of other supply chain participants are too high of a price to pay.
- R3, a corporate software business, is responsible for the development of a well-known set of blockchain solutions for the financial services industry and other industries as well. CargoSmart, a company operating in the supply chain industry, is responsible for the establishment of the Global Shipping Business Network Consortium. This is a not-for-profit blockchain consortium that aims to digitalize the shipping industry and make it possible for operators in the marine industry to collaborate more effectively.

4. Hybrid blockchains

- Hybrid blockchains are blockchains that are controlled by a single organisation but also have some supervision given by the public blockchain. This supervision is required to carry out specific transaction validations, hence hybrid blockchains are important. A hybrid blockchain like IBM Food Trust is shown below as an example. It was developed with the purpose of enhancing productivity throughout the whole of the food supply chain. In the next and last instalment of this blog series, we will go further deeper into the topic of IBM Food Trust.
- The blockchain technology is now going through a period of phenomenal development on a worldwide basis. Applications of blockchain technology may be found in almost every sector today. This is true regardless of the sector being examined. Blockchain technology is being used to improve enterprises in a variety of industries, including the supply chain, healthcare, logistics, finance and others.
- The primary purpose of blockchain applications is to improve the efficiency as well as the transparency of business procedures. Many companies are beginning to see the value that blockchain technology may bring to the expansion of their business.

What is a blockchain platform ?

- Blockchain platforms are still in their infancy and many of them are just limitations of the underlying blockchain technology. They are put to use in the process of carrying out extensive value transfers over a dispersed network by making use of a growing collection of shared, immutable transactions that have been cryptographically sealed. In addition to a time stamp, each record also contains a link to the preceding transactions. It is a decentralised state transition device that stores transactions in an immutable distributed ledger and follows the development of digital assets.
- Since companies are beginning to study the potential of blockchain by building blockchain apps, the need for blockchain platforms has never been higher than it is right now. According to the findings of a study, the global blockchain sector is expected to see growth of 67.3 % between the years 2020 and 2025, going from USD 3.0 billion in 2020 to USD 39.7 billion in 2025. As a direct result of this, there is an increasing need for the ecosystem that is powered by blockchain to expand in a manner that is both safe and fast. In addition to this, we can notice a surge in the proliferation of blockchain platforms.

How many blockchain platforms are there ?

- There are already at least 1,000 blockchains in existence, distributed among four distinct types of blockchain networks.
- Blockchain's primary purpose is to simplify the process of transferring data from one location to another; nevertheless, there are various platforms available on the market. It has applications in a broad range of different systems, including as supply networks, cryptocurrencies, decentralised exchanges, smart contracts, central bank money and many other types of systems.
- There has been a recent uptick in the number of business applications that make use of blockchain technology. One such organisation is the substantial Depository Trust & Clearing Corporation, often known by its acronym, DTCC. The bulk of the world's $48 trillion worth of assets, which includes stocks, bonds, mutual funds and a variety of other products, are among the 90 million transactions that are tracked by the organisation each and every day.

Therefore, this is applicable to more than simply virtual currency. There are numerous more types of blockchains outside cryptocurrency that should be considered.

There are already over 12,000 currencies that are built on blockchain architecture and the number of active blockchains is growing by the day. In order for us to make a choice, we need to perform some research and weigh our options. First, you will need to ascertain whether or not you need blockchain software.

Do you need a blockchain platform ?

- You may use the decision tree to determine if you need a blockchain platform :

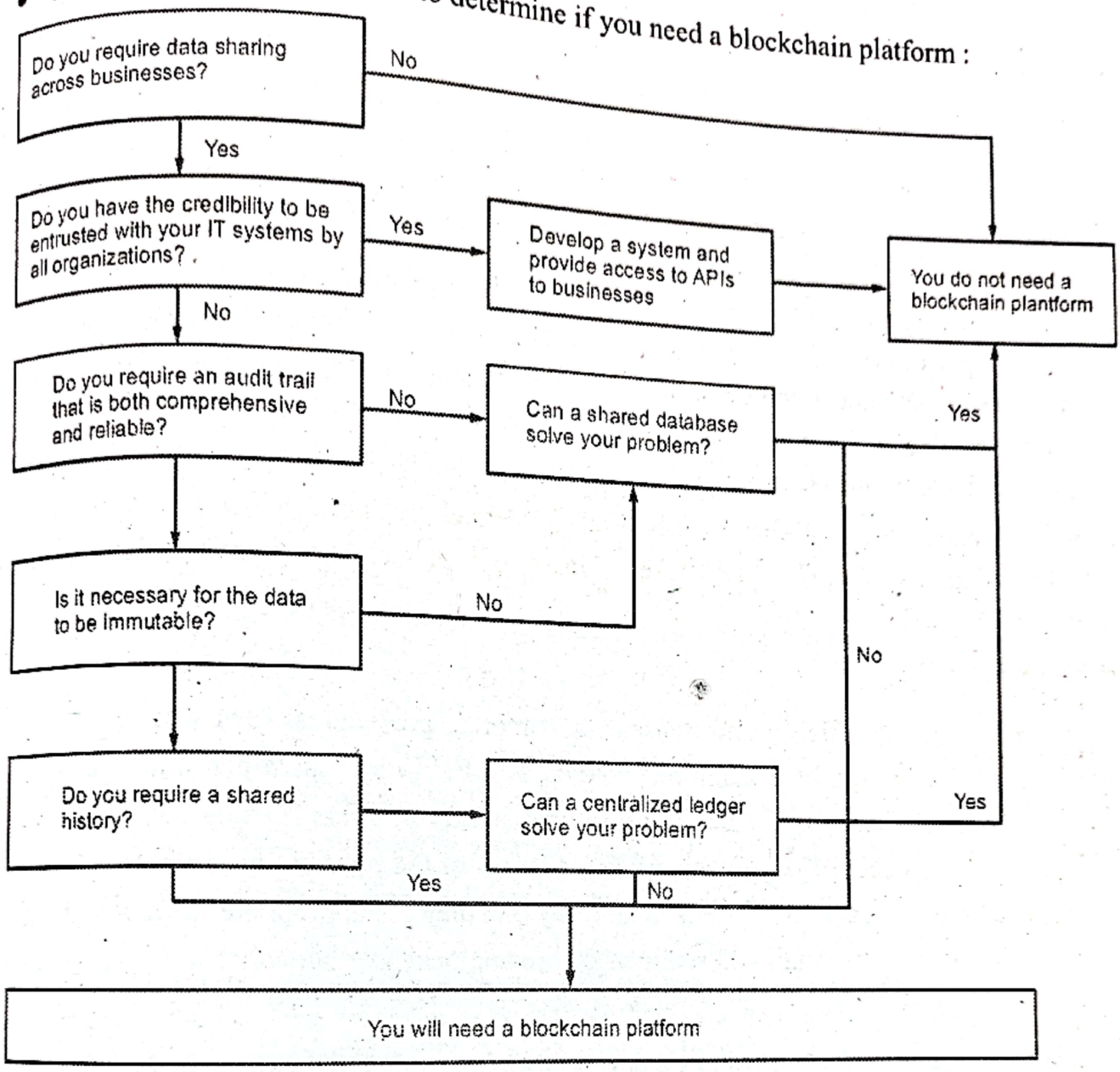


Fig. 3.1.1 Blockchain platform

- Since companies are beginning to study the potential of blockchain by building blockchain apps, the need for blockchain platforms has never been higher than it is right now. According to the findings of a study, the global blockchain sector is expected to see growth of 67.3 % between the years 2020 and 2025, going from USD 3.0 billion in 2020 to USD 39.7 billion in 2025. As a direct result of this, there is an increasing need for the ecosystem that is powered by blockchain to expand in a manner that is both safe and fast. Additionally, this growth may be witnessed in the introduction of platforms that use blockchain technology.

Do you need a blockchain platform ?

- You are welcome to make use of the decision tree in order to establish whether or not you need a blockchain platform:
- If you feel like you need an answer, it's time to do some research! Have no fear, here you will find a list of blockchain systems that will be available in 2022.

3.2 Blockchain Networks : Top Blockchain Platforms in 2022

- In addition to digital currencies like as bitcoin, the blockchain technology may have a range of other applications in the future. The capability of modern technology to improve justice and transparency while at the same time saving companies money and time has an impact on a wide range of sectors, from the fulfilment of contracts to the simplification of administrative procedures.
- There are many more applications for blockchain technology than cryptocurrencies, such as blockchain games. Therefore, let's take a closer look at the most prominent blockchain systems.

IBM blockchain

- In order to establish efficient and transparent corporate procedures, IBM was one of the first corporations to use blockchain technology. The IBM blockchain platform is a well-known resource that is available. Because this platform offers a fully managed and comprehensive blockchain-as-a-service solution, users of the platform have the freedom to combine their blockchain components in any way that they see appropriate. Users are given the ability to construct, install and run their own ownblockchain networks while using this IBM blockchain platform.
- The architecture of the IBM Blockchain development tool has to prioritise user friendliness, speed and the ability to be customised. Additionally, IBM has made investments in a user-friendly interface in order to ease essential operations like as setting up, testing and rapidly deploying smart contracts.

Chainalysis KYT

- The latest phrase to make the rounds in the realm of bitcoin lingo is "Know Your Transaction," an acronym that stands for "Know Your Transaction." This platform offers real-time Application Programming Interfaces (APIs), an easy-to-navigate user interface and industry-leading blockchain intelligence. It assists companies in automating repetitive activities, complying with local and international regulations and using cutting-edge technologies such as DeFi in a secure manner.

Tron

- Tron, an alternative blockchain network with decentralised control, intends to establish a decentralised web. Similar to Ethereum, Tron enables developers to use entire protocols by way of smart contracts stored on the blockchain. More than 2000 transactions can be processed using Tron every second by the vast majority of payment processors, including PayPal. In addition, Tron does not levy any fees on its users for doing transactions.
- The most distinguishing feature of Tron is its exceptional scalability, which gives it the capacity to process thousands of transactions per second. Tron was designed with the high-scale functionality of distributed applications in mind and the Delegated Proof of Stake consensus method was included for better blockchain safety. A decentralised application (DApp) may be installed on the Tron network in a variety of different methods. One of the public chains that is now growing at the quickest rate is Tron.
- The rapid throughput of the Tron network, its excellent scalability and its compatibility with the Ethereum Virtual Machine are the primary contributors to the success of the Tron network (Ethereum Virtual Machine).

Stellar

- Stellar is a decentralised network that makes it easy to store money and transmit money to other people. You may use it to generate, trade and transmit digital copies of a wide variety of currencies, including dollars, bitcoins, pesos and a great many more.
- The majority of financial institutions are already conducting pilot programmes to investigate how well blockchain technology may help them provide services that are open, efficient and safe. Stellar is a blockchain platform that may assist you in the development of financial technology applications, tokens and digital assets that are both secure and highly efficient.
- It is a network that is not governed by any central authority and is used for cryptocurrencies and money. It is not owned by anybody and all it has ultimately belongs to the whole populace. It is capable of processing millions of transactions every single day. To maintain the most up-to-date state of the network, Stellar was developed using blockchain technology, much as Ethereum and Bitcoin.
- Using the Stellar platform, you are able to create your own assets, take part in the exchange of tokens amongst peers and convert currency as you are transferring it.

Ethereum

- Ethereum is a decentralised application platform that was introduced in 2013 by VitalikButerin. It is one of the first and most well-known blockchain networks. It offers a blockchain that is not centralised and has properties similar to those of the bitcoin network.
- On the decentralised platform Ethereum, smart contracts are carried out after they have been created. The reason for its widespread adoption is because it is one of the blockchain systems that is utilised the most often. It also decentralises markets and gives people the ability to build new financial apps. Game settings are also created by it. The primary objective of this technology is to do away with access granted to third parties so that further study may be conducted on financial instruments. Ethereum, which is now one of the most popular blockchain systems, has the biggest group of fundamental protocol developers.

Multichain

- Free variant of the cryptocurrency known as Bitcoin MultiChain is an open source implementation of the Bitcoin blockchain. It is easy to use and can be used to construct individualised blockchains, which may either be public or private. It offers a thoughtfully curated assortment of features and enhancements that were developed with commercial and corporate clients in mind. It would seem that integrating local assets and having the capability to store larger amounts of random data would be beneficial. Consensus-based permitting for consortium blockchains is a novel approach, despite the fact that it relies on consensus as its foundation. Alterations to the chain may be made by modifying the multichain-util text file before to initiating the multichain process. Once the network has been entirely constructed, however, parameters cannot be modified. Therefore, take immediate action.

EOS

- EOS is yet another secure and scalable blockchain platform that can be used to develop decentralised applications. This platform may be looked of as an enhanced version of Ethereum and Bitcoin's blockchains due to the fact that EOS offers a number of capabilities that neither Ethereum nor Bitcoin currently provide.

EOS is a blockchain network that was designed to overcome scalability difficulties. It does this by enabling smart contracts, hosting for decentralised applications (dApps) and decentralised storage of commercial products. You are free to design a financial application using EOS that can carry out transactions in a more timely manner. The creation of executable smart contracts is another option open to developers and it depends on the needs of the company.

- The EOS blockchain gives programmers the ability to build a wide range of apps that can run on the blockchain network to progress their ideas. You may rapidly construct contemporary business solutions, games and financial application software, among other things.

3.3 Hyperledger Fabric

- The provision of a framework for the development of applications or solutions using a modular architecture is the objective of the Hyperledger Fabric project. A plug-and-play integration approach might be used with a variety of different components, such as consensus and membership services. It is possible to use its many modular and flexible design configurations to a wide range of different industrial application scenarios.
- One of the most important aspects of Hyperledger Fabric is its capacity to construct a network that contains other networks. Participating organisations in a fabric network work together, but since most companies would rather keep some aspects of their data confidential, these organisations often maintain distinct connections inside their networks.

Hyperledger sawtooth

- The open-source blockchain project Hyperledger Sawtooth, which is endorsed by both the Linux Foundation and Hyperledger, utilises a cutting-edge voting mechanism known as Proof of Elapsed Time. This approach was developed by Hyperledger. This ground-breaking technology may, with the assistance of hardware-based security technologies, provide "trusted execution environments," which allow programme code to operate in safe fortresses and encrypted computer memory areas. This would be a significant technological advancement.

Klaytn

- Klaytn, a global public blockchain platform, was developed by Ground X, Kakao's subsidiary that focuses on blockchain technology. The blockchain that was developed by Kakao. Due to the fact that Klaytn was developed as a modular network with an adaptable network architecture, it is appropriate for application in business settings. Because of the modular nature of its network design, the Klaytn framework makes it simple for businesses to construct and manage blockchains focused on providing services. Service Chains, which are the essential autonomously running sub-networks of the platform, are the foundation of Klaytn's business-friendly environment. This environment is built on Klaytn. Because of its malleability and versatility, Klaytn may be used in the development of any kind of online service. However, actions that include gambling or speculating on the financial markets are not permitted.

Ripple

- Ripple's enterprise-grade solutions outperform traditional financial services in terms of speed, transparency and cost-effectiveness. This is made possible by the company's decade-long development of its expertise in cryptography and blockchain technology. Ripple customers make use of these technologies in order to purchase cryptocurrencies, make quick payments possible, empower their treasuries, engage new audiences, lower their capital requirements and generate new income.

XDC network

- The XDC Network is a versatile hybrid blockchain platform that joins public and private blockchains together with the use of cross-chain smart contracts in order to give enterprises with both benefits. The XDC Network is a distributed system that functions like a liquid and takes advantage of interoperability. It digitalizes and tokenizes trade transactions, making them more efficient while also lowering reliance on sophisticated foreign exchange infrastructures.

Blockchain platform comparison

- Your requirements are what will define how effective blockchain technologies are for you. However, according to the results of the survey conducted by Gartner, the top five services are as follows :

1. Chainalysis, also known as KYT
2. IBM's Blockchain technology.
3. Ripple
4. Ethereum
5. Hyperledger fabric.

Top blockchain companies : These companies are some of the most successful blockchain ones :

Amazon

- Amazon Web Services is now offering blockchain technology, which makes it possible for businesses to use distributed systems without first having to manually create them from the ground up. This is an excellent strategy for maintaining market dominance in cloud computing, which was Amazon's most profitable business in 2018, resulting in an operational profit of $7.3 billion for the company.

Customers of Amazon Cloud Services include Change Healthcare, the Workday HR management software firm and the DTCC clearing house, to name a few examples of these businesses.

Anheuser-Busch Inbev

- One of the largest breweries in the world is taking part in a test endeavour in which customers may use a barcode scanner on their smartphones to purchase beer from a vending machine and store digital copies of their driver's licences on a distributed ledger. BanQu and the blockchain platform are being used by AB InBev to communicate with local farmers who do not have bank accounts in Africa, which is one of the world's beer markets with the fastest-growing consumer base. Because of this, AB InBev could be able to develop its operations in Africa more quickly and with a greater number of farmers.

Blockchain platforms : Ethereum, Corda.

BBVA

- As part of an agreement worth 170 million euros reached in November of the previous year with the country's grid operator, Red Eléctrica Corporación, Spain's second-largest bank began offering the world's first blockchain-based loan that was syndicated at that time. Additionally, approximately $5 trillion in syndicated loans are distributed throughout the world each year; hence, the openness, safety and efficiency offered by blockchain technology may end up being rather crucial.

3.4 IOTA (MIOTA)

What is IOTA ?

- The ecosystem of the Internet of Things (IoT) makes use of a distributed ledger known as IOTA (MIOTA) to keep track of and carry out transactions involving machines talking to machines and devices talking to devices. The MIOTA money is used by the ledger in order to keep track of all network transactions. IOTA's flagship product is called Tangle and it is a network of nodes that is used for the confirmation of transactions. Tangle is said to be both speedier and more effective than the standard blockchains that are utilised in cryptocurrency transactions. This is according to IOTA.
- IOTA Foundation, a charitable organisation that was responsible for the creation of the ledger, has established collaborations with well-known companies like as Bosch and Volkswagen in order to broaden the application of the platform across connected devices.

Understanding IOTA

- It is anticipated that by the year 2020, there will be billions of devices connected to the internet. During the course of everyday transactions, a device may share data and payment information with a large number of other devices that are part of this Internet of Things (IoT) ecosystem.

- IOTA intends to make the replacement of the currently accepted ways of device transactions the standard. According to the people who developed the ledger, it is a "public permission-less backbone for the Internet of Things" that makes it possible for a wide variety of devices to communicate with one another. To put it another way, this means that it will be accessible to everybody and it will make it easier for connected devices to conduct transactions with one another.
- The developers of IOTA claim that their creation solves a number of problems that are inherent to cryptocurrencies that are based on traditional blockchains. Scalability concerns, problems with network speeds and the fact that a very small number of individuals control the majority of the mining power are among these obstacles. When discussing cryptocurrencies, the term "scalability" refers to the problem of increasing the number of transactions that a blockchain can process without negatively impacting any of the other metrics being measured.
- The bulk of these problems are brought about by an accumulation of unconfirmed Bitcoin transactions on the blockchain. Two of the factors that contribute to the backlog itself are the small block sizes that are used and the difficult tasks that miners have to solve in order to be rewarded with the money. IOTA solves these problems by reimagining the blockchain as Tangle, an innovative system for the organisation of data and the verification of transactions.

History of IOTA

IOTA was established by Sergey Ivancheglo, Serguei Popov, David Snsteb and DominikSchiener, who joined the company at a later time. DominikSchiener came on board later.

In October of 2015, the project's formal announcement was made in the form of a post advertising a token sale that was made on an online bitcoin community. IOTA was first conceived as part of the Jinn project. For the Internet of Things (IoT) ecosystem, the creation of general-purpose processors, also known as ternary hardware, which is technology that is both cost-effective and energy-efficient was the primary objective of the project. During the month of September 2014, Jinn held a crowd sale for its tokens. During the course of the crowd sale, close to one hundred thousand tokens were purchased, bringing in a total of two hundred fifty thousand dollars.

The Jinn tokens got themselves into a lot of difficulty very fast since they were touted as profit-sharing tokens, which are often referred to as security tokens. At the time, initial coin offerings (ICOs) were still in their infant stages and it was unknown to what extent they

were regulated. In 2015, a fresh token sale took place and the previous cryptocurrency was rebranded as IOTA. This time, the tokens were promoted as utility tokens in the marketing materials. Under the new protocol, owners of Jinn tokens will have the opportunity to exchange them in for tokens of similar value. According to David Snsteb, the IOTA project was "spawned" as a result of the Jinn project; hence, it made sense to present IOTA first and then introduce Jinn thereafter.

- IOTA's first transaction was a balance transfer to an address that had all of the MIOTA (the cryptocurrency it utilises) that has ever been mined. This was done so that the network could begin operating immediately. However, according to the sources that have been consulted, a snapshot of the genesis transaction has not yet been seen anywhere on the internet. In the course of distribution, these tokens were sent to other "founder" addresses. It is anticipated that there are a total of 27 quadrillion MIOTAs in the world. The developers of IOTA assert that the total number of MIOTAs "nicely" matches to the highest possible integer value that may be used in JavaScript. Three months after it was first traded on cryptocurrency exchanges, the digital asset known as mIOTA reached an all-time high value of $14.5 billion during the bull market that lasted from 2016 to 2017. Nevertheless, its value gradually decreased, just like that of the vast majority of other cryptocurrencies.

Concerns regarding IOTA

- IOTA has taken the brunt of the criticism for its lack of technical advancement. IOTA's mechanism, like that of the vast majority of cryptocurrencies, is brand new and has not been well tested. The theft of $3.94 million from MIOTA was caused by a phishing attack on the company's network. The IOTA development team responded to the attack by publishing a blog post that detailed how to get a safe seed while using its currency. The article was produced as a response to the breach.
- The process through which the developers of IOTA created their coin is referred to as rolling. In other words, they did not use the common SHA-256 hash function that is used in Bitcoin; rather, they designed their own own encryption method from the ground up. Researchers working on the Digital Currency Initiative at MIT have found that the Curl hash function used by IOTA has severe flaws that need to be addressed. The function always returned the same output, regardless of which of the two inputs was used. A hash function that is not working properly is indicated by the presence of the feature known as **collision**. During the course of their study into the matter, the MIT team said that they believe a malevolent actor may have utilised their technology to do harm to Tangle or steal user funds. The IOTA team has addressed the issue and remedied the problem.

- There is a possibility that IOTA's claims, according to which the use of DAGs will address the scale problems that afflict blockchains, include errors. VitalikButerin, one of Ethereum's co-founders, has cast doubt on the viability of hashgraphs as a solution to scalability issues. Hashgraphs form the basis of Distributed Application Graphs (DAG). According to him, the issue of a blockchain's dependency on computer memory and processing power is not answered by the current iterations of hashgraphs. However, the power and speed of the individual workstations that make up a hashgraph system's network are two factors that restrict the system's capacity to grow.
- IOTA's network implemented a centralised server known as a Coordinator in 2020 with the intention of ensuring the integrity of all transactions. This behaviour has completely undercut the system's claims that it is decentralised, which were made possible by the installation of the Coordinator, which provided a single point of failure. In addition, the slowness of the network has been caused by the absence of parallel processing in systems that are based on Coordinator. However, the IOTA Foundation did plan a future elimination strategy for the Coordinator that they referred to as "The Coordicide."

Future of IOTA

- Even though the market value of IOTA was still much less than what it was in 2017, things seemed to be looking better for this cryptocurrency at the end of the year 2020. It began the year 2020 with a market worth of $446 million, but as of the 19th of December in 2020, that number had increased to more over $900 million. The journey wasn't easy, but the payoff is far higher than 100 %. IOTA stands out among other cryptocurrencies and garners the attention of investors because to its continuous partnerships with large companies and concentration on the development of the Internet of Things (IoT). As of September 28, 2021, the value of IOTA on the market was around $3.2 billion, indicating that it seems to be operating.

ow is IOTA different from bitcoin ?

IOTA's solution to the problems that Bitcoin is experiencing is to do away with a lot of the core notions and geographical restrictions that are associated with a blockchain. IOTA's native money, known as MIOTA, was premined and the network's consensus for approving transactions is handled differently than it is on a blockchain. IOTA's inventors have proposed using a brand-new data structure known as Tangle as a method for arranging numerical representations in the memory of a computer. This mechanism is called Tangle.

- The nonsequential network of nodes that makes up Tangle is technically referred to as a Decentralized Acyclic Graph (DAG). As a direct consequence of this, a single node in a Tangle may serve as a connection to several other nodes. On the other hand, because there is only one path between them, a node cannot refer to itself in that fashion. A conventional blockchain is already a DAG due to the fact that it is a sequentially linked collection. Nevertheless, IOTA's Tangle has a parallel architecture that enables transactions to be processed simultaneously rather than sequentially. This is made possible by Tangle's distributed ledger. The more systems that are linked to the Tangle, the more secure it will become and the more efficiently it will handle transactions.
- A network of computers that are running full nodes, which are responsible for storing the whole history of transactions for a ledger, is required for Bitcoin confirmations and consensus to take place. This approach requires a significant amount of time as well as energy.
- Within the context of Tangle, full node miners are not required. By comparing each new transaction to the two transactions that came before it, the amount of time and memory needed to verify a transaction may be cut down significantly. In the last step of the process, the Proof of Work (PoW) challenge is added to the transaction. This challenge may be solved easily and quickly. The transaction is validated by the IOTA system via the use of a tip selection procedure that makes use of the parameter "confidence." Let's imagine that a contract has previously been approved by 97 different parties. As a result, there is a 97 % likelihood that one of the nodes will finally agree to accept it.
- The significance of a transaction is inextricably linked to the concept of "confidence." As a transaction makes its way through Tangle, its total weight will gradually climb. The more endorsements that a business transaction obtains, the more credibility it has. Once it has been verified and accepted, a transaction is made public throughout the whole network. After the verification of that transaction, a further transaction that has not yet been confirmed may use it as one of the tips it uses to confirm itself.
- This method of transaction confirmation allows MIOTA to be used by a wide variety of machines and devices with varying power requirements since it has a low power consumption and does not incur any costs. Additionally, it does not charge any fees.

R3

- R3 is a financial company that serves as the head of a consortia partnership that is comprised of more than one hundred of the most prestigious financial institutions from around the world. This partnership was formed so that R3 could build and supply distributed ledger technology to the international financial markets. It was founded in 2014 by a consortium of nine financial organisations, including Goldman Sachs, Credit Suisse, JP Morgan and others, with the goal of using blockchain technology in commercial markets.

History

- R3's initial plan was to start a new company, raise $200 million from members and then offer those members a 90 % ownership stake in the company. Nonetheless, it lowered the goal and stated that members would get a sixty percent stake in R3 in November of 2016.
- R3 was harmed during this time period by a series of important withdrawals from the consortium, the first of which was made by Goldman Sachs and further withdrawals were made by Santander and JPMorgan. In response, R3 increased the scope of the cooperation to include ABN AMRO, Fifth Third Bank, Suncorp Group Insurance and Synchrony Financial, which is a supplier of financial services.

R3 has previously been the subject of discussion in connection to the use of the word "blockchain." The company first referred to itself as a "blockchain startup," although this was a misnomer due to the fact that the technology they were developing was a distributed ledger rather than a blockchain. At the beginning of 2017, they revised the language that was seen on their company website in order to correct this issue.

In April of 2017, Calypso Technology and R3 formed a partnership with the intention of providing trade matching services to five different financial institutions. In order to carry out four-zone real-time trade matching testing, the solution utilises R3's Corda platform in conjunction with Calypso's cloud application.

Bryan D'Souza, who had previously worked as an expert in electronic trading for SocieteGenerale, was hired by R3 in the month of August 2019. D'Souza started his career in the financial industry working for R3 Europe as a partner manager.

gal battle with Ripple

Ripple's CEO, Brad Garlinghouse, sent an email to R3's CEO, David Rutter, in June 2017 requesting that the company be released from a contract that Ripple had signed with R3. R3 has been given the authorization to purchase up to 5 billion digital XRP tokens at a price of

$0.0085 per token until September 2019, when the authorization expires. In retaliation, R3 filed a lawsuit against Ripple. Ripple, in turn, filed a lawsuit against R3, stating that R3 had violated the terms of the agreement by refusing to help Ripple in marketing its technology. R3 is now facing both lawsuits. In spite of Ripple's appeals to the contrary, the case was transferred to a court outside of San Francisco. R3 filed an appeal with the courts in Manhattan, New York and Delaware; however, the case was thrown out by the court in Delaware.

3.5 Corda Settler

- In December of 2018, R3 introduced a piece of software known as Corda Settler with the intention of easing the process of conducting cryptocurrency-based transactions across international borders using enterprise blockchains. Corda Settler began adopting XRP as its primary currency despite the fact that the company has previously been involved in legal conflicts with Ripple.

SWIFT integrates with R3

- According to statements made by SWIFT CEO Gottfried Leibbrandt in January 2019, the software platform known as SWIFT GPI, which is utilised by the international financial payments network SWIFT, will be merged with R3. Leibbrandt made the announcement while speaking on the main stage of the Paris Fintech Forum. In a subsequent statement, SWIFT noted that it was now evaluating Corda's capability to handle existing application programming interfaces (APIs), organisational design and other technical characteristics of SWIFT. In addition, SWIFT noted that it was evaluating Corda's ability to link with SWIFT's GPI Link gateway and to monitor payment flows. The integration of R3's Corda with SWIFT GPI was described as a "natural extension" by David Rutter, who also referred to it as "the new standard to settle payments straight across the world." He said that the distributed ledger applications built on the Cordablockchain will be beneficial for SWIFT GPI.

MonetaGo

- In 2019, a young company known as MonetaGo, which develops private blockchains for financial institutions such as banks and other organisations, made the switch from Hyperledger Fabric to Corda.

Jesse Edwards leaves R3

- In March of 2019, R3 lost one of its co-founders when Jesse Edwards resigned from his position with the company. Before he left R3, he was working on a "side fund" for the company that was codenamed "Adroc" (which is Corda spelled backwards). The goal of the fund was to generate between $50 and $60 million and provide assistance to start-up technology companies that were creating Corda platform-based products. Even though R3 and Edwards both claimed that Edwards had been considering starting his own independent venture capital firm to promote Corda adoption globally, it is believed that Edwards left the company due to disagreements over how to encourage investment for technology startups creating software on top of the Corda platform. This is despite the fact that Edwards allegedly left the company due to disagreements over how to encourage investment for tech startups creating software on top of the Corda platform. According to the statement made by the firm, rather than receiving funds from Adroc, it has decided to take what it calls a "internal corporate development role" in order to provide assistance to new companies that are using the Corda platform. According to a statement released by R3, Edwards and R3 came to an agreement over the decision and the latter described him as continuing to be "an investor in and a personal friend of the organisation."

Brazilian Financial Projects

- The Brazilian banks Bradesco and Itau announced in June 2019 that they were working with the company R3 to establish a blockchain infrastructure for use in international commerce and insurance. It was also stated at the time that the Brazilian stock exchange B3 was using Corda for digital identity purposes.

ncorporating DAML

R3 said in June 2019 that it has successfully integrated Corda, Sawtooth from Hyperledger, AWS Aurora from Amazon and the Digital Asset Markup Language (DAML) developed by Digital Asset. This, according to Yuval Rooz, the Chief Executive Officer of Digital Asset, was done in order to enhance the procedures around some of the most popular ledgers that are now in use all over the world.

oldGram Pte. Ltd.

R3 made an announcement in July 2019 about a collaboration between themselves and GoldGram Pte. Ltd., the firm that is responsible for the gold-backed stablecoin known as GoldGramCoin (GGC). The announcement claims that the token will be made available for purchase on R3's Corda platform and will be exchangeable "at any time" for genuine gold that has a purity level of 99.99 %.

- Rutter made a statement on August 1, 2019, indicating that R3 would increase its worldwide staff from 215 to 300, doubling the amount of office space in its London location and recruit 40 more engineers. Rutter made the following statement in reference to the challenges that London would see as a result of Brexit: "London will have immense potential after Brexit." Even while there are undoubtedly certain things that are not yet known for certain, we believe that the city is arranged and positioned in such a way that it will be successful in the years to come. As a result of this, we have the self-assurance necessary to make this substantial and long-term commitment right now. R3 announced in the same month that they would be opening a new office in Dublin as part of their "aggressive recruitment push." The location will be located in Dublin.

How to choose a blockchain platform ?

- Take into account the following criteria while selecting the ideal blockchain platform for any organisation.

Type of blockchain you need

- Before determining which blockchain platform to use, you must first choose the blockchain that is most appropriate for the application you are developing. If you want your users to be verified before they are allowed access to the network, you will need to utilise a network that requires authorization to join. You are thus need to make a decision on the necessity of a permissioned or permissionless blockchain.

Languages supported by the platform

- If you have in-house developers, you should choose a platform that supports the preferred language of those developers. You might search for a system that is compatible with many different kinds of programming languages, such as Python, Java, Javascript and others.

Security

- Numerous organisations are tasked with the duty of safeguarding sensitive data and failure to do so may have negative consequences. To ensure that there are no lapses in security, use a platform that provides continuous security monitoring. EOS and Hyperledger Fabric are now two of the most well-known and widely used platforms for controlling security.

Smart contracts functionality

- Some blockchain systems do not provide support for the use of smart contracts, which are useful tools for validating, enforcing or starting operations on the blockchain network.

3.6 Consensus in Blockchain

- The process by which the peers of a blockchain network come to an agreement on the present state of the data is referred to as consensus. This is accomplished via consensus procedures inside the Blockchain network by building reliability and trust in the system.

Why blockchains need consensus mechanisms ?

- Consensus mechanisms serve as both the basis for and the primary layer of protection for all blockchain-based cryptocurrencies. Before we can go into the many mechanisms that lead to consensus in blockchains, we need to first define what it means for a blockchain to "achieve consensus."
- Transactions may be recorded on a blockchain, which is a digital ledger that is distributed, decentralised and often accessible to the public. Before being added to the chain, each of these transactions is initially represented as its own distinct "block" of data, which has to be verified by an independent peer-to-peer network before it can be added. This method addresses the problem of "double-spending" and contributes to the defence of the blockchain against fraudulent activity.
- Blockchain networks such as Bitcoin and Ethereum make use of procedures known as consensus in order to guarantee that all users, also known as "nodes," are in agreement over a single version of history (also known as consensus protocols or consensus algorithms). These processes are an attempt to enhance the fault tolerance of the system.

3.7 Consensus Algorithm

- A consensus algorithm is a mechanism that enables all of the peers in a Blockchain network to reach a mutual understanding on the present state of the distributed ledger. In this way, consensus algorithms achieve dependability in the Blockchain network and develop trust among unidentified peers in an environment of distributed computing. The consensus process, in its most basic form, guarantees that each new block added to the blockchain is the only version of the truth that has been accepted by every node. This is the only version that can be considered authoritative. The consensus protocol for the Blockchain includes a variety of specific aims, some of which are consensus, collaboration, equality for all nodes and the need that all nodes participate in the consensus process.

hat are consensus mechanisms ?

A consensus method is used inside a network to determine which transactions on a blockchain are valid and which are not. This determination is made by a group of peers, often known as nodes. To bring about this agreement, we use methodologies that are conducive to reaching consensus. These sets of rules contribute to the protection of networks against malicious activity and attacks by hackers.

There are several unique types of consensus procedures that may be used, depending on the blockchain and how it is put to use. The purpose of ensuring that records are correct and true is served by all of them, despite the fact that they differ in terms of the amount of energy they use, the level of security they provide and the scaleability they provide. The following is an overview of some of the most common consensus techniques that distributed systems utilise to gain agreement.

Types of consensus mechanisms

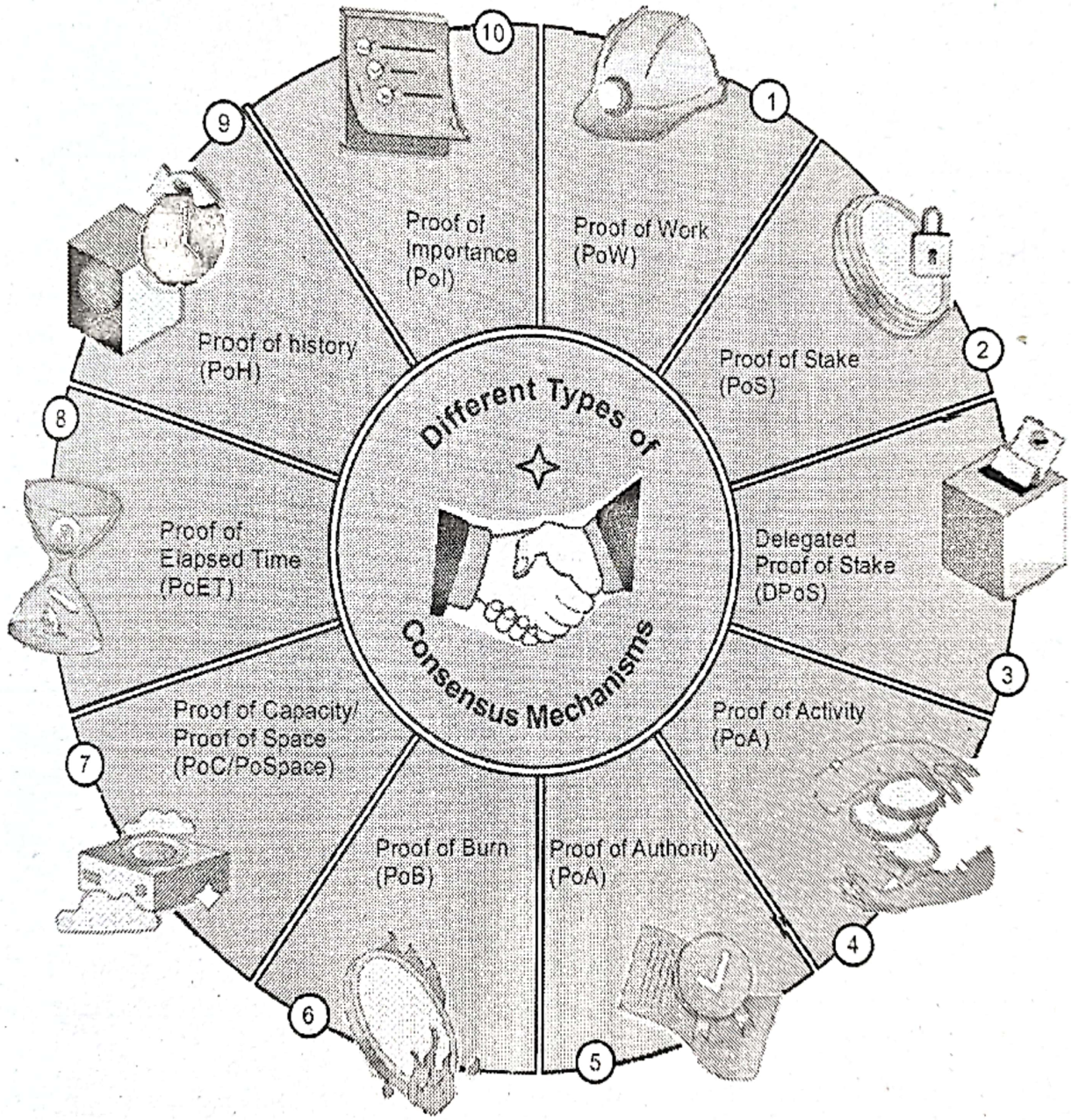


Fig. 3.7.1 Types of consensus mechanisms

3.8 Proof of Work (PoW)

- Proof of Work (PoW), the first consensus mechanism ever established, is used by Bitcoin, Ethereum and a number of other public blockchains. PoW was the first consensus mechanism ever devised. Even though it has a number of scalability issues, it is often thought of as being the most reliable and secure of all the consensus systems. In spite of the fact that the term "proof of work" was first used in the early 1990s, Satoshi Nakamoto, the developer of Bitcoin, was the first person to employ the notion in relation to digital money.
- Users compete against one another in Points of Work to see who can solve the most challenging computational challenges using the most powerful machines. The first user to produce a hash with 64 digits will be granted the right to create a new block and verify transactions. This authority will also belong to them. In addition, the miner who successfully completes the block is entitled to a "block reward," which is a predetermined amount of bitcoin.
- The Proof-of-Work algorithm is notorious for having notoriously high operational costs because to the considerable amount of energy and computing power that is required to generate new blocks. This presents an obstacle for new miners to overcome, which raises difficulties with centralised control and the scalability of the system.
- In addition to that, the charges are not the only item that are costly. The primary argument against PoW is that its power consumption has a negative impact on the environment. As a direct consequence of this, a lot of individuals have started using consensus techniques that are less resource-intensive and more environmentally friendly, such as Proof of Stake (PoS).

3.9 Proof of Stake (PoS)

- As its name suggests, the staking technique is the foundation of this well-liked strategy to reaching agreement. Miners participating in a proof of stake (PoS) system are required to put a certain amount of digital currency "at risk" in order to be eligible for selection as a validator via a random process. In a manner similar to a lottery, the more coins you risk, the better your odds of winning will be in this process.
- The Proof of Stake (PoS) system only uses transaction fees to compensate contributors, in contrast to Proof of Work (PoW), which uses block rewards (newly generated currencies) as an incentive for miners.
- It is believed that PoSis a more robust and environmentally friendly alternative to PoW that is also more secure against an attack of 51 %. The Proof-of-Stake (PoS) technique has come under fire for its propensity to foster centralization since it gives an advantage to organisations who own a greater number of tokens. Tezos, Cardano (ADA) and Solana (SOL) are three examples of well-known PoS systems (XTC).

Delegated Proof of Stake (DPoS)

- Delegated proof of stake or DpoS for short, is a variation on the proof-of-stake (PoS) consensus technique that employs a voting strategy that is based on reputation. The users of the network participate in a "vote" to choose "witnesses," who are also frequently referred to as "block producers," to watch over the network on their behalf. Transactions on a blockchain may only be validated if they are attested to by witnesses who have received the maximum number of votes.
- Users put their tokens at risk by contributing them to a pool in order to vote. When tabulating the results of the vote, the amount of investment made by each voter is included in; the higher the voter's stake, the more influence they have over the outcome. The who voted for the chosen witnesses are eligible to get a reward once those witnesses successfully verify a block's worth of transactions.
- The most credible witnesses are constantly at risk of being disregarded in favour of other candidates who are seen as being more trustworthy and get a greater number of votes. They run the risk of being thrown out of office if they fail to carry out their responsibilities or make an effort to verify fraudulent transactions. This incentivizes witnesses to always tell the truth, which helps to preserve the reliability of the blockchain.
- Although it is not as widely used as PoS, many people feel that DPoS is superior thanPoS in terms of efficacy, democracy and financial inclusion. It is used by the cryptocurrenciesLisk (LSK), EOS.IO (EOS), Steem (STEEM) and BitShares (BTS) (ARK).

3.10 Proof of Activity (PoA)

- Proof of activity is a term that refers to a combination of the PoW and PoS consensus techniques (PoA). Both the Espers (ESP) and Decred (DCR) blockchain projects utilise it in their respective implementations.
- Mining in PoA systems begins, much like mining in PoW systems, with competitors using a significant amount of processing power in an effort to solve a difficult mathematical problem. When the block has been successfully mined, the system transitions to resembling PoS. At this point, the block header that was successfully constructed is sent to the PoA network. The newly created block is subsequently subjected to a randomised validation process, during which a number of validators attest to the correctness of the hash. In a manner analogous to PoS, a validator's chances of getting selected improve in proportion to the amount of crypto that they own. After the block has been signed by all of the designated validators, it is added to the blockchain network and made accessible to be used in recording transactions. The block rewards are then divided between the validators and the miners.

- The Proof-of-Authority (PoA) system was developed to combine the best features of Proof-of-Work (PoW) and Proof-of-Stake (PoS) while avoiding the shortcomings of both of these systems. However, it has been criticised for its energy-intensive mining phase and built-in favouritism towards validators who possess more coins.

3.11 Proof of Authority (PoA)

- Proof of authority (PoA), not to be confused with proof of activity (also known as "PoA"), selects its validators based on reputation. Proof of activity (also known as "PoA") is a synonym for PoA. In 2017, Ethereum co-founder and former Chief Technology Officer Gavin Wood proposed a variation on the Proof-of-Stake model.
- In Proof of Authority, validators do not risk their own currency. Instead, in order for them to be able to verify blocks, they are need to put their reputations at stake. In contrast to this, the majority of blockchain systems do not typically need users to reveal their identities at any point throughout the transaction process.
- Because it requires so little processing power, this strategy uses a lot less resources than some of the methods that came before it, like PoW. It is also among the more economical options, which makes it a widely chosen solution for private networks like JP Morgan, among others (JPMCoin). Two further PoA-based projects are currently in development: the EthereumKovantestnet and VeChain (VET).
- Because only a limited number of users are permitted to join the network, the decentralisation feature, although very scalable, suffers as a result. Because validators have to be recognised in order for transactions to take place, there is an increased risk of both fraud and manipulation by a third party.

3.12 Proof of Burn (PoB)

- Proof of burn is an alternative to Bitcoin's Proof of Work method that is said to be more ecologically friendly (PoB). Miners are granted the power to mine a block in PoB if they have "burned" (destroyed) a certain quantity of tokens in a verifiable manner. Specifically, this involves sending the tokens to a "eater address," from which they cannot be recovered or utilised in any way. The greater the number of coins that are burned, the greater the chance that a person will be picked at random.
- In contrast to Proof-of-Stake mining, in which miners have the option to sell their locked funds or reclaim them in the event that they leave the network, burned coins cannot be reclaimed. This technique of encouraging miners to commit over the long run by requiring

them to give up short-term gains in return for the ability to produce new blocks for the rest of their lives encourages miners to commit. Coins become more rare as a result of burning, which has the dual effect of lowering inflation and raising demand.

- Proof-of-burn is a consensus mechanism that is used by several cryptocurrencies, including Factom, Counterparty and Slimcoin (FCT).

3.13 Proof of Capacity / Proof of Space (PoC / PoSpace)

- Proof of capacity (PoC), also known as proof of space (PoSpace), is a method of mining that awards mining rights based on the amount of accessible space on a miner's hard drive. This differs from the majority of the mining methods that came before it, which awarded mining rights based on the amount of computing power or coins staked.
- The procedure known as "plotting" is used by miners in PoC in order to generate a list of all of the possible hashes in advance. After then, these plots are saved on a hard drive for later use. When a miner has greater storage capacity, there are more viable options available to them. The greater the number of possible answers, the higher the possibility that you will have the correct hash combination and hence be eligible for the payout.
- Because it does not need any expensive or specialised equipment, Proof of Concept makes it feasible for the average person to participate in a network by eliminating barriers to entry. Because of this, it is an alternative to some of the more generally used consensus techniques that are covered in this article that is more decentralised and requires less energy than those other approaches. As a result of the lack of widespread adoption of the system by developers so far, there are concerns that it may be susceptible to attacks by viruses. Currently, the method is being implemented by the cryptocurrenciesSignum (SIGNA), formerly known as Burstcoin (BURST), Storj (STORJ) and Chia (XCH).

3.14 Proof of Elapsed Time (PoET)

- Proof of elapsed time, often known as PoET, is a method that is frequently used on permissioned blockchain networks. This method makes use of trusted computing in order to enforce random waiting times for block creation (those that demand participant identification). It was developed by Intel at the beginning of 2016 and is distinguished by the fact that it is based on a singular set of CPU instructions known as Intel Software Guard Extensions. (SGX).

PoET is a consensus technique that is based on timelotteries. In order for PoET to function, different wait durations are randomly distributed among all of the network nodes. During this period of time, each of these nodes goes into a "sleep" state and awaits further instructions. The individual who awakens first and has to wait the least amount of time before receiving their mining rights is the one who gets to keep them. By ensuring that every participant in the network has an equal chance of victory via the use of randomization, fairness is maintained inside the network.

The process of reaching a consensus with PoET is scalable, very effective and uses less resources than other methods. Sawtooth, which is developed by Hyperledger, makes use of it.

3.15 Proof of History (PoH)

Proof of history, sometimes known as PoH, provides proof of events that took place in the past, as the name suggests. The Proof-of-Hand (PoH) technique developed by Solana makes it possible to directly include "timestamps" into a blockchain, therefore independently validating the gap between transactions.

Using a sequential-hashing verifiable delay function, such as SHA-256, makes it feasible to use this timestamping approach (VDF). It does this by using the result of a transaction as the input for the following hash, which enables everyone to easily understand which event took place in which sequence. Since the VDFs can only be solved by a single CPU score, PoH is able to considerably reduce the processing weight of the blockchain. As a result, it is both speedier and more energy-efficient than the majority of his competitors.

Due to the fact that Solana is the only organisation that now employs it, PoH has not yet been subjected to comprehensive testing.

3.16 Proof of Importance (PoI)

The procedure known as "harvesting" is used by Proof of Importance (PoI), which was first developed by NEM (XEM), to choose its miners based on a set of predetermined criteria. Typical factors include the number and quantity of transactions that have taken place over the last 30 days, the amount of money that has been vested and network activity. The significance score that is assigned to nodes is derived from these components to the greatest extent possible. The higher your score, the greater the possibility that you will be picked to harvest a block and get the transaction fee that is linked with it.

PoI is related to PoS, but it avoids PoS's tendency to automatically reward the affluent by taking into account members' overall network support. This is how PoI avoids being used by scammers. As a consequence of this, the simple act of putting a large POI bet does not guarantee that you will succeed in winning the block.

Review Questions

Q.1	Explain the concept of proof of work ? **(Refer Section 3.8)**	**(5 Marks)**
Q.2	Explain types of blockchain. **(Refer Section 3.1)**	**(6 Marks)**
Q.3	Explain blockchain networks. **(Refer Section 3.2)**	**(5 Marks)**
Q.4	Explain hyperledger fabric ? **(Refer Section 3.3)**	**(6 Marks)**
Q.5	Explain IOTA(MIOTA). **(Refer Section 3.4)**	**(6 Marks)**
Q.6	Explain cordra setller. **(Refer Section 3.5)**	**(6 Marks)**
Q.7	Explain consensus in blockchain. **(Refer Section 3.6)**	**(5 Marks)**
Q.8	How to choose blockchain platform ? **(Refer Section 3.5)**	**(5 Marks)**
Q.9	Why blockchain need consensus mechanism ? **(Refer Section 3.6)**	**(6 Marks)**
Q.10	Explain the types of consensus mechanism. **(Refer Section 3.7)**	**(6 Marks)**

❑❑❑

Unit IV

4 Cryptocurrency - Bitcoin and Token

Syllabus

Introduction, Bitcoin and the Cryptocurrency, Cryptocurrency Basics Types of Cryptocurrency, Cryptocurrency Usage, Cryptowallets: Metamask, Coinbase, Binance.

Contents

4.1 Bitcoin

4.1.1 Introduction

- Bitcoin is a cryptocurrency and a digital payment system invented by an unknown programmer or a group of programmers, under the name Satoshi Nakamoto. It was released as open-source software in 2009. The system is peer-to-peer and transactions take place between users directly, without an intermediary. Since the system works without a central repository or single administrator, bitcoin is called the first decentralized digital currency.
- These transactions are verified by network nodes and recorded in a public distributed ledger called a blockchain. Besides being created as a reward for mining, bitcoin can be exchanged for other currencies, products and services in legal or black markets.
- Bitcoin is a form of digital currency, created and held electronically. No one controls it. Bitcoins are not printed, like dollars or euros - they are produced by people and increasingly businesses, running computers all around the world, using software that solves mathematical problems. However, bitcoin's most important characteristic and the thing that makes it different to conventional money, is that it is decentralized. No single institution controls the bitcoin network. This puts some people at ease, because it means that a large bank can not control their money.
- As of February 2015, over 100,000 merchants and vendors accepted bitcoin as payment. According to a research produced by Cambridge University in 2017, there are 2.9 to 5.8 million unique users using a cryptocurrency wallet, most of them using bitcoin.
- A software developer called Satoshi Nakamoto proposed bitcoin, which was an electronic payment system based on mathematical proof. The idea was to produce a currency independent of any central authority, transferable electronically, more or less instantly, with very low transaction fees.
- This currency is not physically printed in the shadows by a central bank, unaccountable to the population and making its own rules. Those banks can simply produce more money to cover the national debt, thus devaluing their currency. Instead, bitcoin is created digitally, by a community of people that any one can join. Bitcoins are 'mined', using computing power in a distributed network.

4.1.2 Significance of Bitcoin

1. Bitcoins can be used to buy merchandise anonymously.
2. In addition, international payments are easy and cheap because bitcoins are not tied to any country or subject to regulation.
3. Some people just buy bitcoins as an investment, hoping that they will go up in value.

4. Around the world, people are using software programs that follow a mathematical formula to produce bitcoins.
5. The mathematical formula is freely available, so that anyone can check it.
6. Small businesses may like them because there are no credit card fees.
7. The software is also open source, meaning that anyone can look at it to make sure that it does what it is supposed
8. Conventional currency has been based on gold or silver. Theoretically, you knew that if you handed over a dollar at the bank, you could get some gold back(although this did not actually work in practice). But bitcoin is not based on gold; it is based on mathematics.

4.1.3 Working of Bitcoin

- Since no bank is involved so when a person A transfers bitcoin to another person B, all the information is recorded in a public ledger (block), known as the Blockchain. The new block is added to a ledger at every 10 minutes. This ledger records all of the transaction that have taken place since the last ledger.
- Bitcoin is a decentralized digital currency that is exchanged between two parties without involving intermediaries like banks or other financial institutions. As defined in a whitepaper released by the hidden inventor of Bitcoin, Satoshi Nakamoto, Bitcoin is "a purely peer-to-peer version of electronic cash that would allow online payments to be sent directly from one party to another without going through a financial institution".
- Bitcoin achieves elimination of intermediaries with the help of its underlying technology, blockchain.
- Currently if you have to transfer funds to someone, one of the possible ways is by giving cash or alternatively use a trusted intermediary (example, a bank). Both the mechanisms, whether it be physical cash (with the central bank of the country as the guarantor) or electronic transfer, involve an intermediary (in the later case, a bank or another financial institution). When intermediaries are involved, there are transaction costs.
- The blockchain technology helps achieve elimination of intermediaries is by replacing trust that intermediaries bring to the table with cryptographic proof by the use of CPU computing power.
- This cryptographic trust is built into Bitcoin through a wallet, a public key and a private key in the program.Anyone can create a Bitcoin wallet for free by downloading the Bitcoin program. Each wallet contains a public key and a private key. The public key is like an address or an account number via which any person can receive Bitcoins. A private key is

like a digital signature via which a person can send Bitcoins. The name suggests that private keys should be only held and known by the owner and public keys can be shared with anyone for receiving Bitcoins. That is where you would have heard in the news about Bitcoins being lost either due to a private key not being accessible or stolen by hackers. Owners of Bitcoin addresses are not explicitly identified but all transactions on the blockchain are public.Since the inception of Bitcoin in 2009, each and every transaction that has occurred is stored in a ledger, which is considered immutable, non-tamperable and irreversible.

- One can send bitcoins digitally to anyone who has a bitcoin address anywhere in the globe. One person could have multiple addresses for different purposes - Personal, business and the like. Receivers can get to spend them within minutes of receiving the coins. Once given away, like currency, there is no getting them back, unless the receiver decides to give them to you. A bitcoin is not printed currency but is a non-repudiable record of every transaction that it has been through. All this is part of a huge ledger called the blockchain.
- Bitcoin transactions are verified via telecommunication network nodes through cryptography and are then recorded in a decentralized distributed ledger called blockchain. This is one of the distinguishing aspects of Bitcoin from some other crypto assets, where there is centralized exchange (like the stock exchange) through which all transactions need to be routed or validated.
- To understand Bitcoin, one needs to understand the underlying structure, the manner of operation of the Bitcoin ecosystem and the extent of usage of the same in India.
- For transferring bitcoin, person A have its own private key of his own wallet and gets the public key of person B wallet. Every bitcoin wallet can have one or many public keys that can be distributed to everyone but can have only one private key which only the owner of that wallet can know. Both the keys are very long (27-51 characters long) and it's impossible that two wallet have the same key combination.
- In India, currently zebpay and Uno coin are the two best apps. According to zebpay, daily more than 2500 people opening account. That is incredible feat. You can buy bitcoins through IMPS/NEFT/RTGS from an Indian BankAccount. Current price ₹ 2,00,564. Download zebpay through this link http://link.zebpay.com/ref/REF67....
- Alternately you can buy Bitcoins from International Exchanges like CEX.IO, Lake BTC, Hit BTC using your Credit Card / Debit Card / Wire Transfer / Paypal / Skrill. The international exchanges are reputed companies having huge turnovers and use bank grade security.

4.1.4 Bitcoin Economics

One bitcoin is worth roughly about $1,200 now. An early investor in Snapchat has been quoted on the Web as saying that by 2030, the value could be as high as $500,000. One of the reasons that could prompt you to buy a bitcoin today is not so much to use it for payment online but as an investment.

Urban legend has it that someone who was doing a thesis on cryptocurrency bought 5,000 bitcoins for $27 in 2009. Do the math for the value today! And unlike traditional currency that is inflationary in nature, the bitcoin is a deflationary currency. In other words, if there are only so many bitcoins in use and the demand for those rises, the value of a bitcoin would, logically, rise It is debatable whether Bitcoin is a currency at all and why any country would want to replace it with their existing currency as Bitcoin does not have any intrinsic value of its own.

By definition, a currency is "a system of money in general use in a particular country," or "the fact or quality of being generally accepted or in use." Currently, there is some traction in the number of companies using Bitcoin as a mode of payment, however, no major country or economy has accepted it as money in general use. An exception is El Salvador, which adopted Bitcoin as a legal tender in September 2021 and became the first country to do so. One of the important reasons for the remarkable evolution of Bitcoin is the tightening of the Know Your Customer (KYC) and Anti-Money Laundering (AML) regulations by banks and financial institutions. There is now a much greater cross-border exchange of information between the countries about the transactions through the banking system.

As a result, it is also claimed that Bitcoins are widely used as a parallel mechanism for the transactions, which would otherwise be illegal in several countries. Another important aspect is the acceptability of Bitcoin as a global payment mechanism, which is not linked to any particular country's currency and hence, not directly impacted by the developments within a particular country. A bitcoin is generated when an entity, i.e. a person or a business, uses software power to solve a mathematical puzzle that makes the blockchain more secure. The difficulty level of solving the problem is high enough to ensure that it takes time to do it. When you send a bitcoin to a receiver, the transaction is included in the blockchain and broadcast to the network. The blockchain ensures that the same bit coin is not spent twice by the same user. A computer network validates the transaction using algorithms so that the transaction becomes unalterable. Once validated, the transaction is added to others to create a block of data for the ledger.

- We can invest in bitcoin mining. That is the best way to earn but setting up mining serve in India is simply not feasible. Mining requires many hardware tools which are ver expensive. Along with the hardware you also require cheap 24 × 7 electricity, high spe Internet and other factors which are not possible in India.
- Without mining if you buy one bitcoin at a price of$1,000 the price of bitcoin will fluctuat and it can go as high as $10,000 so you make a profit of $9,000.$9,000 in almo 1 or 2 years is a good deal. I know I'm bit to exaggerating but you can never know how th prices will fluctuate as it can even go beyond $10,000. So there should be some possibl way to double sure your investment i.e. to gain both from bitcoin mining and trading. Many Indian now going towards cloud mining techniques. Cloudmining is the easiest method and gain bitcoin is an Indian company which pays you 10 % on your investment every month for 18 months.
- Suppose, you have invested 1 BTC now (currently price of 1 BTC is $1,000) .You will ge a return of 10 % on your investment i.e. 10 % of 1 BTC is 0.1 BTC. After 18 months you get 0.1 × 18 = 1.8 BTC. Suppose price of bitcoin in 1.5 years (18 months) is $10,000. Revenue will be 1.8 × $10,000 = $18,000. Profit = $18,000 – $1,000 = $17,000.

4.1.5 Bitcoin Mining

- In the Bitcoin ecosystem, there is a network of miners who use their CPUs to process transactions.
- Once a user who intends to send Bitcoin enters the public address, number of Bitcoins to be sent and affixes the private key to generate signature, the encrypted information is then sent to the network of miners who are given the task to verify whether there is sufficient balance to transfer and authenticate the transaction.
- The faster the CPU of the miner, the greater are the chances that they will verify and that miner gets rewarded in Bitcoins for facilitating the transfer.
- Here the miner's job is only to provide CPU power, which automatically runs the Bitcoin program to validate Bitcoin transfers. There is no manual intervention by the Bitcoin miner.
- Once the transaction is processed by a Bitcoin miner, this number of transactions is then broadcasted to the network of miners who get the copy or download of the same block.
- People compete to "mine" bitcoins using computers to solve complex math puzzles. This is how bitcoins are created. Currently, a winner is rewarded with 25 bitcoins roughly every 10 minutes. Bitcoin Mining is Just like anyone can join the Internet, anyone can help to verify and record payments into the block chain. This process is called mining. In mining, users offer their computing power. Miners are rewarded with newly created bitcoins and transaction fees. Currently, miners receive 12.5 bitcoins every 10 minutes. This halves every 4 years.

Though each bitcoin transaction is recorded in a public log, names of buyers and sellers are never revealed - only their wallet IDs. While that keeps bitcoin users' transactions private, it also lets them buyor sell anything without easily tracing it back to them. That's why it has become the currency of choice for people online buying drugs or other illicit activities

These blocks through a timestamp mechanism are stored in a sequential or chronological order forming a blockchain. Each miner in the network is supposed to have the updated and complete copy of the ledger or the blockchain if they want to facilitate transfer and earn Bitcoins.

The program is built in such a way that the ledger or the blockchain is automatically updated.

As per the original whitepaper on Bitcoin, the probability of hackers tampering the blockchain is next to zero due to the copy of updated ledger each miner carries. If someone is trying to tamper or hack the ledger by any means to gain unfair advantage, then immediately the miner is considered invalid and fails to process transactions until they have a copy of the untampered ledger.

4.1.6 Legality in India

Now coming to legality part, it is definitely not illegal in india and we got zebpay which allows us to buy and sell bitcoin legally. Legal status, tax and regulation. The legal status of bitcoin varies substantially from country to country and is still undefined or changing in many of them. While some countries have explicitly allowed its use and trade, others have banned or restricted it. Regulations and bans that apply to bitcoin probably extend to similar cryptocurrency systems. In india on 28 December 2013, the Deputy Governor of the Reserve Bank of India, K. C. Chakrabarty, made a statement that the Reserve Bank of India had no plans to regulate bitcoin.

On the regulatory front, India saw two major developments this year :

- In February 2022, in India, the Indian government proposed to introduce taxation on virtual digital assets, which would imply a taxation system for cryptocurrencies, but there is no clarity on whether the Indian government finds cryptocurrencies legal either as "asset" or "currency". India's Finance Minister has categorically stated since then that "taxing cryptocurrencies doesn't mean legalizing them." This indicates the government is still evaluating all the factors associated with cryptocurrencies and it would be early to make any assumptions on their legality.

- Reserve Bank of India won't regulate virtual currency Bitcoin, yet [Economic Times]. The Reserve Bank of India, which has its hands full trying to arrest the slump in the value of the rupee, will first seek to understand Bitcoins - which have attracted regulatory gaze in the United States before seeking to bring it under its purview."As of now we are watching and learning about the developments in Bitcoins but are not regulating it," an RBI spokeswoman wrote in an e-mailed response. In a note published in June, the central bank acknowledged that virtual currencies "pose challenges in the form of regulatory, legal and operational risks."
- Founded in 2009, Bitcoins are digital currencies that are not issued by any central bank or other centralized authority but by an open source software through a process called 'mining.' Over80 % of bitcoins are traded on a Tokyo-based digital currency exchange called Mt. Gox, where one Bitcoin was trading at $107. A federal judge in the United States ruled that Bitcoins are real money and can be regulated under that country's law while ruling on a case related to ponzi scheme. While the RBI is taking a wait-and-watch attitude, India-based Bitcoin users and entrepreneurs, whose number has grown steadily over the past two years, are concerned over the absence of guidelines that govern the use of the virtual currency. They are, however, hopeful that Indian regulator would recognize it as a legitimate currency. "Bitcoin is emerging in India and it would benefit the users if RBI could regulate it or take ownership of it," said Benson Samuel, a Bangalore-based Bitcoin enthusiast who runs a knowledge portal on the virtual currency.

4.2 Cryptocurrency

4.2.1 Cryptocurrency Basics

- We are living in a digital world where everything is heading towards pure digitalization. i.e.
- Most of our works are becoming paperless. Even money transfer and investments are paperless.
- One of such digital payment sector is cryptocurrency.
- Crypto currency is a digital money or virtual money. That is money is not available physically and it is very secure. In simple words we can say cryptocurrency is a money exchange process. Most popular example for cryptocurrency is Bitcoin. It is the first ever introduced cryptocurrency. Some other examples for cryptocurrencies are Ethereum, XRP. It is not possible to counterfeit or double spend because it is secured by cryptography. It is an decentralized process hence they are not controlled by anyone. Cryptocurrencies are tax free and they are not insured too. Government or banks are not responsible for cryptocurrency. Even many countries have banned cryptocurrency.

4.2.1.1 Purpose and Working of Cryptocurrency

- The main purpose of cryptocurrency is to reduce the risk involved in traditional currency. It is very easy to use. We can access it anywhere and anytime. All we need is a smart phone and a good net connection.
- In cryptocurrency the power and the responsibilities are in hands of the currency holder. They help in solving real world problems.
- It uses block chain technology. It is a very brilliant technology because both the buyer and seller details are viewable to each other and so no broker is needed.
- For example if we need to buy a share from a stock market we can do it easily with the help of a broker. We will confirm the exchange order and then we will receive the shares. We do not need to contact the seller in person.
- The reason for choosing a broker is we do not know if the seller has the stock or not. This is known as principle of novation. In cryptocurrency, involvement of third person is not needed because all the transactions are stored in a common location and it is viewable. The identity of the person who made the transaction is encrypted. Most of the cryptocurrencies are made using a process called mining. Mining is nothing but an algorithm. It is the process of adding transaction to the blockchain ledger via nodes on the network with the consensus achieved through a proof of system. But not all cryptocurrencies are made by mining. Some currencies are created using various other techniques such as tokens.

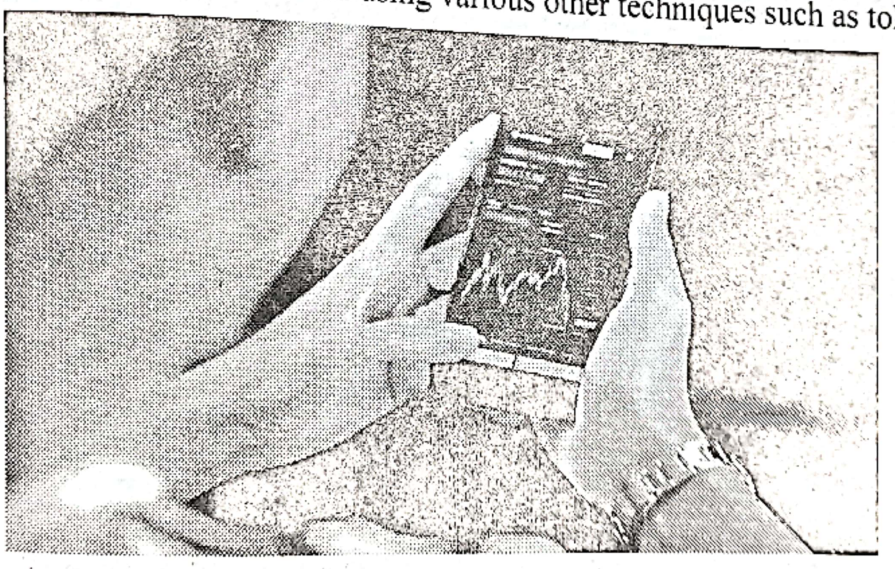


Fig. 4.2.1 : Cryptocurrency secure process

- Transactions done using cryptocurrencies are highly secure. There is a chance of earning huge amount when compared to mutual funds or share market but it comes with a high risk. Because the volatility of cryptocurrency is very high. Either we earn high return or lose what we have.

- Every coin has two faces, similarly there are both good and bad about investing in cryptocurrency. One should know both the faces before investing. Let us consider only the good side about investing
- Returns may be massive and everything is instant. So you can buy or sell when there is a fluctuation in the market very easily.
- There is a cold wallet storage option where we can control our own private key. This key helps us to access our coin in the blockchain.

4.2.1.2 Disadvantages of Cryptocurrency

- Major problem of cryptocurrency is :
  1. The lack of regulation.
  2. Volatility : For example in 2017 cryptocurrency like Bitcoin increased suddenly upto 1000 percentage and then came down.
  3. Fear about hacking and scams due to the reason of digitalisation.
  4. Security issues : There are stories where exchanges are hacked and peoples who held coins in those exchanges lost everything.
  5. There are lot of peoples with less experience and knowledge which leads to huge loss.
  6. Since it is fully digitalized there is a technical difficulty such as network issues and so on.

4.3 Types of Cryptocurrency

1. Bitcoin

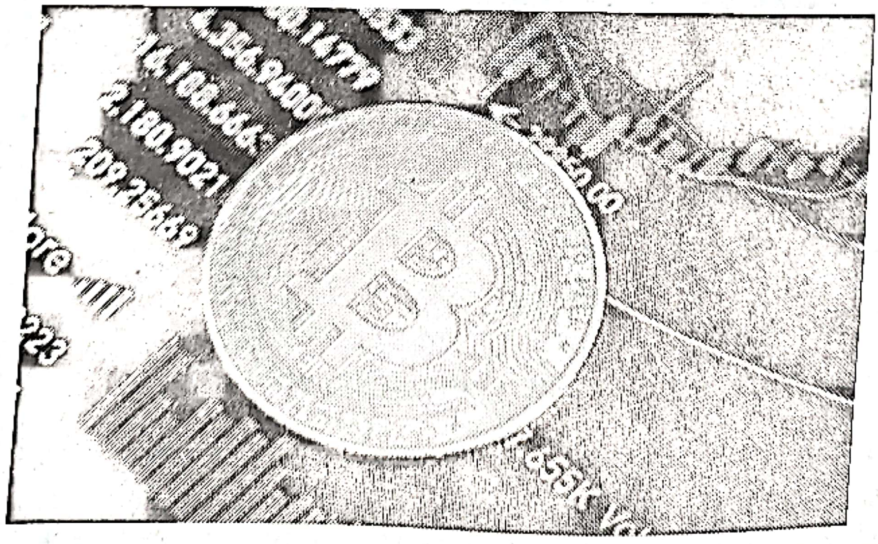


Fig. 4.3.1 : Bitcoin

- **Price :** $38,469
- **Market cap :** $727,550,374,174
- Bitcoin is the most popular digital cryptocurrency in the market and it is also the most expensive currency. It was allegedly created by Satoshi Nakamoto in 2009.

2. Ethereum (ETH)

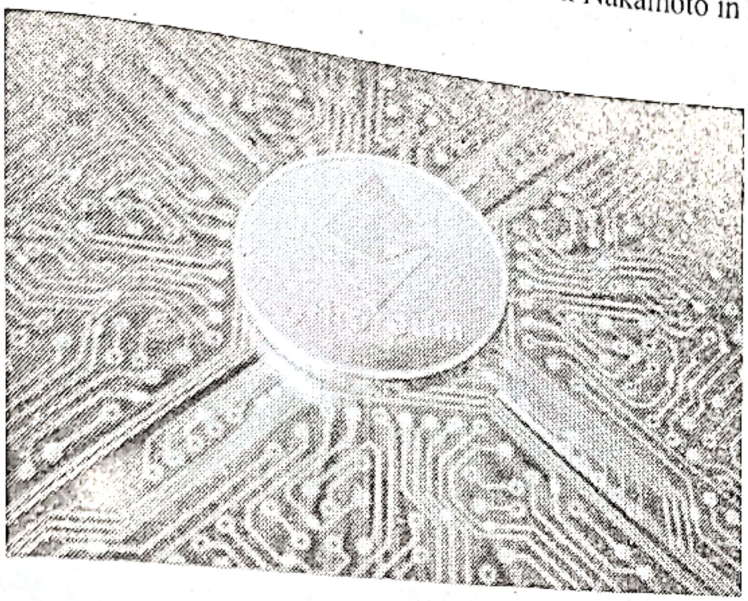


Fig. 4.3.2 : Ethereum

- **Price :** $2643
- **Market cap :** $316,743,239,800
- In the cryptocurrency market, it is the second most recognized currency. Ethereum's smart contract makes it a popular currency.

3. Tether (USDT)

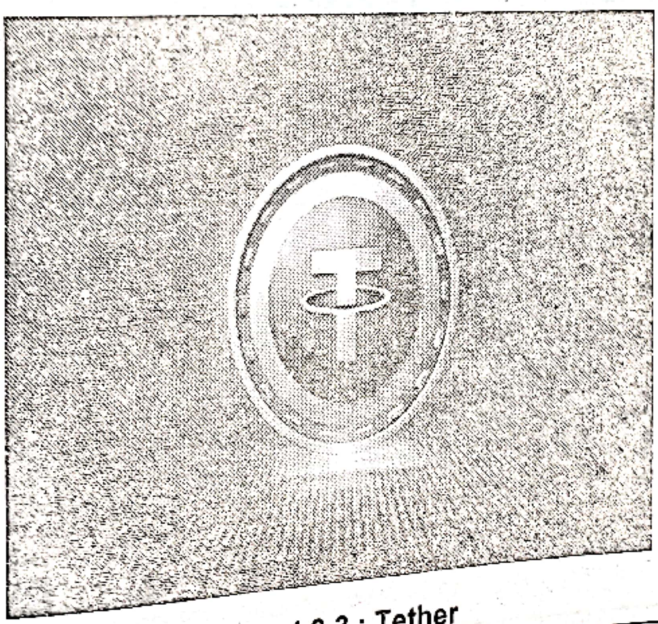


Fig. 4.3.3 : Tether

- **Price :** $1
- **Market cap :** $79,660,411,336
- Tether is known as a stable coin in the cryptocurrency market. Its value is tied to
  value of a specific asset. In this currency's case, it is the US Dollar.

4. **XRP (XRP)**

- **Price :** $0.726778
- **Market cap :** $34,840,877,217
- XRP was created in 2012 as Ripple. Later, its name was changed. It is believed not to
  a secure as it uses a trust-less mechanism to complete the transaction.

5. **Binance coin (BNB)**

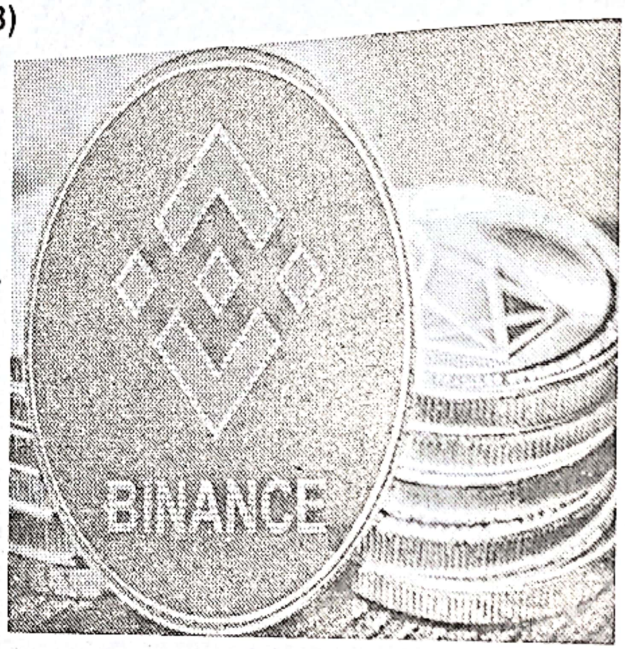


Fig. 4.3.4 : Binance coin

- **Price :** $367
- **Market cap :** $61,755,835,632
- Binance coin was primarily created as a token to pay for discounted trades. Binance coi
  was created by Binance, which is considered among the largest crypto exchanges in th
  world.

6. **Solana (SOL)**

- **Price :** $89
- **Market cap :** 28,382,901,996

- Solana was launched in March 2020. It is famous for completing the transaction in a very short time and powerful web-scale platform.

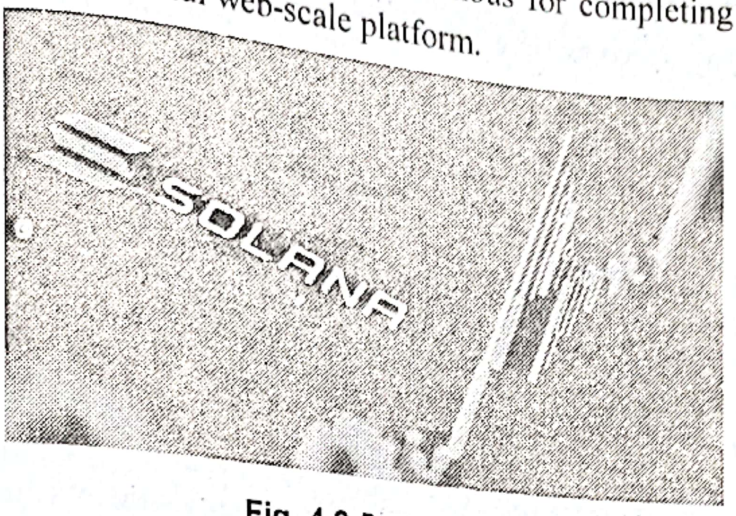


Fig. 4.3.5 : Solana

4.4 Cryptocurrency Usage

- Here are some of the rationales behind why some companies are currently using crypto :
  - Crypto may provide access to new demographic groups. Users often represent a more cutting - edge clientele that values transparency in their transactions. One recent study found that up to 40 % of customers who pay with crypto are new customers of the company and their purchase amounts are twice those of credit card users.
  - Introducing crypto now may help spur internal awareness in your company about this new technology. It also may help position the company in this important emerging space for a future that could include central bank digital currencies.
  - Crypto could enable access to new capital and liquidity pools through traditional investments that have been tokenized, as well as to new asset classes.
  - Crypto furnishes certain options that are simply not available with fiat currency. For example, programmable money can enable real-time and accurate revenue - sharing while enhancing transparency to facilitate back - office reconciliation.
  - More companies are finding that important clients and vendors want to engage by using crypto. Consequently, your business may need to be positioned to receive and disburse crypto to assure smooth exchanges with key stakeholders.
  - Crypto provides a new avenue for enhancing a host of more traditional Treasury activities, such as :
    - Enabling simple, real-time and secure money transfers
    - Helping strengthen control over the capital of the enterprise

- Managing the risks and opportunities of engaging in digital investments
- Crypto may serve as an effective alternative or balancing asset to cash, which may depreciate over time due to inflation. Crypto is an investable asset and some, such as bitcoin, have performed exceedingly well over the past five years. There are, of course, clear volatility risks that need to be thoughtfully considered.

4.4.1 Two Primary Paths for Using Crypto

- The following sections will provide some broad considerations around two different paths as your company embarks on its crypto journey.

Enabling payments : "Hands-off"

- Some companies use crypto just to facilitate payments. One avenue to facilitate payments is to simply convert in and out of crypto to fiat currency to receive or make payments without actually touching it. In other words, the company is taking a "hands-off" approach that keeps crypto off the books.
- Enabling crypto payments, such as bitcoin, without bringing it onto the company's balance sheet may be the easiest and fastest entry point into the use of digital assets. It may require the fewest adjustments across the spectrum of corporate functions and may serve immediate goals, such as reaching a new clientele and growing the volume of each sales transaction. Enterprises adopting this limited use of crypto typically rely on third-party vendors.
- The third-party vendor, acting as an agent for the company, accepts or makes payments in crypto through conversion into and out of fiat currency. This may be the simplest option to pursue. And, in all likelihood, it may cause relatively few disruptions to a company's internal functions, since the "hands-off" approach keeps crypto off the corporate balance sheet.

Enabling payments : "Hands-on"

- If a company is ready to go beyond simply enabling crypto payments and intends to broaden crypto adoption within operations and the treasury function in other words, to go the "hands-on" route it may potentially find a significant increase in benefits, as well as in the number of technical matters to address.

4.5 Crypto Wallet

- Cryptocurrency Wallet is nothing but a software codings used to store, send and receive cryptocurrency. Crypto wallet is a place that stores private and public keys safely and allows you to interact with various blockchain network which enables users to send and receive virtual currency.

Types of Crypto Wallet

- Here are the different types of crypto wallet available.
  1. Software wallet 2. Hardware wallet 3. Paper wallet
- **Software wallet can again be divided into**
  a) Desktop wallet b) Mobile wallet c) Online wallet

a) **Desktop wallet :** Desktop wallet is one which is downloaded and installed on desktop or laptop. It can be only accessed in the system in which it is downloaded. Desktop Wallet also offers high end security until the system is hacked .When the system gets a virus there is the possibility that you may lose your crypto assets.

b) **Mobile wallet :** Mobile wallet is an app that runs on mobile phone , which is more useful that they can be used anytime anywhere. Mobile wallets are usually much smaller and simple as only limited space is available on the mobile.

c) **Online wallet :** Online wallet runs on cloud storage and is accessible from any device conveniently. Online wallet stores private and public key online which is controlled by third party. This makes it more harmful and easily available for hackings and theft.

2. **Hardware wallet**

- Hardware Wallet is more different from Software wallet in the way that they store a users.
- Private and public key on the hardware devices like USB. Even hardware wallet do transaction.
- Online, they store cryptos offline which is highly secured.

3. **Paper wallet**

- Paper wallet is one which provides high level of security to store cryptocurrency. This refer to a physical copy or printout of the public and private keys. Paper wallet can be piece of software that is used to securely generate a pair of keys which are then printed.

4.5.1 Metamask - Introduction

- Creating cryptocurrency wallet chrome extension like MetaMask has become easier, all you will need is a script to develop a wallet of such type. MetaMask wallet and the way to develop a crypto wallet like MetaMask. Let's just get to know the criterion of a good cryptocurrency Wallet! We all know that the cryptocurrency wallet is an app (for mobile) or an extension (for desktop) that helps to hold / send/ receive cryptocurrency easily whenever and wherever needed.

Metamask

- The existing browsers on our systems are centralized. In order to create a de-centralized system we add a plug-in called "Metamask".
- We need it because we need "ether" (currency for blockchain).
- Installed Metamask (gives a virtual Ethereum wallet).
- Created a simple smart contract through Metamask (using fake ether i.e. currency of blockchain).
- In-depth study of state-of-art on smart contract implementation.

Remix IDE

- Platform to create and deploy smart contract, supports solidity.

Solidity :

- A language to create smart contracts, similar to Javascript.
- A good Crypto wallet needs to specifically fit into this criterion :
  - Wallet that supports you to control your private keys.
  - Easy to use User Interface.
  - Good and Active Development community
  - Well developed Backup and restore features.
  - Compatible with different OS.

4.5.1.1 Insights of Metamask

- MetaMask wallet is an Ethereum wallet that allows you to connect with Etheruem Blockchain. It also acts as a DApp browser for blockchain based applications .The Metamask wallet is used as browser extensions in various browsers. Being one of the best Ethereum wallet for desktop users, MetaMask allows one to manage, transfer and receive ethers along with the facilitation of usage of thousands of ERC20 tokens present in the ethereum blockchain. It has also recorded about 1,000,000 downloads overtime.

4.5.1.2 MetaMask Feature

- Intutive and Easy to use UI- With the help of user experience engineer.
- Multilingual - Present with 18 national/ international language HD (Hierarchical deterministic)Wallet- helps to backup using thesingle seed produced during the setup.
- Custom fee - for transactions within the ethereum system. ERC20 tokens- can easily add ERC20 tokens in the MetaMask wallet.
- Integrated with Crypto exchanges - allows users to buy ether from Coinbase and Shapshift network options.

4.5.1.3 Supported Platforms

- Since Metmask is a web wallet that is supported in various browsers via extensions, one can install the MetaMask add-on Chorme, Firefox, Opera and also in Brave browser.
- Once the MetaMask extension is been installed in the brower, one can easily transfer ETH on MetaMask. Lets see the whole process. To receive ETH, select the three dots that's present in the top right corner and further select Show QR Code.
- After that, a QR code would show up along with an ETH address that will be below the QR code. The code can then be scanned or copied to send ETH to the wallet.
- Now in order to send ETH from MetaMask to other wallet, one can select the Send button present in the home screen of the wallet page.
- Then one needs to fill up the Recipient Address Amount that will be needed to transfer out.

4.5.2 Coinbase

- Coinbase is a secure online platform for buying, selling, transferring and storing digital currency. Our mission is to create an open financial system for the world and to be the leading global brand for helping people convert digital currency into and out of their local currency.
- Sending or receiving digital currency between online crypto balances, friends or merchants on Coinbase is free! Handlling security and backups is not worrisome. It is a "one stop shop" - It offer Primary balance service, an exchange and merchant tools within one simple interface. Coinbase is a platform on which many applications are being built using our API. Customer is responsible for miner's fees on external transactions.

4.5.3 Binance

- Binance is a cryptocurrency exchange which is the largest exchange in the world in terms of daily trading volume of cryptocurrencies. It was founded in 2017 and is registered in the Cayman Islands.
- Binance was founded by Changpeng Zhao, a developer who had previously created high frequency trading software. Binance was initially based in China but later moved its headquarters out of China following the Chinese government's increasing regulation of cryptocurrency.
- In 2021, Binance was put under investigation by both the United States Department of Justice and Internal Revenue Service on allegations of money laundering and tax offenses. The UK's Financial Conduct Authority ordered Binance to stop all regulated activity in the United Kingdom in June 2021.

4.5.4 Comparision of Metamask , Binance and Coinbase

Title	Binance Wallet	Coinbase Wallet	Metamask
Platform Supported	1. SAAS 2. iphone 3. ipad 4. Android	1. Windows 2. MAC 3. iphone 4. ipad 5. Android	1. SAAS 2. iphone 3. ipad 4. Android
Audience	Crypto currencies users looking for a binance wallet browser extension to send and receive transactions.	Traders in need of storing all kinds of digital assets with a crypto wallet.	Crypto wallet and gateway for blockchain apps for anyone.
Support	Online	Online	Online
Pricing	Free Version	Free Version	Free Version
Training	Documentation	Documentation	Documentation
Company Information	Build and Build(Binance) Founded 2019 www.bnbchain.world/en/binance-wallet	Coinbase Founded 2012 Unitedstates Coinbase.com	Consensys Founded 2014 United States Metamask.io

Table 4.5 : Comparision of Metamask, Binance and Coinbase

Review Questions

Q.1 What is cryptocurrency ? explain it in brief. **(Refer Section 4.2.1)** **(5 Marks)**

Q.2 Purpose and working of cryptocurrency . **(Refer Section 4.2.1.1)** **(6 Marks)**

Q.3 State and explain advantages of cryptocurrency **(Refer Section 4.4)** **(5 Marks)**

Q.4 State and explain disadvantages of cryptocurrency. **(Refer Section 4.2.1.2)** **(5 Marks)**

Q.5	What is Bitcoin ? Describe how it works. **(Refer Section 4.1.1)**	**(6 Marks)**
Q.6	What is significance of Bitcoin. **(Refer Section 4.1.2)**	**(5 Marks)**
Q.7	State and explain different types of cryptocurrencies ? **(Refer Section 4.3)**	**(6 Marks)**
Q.8	Write a short note on Metamask. **(Refer Section 4.5.1)**	**(5 Marks)**
Q.9	Explain cryptowallet in detail ? **(Refer Section 4.5)**	**(6 Marks)**
Q.10	Compare Binance, Coinbase and Metamask. **(Refer Section 4.5.4)**	**(6 Marks)**
Q.11	Write a short note on Binance. **(Refer Section 4.5.3)**	**(5 Marks)**
Q.12	Write a short note on Coinbase. **(Refer Section 4.5.2)**	**(5 Marks)**

❑❑❑

Unit V

5 Blockchain Ethereum Platform using Solidity

Syllabus

What is Ethereum, Types of Ethereum Networks, EVM (Ethereum Virtual Machine), Introduction to smart contracts, Purpose and types of Smart Contracts, Implementing and deploying smart contracts using Solidity, Swarm (Decentralized Storage Platform), Whisper (Decentralized Messaging Platform).

Contents

5.1 What is Ethereum ?

- Ethereum is an open-source operating system that deals with smart contract functionality. This is a distributed computing platform which is used to facilitate to the development of decentralized digital applications (DApps) and blockchain technology related application development. Ethereum affords a decentralized digital gadget referred to as an Ethereum virtual machine (EVM) that is capable of running scripts with the aid of a global community of public nodes.
- Ethereum is the major distributed software application. It supports you to build smart contracts and distributed applications without downtime or third-party interference. Ethereum permits the developer to create and broadcast next-generation distributed applications.
- The Ethereum platform permits developers to form powerful decentralized applications with in-built economic functions. Though providing high accessibility, auditability, transparency and neutrality, it also reduces or eliminates censorship and decreases assured counterparty risks.
- These centralized systems are good extensive models for software applications. This system directly controls the operation of the individual units and the flow of information from a single center. In this kind of system, individuals are depended on the central power to send and receive information.
- But, there are issues with the centralized system are :
  - Only one point of controller for system and any disaster
  - The thing is it can be corrupt easily
  - User restricted access
  - The storage tower conclusion

5.1.1 History of Ethereum

- Below diagram shows evolution of ethereum.

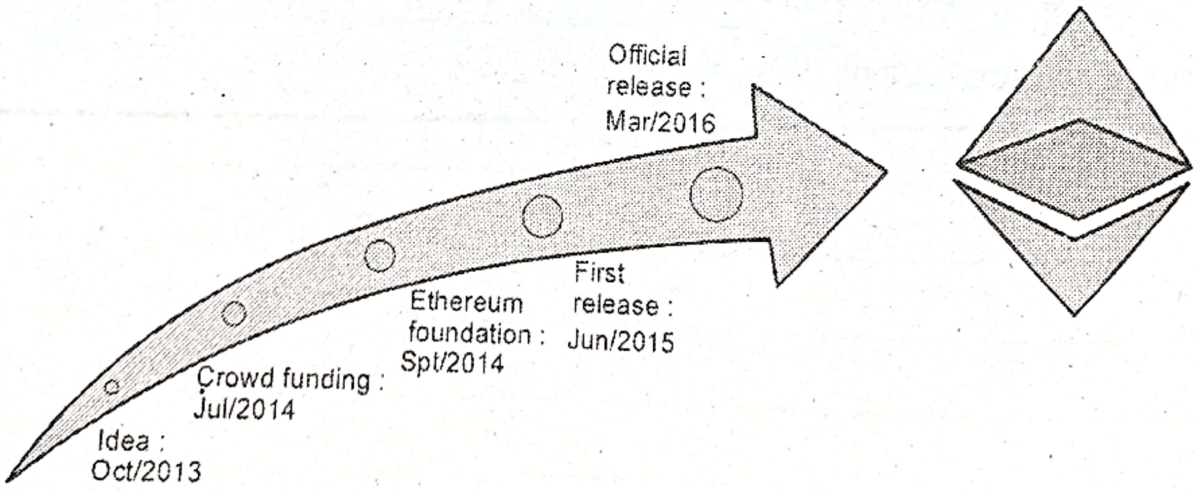


Fig. 5.1.1 : History of Ethereum

Russian-Canadian programmer co-founded Ethereum when he was 19 years old.

Year	Development
2013	Firstly on paper describe by Vitalik Buterin. He was also developer involved in Bit-coins.
2014	First Ethereum Software project Ethereum Switzerland Gmbh developed by Swiss Firm.
2015	The name of first version of Ethereum was Frontier.
March 2016	Second biggest version upgraded with A planned protocol Homestead.
May 2016	Ethereum becomes the most widespread media coverage once the DAO raised a record $150 million in crowd sale.
July 2016	The network split into two extensive types: Ethereum (ETH) and Ethereum Classic (ETC).
June 2017	Ethereum rallies overhead $400 demo a 5001 % increase from the time when Jan 1st, 2017.
May 2017	Ethereum will ultimately reach the success of Bitcoins
June 2018	The DAO remained hacked by an unknown group appealing $50 worth of ETH.

Table 5.1.1 : History of Ethereum

5.1.2 Ether

Let's discuss first definition of Ether, Ether is a chargeable token of the Ethereum blockchain system. Ethereum distant indexed by means of "ETH" on cryptocurrency interactions. Its just like lets you pay transaction costs first and computational facilities on the Ethereum communal. In the Ethereum established network on each and every occasion the contract is finished if it is not purchase, Ether is fully paid.

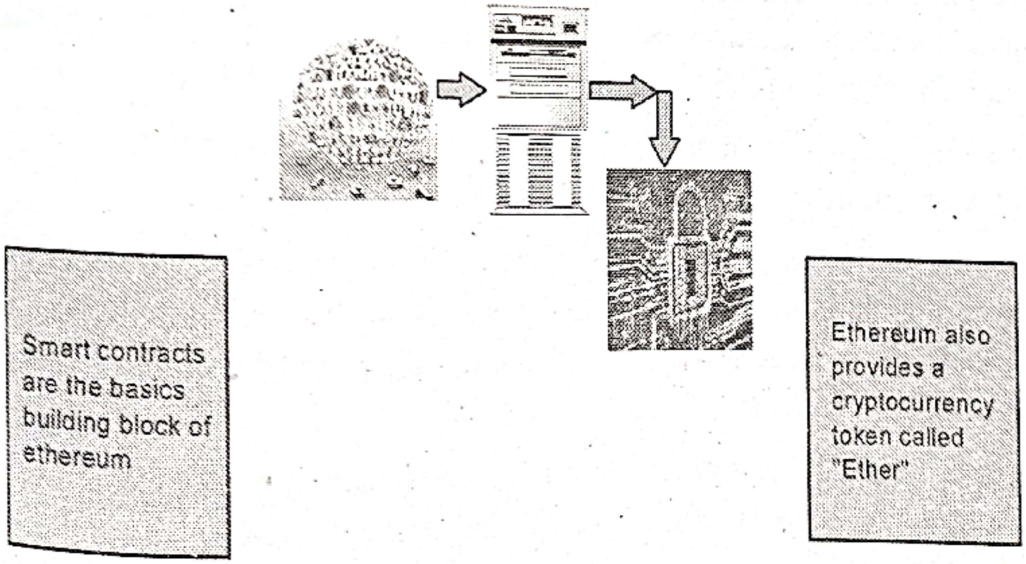


Fig. 5.1.2 : Ether

- **Gas :** In Ethereum network to perform an any transaction , the user firstly requires to make a payment for usage (to the miner) Ether via an in-between token called 'Gas.' Gas is unit which is used to allows you to measures the whatever computational work required for running a any smart contract or other transactions on ethereum network.
- Let us discuss how to calculate In Ethereum, the transactions fee is in Ether, which is given as

$$\text{Ether} = \text{Tx\_Fees} = \text{Gas\_Limit} * \text{Gas\_Price}$$

Where,

Gas_Limit = It is Refers to the usage of amount of gas that is for the computation

Gas_Price = The amount of Ether a user is required to pay

- Ethereum is a nothing but decentralized blockchain platform which is used to establish peer-to-peer network useful to securely execute and verifies application code and it is called as smart contracts. Smart contracts permit applicants to transact with every one other without a trusted principal authority. These transaction records are undisputable supportable and securely can distributed across the whole network, giving applicants full rights and visibility into the transaction data. These transactions are accessible or sent from and received by user-created Ethereum accounts only. The task is like, A sender must make transactions and devote Ether, as a cost of processing each transactions on the network is Ethereum's built-in crypto currency.

Ethereum accounts and its types :

- Accounts are performing one of the main construction chunks of the Ethereum blockchain network. For interaction between accounts among networks the state is created or updated as a result. State transition is represented by operations performed between and on the accounts in ethereum network. State transition is accomplished by means of what's named the Ethereum state transition function, which mechanisms as follows :

1. First thing related to transition is approve the transaction validity by inspection the syntax or grammar, sign validity and its nonce.
2. Then transaction fee is calculated and the sending address is determined by means of this signature. Additionally, Sender's account sense of balance is check and deducted for that reason and nonce is incremented. An error is reverted if the account balance is not enough for particular transaction.
3. Then responsibility to make available sufficient ether (gas price) to refuge the price of the transaction. This is charged per byte incrementally rendering to the proportions of the transaction.

4. In this phase, the genuine transferal of cost occurs. The movement is from the sender's account to receivers account. If account specified in the transaction is not exist still then the account is formed spontaneously. If the destination account type is a contract, then its contract code is executed. In some cases contract code will be executed fully if enough gas is available or else, it will run up to the point where it runs out of gas.
5. In cases due to insufficient account balance or gas the transaction failure, then all state-run changes are moved back with the exception of fee expenses, which is paid to the miners.
6. As a final point, the remainder (if any) of the fee is sent back to the sender as alteration and fee is paid to the miners consequently. By the side of this fact, the function returns the resultant state.

Types of accounts :

- There are two types of accounts associated with Ethereum :
  - Externally owned contacts
  - Contract accounts
- The first account type is externally owned accounts (EOAs) and the other type is contract accounts. EOAs are like to accounts that are controlled by a private key in bitcoin. Contract accounts are having private key along with the accounts that have code associated with them. An EOA has ether balance, is capable to send transactions and has no related code, whereas a Contract Account (CA) has ether balance, associated code and the ability to get caused and accomplish code in response to a transaction or a message that due to the Turing extensiveness property of the Ethereum blockchain network, the code within contract accounts can be of one of the level of complexity. The code is executed by EVM by each mining node on the Ethereum network. In addition, contract accounts are able to maintain their own permanent state and can call other contracts. It is envisaged that in the serenity release, the distinction between externally owned accounts and contract accounts may be eliminated.

Block :

- A Blocks are the main building blocks of a blockchain network. Ethereum blocks consist of various components, which are described as follows :
  - The block header
  - The transactions list
  - The list of headers of ommers or uncles

Prediction : Salary Base

- The transaction list is just a list of all transactions involved in the block. In adding, the list of headers of Uncles is also incorporated in the block. The utmost significant and difficult part is the block header.
- **Block header** - The Block headers are the most serious and comprehensive components of an Ethereum block. The header contains valued information, which is described in detail here.
- **Parent hash** - This is the Keccak 256-bit hash of the parent (earlier)block's header.
- **Ommers hash** - This is the Keccak 256-bit hash of the list of Ommers (Uncles) blocks included in the block.
- **Beneficiary** - Beneficiary field contains the 160-bit address of the receiver that will accept the mining payment once the block is fruitfully mined.
- **State root** - This field contains the Keccak 256-bit hash of the root node of the state tries. It is designed afterwards entirely transactions have been processed and finalized.
- **Transactions root** - This root is the Keccak 256-bit hash of the root node of the transaction tries. Transaction tries Denotes the list of transactions involved in the block.
- **Receipts root** - The receipts root is the keccak 256 bit hash of the root node of the transaction receipt tries. This tries is Collected of receipts of entirely transactions involved in the block.
- Transaction receipts are produced afterwards every single transaction is managed and encompass useful post- transaction evidence.
- **Logs bloom** - The logs bloom worked as a bloom filter that is collected of the logger address and log topics from the log entry of each transaction receipt of the involved transaction list in the block.
- **Difficulty** - The status of current block difficulty level.
- **Number** - The count of total number of all earlier blocks; the origin block is block zero.
- **Gas limit :** The field contains the value that represents the limit set on the gas intake per block.
- **Gas used :** The field contains the total gas spent by the transactions included in the block.
- **Timestamp :** Timestamp is the period Unix time of the time of block initialization.
- **Extra data :** Extra data field can be used to supply arbitrary data related to the block.

Mixhash : Mixhash field encompasses a 256-bit hash that as soon as combined with the nonce is used to prove that acceptable computational effort has been expended in order to create this block.

Nonce - Nonce is a 64-bit hash that is used to evidence, in grouping with the mixhash field, that acceptable computational effort has been expended in order to create this block.

5.1.3 Features of Ethereum

i) **Ether :** Ether is Ethereum's cryptocurrency.

ii) **Smart contracts :** Ethereum permits the improvement and distribution of these types of contracts.

iii) **Ethereum virtual machine :** Ethereum provides the fundamental technology the structural design and the software that comprehends smart contracts and permits you to interact with it.

iv) **Decentralized applications (Dapps) :** A decentralized application is called a Dapp (also meant DAPP, App or DApp) for short. Ethereum permits you to generate associated applications, called decentralized applications.

v) **Decentralized Autonomous Organizations (DAOs) :** Ethereum permits you to create these for independent administrative.

5.1.4 Real - World Applications of Ethereum

i) **Voting systems -** Voting systems are implementing Ethereum. The outcomes of polls are widely presented, confirming a transparent and fair self-governing procedure by removing passed by vote misuses.

ii) **Banking systems -** Ethereum is accomplishment approved extensively in banking application since with Ethereum's decentralized system, provides strong security for data and unauthorized access to hacker is denied. It also permits payments on an Ethereum-based network, so banks are also using Ethereum as a network to make settlements and payments.

iii) **Shipping -** Deploying Ethereum in shipping benefits with the pursuing of cargo and prevents goods from being misdirected or imitated. Ethereum make available the derivation and tracking framework for any asset required in a typical supply chain.

iv) **Agreements -** With Ethereum smart contracts, arrangements can be preserved and implemented without some alteration. So in a business that has disjointed applicants, is subject to disagreements and needs digital contracts to be contemporaneous, Ethereum is a technology for developed smart contracts and for digitally recording the agreements and the transactions based on them.

5.2 Ethereum Network

- The Ethereum network permits users to build and run apps, smart contracts and other transactions. These features are not offered in Bitcoin.

5.2.1 Components of Ethereum Network

1) Component-1 : Nodes

- Two types of nodes in an Ethereum network. They are as follows.

  i) **Mining node :** This node in authority for writing all the transactions that have happened in the Ethereum network in the block.

  ii) **Ethereum virtual machine node :** This node in the Ethereum network in which Smart Contracts and inventor in which there are a set of rules established on which both the parties approve to cooperate with each one other. The contract will be spontaneously implemented when the already defined rules are implemented. By defaulting, this node employs a 30303 port number for the purpose of communication among themselves.

2) Component-2 : Ether

- Like a bitcoin used blockchain network, Ether is a type of crypto currency used in the Ethereum network. It is a like peer-to-peer currency. It pathways and promoters every transaction in the network.
- In the world ether is the second-largest used crypto currency. First crypto currency was Bitcoin. The advantages of ether tokens can't be swapped by other crypto currencies to reduce computing power for Ethereum transactions. Ether is not free as a commission for any execution that affects the state in Ethereum. It is used in the Ethereum algorithm as an encouragement for miners who associate blocks to the blockchain using a proof-of-work method.

3) Component-3 : Gas

- Gas is an inner currency of the Ethereum network. This is needful to run applications on the Ethereum network, considerable as we need gas to run a vehicle.
- To comprehensive every transaction on the Ethereum network, a customer must first make a payment then send out ether and after that the intermediate monetary value is known as gas.
- Gas is a unit of quantity on the Ethereum network for the calculating power used to execute a smart contract or a transaction.

- The cost of gas is very low compared to Ether. The execution and resource utilization costs are predetermined in Ethereum in terms of Gas units, called gwei.

4) Component-4 : Ethereum accounts

- Two types of Ethereum accounts. They are as follows.
  - i) **Externally owned account** - These types of accounts are useful to store transactions.
  - ii) **Contract account** - These types of accounts are useful to store the details of Smart Contracts.

5) Component-5 : Nonce

- Designed for on the outside maintained accounts, nonce means that the number of transactions through this account. For a contract account, nonce means the number of contracts generated through this account.

6) Component-6 : Storage root

- This is the main root node of a Merkle tree. In Hash all details of the account is stored. The main root of the Merkle tree is the authentication of all transactions.

7) Component-7 : Ethash

- The proposed PoW algorithm for Ethereum 1.0 is Ethash. It's the furthermost current version of Dagger-Hashimoto, though, it's no extensive proper to call it that since many of the algorithms' preliminary characteristics taken histrionically altered in the development.

5.2.2 Types of Ethereum Network

Two types of Ethereum network :

1) Mainnet :

- A mainnet is a self-governing blockchain running its own network with its own technology and protocol.
- Mainnet is a live blockchain anywhere its particular crypto currencies or tokens are in usage, as related to a testnet or developments running on top of other popular networks such as Ethereum.
- The Programmers are use testnet to troubleshoot and experimental any new features on a blockchain.
- The main dissimilarity between testnets and mainnets is that the previous is a blockchain project that is in progress, while the later encompasses a completely developed blockchain.

- A limited critical steps may take place in advance the mainnet stage.
- These can include facility to provide like a token deal, giving an invention the funding to yield and test features. As soon as this phase is effectively executed, the mainnet stage is frequently revolved on sale.
- This would signify that the blockchain is entirely up and running.
- More than a few blockchain startups chosen to use their individual tokens secured to the Ethereum network.
- These remained ERC-20 tokens which are anticipated to be used exclusively on Ethereum's platform.
- Upon accomplishment of the ICO, the mainnet will be released.
- This ordinarily will be denoted with a built-in token somewhat than the ERC-20.
- The subsequent stage in the procedure is acknowledged as mainnet swap.
- This encompasses a swap amongst the ERC-20 tokens in reoccurrence for the new coins on the blockchain.
- As soon as the mainnet swap is accomplished, the new coins will generally be demolished.
- After destroying coin it will ensure that only new coin will be used.

2) Testnets :

- Currently there are only three testnets in use and every one performs in the same way to the construction of blockchain. Originators may have an individual preference or preferred testnet and projects naturally develop on only one of them.
- **Ropsten :** This is a proof-of-work of blockchain that furthermost carefully remind you of Ethereum; you can effortlessly mine faux-Ether.
- **Kovan :** This is a proof-of-authority of blockchain, taking place by the Parity team. Ether are rather than to be mined they are to be requested.
- **Rinkeby :** This is a proof-of-authority blockchain, taking place by the Geth team. Ether are rather than to be mined they are to be requested.

Connecting to a testnet :

- Ethereum addresses and private keys that work on Ethereum, work on each testnet like extremely careful not to send Ether or tokens on the Ethereum main-net to a testnet address or you will lose your assets.

5.3 EVM (Ethereum Virtual Machine)

The EVM is the part of Ethereum that holder's smart contract arrangement and finishing. Guileless value transferal transactions from one EOA to another don't need to include it, basically talking, but the whole thing else will include a state update calculated by the EVM. On a high level, the EVM running on the Ethereum blockchain can be believed of as a global distributed computer holding millions of executable bits and pieces, each with its own perpetual data store.

The EVM is a quasi - Turing-complete state-run machine; "quasi" because all finishing processes are incomplete to a limited figure of computational steps by the amount of gas accessible for any given smart contract execution. By means of such, the halting problem is "solved". (here all program executions will halt) and the condition where execution might (by accident or maliciously) run forever, thus transporting the Ethereum platform to halt in its entirety, is avoided.

The EVM is similar a stack-based architecture, storage all in-memory values on a stack. EVM works with a word size of 256 bits and has several addressable data components :

- An immutable program code ROM, loaded with the bytecode of the smart contract to be executed
- A volatile memory, with every location explicitly initialized to zero.
- A permanent storage that is part of the Ethereum state, also zero-initialized

There is also a set of environment variables and data that is available during execution.

The Ethereum Virtual Machine (EVM) Architecture and Execution Context shows the EVM architecture and execution context. Fig. 5.3.1. (See Fig. 5.3.1 on next page)

5.3.1 Comparison with Existing Technology

The term "virtual machine" is frequently applied to the virtualization of an actual computer, naturally by a "hypervisor" such as VirtualBox or QEMU or of a whole operating system illustration, such as Linux's KVM. These must afford a software generalization, correspondingly, of physical hardware and of system calls and other kernel functionality.

The EVM functions in a much more restricted area: it is just a calculation machine and as such offers an idea of just calculation and storing, similar to the Java Virtual Machine (JVM) design, for example. On or after a high-level lookout, the JVM is planned to offer a runtime environment that is agnostic of the underlying host OS or hardware, enabling compatibility across a wide variety of systems. High-level programming languages such as

Java or Scala (which use the JVM) or C# (which uses .NET) are compiled into the bytecode instruction set of their respective virtual machine. In the same way, the EVM executes its own bytecode instruction set (described in the next section), which higher-level smart contract programming languages such as LLL, Serpent, Mutan or Solidity are compiled into.

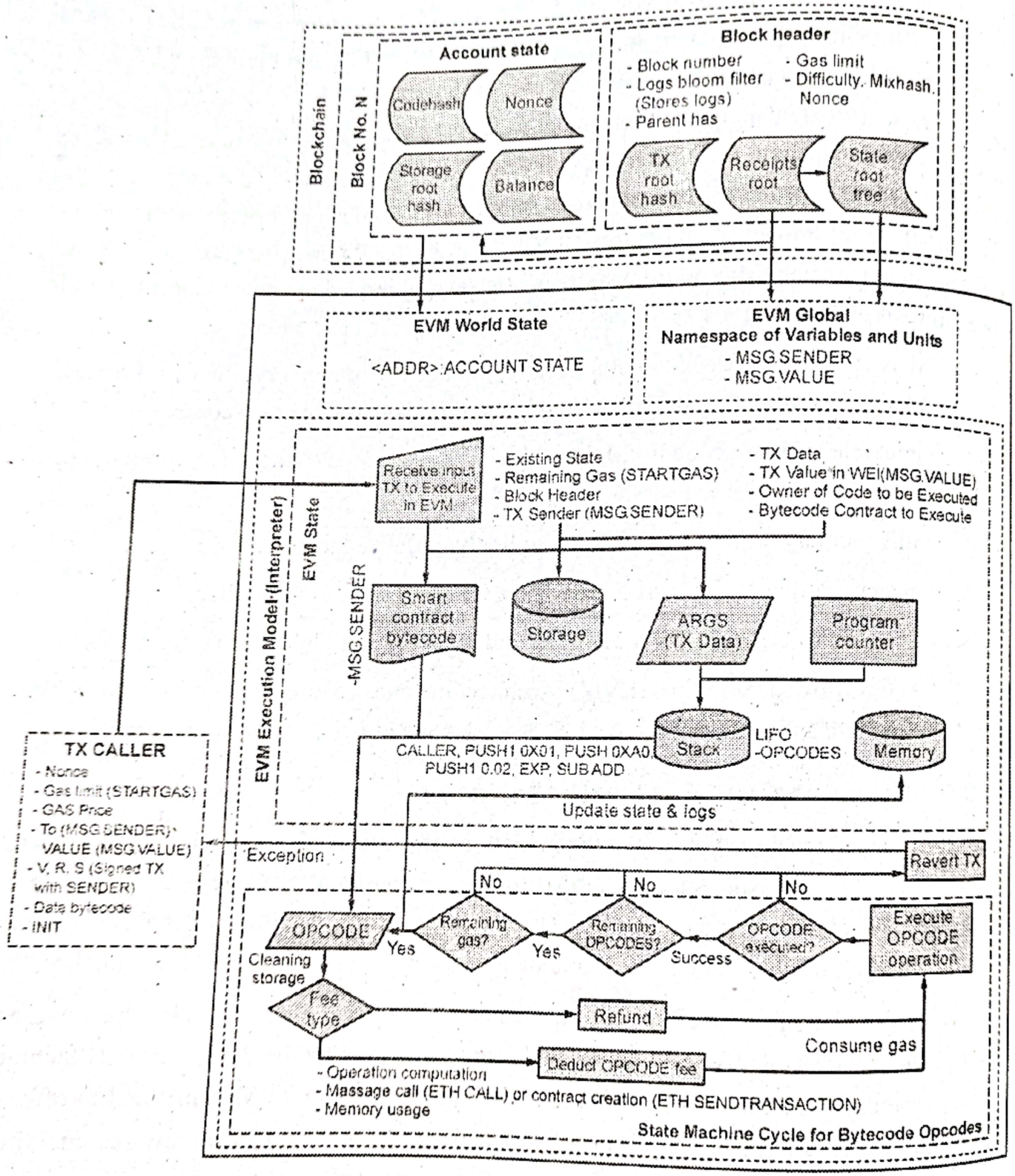


Fig. 5.3.1 : Architecture of EVM

- The EVM, therefore, has no scheduling capability, because execution ordering is organized externally to it Ethereum clients run through verified block transactions to determine which smart contracts need executing and in which order. In this sense, the Ethereum world computer is single-threaded, like JavaScript. Neither does the EVM have any "system interface" handling or "hardware support" there is no physical machine to interface with. The Ethereum world computer is completely virtual.

1) The EVM instruction set (Bytecode operations)

- The EVM instruction set deals most of the actions you might expect, including :
  - Mathematics like arithmetic and bitwise logic operations
  - Execution or implementation context inquiries
  - It's used Stack, memory and storage access
  - Control flow operations
  - Logging, calling and other operators
- In adding to the typical bytecode operations, the EVM also has right to use to account information (e.g., address and balance) and block information (e.g., block number and current gas price).

2) Ethereum state

- The job of the EVM is to keep informed the Ethereum state by calculating valid state changes as a result of smart contract code execution, as well-defined by the Ethereum protocol. This feature hints to the explanation of Ethereum as a *transaction-based state machine*, which replicates the point that exterior actors (i.e., account holders and miners) initiate state transitions by building, accommodating and ordering transactions. It is valuable at this fact to consider what creates the Ethereum state.
- At the topmost level, we have the Ethereum *world state*. The world state is a mapping of Ethereum addresses (160-bit values) to *accounts*. At the lower level, each Ethereum address represents an account comprising an ether *balance* (stored as the number of wei owned by the account), a *nonce* (representing the number of transactions successfully sent from this account if it is an EOA or the number of contracts created by it if it is a contract account), the account's *storage* (which is a permanent data store, only used by smart contracts) and the account's *program code* (again, only if the account is a smart contract account). An EOA will always have no code and an empty storage.

- When a transaction results in smart contract code execution, an EVM is instantiated with all the information required in relation to the current block being created and the specific transaction being processed. In particular, the EVM's program code ROM is loaded with the code of the contract account being called, the program counter is set to zero, the storage is loaded from the contract account's storage, the memory is set to all zeros and all the block and environment variables are set. A key variable is the gas supply for this execution, which is set to the amount of gas paid for by the sender at the start of the transaction. As code execution progresses, the gas supply is reduced according to the gas cost of the operations executed. If at any point the gas supply is reduced to zero we get an "Out of Gas" (OOG) exception; execution immediately halts and the transaction is abandoned. No changes to the Ethereum state are applied, except for the sender's nonce being incremented and their ether balance going down to pay the block's beneficiary for the resources used to execute the code to the halting point. At this point, you can think of the EVM running on a sandboxed copy of the Ethereum world state, with this sandboxed version being discarded completely if execution cannot complete for whatever reason. However, if execution does complete successfully, then the real-world state is updated to match the sandboxed version, including any changes to the called contract's storage data, any new contracts created and any ether balance transfers that were initiated.
- Note that because a smart contract can itself effectively initiate transactions, code execution is a recursive process. A contract can call other contracts, with each call resulting in another EVM being instantiated around the new target of the call. Each instantiation has its sandbox world state initialized from the sandbox of the EVM at the level above. Each instantiation is also given a specified amount of gas for its gas supply (not exceeding the amount of gas remaining in the level above, of course) and so may itself halt with an exception due to being given too little gas to complete its execution. Again, in such cases, the sandbox state is discarded and execution returns to the EVM at the level above.

5.4 Introduction to Smart Contracts

- A smart contract blockchain is a platform on which digital transaction procedures are embedded. It codifies the contract terms and specifies a fixed set of instructions all over the place the contract. The blockchain network keeps a crystal clear, secure and unchallengeable transaction.

The word smart contract has be there cast-off over the ages to describe a wide change of diverse things. In the 1990s, cryptographer Nick Szabo coined the term and defined it as "a set of promises, specified in digital form, together with protocols surrounded by which the parties accomplish on the other possibilities." Meanwhile then, the conception of smart contracts has developed, exclusively after the overview of distributed blockchain platforms with the origination of Bitcoin in 2009. In the perspective of Ethereum, the term is in fact a bit of a contradiction, known that Ethereum smart contracts are neither smart nor permitted contracts, but the term has stuck. We use the term "smart contracts" to refer to immutable computer programs that run deterministically in the perspective of an Ethereum Virtual Machine as amount of the Ethereum network protocol i.e., on the distributed Ethereum world computer.

5.4.1 Definition Related to Smart Contract

1) **Computer programs :** Smart contracts are simply computer programs. The word "contract" has no permissible meaning in this context.

2) **Immutable :** Once set up, the code of a smart contract is unchangeable. Unlike by means of traditional software, the one way to change a smart contract is to deploy a new instance.

3) **Deterministic :** The result of the execution of a smart contract is the similar for everybody who runs it, particular the perspective of the transaction that introduced its execution and the state of the Ethereum blockchain at the instant of execution.

4) **EVM context :** Smart contracts function with a very restricted execution perspective. They can access their private state, the perspective of the transaction that called them and some facts about the latest blocks.

5) **Decentralized world computer :** The EVM runs as a local instance on every single Ethereum node, but then since all instances of the EVM function on the similar initial state and yield the same final state, the system as a complete operates as a single "world computer."

5.4.2 Functioning of Smart Contracts

- A traditional (physical) contract includes two or more parties, such as individuals, objects and governments. They settle to contract terms and conditions to implement transactions via a third party. This third party might be a lawyer, a government organization or any other unit. It is here to take care of the records and execution of the contract. Not only does this add to audit and enforcement costs, but it also increases the risk of loss due to fraud.

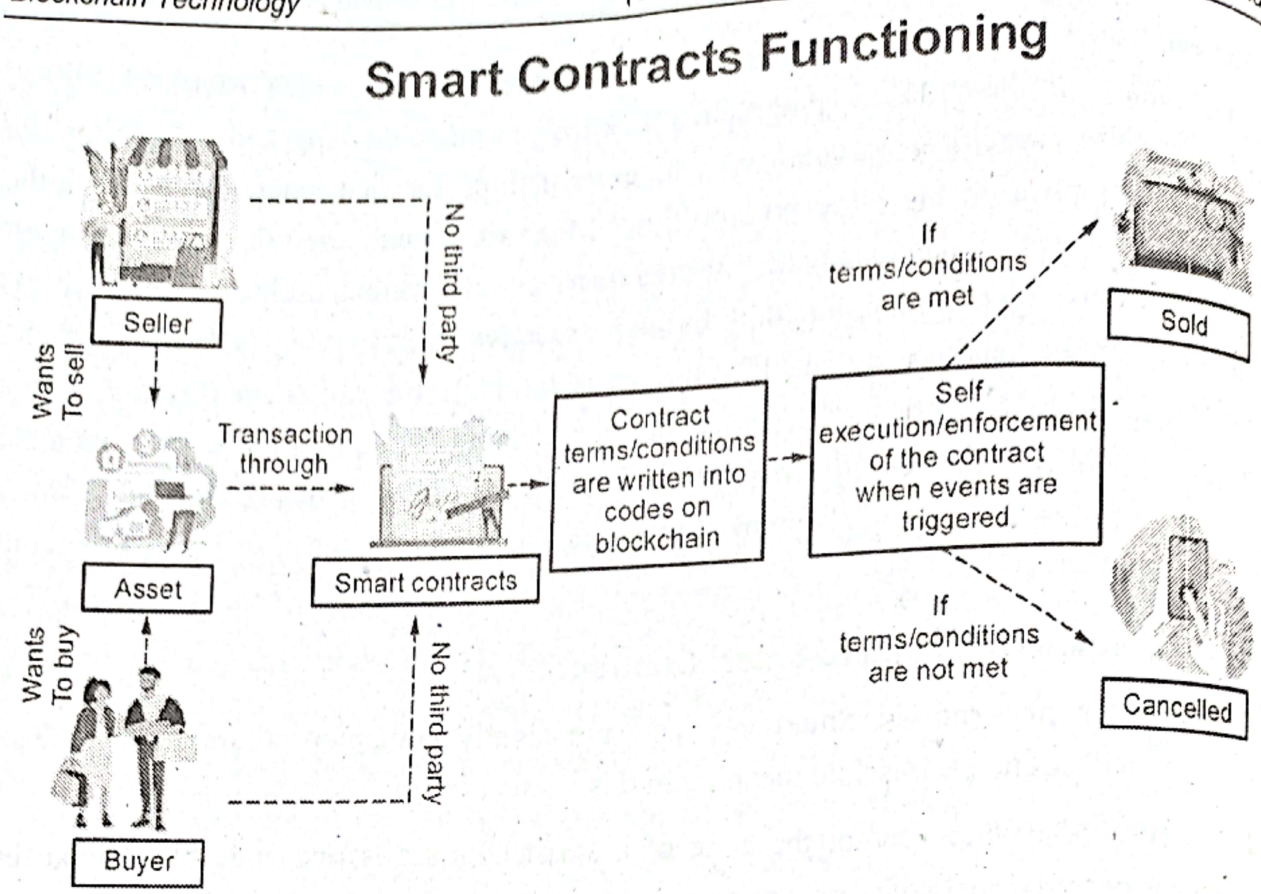


Fig. 5.4.1 : Functioning of smart contract

5.4.3 Let Us Explain the Process

- There are two parties (a buyer and a seller) involved in buying and selling a strength.
- These two parties go in a smart contract, an absolutely digital and self-executing contract, with its terms or clauses written in codes on a distributed blockchain network.
- These codes insist on the contract's terms, which both parties must approve to for the contract to be spontaneously imposed. The transaction happens when parties come across these terms and rules.
- The smart contracts raised area deals complete transparency and high-end security. As well, it restricts altering of the data and permits the two parties to pathway the transaction. The personalities of the parties involved, however, remain confidential.

5.4.4 Life Cycle of a Smart Contract

- Smart contracts are normally written in a top language, such as Solidity. Then in direction to run, they must be assembled to the low-level bytecode that runs in the EVM. Just the once compiled, they are set up on the Ethereum platform using a distinctive contract formation transaction, which is known as such by being sent to the special contract creation address, namely 0x0 (see [contract_reg]). Each contract is identified by an Ethereum address, which is derived from the contract creation transaction as a function of the

originating account and nonce. The Ethereum address of a contract can be used in a transaction as the recipient, sending funds to the contract or calling one of the contract's functions. Note that, unlike with EOAs, there are no keys associated with an account created for a new smart contract. As the contract creator, you don't get any special privileges at the protocol level (although you can explicitly code them into the smart contract). You certainly don't receive the private key for the contract account, which in fact does not exist we can say that smart contract accounts own themselves.

- Importantly, contracts only run if they are called by a transaction. All smart contracts in Ethereum are executed, ultimately, because of a transaction initiated from an EOA. A contract can call another contract that can call another contract and so on, but the first contract in such a chain of execution will always have been called by a transaction from an EOA. Contracts under no circumstances run "on their own" or "in the background." Contracts efficiently lie inactive up until a transaction triggers execution, either one directly or indirectly as part of a chain of contract calls. It is also worth noting that smart contracts are not implemented "in parallel" in some logic the Ethereum world computer can be measured to be a single-threaded machine.
- Transactions are atomic, regardless of how many contracts they call or what those contracts do when called. Transactions execute in their entirety, with any changes in the global state (contracts, accounts, etc.) recorded only if all execution terminates successfully. Successful termination means that the program executed without an error and reached the end of execution. If execution fails due to an error, all of its effects (changes in state) are "rolled back" as if the transaction never ran. A failed transaction is still recorded as having been attempted and the ether spent on gas for the execution is deducted from the originating account, but it otherwise has no other effects on contract or account state.
- As mentioned previously, it is important to remember that a contract's code cannot be changed. However, a contract can be "deleted," eliminating the code and its internal state (storage) from its address, separation a blank account. A few transactions sent to that account address afterwards the contract has remained deleted do not outcome in any code execution, because there is no longer any code there to execute. To delete a contract, you execute an EVM opcode called SELFDESTRUCT (previously called SUICIDE). That operation costs "negative gas," a gas reimburse, thereby the release of network client resources from the deletion of stored state. Removing a contract in this technique does not remove the transaction history (past) of the contract, since the blockchain itself is unchallengeable. It is similarly significant to note that the SELFDESTRUCT ability will only be available if the contract author programmed the smart contract to have that functionality. If the contract's code does not have a SELFDESTRUCT opcode or it is unreachable, the smart contract cannot be deleted.

5.4.5 Introduction to Ethereum High-Level Languages

- The EVM is a virtual machine that runs a distinct form of code called EVM bytecode, analogous to your computer's CPU, which runs machine code such as x86_64. In this section we will look at how smart contracts are written to run on the EVM.
- Even though it is potential to program smart contracts straight in bytecode, EVM bytecode is somewhat unmanageable and very challenging for programmers to read and realize. As a substitute, most Ethereum developers use a top language to write programs and a compiler to convert them into bytecode.
- Even though any high-level language could be altered to write smart contracts, adapting an uninformed language to be compliable to EVM bytecode is moderately a unwieldy exercise and would in general lead to some amount of misunderstanding. Smart contracts operate in a highly controlled and minimalistic execution environment (the EVM). In adding, a special set of EVM-specific system variables and functions needs to be available. By means of such, it is informal to build a smart contract language from scrape than it is to variety a general-purpose language appropriate for writing smart contracts. By means of a result, a number of special-purpose languages have appeared for programming smart contracts. Ethereum has a number of such languages, composed with the compilers required to yield EVM-executable bytecode.
- In all-purpose, programming languages can be classified into two widespread programming paradigms: declarative and overbearing, also known as efficient and practical, respectively. In declarative programming, we write functions that express the logic of a program, but then not its flow. Declarative programming is used to create programs where there are no side effects, meaning that there are no changes to state outer of a function. Declarative programming languages include Haskell and SQL. Imperative programming, by contrast, is where a programmer writes a set of procedures that combine the logic and flow of a program. Imperative programming languages include C++ and Java. Some languages are "hybrid," sense that they inspire declarative programming but can also be used to express an imperative programming paradigm. Such hybrids include Lisp, JavaScript and Python. In general, any imperative language can be used to write in a declarative model, but it frequently results in unsophisticated code. By means of evaluation, pure declarative languages cannot be used to write in an imperative paradigm. In purely declarative languages, there are no "variables."

- Even though imperative programming is additional frequently used by programmers, it can be very tough to write programs that execute accurately as predictable. The capability of any portion of the program to modification the state of any other makes it problematic to reason about a program's execution and make known to many opportunities for bugs. Declarative programming, by comparison, makes it easier to understand how a program will behave: since it has no side effects, any part of a program can be understood in isolation.
- In smart contracts, bugs accurately charge money. As a result, it is critically important to write smart contracts without unintentional effects. To prepare that, you must be able to evidently reason about the predictable actions of the program. So, declarative languages play a much superior role in smart contracts than they do in general-purpose software. Still, as you will see, the most widely used language for smart contracts (Solidity) is imperative. Programmers, like most humans, resist change!

5.4.6 Currently Supported High-Level Programming Languages for Smart Contracts Include (Ordered by Approximate Age)

1) **LLL :** A functional (declarative) programming language, with Lisp-like syntax. This one was the first high-level language for Ethereum smart contracts but is infrequently used today.

2) **Serpent :** A procedural (imperative) programming language with a syntax similar to Python. Can also be used to write functional (declarative) code, though it is not completely permitted of side effects.

3) **Solidity :** A procedural (imperative) programming language with a syntax similar to JavaScript, C++ or Java. The best popular and regularly used language for Ethereum smart contracts.

4) **Vyper :** A extra newly developed language, similar to Serpent and again with Python-like syntax. Intended to get closer to a pure-functional Python-like language than Serpent, but not to replace Serpent.

5) **Bamboo :** A newly developed language, predisposed by Erlang, with obvious state transitions and short of iterative flows (loops). Intended to decrease side effects and increase auditability. Actual new and up till now to be extensively approved.

However, of all of these Solidity is by far the most popular, to the point of being the de facto high-level language of Ethereum and even other EVM-like blockchains.

5.5 Purpose and Types of Smart Contracts

- On blockchain, the goal of a smart contract is to simplify business and trade between both anonymous and identified parties, sometimes without the need for a middleman. A smart contract scales down on formality and costs associated with traditional methods, without compromising on authenticity and credibility.
- **Some advantages of smart contracts are :**
  - **Security** - As the distributed record book is impregnable and immune to alterations.
  - **Disintermediation** - Enables parties to enter into agreements with reduced dependence on middlemen.
  - **Near real-time execution** - As it takes place almost simultaneously for all parties, across participating computers, once the necessary criteria are satisfied.
  - **Transparency** - Creates an environment of trust as the logic and information in the contract is visible to all participants in the blockchain network implementing smart contracts is not without its share of challenges, some of these if unaddressed can hinder its immediate adoption.
  - **Confidentiality** - However enterprises want transparency, they hesitate to put their contractual information, which may contain competitive strategies, on the blockchain. While a blockchain platform like Hyperledger is permission-driven and enables parties to engage in a private smart contract (visible only to people party to the contract), Ethereum, a blockchain platform, does not have an option for private smart contracts. Enterprises will, therefore, have to select their blockchain platform based on need.
  - **Accuracy** - Meanwhile a smart contract is a computer program, each term and condition of the contract wants to be coded. There is possibility of misinterpretation and omission by the coder, which may lead to loopholes in the contract. I believe the more we use smart contracts, the more we will encounter these loopholes and code against them.
  - **Unreliable inputs** - These could lead to false contracts or non-execution of contracts. In the case of a traditional contract, the parties can proceed to a judicial court for redressal. Unfortunately this is not a possibility with smart contracts where legal validity is still being debated upon.
  - **Bugs and errors in the code** - These could lead to disputes and procedural difficulties related to identifying errors and the parties responsible for those. They could also cause unforeseen repercussions. This is exactly what happened in June 2016, when a hacker exploited a vulnerability in the code of the Decentralized Autonomous Organization (DAO), which is a piece of smart contract built on Ethereum and made away with 50 million Ether, a bitcoin-like digital currency.

- **Rogue contracts** - Taking advantage of self-execution and anonymity of smart contracts, illegal activities could also be conducted by smugglers, terrorists, hackers and others.

- Smart contracts have the potential to introduce radical change in the way international business and trade are executed by speeding up transactions, reducing paperwork and bringing about cost-efficiency.

Types of smart contract :

- There are three types of self-executing contracts based on their applications :

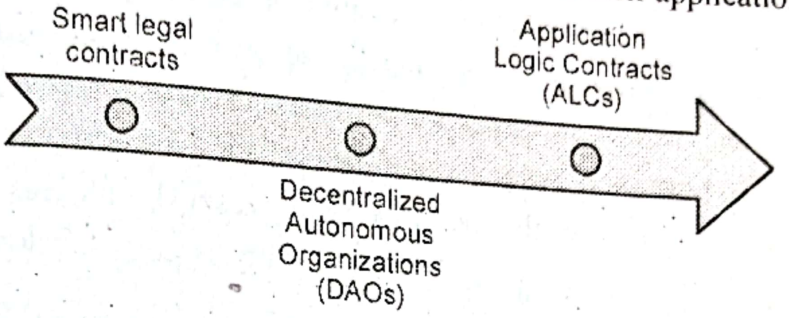


Fig. 5.5.1 : Types of smart contract

1. **Smart legal contracts :** These contracts are legally enforceable and require the parties to fulfill their votive responsibilities. Disappointment to do so may result in strict legal actions in contradiction of them.

2. **Decentralized autonomous organizations :** These are blockchain groups that are guaranteed to precise rules coded into blockchain contracts mutual with governance mechanisms. Henceforth, any action taken by the community members gets replaced by a self-enforcing code.

3. **Application logic contracts :** These contracts comprise an application-based code that leftovers in sync with other blockchain contracts. It permits communication across altered devices, such as the union of the Internet of Things with blockchain technology.

Building a smart contract with solidity

Solidity

- Solidity is a language like to JavaScript which permits you to develop contracts and compile to EVM bytecode. It is presently the prize language of Ethereum and the most popular.

i) **Solidity documentation -** Solidity is the flagship Ethereum high level language that is used to write contracts.

ii) Solidity online realtime compiler

Serpents

- Serpent is a language similar to Python which can be used to develop contracts and compile to EVM bytecode. It is intended to be outstandingly clean and simple, joining many of the efficiency benefits of a low-level language with ease-of-use in programming style and at the same time adding special domain-specific features for contract programming. Serpent is compiled using LLL.

  i) Serpent on the ethereum wiki

  ii) Serpent EVM compiler

- Solidity was created by Dr. Gavin Wood as a language explicitly for writing smart contracts with features to straight provision execution in the distributed situation of the Ethereum world computer. The resultant attributes are fairly general and so it has ended up being used for coding smart contracts on several other blockchain platforms. It was developed by Christian Reitiwessner and then also by Alex Beregszaszi, Liana Husikyan, Yoichi Hirai and several former Ethereum core contributors. Solidity is now developed and maintained as an free project on GitHub.
- The main "product" of the Solidity project is the Solidity compiler, solc, which transforms programs written in the Solidity language to EVM bytecode. The project also accomplishes the significant application binary interface (ABI) standard for Ethereum smart contracts, which we will explore in detail in this chapter. Each version of the Solidity compiler corresponds to and compiles a specific version of the Solidity language.
- To get taking place, we will download a binary executable of the Solidity compiler. At that point we will develop and compile a simple contract.

Selecting a version of solidity

- Solidity observers a versioning model called *semantic versioning*, which identifies version numbers structured as three numbers separated by dots : *MAJOR.MINOR.PATCH*.
- The "major" number is incremented for major and *backward-incompatible* modifications, the "minor" number is incremented as backward-compatible structures are added in in the middle of major releases and the "patch" number is incremented for backward-compatible bug fixes.
- At the time of writing, Solidity is at version 0.4.24. The procedures for major version 0, which is for preliminary development of a project, are different: whatever thing may modification at any time. In training, Solidity gives the "minor" number as if it be there the

major version and the "patch" number as if it be there the minor version. Therefore, in 0.4.24, 4 is considered to be the major version and 24 the minor version.

- The 0.5 major version release of Solidity is expected imminently.
- Your Solidity programs can contain a pragma directive that specifies the minimum and maximum versions of Solidity that it is compatible with and can be used to compile your contract.
- Since Solidity is speedily developing, it is frequently well to install the newest release.

Download and Install

- There are a number of techniques you can use to download and install Solidity, either one as a binary release or by compiling from source code. Here's how to install the most recent binary release of Solidity on an Ubuntu/Debian operating system, using the apt package manager :

First Step :- **$ sudo add-apt-repository ppa:ethereum/ethereum**

Second Step :- **$ sudo apt update**

Thirst Step :- **$ sudo apt install solc**

One time you have solc installed, checked the version by running following command:-

$ solc --version

solc, is nothing but the solidity compiler command line interface

Version : 0.4.24+commit.e67f0147.Linux.g++

- At hand number of additional ways to install Solidity, dependent on your operating system and necessities, including compiling from the source code directly.

Development environment

- In the direction of develop in Solidity, you can use any text editor and solc on the command line. Nevertheless, you might discover that some text editors designed for improvement, such as Emacs, Vim and Atom, offer added features such as syntax underlining and macros that make Solidity growth at ease.
- There are correspondingly web-based development environs, such as Remix IDE and EthFiddle.
- Use the tools that make you productive. In the end, Solidity programs are just plain text such as nano (Linux/Unix), TextEdit (macOS) or even NotePad (Windows). Just save your program source code by means of a *.sol* extension and it will be acknowledged by the Solidity compiler as a Solidity program.

- **Atom Ethereum interface** - This is Plugin for the Atom editor that structures syntax underlining, compiling and a runtime environment (needs backend node).
- **Atom solidity linter** - This is Plugin for the Atom editor that offers Solidity lining.
- **Vim solidity** - This is Plugin for the Vim editor as long as syntax highlighting.
- **Vim syntastic** - This is Plugin for the Vim editor provided that compile inspection.
- **Smart contract programming :** Solidity.

```
contract Sample {
    uint value;
    function set_Value(uint p_Value) {  value = p_Value;
    }
    function get_Value() returns (uint) {  return value;
    }
}
Var logIncrement = OtherExample.LogIncrement({sender:userAddress,uint value});

logIncrement.watch(function(err, result) {
    // do something with result
})
```

5.6 Implementing and Deploying Smart Contracts using Solidity

Development workflow :

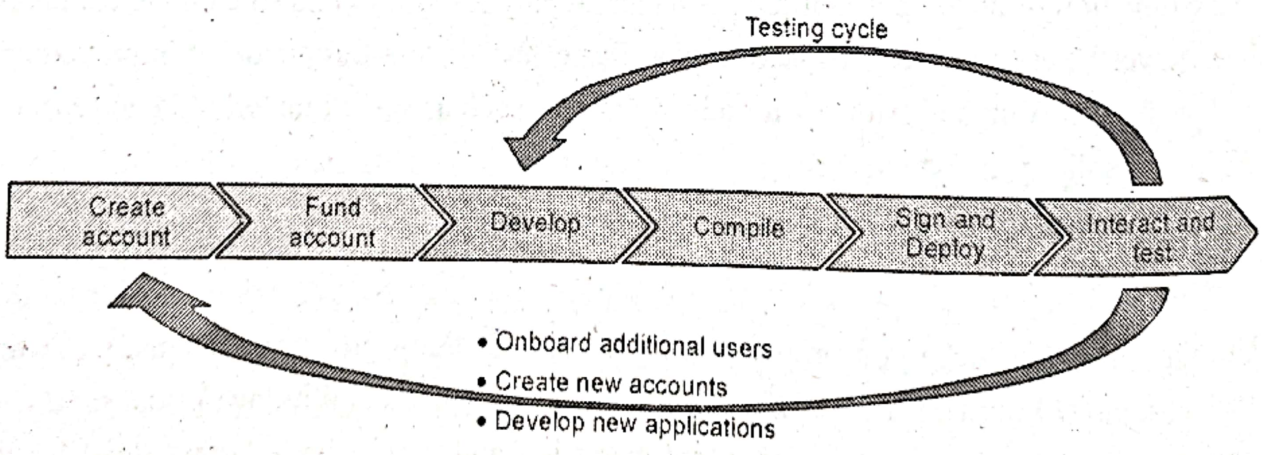
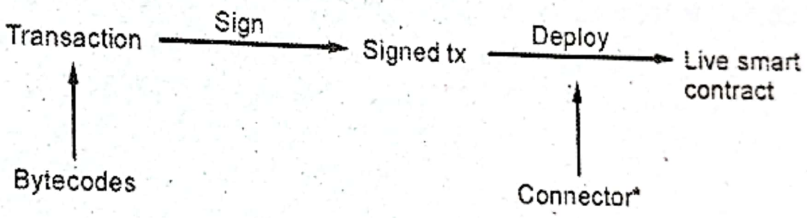


Fig. 5.6.1 : Development work flow

Steps to follow for development :

1) **Step 1 : Create account :**
   - Programmatically : Go, Python, C++, JavaScript, Haskell
   - Tools -MyEtherWallet.com, MetaMask, TestRPC, Many other websites
2) **Step 2 : Fund account :**
   - From friends
   - Faucet
   - Exchanges (for public blockchain)
3) **Step 3 : Develop**
   - Ethereum application components
   - Base application can be developed in any language
   - Smart contract : Developed in Solidity or one of the other contract compatible languages
   - Connector library : Facilitates communication between base application and smart contracts (Metamask)
4) **Step 5 : Sign and Develop :**



   - Library that facilitates communication and connection with Blockchain; Connects your code to a running node.
5) **Step 5 : Interact and Test :**
   - TestRPC / TestChain
   - Local development or Test Blockchain
   - **EthereumJS TestRPC :** https://github.com/ethereumjs/testrpc is suitable for development and testing.
   - It's a comprehensive blockchain-in-memory that runs only on your development machine.
   - Its procedures transactions rapidly as a substitute of waiting for the defaulting block time - so you can test that your code works rapidly - and it tells you instantly when your smart contracts run into errors.
   - It similarly marks a great client for automatic testing.

- Truffle distinguishes how to use its special features to rapidity up test runtime by almost 90 %.

Writing a simple solidity program

- Once we first built the Faucet contract, we used the Remix IDE to compile and set up the contract. In this section, we will reconsider, develop and elaborate Faucet.

Example 1. Faucet.sol: A Solidity contract implementing a faucet

Link : code/Solidity/Faucet.sol[]

Compiling with the solidity compiler (solc)

- At this moment, we will use the Solidity compiler on the command line to compile our contract in a straight line. The Solidity compiler solc deals a variation of choices, which you can understand by passing the --help argument.
- We use the --bin and --optimize arguments of solc to yield an optimized binary of our example contract :

```
$ solc --optimize --bin Faucet.sol
======= Faucet.sol:Faucet =======
Binary:
6060604052341561000f57600080fd5b60cf8061001d6000396000f300606060405260043610603e5
763ffffffff7c0100000000000000000000000000000000000000000000000000000006000350416
632e1a7d4d81146040575b005b3415604a57600080fd5b603e600435670163457 85d8a00008111156
0635760080fd5b73ffffffffffffffffffffffffffffffffffffffff31681156108fc0282604051
600060405180830381858888f193505050501560a057600080fd5b505600a165627a7a723058203
556d79355f2da19e773a9551e95f1ca7457f2b5fbbf4eacf7748ab59d2532130029
```

- The outcome that solc yields is a hex-serialized binary that can be give in to the Ethereum blockchain.

The Ethereum contract ABI

- In computer software, an *application binary interface* is an interface in the middle of two program modules; frequently, between the operating system and user programs. An ABI defines in what way data structures and functions are gain access to in *machine code*; this is not to be disordered with an API, which defines this right to use in high-level, often human-readable layouts as *source code*. The ABI is therefore the main way of encoding and decoding data into and out of machine code.
- In Ethereum, the ABI is used to encode contract calls for the EVM and to read data out of transactions. The persistence of an ABI is to define the purposes in the contract that can be raised and designate how each function will accept arguments and return its result.

- A contract's ABI is definite as a JSON array of function similes and events. A function description is a JSON object with field's type, name, inputs, outputs, constant and payable. An event description object has fields type, name, inputs and anonymous.
- We use the solc command-line Solidity compiler to yield the ABI for our *Faucet.sol* example contract :

```
$ solc --abi Faucet.sol
======= Faucet.sol:Faucet =======
Contract JSON ABI
[{"constant":false,"inputs":[{"name":"withdraw_amount","type":"uint256"}], \
"name":"withdraw","outputs":[],"payable":false,"stateMutability":"nonpayable", \
"type":"function"},{"payable":true,"stateMutability":"payable", \
"type":"fallback"}]
```

- As you can understand, the compiler yields a JSON array relating the two functions that are defined by *Faucet.sol*. This JSON can be used by any application that needs to entree the Faucet contract as soon as it is set up. By means of the ABI, an application such as a wallet or DApp browser can build transactions that call the functions in Faucet with the exact arguments and argument types. Consider example, a wallet would be familiar with that to call the function withdraw it would have to offer a uint256 argument named withdraw_amount. The wallet could prompt the user to offer that value, then form a transaction that encodes it and executes the withdraw function.
- Altogether that is desired for an application to interrelate with a contract is an ABI and the address wherever the contract has been deployed.

Selecting a Solidity Compiler and Language Version

- As we saw in the earlier code, our Faucet contract compiles successfully with Solidity version 0.4.21. But what if we had used a different version of the Solidity compiler ? The language is still in constant instability and effects may change in unpredicted ways. Our contract is impartially simple, but what if our program used a feature that was only added in Solidity version 0.4.19 and we tried to compile it with 0.4.18 ?
- To determination such problems, Solidity offers a *compiler directive* known as a *version pragma* that trains the compiler that the program supposes a particular compiler (and language) version. Let's look at an example :

```
pragma solidity ^0.4.19;
```

- The solidity compiler announces the version pragma and will yield an error if the compiler version is unable to get along with the version pragma. In this circumstance, our version pragma says that this program can be compiled by a Solidity compiler with a minimum

version of 0.4.19. The symbol ^ states, though, that we permit compilation with any *minor alteration* above 0.4.19; e.g., 0.4.20, on the other hand not 0.5.0 (which is a major revision, not a minor revision). Pragma directives are not compiled into EVM bytecode. They are simply used by the compiler to checked compatibility.

- Let's add a pragma directive to our Faucet contract.

> Example 2. Faucet2.sol: Adding the version pragma to Faucet
>
> Link : code/Solidity/Faucet2.sol[]

- Adding a version pragma is a greatest exercise, as it avoids difficulties with incompatible compiler and language versions.

Programming with solidity

- In this unit, we will aspect at some of the competences of the solidity language. As we stated in, our first contract example was very simple and also defective in various ways. We'll progressively recover it here, although discovering how to use solidity. As solidity is quite complex and rapidly evolving.

Data types

- First, let's aspect at some of the basic data types offered in solidity :

Boolean (bool)

- Boolean value like true or false, by means of logical operators ! (not), && (and), || (or), == (equal) and != (not equal).

Integer (int, uint)

- There are Signed (int) and unsigned (uint) integers, stated in additions of 8 bits from int8 to uint256. Deprived of a size suffix, 256-bit amounts are used, to match the word size of the EVM.

Fixed point (fixed, ufixed)

- Fixed-point numbers, stated with (u) fixedMxN wherever *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

- A 20-byte Ethereum address. The address object has countless cooperative member functions, the leading ones being balance which is returns the account balance and transfer which is transfers ether to the account.

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 unit's 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.

- Let's use one of the unit multipliers to improve the readability of our example contract. In the withdraw function we limit the maximum withdrawal, expressing the limit in wei, the base unit of ether :

```
require(withdraw_amount <= 10000000000000000);
```

- It is not very informal to read. We can increase our code by using the unit multiplier ether, to express the value in ether instead of wei :

```
require(withdraw_amount <= 0.1 ether);
```

Predefined global variables and functions

- When a contract is accomplished in the EVM, it has right to use to a small set of comprehensive objects. These consist of the block, msg and tx objects. In adding, Solidity disclosures a number of EVM opcodes as predefined functions. In this section we will study the variables and functions you can access starting in a smart contract in solidity.
- **Transaction/message call context -** This is the msg object usedful to the transaction call (EOA originated) or message call (contract originated) that thrown this contract implementation. It encompasses a number of valuable attributes :

msg.sender

- It signifies the address that introduced this contract call, not automatically the inventing EOA that sent the transaction. If our contract was so-called straight by an EOA transaction, at that point this is the address that employed the transaction, but or else it will be a contract address.

msg.value

- The value of ether sent with this call (in wei).

msg.gas

- The amount of gas leftward in the gas supply of this execution environs. This was denounced in Solidity v0.4.21 and substituted by the gasleft function.

msg.data

- The data payload of this call into our contract.

msg.sig

- The first four bytes of the data payload, which is the function selector.

Note : *Each and every time a contract calls alternative contract, the values of all the attributes of msg modification to reproduce the new caller's information. The only exclusion to this is the representative call function, which runs the code of substitute contract/library inside the unique msg context.*

Transaction context

- The tx object offers a means of retrieving transaction-related information :

tx.gasprice

- The gas price in the calling transaction.

tx.origin

- The address of the instigating EOA for this transaction. WARNING: unsafe!

Block context

- The block object encompasses information about the current block :

block.blockhash(__blockNumber__)

- The block hash of the stated block number, up to 256 blocks in the earlier. Denounced and substituted with the blockhash function in Solidity v0.4.22.

block.coinbase

- The address of the receiver of the current block's fees and block recompense.

block.difficulty

- The trouble (proof of work) of the current block.

block.gaslimit

- The all-out amount of gas that can be expended through all transactions included in the current block.

block.number

- The present block number (blockchain height).

block.timestamp

- The timestamp located in the present block by the miner (number of seconds since the UNIX epoch).

Address object

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

address.balance

- The balance of the address, in wei. For example, the present contract balance is address(this).balance.

address.transfer(__amount__)

- Transfers the amount (in wei) to this address, tossing an exemption on some error.

address.send(__amount__)

- Like to transfer, only as an alternative of throwing an exception, it returns false on error.

address.call(__payload__)

- This is Low-level CALL function can build a random message call with a data payload. Returns false on error.

address.callcode(__payload__)

- This is Low-level CALLCODE function, similar address(this).call(...) but through this contract's code substituted with that of address. Returns false on error. WARNING: advanced use only!

address.delegatecall()

- This is Low-level DELEGATECALL function, similar callcode(...) but by the full msg context seen by the present contract. Returns false on error.

Built-in functions

- There are many built in functions here we will discussed some of them.

addmod, mulmod

- Useful for modulo addition and multiplication. For example, addmod(x,y,k) calculates (x + y) % k.

keccak256, sha256, sha3, ripemd160

- This is Functions to calculate hashes with various standard hash algorithms.

ecrecover

- Its recovers the address used to sign a message from the signature.

selfdestrunct(__recipient_address__)

- It is used to deletes the present contract, sending any outstanding ether in the account to the receiver address.

this

- The address of the presently performing contract account.

Contract definition

- Solidity's primary data type is contract; our Faucet example just defines a contract object. Comparable to any object in an object-oriented language, the contract is a container that comprises data and methods.
- Solidity deals two other object types that are similar to a contract :

interface

- An interface meaning is organized exactly similar a contract, excluding none of the functions are definite, they are only stated. This kind of declaration is frequently called a *stub*; it expresses you the functions' arguments and yield types without any implementation. An interface requires the "shape" of a contract; when inherited, each of the functions declared by the interface must be defined by the child.

library

- A library contract is one and only that is intended to be arranged only once and used by other contracts, by means of the delegate call method.

Functions

- Contained by a contract, we describe functions that can be called by an EOA transaction or another contract. In our Faucet example, we have two functions: withdraw and the (unnamed) *fallback* function.
- The syntax we use to state a function in Solidity is as follows :

```
function Function_Name([parameters list]) { like public|private|internal|external}
[like pure|constant|view|payable] [modifiers] [returns (return types)]
```

- Let's look at each of these components :

FunctionName

- The name of the function, which is used to call the function in a transaction (from an EOA), from alternative contract or uneven from in the same contract.

- If the function without name then such case it is the *fallback* function, which is used to called when no other function with some named. The fallback function don't any arguments or return anything.

parameters

- Subsequent the name, we state the arguments that must be approved to the function, with their names and types. In our Faucet example we defined uint withdraw_amount as the only argument to the withdraw function.

Functions visibility mode :

public

- Public is the default; such functions can be called by other contracts or EOA transactions or from within the contract.

external

- External functions they cannot be called from within the contract except explicitly prefixed with the keyword this.

internal

- Internal functions are only manageable from within the contract they cannot be called by another contract or EOA transaction. They can be called by derived contracts (those that inherit this one).

private

- Private functions are similar to internal functions but not called by derived contracts.
- Keep attention that the terms *internal* and *private* are to some extent misrepresentative. Particular function or data limited a contract is constantly *visible* on the public blockchain, importance that anybody can see the code or data. The keywords labeled here only interrupt how and when a function can be *called*.
- The additional set of keywords (pure, constant, view, payable) disturb the behavior of the function :

constant or view

- A function noticeable as a *view* possibilities not to change any state. The term *constant* is an code-named for view that will be denounced in a future announcement. At this time, the compiler does not implement the view modifier, only creating a advice, on the other hand this is predictable to developed an imposed keyword in v0.5 of Solidity.

pure

- A pure function is one that neither reads nor writes any variables in storage. It can only operate on arguments and return data, without reference to any stored data.

payable

- This payable function is can be useful to receive incoming or receiving payments. If we not declared functions as payable will reject incoming payments. There are two exclusions, outstanding to design judgments in the EVM: coinbase payments and SELFDESTRUCT inheritance will be paid uniform if the fallback function is not declared as payable, but this kinds sense because code execution is not part of those payments anyhow.

Contract constructor and self-destruct

- Here is a superior function that is only used just the once. Once a contract is produced, it also used object oriented programming concept runs the *constructor function* if one exists, to initialize the state of the contract. The constructor is run in the similar transaction as the contract formation. The constructor function is noncompulsory; you'll notice our Faucet example doesn't have one.
- Constructors can be specified in two ways. Up and doing to and including in solidity v0.4.21, the constructor is a function whose name matches the name of the contract, as you can see here :

```
contract MyContract_1 {
    function MyContract_1() {
        // This is the constructor used to initialize state of contract.
    }
}
```

- The trouble with this format is that if the contract name is altered and the constructor function name is not altered, it is no longer a constructor then I will consider as function. This can reason some attractive horrible, unanticipated and difficult-to-find bugs. Visualize for example if the constructor is set the vendor of the contract for resolves of control. If the function is not essentially the constructor since of a specifying error, not only will the vendor be left unset at the period of contract formation, but the function may also be installed as a perpetual and "callable" share of the contract, like a usual function, permitting any third party to take over the contract and developed the "owner" afterwards contract formation.

- Towards address the possible difficulties with constructor functions presence founded on having an undistinguishable name as the contract, Solidity v0.4.22 announces a constructor keyword that works like a constructor function but does not have a name. Give name again the contract does not disturb the constructor at all. Likewise, it is informal to recognize which function is the constructor. It looks like this :

```
pragma ^0.4.22
contract MyContract_1
{
    constructor ()
    {
    // This is the constructor
    }
}
```

- Towards review, a contract's life cycle jumps with a formation transaction from an EOA or contract account. If here is a constructor, it is implemented as part of contract formation, to modify the state of the contract as it is existence created and is then rejected.
- The additional conclusion of the contract's life cycle is *contract destruction*. To destroyed contracts by a calling special EVM opcode that is SELFDESTRUCT. It is a called as SUICIDE, but that name was denounced outstanding to the negative suggestions of the word. In solidity, this opcode is unprotected as a high-level in-built function called self-destruct, which takes one argument as address to receive any ether balance outstanding in the contract account. It looks like this :

```
selfdestruct(address recipient);
```

- Note that you must explicitly add this command to your contract if you want it to be delectable this is the only way a contract can be deleted and it is not present by default. In this way, users of a contract who might rely on a contract being there forever can be certain that a contract can't be deleted if it doesn't contain a SELFDESTRUCT opcode.

Adding a constructor and self-destruct to our faucet example

- The Faucet example contract we introduced in does not have any constructor or selfdestruct functions. It is an eternal contract that cannot be deleted. Let's change that, by adding a constructor and selfdestruct function. We probably want selfdestruct to be callable *only* by the EOA that originally created the contract. By convention, this is usually stored in an

address variable called owner. Our constructor sets the owner variable and the selfdestruct function will first check that the owner called it directly.

- First, our constructor :

```
// Version of Solidity compiler this program was written for pragma solidity ^0.4.22;
// Our first contract is a faucet!
contract Faucet
{
    address owner;
    // Initialize Faucet contract: set owner
    constructor() {
        owner = msg.sender;
    }

[...]
```

- We've altered the pragma directive to require v0.4.22 as the lowest version for this example, by means of we are using the new constructor keyword announced in v0.4.22 of Solidity. In this contract currently takes an address type variable named owner. The name "owner" is not superior in any mode. We might call this address variable "potato" and still use it the similar mode. The name owner just makes its purpose clear.
- Next, our constructor, which runs as amount of the contract formation transaction, allocates the address from msg.sender to the owner variable. We used msg.sender in the withdraw function to recognize the originator of the withdrawal demand. In the constructor though the msg.sender is the EOA or contract address that started contract formation. We distinguish this is the case *since* this is a constructor function: it simply runs once, throughout contract creation.
- At the present we can add a function to terminate or destroy the contract. We necessity to make sure that only the owner can run this function, so we determination use a require statement to control access. This is the sample look of function :

```
//Contract destructor
function destroy() public
{
    require(msg.sender == owner);
    selfdestruct(owner);
}
```

- If anybody calls this destroy function from an address additional than owner, it will be unsuccessful. But if the similar address put away in owner by the constructor calls it, the contract resolve self-destruct and send any outstanding balance to the owner address.

Function modifiers

- Solidity deals with a superior kind of function called a *function modifier*. We can apply modifiers to functions by addition the modifier name in the function declaration. Modifiers are furthermost frequently used to make conditions that spread over to many functions in a contract. We take a right to use control statement previously, in our destroy function. Let's make a function modifier that states that condition :

```
modifier only_Owner {
  require(msg.sender == owner);
  _;
}
```

- This function modifier, named only_Owner, sets a condition on any function that it modifies, needful that the address kept as the owner of the contract is the similar as the address of the transaction's msg.sender. This is the simple plan for design a pattern for access control, permitting only the owner of a contract to accomplish any function that has the only_Owner modifier.
- This placeholder (like _;) is replaced by the code of the function that is being modified. Essentially, the modifier is "wrapped around" the modified function, employing its code in the location recognized by the underscore character.
- To put on a modifier, you improve its name to the function announcement. More than one modifier can be applied to a function; they are applied in the order they are stated, as a comma-separated list.
- Rewriting destroy function to use the only_Owner modifier :

```
function destroy() public only_Owner {
  selfdestruct(owner);
}
```

- The function modifier's name (only_Owner) is after the keyword public and tells us that the destroy function is modified by the only_Owner modifier. Essentially, you can read this as "Only the owner can destroy this contract."

- Function modifiers are an enormously useful tool because they allow us to write preconditions for functions and apply them constantly, making the code easier to read and, as a result, easier to inspection for security. They are most often used for access control, but they are quite versatile and can be used for a variety of other purposes.
- Inside a modifier, you can access all the values (variables and arguments) visible to the modified function. In this case, we can access the owner variable, which is declared within the contract. However, the inverse is not true: you cannot right to use any of the modifier's variables inside the modified function.

Contract inheritance

- Solidity's contract object supports *inheritance*, which is a mechanism for outspreading a base contract with additional functionality. To use inheritance, require a parent contract with the keyword is :

```
contract _Child is _Parent
{
  .....
}
```

- Through this construct, the child contract inherits all the procedures, functionality and variables of Parent. Solidity also provisions multiple inheritance, which can be stated by comma-separated contract terms after the keyword is :

```
contract _Child is _Parent1, _Parent2
{
  .....
}
```

- Contract inheritance permits us to write our contracts in such a manner as to attain modularity, extensibility and use again.
- We start by defining a base contract retained, which has an owner variable, set it in the contract's constructor :

```
contract By_owned {
    address owner_1;
    // Contract constructor: set owner_1
```

```
        constructor()
        {
            owner_1 = msg.sender;
        }
        // Access control modifier
        modifier only_Owner {
          require(msg.sender == owner_1);
          _;
        }
    }
```

- Next, we define a base contract mortal, which inherits owned :

```
    contract mortal is By_owned {
        // Contract destructor
        function destroy() public only_Owner
        {
            selfdestruct(owner_1);
        }
    }
```

- Let see, the temporal contract can use the only_Owner function modifier, defined in By_owned. It incidentally also uses the owner_1 address variable and the constructor distinct in owned_1. Inheritance marks each contract simpler and immersed on its full functionality, permitting us to achieve the particulars in a modular way.
- Now we can additional extend the By_owned contract, inheriting its competences in Faucet :

```
contract Faucet is mortal
{
  // Give out ether to anyone who asks
  function withdraw(uint withdraw_amount) public
  {
    // Limit withdrawal amount
    require(withdraw_amount <= 0.1 ether);
    // Send the amount to the address that requested it
    msg.sender.transfer(withdraw_amount);
  }
  // Accept any incoming amount
  function () public payable {}
}
```

- By inheriting mortal, which in try inherits By_owned, the Faucet contract now has the constructor and destroy functions and a defined owner. The functionality is the similar as when those functions were within Faucet, but now we can reprocess those functions in other contracts devoid of writing them yet again. Code reprocess and modularity mark our code cleanser, informal to read and at ease to review.

Error handling concept with assert function, require function, revert function :

- A contract call can dismiss and reoccurrence an error. Error handling mechanism in solidity is handled by the four functions :

1) **assert() function :** Assert is used when the outcome is predictable to be true, significance that we use assert to test inner conditions.

2) **require() function :** Require is used when testing inputs (condition such as function arguments or transaction fields), set our beliefs for those conditions. This function acts as a gate condition, checking execution of the rest of the function and creating an error if it is not satisfied.

3) **revert() function :** Revert functions halt the execution of the contract and revert any state variations. The revert function can also takings an error message as the one argument and it is recorded in the transaction log.

4) **throw() function :** The throw function is obsolete and will be removed in future versions of Solidity; you should use revert instead.

- In condition such as a contract terminates with a some error, due to that all the state changes (like changes to variables, balances and more) are reverted, in all way chain of contract calls if more than one contract was called. This make sure that transactions are *atomic*, significance they either one comprehensive effectively or have no influence on state and are reverted completely.

Events

- After completion of transaction successfully or not it generate transaction receipt as acknowledgement which contain log entries to release evidence about actions that happened throughout the execution of the particular transaction. *Events* are the Solidity high-level objects that are used to build these logs.
- Events are particularly beneficial for light clients and DApp facilities, useful on "watch" for particular events and report them to the user interface or kind an alteration in the state of the application to replicate an event in an essential contract.

- Serialized and recorded are the arguments taken by Event objects in the transaction logs in blockchain network. Before passing argument we can supply the keyword indexed to mark the value part of an indexed table (like hash table) useful for searched or filtered by an application.
- Let's see we will add two events, one which is apply for any withdrawals and another one to log any deposits. We will call these events Withdrawal and Deposit, respectively. First, we define the events in the Faucet contract :

```
contract Faucet is mortal
{
        event Withdrawal(address indexed to, uint amount);
        event Deposit(address indexed from, uint amount);

        [...]
}
```

- We've selected to mark the addresses indexed, to permit searching and filtering in every user interface constructed to right to use our Faucet.
- Next, we use the emit keyword to incorporate the event data in the transaction logs :

```
// Give out ether to anyone who asks
function withdraw(uint withdraw_amount) public
{
  [...]
  msg.sender.transfer(withdraw_amount);
  emit Withdrawal(msg.sender, withdraw_amount);
}
// Accept any incoming amount
function () public payable
{
  emit Deposit(msg.sender, msg.value);
}
```

- The resulting *Faucet.sol* contract appearances like Faucet8.sol: Revised Faucet contract, with events.

Catching events : Lets discuss truffle and its commands

```
$ truffle develop
truffle(develop)> compile
truffle(develop)> migrate
Using network 'develop'.
Running migration: 1_initial_migration.js
  Deploying Migrations...
  ... 0xb77ceae7c3f5afb7fbe3a6c5974d352aa844f53f955ee7d707ef6f3f8e6b4e61
  Migrations: 0x8cdaf0cd259887258bc13a92c0a6da92698644c0
Saving successful migration to network...
  ... 0xd7bc86d31bee32fa3988f1c1eabce403a1b5d570340a3a9cdba53a472ee8c956
Saving artifacts...
Running migration: 2_deploy_contracts.js
  Deploying Faucet...
  ... 0xfa850d754314c3fb83f43ca1fa6ee20bc9652d891c00a2f63fd43ab5bfb0d781
  Faucet: 0x345ca3e014aaf5dca488057592ee47305d9b3e10
Saving successful migration to network...
  ... 0xf36163615f41ef7ed8f4a8f192149a0bf633fe1a2398ce001bf44c43dc7bdda0
Saving artifacts...

truffle(develop)> Faucet.deployed().then(i => {FaucetDeployed = i})
truffle(develop)> FaucetDeployed.send(web3.toWei(1, "ether")).then(res => \
          { console.log(res.logs[0].event, res.logs[0].args) })
Deposit { from: '0x627306090abab3a6e1400e9345bc60c78a8bef57',
  amount: BigNumber { s: 1, e: 18, c: [ 10000 ] } }
truffle(develop)> FaucetDeployed.withdraw(web3.toWei(0.1, "ether")).then(res => \
          { console.log(res.logs[0].event, res.logs[0].args) })
Withdrawal { to: '0x627306090abab3a6e1400e9345bc60c78a8bef57',
  amount: BigNumber { s: 1, e: 17, c: [ 1000 ] } }
```

- logs[0] - First log entry holds an event name in logs[0].event and the arguments in logs[0].args. Using this log we can understand the released event name and the event arguments.
- Events are a fruitful mechanism, which is not only for intra-contract communication, but also for debugging during development.

How to call other contracts using send(), call(), callcode(), delegatecall() methods ?

- To perform some operation we need to call other contract which is useful but theoretically risky operation. So sometime we don't know about working of another contracts it can arise risky operation. So sometime we don't know about working of another contracts it can arise risk in your contract. Whenever we writing contract at time we should expect to deal with EOAs because there is not anything to halt randomly difficult and feasibly destructive contracts on or after calling into and presence called by your code.

How to create a new instance ?

- The not dangerous technique to call an additional contract is if you build that additional contract yourself. So we can used certain of its interfaces and behavior. By using new keyword we can simply instantiate it. In Solidity programming the new keyword used to create the contract on the blockchain and also return an object that you can use to reference it.
- Let's say you need to make and call a Faucet contract from in an additional contract called Token :

```
contract Token is mortal
{
        Faucet _faucet;
        constructor()
        {
                _faucet = new Faucet();
        }
}
```

- This mechanism for contract building guarantees that you distinguish the particular kind of the contract and its interface.

```
import "Faucet.sol";
contract Token is mortal
{
        Faucet _faucet;
        constructor()
        {
                _faucet = new Faucet();
        }
}
```

- You can optionally insist on the value of ether transfer on formation and pass arguments to the new contract's constructor :

```
import "Faucet.sol";
contract Token is mortal {
        Faucet _faucet;
        constructor() {
                _faucet = (new Faucet).value(0.5 ether)();
        }
}
```

- You can similarly then call the Faucet functions. In this below example, we call the destroy function of Faucet from in the destroy function of Token :

```
import "Faucet.sol";
contract Token is mortal
{
    Faucet _faucet;
    constructor()
    {
        _faucet = (new Faucet).value(0.5 ether)();
    }

    function destroy() ownerOnly {
        _faucet.destroy();
    }
}
```

Addressing an existing instance in contract :

- Additional way you can call a contract is by forming the address of a present instance of the contract. By this technique, you put on a recognized interface to a present instance. It is consequently critically significant that you recognize for sure that the instance you are addressing is in detail of the kind you undertake.

Raw call, delegatecall functions :

- Solidity offers various even additional "low-level" functions for calling other contracts. These parallel in a straight line to EVM opcodes of the similar name and permit us to build a contract-to-contract call physically. By means of way, they signify the furthermost flexible *and* the furthermost risky mechanisms for calling other contracts.
- Below example shows the call() function :

```
contract Token is mortal
{
    constructor(address _faucet)
    {
        _faucet.call("withdraw", 0.1 ether);
    }
}
```

- This type of call is called as a *blind* call into a function, actual abundant similar to constructing a rare transaction simply from in a contract's background. This can interpretation your contract to a number of security risks, furthermost highly *reentrancy*.

The function return false if there is a problem, so you can estimate the return value for error handling :

```
contract Token is mortal {
        constructor(address _faucet) {
                if !(_faucet.call("withdraw", 0.1 ether)) {
                        revert("Withdrawal from faucet failed");
                }
        }
}
```

- Additional alternative of call is delegatecall, which substituted the additional risky callcode.

Gas considerations in smart contract :

- Gas is an extremely significant attention in smart contract programming.
- Gas is a reserve making the all-out total of calculation that Ethereum will permit a transaction to put away.
- If the gas limit is overdone throughout calculation, the next sequence of events occurs :
  1) It can caught an "out of gas" exception.
  2) The state-run of the contract subsequent to execution is returned (reverted).
  3) Transaction fee is pay for the gas usage of ether and it is *not* reimbursed.
- Since gas is paid by the user who pledges the transaction, users are dispirited from calling functions that have in elevation gas cost. The role of programmer is to minimize the gas cost of a contract's functions.

Avoid dynamically sized arrays in smart contract :

- Some loop from side to side a dynamically sized array anywhere a function does actions on every component or searches for a specific element announces the risk of using too much gas.

Avoid calls to other contracts

- The risk is there in calling other contracts specially once the gas cost of their functions is not known, announces the risk of consecutively out of gas. We can achieve by avoiding the libraries that are not well tested and broadly used. The smaller amount inspection a library has acknowledged from other programmers, the better the risk of using it.

The function return false if there is a problem, so you can estimate the return value for error handling :

```
contract Token is mortal {
        constructor(address _faucet) {
                if !(_faucet.call("withdraw", 0.1 ether)) {
                        revert("Withdrawal from faucet failed");
                }
        }
}
```

- Additional alternative of call is delegatecall, which substituted the additional risky callcode.

Gas considerations in smart contract :

- Gas is an extremely significant attention in smart contract programming.
- Gas is a reserve making the all-out total of calculation that Ethereum will permit a transaction to put away.
- If the gas limit is overdone throughout calculation, the next sequence of events occurs :
  1) It can caught an "out of gas" exception.
  2) The state-run of the contract subsequent to execution is returned (reverted).
  3) Transaction fee is pay for the gas usage of ether and it is *not* reimbursed.
- Since gas is paid by the user who pledges the transaction, users are dispirited from calling functions that have in elevation gas cost. The role of programmer is to minimize the gas cost of a contract's functions.

Avoid dynamically sized arrays in smart contract :

- Some loop from side to side a dynamically sized array anywhere a function does actions on every component or searches for a specific element announces the risk of using too much gas.

Avoid calls to other contracts

- The risk is there in calling other contracts specially once the gas cost of their functions is not known, announces the risk of consecutively out of gas. We can achieve by avoiding the libraries that are not well tested and broadly used. The smaller amount inspection a library has acknowledged from other programmers, the better the risk of using it.

The function return false if there is a problem, so you can estimate the return value for error handling :

```
contract Token is mortal {
        constructor(address _faucet) {
                if !(_faucet.call("withdraw", 0.1 ether)) {
                        revert("Withdrawal from faucet failed");
                }
        }
}
```

- Additional alternative of call is delegatecall, which substituted the additional risky callcode.

Gas considerations in smart contract :

- Gas is an extremely significant attention in smart contract programming.
- Gas is a reserve making the all-out total of calculation that Ethereum will permit a transaction to put away.
- If the gas limit is overdone throughout calculation, the next sequence of events occurs :
  1) It can caught an "out of gas" exception.
  2) The state-run of the contract subsequent to execution is returned (reverted).
  3) Transaction fee is pay for the gas usage of ether and it is *not* reimbursed.
- Since gas is paid by the user who pledges the transaction, users are dispirited from calling functions that have in elevation gas cost. The role of programmer is to minimize the gas cost of a contract's functions.

Avoid dynamically sized arrays in smart contract :

- Some loop from side to side a dynamically sized array anywhere a function does actions on every component or searches for a specific element announces the risk of using too much gas.

Avoid calls to other contracts

- The risk is there in calling other contracts specially once the gas cost of their functions is not known, announces the risk of consecutively out of gas. We can achieve by avoiding the libraries that are not well tested and broadly used. The smaller amount inspection a library has acknowledged from other programmers, the better the risk of using it.

Estimating gas cost of contract :

Following procedure elaborate the estimate of the gas necessary to execute a certain method of a contract considering its arguments :

```
var contract = web3.eth.contract(abi).at(address);
var gasEstimate = contract.myAweSomeMethod.estimateGas(arg1, arg2,
{from: account});
```

Where gasEstimate used to calculate number of gas units needed for its execution. This one is an estimate for the reason that of the Turing completeness of the EVM and it is comparatively insignificant to make a function that will take immensely dissimilar amounts of gas to execute not the same calls. Uniform construction code can alteration execution pathways in understated ways, resultant in immensely different gas costs from one call to the next.

To get the gas price as of the network you can use :

```
var gasPrice_1 = web3.eth.getGasPrice();   // getGasPrice will return cost
var gasCostInEther = web3.fromWei((gasEstimate * gasPrice), 'ether');
```

Example to estimates gas function :

```
var FaucetContract = artifacts.require("./Faucet.sol");
FaucetContract.web3.eth.getGasPrice(function(error, result) {
  var gasPrice = Number(result);
  console.log("Gas Price is " + gasPrice + " wei"); // "10000000000000"

  // Get the contract instance
  FaucetContract.deployed().then(function(FaucetContractInstance) {

        // Use the keyword 'estimateGas' after the function name to get the gas
        // estimation for this particular function (aprove)
        FaucetContractInstance.send(web3.toWei(1, "ether"));
    return FaucetContractInstance.withdraw.estimateGas(web3.toWei(0.1, "ether"));

  }).then(function(result) {
    var gas = Number(result);

    console.log("gas estimation = " + gas + " units");
    console.log("gas cost estimation = " + (gas * gasPrice) + " wei");
    console.log("gas cost estimation = " +
        FaucetContract.web3.fromWei((gas * gasPrice), 'ether') + " ether");
  });
});
```

- Below demonstration shows the execution in truffle :

```
$ truffle develop
truffle(develop)> exec gas_estimates.js
Using network 'develop'.
Gas Price is 20000000000 wei
gas estimation = 31397 units
gas cost estimation = 627940000000000 wei
gas cost estimation = 0.00062794 ether
```

- It is mentioned that you estimate the gas cost of functions as portion of your growth workflow, to avoid some disclosures once arranging contracts to the mainnet.

5.7 Decentralized Applications (DApps)

- At the initial generations of Ethereum, the originators' idea was abundant larger than "smart contracts": not at all a smaller amount than reinventing the web and building a new world of DApps, appropriately called web3.
- Smart contracts are a way to distribute the supervisory logic and payment functions of requests.
- Web3 DApps are around distributing all other parts of an application: storage, messaging, naming, etc.

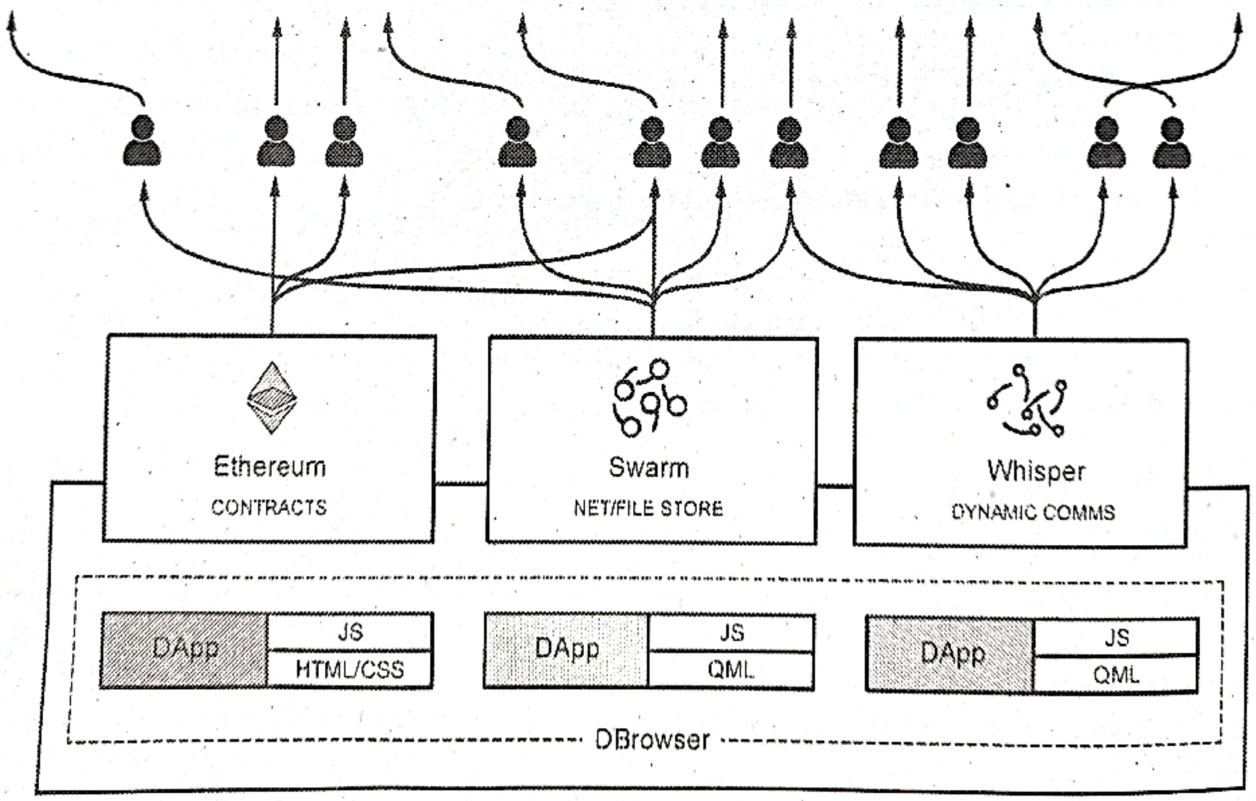


Fig. 5.7.1 : Decentralized applications

What is a Decentralized Application (DApp) ?

- This is an application which is frequently or completely distributed or decentralized.
- Think through altogether the unlikely features of an application that might be decentralized:
  i) Backend software or application logic
  ii) Frontend software
  iii) Data storage
  iv) Message communications
  v) Name or term resolution
- All of these can be to some extent centralized or to some extent distributed or decentralized.
- Let us discuss one example, developing front end as web application which is run on centralized server or mobile application that execute on mobile device. At another end backend and storage of data on our own or private server and known databases or you can use a smart contract and P2P storage.
- Following are the three advantages for creating DApp that distinctive centralized architecture cannot offer :

1) **Resiliency :** Since the business logic is measured by a smart contract, a DApp backend will be fully distributed and accomplished on a blockchain platform. Not like an application installed on a central server, a DApp will have no interruption and will remain to be accessible as extended as the platform is unmoving operating.

2) **Transparency :** The blockchain environment of a DApp permits everybody to review the code and be certain approximately its function. Some interface over and done with the DApp will be put away repeatedly in the blockchain.

3) **Censorship resistance :** User takes right of entry to an Ethereum node, the user will at all times be capable to cooperate with a DApp devoid of interference on or after any central controller. Not any service supplier or uniform the owner of the smart contract, be able to change the code when it is installed on the network.

5.7.1 Ethereum Swarm

- **Blockchains**, such as **Ethereum**, allow the implementation of distributed or decentralized applications **(DApps)**. The foremost knowledge of DApps contains of installing an application in the method of smart contracts on an unchallengeable blockchain, therefore eliminating trusted application servers and single points of failure. Ethereum Swarm is designed to enhance the emergence of DApps and by extension the Web 3.0 paradigm, by serving as a decentralized data storage solution.

- Though, it is unpredictably hard to make the Web 3.0 model workable in a purely decentralized way. There are two main reasons for this. Firstly, interacting with smart contracts is complicated and provides a very poor user experience. Intended for this purpose, furthermost DApps offer a web interface, comprising of an off-chain frontend that is held on a traditional web server and helped over the HTTP protocol. This introduces a trustworthy centralized module interested in the setup. Secondly, it is very exclusive to accumulation huge amounts of data on a blockchain, which is why DApps frequently have need of a way of putting away some of their data off-chain. Once more, by means of a database management system or a traditional file system spirits in contradiction of the distributed model.

Decentralized storage :

- Decentralized storage is dependence on central mechanisms can be found in decentralized storage solutions. The idea is simple: a peer-to-peer (P2P) network of collaborating nodes is used to pool resources. The P2P network acts as a distributed cloud storage solution with built-in redundancy. In theory, any type of data can be hosted and served from such a decentralized network, including off-chain DApp data and the files constituting the front-end for DApps.
- It could be the furthermost well-known disseminated storage explanation is the IPFS stands, its uses a distributed hash table data-structure to collection contented crossways a network of nodes. However, IPFS content is not guaranteed to be available, unless the original data owner keeps serving it from its own host. This is due to the fact that content propagation through the network is prioritized according to popularity and unpopular content may be garbage collected. The nonexistence of encouragements or nodes to host contented is an over-all problem in distributed storage solutions.

5.7.2 Swarm Architecture

Swarm is Ethereum's implementation of a decentralized file storage network. This one is reinforced by the Ethereum Geth client and interrelating with the storage network is closely linked to the Ethereum blockchain and requires an Ethereum account.

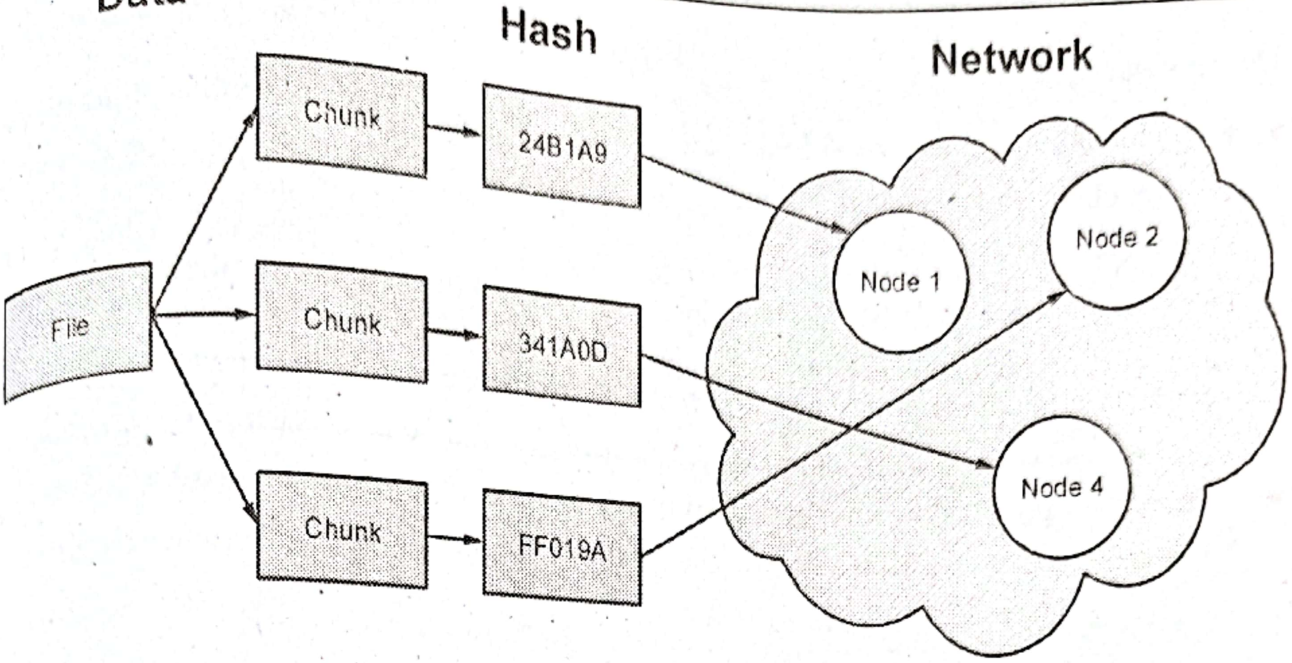


Fig. 5.7.2 : Swarm distributed storage model

- The overhead figure demonstrates by what means Ethereum Swarm distributes data across the P2P network. Data is divided up into blocks called chunks, maximum size limit of 4K bytes. The network layer is agnostic to what these chunks represent, for instance, whether they are part of a file or any other piece of data. Chunks are distributed across the network and addressed by a 32-byte hash of their content. This guarantees that data integrity can be proved, but announces a problem with storing contented that may be altered. Hash addressing is also not actual user-friendly. For this reason, another layer, the Ethereum Name Service (ENS) allows users to register human-readable names for their content. ENS is implemented as a smart contract on the Ethereum network and can be well-thought-out the equivalent of the domain name service (DNS) that facilitates content naming in traditional internet services.

Incentive layer :

- Ethereum Swarm distinguishes themself from IPFS in that it does not just reference and possibly cache or contented through accessible on a content owner's own storage. Instead, it actually constitutes a cloud service onto which content can be uploaded.
- Currently, there is no guarantee uploaded content will remain available, as nodes may join and leave the network at will or even reduce their storage capacity. A forthcoming enticement layer is prearranged, in order to pay compensation node owners for contribution storage space. Close integration with Ethereum made this possible.

Using swarm :

- In the direction of associate to Ethereum Swarm a running instance of Geth is required. The Swarm client themself can be found from the Swarm download page for dissimilar platforms.
- Once the Swarm executable has been installed, you can then connect to the network using an existing account managed by the running Geth instance :

```
swarm –bzzaccount <accout>
```

- Swarm then provides an endpoint on port 8500. Navigating to http://localhost:8500 in a browser will open up a search box for the swarm network.

5.7.3 Whisper

- Web 3.0 is the decentralized evolution of the World Wide Web. The concept intentions to substitute centralized web applications with so-called decentralized applications (DApps), which are implemented on a trusted peer-to-peer (P2P) network without central points of control and central points of failure. In the Ethereum environment, Web 3.0 is executed in the three pillars forms, of which one is the Ethereum Whisper protocol, which is aimed to bring about the appearance of DApps and by addition Web 3.0, by substitute as a protected and distributed messaging protocol.
- First pillar of this network is smart contract technology, contract is run on the Ethereum blockchain as a trustworthy unchallengeable backend. With smart contracts, the code of the decentralized application is executed on top of a trusted P2P protocol, instead of a web server. Second pillar, distributed storage, can be originate in the form of Swarm. So its permits the off-chain measures of DApps, such as web interfaces and bigger portions of data, to be deposited in a distributed way, removing the necessity for central file storage or databases.
- Third pillar of the Web 3.0 idea includes privacy-focused safe messaging. There are a number of circumstances in which DApps essential to connect through a message bus exterior the context of blockchain transactions. Interchanging of messages point-to-point or in a broadcast fashion permits by message buses to applicants or users. Conventionally, this is accomplished by central message servers. Reasons for DApps to keep communication off-chain include :

Privacy :

- Temporary limits for the validity of a message (a time-to-live property).
- The rate of on-chain transactions.

The whisper protocol

- Ethereum Whisper is designed as a flexible and secure messaging protocol that protects user privacy. This protocol surveys a "darkness" norm, significance that it confuses message content and sender and receiver particulars to viewers, which also worth that this evidence cannot be expanded over packet investigation. This principle is akin to the Thor projects effort to provide anonymous web browsing.
- Asymmetrically or symmetrically messages are encrypted by default. A public keys for encryption and private keys for decryption used in Asymmetric encryption. This form of encryption is used for one-to-one communication. Though symmetric encryption facilitates one-to-many messages using a single encryption and decryption key. Messages are received by a participant if they can be decrypted. Thus, the owner of private keys can receive messages destined only for them. One-to-many communications can be received by anyone in possession of the correct symmetric key. By the way robust link through Ethereum means that all contributors at present have public/private key pairs, building this fully encrypted model possible.

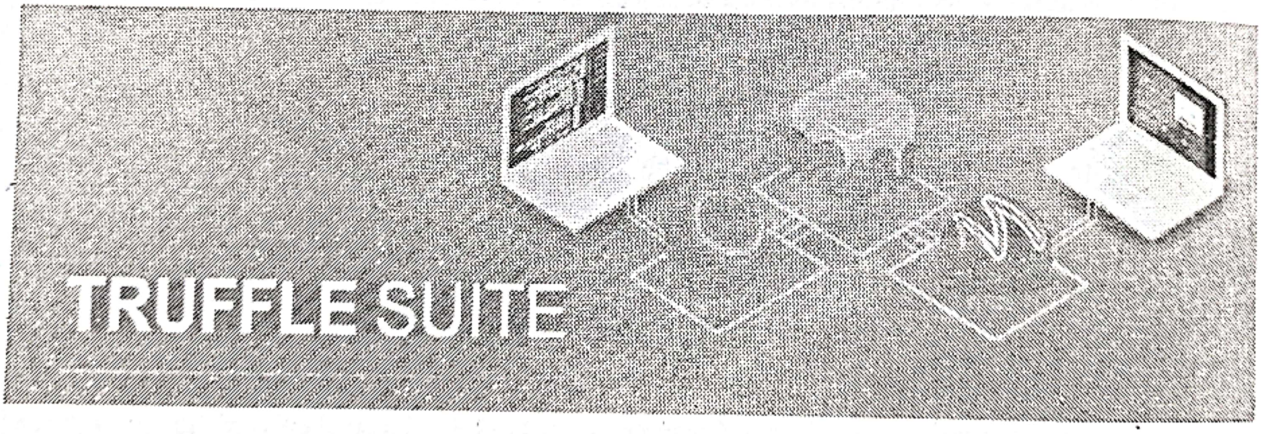
Protocol details

- RLPx transport protocol that is on the inside used by Ethereum for communication between nodes used to implement Ethereum Whisper protocol on top. While the protocol has been designed for relatively low latency (< 5 seconds), this one is not suitable for real-time communication. Due to the underlying broadcast nature of the protocol, Whisper also has bandwidth limitations. The maximum size of a message is capped at 64 K Bytes, although most messages are much smaller in practice.
- Whisper messages also have a time-to-live (TTL) associated with them, meaning that they expire after a certain time. If timeout the messages are not valid for specified time this TTL property used in Whisper protocol. TTL is advantageous in a number of circumstances, for example, when dissemination a provisional price deal in trading.
- To avoid spam, nodes must execute a proof of work algorithm to send a message. The quantity of work to be performed is relative to the size and TTL of a message.

- The envelope of encoded messages within Whisper having following fields :
  - **Version :** Version number of the protocol. This is used to identify the decryption format.
  - **Expiry :** The expiry time of the message in the form of a UNIX timestamp.
  - **TTL :** The time to live in seconds.
  - **Topic :** Arbitrary data field that can be useful as an indication of whether a message is "of interest" to a node.
  - **AESNonce :** A unique number that is used by symmetric encryption in combination with the key.
  - **Data :** The encrypted payload of the message.
  - **EnvNonce :** A number used by the proof of work algorithm.

Case studies : Study truffle development environment :

- Truffle suite is a development environment used to develop the distributed applications based on Ethereum Blockchain. Truffle is a greatest key for construction DApps which is usedful to Compiling Contracts, Deploying Contracts, Injecting it into a web app, Creating front-end for DApps and Testing.



Truffle Suite - Truffle Ethereum

- Three main components of Truffle Suite : Truffle, Ganache, Drizzle.
  - **Truffle :** Truffle is Development Environment for blockchain application, which is also work as Testing Framework and Asset pipeline for Ethereum Blokchains.
  - **Ganache :** Ganache is a personal Ethereum Blockchain. Ganache is used to test smart contracts where you can deploy contracts, develop applications, run tests and perform other tasks without any cost.

- **Drizzle :** Drizzle is a collection of libraries used to create easy and better front-end for Ethereum DApps.

Features of Truffle Ethereum

- The features of Truffle that makes Truffle a powerful tool to build Ethereum based DApps :
  - To Built-in support to compile the contracts, deploy the contracts and link smart contracts
  - Automated contract testing facility
  - Supports to console apps as well as web apps
  - Supports to network management and package management
  - Truffle console to in a straight line interconnect with smart contracts
  - Supports to close-fitting integration

Concept of MetaMask :

- MetaMask is an easy-to-use browser plugin (for Google-Chrome, Firefox and Brave browser), that provides a graphical user interface to make Ethereum transactions. It allows you to run Ethereum DApps on your browser without running a full Ethereum node on your system. Basically, MetaMask acts as a bridge between Ethereum Blockchain and the browser. MetaMask is open-source and provides the following exciting features :
  - You can change the code of MetaMask to make it what you want it to be
  - Provides built-in coin purchasing
  - Local-key Storage

Installing Truffle and Creating a Truffle Project on Ubuntu

- In this section of Truffle Ethereum , we will see how to install Truffle and how to create a Truffle project.
- To install Truffle, you will have to run a simple command as below :

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

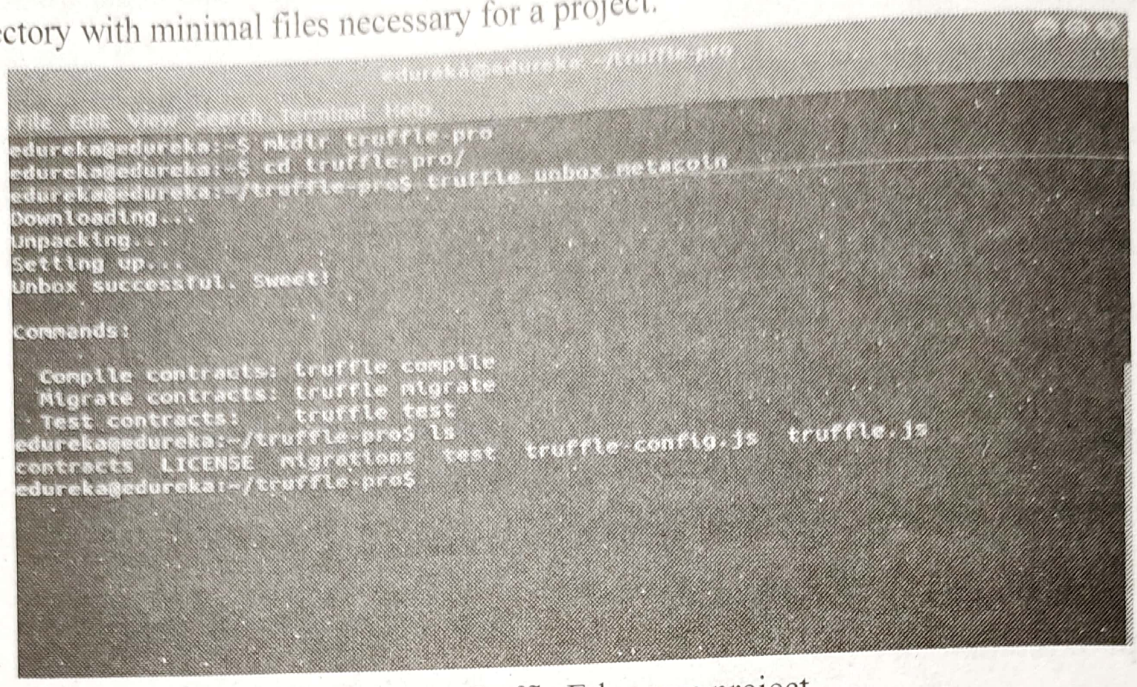
- Now, let's get to creating a project in Truffle. First, let us create a new directory and get into that directory using the following command :

```
$ mkdir truffle-pro
$ cd truffle-pro
```

- To create a project, execute the following command :

```
$ truffle unbox metacoin
```

- When this command is successfully executed, you will see a project structure present in that directory with minimal files necessary for a project.



- That's it! You have created a simple Truffle Ethereum project.

Review Questions

Q.1	What is Ethereum ? Also explain features of Ethereum.	**(5 Marks)**
Q.2	Explain in brief concept of smart contract.	**(5 Marks)**
Q.3	What is EVM ? Explain architecture of EVM.	**(6 Marks)**
Q.4	Explain working of smart contracts on EVM.	**(4 Marks)**
Q.5	Describe the flow of smart contract execution on EVM.	**(4 Marks)**
Q.6	Write short note on ether and gas concepts related to Ethereum.	**(4 Marks)**
Q.7	Elaborate in details Life cycle of smart contract with functioning.	**(6 marks)**
Q.8	Describe in details real time application of Ethereum.	**(4 marks)**
Q.9	Write a short note on Ethereum network.	**(3 marks)**
Q.10	Explain types of Ethereum network in details.	**(5 Marks)**
Q.11	Write a short note on Decentralized Applications (DApps).	**(5 Marks)**
Q.12	Need of smart contract over traditional contract system.	**(4 Marks)**
Q.13	Explain swarm in details.	**(4 Marks)**
Q.14	Write a short note on whisper.	**(4 Marks)**

❑❑❑

Unit VI

6 Blockchain Case Studies

Syllabus

Prominent Blockchain Applications, Retail, Banking and Financial Services, Government Sector, Healthcare, IOT, Energy and Utilities, Blockchain Integration with other Domains.

Contents

6.1 Prominent Blockchain Applications

6.1.1 Blockchain Applications

- The idea of a blockchain was first came in function as the mechanism supporting to Bitcoin (CRYPTO:BTC). To solve the problem of double spending associated with digital currencies, Satoshi Nakamoto developed an unchallengeable ledger of transactions that chains the blocks of data together using digital cryptography. While the idea works extremely fine for Bitcoin and other cryptocurrencies. There are loads of other useful and prominent applications of blockchain technology. Here are some prominent of of them.

1. Money transfers
2. Financial exchanges
3. Lending
4. Insurance
5. Real estate
6. Secure personal information
7. Voting
8. Government benefits
9. Securely share medical information
10. Artist royalties
11. Non-fungible tokens
12. Logistics and supply chain tracking
13. Secure Internet of Things networks
14. Data Storage

6.2 Retail Banking and Financial Services

- Blockchain technology is a capable aspect that can bring new norms to every use case. As it is known that Blockchain is a digital, public database or ledger that minutes online transactions. Early eagerness for this technology among corporate banks, infrastructure firms and capital markets is now coming to retail banks. Today retail banking has to accept new technologies, including Blockchain, to improve their services.

- Blockchain technology in retail banking can offer three key strengths to this sector including :
  - Data Handling
  - Disintermediation
  - Trust
- With the materialization of blockchain technology, banks look more at the profit that it can bring. The experience in retail banking shows how exciting blockchain is for the clients. Most of them believe that by implementing Hyperledger,
- For example,
  1. They can change their operations fundamentally
  2. Improve security
  3. Provide better services
  4. Further the opportunities blockchain offers for retail banking in future
- Here are some of the most significant applications of blockchain technology in retail banking that one must know. The use cases given below can help you understand how and where blockchain technology can be used in retail banking more effectively and efficiently.

Remittances

- Cross-border payments are increasing day by day. what's more is that the market is about to expand its growth 3 % per year. However, traditional payment processing tends to be opaque, highly mediated and clunky, which results in higher fees. However, blockchain technology can help to reduce this effect.

ID fraud prevention

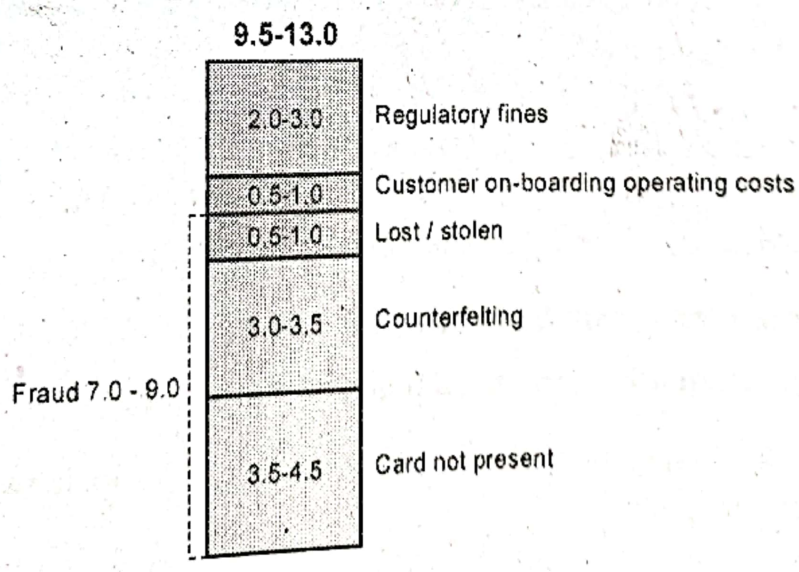


Fig. 6.2.1 : ID fraud prevention

- Blockchain technology has also been tested and rolled out for effective ID fraud prevention and detection. With private key management, blockchain technology can help the customers to share and control their data without any intermediary.

Risk assessment with the use of customer data

- Retail banking companies can use blockchain technology to gather a large volume of data that can be protected and anonymized by the encryption protocols of ledgers. Additionally, to make better risk-management decisions, banks can view data theoretically that any bank has uploaded on the network. The result would be in the form of more efficient processes, potential for more informed credit allocation process as well as faster decision.
- What if we could revolutionize the financial world, where old processes and paperwork are replaced by newfound cooperation, innovation and speed. Where fraud and crime could one day be put to rest by collective trust in a highly secure, shared view of the truth.
- It is already happening. Leading financial institutions are trailblazing the way forward with IBM Blockchain, working together to remove longstanding friction, create new solutions and deliver tangible business outcomes. Now it is time to come together and join hands.

6.2.1 Benefits

Operational simplification

- Blockchain enables multi-party tracking , real-time and management of bank guarantees and letters of credit.

Automated compliance

- Depend on faster and more accurate reporting with an automated compliance process that draws on immutable data records.

Faster settlement

- Benefit from the near real-time, point-to-point transfer of funds between financial institutions, removing friction and accelerating settlement.

6.2.2 Blockchain for Financial Services Solutions

1. Operate with more liquidity and speed using digital assets

- A exclusive digital representation of a financial tool enables trade with more liquidity and speed at lower cost.

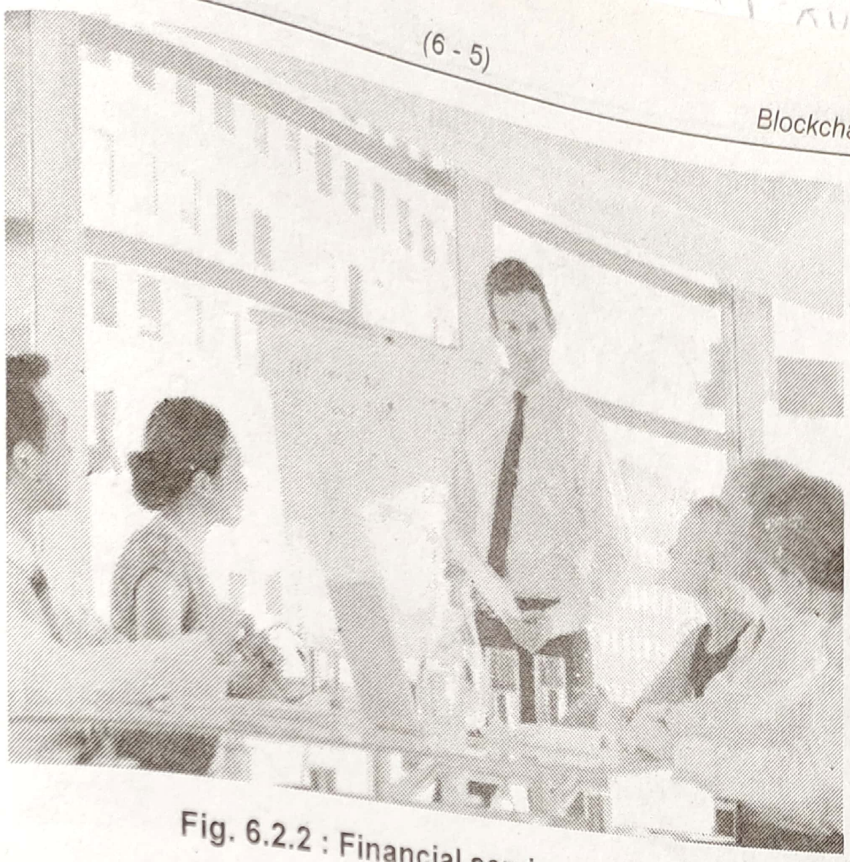


Fig. 6.2.2 : Financial service solutions

2. Transforming trade finance and trade with we.trade

Fig. 6.2.3 : Financial service solutions

- By joining we can trade, the operate finance network convened by Blockchain, businesses are creating an ecosystem of trust for global trade. Its standardized rules and simplified trading options decrease risk and increase opportunity for banks and subject matter experts.

3. ANZ bank is helping reduce the potential for fraud

Fig. 6.2.4 : Financial service solutions

- A consortium including ANZ bank is functioning together to make over financial guarantees. All participants now have a single source of information that increases efficiency and reduces the frauds.

6.3 Government Sector

- A blockchain based digital government can :
  - Protect the data
  - Streamline the processes
  - Will reduce fraud
  - Will reduce waste
- On a blockchain based government the stake holders viz model, individuals, businesses and governments share resources over a distributed ledger which is secured making use cryptography. This structure will certainly eliminates a single point of failure and intrinsically protects sensitive citizen and government data. A blockchain-based government has the latent to solve inheritance pain points and has the following prominent advantages :
  - Secure storage of government, citizen and business data
  - Reduction of labor-intensive processes

- Reduction of excessive costs associated with managing accountability
- Reduced potential for corruption and abuse
- Increased trust in government and online civil systems
- The distributed ledger system can be leveraged to support an group of government and public sector applications, including namely
  1) Digital currency/payments
  2) Land registration
  3) Identity management
  4) Supply chain traceability
  5) Health care
  6) Corporate registration
  7) Taxation
  8) Voting (elections and proxy) and
  9) Legal entities management.

6.3.1 Blockchain Use Cases in Government and Public Sector

- Blockchain use cases in government and public sector are :
  - Smart Cities
  - Tracking Vaccinations
  - Tracking Loans and Student Grants
  - Payroll Tax Collection
  - Central Banking
  - Validation of Education and Professional Qualifications

1) Impact of blockchain on smart cities

- A smart city uses information technology and data to incorporate and administer physical, social and business infrastructures to rationalize services to its population while ensuring competent and best possible utilization of available resources. In combination with technologies, IoT, cloud computing and blockchain technology, governments can deliver pioneering services and solutions to the citizens.

2) Impact of blockchain on central banking

- Real-time gross completion is the constant process of settling interbank payments in central bank records as contrasting to settlement at the end of each day. Blockchain enables a major increase in business volume and network buoyancy which enables central banks to route RTGS at a faster pace, with finely tuned security.

3) Impact of blockchain on validation of educational and professional qualifications

- Keeping academic and professional achievement data on an encrypted uniqueness wallet empowers individuals to control access to their data. It also enables schools, colleges, universities, organizations and employers to verify and validate attestations for courses and work achieved.

4) Impact of blockchain on vaccination tracking

- Recording of the vaccination information on the blockchain enables schools, colleges, organizations ,insurance and medical providers to verify and validate vaccinations swiftly. This development repeatedly triggers corresponding micropayments and delegates access to benefits based on medical status.

5) Impact of blockchain manage tracking student loans and grants

- Intelligent and smart contracts can be planned, developed and programmed to manage loan and grant applications, distribute loans and track compliance with the terms and conditions. This automated performance tracking enables real-time data and increased transparency, compliance and security.

6) Impact of blockchain collection of payroll tax

- Smart contracts can streamline the tax collection process by matching tax data with income transactions and calculating tax and social security deductions. A blockchain-based system automatically transfers net salary and tax payments to their respective recipients. Coordinated automation brings efficiency, speed and security to tax collection.

6.4 Health Care

- Blockchain is an rising technology being practically applied for creating innovative solutions in various sectors, including healthcare. Thus, it can improve the performance, security and transparency of sharing medical data in the health care system. This technology is helpful to medical institutions to gain insight and enhance the analysis of medical records. In this paper, we studied blockchain technology and its significant benefits

in healthcare. A blockchain network is used in the healthcare system to preserve and exchange patient data through hospitals, diagnostic laboratories, pharmacy firms and physicians. Blockchain applications can accurately identify severe mistakes and even dangerous ones in the medical field. Various capabilities, enablers and unified work-flow process of blockchain technology to support healthcare globally are discussed diagrammatically.

Blockchain plays a decisive part in handling deception in clinical trials; here, the potential of this technology offer is to improve data efficiency for healthcare. It can help avoid the fear of data manipulation in healthcare and supports a unique data storage pattern at the highest level of security. It provides versatility, interconnection, accountability and authentication for data access. For different purposes, health records must be kept safe and confidential. Blockchain helps for the decentralised protection of data in healthcare and avoids specific threats. Fig. 6.4.1 shows blockchain in healthcare.

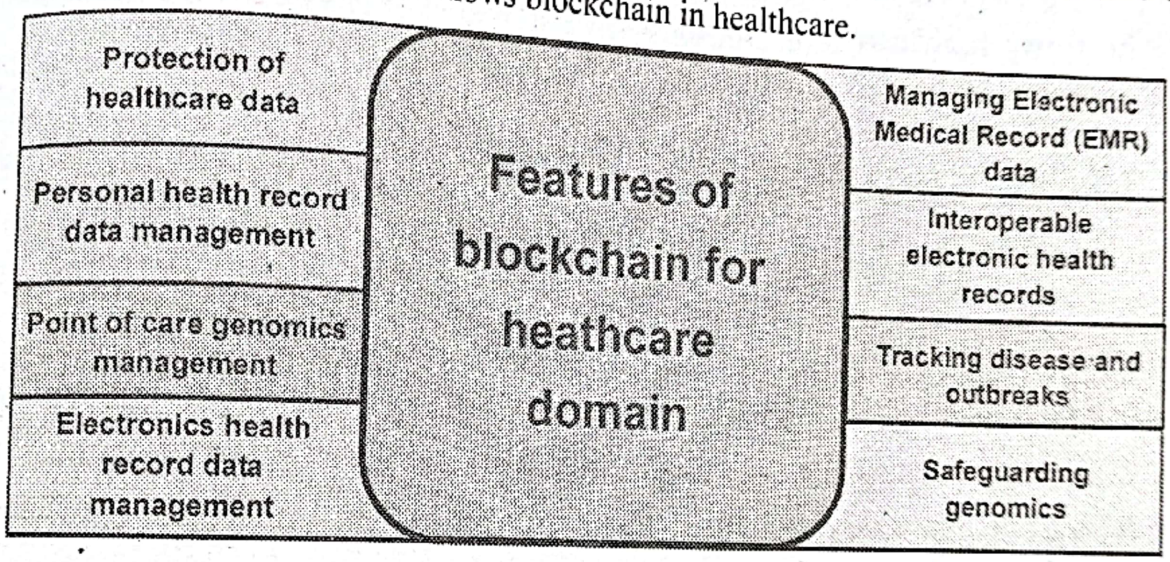


Fig. 6.4.1 : Blockchain in healthcare

6.4.1 Blockchain Healthcare Use Cases in Digital Health

1. Supply chain transparency

- A major test across the healthcare area is ensuring the origin of medical goods to prove their authenticity. Using a blockchain-based system to track items from the manufacturing point and at each stage through the supply chain enables customers to have full visibility and transparency of the goods they are buying.

2. Patient-centric electronic health records

- Healthcare systems in every country and region are stressed and struggling with the problem of data siloes, meaning that patients and their healthcare providers have an incomplete view of medical histories.

3. Smart contracts for insurance and supply chain settlements

- Companies such as Chronicled and Curisium provide blockchain-based systems where various players in the healthcare sector, such as pharmaceutical companies.

4. Medical staff credential verification

- Similar to tracking the attribution of a medical good, blockchain technology can be used to track the experience of medical professionals, where trusted medical institutions and healthcare organizations can record the credentials of their staff, in turn helping to streamline the hiring process for healthcare organizations

5. IoT security for remote monitoring

- One of the biggest trends in digital health is the adoption of remote monitoring solutions, where all kinds of sensors measuring patients' vital signs are being used to help give healthcare practitioners more visibility into patients' health, enabling more proactive and preventative care.

6.5 Internet of Things (IoT)

- Blockchain empowers the IoT devices to improve security and bring transparency in IoT ecosystems.
- Blockchain offers a scalable and decentralized environment to IoT devices, platforms and applications. Banks and Financial institutes like ING, Deutsche Bank and HSBC are doing PoC to verify and validate the blockchain technology. Apart from financial institutes, a wide range of companies have designed to practice the blockchain's potential. On the other hand, the Internet of Things (IoT) opens up limitless opportunities for businesses to run smart operations. Every device around us is now equipped with sensors, sending data to the cloud. Therefore, combining these two technologies can make the systems efficient.
- IoT with blockchain can have a important impact across multiple industries :
  - Supply chain and logistics
  - Automotive industry
  - Smart homes
  - Pharmacy industry
  - Agriculture
  - Water management
  - Sharing economy

Working of Blockchain technology for IoT use cases

The uses of blockchain IoT depend importantly on the three basic qualities of blockchain technology in the form of a data structure. The three basic properties of blockchain technology that could benefit IoT use cases include,

1. Distribution
2. Decentralization
3. Immutability

A simple explanation of using these three blockchain properties for improving IoT use cases could verify the feasibility of blockchain IoT projects. Let us assume the example of surveillance cameras as IoT devices to understand how these three traits could benefit IoT. Now, if a burglar wants to compromise a surveillance camera and prevent the crime from being recorded, they can attack the server which runs the database storing the videos.

1. Distribution

- With blockchain, the data does not stay in a single place and is distributed throughout different computers on the network. As a result, it is quite difficult to hack the surveillance system with multiple target devices. In addition, the redundancy in storage offered by blockchain improves security and data access. How ? Users in the IoT ecosystems could easily submit and retrieve their desired data from various devices effortlessly.

2. Immutability

- The blockchain IoT use cases are also evident in examples where the burglar might claim that video evidence recorded by surveillance cameras is forged. In such cases, the immutability of blockchain comes to mind. It implies the detection of any changes in the stored data. As a result, the court could verify the burglar's claim by searching for attempts to modify the data.

3. Decentralization

- Although immutability and distribution safeguard the integrity of IoT device data on blockchain networks, decentralization could be a prominent setback. Decentralization could open up sensitive data of users to third parties. However, it is possible to find a way around such setbacks. The IoT blockchain use cases could involve the storage of access logs and permissions as a preferred solution.

6.6 Energy and Utilities

- As an industry repeatedly catalyzed by innovations from solar panels to smart meters, the energy sector is constantly being disrupted. Often extensive with inefficiencies, blockchain is becoming an increasingly important tool here , credit to its ability to remove mediators, provide transparent immutable records, offer heightened security and computerize settlements. More and more blockchain use cases in energy and utilities are emerging. From peer-to-peer energy trading markets to tracing and verifying the origin of energy, blockchain in energy and utilities is gaining traction fast.
- **Blockchain use cases in energy and utilities are :**

1. **Peer-to-Peer energy trading :** Peer-to-Peer energy (P2P) trading allows consumers to buy and sell excess energy amongst themselves. And credit to distributed ledger technology, blockchain is enabling this by removing mediators, allowing for a truly peer-to-peer exchange. This cuts down the cost of excess solar energy being escorted back to the grid and gives consumers more control to purchase energy from their neighbors at a cheaper price than their provider.
2. **Renewable Energy Certificates (RECs) :** Another compelling use case can be found for blockchain in energy and sustainability, particularly when it comes to Renewable Energy Certificates (RECs). Through the use of a cryptographically secure unchallengeable digital ledger, providers can mitigate fraud and trace and validate REC transactions instantly and automatically.
3. **Automatic settlement of trades :** Automatic settlement of trades is perhaps one of the most everywhere ways that all types of energy and utilities providers can benefit from blockchain technology. credit to its ability to remove the middlemen, providers can automatically record and settle transactions. Trades can even be done on a peer-to-peer basis thanks to smart contracts and carried out between providers without the need for a clearing house or broker, making for a more streamlined and cost-effective approach.
4. **Microgrids :** Blockchain investment in the energy sector is expected to reach more than $5.8 billion by 2025 with microgrids playing a leading role reducing transmission losses and deferring expensive network upgrades. The nature of microgrid's distributed generation offers efficient energy management, continuity of supply, as well as a reliable back-up power to safeguard against outages.
5. **Smart meters :** Blockchain utilities use cases are also on the rise with a key example being smart meters. Smart meters let for a more accurate recording of the energy that consumers' use since power usage is determined at regular intervals throughout the day. This information is then sent back to the utility sector, traditionally via several intermediaries.

6.7 Blockchain Integration with Other Domains

- The blockchain technology can be used in creation of create digital assets such as :
  - Stocks
  - Bonds,
  - Land titles
  - Telecommunication industry to keep call records
  - Frequent flyer miles.
- Blockchain is growing very swiftly. Due to the benefits, it has many industries and companies to have started using blockchain. It is said that 75 % of the companies in the world will be using blockchain.

DNS on Blockchain : The next evolution of domain names

- The DNS, the Domain Name System, is a service at the centre of how the Internet operates. It functions as a public directory that associates domain names with resources on the Internet, such as IP addresses. When a user enters an address in his browser, a DNS server translates this humanly understandable address into an IP address that is understandable by computers and networks.
- A Blockchain is a data structure accessible to all and distributed over a decentralized network; the data is replicated on each node of the network, there is no central authority. Everyone has the possibility to read its contents, add data and even join the network. The concept was first implemented in 2009 with Bitcoin but today there are many different Blockchain technologies, each with their own properties.
- The data is entered on a blockchain via transactions. The transactions are grouped into blocks, each block is then validated by the network and then brought together. Thus, a Blockchain contains the history of all the transactions carried out since its creation.

Review Questions

Q.1 List and explain applications of blockchain technology. **(Refer Section 6.1.1)** **(7 Marks)**

Q.2 Write a short notes on application of blockchain technology. **(Refer Section 6.1.1)** **(7 Marks)**

Q.3 How blockchain technology can be used in financial and banking services. **(Refer Section 6.2)** **(7 Marks)**

Q.4 Write a short notes on use of blockchain technology in government sector. **(Refer Section 6.3)** **(6 Marks)**

Q.5 Write a short notes on usage of blockchain in healthcare sector. **(Refer Section 6.4)** **(6 Marks)**

Q.6 Explain blockchain significance in IOT. **(Refer Section 6.5)** **(6 Marks)**

Q.7 Explain one case study how blockchain supports IOT. **(Refer Section 6.5)** **(7 Marks)**

Q.8 Explain importance of blockchain in enery and utilities. **(Refer Section 6.6)** **(7 Marks)**

❑❑❑