UNIT-I

Blockchain is a buzzword in today's technology and this technology is described as the most disruptive technology of the decade. Thus, Blockchain is used for the secure transference of items like money, contracts, property rights, stocks, and even networks without any requirement of Third Party Intermediaries like Governments, banks, etc. Once the data is stored in the Blockchain it becomes very difficult to manipulate the stored data. A Blockchain is a Network Protocol like SMTP. However, Blockchain cannot be run without the Internet. BlockChain is useful in many areas like Banking, Finance, Healthcare, Insurance, etc.

A blockchain is an open, distributed ledger that can record transactions between two parties efficiently and in a verifiable and permanent way without the need for a central authority.

Key Characteristics:

- **Open**: Anyone can access blockchain.
- **Distributed or Decentralised:** Not under the control of any single authority.
- **Efficient:** Fast and Scalable.
- **Verifiable:** Everyone can check the validity of information because each node maintains a copy of the transactions.
- **Permanent:** Once a transaction is done, it is persistent and can't be altered.

Blockchain can be defined as the Chain of Blocks that contain some specific Information. Thus, a Blockchain is a ledger i.e file that constantly grows and keeps the record of all transactions permanently. This process takes place in a secure, chronological (Chronological means every transaction happens after the previous one) and immutable way. Each time when a block is completed in storing information, a new block is generated.

Distributed Systems:

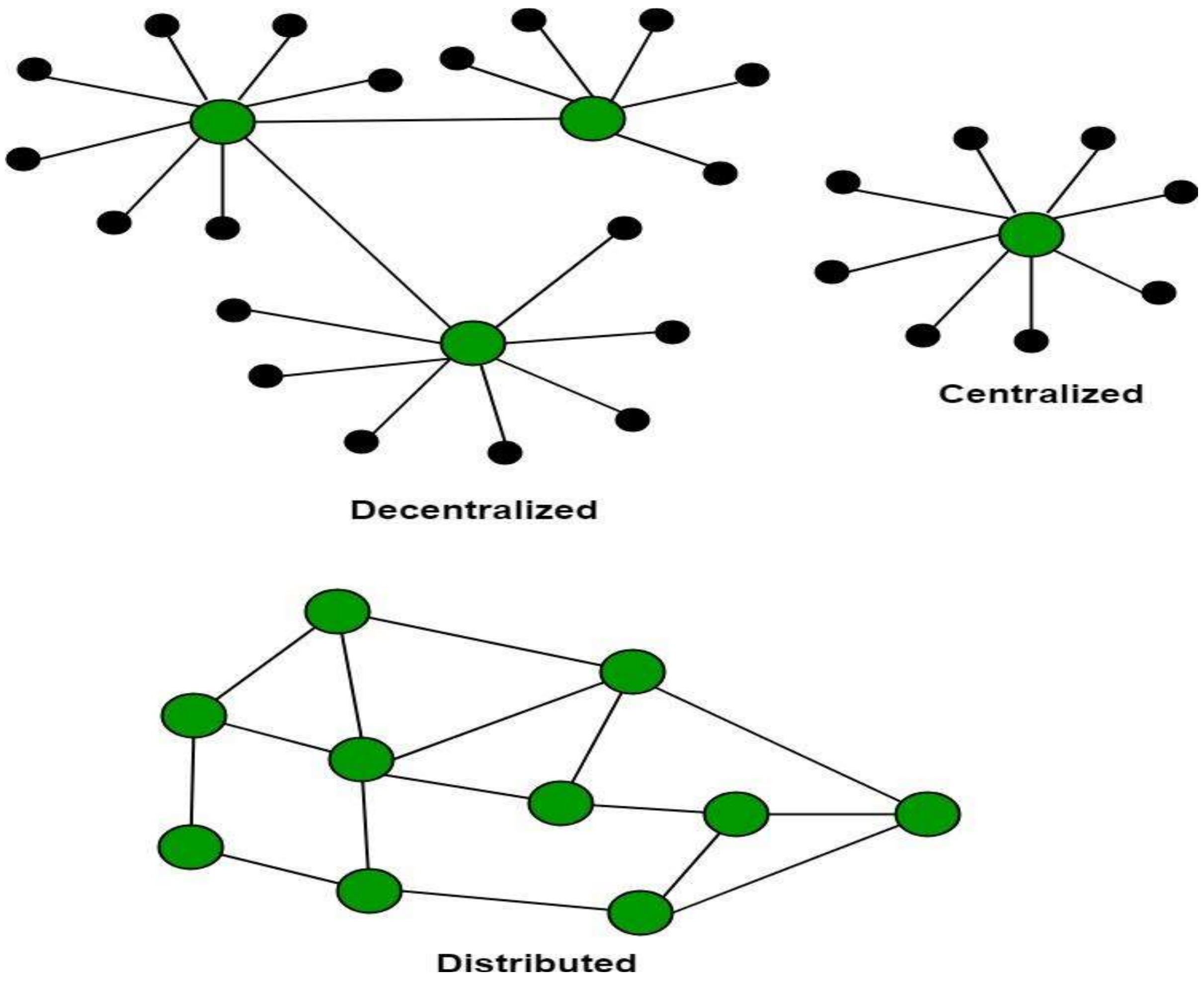
Understanding distributed systems is essential to our understanding blockchain, as blockchain was a distributed system at its core. It is a distributed ledger that can be centralized or decentralized. A blockchain is originally intended to be and is usually used as a decentralized platform. It can be thought of as a system that has properties of the both decentralized and distributed paradigms. It is a decentralized-distributed system.

Distributed systems are a computing paradigm whereby two or more nodes work with each other in a coordinated fashion to achieve a common outcome. It is modeled in such a way that end users see it as a single logical platform. For example, Google's search engine is based on a large distributed system; however, to a user, it looks like a single, coherent platform.

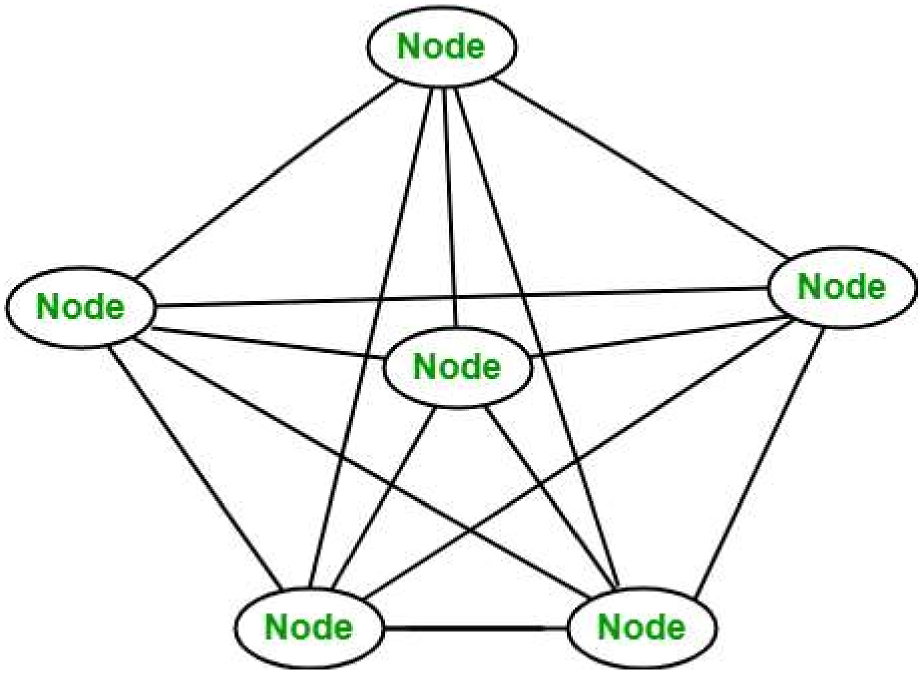
A **node** can be defined as an individual player in a distributed system. All nodes are capable of sending and receiving messages to and from each other. There is no Central Server or System which keeps the data of Blockchain. The data is distributed over Millions of Computers around the world which are connected with the Blockchain. This system allows Notarization of Data as it is present on every Node and is publicly verifiable.A node can be defined as an individual player in a distributed system. All nodes are capable of sending and receiving messages to and from each other.

Nodes can be honest, faulty, or malicious and have their own memory and processor. A node that can exhibit arbitrary behavior is also known as a Byzantine node. This arbitrary behavior can be intentionally malicious, which is detrimental to the operation of the network. Generally, any unexpected behavior of a node on the network can be categorized as Byzantine. This term arbitrarily encompasses any behavior that is unexpected or malicious.

The main challenge in distributed system design is coordination between nodes and fault tolerance. Even if some of the nodes become faulty or network links break, the distributed system should tolerate this and should continue to work flawlessly in order to achieve the desired result. This has been an area of active research for many years and several algorithms and mechanisms has been proposed to overcome these issues.



A network of nodes: A node is a computer connected to the Blockchain Network. Node gets connected with Blockchain using the client. Client helps in validating and propagates transaction on to the Blockchain. When a computer connects to the Blockchain, a copy of the Blockchain data gets downloaded into the system and the node comes in sync with the latest block of data on Blockchain. The Node connected to the Blockchain which helps in the execution of a Transaction in return for an incentive is called Miners.



Disadvantages of current transaction system:

- Cash can only be used in low amount transaction locally.
- Huge waiting time in the processing of transactions.
- Need to third party for verification and execution of Transaction make the process complex.
- If the Central Server like Banks is compromised, whole System is affected including the participants.
- Organization doing validation charge high process thus making the process expensive.

Building trust with Blockchain:

Blockchain enhances trust across a business network. It's not that you can't trust those who you conduct business with its that you don't need to when operating on a Blockchain network.
Blockchain builts trust through the following five attributes:

- **Distributed:** The distributed ledger is shared and updated with every incoming transaction among the nodes connected to the Blockchain. All this is done in real-time as there is no central server controlling the data.
- **Secure:** There is no unauthorized access to Blockchain made possible through Permissions and Cryptography.

- **Transparent:** Because every node or participant in Blockchain has a copy of the Blockchain data, they have access to all transaction data. They themselves can verify the identities without the need for mediators.
- **Consensus-based:** All relevant network participants must agree that a transaction is valid. This is achieved through the use of consensus algorithms.
- **Flexible:** Smart Contracts which are executed based on certain conditions can be written into the platform. Blockchain Network can evolve in pace with business processes.

History of Blockchain:

- In 1991, researcher scientists named Stuart Haber and W. Scott Stornetta introduce Blockchain Technology. These scientists wanted some Computational practical Solution for time-stamping the digital documents so that they couldn't be tempered or misdated. So both scientists together developed a system with the help of Cryptography. In this System, the time-stamped documents are stored in a Chain of Blocks.

- After that in 1992, Merkle Trees formed a legal corporation by using a system developed by Stuart Haber and W. Scott Stornetta with some more features. Hence, Blockchain Technology became efficient to store several documents to be collected into one block. Merkle used a Secured Chain of Block which stores multiple data records in a sequence. However, this Technology became unused when Patent came into existence in 2004.

- However, in the same year 2004, Cryptographic activist Hal Finney introduced a system for digital cash known as "Reusable Proof of Work". This step was the game-changer in the history of Blockchain and Cryptography. This System helps others to solve the Double Spending Problem by keeping the ownership of tokens registered on a trusted server.

- After that in 2008, Satoshi Nakamoto conceptualized the concept of "Distributed Blockchain" under his white paper: "A Peer to Peer Electronic Cash System". He modified the model of Merkle Tree and created a system that is more secure and contains the secure history of data exchange. His System follows a peer-to-peer network of time stamping. His system became so useful that Blockchain become the backbone of

Cryptography.

- After that, the evolution of Blockchain is steady and promising and became a need in various fields. Blockchain technology is so secure that the following surprising news will give proof about that. A person named, James Howells was an IT worker in the United Kingdom, he starts mining bitcoins which are part of Blockchain in 2009 and stopped this in 2013. He spends $17,000 on it and after he stopped he sells the parts of his laptop on eBay and keep the drive with him so that when he needs to work again on bitcoin he will utilize it but while cleaning his house in 2013, he thrashed his drive with garbage and now his bitcoins cost nearly $127 million. This money now remains unclaimed in the Bitcoin system.

The blockchain is the public ledger of all Bitcoin transactions that have ever been exe- cuted. It is constantly growing as miners add new blocks to it (every 10 minutes) to record the most recent transactions. The blocks are added to the blockchain in a lin- ear, chronological order. Each full node (i.e., every computer connected to the Bitcoin network using a client that performs the task of validating and relaying transactions) has a copy of the blockchain, which is downloaded automatically when the miner joins the Bitcoin network. The blockchain has complete information about addresses and balances from the genesis block (the very first transactions ever executed) to the most recently completed block.

Blockchain is the backbone Technology of Digital CryptoCurrency BitCoin. The blockchain is a distributed database of records of all transactions or digital event that have been executed and shared among participating parties. Each transaction verified by the majority of participants of the system. It contains every single record of each transaction. BitCoin is the most popular cryptocurrency an example of the blockchain. Blockchain Technology Records Transaction in Digital Ledger which is distributed over the Network thus making it incorruptible. Anything of value like Land Assets, Cars, etc. can be recorded on Blockchain as a Transaction.

One of the famous use of Blockchain is Bitcoin. The bitcoin is a cryptocurrency and is used to exchange digital assets online. Bitcoin uses cryptographic proof instead of third-party trust for two parties to execute transactions over the internet. Each transaction protects through digital signature.

CAP theorem:

The **CAP theorem**, also known as Brewer's theorem, was introduced by Eric Brewer in 1998 as a conjecture. In 2002, it was proven as a theorem by Seth Gilbert and Nancy Lynch. The theorem states that any distributed system cannot have consistency, availability, and partition tolerance simultaneously:

- **Consistency** is a property that ensures that all nodes in a distributed system have a single, current, and identical copy of the data.

  Consistency is achieved using consensus algorithms in order to ensure that all nodes have the same copy of the data. This is also called **state machine replication**. The blockchain is a means for achieving state machine replication.

- **Availability** means that the nodes in the system are up, accessible for use, and are accepting incoming requests and responding with data without any failures as and when required. In other words, data is available at each node and the nodes are responding...

The CAP theorem states that **a distributed database system has to make a tradeoff between Consistency and Availability when a Partition occurs**. A distributed database system is bound to have partitions in a real-world system due to network failure or some other reason.

The CAP Theorem is comprised of three components (hence its name) as they relate to distributed data stores:

Consistency. All reads receive the most recent write or an error.

Availability. All reads contain data, but it might not be the most recent.

Partition tolerance.

The CAP Theorem is comprised of three components (hence its name) as they relate to distributed data stores:

- **Consistency.** All reads receive the most recent write or an error.
- **Availability.** All reads contain data, but it might not be the most recent.
- **Partition tolerance.** The system continues to operate despite network failures (ie; dropped partitions, slow network connections, or unavailable network connections between nodes.)

In normal operations, your data store provides all three functions. But the CAP theorem maintains that when a distributed database experiences a network failure, you can provide either consistency or availability.

It's a tradeoff. All other times, all three can be provided. But, in the event of a network failure, a choice must be made.In the theorem, partition tolerance is a must. The assumption is that the system operates on a distributed data store so the system, by nature, operates with network partitions. Network failures will happen, so to offer any kind of reliable service, partition tolerance is necessary—the P of CAP.
That leaves a decision between the other two, C and A. When a network failure happens, one can choose to guarantee consistency or availability:

- High consistency comes at the cost of lower availability.
- High availability comes at the cost of lower consistency.

Benefits and limitations of blockchain:

Numerous benefits of blockchain technology are being discussed in the industry and proposed by thought leaders around the world in blockchain space. The top 10 benefits are listed and discussed as follows.

Decentralization :

This is a core concept and benefit of blockchain. There is no need for a trusted third party or intermediary to validate transactions; instead a consensus mechanism is used to agree on the validity of transactions.

Transparency and trust :

As blockchains are shared and everyone can see what is on the blockchain, this allows the system to be transparent and as a result trust is established. This is more relevant in scenarios such as the disbursement of funds or benefits where personal discretion should be restricted.

Immutability:

Once the data has been written to the blockchain, it is extremely difficult to change it back. It is not truly immutable but, due to the fact that changing data is extremely difficult and almost impossible, this is seen as a benefit to maintaining an immutable ledger of transactions.

High availability:

As the system is based on thousands of nodes in a peer-to-peer network, and the data is replicated and updated on each and every node, the system becomes highly available. Even if nodes leave the network or become inaccessible, the network as a whole continues to work, thus making it highly available.

Highly secure:

All transactions on a blockchain are cryptographically secured and provide integrity.

Simplification of current paradigms:

The current model in many industries such as finance or health is rather disorganized, wherein multiple entities maintain their own databases and data sharing can become very difficult due to the disparate nature of the systems. But as a blockchain can serve as a single shared ledger among interested parties, this can result in simplifying this model by reducing the complexity of managing the separate systems maintained by each entity.

Faster dealings:

In the financial industry, especially in post-trade settlement functions, blockchain can play a vital role by allowing the quicker settlement of trades as it does not require a lengthy process of verification, reconciliation, and clearance because a single version of agreed upon data is already available on a shared ledger between financial organizations.

Cost saving:

As no third party or clearing houses are required in the blockchain model, this can massively eliminate overhead costs in the form of fees that are paid to clearing houses or trusted third parties.

Decentralization:

Decentralization is a core benefit and service provided by blockchain technology. By design, blockchain is a perfect vehicle for providing a platform that does not need any intermediaries and that can function with many different leaders chosen via consensus mechanisms. This model allows anyone to compete to become the decision-making authority. A consensus mechanism governs this competition, and the most famous method is known as **Proof of Work** (**PoW**).

Decentralization is applied in varying degrees from a semi-decentralized model to a fully decentralized one depending on the requirements and circumstances. Decentralization can be viewed from a blockchain perspective as a mechanism that provides a way to remodel existing applications and paradigms, or to build new applications, to give full control to users.

Information and communication technology (**ICT**) has conventionally been based on a centralized paradigm whereby database or application servers are under the control of a central authority, such as a system administrator. With Bitcoin and the advent of blockchain technology, this model has changed, and now the technology exists to allow anyone to start a decentralized system and operate it with no single point of failure or single trusted authority. It can either be run autonomously or by requiring some human intervention, depending on the type and model of governance used in the decentralized application running on the blockchain.

The following diagram shows the different types of systems that currently exist: central, distributed, and decentralized.

Different types of networks/systems

Centralized systems are conventional (client-server) IT systems in which there is a single authority that controls the system, and who is solely in charge of all operations on the system. All users of a centralized system are dependent on a single source of service. The majority of online service providers, including Google, Amazon, eBay, and Apple's App Store, use this conventional model to deliver services.

In a **distributed system**, data and computation are spread across multiple nodes in the network. Sometimes, this term is confused with *parallel computing*. While there is some overlap in the definition, the main difference between these systems is that in a parallel computing system, computation is performed by all nodes simultaneously in order to achieve the result; for example, parallel computing platforms are used in weather research and forecasting, simulation, and financial modeling. On the other hand, in a distributed system, computation may not happen in parallel and data is replicated across multiple nodes that users view as a single, coherent system. Variations of both of these models are used to achieve fault tolerance and speed. In the parallel system model, there is still a central authority that has control over all nodes and governs processing. This means that the system is still centralized in nature.

The critical difference between a decentralized system and distributed system is that in a distributed system, there is still a central authority that governs the entire system, whereas in a decentralized system, no such authority exists.

A **decentralized system** is a type of network where nodes are not dependent on a single master node; instead, control is distributed among many nodes. This is analogous to a model where each department in an organization is in charge of its own database server, thus taking away the power from the central server and distributing it to the sub-departments, who manage their own databases.

A significant innovation in the decentralized paradigm that has given rise to this new era of decentralization of applications is **decentralized consensus**. This mechanism came into play with Bitcoin, and it enables a user to agree on something via a consensus algorithm without the need for a central, trusted third party, intermediary, or service provider.

We can also now view the different types of networks shown earlier from a different perspective, where we highlight the controlling authority of these networks as a symbolic hand, as shown in the following diagram. This model provides a clearer understanding of the differences between these networks from a decentralization point of view,

Different types of networks/systems depicting decentralization from a modern perspective

In the middle we have distributed systems, where we still have a central controller but the system comprises many dispersed nodes. On the right-hand side, notice that there is no hand/controller controlling the networks.

This is the key difference between decentralized and distributed networks. A decentralized system may look like a distributed system from a topological point of view, but it doesn't have a central authority that controls the network.

A traditional distributed system comprises many servers performing different roles

The following diagram shows a decentralized system (based on blockchain) where an exact replica of the applications and data is maintained across the entire network on each participating node:

A comparison between centralized and decentralized systems (networks/applications) is shown in the following table:

Feature	Centralized	Decentralized
Ownership	Service provider	All users
Architecture	Client/server	Distributed, different topologies
Security	Basic	More secure
High availability	No	Yes
Fault tolerance	Basic, single point of failure	Highly tolerant, as service is replicated
Collusion resistance	Basic, because it's under the control of a group or even single individual	Highly resistant, as consensus algorithms ensure defense against adversaries
Application architecture	Single application	Application replicated across all nodes on the network
Trust	Consumers have to trust the service provider	No mutual trust required
Cost for consumer	Higher	Lower

The comparison in the table only covers some main features and is not an exhaustive list of all features. There may be other features of interest that can be compared too, but this list should provide a good level of comparison.

Now we will discuss what methods can be used to achieve decentralization.

Methods of Decentralization:

Two methods can be used to achieve decentralization: disintermediation and competition. These methods will be discussed in detail in the sections that follow.

The concept of **disintermediation** can be explained with the aid of an example. Imagine that you want to send money to a friend in another country. You go to a bank, which, for a fee, will transfer your money to the bank in that country. In this case, the bank maintains a central database that is updated, confirming that you have sent the money. With blockchain technology, it is possible to send this money directly to your friend without the need for a bank. All you need is the address of your friend on the blockchain. This way, the intermediary (that is, the bank) is no longer required, and decentralization is achieved by disintermediation. It is debatable, however, how practical decentralization through disintermediation is in the financial sector due to the massive regulatory and compliance requirements. Nevertheless, this model can be used not only in finance but in many other industries as well, such as health, law, and the public sector. In the health industry, where patients, instead of relying on a trusted third party (such as the hospital record system) can be in full control of their own identity and their data that they can share directly with only those entities that they trust. As a general solution, blockchain can serve as a decentralized health record management system where health records can be exchanged securely and directly between different entities (hospitals, pharmaceutical companies, patients) globally without any central authority.

Contest-driven decentralization:

In the method involving **competition**, different service providers compete with each other in order to be selected for the provision of services by the system. This paradigm does not achieve complete decentralization. However, to a certain degree, it ensures that an intermediary or service provider is not monopolizing the service. In the context of blockchain technology, a system can be envisioned in which smart contracts can choose an external data provider from a large number of providers based on their reputation, previous score, reviews, and quality of service.

This method will not result in full decentralization, but it allows smart contracts to make a free choice based on the criteria just mentioned. This way, an environment of competition is cultivated among service providers where they compete with each other to become the data provider of choice.

In the following diagram, varying levels of decentralization are shown. On the left side, the conventional approach is shown where a central system is in control; on the right side, complete disintermediation is achieved, as intermediaries are entirely removed. Competing intermediaries or service providers are shown in the center. At that level, intermediaries or service providers are selected based on reputation or voting, thus achieving partial decentralization:

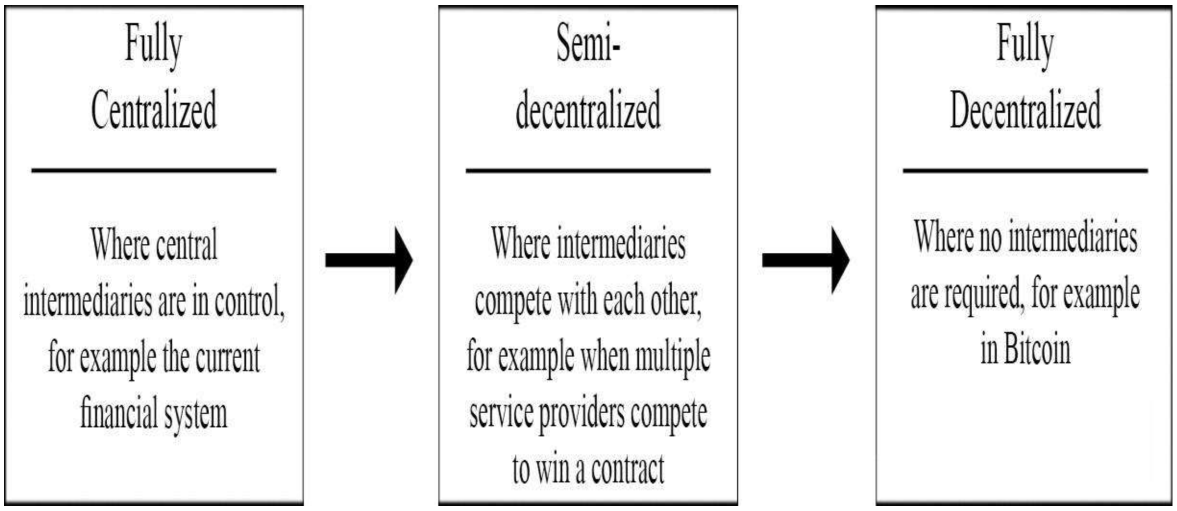

Fully Centralized
Where central intermediaries are in control, for example the current financial system
Semi-decentralized
Where intermediaries compete with each other, for example when multiple service providers compete to win a contract
Fully Decentralized
Where no intermediaries are required, for example in Bitcoin

Figure : Scale of decentralization

There are many benefits of decentralization, including transparency, efficiency, cost saving, development of trusted ecosystems, and in some cases privacy and anonymity. Some challenges, such as security requirements, software bugs, and human error, need to be examined thoroughly.

This view raises some fundamental questions. Is a blockchain really needed? When is a blockchain required? In what circumstances is blockchain preferable to traditional databases? To answer these questions, go through the simple set of questions presented below:

Question	Yes/No	Recommended solution
Is high data throughput required?	Yes	Use a traditional database.
	No	A central database might still be useful if other requirements are met. For example, if users trust each other, then perhaps there is no need for a blockchain. However, if they don't or trust cannot be established for any reason, blockchain can be helpful.

Are updates centrally controlled?	Yes	Use a traditional database.
	No	You may investigate how a public/private blockchain can help.
Do users trust each other?	Yes	Use a traditional database.
	No	Use a public blockchain.
Are users anonymous?	Yes	Use a public blockchain.
	No	Use a private blockchain.
Is consensus required to be maintained within a consortium?	Yes	Use a private blockchain.
	No	Use a public blockchain.
Is strict data immutability required?	Yes	Use a blockchain.
	No	Use a central/traditional database.

Answering all of these questions can help you decide whether or not a blockchain is required or suitable for solving the problem. Beyond the questions posed in this model, there are many other issues to consider, such as latency, choice of consensus mechanisms, whether consensus is required or not, and where consensus is going to be achieved. If consensus is maintained internally by a consortium, then a private blockchain should be used; otherwise, if consensus is required publicly among multiple entities, then a public blockchain solution should be considered. Other aspects, such as immutability, should also be considered when deciding whether to use a blockchain or a traditional database. If strict data immutability is required, then a public blockchain should be used; otherwise, a central database may be an option.

As blockchain technology matures, there will be more questions raised regarding this selection model. For now, however, this set of questions is sufficient for deciding whether a blockchain-based solution is suitable or not.

Now we understand different methods of decentralization and have looked at how to decide whether a blockchain is required or not in a particular scenario. Let's now look at the process of decentralization, that is, how we can take an existing system and decentralize it.

Routes to decentralization:

There are systems that pre-date blockchain and Bitcoin, including BitTorrent and the Gnutella file-sharing system, which to a certain degree could be classified as decentralized, but due to a lack of any incentivization mechanism, participation from the community gradually decreased. There wasn't any incentive to keep the users interested in participating in the growth of the network. With the advent of blockchain technology, many initiatives are being taken to leverage this new technology to achieve decentralization. The Bitcoin blockchain is typically the first choice for many, as it has proven to be the most resilient and secure blockchain and has a market cap of nearly $166 billion at the time of writing. Alternatively, other blockchains, such as Ethereum, serve as the tool of choice for many developers for building decentralized applications. Compared to Bitcoin, Ethereum has become a more prominent choice because of the flexibility it allows for programming any business logic into the blockchain by using **smart contracts**.

How to decentralize

The framework raises four questions whose answers provide a clear understanding of how a system can be decentralized:

1. What is being decentralized?
2. What level of decentralization is required?
3. What blockchain is used?
4. What security mechanism is used?

The first question simply asks you to identify what system is being decentralized. This can be any system, such as an identity system or a trading system.

The second question asks you to specify the level of decentralization required by examining the scale of decentralization, as discussed earlier. It can be full disintermediation or partial disintermediation.

The third question asks developers to determine which blockchain is suitable for a particular application. It can be Bitcoin blockchain, Ethereum blockchain, or any other blockchain that is deemed fit for the specific application.

Finally, a fundamental question that needs to be addressed is how the security of a decentralized system will be guaranteed. For example, the security mechanism can be atomicity-based, where either the transaction executes in full or does not execute at all. This deterministic approach ensures the integrity of the system. Other mechanisms may include one based on reputation, which allows for varying degrees of trust in a system.

In the following section, let's evaluate a money transfer system as an example of an application selected to be decentralized.

Decentralization framework example:

The four questions discussed previously are used to evaluate the decentralization requirements of this application. The answers to these questions are as follows:

1. Money transfer system
2. Disintermediation
3. Bitcoin
4. Atomicity

The responses indicate that the money transfer system can be decentralized by removing the intermediary, implemented on the Bitcoin blockchain, and that a security guarantee will be provided via atomicity. Atomicity will ensure that transactions execute successfully in full or do not execute at all. We have chosen the Bitcoin blockchain because it is the longest established blockchain and has stood the test of time.

Similarly, this framework can be used for any other system that needs to be evaluated in terms of decentralization. The answers to these four simple questions help clarify what approach to take to decentralize the system.

To achieve complete decentralization, it is necessary that the environment around the blockchain also be decentralized. We'll look at the full ecosystem of decentralization next.

The blockchain is a distributed ledger that runs on top of conventional systems. These elements include storage, communication, and computation.

Storage

Data can be stored directly in a blockchain, and with this fact it achieves decentralization. However, a significant disadvantage of this approach is that a blockchain is not suitable for storing large amounts of data by design. It can store simple transactions and some arbitrary data, but it is certainly not suitable for storing images or large blobs of data, as is the case with traditional database systems.

A better alternative for storing data is to use **distributed hash tables** (**DHTs**). DHTs were used initially in peer-to-peer file sharing software, such as BitTorrent, Napster, Kazaa, and Gnutella. DHT research was made popular by the CAN, Chord, Pastry, and Tapestry projects. BitTorrent is the most scalable and fastest network, but the issue with BitTorrent and the others is that there is no incentive for users to keep the files indefinitely. Users generally don't keep files permanently, and if nodes that have data still required by someone leave the network, there is no way to retrieve it except by having the required nodes rejoin the network so that the files once again become available.

Two primary requirements here are high availability and link stability, which means that data should be available when required and network links also should always be accessible. **Inter-Planetary File System** (**IPFS**) by Juan Benet possesses both of these properties, and its vision is to provide a decentralized World Wide Web by replacing the HTTP protocol. IPFS uses Kademlia DHT and Merkle **Directed Acyclic Graphs** (**DAGs**) to provide storage and searching functionality, respectively.

The incentive mechanism for storing data is based on a protocol known as Filecoin, which pays incentives to nodes that store data using the Bitswap mechanism. The Bitswap mechanism lets nodes keep a simple ledger of bytes sent or bytes received in a one-to-one relationship. Also, a Git-based version control mechanism is used in IPFS to provide structure and control over the versioning of data.

There are other alternatives for data storage, such as Ethereum Swarm, Storj, and MaidSafe. Ethereum has its own decentralized and distributed ecosystem that uses Swarm for storage and the Whisper protocol for communication. MaidSafe aims to provide a decentralized World Wide Web. All of these projects are discussed later in this book in greater detail.

BigChainDB is another storage layer decentralization project aimed at providing a scalable, fast, and linearly scalable decentralized database as opposed to a traditional filesystem. BigChainDB complements decentralized processing platforms and filesystems such as Ethereum and IPFS.

Communication

The Internet (the communication layer in blockchain) is considered to be decentralized. This belief is correct to some extent, as the original vision of the Internet was to develop a decentralized communications system. Services such as email and online storage are now all based on a paradigm where the service provider is in control, and users trust such providers to grant them access to the service as requested. This model is based on the unconditional trust of a central authority (the service provider) where users are not in control of their data. Even user passwords are stored on trusted third-party systems.

Thus, there is a need to provide control to individual users in such a way that access to their data is guaranteed and is not dependent on a single third party.

Access to the Internet (the communication layer) is based on **Internet Service Providers** (**ISPs**) who act as a central hub for Internet users. If the ISP is shut down for any reason, then no communication is possible with this model.

An alternative is to use **mesh networks**. Even though they are limited in functionality when compared to the Internet, they still provide a decentralized alternative where nodes can talk directly to each other without a central hub such as an ISP.

Now imagine a network that allows users to be in control of their communication; no one can shut it down for any reason. This could be the next step toward decentralizing communication networks in the blockchain ecosystem. It must be noted that this model may only be vital in a jurisdiction where the Internet is censored and controlled by the government.

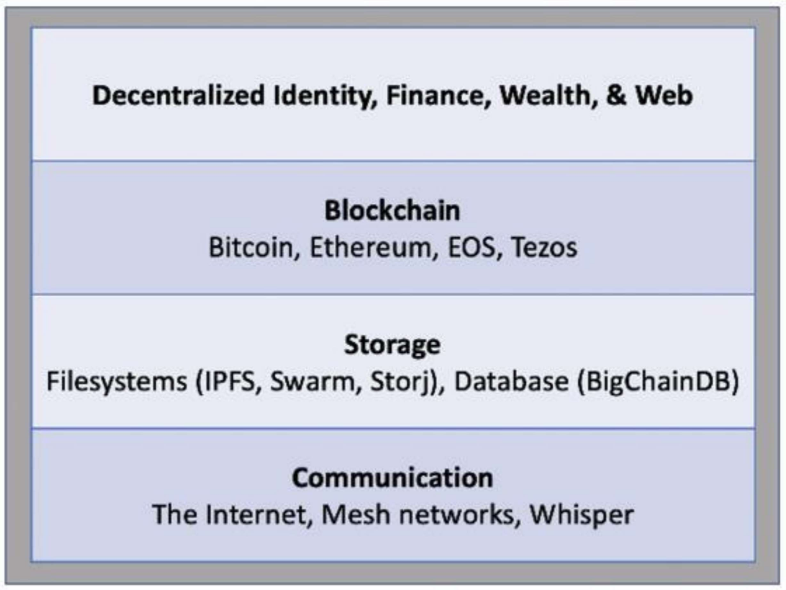
As mentioned earlier, the original vision of the Internet was to build a decentralized network; however, over the years, with the advent of large-scale service providers such as Google, Amazon, and eBay, control is shifting toward these big players. For example, email is a decentralized system at its core; that is, anyone can run an email server with minimal effort and can start sending and receiving emails. There are better alternatives available. For example, Gmail and Outlook already provide managed services for end users, so there is a natural inclination toward selecting one of these large centralized services as they are more convenient and free to use. This is one example that shows how the Internet has moved toward centralization.

Free services, however, are offered at the cost of exposing valuable personal data, and many users are unaware of this fact. Blockchain has revived the vision of decentralization across the world, and now concerted efforts are being made to harness this technology and take advantage of the benefits that it can provide.

Computing power and decentralization:

Decentralization of computing or processing power is achieved by a blockchain technology such as Ethereum, where smart contracts with embedded business logic can run on the blockchain network. Other blockchain technologies also provide similar processing-layer platforms, where business logic can run over the network in a decentralized manner.

The following diagram shows an overview of a decentralized ecosystem. In the bottom layer, the Internet or mesh networks provide a decentralized communication layer. In the next layer up, a storage layer uses technologies such as IPFS and BigChainDB to enable decentralization. Finally, in the next level up, you can see that the blockchain serves as a decentralized processing (computation) layer. Blockchain can, in a limited way, provide a storage layer too, but that severely hampers the speed and capacity of the system. Therefore, other solutions such as IPFS and BigChainDB are more suitable for storing large amounts of data in a decentralized way. The Identity and Wealth layers are shown at the top level. Identity on the Internet is a vast topic, and systems such as bitAuth and OpenID provide authentication and identification services with varying degrees of decentralization and security assumptions:


Decentralized Identity, Finance, Wealth, & Web
Blockchain
Bitcoin, Ethereum, EOS, Tezos
Storage
Filesystems (IPFS, Swarm, Storj), Database (BigChainDB)
Communication
The Internet, Mesh networks, Whisper

Decentralized ecosystem

The blockchain is capable of providing solutions to various issues relating to decentralization. A concept relevant to identity known as **Zooko's Triangle** requires that the naming system in a network protocol is secure, decentralized, and able to provide human-meaningful and memorable names to the users. Conjecture has it that a system can have only two of these properties simultaneously.

Nevertheless, with the advent of blockchain in the form of **Namecoin**, this problem was resolved. It is now possible to achieve security, decentralization, and human-meaningful names with the Namecoin blockchain. However, this is not a panacea, and it comes with many challenges, such as reliance on users to store and maintain private keys securely. This opens up other general questions about the suitability of decentralization to a particular problem.

Decentralization may not be appropriate for every scenario. Centralized systems with well-established reputations tend to work better in many cases. For example, email platforms from reputable companies such as Google or Microsoft would provide a better service than a scenario where individual email servers are hosted by users on the Internet.

There are many projects underway that are developing solutions for a more comprehensive distributed blockchain system. For example, Swarm and Whisper are developed to provide decentralized storage and communication for Ethereum.

UNIT-II

Cryptography in Blockchain:

The BlockChain is the invention that allows digitally generated information to be allocated without being copied. BlockChainTechnology is the heart of the new internet i.e. digital currency, BitCoin and any other online transaction. Tech experts found a big potential in this technology. “BlockChain is an incorruptible digital ledger of economic transaction that can be programmed to record not just financial transactions but virtually everything of value.”In plain layout, the data is not owned by any single computer but by a chain of computers so that the blocks of data are secured and bound to each other using chain, that technology is known as BlockChain technology. There is no transaction cost due to BlockChain, in Layman language BlockChain is a process to pass information or data from A to B in a safe and automated manner.

Cryptocurrency works on the principle of BlockChain Technology, that is why, BlockChain is the most trending item of current era, due to it’s secure nature cryptocurrency is widely accepted. It’s value is increasing day by day. Many oil industries, IBM Technologies, Mercedes Benz, Swiss Bank, Samsung, and even Google is planning to launch their own ryptocurrency in 2019 for safe and secure transactions. Now, this technology is disrupting almost every marketshare due to its popularity and demand in the world.

Satoshi Nakamoto introduced the concept of BlockChain in 2008 in the form of cryptocurrency BitCoin. It’s function is to allow users to secure and control their monetary values so that no third party like government or banks would be able to access or control it. It is a process to carry everyone to the highest grade of liability.Three technologies work behind the BlockChain Technology-

- Private Key Cryptography
- Peer 2 Peer Network
- BlockChain’s Protocol Program

Private Key Cryptography

- Peer 2 Peer Network
- BlockChain’s Protocol Program

Introduction – cryptographic primitives:

Private Key Cryptography

- Peer 2 Peer Network
- BlockChain's Protocol Program

Blockchain, as one of the crypto-intensive creatures, has become a very hot topic recently. Although many surveys have recently been dedicated to the security and privacy issues of blockchains, there still lacks a systematic examination on the cryptographic primitives in blockchains.

Since its introduction in the early 1980s (Chaum, 1982), the design of e-cash has always been one of the main research topics in the field of cryptography. However, the one without any trusted third party remained an open problem till Bitcoin (Nakamoto) launched in 2009. Due to its decentralization, unforgeability, double-spending resistance and pseudonymity, this brand new e-cash system has brought a remarkable culmination of cryptocurrency research and its applications. Based on its main framework, many new cryptocurrencies including decentralized (such as (Litecoin), Nxtcoin (Nxt)) and centralized ones (such as RScoin (Danezis and Meiklejohn, 2016)) have been proposed. The market value of these cryptocurrencies has increased more than 30 times during 2017 (from about $17 billion on 1st Jan. to $591 billion on 31st Dec.) (Coinmarketcap). As the core technology behind Bitcoin, the blockchain has demonstrated its capability of innovation and infiltration in many domains, including finance, insurance, industry, healthcare, agriculture and so on.

There are many recent surveys have been dedicated to the security and privacy issues of blockchains .

classify cryptographic primitives in blockchains into two categories: primary and optional. The former category includes cryptographic hashes and standard digital signatures that are essential for ensuring the blockchain as a globe ledger with tamper-proof, public verifiability and achievable consensus. While the latter category, mainly used for enhancing the privacy and anonymity of blockchain-based transactions, covers some special signatures (such as ring signatures), commitments, accumulators, zero-knowledge proofs and so on.

Special signature primitives for blockchains: To enhance the privacy and anonymity of transactions, some advanced signature primitives such as ring signature and multi-signature are also widely applied in blockchains.

1.Ring signatures :

Anonymity is always required in information systems (Shen et al., 2018), especially in the e-cash system. However, Bitcoin can only provide pseudonymity due to the linkability of transactions. Therefore, many new alternative cryptocurrencies have been proposed to address this problem. From a perspective of cryptography, there are many kinds of signatures for achieving anonymity, such as blind signature (Chaum, 1982), ring signature (Rivest et al., 2001), group signature (Chaum and van Heyst, 1991) and DC-nets (Chaum, 1988). However, only ring signature and its variants have been used in blockchains for anonymity.

2. One-time (ring) signatures:

Lamport in 1979 (Lamport, 1979) proposed the concept of one-time signature (OTS), where the signing key can be used *securely but only once*, and the signing key would be revealed if it is used twice or more. OTS is frequently used as a building block in constructions of encryptions and authenticated key agreements.

3. Borromean (ring) signatures:

Another interesting primitive related to ring signature and blockchain is the so-called Borromean (ring) signature (BRS), proposed by Maxwell and Poelstra in 2015 (Maxwell and Poelstra, 2015). Poelstra (Poelstra, 2017) claimed that BRS is now used in Elements (Element, 2015), Liquid (Liquid) and Monero.

4. Multi-signatures:

The primitive of multi-signature allows a single signature to work as several ordinary signatures on the same message. One of the critical requirements of multi-signature is that the single signature has the same size as one regular signature.

Assymetric cryptography:

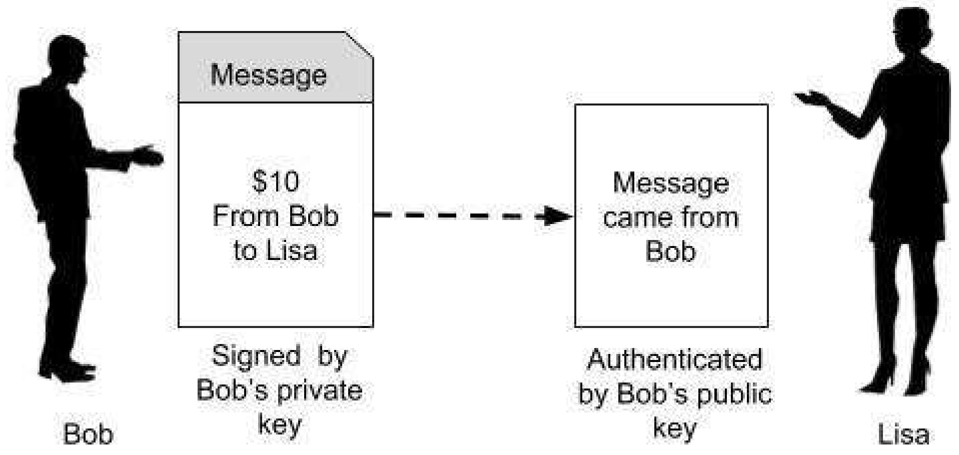
Public Key Cryptography or in short PKI is also known as asymmetric cryptography. It uses two pairs of keys - public and private. A key is a some long binary number. The public key is distributed worldwide and is truly public as its name suggests. The private key is to be strictly held private and one should never lose it.

In case of Bitcoin, if you ever lose the private key to your Bitcoin wallet, the entire contents of your wallets would be instantly vulnerable to theft and before you know it, all your money (the contents of your wallet) would be gone with no mechanism in the system to trace out who stole it - that is the anonymity in the system that I mentioned earlier.

The PKI accomplies two functions - authentication and the message privacy through encryption/decryption mechanism. I will now explain both these functions

Authentication

When the two parties exchange messages, it is important to establish a trust between the sender and the receiver. Especially, the receiver must trust the source of message. Going to our earlier scenario (depicted in Figure 1) of Bob sending some money to Lisa for purchasing of some goods from her, let us see how the PKI builds this trust between Bob and Lisa. Look at below image


Message
$10 From Bob to Lisa
Message came from Bob
Signed by Bob's private key
Authenticated by Bob's public key
Bob
Lisa

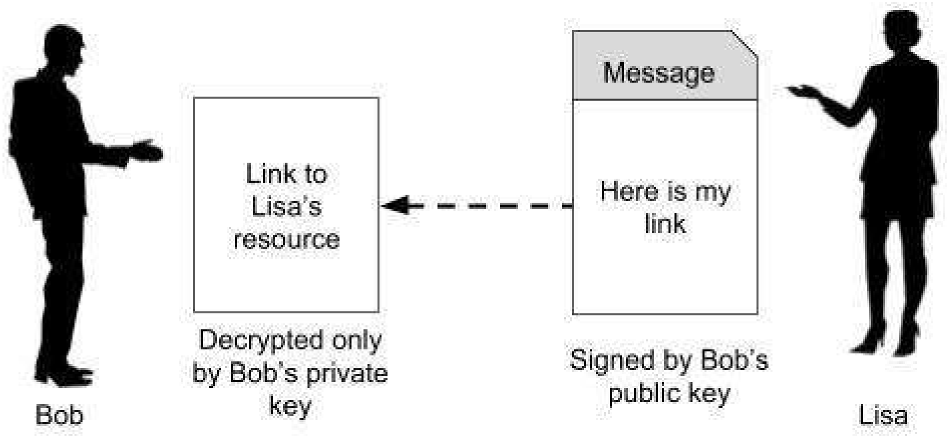
In the first place, if Bob wants to send some money to Lisa, he has to create a private/public key of its own. Note that both keys are always paired together and you can not mix the private and public keys of different individuals or different instances.

Now, Bob says that he is sending $10 to Lisa. So he creates a message (a plain-text message) containing Bob's (sender) public key, Lisa's (receiver) public key, and the amount ($10).

The purpose of this remittance such as "I want to buy pumpkin from you" is also added into the message. The entire message is now signed using Bob's private key. When Lisa receives this message, she will use the signature verification algorithm of PKI and Bob's public key to ensure that the message indeed originated from Bob. How the PKI works is beyond the scope of this tutorial. The interested reader is referred to this site for a more detailed discussion on PKI. This establishes the authenticity of the message originator. Now, let us look at the message privacy

Message Privacy:

Now, as Lisa has received her payment, she wants to send the link to her ebook which Bob wants to buy. So Lisa would create a message and send it to Bob as shown in image


Bob
Link to Lisa's resource
Decrypted only by Bob's private key
Message
Here is my link
Signed by Bob's public key
Lisa

Public and private keys -line interface:

In order to understand public key cryptography, the first concept that needs to be looked at is the idea of public and private keys.

A private key, as the names suggests, is basically a randomly generated number that is kept secret and held privately by the users. Private key needs to be protected and no unauthorized access should be granted to that key; otherwise, the whole scheme of public key cryptography will be jeopardized as this is the key that is used to decrypt messages. Private keys can be of various lengths depending upon the type and class of algorithms used. For example, in RSA, typically, a key of 1024-bit or 2048-bits is used. 1024-bit key size is no longer considered secure and at least 2048 bit is recommended to be used in practice.

A public key is the public part of the private-public key pair. A public key is available publicly and published by the private key owner. Anyone who would then like to send the publisher of the public key an encrypted message.

The Lisa creates a message such as "Here is the link to my ebook which you had requested", signs it with Bob's public key that she has received in Bob's request message and also encrypts the message using some secret key which is shared between the two during HTTPS handshake.

Now, Lisa is sure that only Bob can decode the message using the private key that is held by Bob alone. Also, somebody intercepting the message would not be able to recover its contents because the contents are encrypted by a secret key held only by Bob and Alice. This guarantees to Lisa that access to her ebook is granted only to Bob.

Having seen both the features, Authentication and Message Privacy, implied by PKI, let us move ahead to see how Bitcoin makes use of PKI to secure the public ledger .

Public And Private Keys:

Bitcoin, as well as all other major cryptocurrencies that came after it, is built upon public-key cryptography, a cryptographic system that uses pairs of keys: public keys, which are publicly known and essential for identification, and private keys, which are kept secret and are used for authentication and encryption.

Major cryptocurrencies like Bitcoin, Ethereum, and Bitcoin Cash function using three fundamental pieces of information: the address, associated with a balance and used for sending and receiving funds, and the address' corresponding public and private keys. The generation of a bitcoin address begins with the generation of a private key. From there, its corresponding public key can be derived using a known algorithm. The address, which can then be used in transactions, is a shorter, representative form of the public key.

The private key is what grants a cryptocurrency user ownership of the funds on a given address. The Blockchain wallet automatically generates and stores private keys for you. When you send from a Blockchain wallet, the software signs the transaction with your private key (without actually disclosing it), which indicates to the entire network that you have the authority to transfer the funds on the address you're sending from.

The security of this system comes from the one-way street that is getting from the private key to the public address. It is not possible to derive the public key from the address; likewise, it is impossible to derive the private key from the public key.

In the Blockchain.com Wallet, your 12-word Secret Private Key Recovery Phrase is a seed of all the private keys of all the addresses generated within the wallet. This is what allows you to restore access to your funds even if you lose access to your original wallet. Using the recovery phrase will allow you to recover your crypto.

Bitcoin improvement proposals (BIPs):

A Bitcoin Improvement Proposal (BIP) is a design document for introducing features or information to Bitcoin. This is the standard way of communicating ideas since Bitcoin has no formal structure.

The first BIP (BIP 0001) was submitted by Amir Taaki on 2011-08-19 and described what a BIP is.

Types

There are three types of BIPs:

- Standards Track BIPs - Changes to the network protocol, block or transaction validation, or anything affecting interoperability.
- Informational BIPs - Design issues, general guidelines. This type of BIP is NOT for proposing new features and do not represent community consensus
- Process BIPs - Describes or proposes a change in process. Similar to Standards BIPs but apply outside the Bitcoin protocol.

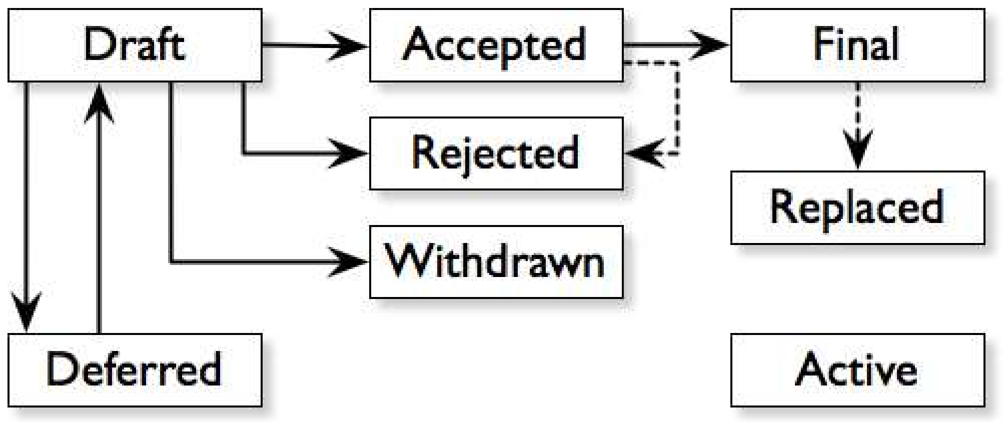
Layers

BIP 0123 established four layers for Standards BIPs:

1. Consensus
2. Peer Services
3. API/RPC
4. Applications

Workflow:

As described in BIP 0001 the workflow of a BIP is as follows:



ConsensusAlgorithms:

We know that Blockchain is a distributed decentralized network that provides immutability, privacy, security, and transparency. There is no central authority present to validate and verify the transactions, yet every transaction in the Blockchain is considered to be completely secured and verified. This is possible only because of the presence of the consensus protocol which is a core part of any Blockchain network.

A consensus algorithm is a procedure through which all the peers of the Blockchain network reach a common agreement about the present state of the distributed ledger. In this way, consensus algorithms achieve reliability in the Blockchain network and establish trust between unknown peers in a distributed computing environment. Essentially, the consensus protocol makes sure that every new block that is added to the Blockchain is the one and only version of the truth that is agreed upon by all the nodes in the Blockchain.

The Blockchain consensus protocol consists of some specific objectives such as coming to an agreement, collaboration, co-operation, equal rights to every node, and mandatory participation of each node in the consensus process. Thus, a consensus algorithm aims at finding a common agreement that is a win for the entire network.

Now, we will discuss various consensus algorithms and how they work.

ProofofWork(PoW):

This consensus algorithm is used to select a miner for the next block generation. Bitcoin uses this PoW consensus algorithm. The central idea behind this algorithm is to solve a complex mathematical puzzle and easily give out a solution. This mathematical puzzle requires a lot of computational power and thus, the node who solves the puzzle as soon as possible gets to mine the next block. For more details on PoW, please read Proof of Work (PoW) Consensus

PracticalByzantineFaultTolerance(PBFT):
Please refer to the existing article on practical Byzantine Fault Tolerance(pBFT).

ProofofStake(PoS):
This is the most common alternative to PoW. Ethereum has shifted from PoW to PoS consensus. In this type of consensus algorithm, instead of investing in expensive hardware to solve a complex puzzle, validators invest in the coins of the system by locking up some of their coins as stake. After that, all the validators will start validating the blocks. Validators will validate blocks by placing a bet on it if they discover a block which they think can be added to the chain. Based on the actual blocks added in the Blockchain, all the validators get a reward proportionate to their bets and their stake increase accordingly.
In the end, a validator is chosen to generate a new block based on their economic stake in the network. Thus, PoS encourages validators through an incentive mechanism to reach to an agreement.

ProofofBurn(PoB):
With PoB, instead of investing into expensive hardware equipment, validators 'burn' coins by sending them to an address from where they are irretrievable. By committing the coins to an unreachable address, validators earn a privilege to mine on the system based on a random selection process. Thus, burning coins here means that validators have a long-term commitment in exchange for their short-term loss.

Depending on how the PoB is implemented, miners may burn the native currency of the Blockchain application or the currency of an alternative chain, such as bitcoin. While PoB is an interesting alternative to PoW, the protocol still wastes resources needlessly. And it is also questioned that mining power simply goes to those who are willing to burn more money.

ProofofCapacity:

In the Proof of Capacity consensus, validators are supposed to invest their hard drive space instead of investing in expensive hardware or burning coins. The more hard drive space validators have, the better are their chances of getting selected for mining the next block and earning the block reward.

ProofofElapsedTime:

PoET is one of the fairest consensus algorithms which chooses the next block using fair means only. It is widely used in permissionned Blockchain networks. In this algorithm, every validator on the network gets a fair chance to create their own block. All the nodes do so by waiting for random amount of time, adding a proof of their wait in the block. The created blocks are broadcasted to the network for others consideration. The winner is the validator which has least timer value in the proof part. The block from the winning validator node gets appended to the Blockchain. There are additional checks in the algorithm to stop nodes from always winning the election, stop nodes from generating a lowest timer value.

There also exist other consensus algorithms like Proof of Activity, Proof of Weight, Proof of Importance, Leased Proof of Stake, etc. It is therefore important to wisely choose one as per the business network requirement because Blockchain networks cannot function properly without the consensus algorithms to verify each and every transaction that is being commited.

UNIT – III

INTRODUCTION TO BITCOIN:

Bitcoin is the first application of the blockchain technology.Bitcoin has started a revolution with the introduction of the very first fully decentralized digital currency, and one that has proven to be extremely secure and stable. This has also sparked a great interest in academic and industrial research and introduced many new research areas.

Since its introduction in 2008, bitcoin has gained much popularity and is currently the most successful digital currency in the world with billions of dollars invested in it. It is built on decades of research in the field of cryptography, digital cash, and distributed computing. In the following section, a brief history is presented in order to provide the background required to understand the foundations behind the invention of bitcoin.

Bitcoin: A Peer-to-Peer Electronic Cash System was written by Satoshi Nakamoto. The first key idea introduce was that purely peer-to-peer electronic cash that does need an intermediary bank to transfer payments between peers.
Bitcoin is built on decades of cryptographic research such as the research in Merkle trees, hash functions, public key cryptography, and digital signatures. Moreover, ideas such as BitGold, b-money, hashcash, and cryptographic time stamping provided the foundations for bitcoin invention. All these technologies are cleverly combined in bitcoin to create the world's first decentralized currency.

Bitcoin can be defined in various ways: it's a protocol, a digital currency, and a platform. It is a combination of peer-to-peer network, protocols, and software that facilitate the creation and usage of the digital currency named bitcoin. Note that Bitcoin with a capital B is used to refer to the Bitcoin protocol, whereas bitcoin with a lowercase b is used to refer to bitcoin, the currency. Nodes in this peer-to-peer network talk to each other using the Bitcoin protocol.

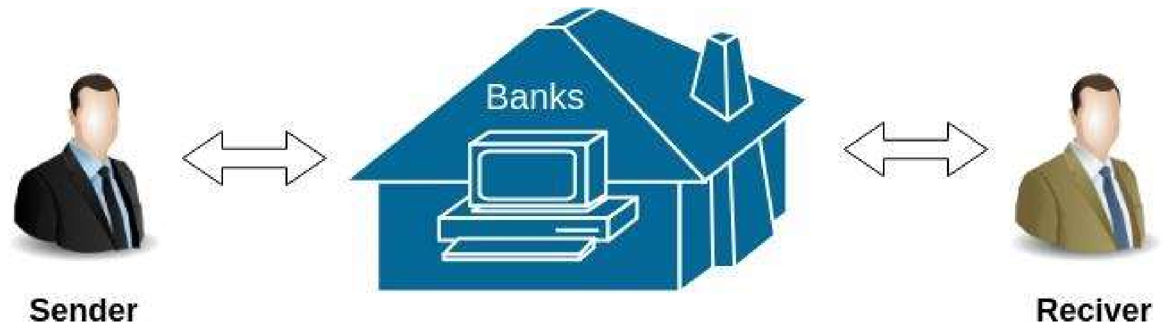
Decentralization of currency was made possible for the first time with the invention of bitcoin. Moreover, the double spending problem was solved in an elegant and ingenious way in bitcoin. Double spending problem arises when, for example, a user sends coins to two different users at the same time and they are verified independently as valid transactions.

Bitcoin Working Mechanism:

When you send an email to another person, you just type an email address and can communicate directly to that person. It is the same thing when you send an instant message. This type of communication between two parties is commonly known as Peer-to-Peer communication.

Whenever you want to transfer money to someone over the internet, you need to use a service of third-party such as banks, a credit card, a PayPal, or some other type of money transfer services. The reason for using third-party is to ensure that you are transferring that money. In other words, you need to be able to verify that both parties have done what they need to do in real exchange.

For example, Suppose you click on a photo that you want to send it to another person, so you can simply attach that photo to an email, type the receiver email address and send it. The other person will receive the photo, and you think it would end, but it is not. Now, we have two copies of photo, one is a simple email, and another is an original file which is still on my computer. Here, we send the copy of the file of the photo, not the original file. This issue is commonly known as the double-spend problem.



The double-spend problem provides a challenge to determine whether a transaction is real or not. How you can send a bitcoin to someone over the internet without needing a bank or some other institution to certify the transfer took place. The answer arises in a global network of thousands of computers called a Bitcoin Network and a special type of decentralized laser technology called **blockchain**.

Transactions & Structure:

Transactions are at the core of the bitcoin ecosystem. Transactions can be as simple as just sending some bitcoins to a bitcoin address, or it can be quite complex depending on the requirements. Each transaction is composed of at least one input and output.

Inputs can be thought of as coins being spent that have been created in a previous transaction and outputs as coins being created. If a transaction is minting new coins, then there is no input and therefore no signature is needed. If a transaction is to sends coins to some other user (a bitcoin address), then it needs to be signed by the sender with their private key and a reference is also required to the previous transaction in order to show the origin of the coins. Coins are, in fact, unspent transaction outputs represented in Satoshis.

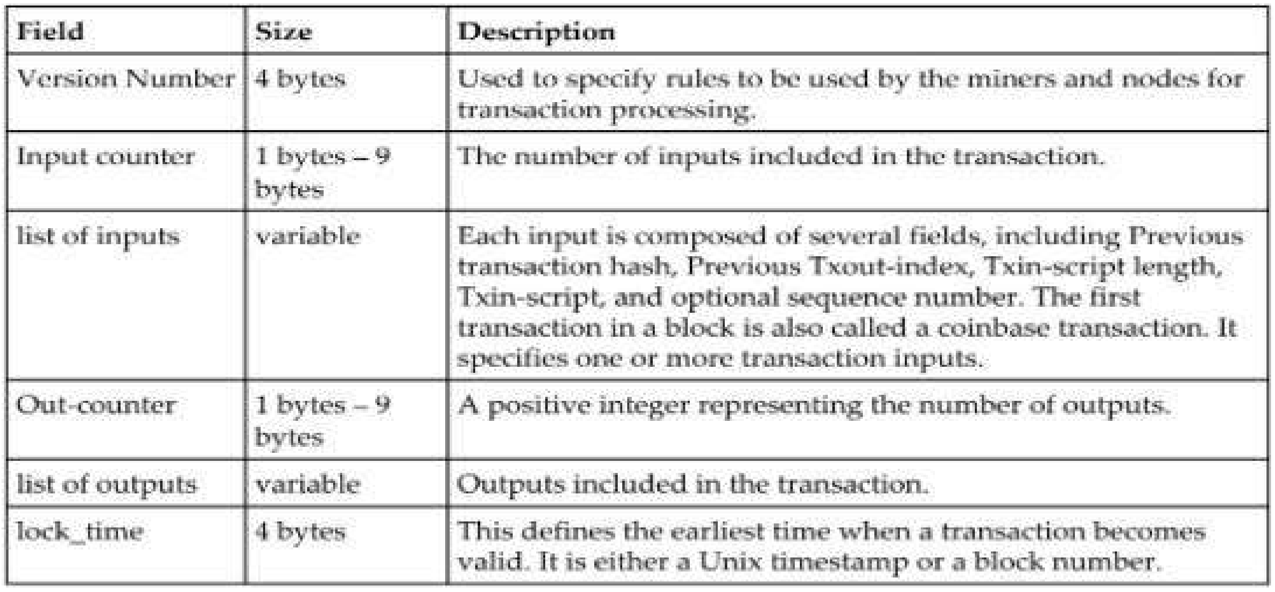
Transactions are not encrypted and are publicly visible in the blockchain. Blocks are made up of transactions and these can be viewed using any online blockchain explorer.

The transaction life cycle

1. A user/sender sends a transaction using wallet software or some other interface.
2. The wallet software signs the transaction using the sender's private key.
3. The transaction is broadcasted to the Bitcoin network using a flooding algorithm.
4. Mining nodes include this transaction in the next block to be mined.
5. Mining starts once a miner who solves the Proof of Work problem broadcasts the newly mined block to the network.
6. The nodes verify the block and propagate the block further, and confirmation starts to generate.
7. Finally, the confirmations start to appear in the receiver's wallet and after approximately six confirmations, the transaction is considered finalized and confirmed. However, six is just a recommended number, the transaction can be considered final even after the first confirmation. The key idea behind waiting for six confirmations is that the probability of double spending is virtually eliminated after six confirmations.

TRANSACTION STRUCTURE

A transaction at a high level contains metadata, inputs, and outputs. Transactions are combined to create a block. The transaction structure is shown in the following table:



Field	Size	Description
Version Number	4 bytes	Used to specify rules to be used by the miners and nodes for transaction processing.
Input counter	1 bytes – 9 bytes	The number of inputs included in the transaction.
list of inputs	variable	Each input is composed of several fields, including Previous transaction hash, Previous Txout-index, Txin-script length, Txin-script, and optional sequence number. The first transaction in a block is also called a coinbase transaction. It specifies one or more transaction inputs.
Out-counter	1 bytes – 9 bytes	A positive integer representing the number of outputs.
list of outputs	variable	Outputs included in the transaction.
lock_time	4 bytes	This defines the earliest time when a transaction becomes valid. It is either a Unix timestamp or a block number.

MetaData: This part of the transaction contains some values such as the size of the transaction, the number of inputs and outputs, the hash of the transaction, and a lock_time field. Every transaction has a prefix specifying the version number.

Inputs: Generally, each input spends a previous output. Each output is considered an Unspent Transaction Output (UTXO) until an input consumes it.

Outputs: Outputs have only two fields, and they contain instructions for the sending of bitcoins. The first field contains the amount of Satoshis, whereas the second field is a locking script that contains the conditions that need to be met in order for the output to be spent.

Verification: Verification is performed using bitcoin's scripting language.

Types of transaction:

There are various scripts available in bitcoin to handle the value transfer from the source to the destination. These scripts range from very simple to quite complex depending upon the requirements of the transaction. Standard transactions are evaluated using IsStandard() and IsStandardTx() tests and only standard transactions that pass the test are generally allowed to be mined or broadcasted on the bitcoin network. However, nonstandard transactions are valid and allowed on the network.

Pay to Public Key Hash (P2PKH):

P2PKH is the most commonly used transaction type and is used to send transactions to the bitcoin addresses. The format of the transaction is shown as folows:

ScriptPubKey: OP_DUP OP_HASH160 OP_EQUALVERIFY OP_CHECKSIG ScriptSig: The ScriptPubKey and ScriptSig parameters are concatenated together and executed.

Pay to Script Hash (P2SH):

P2SH is used in order to send transactions to a script hash (that is, the addresses starting with 3) and was standardized in BIP16. In addition to passing the script, the redeem script is also evaluated and must be valid. The template is shown as follows:

ScriptPubKey: OP_HASH160 OP_EQUAL

ScriptSig: [...]

MultiSig (Pay to MultiSig): M of n multisignature transaction script is a complex type of script where it is possible to construct a script that required multiple signatures to be valid in order to redeem a transaction. Various complex transactions such as escrow and deposits can be built using this script. The template is shown here:

ScriptPubKey: [. . .] OP_CHECKMULTISIG

ScriptSig: 0 [. . .] Raw multisig is obsolete, and multisig is usually part of the P2SH redeem script, mentioned in the previous bullet point.

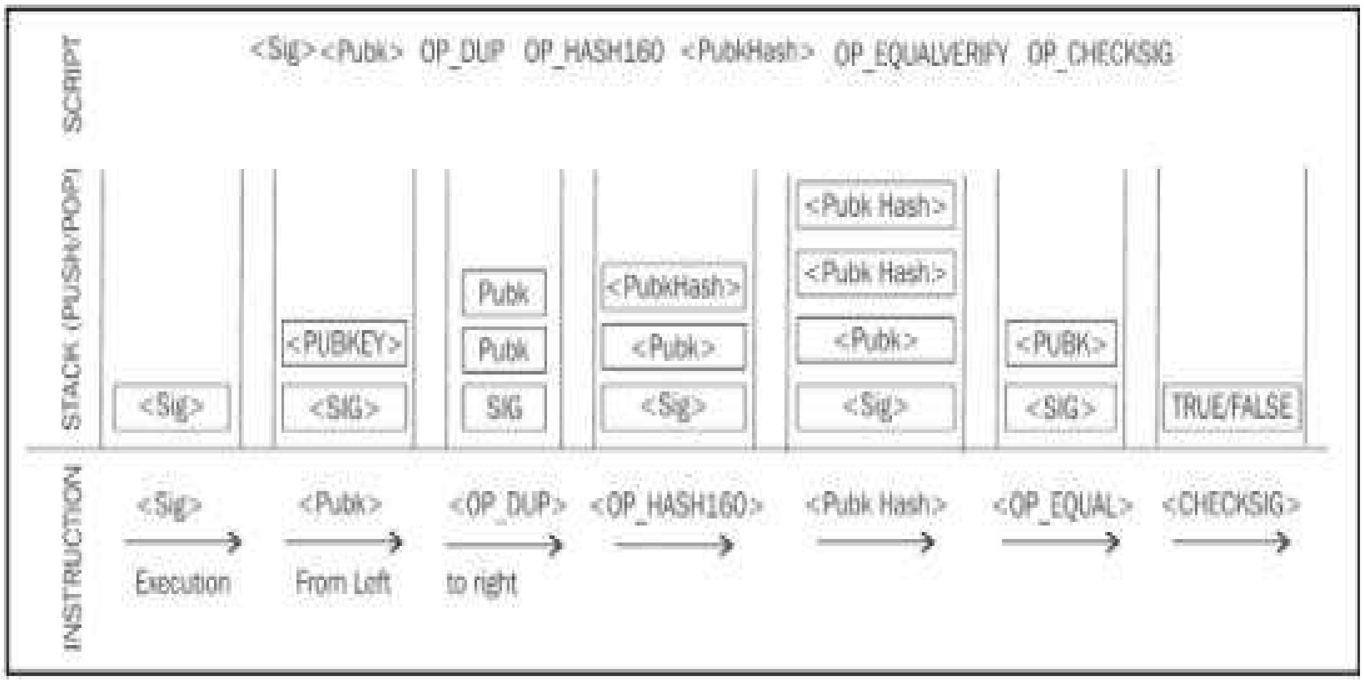
Pay to Pubkey: This script is a very simple script that is commonly used in coinbase transactions. It is now obsolete and was used in an old version of bitcoin. The public key is stored within the script in this case, and the unlocking script is required to sign the transaction with the private key. The template is shown as follows:

OP_CHECKSIG Null data/OP_RETURN: This script is used to store arbitrary data on the blockchain for a fee. The limit of the message is 40 bytes. The output of this script is unredeemable because OP_RETURN will fail the validation in any case. ScriptSig is not required in this case.

The template is very simple and is shown as follows:

OP_RETURN<data>

A P2PKH script execution is shown as follows:



P2PKH script execution:

All transactions are eventually encoded into the hex before transmitting over the bitcoin network.

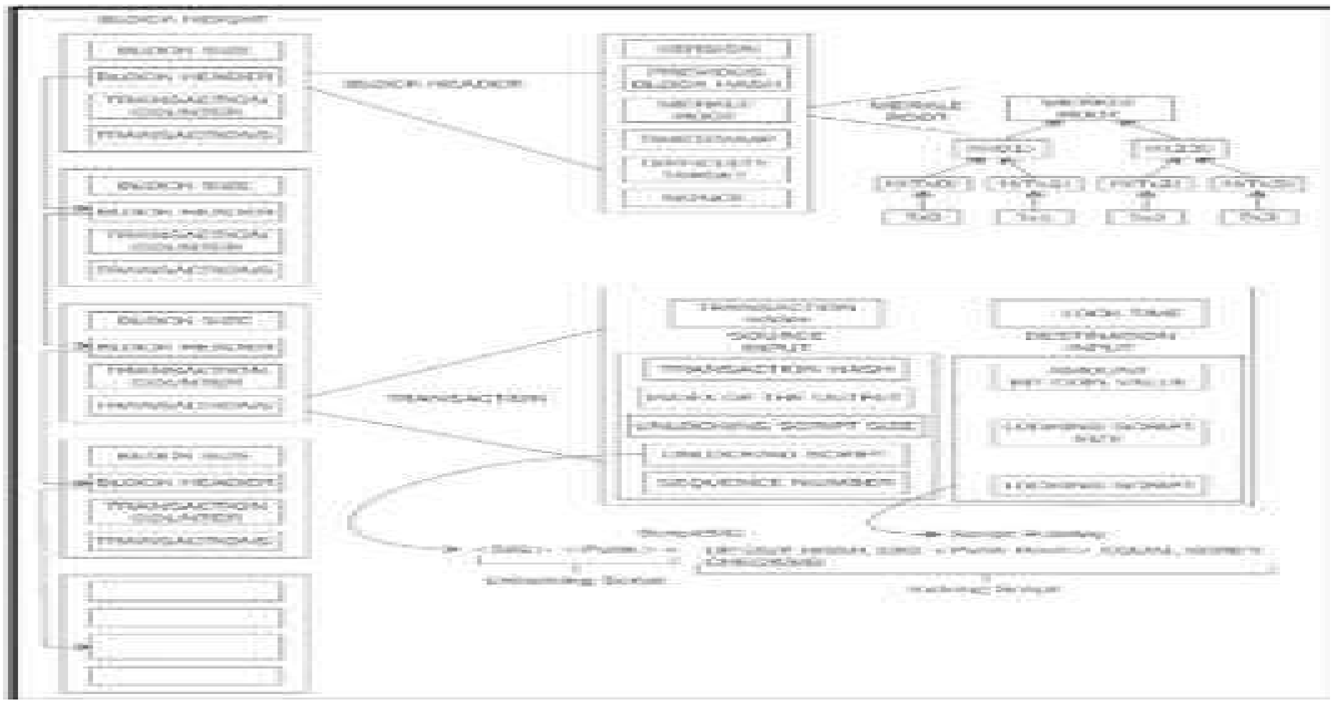
Blockchain is a public ledger of a timestamped, ordered, and immutable list of all transactions on the bitcoin network. Each block is identified by a hash in the chain and is linked to its previous block by referencing the previous block's hash. In the following structure of a block, a block header is described, followed by a detailed diagram that provides an insight into the blockchain structure.

The structure of a block

Bytes	**Name**	**Description**
80	Block header	This includes fields from the block header described in the next section.
variable	Transaction counter	The field contains the total number of transactions in the block, including the coinbase transaction.
variable	Transactions	All transactions in the block.

The structure of a block header

Bytes	**Name**	**Description**
4	Version	The block version number that dictates the block validation rules to follow.
32	previous block header hash	This is a double SHA256 hash of the previous block's header.
32	merkle root hash	This is a double SHA256 hash of the merkle tree of all transactions included in the block.
4	Timestamp	This field contains the approximate creation time of the block in the Unix epoch time format. More precisely, this is the time when the miner has started hashing the header (the time from the miner's point of view).
4	Difficulty target	This is the difficulty target of the block.
4	Nonce	This is an arbitrary number that miners change repeatedly in order to produce a hash that fulfills the difficulty target threshold.



A visualization of blockchain, block, block header, transaction and script.

As shown in the preceding diagram, blockchain is a chain of blocks where each block is linked to its previous block by referencing the previous block header's hash. This linking makes sure that no transaction can be modified unless the block that records it and all blocks that follow it are also modified. The first block is not linked to any previous block and is known as the genesis block.

The Genesis Block:

This is the first block in the bitcoin blockchain. The genesis block was hardcoded in the bitcoin core software.

Bitcoin provides protection against double spending by enforcing strict rules on transaction verification and via mining. Blocks are added in the blockchain only after strict rule checking and successful Proof of Work solution. Block height is the number of blocks before a particular block in the blockchain. The current height (at the time of writing this) of the blockchain is 434755 blocks. Proof of Work is used to secure the blockchain.

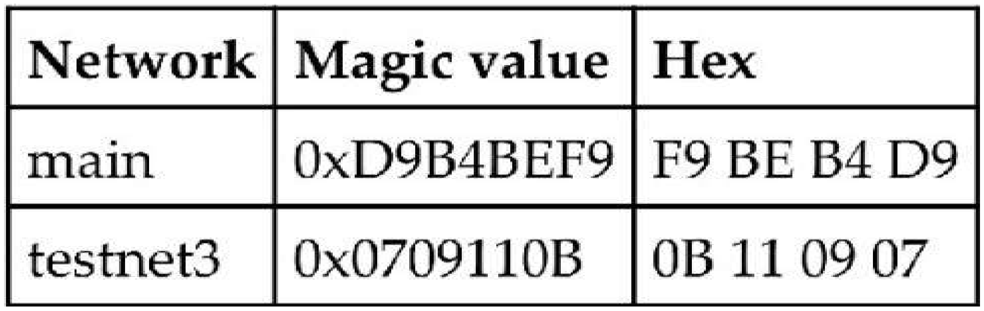
Each block contains one or more transactions, out of which the first transaction is a coinbase transaction. There is a special condition for coinbase transactions that prevent them to be spent until at least 100 blocks in order to avoid a situation where the block may be declared stale later on.

Stale blocks are created when a block is solved and every other miner who is still working to find a solution to the hash puzzle is working on that block. Mining and hash puzzles will be discussed later in the chapter in detail. As the block is no longer required to be worked on, this is considered a stale block.
Orphan Blocks : are also called detached blocks and were accepted at one point in time by the network as valid blocks but were rejected when a proven longer chain was created that did not include this initially accepted block. They are not part of the main chain and can occur at times when two miners manage to produce the blocks at the same time.

The Bitcoin Network :

The bitcoin network is a P2P network where nodes exchange transactions and blocks. There are different types of nodes on the network. There are two main types of nodes, full nodes and SPV nodes. Full nodes, as the name implies, are implementations of bitcoin core clients performing the wallet, miner, full blockchain storage, and network routing functions. However, it is not necessary to perform all these functions. SPV nodes or lightweight clients perform only wallet and network routing functionality. The latest version of Bitcoin protocol is 70014 and was introduced with bitcoin core client 0.13.0.

Bitcoin network is identified by its different magic values. A list is shown as follows:



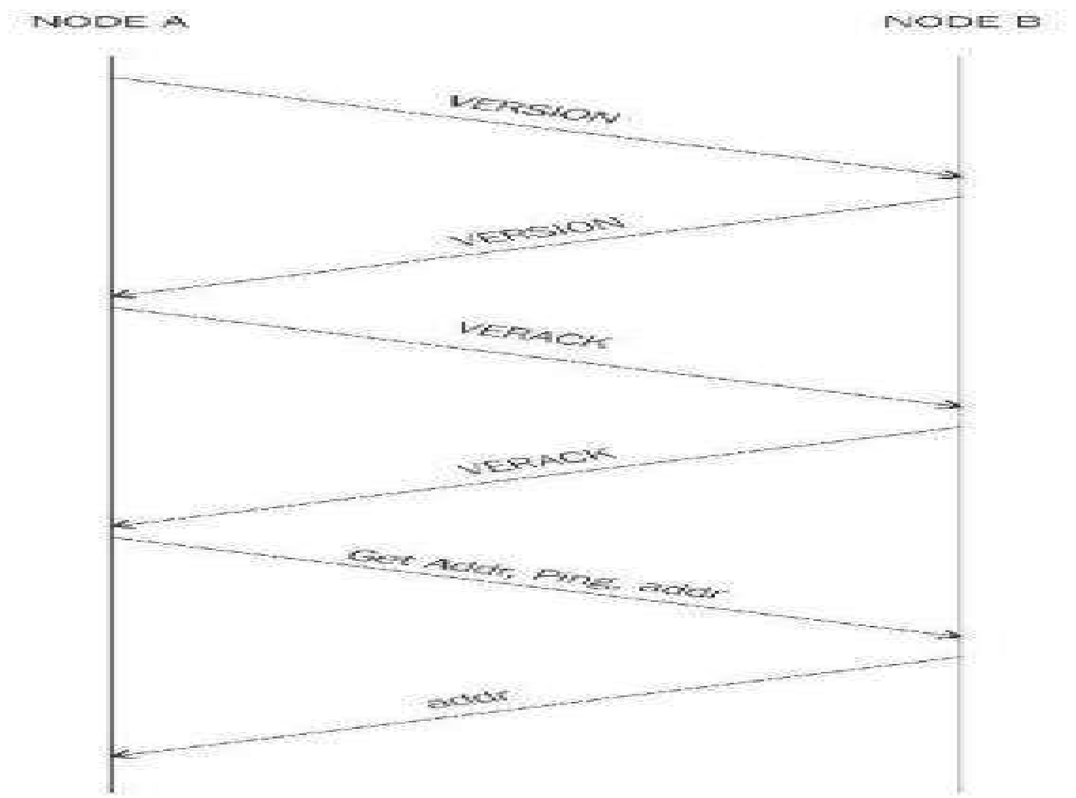
Network	Magic value	Hex
main	0xD9B4BEF9	F9 BE B4 D9
testnet3	0x0709110B	0B 11 09 07

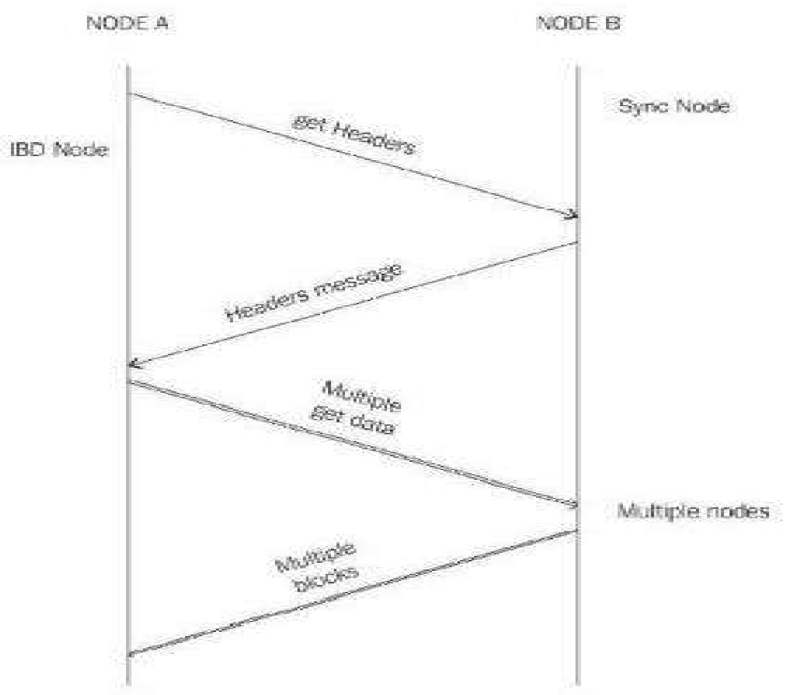
A full node performs four functions: wallet, miner, blockchain, and the network routing node.
When a bitcoin core node starts up, first, it initiates the discovery of all peers. This is achieved by querying DNS seeds that are hardcoded into the bitcoin core client and are maintained by bitcoin community members. This lookup returns a number of DNS A records. The bitcoin protocol works on TCP port 8333 by default for the main network and TCP 18333 for testnet.

First, the client sends a protocol message Version that contains various fields, such as version, services, timestamp, network address, nonce, and some other fields. The remote node responds with its own version message followed by verack message exchange between both nodes, indicating that the connection has been established.

After this, Getaddr and addr messages are exchanged to find the peers that the client do not know. Meanwhile, either of the nodes can send a ping message to see whether the connection is still live. Now the block download can begin.

If the node already has all blocks fully synchronized, then it listens for new blocks using the Inv protocol message; otherwise, it first checks whether it has a response to inv messages and have inventories already. If yes, then it requests the blocks using the Getdata protocol message; if not, then it requests inventories using the GetBlocks message. This method was used until version 0.9.3





Wallets: The wallet software is used to store private or public keys and bitcoin address. It performs various functions, such as receiving and sending bitcoins. Nowadays, software usually offers both functionalities: bitcoin client and wallet. On the disk, the bitcoin core client wallets are stored as the Berkeley DB file:

```
:~/.bitcoin$ file wallet.dat
```

wallet.dat: Berkeley DB (Btree, version 9, native byte-order) Private keys can be generated in different ways and are used by different types of wallets.

Wallets do not store any coins, and there is no concept of wallets storing balance or coins for a user. In fact, in the bitcoin network, coins do not exist; instead, only transaction information is stored on the blockchain (more precisely, UTXO unspent outputs), which are then used to calculate the amount of bitcoins.

WALLET TYPES In bitcoin, there are different types of wallets that can be used to store private keys. As a software program, they also provide some functions to the users to manage and carry out transactions on the bitcoin network.

Non-deterministic wallets :

These wallets contain randomly generated private keys and are also called Just a Bunch of Key wallets. The bitcoin core client generates some keys when first started and generates keys as and when required. Managing a large number of keys is very difficult and an error-prone process can lead to theft and loss of coins. Moreover, there is a need to create regular backups of the keys and protect them appropriately in order to prevent theft or loss.

Deterministic wallets:

In this type of wallet, keys are derived out of a seed value via hash functions. This seed number is generated randomly and is commonly represented by humanreadable mnemonic code words. Mnemonic code words are defined in BIP39. This phrase can be used to recover all keys and makes private key management comparatively easier.

Hierarchical deterministic wallets :

Defined in BIP32 and BIP44, HD wallets store keys in a tree structure derived from a seed. The seed generates the parent key (master key), which is used to generate child keys and, subsequently, grandchild keys. Key generation in HD wallets does not generate keys directly; instead, it produces some information (private key generation information) that can be used to generate a sequence of private keys. The complete hierarchy of private keys in an HD wallet is easily recoverable if the master private key is known. It is because of this property that HD wallets are very easy to maintain and are highly portable.

Brain wallets:

The master private key can also be derived from the hash of passwords that are memorized. The key idea is that this passphrase is used to derive the private key and if used in HD wallets, this can result in a full HD wallet that is derived from a single memorized password. This is known as brain wallet. This method is prone to password guessing and brute force attacks but techniques such as key stretching can be used to slow down the progress made by the attacker. Paper wallets As the name implies, this is a paper-based wallet with the required key material printed on it. It requires physical security to be stored. Paper wallets can be generated online from various service providers, such as https://bitcoinpaperwallet.com/ or https://www.bitaddress.org/.

Hardware wallets:

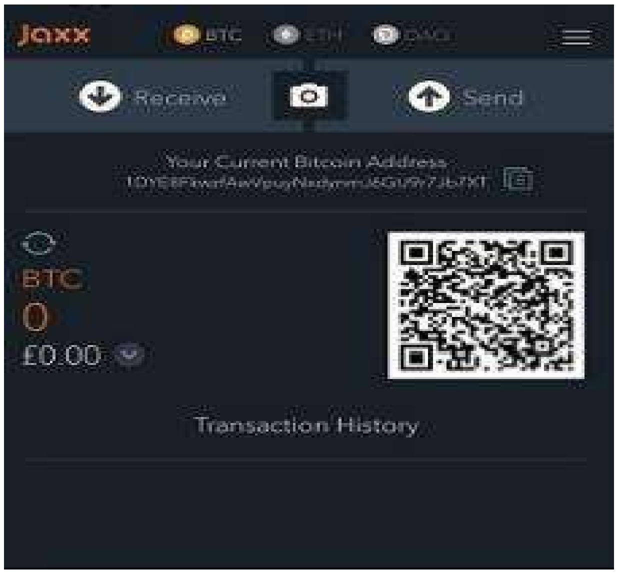
Another method is to use a tamper-resistant device to store keys. This tamper-resistant device can be custombuilt or with the advent of NFC-enabled phones, this can also be a secure element (SE) in NFC phones. Trezor and Ledger wallets (various types) are the most commonly used bitcoin hardware wallets.

Online wallets :

Online wallets, as the name implies, are stored entirely online and are provided as a service usually via cloud. They provide a web interface to the users to manage their wallets and perform various functions such as making and receiving payments. They are easy to use but imply that the user trust the online wallet service provider.

Mobile wallets :

Mobile wallets, as the name suggests, are installed on mobile devices. They can provide various methods to make payments, most notably the ability to use smart phone cameras to scan QR codes quickly and make payments. Mobile wallets are available for the Android platform and iOS, for example, breadwallet, copay, and Jaxx.



Jaxx Mobile wallet

Bitcoin payments:

Bitcoins can be accepted as payments using various techniques. Bitcoin is not recognized as a legal currency in many jurisdictions, but it is increasingly being accepted as a payment method by many online merchants and e-commerce websites. There are a numbers of ways in which buyers can pay the business that accepts bitcoins. For example, in an online shop, bitcoin merchant solutions can be used, whereas in traditional physical shops, point of sale terminals and other specialized hardware can be used.

Customers can simply scan the QR barcode with the seller's payment URI in it and pay using their mobile devices. Bitcoin URIs allow users to make payments by simply clicking on links or scanning QR codes. URI (Uniform Resource Idenfier) is basically a string that represents the transaction information. It is defined in BIP21.
The QR code can be displayed near the point of the sale terminal. Nearly all bitcoin wallets support this feature. Business can use the following screenshot to advertise that they can accept bitcoins as payment.

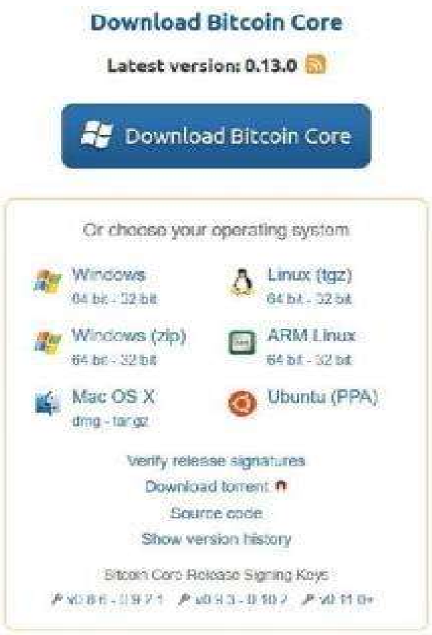
Various payment solutions, such as xbtterminal and 34 bytes bitcoin POS terminal are available commercially. 34 bytes POS solution.
Bitcoin payment processor, offered by many online service providers, allows integration with e-commerce websites.

Bitcoin investment and buying and selling bitcoins
There are many online exchanges where users can buy and sell bitcoins. This is a big business on the Internet now and it offers bitcoin trading, CFDs, spread betting, margin trading, and various other choices. Traders can buy bitcoins or trade by opening long or short positions to make profit when bitcoin's price goes up or down. Several other features, such as exchanging bitcoins for other virtual currencies, are also possible, and many

online bitcoin exchanges provide this function. Advanced market data, trading strategies, charts, and relevant data to support traders is also available. An example is shown from CEX.IO here.

Bitcoin installation

The bitcoin core client can be installed from https://bitcoin.org/en/download. This is available for different architectures and platforms ranging from x86 windows to ARM Linux, as shown in the following image:



Download Bitcoin Core

Latest version: 0.13.0

Download Bitcoin Core

Or choose your operating system

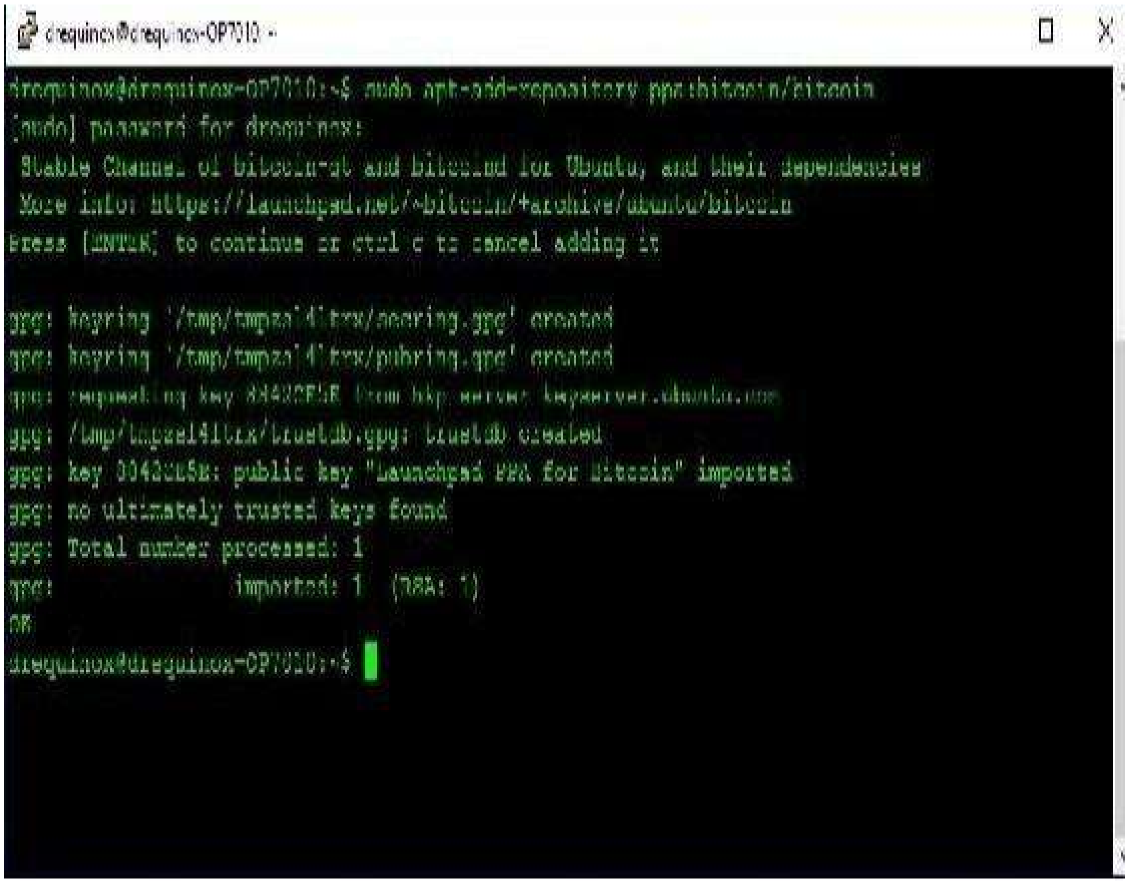
Windows 64 bit - 32 bit	Linux (tgz) 64 bit - 32 bit
Windows (zip) 64 bit - 32 bit	ARM Linux 64 bit - 32 bit
Mac OS X dmg - tar.gz	Ubuntu (PPA)

Verify release signatures
Download torrent
Source code
Show version history

Bitcoin Core Release Signing Keys
v0.8.6 - 0.9.2.1 v0.9.3 - 0.10.2 v0.11.0+

SETTING UP A BITCOIN NODE

A sample run of the bitcoin core installation on Ubuntu is shown here; for other platforms, you can get details from www.bitcoin.org.



```
drequinox@drequinox-OP7010:~$ sudo apt-add-repository ppa:bitcoin/bitcoin
[sudo] password for drequinox:
 Stable Channel of bitcoin-qt and bitcoind for Ubuntu, and their dependencies
 More info: https://launchpad.net/~bitcoin/+archive/ubuntu/bitcoin
Press [ENTER] to continue or ctrl c to cancel adding it

gpg: keyring '/tmp/tmpzai41trx/secring.gpg' created
gpg: keyring '/tmp/tmpzai41trx/pubring.gpg' created
gpg: requesting key 8842CE5E from hkp server keyserver.ubuntu.com
gpg: /tmp/tmpzai41trx/trustdb.gpg: trustdb created
gpg: key 8842CE5E: public key "Launchpad PPA for Bitcoin" imported
gpg: no ultimately trusted keys found
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
OK
drequinox@drequinox-OP7010:~$
```

Step 2:

drequinox@drequinox-OP7010:~$ sudo apt-get update

Depending on the client required, users can use either of the following commands, or they can issue both commands at once:

sudo apt-get install bitcoind

sudo apt-get install bitcoin-qt

drequinox@drequinox-OP7010:~$ sudo apt-get install bitcoin-qt bitcoind

Reading package lists... Done Building dependency tree

Reading state information... Done ......

SETTING UP THE SOURCE CODE

The bitcoin source code can be downloaded and compiled if users wish to participate in the bitcoin code or for learning purpose. Git can be used to download the bitcoin source code:

```
 $ sudo apt-get install git
 $ mkdir bcsource
$cd bcsource
drequinox@drequinox-OP7010:~/bcsource $ git clone https://github.com/bitcoin/bitcoin.git
Cloning into 'bitcoin'...
remote:
Counting objects: 78960, done.
remote: Compressing objects: 100% (3/3), done.
 remote: Total 78960 (delta 0), reused 0 (delta 0), pack-reused 78957
Receiving objects: 100% (78960/78960), 72.53 MiB | 1.85 MiB/s, done.
Resolving deltas: 100% (57908/57908), done.
Checking connectivity... done.
```

Change the directory to bitcoin:

```
drequinox@drequinox-OP7010:~/bcsource$ cd bitcoin
```

After the preceding steps are completed, the code can be compiled:

```
drequinox@drequinoxOP7010:~/bcsource/bitcoin$./autogen.sh
drequinox@drequinoxOP7010:~/bcsource/bitcoin$./configure.sh
drequinox@drequinoxOP7010:~/bcsource/bitcoin$make drequinox@drequinoxOP7010:~/bcsource/bitcoin$
sudo make install
```

SETTING UP BITCOIN.CONF

bitcoin.conf file is a configuration file that is used by the bitcoin core client to save configuration settings. All command line options for the bitcoind client with the exception of -conf switch can be set up in the configuration file, and when bitcoin-qt or bitcoind will start up, it will take the configuration information from that file. In Linux systems, this is usually found in $HOME/.bitcoin/, or it can also specified in the command line using the -conf= switch to bitcoind core client software.

STARTING UP A NODE IN TESTNET

The bitcoin node can be started in the testnet mode if you want to test the bitcoin network and run an experiment. This is a faster network as compared to the live network and has relaxed rules for mining and transactions. Various faucet services are available for the bitcoin test network. One example is Bitcoin TestNet sandbox, where users can request bitcoins to be paid to their testnet bitcoin address. This can be accessed via https://testnet.manu.backend.hamburg/. This is very useful for experimentation with transactions on test net.

The command line to start up test net is as follows:

```
bitcoind --testnet -daemon
bitcoin-cli –testnet
bitcoin-qt –testnet
```

STARTING UP A NODE IN REGTEST

The regtest mode (regression testing mode) can be used to create a local blockchain for testing purposes. The following commands can be used to start up a node in the reg test mode

```
bitcoind -regtest -daemon
Bitcoin server starting
```

Blocks can be generated using the following command:

```
bitcoin-cli -regtest
```

generate 200 Relevant log messages can be viewed in the .bitcoin/regtest directory on a Linux system under debug.log.

After block generation, the balance can be viewed as follows:

```
drequinox@drequinoxOP7010:
~/.bitcoin/regtest
$ bitcoin-cli - regtest getbalance 8750.00000000
```

The node can be stopped using this:

```
drequinox@drequinox-OP7010:~/.bitcoin$ bitcoin-cli -regtest stop
Bitcoin server stopping
```

STARTING UP A NODE IN LIVE MAINNET

Bitcoind is the core client software that can be run as a daemon, and it provides the JSON RPC interface. Bitcoin-cli is the command line feature-rich tool to interact with the daemon; the daemon then interacts with the blockchain and performs various functions. Bitcoin-cli calls only JSON-RPC functions and does not perform any actions on its own on the blockchain.

Bitcoin-qt is the bitcoin core client GUI. When the wallet software starts up first, it verifies the blocks on the disk and then starts up and shows the following GUI:

Bitcoin Core QT client, just after installation, showing that blockchain is not in sync The verification process is not specific to the Bitcoin-qt client; it is performed by the bitcoind client as well. **EXPERIMENTING WITH BITCOIN CLI**

Bitcoin-cli is the command-line interface available with the bitcoin core client and can be used to perform various functions using the RPC interface provided by the bitcoin core client. A sample run of bitcoin-cli getinfo; the same format can be used to invoke other commands A list of all commands can be shown via the following command: Testnet bitcoin-cli, this is just the first few lines of the output, actual output has many commands

HTTP REST: Starting from bitcoin core client 0.10.0, the HTTP REST interface is also available. By default, this runs on the same TCP port 8332 as JSON-RPC.

Bitcoin programming and the command-line interface

Bitcoin programming is a very rich field now. The bitcoin core client exposes various JSON RPC commands that can be used to construct raw transactions and perform other functions via custom scripts or programs. Also, the command line tool Bitcoin-cli is available, which makes use of the JSON-RPC interface and provides a rich toolset to work with Bitcoin.

These APIs are also available via many online service provider in the form of bitcoin APIs, and they provide a simple HTTP REST interface. Bitcoin APIs, such as blockchain.info and bitpay, block.io, and many others, offer a myriad of options to develop bitcoin-based solutions. Various libraries are available for bitcoin programming. A list is shown as follows, and those if you interested can further explore the libraries.

Libbitcoin: Available at https://libbitcoin.dyne.org/ and provides powerful command line utilities and clients.

Pycoin: Available at https://github.com/richardkiss/pycoin, is a library for Python.

Bitcoinj: This library is available at https://bitcoinj.github.io/ and is implemented in Java.

There are many online bitcoin APIs available,the most commonly used APIs are listed as follows:

https://bitcore.io/

https://bitcoinjs.org/

https://blockchain.info/api

All APIs offer more or less the same type of functionality, and it gets difficult to choose which API is the best.

Bitcoin improvement proposals (BIPs) :

These documents are used to propose or inform the bitcoin community about the improvements suggested, the design issues, or information about some aspects of the bitcoin ecosystem. There are three types of bitcoin improvement proposals, abbreviated as BIPs:

Standard BIP: Used to describe the major changes that have a major impact on the bitcoin system, for example, block size changes, network protocol changes, or transaction verification changes.

Process BIP: A major difference between standard and process BIPs is that standard BIPs cover protocol changes, whereas process BIPs usually deal with proposing a change in a process that is outside the core Bitcoin protocol. These are implemented only after a consensus among bitcoin users.

Informational BIP: These are usually used to just advise or record some information about the bitcoin ecosystem, such as design issues.

UNIT-4

ETHEREUM :

Ethereum is a platform powered by blockchain technology that is best known for its native cryptocurrency, called ether, or ETH, or simply ethereum. The distributed nature of blockchain technology is what makes the Ethereum platform secure, and that security enables ETH to accrue value.

Ethereum, just like any other blockchain, can be visualized as a transaction-based state machine. The idea is that a genesis state is transformed into a final state by executing transactions incrementally. The final transformation is then accepted as the absolute undisputed version of the state. In the following diagram, the Ethereum state transition function is shown, where a transaction execution has resulted in a state transition.

Elements of Ethereum Block chain:

In the following section, you will be introduced to various components of the Ethereum network and the blockchain. First, the basic concept of the EVM is givenin the next section.

Ethereum virtual machine(EVM)

EVM is a simple stack-based execution machine that runs bytecode instructions in order to transform the system state from one state to another. The word size of the virtual machine is set to 256-bit. The stack size is limited to 1024 elements and is based on the **LIFO** (**Last in First Out**) queue. EVM is a Turing-complete machine but is limited by the amount of gas that is required to runany instruction. This means that infinite loops that can result in denial of service attacks are not possible due to gas requirements.

EVM also supports exception handling in case exceptions occur, such as not having enough gas or invalid instructions, in which case the machine would immediately halt and return the error to the executing agent.EVM is a fully isolated and sandboxed runtime environment. EVM is a stack-based architecture.EVM is big-endian by design and it uses 256-bit wide words. This word size allows for Keccak 256-bit hash and elliptic curve cryptography computations.

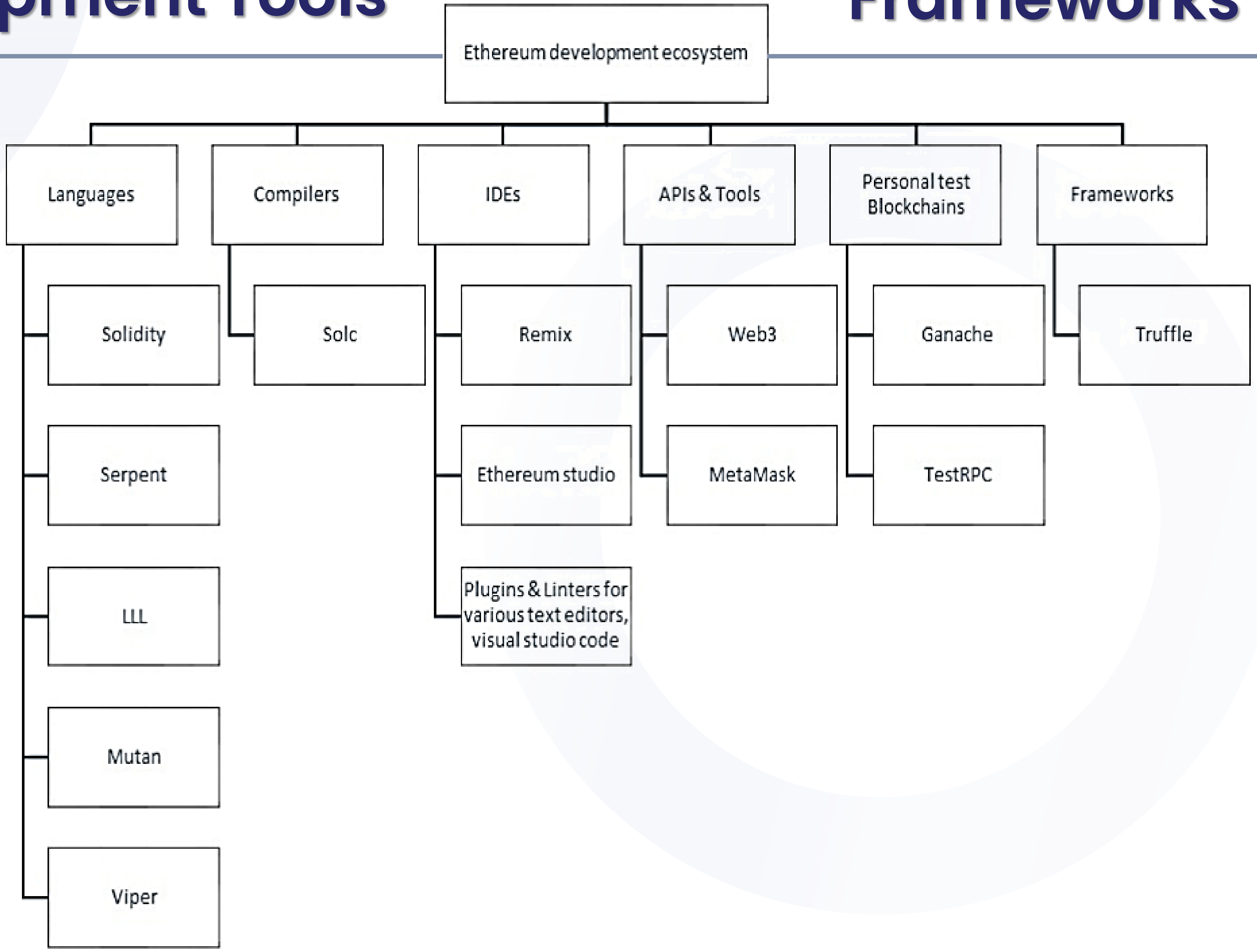
INTRODUCING

SOLIDITY

By

Ch.Aravind Kumar

Development Tools & Frameworks



Solidity language

- Solidity is a domain-specific language of choice for programming contracts in Ethereum.
- There are, however, other languages that can be used, such as Serpent, Mutan, and LLL (Low-level Lisp-like Language) but Solidity is the most popular at the time of writing this. Its syntax is closer to both JavaScript and C.
- It is a statically typed language, which means that variable type checking in Solidity is carried out at compile time.
- Solidity is also called a contract-oriented language. In Solidity, contracts are equivalent to the concept of classes in other object-oriented programming languages.

Data Types

Value types & **Reference types**

Value-type variables store their own data.

- Boolean
- Integers
- Address
- Literals
  - Integer literals
  - String literals
  - Hexadecimal literals
- Enums
- Function types
  - Internal
  - External

Reference type variables store the location of the data. They don't share the data directly. With the help of reference type, two different variables can refer to the same location where any change in one variable can affect the other one.

- Arrays
- Structs
- Data location
- Mappings

Value Types

Boolean

- This data type has two possible values, true or false

Example:

```
bool v = true;
bool v = false;
```

This statement assigns the value true to v.

Integers

- This data type represents integers. The following table shows various keywords used to declare integer data types:

Keyword	Types	Details
`int`	Signed integer	`int8` to `int256`, which means that keywords are available from `int8` up to `int256` in increments of 8, for example: `int8, int16, int24.`
`uint`	Unsigned integer	`uint8, uint16,` ... to `uint256`, unsigned integer from 8 bits to 256 bits. The usage is dependent on the requirements that how many bits are required to be stored in the variable

Example :

```
uint256 x;
uint y;
uint256 z;
uint constant z=10+10;
```

Note: These types can also be declared with the constant keyword

Address

- This data type holds a 160-bit long (20 byte) value.
- This type has several members that can be used to interact with and query the contracts.

  - **Balance:** The balance member returns the balance of the address in Wei.

  - **Send:** This member is used to send an amount of ether to an address (Ethereum's 160-bit address) and returns true or false depending on the result of the transaction

Example

```
address to = 0x6414cc08d148dce9ebf5a2d0b7c220ed2d3203da;
address from = this;
if (to.balance < 10 && from.balance > 50) to.send(20);
```

  - **Call functions:** The `call`, `callcode`, and `delegatecall` calls are provided in order to interact with functions that do not have ABI. These functions should be used with caution as they are not safe to use due to the impact on type safety and security of the contracts.

  - **Array value types (fixed size and dynamically sized byte arrays):** Solidity has fixed size and dynamically sized byte arrays. Fixed size keywords range from `bytes1` to `bytes32`, whereas dynamically sized keywords include `bytes` and `string`. The `bytes` keyword is used for raw byte data and `string` is used for strings encoded in UTF-8. As these arrays are returned by the value, calling them will incur gas cost. `length` is a member of array value types and returns the length of the byte array.

Example :

static (fixed size) array

```
bytes32[10] bankAccounts;
```

dynamically sized array

```
bytes32[] trades;
```

Get length of trades

```
trades.length;
```

Literals

- These used to represent a fixed value.

Integer literals :

- Integer literals are a sequence of decimal numbers in the range of 0-9.

Example

```
uint8 x = 2;
```

String literals :

- String literals specify a set of characters written with double or single quotes.

Example

```
'packt' "packt"
```

Hexadecimal literals :

- Hexadecimal literals are prefixed with the keyword hex and specified within double or single quotation marks.

Example

```
(hex'AABBCC');
```

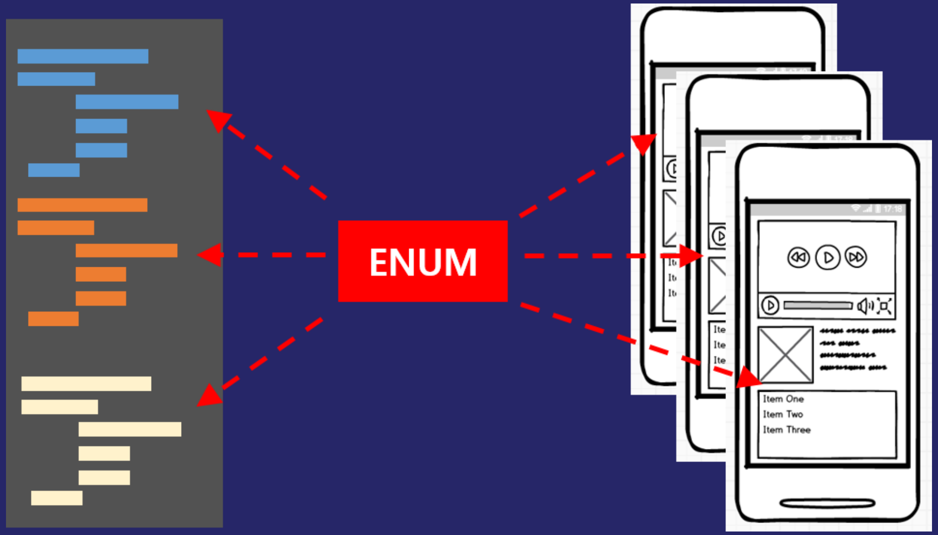
Enums

- This allows the creation of user-defined types.

Example

```
enum Order {Filled, Placed, Expired };
Order private ord;
ord=Order.Filled;
```

- Explicit conversion to and from all integer types is allowed with enums.



Reference types

Arrays

- Arrays represent a contiguous set of elements of the same size and type laid out at a memory location.
- The concept is the same as any other programming language. Arrays have two members named `length` and `push`

```
uint[] OrderIds;
```

Structs

- This data type represents integers. The following table shows various keywords used to declare integer data types:

```
pragma solidity ^0.4.0;
contract TestStruct {
        struct Trade {
                        uint tradeid;
                        uint quantity;
                        uint price;
                        string trader;
                       } //This struct can be initialized and used as below Trade
                        tStruct = Trade({tradeid:123, quantity:1, price:1,
                        trader:"equinox"});
                 }
```

Data location

- Data location specifies where a particular complex data type will be stored. Depending on the default or annotation specified, the location can be storage or memory. This is applicable to arrays and structs and can be specified using the `storage` or `memory` keywords.

- As copying between memory and storage can be quite expensive, specifying a location can be helpful to control the gas expenditure at times. **Calldata** is another memory location that is used to store function arguments.

- Parameters of external functions use **calldata** memory. By default, parameters of functions are stored in **memory**, whereas all other local variables make use of **storage**. State variables, on the other hand, are required to use storage.

Mappings

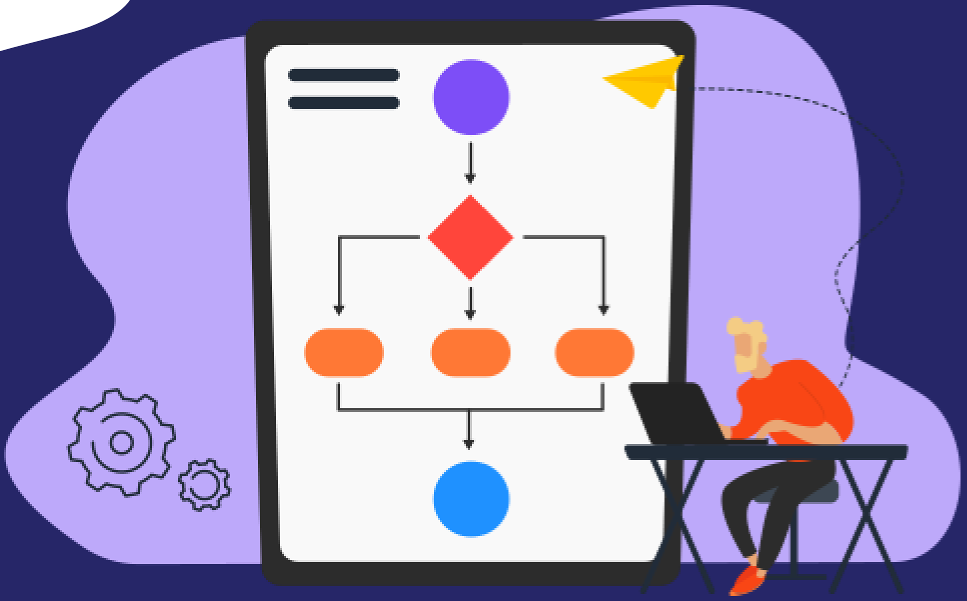
- Mappings are used for a key to value mapping. This is a way to associate a value with a key. All values in this map are already initialized with all zeroes,

```
mapping (address => uint) offers;
```

- This example shows that offers is declared as a mapping. Another example makes this clearer:

```
mapping (string => uint) bids;
bids["packt"] = 10;
```

- This is basically a dictionary or a hash table where string values are mapped to integer values. The mapping named `bids` has string `packt` mapped to value 10



Global variables

- Solidity provides a number of global variables that are always available in the global namespace.
- These variables provide information about blocks and transactions.
- Additionally, cryptographic functions and address related variables are available as well.

- A subset of available functions and variables is shown as follows:

```
keccak256(...) returns (bytes32)
```

// This function is used to compute the Keccak-256 hash of the argument provided to the function:

```
ecrecover(bytes32 hash, uint8 v, bytes32 r, bytes32 s) returns (address)
```

// This function returns the associated address of the public key from the elliptic curve signature:

```
block.number
```

// This returns the current block number.

Control structures

- Control structures available in solidity language are `if...else`, `do`, `while`, `for`, `break`, `continue`, and `return`. They work exactly the same as other languages such as C-language or JavaScript.

If…else

```
if (x == 0)
      y = 0;
else
      z = 1;
```

do

```
do{
       x++;
    }
(while z>1);
```

while

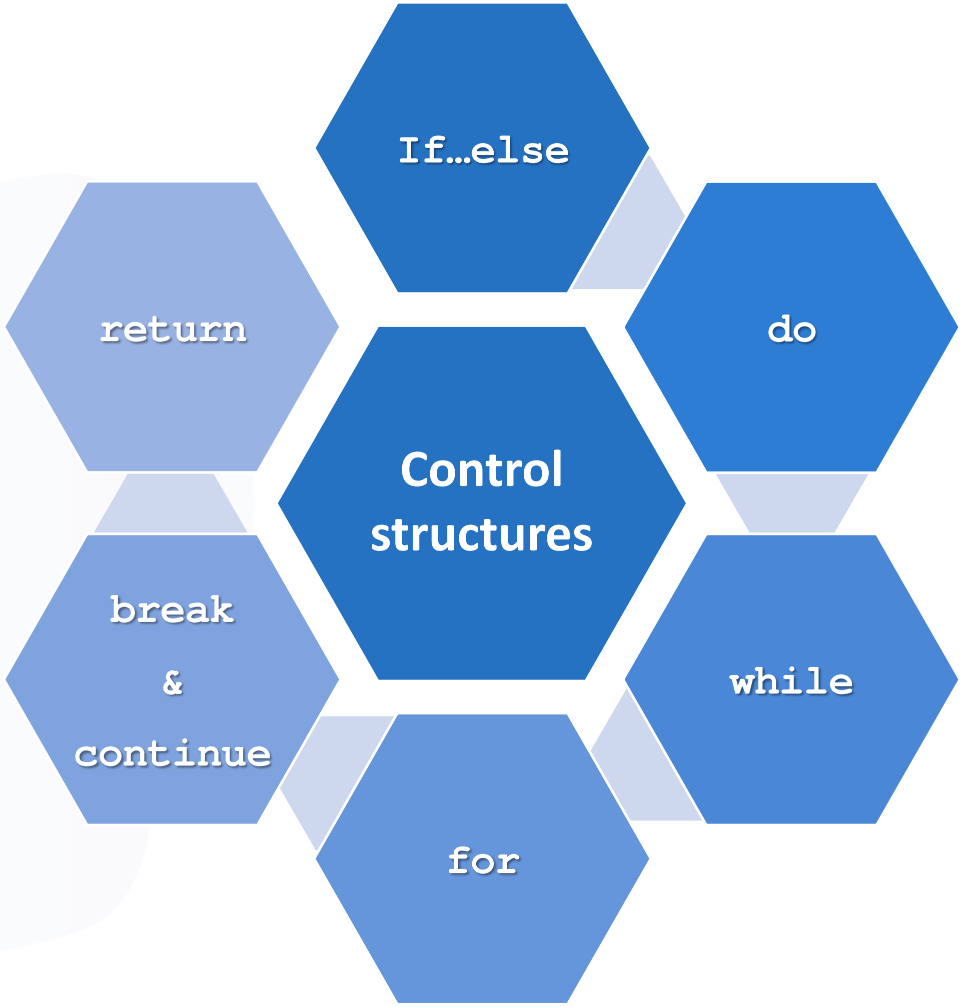
```
while(x > 0){
z++;
}
```

return

```
return 0;
```

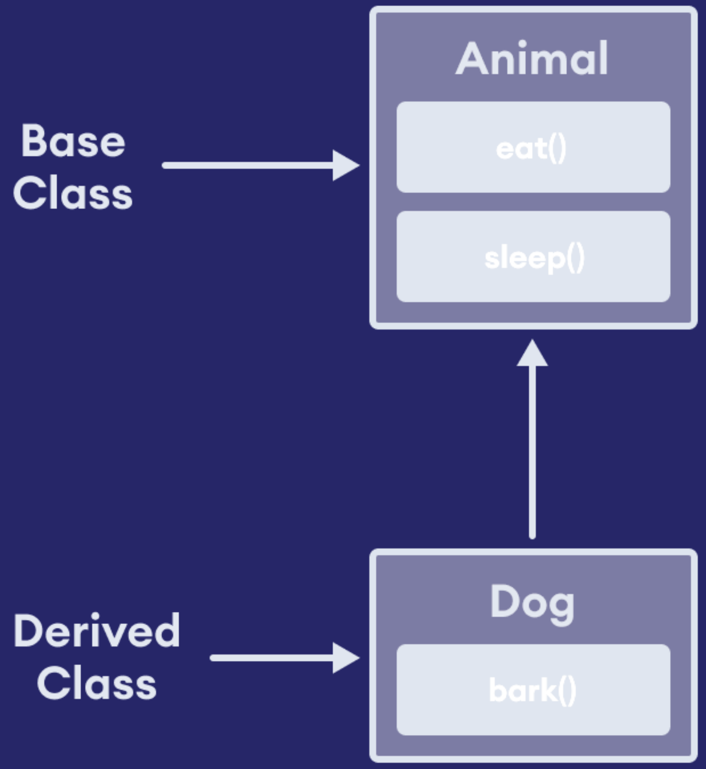
for
break
&
continue

```
for(uint8 x=0; x<=10; x++) {
         //perform some work
         z++;
         if(z == 5) break;
         }
```



Events

- Events in Solidity can be used to log certain events in EVM logs.
- These are quite useful when external interfaces are required to be notified of any change or event in the contract.
- These logs are stored on the blockchain in transaction logs. Logs cannot be accessed from the contracts but are used as a mechanism to notify change of state or the occurrence of an event (meeting a condition) in the contract.



Inheritance

- Inheritance is supported in Solidity. The is keyword is used to derive a contract from another contract. In the following example, `valueChecker2` is derived from the `valueChecker` contract. The derived contract has access to all non-private members of the parent contract

```
pragma solidity ^0.4.0;
contract valueChecker
{
          uint8 price = 20;
          event valueEvent(bool returnValue);
          function Matcher(uint8 x) public returns (bool)
          {
                    if (x>=price)
                    {
                              valueEvent(true);
                              return true;
                    }
          }
 }
contract valueChecker2 is valueChecker
{
          function Matcher2() public view returns (uint)
          {
                    return price+10;
          }
}
```

Libraries

- Libraries are deployed only once at a specific address and their code is called via `CALLCODE` or `DELEGATECALL` opcode of the EVM. The key idea behind libraries is code reusability.
- They are similar to contracts and act as base contracts to the calling contracts. A library can be declared as shown in the following example

```
library Addition
{
        function Add(uint x,uint y) returns (uint z)
        {
                return x + y;
        }
}
```

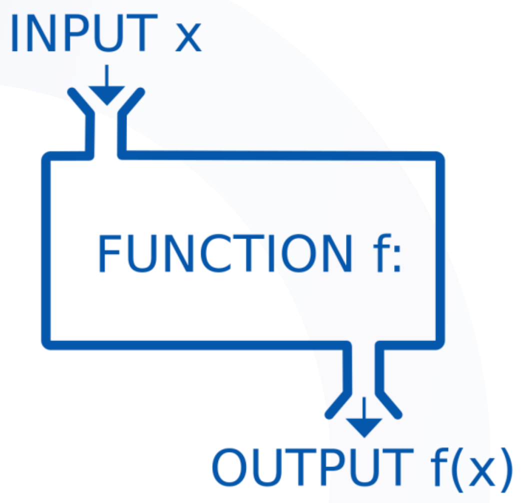
- This library can then be called in the contract, as shown here. First, it needs to be imported and then it can be used anywhere in the code.

```
import "Addition.sol"
function Addtwovalues() returns(uint)
{
        return Addition.Add(100,100);
}
```

Functions

- Functions in Solidity are modules of code that are associated with a contract. Functions are declared with a name, optional parameters, access modifier, optional constant keyword, and optional return type.

```
function orderMatcher (uint x)
private constant returns(bool return value)
```



How to define a function:

```
function <name of the function> (<parameters>) <vibility specifier> returns
(<return datatype>  <name of the variable>)
```

Input parameters of a function

- Input parameters of a function are declared in the form of <data type> <parameter name>

```
contract myContract
{
          function checkValues(uint x, uint y)
          {
          }
}
```

Output parameters of a function

- Output parameters of a function are declared in the form of <data type> <parameter name>

```
contract myContract
{
           function getValue() returns (uint z)
           {
                    z=x+y;
           }
}
```

Ch.Aravind

HYPERLEDGER

HYPERLEDGER

- Hyperledger is not a blockchain, but a project that was initiated by the Linux Foundation in December 2015 to advance blockchain technology.
- This project is a collaborative effort by its members to build an open source distributed ledger framework that can be used to develop and implement cross-industry blockchain applications and systems.
- The principal focus is to create and run platforms that support global business transactions.
- The project also focuses on improving the reliability and performance of blockchain systems.

PROJECTS UNDER HYPERLEDGER

There are two categories of projects under Hyperledger.

Blockchain projects

- Hyperledger Fabric
- Hyperledger Sawtooth Lake
- Hyperledger Iroha
- Hyperledger Burrow
- Hyperledger Indy

Relevant tools or modules that support these blockchains

- Hyperledger Cello
- Hyperledger Composer
- Hyperledger Explorer
- Hyperledger Quilt

PROJECTS

- Iroha was contributed by **Soramitsu, Hitachi, NTT Data, and Colu** in **September 2016**. Iroha is aiming to build a **library of reusable components** that users can choose to run on their own Hyperledger-based distributed ledgers.

- This project is currently in the incubation state. Hyperledger Burrow was contributed by **Monax**, who develop blockchain development and deployment platforms for business.
- Hyperledger Burrow introduces a modular blockchain platform and an **Ethereum Virtual Machine (EVM)** based smart contract execution environment.

- The Sawtooth Lake is a blockchain project proposed by **Intel in April 2016** with some key innovations focusing on the **decoupling of ledgers from transactions**, flexible usage across multiple business areas using transaction families, and pluggable consensus.

- This project is under incubation under Hyperledger. Indy is a distributed ledger developed for building a **decentralized identity**.
- It provides tools, utility libraries, and modules which can be used to build blockchain based digital identities

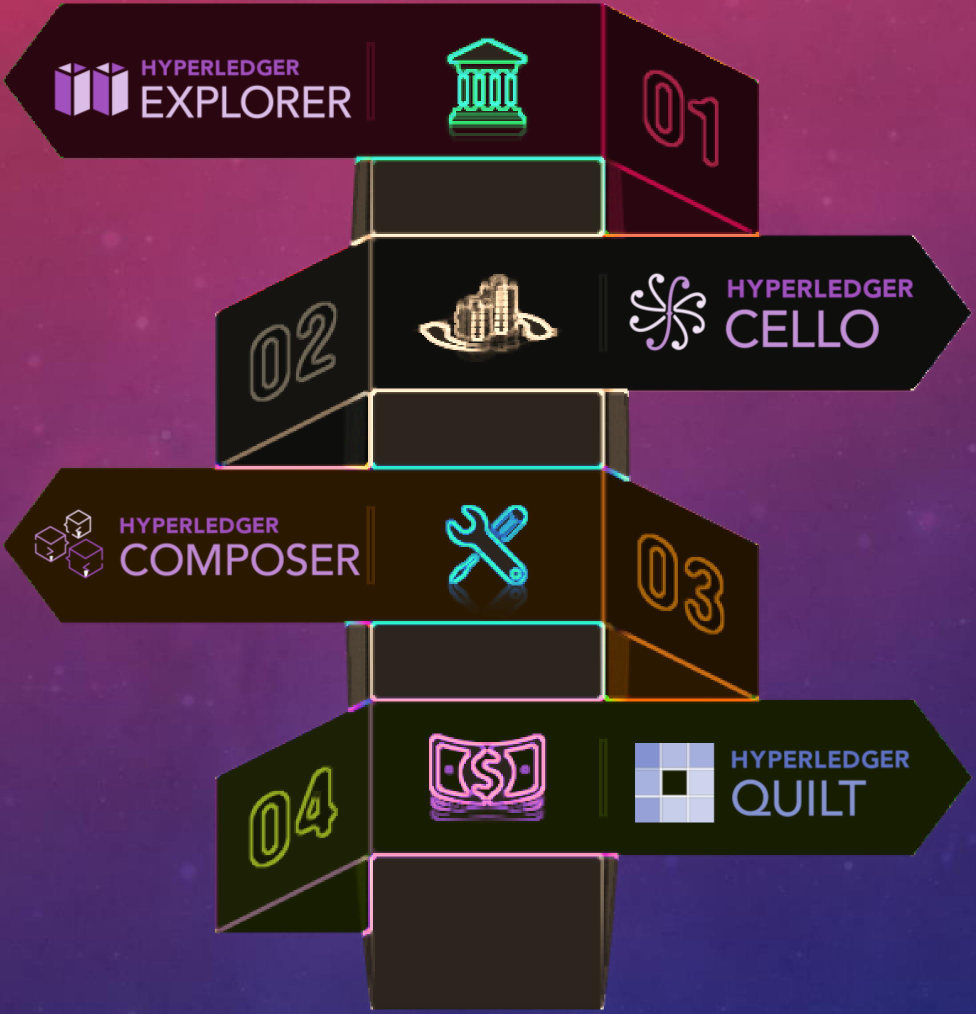
- The fabric is a blockchain project that was proposed by **IBM** and **DAH (Digital Asset Holdings)**
- This blockchain framework implementation is intended to provide a foundation for the development of blockchain solutions with a modular architecture.

HYPERLEDGER IROHA
HYPERLEDGER SAWTOOTH
HYPERLEDGER BURROW
HYPERLEDGER FABRIC
HYPERLEDGER INDY

TOOL or MODULES

- This project aims to build a **blockchain explorer for Hyperledger Fabric** that can be used to view and query the transactions, blocks, and associated data from the blockchain.
- It also provides network information and the ability to interact with chain code.

- This utility makes the **development of blockchain solutions** easier by **allowing business processes to be described in a business language**, while abstracting away the low-level smart contract development details.



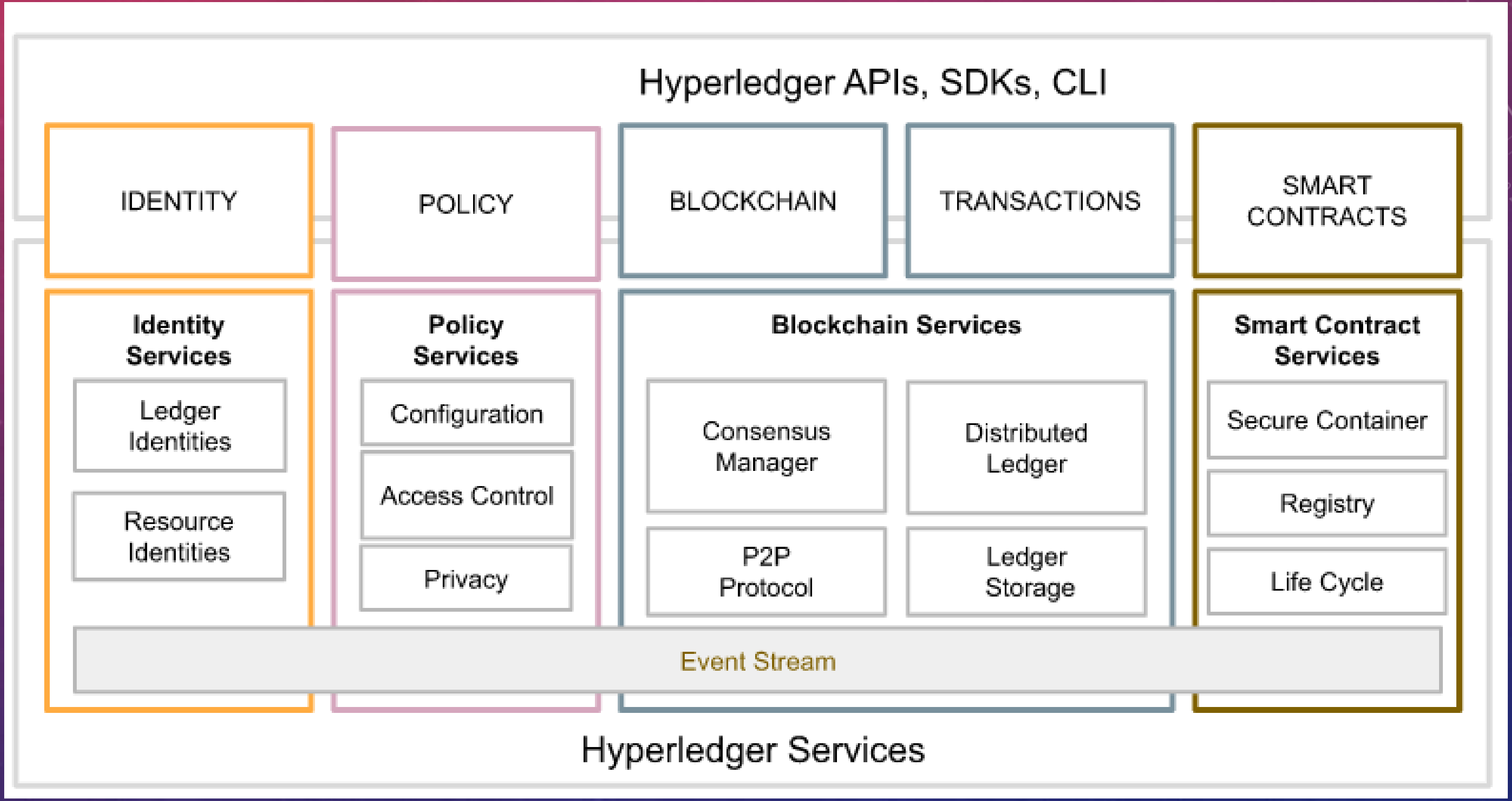
- The aim behind Cello is to allow **easy deployment of blockchains**. This will provide an ability to allow "**as a service**" deployments of blockchain service. Currently, this project is in the incubation stage.

- This utility implements **the Interledger protocol**, which facilitates interoperability across different distributed and non-distributed ledger networks.

HYPERLEDGER AS A PROTOCOL

- Hyperledger is aiming to build new blockchain platforms that are driven by industry use cases.
- As there have been many contributions made to the Hyperledger project by the community, Hyperledger blockchain platform is evolving into a protocol for business transactions. Hyperledger is also evolving into a specification that can be used as a reference to build blockchain platforms as compared to earlier blockchain solutions that address only a specific type of industry or requirement.
- a reference architecture is presented that has been published by the Hyperledger project.

THE REFERENCE ARCHITECTURE



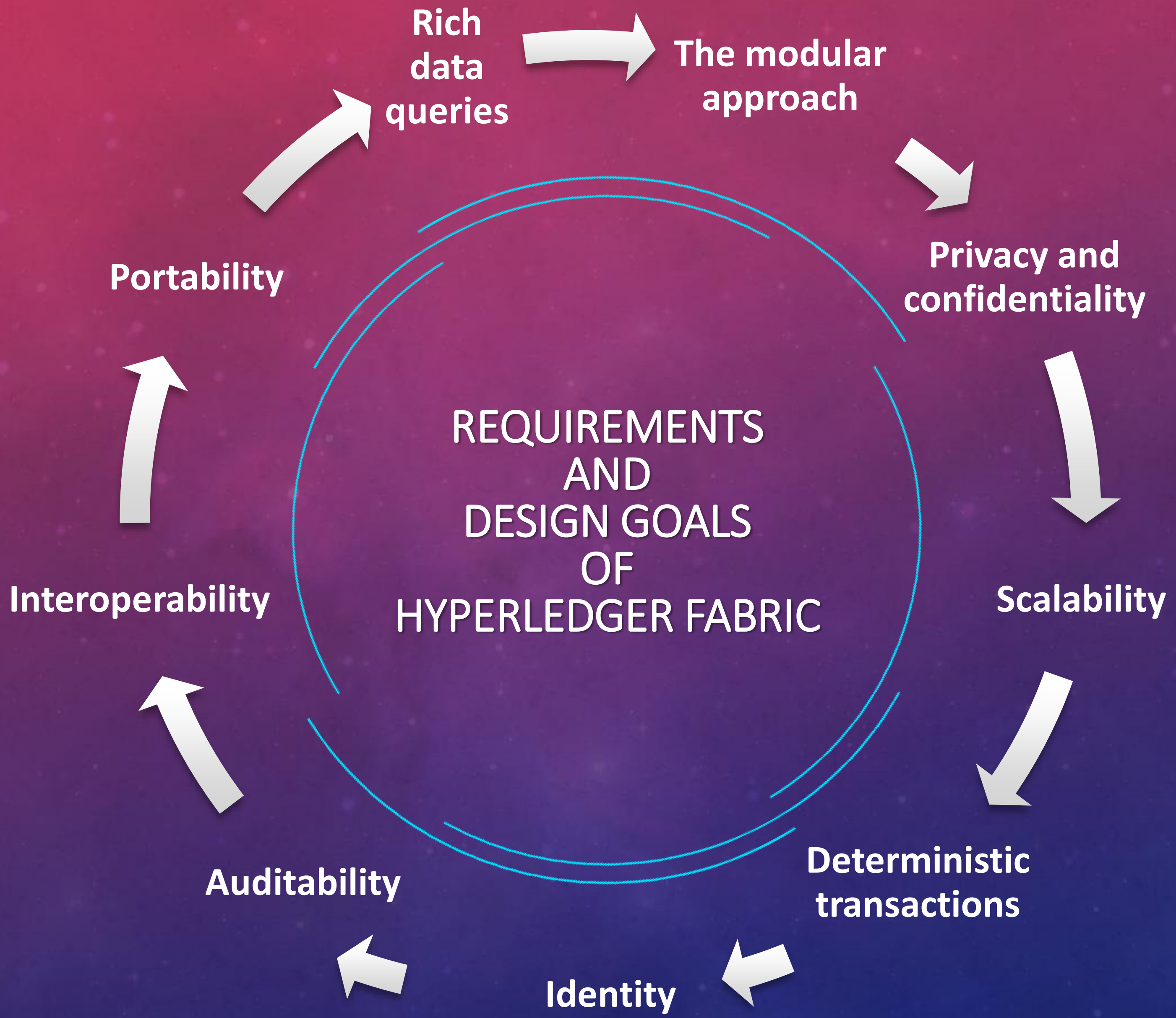
- **Identity** that provides authorization, identification, and authentication services under membership services.

- **Policy** component, which provides policy services.

- **Ledger and transactions,** which consists of the distributed ledger, ordering service, network protocols, and endorsement and validation services. This ledger is updateable only via consensus among the participants of the blockchain network.

- **Smart contracts layer**, which provides chaincode services in Hyperledger and makes use of secure container technology to host smart contracts. We will see all these in more detail in the Hyperledger Fabric section shortly.

REQUIREMENTS AND DESIGN GOALS OF HYPERLEDGER FABRIC



Fabric

- Fabric can be defined as a collection of components providing a foundation layer that can be used to deliver a blockchain network.
- There are various types and capabilities of a fabric network, but all fabrics share common attributes such as immutability and are consensus-driven.

HYPERLEDGER FABRIC

- The fabric is the contribution made initially by IBM and Digital Assets Holding to the Hyperledger project. This contribution aims to enable a modular, open, and flexible approach towards building blockchain networks.
- Various functions in the fabric are pluggable, and it also allows the use of any language to develop smart contracts. This functionality is possible because it is based on container technology (Docker), which can host any language
- Chaincode is sandboxed in a secure container, which includes a secure operating system, chaincode language, runtime environment, and SDKs for Go, Java, and Node.js.
- Transactions in the fabric are private, confidential, and anonymous for general users, but they can still be traced and linked to the users by authorized auditors.
- As a permissioned network, all participants are required to be registered with the membership services to access the blockchain network

Membership services

- User identity verification
- User registration
- Assign appropriate permissions to the users depending on their roles

- Membership services make use of a **certificate authority (Fabric CA)** in order to support identity management and authorization operations.

- Fabric CA issues enrollment certificates (E-Certs), which are produced by enrollment certificate authority (E-CA)

Blockchain services

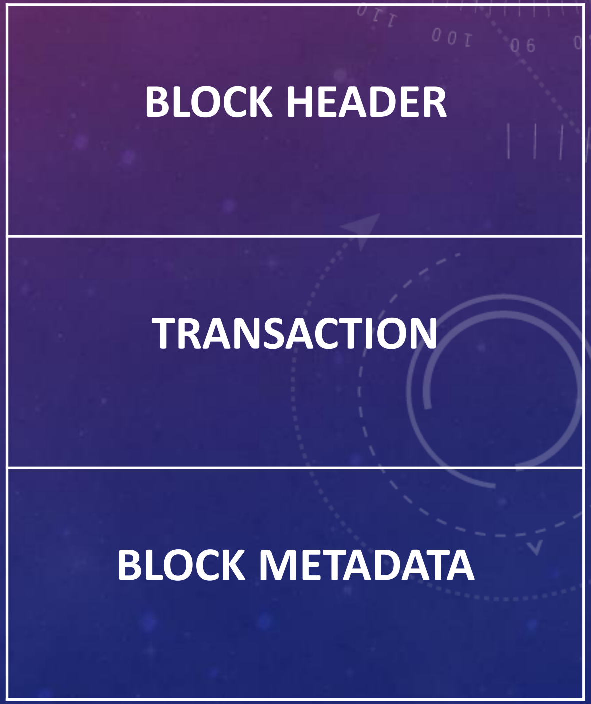
Consensus services:

- A consensus service is responsible for providing the interface to the consensus mechanism.
- Consensus in Hyperledger V1 is implemented as a peer called orderer, which is responsible for ordering the transactions in sequence into a block. Orderer does not hold smart contracts or ledgers.
- There are two types of ordering services available in Hyperledger Fabric:
  - **SOLO:** This is a basic ordering service intended to be used for development and testing purposes.
  - **Kafka:** This is an implementation of Apache Kafka, which provides ordering service. It should be noted that currently Kafka only provides crash fault tolerance but does not provide byzantine fault tolerance. This is acceptable in a permissioned network where chances of malicious actors are almost none.

Distributed ledger

- **Blockchain** and **world state** are two main elements of the distributed ledger. Blockchain is simply a cryptographically linked list of blocks and world state is a key-value database.
- These transactions contain chaincode, which runs transactions that can result in updating the world state. Each node saves the world state on disk in LevelDB or CouchDB depending on the implementation.

- **Block Header** consists of three fields, namely Number, Previous hash, and Data hash.
- **Transaction** is made up of multiple fields such as transaction type, version, timestamp, channel ID, transaction ID, epoch, payload visibility, chaincode path, chaincode name, chaincode version, creator identity, signature, chaincode type, input, timeout, endorser identities and signatures, proposal hash, chaincode events, response status, namespace, read set, write set, start key, end key, list of read, and Merkle tree query summary.
- **Block Metadata** consists of creator identity, relevant signatures, last configuration block number, flag for each transaction included in the block, and last offset persisted (kafka).



The peer to peer protocol

- The P2P protocol in the Hyperledger Fabric is built using google RPC (gRPC). It uses protocol buffers to define the structure of the messages.
- There are four main types of messages in Hyperledger Fabric:
  - **Discovery**
  - **Transaction**
  - **Synchronization**
  - **Consensus**

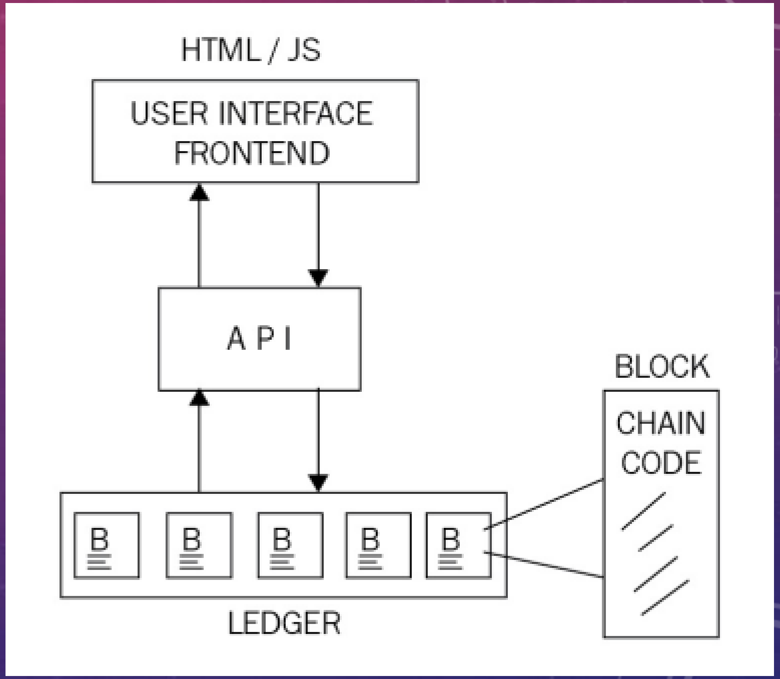
  - **Discovery** messages are exchanged between nodes when starting up in order to discover other peers on the network.
  - **Transaction** messages are used to deploy, invoke, and query transactions.
  - **Consensus** messages are exchanged during consensus.
  - **Synchronization** messages are passed between nodes to synchronize and keep the blockchain updated on all nodes.

Ledger storage

- In order to save the state of the ledger, by default, LevelDB is used which is available at each peer. An alternative is to use CouchDB which provides the ability to run rich queries.

Chaincode services

- These services allow the creation of secure containers that are used to execute the chaincode.
- **Secure container:** Chaincode is deployed in Docker containers that provide a locked down sandboxed environment for smart contract execution. Currently, Golang is supported as the main smart contract language, but any other mainstream languages can be added and enabled if required.
- **Secure registry:** This provides a record of all images containing smart contracts.



Components of the fabric

- There are various components that can be part of the Hyperledger Fabric blockchain.
- These components include but are not limited to the ledger, chaincode, consensus mechanism, access control, events, system monitoring and management, wallets, and system integration components.

Peers

- Peers participate in maintaining the state of the distributed ledger. They also hold a local copy of the distributed ledger.
- Peers communicate via gossip protocol.
- There are three types of peers in the Hyperledger Fabric network:
    - **Endorsing peers or endorsers** which simulate the transaction execution and generate a read-write set. Read is a simulation of transaction's reading of data from the ledger and write is the set of updates that would be made to the ledger if and when the transaction is executed and committed to the ledger. Endorses execute and endorse transactions. It should be noted that an endorser is also a committer too. Endorsement policies are implemented with chaincode and specify the rules for transaction endorsement.
    - **Committing peers or committers** which receives transaction endorsed by endorsers, verify them and then update the ledger with the read-write set. A committer verifies the read-write set generated by the endorsers along with transaction validation.
    - **Submitters** is the third type of peers which has not been implemented yet. It is on the development roadmap and will be implemented

Orderer nodes

- Ordering nodes receive transactions from endorsers along with read-write sets, arrange them in a sequence, and send those to committing peers. Committing peers then perform validation and committing to the ledger.

- All peers make use of certificates issued by membership services.

Clients

- Clients are software that makes use of APIs to interact with the Hyperledger Fabric and propose transactions.

Channels

- Channels allow the flow of confidential transactions between different parties on the network. They allow using the same blockchain network but with separate blockchains.

- Channels allow only members of the channel to view the transaction related to them, all other members of the network will not be able to view the transactions.

World state database

- World state reflects all committed transaction on the blockchain. This is basically a key-value store which is updated as a result of transactions and chaincode execution. For this purpose, either LevelDB or CouchDB is used.
- LevelDB is a key-value store whereas CouchDB stores data as JSON objects which allows rich queries to run against the database.

Transactions

- Transaction messages can be divided into two types: deployment transactions and invocation transactions. The former is used to deploy new chaincode to the ledger, and the latter is used to call functions from the smart contract. Transactions can be either public or confidential.
- Public transactions are open and available to all participants whilst confidential transactions are visible only in a channel open to its participants.

Membership Service Provider (MSP)

- MSP is a modular component that is used to manage identities on the blockchain network. This provider is used to authenticate clients who want to join the blockchain network.

Smart contracts

- In Hyperledger Fabric same concept of smart contracts is implemented but they are called chain code instead of smart contracts.
- They contain conditions and parameters to execute transactions and update the ledger. Chaincode is usually written in Golang

Crypto service provider

- As the name suggests this is a service that provides cryptographic algorithms and standards for usage in the blockchain network.
- This service provides key management, signature and verification operations, and encryption-decryption mechanisms.
- This service is used with the membership service to provide support for cryptographic operations for elements of blockchain such as endorsers, clients, and other nodes and peers.

Scalability and Other Challenges

- Even though various use cases and proof of concept systems have been developed and the technology works well for many of the scenarios, there still is a need to address some fundamental limitations that are present in blockchains in order to make this technology more adaptable.

- At the top of the list of these issues comes scalability and then privacy.

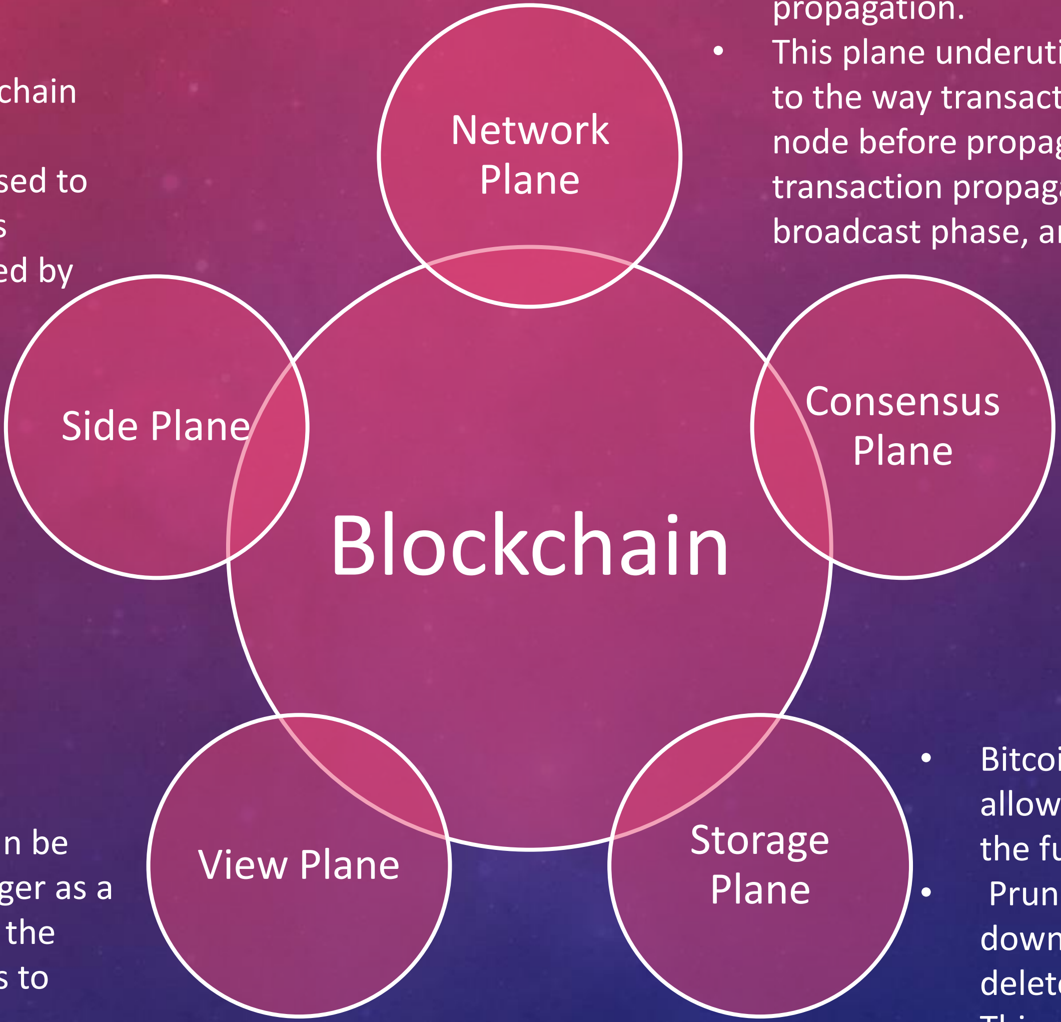
- Both of these are important limitations to address, especially as blockchains are envisioned to be used in privacy-demanding industries too. There are specific requirements around confidentiality of transactions in finance, law, and health.

- scalability is generally a concern where blockchains do not meet the adequate performance levels expected by the users.

Approach toward tackling the scalability issue:

- **Block size increase**
  - This is the most debated proposal for increasing blockchain performance (transaction processing throughput).
  - Currently, Bitcoin can process only about three to seven transactions per second, which is a major inhibiting factor in adapting the Bitcoin blockchain for processing microtransactions.
  - Block size in Bitcoin is hardcoded to be 1 MB, but if the block size is increased, it can hold more transactions and can result in faster confirmation time. There are several **Bitcoin Improvement Proposals (BIPs)** made in favor of block size increase. These include BIP 100, BIP 101, BIP 102, BIP 103, and BIP 109.
  - In Ethereum, the block size is not limited by hardcoding; instead, it is controlled by a gas limit. In theory, there is no limit on the size of a block in Ethereum because it's dependent on the amount of gas, which can increase over time

- a blockchain can be divided into various abstract layers called planes. Each plane is responsible for performing specific functions. These include the network plane, consensus plane, storage plane, view plane, and side plane.



- A key function of the network plane is transaction propagation.
- This plane underutilizes the network bandwidth due to the way transaction validation is performed by a node before propagation and duplication of transaction propagation, first in the transaction broadcast phase, and then after mining in a block.

- This plane represents the idea of off-chain transactions whereby the concept of payment or transaction channels is used to offload the processing of transactions between participants but is still backed by the main Bitcoin blockchain.

- This layer is responsible for mining and achieving consensus.
- Bottlenecks in this layer revolve around limitations in PoW algorithms whereby increasing consensus speed and bandwidth results in compromising the security of the network due to an increase in the number of forks.

- Bitcoin miners do not need the full blockchain to operate, and a view can be constructed out of the complete ledger as a representation of the entire state of the system, which is sufficient for miners to function.

- Bitcoin has a method available called pruning, which allow s a node to operate without the need to keep the full blockchain in its storage.
- Pruning means that when a Bitcoin node has downloaded the blockchain and validated it, it deletes the old data that it has already validated. This saves storage space. This functionality has resulted in major improvements from a storage point of view.

Block interval reduction

- Another proposal is to reduce the time between each block generation. The time between blocks can be decreased to achieve faster finalization of blocks but may result in less security due to the increased number of forks.
- Ethereum has achieved a block time of approximately 14 seconds.

Invertible Bloom Lookup Tables

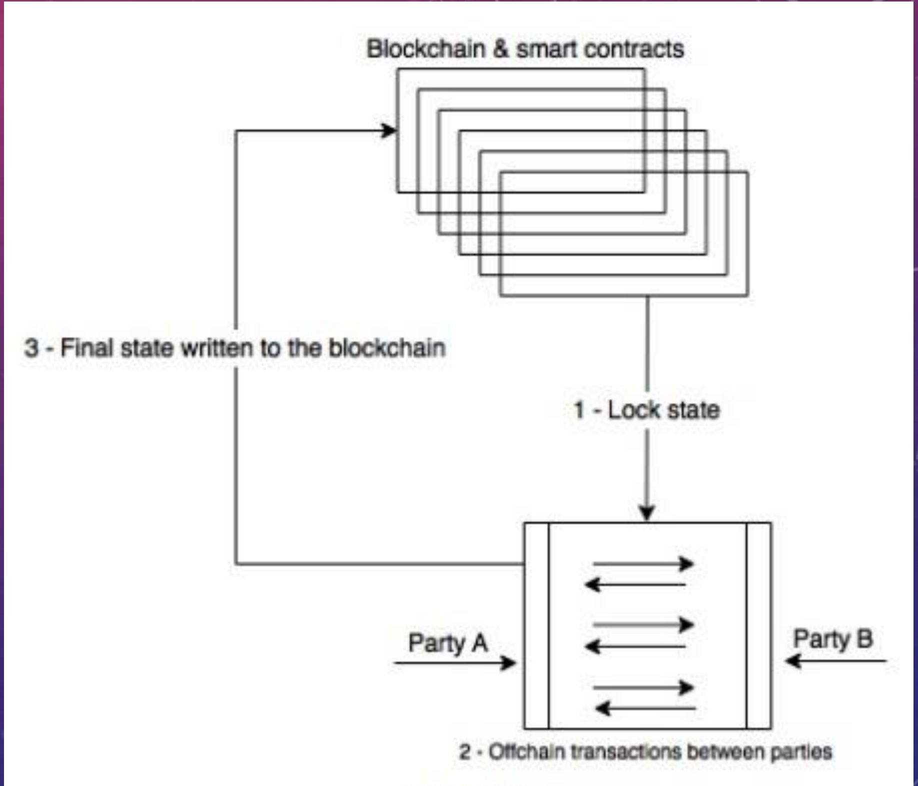
- The key idea is based on the fact that there is no need to transfer all transactions between nodes; instead, only those that are not already available in the transaction pool of the syncing node are transferred.
- This allows quicker transaction pool synchronization between nodes, thus increasing the overall scalability and speed of the Bitcoin

Sharding

- The key idea behind sharding is to split up the tasks into multiple chunks that are then processed by multiple nodes. This results in improved throughput and reduced storage requirements.
- In blockchains, a similar scheme is employed whereby the state of the network is partitioned into multiple shards. The state usually includes balances, code, nonce, and storage. Shards are loosely coupled partitions of a blockchain that run on the same network. There are a few challenges related to inter-shard communication and consensus on the history of each shard.

State channels

- The basic idea is to use side channels for state updating and processing transactions off the main chain; once the state is finalized, it is written back to the main chain, thus offloading the time-consuming operations from the main blockchain.
- State channels work by performing the following three steps:
  1. First, a part of the blockchain state is locked under a smart contract, ensuring the agreement and business logic between participants.
  2. Now off-chain transaction processing and interaction is started between the participants that update the state only between themselves for now. In this step, almost any number of transactions can be performed without requiring the blockchain and this is what makes the process fast and a best candidate for solving blockchain scalability issues. However, it could be argued that this is not a real on-blockchain solution such as, for example, sharding, but the end result is a faster, lighter, and robust network which can prove very useful in micropayment networks, IoT networks, and many other applications.
  3. Once the final state is achieved, the state channel is closed and the final state is written back to the main blockchain. At this stage, the locked part of the blockchain is also unlocked.


Blockchain & smart contracts
3 - Final state written to the blockchain
1 - Lock state
Party A
Party B
2 - Offchain transactions between parties
State channels

- Private blockchain
- Proof of Stake
- Sidechains
- Subchains
- Tree chains
- Block propagation

Plasma

- This proposal describes the idea of running smart contracts on root blockchain (Ethereum MainNet) and have child blockchains that perform high number of transactions to feedback small amounts of commitments to the parent chain.
- In this scheme, blockchains are arranged in a tree hierarchy with mining performed only on the root (main) blockchain which feeds the proofs of security down to child chains. This is also called a Layer-2 system, like state channels also operate on Layer 2, and not on the main chain.

Privacy

- Privacy of transactions is a much-desired property of blockchains.
- However, due to its very nature, especially in public blockchains, everything is transparent, thus inhibiting its usage in various industries where privacy is of paramount importance, such as finance, health, and many others.
- There are different proposals made to address the privacy issue and some progress has already been made. Several techniques, such as Indistinguishability Obfuscation (IO), usage of homomorphic encryption, ZKPs, and ring signatures

Indistinguishability Obfuscation

- This cryptographic technique may serve as a silver bullet to all privacy and confidentiality issues in blockchains but the technology is not yet ready for production deployments.
- IO allows for code obfuscation, which is a very ripe research topic in cryptography and, if applied to blockchains, can serve as an unbreakable obfuscation mechanism that will turn smart contracts into a black box.

Homomorphic encryption

- This type of encryption allows operations to be performed on encrypted data. Imagine a scenario where the data is sent to a cloud server for processing. The server processes it and returns the output without knowing anything about the data that it has processed.
- d fully homomorphic encryption that allows all operations on encrypted data is still not fully deployable in production

Zero-Knowledge Proofs

- Zero-Knowledge Proof is a cryptographic technique where no information is revealed during a transaction except for the interchange of some value known to both the prover and verifiers (the two ends of the process).
- The idea behind zero-knowledge proof is that a user can prove to another user that they know an absolute value without actually revealing any other or extra information.

State channels

- Privacy using state channels is also possible, simply due to the fact that all transactions are run off-chain and the main blockchain does not see the transaction at all except for the final state output, thus ensuring privacy and confidentiality.

Secure multiparty computation

- The concept of secure multiparty computation is not new and is based on the notion that data is split into multiple partitions between participating parties under a secret sharing mechanism which then does the actual processing on the data without the need of the reconstructing data on a single machine. The output produced after processing is also shared between the parties.

Usage of hardware to provide confidentiality

Confidential transactions

Security

- Even though blockchains are generally secure and make use of asymmetric and symmetric cryptography as required throughout the blockchain network, there still are few caveats that can result in compromising the security of the blockchain.
- There are a few examples of transaction malleability, eclipse attacks, and the possibility of double spending in bitcoin that, in certain scenarios, have been shown to work by various researchers. Transaction malleability opens up the possibility of double withdrawal or deposit by allowing a hacker to change a transaction's unique ID before the Bitcoin network can confirm it, resulting in a scenario where it would seem that transactions did not occur.
- BIP 62 is one of the proposals along with SegWit that have suggested solutions to solve this issue.

THANK YOU!