

Blockchain Technology (BCS613A)

MODULE 2: Decentralization, Cryptographic primitives

Course Notes

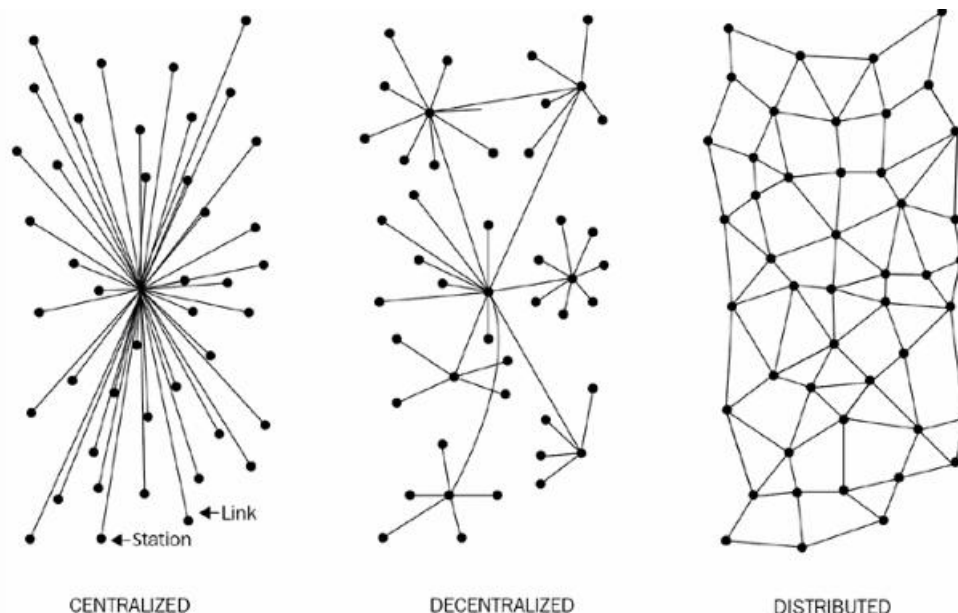
Syllabus:

Decentralization using blockchain, Methods of decentralization, Blockchain and full ecosystem decentralization, Smart contract, Decentralized organizations, Decentralized autonomous organizations, Decentralized autonomous corporations, Decentralized autonomous societies Decentralized applications, Platforms for decentralization.

Cryptographic primitives: Symmetric cryptography, Asymmetric cryptography, Public and private keys, Hash functions: Compression of arbitrary messages into fixed length digest, Easy to compute, Pre-image resistance, Second pre-image resistance, Collision resistance, Message Digest (MD), Secure Hash Algorithms (SHAs), Merkle trees, Patricia trees, Distributed hash tables (DHTs), Digital signatures, Elliptic Curve Digital signature algorithm (ECDSA).

Decentralization using blockchain

- 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.
- **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.



Different types of networks/systems

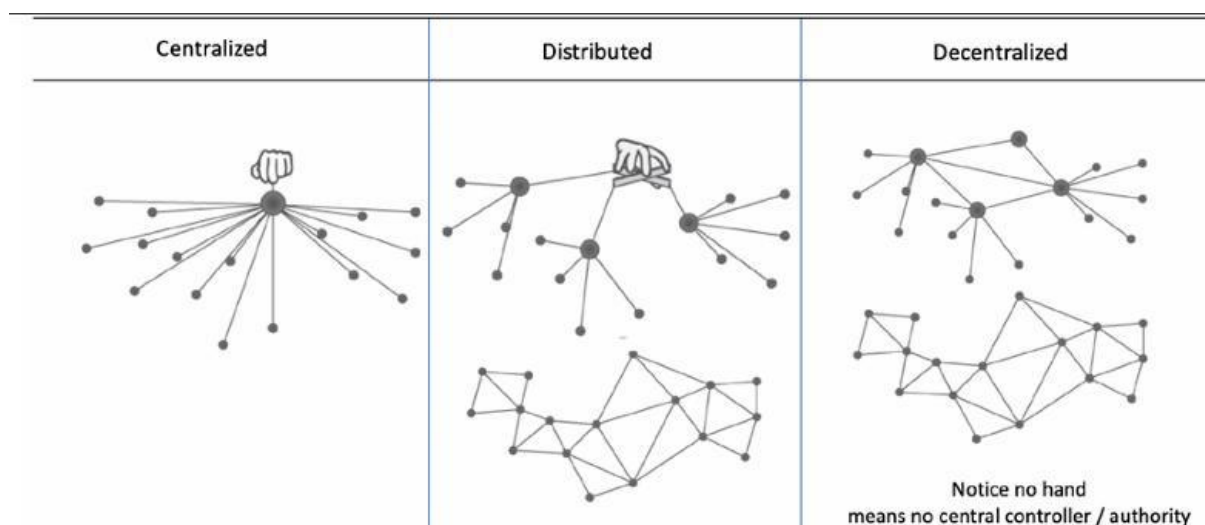
Centralized systems are conventional (client-server) IT systems in which there is a single authority that controls the system, and who is solely in charge of all

operations on the system. The majority of online service providers, including Google, Amazon, eBay, and Apple's App Store, use this conventional model to deliver services.

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. A significant innovation in the decentralized paradigm 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.

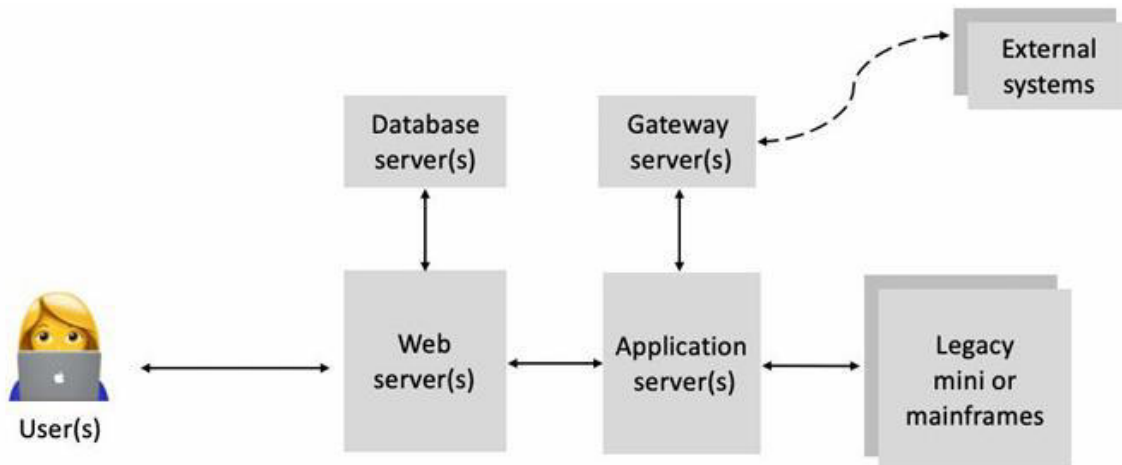
In a distributed system, data and computation are spread across multiple nodes in the network. Sometimes, this term is confused with parallel computing. In a parallel computing system, computation is performed by all nodes simultaneously in order to achieve the result. On the other hand, in a distributed system, computation may not happen in parallel and data is replicated across multiple nodes that users view as a single, coherent system.

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.



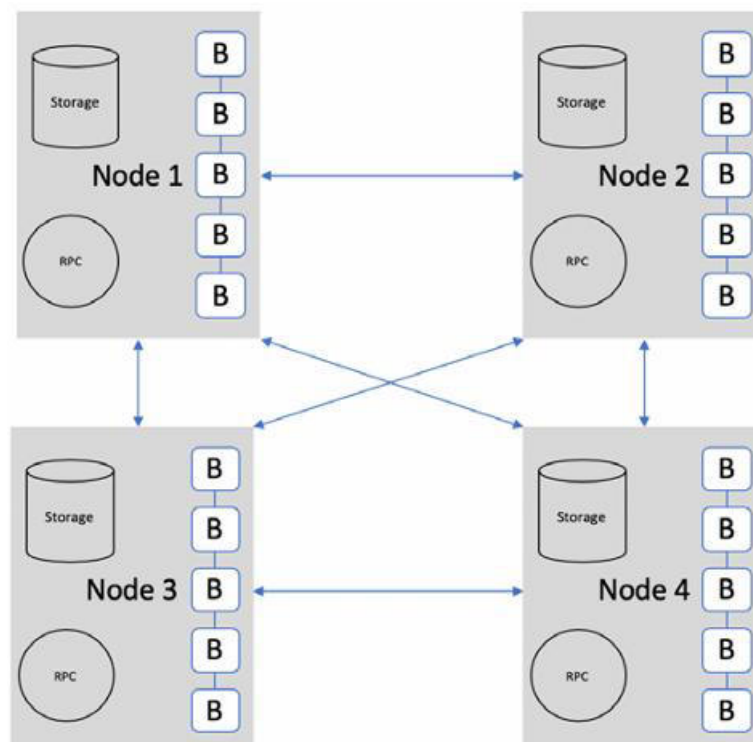
Different types of networks/systems depicting decentralization from a modern perspective

The centralized model is the traditional one in which a central controller exists, and it can be viewed as a depiction of the usual client/server model. In the distributed systems, where we still have a central controller but the system comprises many dispersed nodes. 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 blockchain-based decentralized system

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

Methods of decentralization

Two methods can be used to achieve decentralization: disintermediation and competition.

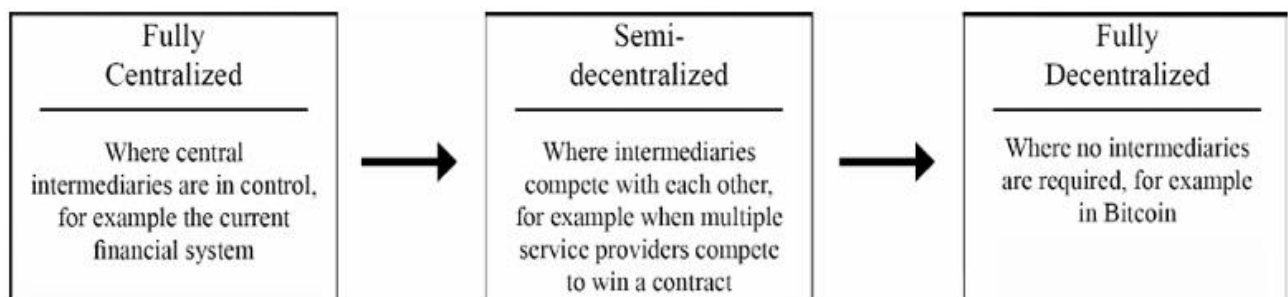
Disintermediation

- With blockchain technology, for example, it is possible to send money directly to someone without the need for a bank.
- 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.
- This way, an environment of competition is cultivated among service providers where they compete with each other to become the data provider of choice.



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.
- For example, in a decentralized system such as Bitcoin or Ethereum where security is normally provided by private keys, how can we ensure that an asset or a token associated with these private keys cannot be rendered useless due to negligence or bugs in the code? What if the private keys are lost due to user negligence? What if due to a bug in the smart contract code the decentralized application becomes vulnerable to attack?

- 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:

- 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.
- If strict data immutability is required, then a public blockchain should be used; otherwise, a central database may be an option.

Blockchain and full ecosystem decentralization

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. However, 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.
- 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.
- 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.
- 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.

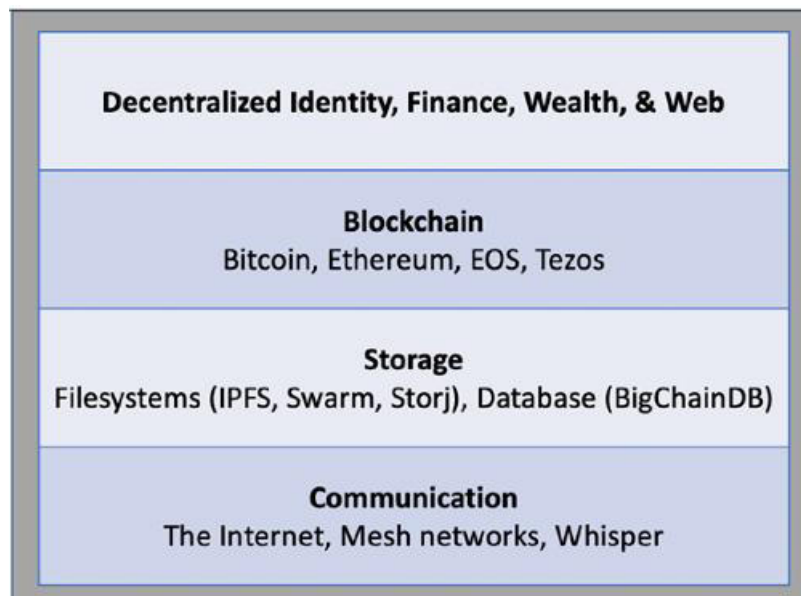
Communication

- The Internet (the communication layer in blockchain) is considered to be decentralized.
- 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.

- 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, they provide a decentralized alternative where nodes can talk directly to each other without a central hub such as an ISP.
- 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.
- 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, the blockchain serves as a decentralized processing (computation) layer. Blockchain can, 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. Systems such as bitAuth and OpenID provide authentication and identification services with varying degrees of decentralization and security assumptions:



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.
- Decentralization may not be appropriate for every scenario. Centralized systems with well established reputations tend to work better in many cases.
- 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.
- With the advent of blockchain technology, it is now possible to build software versions of traditional physical organizations in the form of Decentralized Organizations (DOs) and other similar constructs.

Smart contracts

- A smart contract is a software program that usually runs on a blockchain. Due to the security benefits that blockchain technology provides, blockchain has become a standard decentralized execution platform for smart contracts.
- A smart contract usually contains some business logic and a limited amount of data. The business logic is executed if specific criteria are met.

- Actors or participants in the blockchain use these smart contracts, or they run autonomously on behalf of the network participants.

Autonomous agents

An Autonomous Agent (AA) is an artificially intelligent software entity that acts on the behalf of its owner to achieve some desirable goals without requiring any or minimal intervention from its owner.

Decentralized organizations

- DOs are software programs that run on a blockchain and are based on the idea of actual organizations with people and protocols.
- Once a DO is added to the blockchain in the form of a smart contract or a set of smart contracts, it becomes decentralized and parties interact with each other based on the code defined within the DO software.

Decentralized autonomous organizations

- Just like DOs, a decentralized autonomous organization (DAO) is also a computer program that runs on top of a blockchain, and embedded within it are governance and business logic rules.
- DAOs and DOs are fundamentally the same thing. The main difference, however, is that DAOs are autonomous, which means that they are fully automated and contain artificially intelligent logic.
- DOs, on the other hand, lack this feature and rely on human input to execute business logic.
- Ethereum blockchain led the way with the introduction of DAOs. In a DAO, the code is considered the governing entity rather than people or paper contracts.
- However, a human curator maintains this code and acts as a proposal evaluator for the community. DAOs are capable of hiring external contractors if enough input is received from the token holders (participants).
- Currently, DAOs do not have any legal status, even though they may contain some intelligent code that enforces certain protocols and conditions. However, these rules have no value in the real-world legal system at present.
- The fact that DAOs are purely decentralized entities enables them to run in any jurisdiction. Thus, they raise a big question as to how the current legal system could be applied to such a varied mix of jurisdictions and geographies.

Decentralized autonomous corporations

- Decentralized autonomous corporations (DACs) are similar to DAOs in concept, though considered to be a subset of them.
- The definitions of DACs and DAOs may sometimes overlap, but the general distinction is that DAOs are usually considered to be nonprofit, whereas DACs can earn a profit via shares offered to the participants and to whom they can pay dividends.
- DACs can run a business automatically without human intervention based on the logic programmed into them.

Decentralized autonomous societies

- Decentralized autonomous societies (DASes) are a concept whereby an entire society can function on a blockchain with the help of multiple, complex smart contracts and a combination of DAOs and decentralized applications (DApps) running autonomously.
- This model does not necessarily translate to a free-for-all approach, nor is it based on an entirely libertarian ideology.
- Many services that a government commonly offers can be delivered via blockchains, such as government identity card systems, passports, and records of deeds, marriages, and births.
- Another theory is that, if a government is corrupt and central systems do not provide the levels of trust that a society needs, then that society can start its own virtual one on a blockchain that is driven by decentralized consensus and transparency.
- This concept might look like a libertarian's or cypherpunk's dream, but it is entirely possible on a blockchain.

Decentralized applications

DAOs, DACs, and DOs are DApps that run on top of a blockchain in a peer-to-peer network. They represent the latest advancement in decentralization technology. DApps at a fundamental level are software programs that execute using either of the following methods. They are categorized as Type 1, Type 2, or Type 3 DApps:

Type 1: Run on their own dedicated blockchain, for example, standard smart contract based DApps running on Ethereum. If required, they make use of a native token, for example, ETH on Ethereum blockchain.

Type 2: Use an existing established blockchain. that is, make use of Type 1 blockchain but contains its own stable coins and mechanism of distribution and control, for example, smart contract based tokenization DApps running Ethereum blockchain.

Type 3: Use the protocols of Type 2 DApps; for example, the SAFE Network uses the OMNI network protocol.

Another example is the USDT token (Tethers) which can be considered a Type 3 DApp. The original USDT uses the OMNI layer (a Type 2 DApp) on top of the Bitcoin network. USDT is also available on Ethereum using ERC20 tokens.

There are various categories of DApps covering media, social, finance, games, insurance, and health. There are various decentralized platforms (or blockchains) running, such as Ethereum, EOS, NEO, Loom, and Steem. The highest number of DApps currently is on Ethereum.

Requirements of a DApp

For an application to be considered decentralized, it must meet the following criteria. The General Theory of Decentralized Applications, DApps:

1. The DApp should be fully open source and autonomous, and no single entity should be in control of a majority of its tokens. All changes to the application must be consensus driven based on the feedback given by the community.
2. Data and records of operations of the application must be cryptographically secured and stored on a public, decentralized blockchain to avoid any central points of failure.
3. A cryptographic token must be used by the application to provide access for and incentivize those who contribute value to the applications, for example, miners in Bitcoin.
4. The tokens (if applicable) must be generated by the decentralized application using consensus and an applicable cryptographic algorithm. This generation of tokens acts as a proof of the value to contributors (for example, miners).

Generally, DApps now provide all sorts of different services, including but not limited to financial applications, gaming, social media, and health.

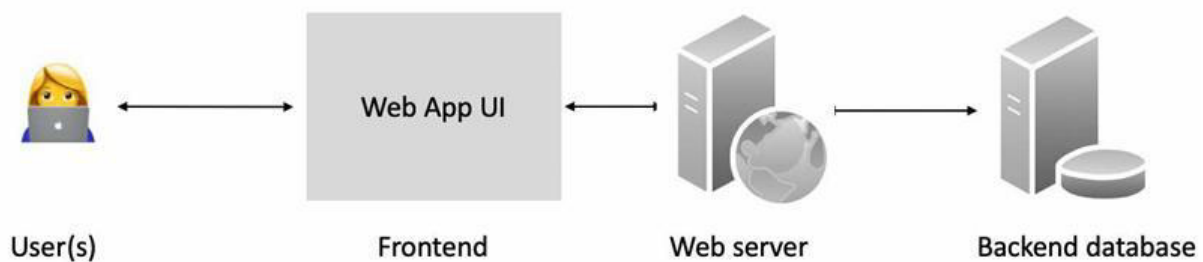
Operations of a DApp

Establishment of consensus by a DApp can be achieved using consensus algorithms such as PoW and Proof of Stake (PoS). So far, only PoW has been found to be incredibly resistant to attacks, as is evident from the success of and trust people have put in the Bitcoin network. Furthermore, a DApp can distribute tokens (coins) via mining, fundraising, and development.

Design of a DApp

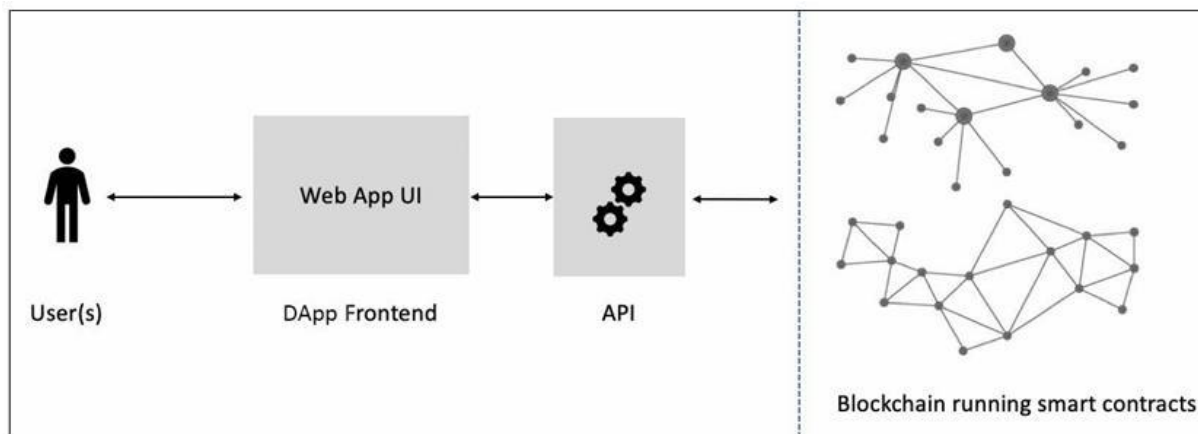
A DApp is a software application that runs on a decentralized network such as a distributed ledger. They have recently become very popular due to the development of various decentralized platforms such as Ethereum, EOS, and Tezos

Traditional apps commonly consist of a user interface and usually a web server or an application server and a backend database. This is a common client/server architecture. This is visualized in the following diagram:



Traditional application architecture (generic client/server)

A DApp on the other hand has a blockchain as a backend and can be visualized as depicted in the following diagram. The key element that plays a vital role in the creation of a DApp is a smart contract that runs on the blockchain and has business logic embedded within it:



Generic DApp architecture

Note that the frontend in either a DApp or app architecture can either be a thick client, a mobile app, or a web frontend (a web user interface). However, it is usually a web frontend commonly written using a JavaScript framework such as React or Angular. The following comparison table highlights the key properties of and differences between these different types of decentralized entities:

Entity	Autonomous?	Software?	Owned?	Capital?	Legal status	Cost
DO	No	No	Yes	Yes	Yes	High
DAO	Yes	Yes	No	Yes	Unsettled	Low
DAC	Yes	Yes	Yes	Yes	Unsettled	Low
DAS	Yes	Yes	No	Possible	Unsettled	Low
DApp	Yes	Yes	Yes	Optional tokens	Unsettled	Use case dependent

DApp examples

Examples of some DApps are provided here.

KYC-Chain

This application provides the facility to manage Know Your Customer (KYC) data securely and conveniently based on smart contracts.

OpenBazaar

This is a decentralized peer-to-peer network that enables commercial activities directly between sellers and buyers instead of relying on a central party, such as eBay or Amazon. It should be noted that this system is not built on top of a blockchain; instead, distributed hash tables are used in a peer-to-peer network to enable direct communication and data sharing among peers. It makes use of Bitcoin and various other cryptocurrencies as a payment method.

Lazooz

This is the decentralized equivalent of Uber. It allows peer-to-peer ride sharing and users to be incentivized by proof of movement, and they can earn Zooz coins.

Platforms for decentralization

- The fundamental feature of blockchain networks is to provide decentralization.
- Therefore, any blockchain network, such as Bitcoin, Ethereum, Hyperledger Fabric, or Quorum, can be used to provide a decentralization service.
- Many organizations around the world have introduced platforms that promise to make distributed application development easy, accessible, and secure.
- Some of these platforms are described as follows.

Ethereum

- Ethereum tops the list as being the first blockchain to introduce a Turing-complete language and the concept of a virtual machine.
- This is in stark contrast to the limited scripting language in Bitcoin and many other cryptocurrencies.
- With the availability of its Turing-complete language, Solidity, endless possibilities have opened for the development of decentralized applications.
- This blockchain was first proposed in 2013 by Vitalik Buterin, and it provides a public blockchain to develop smart contracts and decentralized applications.
- Currency tokens on Ethereum are called ethers.

MaidSafe

- This is not a blockchain, but a decentralized and autonomous network.
- MaidSafe provides a SAFE (Secure Access for Everyone) network that is made up of unused computing resources, such as storage, processing power, and the data connections of its users.
- The files on the network are divided into small chunks of data, which are encrypted and distributed randomly throughout the network.
- This data can only be retrieved by its respective owner. One key innovation of MaidSafe is that duplicate files are automatically rejected on the network, which helps reduce the need for additional computing resources needed to manage the load.
- It uses Safecoin as a token to incentivize its contributors.

Lisk

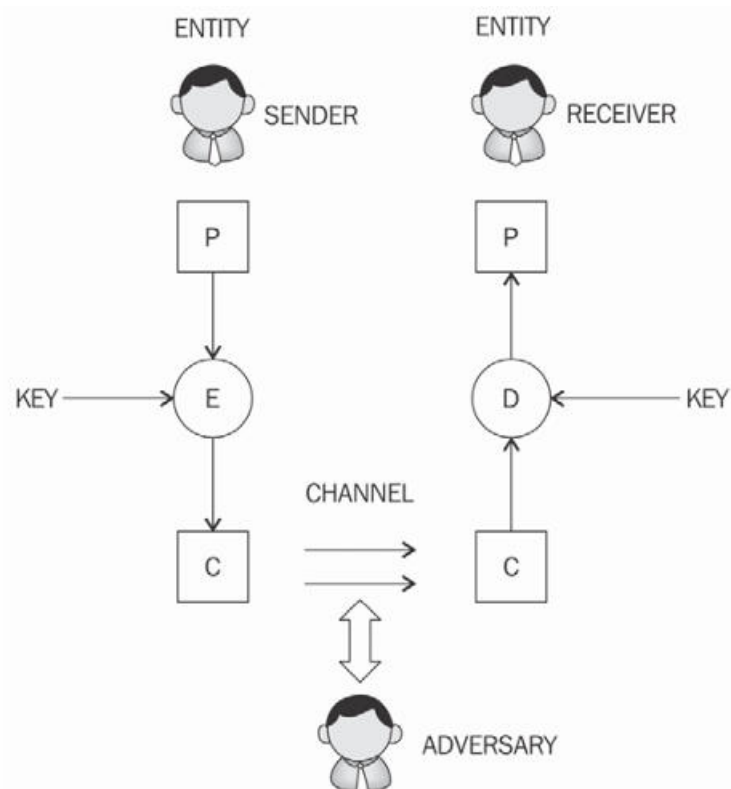
- Lisk is a blockchain application development and cryptocurrency platform.
- It allows developers to use JavaScript to build decentralized applications and host them in their respective sidechains.
- Lisk uses the Delegated Proof of Stake (DPOS) mechanism for consensus, whereby 101 nodes can be elected to secure the network and propose blocks.
- It uses the Node.js and JavaScript backend, while the frontend allows the use of standard technologies, such as CSS3, HTML5, and JavaScript.
- Lisk uses LSK coin as a currency on the blockchain. Another derivative of Lisk is Rise, which is a Lisk-based DApp and digital currency platform.
- It offers greater focus on the security of the system.

EOS

- This is a blockchain protocol launched in January 2018, with its own cryptocurrency called EOS.
- EOS raised an incredible 4 billion USD in 2018 through its Initial Coin Offering (ICO).
- The key purpose behind EOS is to build a decentralized operating system.
- Its throughput is significantly higher (approx. 3,996 transactions per second (TPS)) than other common blockchain platforms, such as Bitcoin (approx. 7 TPS) and Ethereum (approx. 15 TPS).

Cryptography

Cryptography is the science of making information secure in the presence of adversaries. Ciphers are algorithms used to encrypt or decrypt data so that if intercepted by an adversary, the data is meaningless to them without decryption, which requires a secret key.



Generic encryption and decryption model

In the preceding diagram, P, E, C, and D represent plaintext, encryption, ciphertext, and decryption, respectively.

- Entity: Either a person or system that sends, receives, or performs operations on data
- Sender: This is an entity that transmits the data
- Receiver: This is an entity that takes delivery of the data
- Adversary: This is an entity that tries to circumvent the security service
- Key: A key is data that is used to encrypt the plaintext and also to decrypt the ciphertext
- Channel: A channel provides a medium of communication between entities

Cryptography provides various services:

Confidentiality

Confidentiality is the assurance that information is only available to authorized entities.

Integrity

Integrity is the assurance that information is modifiable only by authorized entities.

Authentication

- Authentication provides assurance about the identity of an entity or the validity of a message. There are two types of authentication mechanisms, entity authentication and data origin authentication:
- **Entity authentication** is the assurance that an entity is currently involved and active in a communication session. Traditionally, users are issued a username and password that is used to gain access to the various platforms with which they are working. This practice is known as single-factor authentication.
- The use of additional techniques for user identification is known as **multi-factor authentication**.
- The first method uses something you have, such as a hardware token or a smart card. In this case, a user can use a hardware token in addition to login credentials to gain access to a system. This mechanism protects the user by requiring two factors of authentication.
- The second method uses something you are, which uses biometric features to identify the user. With this method, a user's fingerprint, retina, iris, or hand geometry is used to provide an additional factor for authentication.

Data origin authentication

Also known as message authentication, data origin authentication is an assurance that the source of the information is indeed verified. Data origin authentication guarantees data integrity because, if a source is corroborated, then the data must not have been altered. Various methods, such as Message Authentication Codes (MACs) and digital signatures, are most commonly used.

Non-repudiation

Non-repudiation is the assurance that an entity cannot deny a previous commitment or action by providing undeniable evidence. This service produces cryptographic evidence in electronic transactions so that in case of disputes, it can be used as confirmation of an action.

The primary requirements of a non-repudiation protocol are fairness, effectiveness, and timeliness. In many scenarios, there are multiple participants involved in a transaction, as opposed to only two parties. For example, in electronic trading systems, there can be many entities, including clearing agents, brokers, and traders. To address this problem, multi-party non-repudiation (MPNR) protocols have been developed.

Accountability

- Accountability is the assurance that actions affecting security can be traced back to the responsible party.
- This is usually provided by logging and audit mechanisms in systems where a detailed audit is required due to the nature of the business.
- Detailed logs are vital to trace an entity's actions, such as when a trade is placed in an audit record with the date and timestamp and the entity's identity is generated and saved in the log file.
- This log file can optionally be encrypted and be part of the database or a standalone ASCII text log file on a system.

Cryptographic primitives**Cryptographic primitives**

Cryptographic primitives are the basic building blocks of a security protocol or system. A security protocol is a set of steps taken to achieve the required security goals by utilizing appropriate security mechanisms. Various types of security protocols are in use, such as authentication protocols, non-repudiation protocols, and key management protocols.

Symmetric cryptography

Symmetric cryptography refers to a type of cryptography where the key that is used to encrypt the data is the one used for decrypting the data. Thus, it is also known as shared key cryptography. The key must be established or agreed upon before the data exchange occurs between the communicating parties.

In symmetric key systems, only one key is required, which is shared between the sender and the receiver. It is used for both encryption and decryption.

Keys can also be ephemeral (temporary) or static. Ephemeral keys are intended to be used only for a short period of time, such as in a single session between the participants, whereas static keys are intended for long-term usage.

Another type of key is called the master key, which is used for the protection, encryption, decryption, and generation of other keys.

There are different methods to generate keys:

1. Random, where a random number generator is used to generate a random set of bytes that can be used as a key.
2. Key derivation-based, where a single key or multiple keys are derived from a password. A key derivation function (KDF) is used for this purpose. Commonly used key derivation functions are Password Based Key Derivation Function 1 (PBKDF1), PBKDF2, Argon 2, and Scrypt.
3. Using a key agreement protocol, two or more participants run a protocol that produces a key and that key is then shared between participants. In key agreement schemes, all participants contribute equally in the effort to generate the shared secret key. The most commonly used key agreement protocol is the Diffie-Hellman key exchange protocol.

In encryption schemes, there are also some random numbers that play a vital role in the operation of the encryption process:

- The nonce: This is a number that can be used only once in a cryptographic protocol. Nonces can be generated from a large pool of random numbers or they can also be sequential. The most common use of nonces is to prevent replay attacks in cryptographic protocols.
- The initial value or initialization vector (IV) is a random number, but it must be chosen in an unpredictable manner. This means that it cannot be

sequential. IVs are used extensively in encryption algorithms to provide increased security.

- The salt: Salt is a cryptographically strong random value that is typically used in hash functions to provide defense against dictionary or rainbow attacks. Using dictionary attacks, hashing-based password schemes can be broken by trying hashes of millions of words from a dictionary in a brute-force manner and matching it with the hashed password. If a salt is used, then a dictionary attack becomes difficult to run because a random salt makes each password unique.

Message authentication codes (MACs)

Message authentication codes (MACs) are sometimes called keyed hash functions, and they can be used to provide message integrity and authentication. They are used to provide data origin authentication. These are symmetric cryptographic primitives that use a shared key between the sender and the receiver. MACs can be constructed using block ciphers or hash functions.

Hash-based MACs (HMACs)

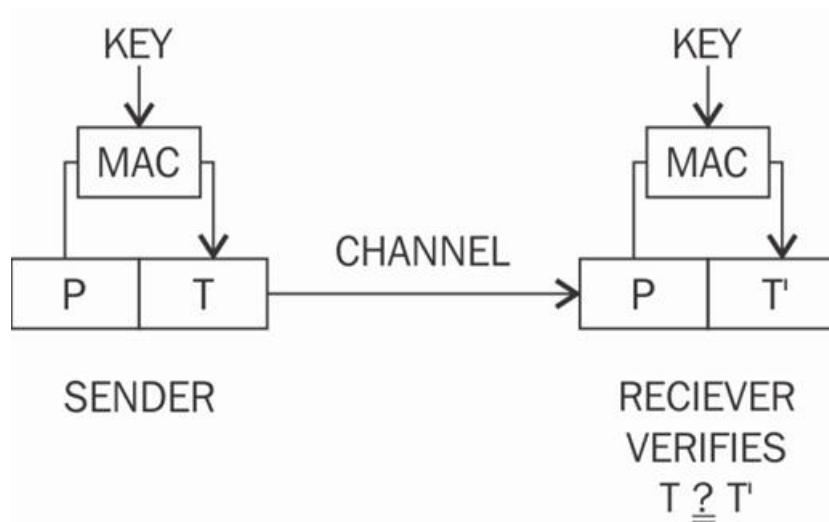
Similar to the hash function, hash-based MACs (HMACs) produce a fixed-length output and take an arbitrarily long message as the input. In this scheme, the sender signs a message using the MAC and the receiver verifies it using the shared key.

The key is hashed with the message using either of the two methods known as secret prefix or secret suffix. With the secret prefix method, the key is concatenated with the message; that is, the key comes first and the message comes afterward, whereas with the secret suffix method, the key comes after the message, as shown in the following equations:

Secret prefix: $M = \text{MAC}_k(x) = h(k||x)$

Secret suffix: $M = \text{MAC}_k(x) = h(x||k)$

Some attacks on both schemes have occurred. There are HMAC constructions schemes that use various techniques, such as ipad (inner padding) and opad (outer padding) that have been proposed by cryptographic researchers. These are considered secure with some assumptions:



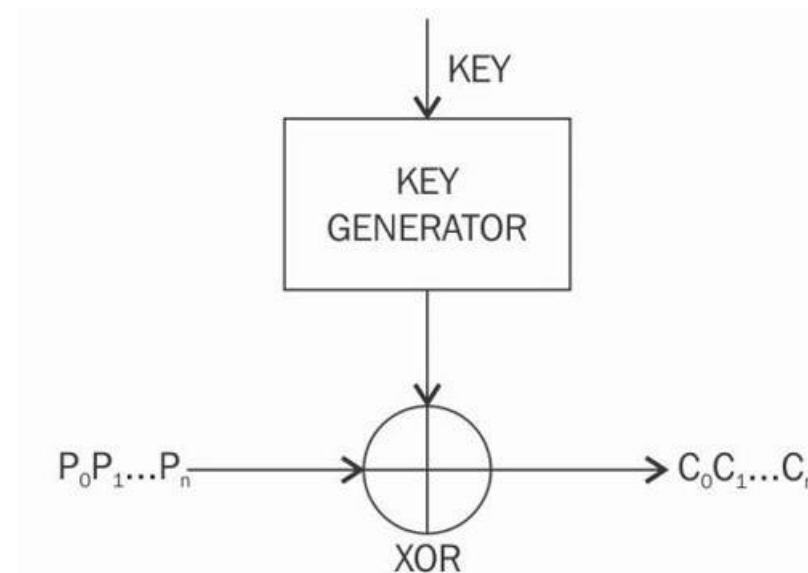
Operation of a MAC function

There are various powerful applications of hash functions used in peer-to-peer networks and blockchain technologies, such as Merkle trees, Patricia trees, and distributed hash tables.

There are two types of symmetric ciphers: stream ciphers and block ciphers. Data Encryption Standard (DES) and Advanced Encryption Standard (AES) are typical examples of block ciphers, whereas RC4 and A5 are commonly used stream ciphers.

Stream ciphers

- Stream ciphers are encryption algorithms that apply encryption algorithms on a bit-by-bit basis (one bit at a time) to plaintext using a keystream.
- There are two types of stream ciphers: synchronous stream ciphers and asynchronous stream ciphers.
- Synchronous stream ciphers are those where the keystream is dependent only on the key. Asynchronous stream ciphers have a keystream that is also dependent on the encrypted data.
- In stream ciphers, encryption and decryption are the same function because they are simple modulo 2 additions or XOR operations.
- Various techniques ranging from PRNGs to true hardware RNGs have been developed to generate random numbers.

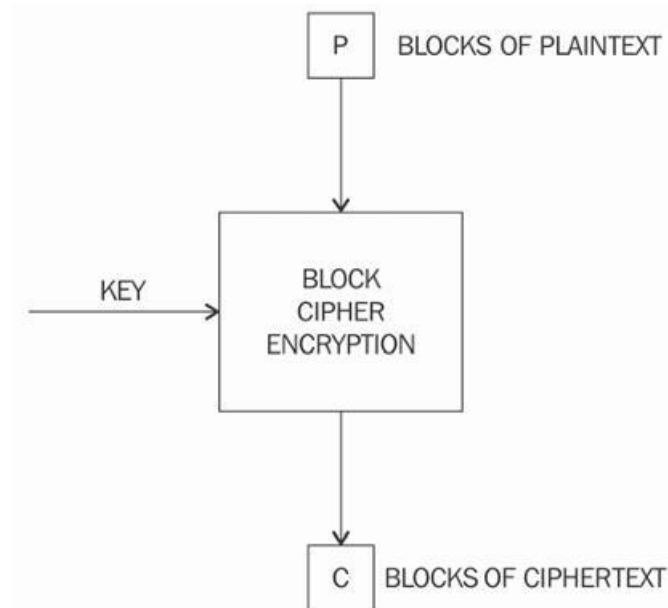


Operation of a stream cipher

Block ciphers

- Block ciphers are encryption algorithms that break up the text to be encrypted (plaintext) into blocks of a fixed length and apply the encryption block by block.
- Block ciphers are generally built using a design strategy known as a Feistel cipher.
- Recent block ciphers such as AES (Rijndael) have been built using a combination of substitution and permutation called a Substitution-Permutation Network (SPN).
- Feistel ciphers are based on the Feistel network, which is a structure developed by Horst Feistel.
- This structure is based on the idea of combining multiple rounds of repeated operations to achieve desirable cryptographic properties known as confusion and diffusion.
- Feistel networks operate by dividing data into two blocks (left and right) and processing these blocks via keyed round functions in iterations to provide sufficient pseudorandom permutations.
- Confusion adds complexity to the relationship between the encrypted text and plaintext. This is achieved by substitution. Substitution is performed using lookup tables called S-boxes.
- The diffusion property spreads the plaintext statistically over the encrypted data. This ensures that even if a single bit is changed in the input text, it results in changing at least half (on average) of the bits in the ciphertext.
- Confusion is required to make finding the encryption key very difficult, even if many encrypted and decrypted data pairs are created using the same key. In practice, this is achieved by transposition or permutation.

A key advantage of using a Feistel cipher is that encryption and decryption operations are almost identical and only require a reversal of the encryption process to achieve decryption. DES is a prime example of Feistel-based ciphers:



Simplified operation of a block cipher

Various modes of operation for block ciphers are Electronic Code Book (ECB), Cipher Block Chaining (CBC), Output Feedback (OFB) mode, and Counter (CTR) mode.

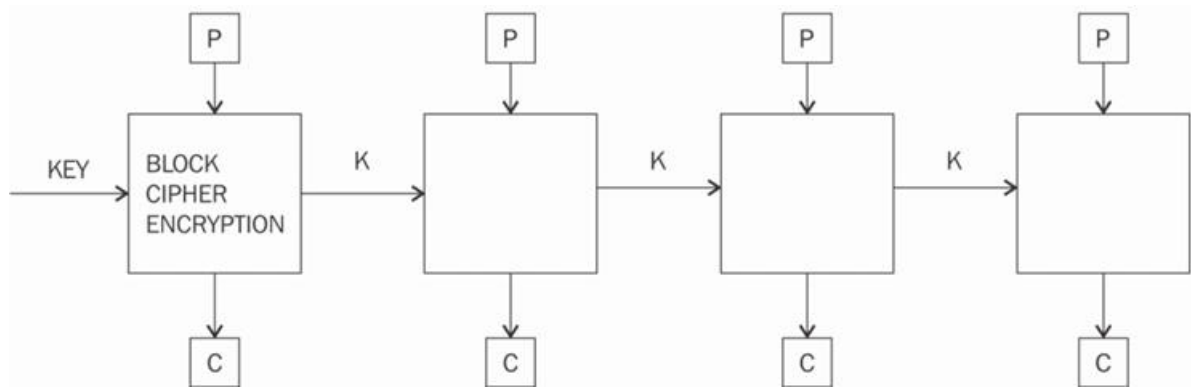
Block encryption mode

In block encryption mode, the plaintext is divided into blocks of fixed length depending on the type of cipher used. Then the encryption function is applied to each block.

Electronic codebook

Electronic codebook (ECB) is a basic mode of operation in which the encrypted data is produced as a result of applying the encryption algorithm to each block of plaintext, one by one. This is the most straightforward mode, but it should not be used in practice as it is insecure and can reveal information:

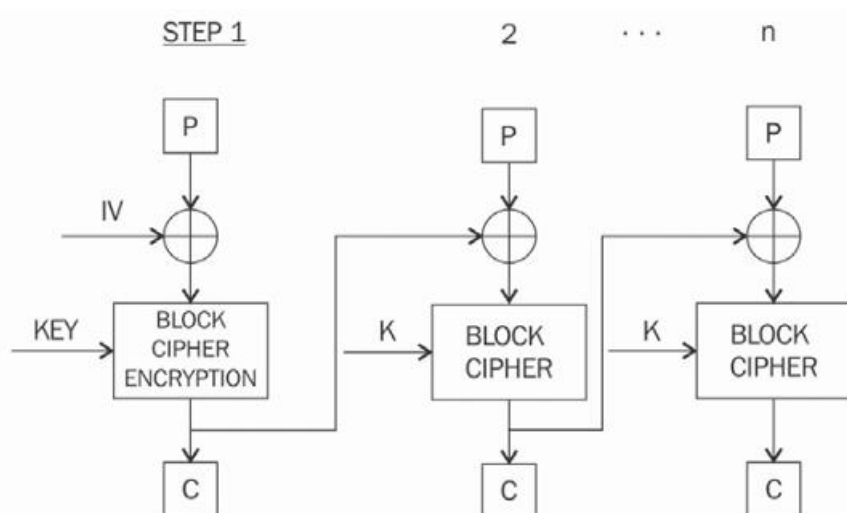
The following diagram shows that we have plain text P provided as an input to the block cipher encryption function along with a key, and ciphertext C is produced as output.



Electronic codebook mode for block ciphers

Cipher block chaining

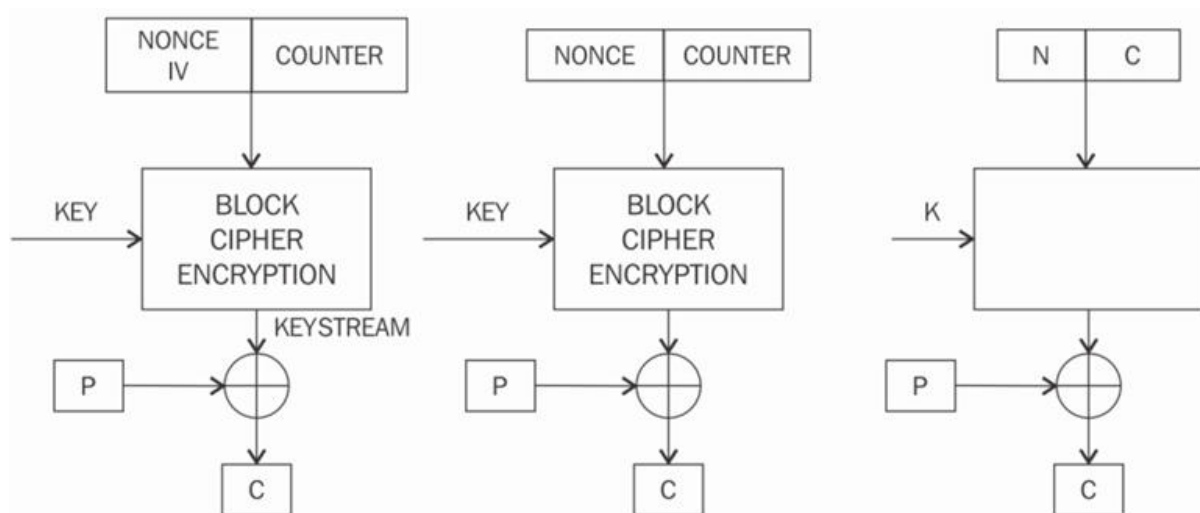
In cipher block chaining (CBC) mode, each block of plaintext is XORed with the previously encrypted block. CBC mode uses the IV to encrypt the first block. It is recommended that the IV be randomly chosen:



Cipher block chaining mode

Counter mode

The counter (CTR) mode effectively uses a block cipher as a stream cipher. In this case, a unique nonce is supplied that is concatenated with the counter value to produce a keystream:



Counter mode

CTR mode, as shown in the preceding diagram, works by utilizing a nonce (N) and a counter (C) that feed into the block cipher encryption function. The block cipher encryption function takes the secret key (KEY) as input and produces a keystream (a stream of pseudorandom or random characters), which, when XORed with the plaintext (P), produces the ciphertext (C).

Keystream generation mode

In keystream generation mode, the encryption function generates a keystream that is used in stream ciphers. Specifically, the keystream is usually XORed with the plaintext stream to produce encrypted text.

Message authentication mode

- In message authentication mode, a message authentication code (MAC) is produced from an encryption function.
- A MAC is a cryptographic checksum that provides an integrity service. The most common method to generate a MAC using block ciphers is CBC-MAC.
- A MAC can be used to check if a message has been modified by an unauthorized entity. This can be achieved by encrypting the message with a key using the MAC function.
- The resulting message and the MAC of the message, once received by the receiver, are checked by encrypting the message received, again with the key, and comparing it with the MAC received from the sender.
- If they both match, then it means that the message has not been modified by some unauthorized entity, thus an integrity service is provided. If they don't match, then it means that the message has been altered by some unauthorized entity during transmission.

- The MAC of the message is, the output of the last round of the CBC operation. The length of the MAC output is the same as the block length of the block cipher used to generate the MAC.
- It should also be noted that MACs work like digital signatures. However, they cannot provide a non repudiation service due to their symmetric nature.

Cryptographic hash mode

- Hash functions are primarily used to compress a message to a fixed-length digest.
- In cryptographic hash mode, block ciphers are used as a compression function to produce a hash of plaintext.
- There are also other modes, such as Cipher Feedback (CFB) mode, Galois Counter (GCM) mode, and Output Feedback (OFB) mode, which are also used in various scenarios.

Data Encryption Standard (DES)

- DES was introduced by NIST as a standard algorithm for encryption, and it was in widespread use during the 1980s and 1990s.
- However, it did not prove to be very resistant to brute force attacks, due to advances in technology and cryptography research.
- In July 1998, for example, the Electronic Frontier Foundation (EFF) broke DES using a special-purpose machine called EFF DES cracker (or Deep Crack).
- This problem was addressed with the introduction of Triple DES (3DES), which proposed the use of a 168-bit key by means of three 56-bit keys and the same number of executions of the DES algorithm, thus making brute force attacks almost impossible.
- However, other limitations, such as slow performance and 64 bit block size, were not desirable.
- With the Data Encryption Standard (DES), plaintext messages are converted into 64-bit blocks, each encrypted using a key.
- The key length is 64 bits but contains only 56 usable bits; thus, the last bit of each byte in the key is a parity bit for the corresponding byte. DES uses a key of only 56 bits, which raised some concerns.
- DES consists of 16 identical rounds of an operation.

DES Algorithm

1. Initialize: All 64 bits of an incoming message and all 56 bits of the secret key are separately permuted (shuffled).

2. Each incoming 64-bit message is broken into two 32-bit halves denoted by L_i and R_i , respectively.
3. The 56 bits of the key are also broken into two 28-bit halves, and each half is rotated one or two bit positions, depending on the round.
4. All 56 bits of the key are permuted, producing version k_i of key on round i .
5. The description of function $F()$: L_i and R_i are determined by

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus F(R_{i-1}, k_i)$$

6. All 64 bits of a message are permuted.

Advanced Encryption Standard (AES)

- In 2001, the encryption algorithm named Rijndael invented by cryptographers Joan Daemen and Vincent Rijmen was standardized as AES.
- So far, no attack has been found against AES that is more effective than the brute-force method.
- In the AES standard, only a 128-bit block size is allowed. However, key sizes of 128 bits, 192 bits, and 256 bits are permissible.

How AES works

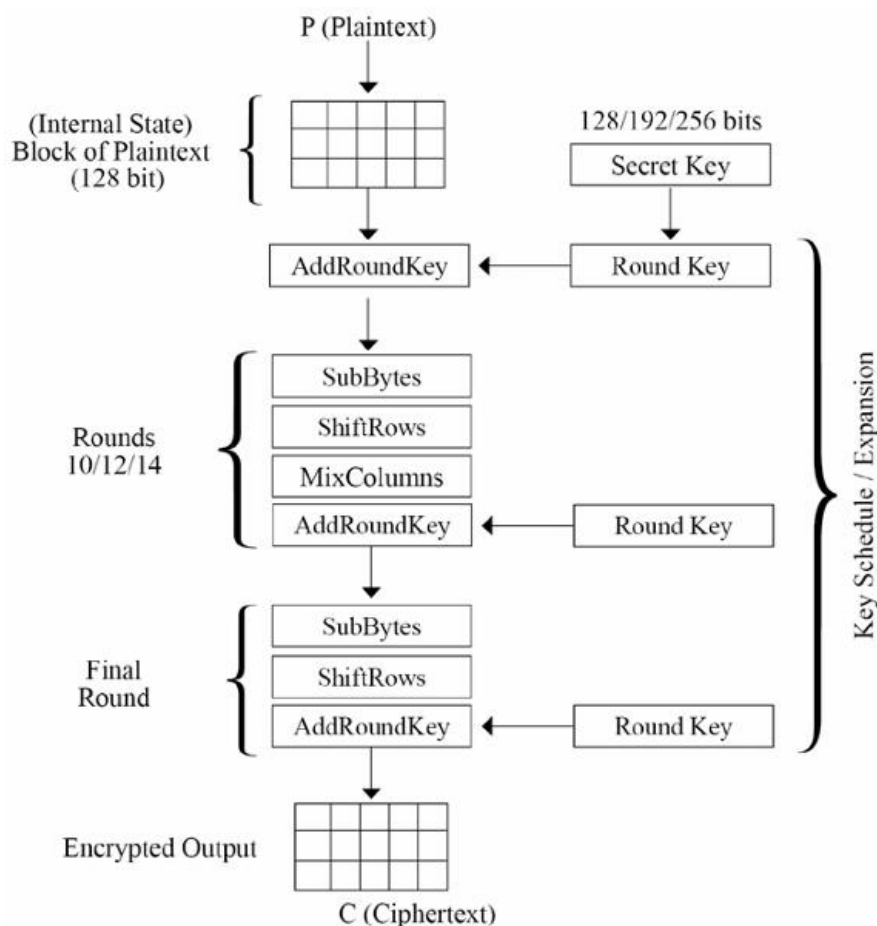
The following table shows the key sizes and the required number of rounds:

Key size	Number of rounds required
128-bit	10 rounds
192-bit	12 rounds
256-bit	14 rounds

The following four operations are performed sequentially step by step to encrypt the input:

AddRoundKey: In this step, the state array is XORed with a subkey, which is derived from the master key.

SubBytes: This is the substitution step where a lookup table (S-box) is used to replace all bytes of the state array.



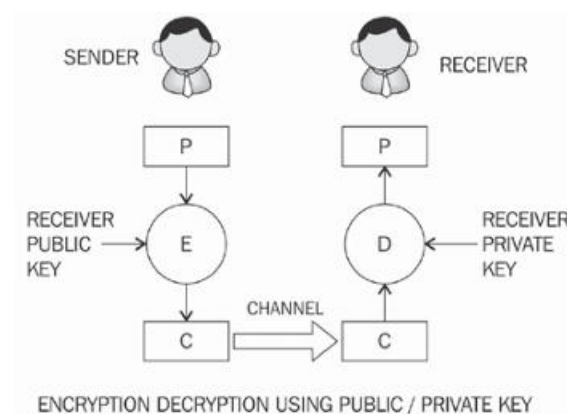
ShiftRows: This step is used to shift each row to the left, except for the first one, in the state array in a cyclic and incremental manner.

MixColumns: Finally, all bytes are mixed in a linear fashion (linear transformation), column-wise.

This is one round of AES. In the final round, stage 4 is replaced with AddRoundKey to ensure that the first three steps cannot be simply reversed, as shown in the above diagram.

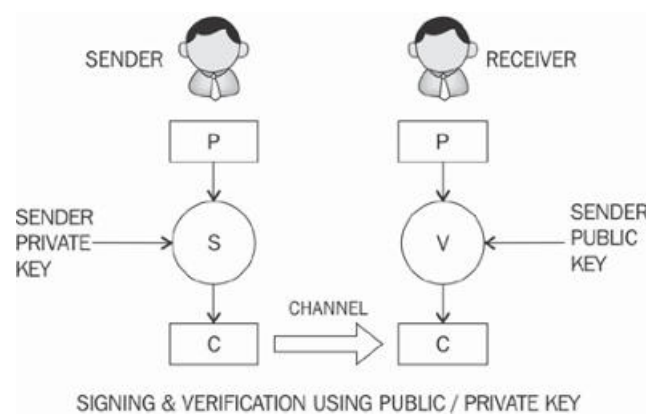
Asymmetric cryptography

- Asymmetric cryptography refers to a type of cryptography where the key that is used to encrypt the data is different from the key that is used to decrypt the data.
- These keys are called private and public keys, respectively.
- Asymmetric cryptography is also known as public key cryptography. It uses both public and private keys to encrypt and decrypt data, respectively.
- Various asymmetric cryptography schemes are in use, including RSA (named after its founders, Rivest, Shamir, and Adelman), DSA (Digital Signature Algorithm), and ElGamal encryption.



Encryption/decryption using public/private keys

- The preceding diagram illustrates how a sender encrypts data **P** using the recipient's public key and encryption function **E**, and produces an output encrypted data **C**, which is then transmitted over the network to the receiver.
- Once it reaches the receiver, it can be decrypted using the receiver's private key by feeding the **C** encrypted data into function **D**, which will output plaintext **P**.
- The private key remains on the receiver's side, and there is no need to share keys in order to perform encryption and decryption.
- The following diagram shows how the receiver uses public key cryptography to verify the integrity of the received message.
- In this model, the sender signs the data using their private key and transmits the message across to the receiver. Once the message is received, it is verified for integrity by the sender's public key.



Model of a public-key cryptography signature scheme

- The preceding diagram shows that the sender digitally signs the plaintext P with their private key using signing function S, and produces data C, which is sent to the receiver, who verifies C using the sender's public key and function V to ensure the message has indeed come from the sender.
- Public key cryptography algorithms are based on various underlying mathematical functions.

The three main categories of asymmetric algorithms are:

Integer factorization

- Integer factorization schemes are based on the fact that large integers are very hard to factor. RSA is a prime example of this type of algorithm.

Discrete logarithm

- A discrete logarithm scheme is based on a problem in modular arithmetic. It is easy to calculate the result of a modulo function, but it is computationally impractical to find the exponent of the generator.
- In other words, it is extremely difficult to find the input from the result (output). This is called a one-way function.
- For example, consider the following equation:

$$3^2 \bmod 10 = 9$$
- This difficult problem is commonly used in the Diffie-Hellman key exchange and digital signature algorithms.

Elliptic curves

- The elliptic curves algorithm is based on the discrete logarithm problem. An elliptic curve is an algebraic cubic curve over a field, which can be defined by an equation, as shown here.

$$y^2 = x^3 + ax + b$$

- The curve is non-singular, which means that it has no cusps or self-intersections. It has two variables a and b, as well as a point of infinity.
- Here, a and b are integers whose values are elements of the field on which the elliptic curve is defined.
- Elliptic curves can be defined over reals, rational numbers, complex numbers, or finite fields. For cryptographic purposes, an elliptic curve over prime finite fields is used instead of real numbers. Additionally, the prime should be greater than 3. Different curves can be generated by varying the values of a and/or b.
- The most prominently used cryptosystems based on elliptic curves are the Elliptic Curve Digital Signatures Algorithm (ECDSA) and the Elliptic Curve Diffie-Hellman (ECDH) key exchange.

Public and private keys

- A private key, as the name suggests, is a randomly generated number that is kept secret and held privately by its users.
- Private keys need to be protected and no unauthorized access should be granted to that key.
- Private keys can be of various lengths, depending on the type and class of algorithms used. For example, in RSA, typically, a key of 1,024 bits or 2,048 bits is used.
- The 1,024-bit key size is no longer considered secure, and at least a 2,048-bit key size is recommended.
- A public key is freely available and published by the private key owner. Anyone who would then like to send the publisher of the public key an encrypted message can do so, by encrypting the message using the published public key and sending it to the holder of the private key.
- No one else is able to decrypt the message because the corresponding private key is held securely by the intended recipient.
- Once the public key-encrypted message is received, the recipient can decrypt the message using the private key. There are a few concerns, however, regarding public keys. These include authenticity and identification of the publisher of the public keys.

RSA

RSA was invented in 1977 by Ron Rivest, Adi Shamir, and Leonard Adelman, hence the name RSA.

This type of public key cryptography is based on the integer factorization problem, where the multiplication of two large prime numbers is easy, but it is difficult to factor the product back to the two original numbers. The crux of the work involved with the RSA algorithm happens during the key generation process. An RSA key pair is generated by performing the following steps:

1. Modulus generation:

Select p and q , which are very large prime numbers

Multiply p and q , $n=p.q$ to generate modulus n

2. Generate the co-prime:

Assume a number called e .

e should satisfy a certain condition; that is, $1 < e < (p-1)(q-1)$ and no number other than 1 can be divided into e and $(p-1)(q-1)$. This is called a co-prime, that is, e is the co-prime of $(p-1)(q-1)$.

3. Generate the public key:

(n, e) is the public key. The public key can be shared with anyone; however, p and q need to be kept secret.

4. Generate the private key:

The private key is called d and is calculated from p , q , and e . The private key is basically the inverse of e modulo $(p-1)(q-1)$.

$$ed = 1 \bmod (p-1)(q-1)$$

p and q should be large enough for the modulus n to become extremely difficult (computationally impractical) to factor.

RSA uses the following equation to produce ciphertext:

$$C = P^e \bmod n$$

Decryption in RSA is provided in the following equation:

$$P = C^d \bmod n$$

Example

For an RSA encryption of a 4-bit message $M = 9$, we choose $p = 3$ and $q = 11$. Find the public and the private keys and show the ciphertext.

Solution:

Find $n = pq = 33$

Select $e = 3$, which is relatively prime to $(p - 1)(q - 1) = 20$

Find d such that, $e * d \bmod (p - 1)(q - 1) = 3 * d \bmod 20 = 1$, we can get $d = 7$

The public key and the private key should be $\{3, 33\}$ and $\{7, 33\}$

To encrypt the message

$$C = M^e \bmod n = 9^3 \bmod 33 = 3$$

The decryption process is the reverse of this action

$$M = C^d \bmod n = 3^7 \bmod 33 = 9$$

Hash functions

- Hash functions are used to create fixed-length digests of arbitrarily long input strings.
- Hash functions are keyless, and they provide a data integrity service.
- They are usually built using iterated and dedicated hash function construction techniques.
- Various families of hash functions are available, such as MD, SHA1, SHA-2, SHA-3, RIPEMD, and Whirlpool.
- Hash functions are commonly used for digital signatures and message authentication codes, such as HMACs.
- They have three security properties, namely, pre-image resistance, second pre-image resistance, and collision resistance.
- Some applications use hash functions as a means for generating PRNGs.

The properties of Hash functions are:**Compression of arbitrary messages into fixed-length digests**

- This property relates to the fact that a hash function must be able to take an input text of any length and output a fixed-length compressed message.
- Hash functions produce a compressed output in various bit sizes, usually between 128-bits and 512-bits.

Easy to compute

- Hash functions are efficient and fast one-way functions. It is required that hash functions be very quick to compute regardless of the message size.
- The efficiency may decrease if the message is too big, but the function should still be fast enough for practical use.

Pre-image resistance

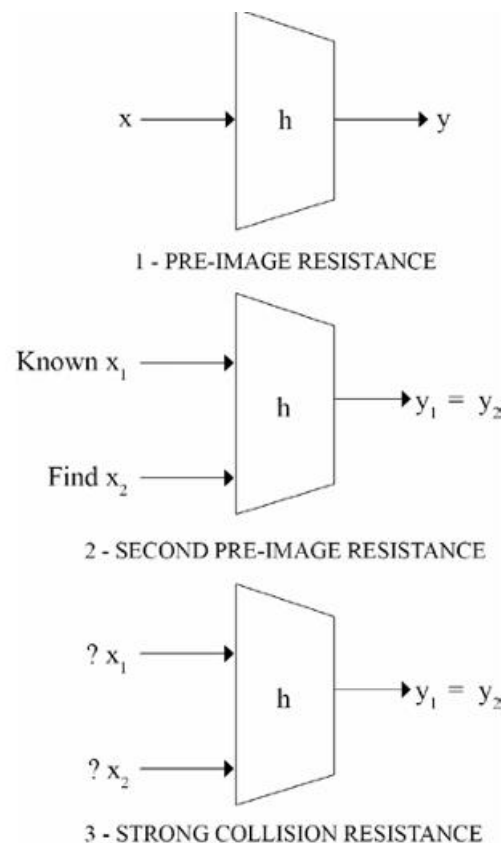
- This property can be explained by using the simple equation: $h(x) = y$ Here, h is the hash function, x is the input, and y is the hash.
- The first security property requires that y cannot be reverse-computed to x .
- x is considered a pre-image of y , hence the name pre image resistance. This is also called a one-way property.

Second pre-image resistance

The second pre-image resistance property requires that given x and $h(x)$, it is almost impossible to find any other message m , where $m \neq x$ and hash of $m =$ hash of x or $h(m) = h(x)$. This property is also known as weak collision resistance.

Collision resistance

- The collision resistance property requires that two different input messages should not hash to the same output.
- In other words, $h(x) \neq h(z)$. This property is also known as strong collision resistance.
- Due to their very nature, hash functions will always have some collisions. This is a situation where two different messages hash to the same output.
- However, they should be computationally impractical to find. A concept known as the avalanche effect is desirable in all cryptographic hash functions.
- The avalanche effect specifies that a small change, even a single character change in the input text, will result in an entirely different hash output.
- Hash functions are usually designed by following an iterated hash functions approach. With this method, the input message is compressed in multiple rounds on a block-by-block basis in order to produce the compressed output.
- A popular type of iterated hash function is the **Merkle-Damgard construction**. This construction is based on the idea of dividing the input data into equal block sizes and then feeding them through the compression functions in an iterative manner.
- The collision resistance of the property of compression functions ensures that the hash output is also collision-resistant.



Three security properties of hash functions

Multiple categories of hash functions:

Message Digest

- Message Digest (MD) functions were prevalent in the early 1990s. MD4 and MD5 fall into this category.
- Both MD functions were found to be insecure and are not recommended for use anymore. MD5 is a 128-bit hash function that was commonly used for file integrity checks

Secure Hash Algorithms

The most common Secure Hash Algorithms (SHAs):

SHA-0:

This is a 160-bit function introduced by the U.S. National Institute of Standards and Technology (NIST) in 1993.

SHA-1:

- SHA-1 was introduced in 1995 by NIST as a replacement for SHA-0. This is also a 160-bit hash function. SHA-1 is used commonly in SSL and TLS implementations.

- SHA-1 is now considered insecure, and it is being deprecated by certificate authorities. Its usage is discouraged in any new implementations.

SHA-2:

This category includes four functions defined by the number of bits of the hash: SHA-224, SHA-256, SHA-384, and SHA-512.

SHA-3:

- This is the latest family of SHA functions. SHA3-224, SHA3-256, SHA3-384, and SHA3-512 are members of this family.
- SHA3 is a NIST-standardized version of Keccak. Keccak uses a new approach called sponge construction instead of the commonly used Merkle Damgard transformation.

RIPEMD:

- RIPEMD is the acronym for RACE Integrity Primitives Evaluation Message Digest. It is based on the design ideas used to build MD4.
- There are multiple versions of RIPEMD, including 128-bit, 160-bit, 256-bit, and 320-bit.

Whirlpool:

This is based on a modified version of the Rijndael cipher known as W. It uses the Miyaguchi-Preneel compression function, which is a type of one-way function used for the compression of two fixed-length inputs into a single fixed-length output. It is a single block length compression function.

Hash functions have many practical applications ranging from simple file integrity checks and password storage to use in cryptographic protocols and algorithms. They are used in hash tables, distributed hash tables, bloom filters, virus fingerprinting, peer-to-peer (P2P) file-sharing, and many other applications. Hash functions play a vital role in blockchain. The Proof-of-Work (PoW) function in particular uses SHA-256 twice in order to verify the computational effort spent by miners. RIPEMD 160 is used to produce Bitcoin addresses.

Design of Secure Hash Algorithms (SHA)

SHA-256 and SHA-3 are used in Bitcoin and Ethereum, respectively.

Design of SHA-256

SHA-256 has an input message size limit of $2^{64} - 1$ bits. The block size is 512 bits, and it has a word size of 32 bits. The output is a 256-bit digest.

The compression function processes a 512-bit message block and a 256-bit intermediate hash value.

There are two main components of this function: the compression function and a message schedule.

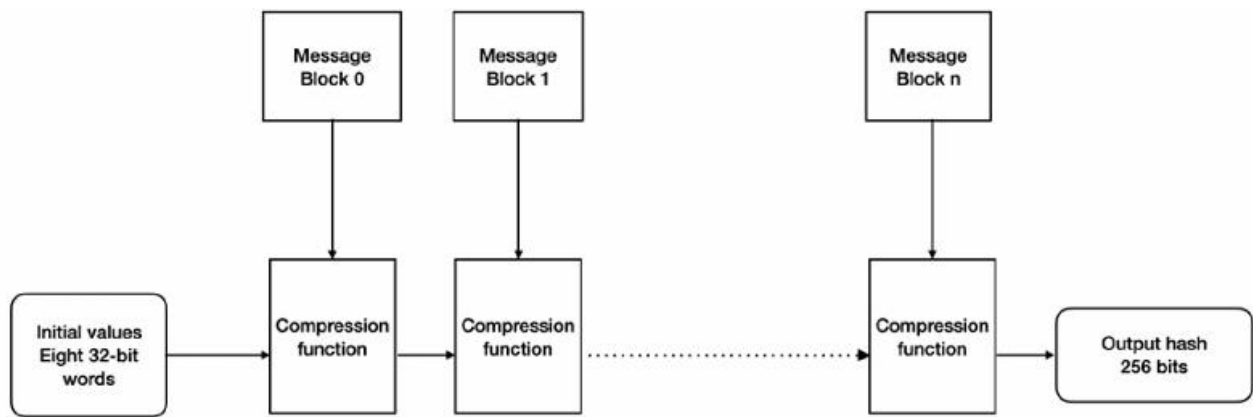
The algorithm works as follows, in nine steps:

Pre-processing

1. Padding of the message is used to adjust the length of a block to 512 bits if it is smaller than the required block size of 512 bits.
2. Parsing the message into message blocks, which ensures that the message and its padding is divided into equal blocks of 512 bits.
3. Setting up the initial hash value, which consists of the eight 32-bit words obtained by taking the first 32 bits of the fractional parts of the square roots of the first eight prime numbers. These initial values are fixed and chosen to initialize the process. They provide a level of confidence that no backdoor exists in the algorithm.

Hash computation

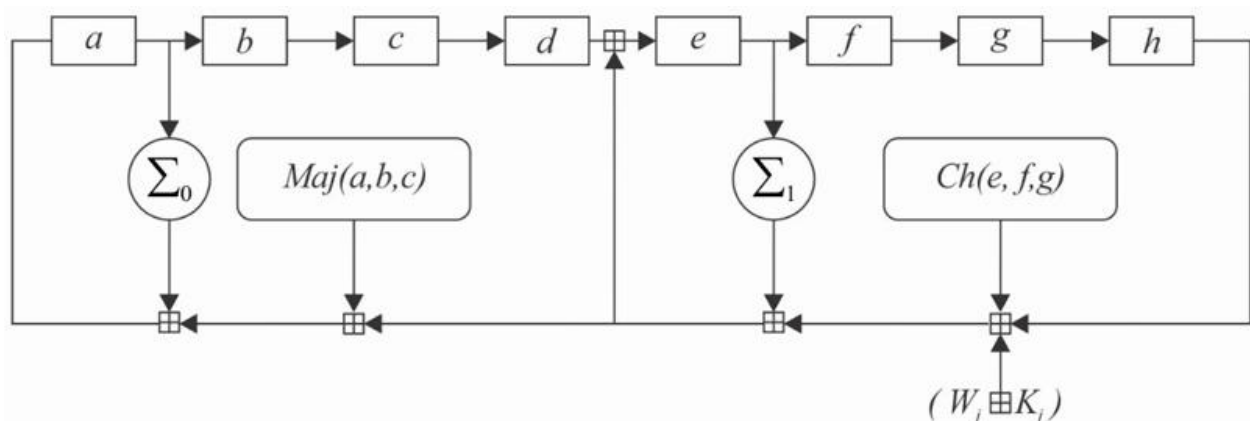
4. Each message block is then processed in a sequence, and it requires 64 rounds to compute the full hash output. Each round uses slightly different constants to ensure that no two rounds are the same.
5. The message schedule is prepared.
6. Eight working variables are initialized.
7. The compression function runs 64 times.
8. The intermediate hash value is calculated.
9. Finally, after repeating steps 5 through 8 until all blocks (chunks of data) in the input message are processed, the output hash is produced by concatenating intermediate hash values.



SHA-256 high level overview

SHA-256 is a Merkle Damgard construction that takes the input message and divides it into equal blocks of 512 bits. Initial values or the initialization vector are composed of eight 32 bit words (256 bits) that are fed into the compression function with the first message. Subsequent blocks are fed into the compression function until all blocks are processed and finally, the output hash is produced.

The compression function of SHA-256 is shown in the following diagram:



One round of an SHA-256 compression function

In the preceding diagram, a, b, c, d, e, f, g, and h are the registers for 8 working variables. Maj and Ch functions are applied bitwise. Σ_0 and Σ_1 perform bitwise rotation. The round constants are W_j and K_j , which are added in the main loop (compressor function) of the hash function, which runs 64 times.

Design of SHA-3 (Keccak)

- The structure of SHA-3 is very different from that of SHA-1 and SHA-2.
- The key idea behind SHA-3 is based on unkeyed permutations, as opposed to other typical hash function constructions that used keyed permutations.
- Keccak also does not make use of the Merkle Damgard transformation that is commonly used to handle arbitrary-length input messages in hash functions.

- A newer approach, called sponge and squeeze construction, is used in Keccak. It is a random permutation model.
- Different variants of SHA-3 have been standardized, such as SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE128, and SHAKE256. SHAKE128 and SHAKE256 are extendable-output functions (XOFs), which allow the output to be extended to any desired length.
- The following diagram shows the sponge and squeeze model, which is the basis of SHA-3 or Keccak.
- Analogous to a sponge, the data (m input data) is first absorbed into the sponge after applying padding.
- It is then changed into a subset of permutation state using XOR (exclusive OR), and then the output is squeezed out of the sponge function that represents the transformed state.
- The rate r is the input block size of the sponge function, while capacity c determines the security level:

The SHA-3 absorbing and squeezing function in SHA3

In the preceding diagram, state size b is calculated by adding bit rate r and capacity bits c . r and c can be any values as long as sizes of $r + c$ are 25, 50, 100, 200, 400, 800, or 1600. The state is a 3-dimensional bit matrix. The initial state is set to 0. The data m is entered into the absorb phase block by block via XOR $\oplus$ after applying padding.

The following table shows the value of bit rate r (block size) and capacity c required to achieve the desired output hash size under the most efficient setting of $r + c = 1600$:

r (block size)	c (capacity)	Output hash size
1152	448	224
1088	512	256
832	768	384
576	1024	512

The function f is a permutation function. It contains five transformation operations named Theta, Rho, Pi, Chi, and Iota described as follows.

θ – Theta: XOR bits in the state, used for mixing

ρ – Rho: Diffusion function performing rotation of bits

π – Pi: Diffusion function

χ – Chi: XOR each bit, bitwise combine

ι – Iota: Combination with round constants

These five operations combined form a round. In the SHA-3 standard, the number of rounds is 24 to achieve the desired level of security.

Elliptic curve cryptography

- ECC is based on the discrete logarithm problem founded upon elliptic curves over finite fields (Galois fields).
- The main benefit of ECC over other types of public key algorithms is that it requires a smaller key size, while providing the same level of security as, for example, RSA.
- Two notable schemes that originate from ECC are ECDH for key exchange and ECDSA for digital signatures.
- ECC can also be used for encryption, but it is not usually used for this purpose in practice. Instead, key exchange and digital signatures are used more commonly. The same level of security can be achieved with ECC when only using 256-bit operands as compared to 3,072 bits in RSA.

An elliptic curve is defined in the following equation:

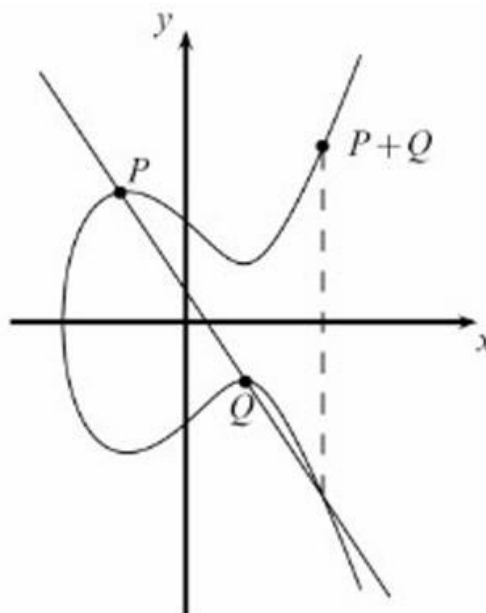
$$y^2 = x^3 + Ax + B \pmod{P}$$

- Here, A and B belong to a finite field Z_p or F_p (prime finite field), along with a special value called the point of infinity. The point of infinity ∞ is used to provide identity operations for points on the curve.
- Furthermore, a condition also needs to be met that ensures that the equation has no repeated roots. This means that the curve is non-singular.

Basic group operations on elliptic curves are point addition and point doubling. Point addition is a process where two different points are added, and point doubling means that the same point is added to itself.

Point addition

- Point addition is shown in the following diagram. This is a geometric representation of point addition on elliptic curves. In this method, a diagonal line is drawn through the curve that intersects the curve at two points shown below P and Q , which yields a third point between the curve and the line.
- This point is mirrored as $P+Q$, which represents the result of the addition as R . This is shown as $P+Q$ in the following diagram:



Point addition over R

In this case, two points are added to compute the coordinates of the third point on the curve: $P + Q = R$

This means that coordinates are added, as shown in the following equation:

$$(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$$

The equation of point addition is as follows:

$$X_3 = s^2 - x_1 - x_2 \bmod p$$

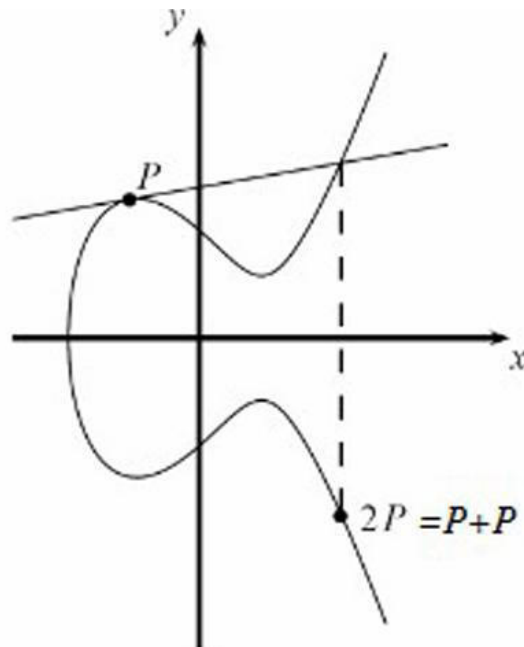
$$y_3 = s(x_1 - x_3) - y_1 \bmod p$$

where,

$$s = \frac{(y_2 - y_1)}{(x_2 - x_1)} \bmod p$$

Point doubling

- The other group operation on elliptic curves is called point doubling. This is a process where P is added to itself.
- In this method, a tangent line is drawn through the curve, as shown in the following graph. The second point is obtained, which is at the intersection of the tangent line drawn and the curve. This point is then mirrored to yield the result, which is shown as $2P = P + P$



Graph representing point doubling

The equation of point doubling is as follows:

$$X_3 = s^2 - 2x_1 \bmod p$$

$$y_3 = s(x_1 - x_3) - y_1 \bmod p$$

Point multiplication

- This operation is also called scalar point multiplication. It is used to multiply points on elliptic curves by a given integer.
- Let this integer be d and the point P . We get dP by repeatedly adding P , d times. This operation can be described as follows:
- $P + P + \dots + P = dP$, where P is added d times.
- Any point on the curve can be added multiple times to itself. The result of this operation is always another point on the curve.
- The addition operation is not efficient for large values of d . However, point multiplication operation can be made by combining point addition and doubling operations to achieve exponential performance gain.
- Consider that dP results in producing another point on the curve; (i.e) point T . We can say that with all these doublings and additions we have computed a point called T .
- Even if we know points P and T , it is computationally infeasible to reconstruct the sequence of all the double and addition operations that we did to figure out the number d . This means that it is a one-way function, and it is the basis of the Elliptic Curve Discrete Logarithm Problem (ECDLP).

The discrete logarithm problem in ECC

The discrete logarithm problem in ECC is based on the idea that, under certain conditions, all points on an elliptic curve form a cyclic group.

On an elliptic curve, a private key is a randomly selected integer, whereas the public key is a point on the curve. The discrete logarithm problem is used to find the private key (an integer) where that integer falls within all points on the elliptic curve.

Consider an elliptic curve E , with two elements P and T . The discrete logarithmic problem is to find the integer d , where $1 \leq d \leq \#E$, such that:

$$P + P + \dots + P = dP = T$$

Here, T is the public key (a point on the curve, (x, y)), and d is the private key.

$\#E$ represents the order of the elliptic curve, which means the number of points that are present in the cyclic group of the elliptic curve. A cyclic group is formed by a combination of points on the elliptic curve and the point of infinity.

The initial starting point P is a public parameter, and the public key T is also published, whereas d , the private key, is kept secret.

A key pair is linked with the specific domain parameters of an elliptic curve. Domain parameters include field size, field representation, two elements from the field a

and b , two field elements X_g and Y_g , order n of point G , which is calculated as $G = (X_g, Y_g)$, and the cofactor $h = \#E(F_q)/n$.

- P is the prime p that specifies the size of the finite field.
- a and b are the coefficients of the elliptic curve equation.
- G is the base point that generates the required subgroup, also known as the generator.
- n is the order of the subgroup.
- h is the cofactor of the subgroup.

Digital signatures

- Digital signatures provide a means of associating a message with an entity from which the message has originated. Digital signatures are used to provide data origin authentication and non-repudiation.
- Digital signatures are used in blockchains, where transactions are digitally signed by senders using their private key.
- This digital signing proves that the sender is the rightful owner of the asset; for example, bitcoins.
- These transactions are verified again by other nodes on the network to ensure that the funds indeed belong to the node (user) who claims to be the owner.

RSA digital signature algorithms

RSA-based digital signature algorithms are calculated using the two steps.

1. Calculate the hash value of the data packet. This will provide the data integrity guarantee, as the hash can be computed at the receiver's end again and matched with the original hash to check whether the data has been modified in transit.
2. Sign the hash value with the signer's private key. As only the signer has the private key, the authenticity of the signature and the signed data is ensured.

Digital signatures have some important properties:

1. **Authenticity** means that the digital signatures are verifiable by a receiving party.
2. The **unforgeability** property ensures that no one else can produce the signed message produced by a legitimate sender. Digital signatures must provide protection against forgery. Forgery means an adversary fabricating a valid signature for a message without any access to the legitimate signer's private key. This is also called the property of non-repudiation.

3. **Non-reusability** means that the digital signature cannot be separated from a message and used again for another message. In other words, the digital signature is firmly bound to the corresponding message and cannot be simply cut from its original message and attached to another.

Digital signing (left) and verification process (right)

If a sender wants to send an authenticated message to a receiver, there are two methods that can be used: sign then encrypt and encrypt then sign.

Sign then encrypt

With this approach, the sender digitally signs the data using the private key, appends the signature to the data, and then encrypts the data and the digital signature using the receiver's public key. This is considered a more secure scheme compared to the encrypt then sign scheme.

Encrypt then sign

With this method, the sender encrypts the data using the receiver's public key and then digitally signs the encrypted data.

- Various schemes, such as RSA-, DSA-, and ECDSA-based digital signature schemes, are used in practice. RSA is the most commonly used; however, ECDSA based schemes are also becoming quite popular.

- This is beneficial in blockchains because ECC provides the same level of security that RSA does, but it uses less space.
- Also, the generation of keys is much faster in ECC compared to RSA, so it helps with the overall performance of the system. The following table shows that ECC is able to provide the same level of cryptographic strength as an RSA-based system with smaller key sizes:

RSA key size (bits)	Elliptic curve key size (bits)
1,024	160
2,048	224
3,072	256
7,680	384
15,360	521

The elliptic curve digital signature algorithm

- ECDSA is a DSA based on elliptic curves. The DSA is a standard for digital signatures.
- It is based on modular exponentiation and the discrete logarithm problem.
- It is used on the Bitcoin and Ethereum blockchain platforms to validate messages and provide data integrity services.

In order to sign and verify using the ECDSA scheme, the first key pair needs to be generated:

1. First, define an elliptic curve E with the following:

Modulus P

Coefficients a and b

Generator point A that forms a cyclic group of prime order q

2. An integer d is chosen randomly so that $0 < d < q$.
3. Calculate public key B so that $B = d A$

The public key is in the form:

$$K_{pb} = (p, a, b, q, A, B)$$

The private key is a randomly chosen integer d in step 2:

$$K_{pr} = d$$

Now, the signature can be generated using the private and public key.

4. An ephemeral key K_e is chosen, where $0 < K_e < q$. It should be ensured that K_e is truly random and that no two signatures have the same key; otherwise, the private key can be calculated.
5. The value R is calculated using $R = K_e A$; that is, by multiplying A (the generator point) and the random ephemeral key.
6. Initialize a variable r with the x coordinate value of point R so that $r = xR$.
7. The signature can be calculated as follows:

$$S = (h(m) + dr)K_e^{-1} \bmod q$$

Here, m is the message for which the signature is being computed, and $h(m)$ is the hash of the message m .

8. Signature verification is carried out by following this process:

Auxiliary value w is calculated as $w = s^{-1} \bmod q$

Auxiliary value $u_1 = w \cdot h(m) \bmod q$

Auxiliary value $u_2 = w \cdot r \bmod q$

Calculate point P , $P = u_1A + u_2B$

9. Verification is carried out as follows:

r, s is accepted as a valid signature if the x -coordinate of point P calculated in Step 4 has the same value as the signature parameter $r \bmod q$; that is:

$X_p = r \bmod q$ means valid signature

$X_p \neq r \bmod q$ means invalid signature

Applications of cryptographic hash functions

There are various constructs that have been built using basic cryptographic parameters to solve different problems in computing. These constructs are also used in blockchains to provide various protocol-specific services.

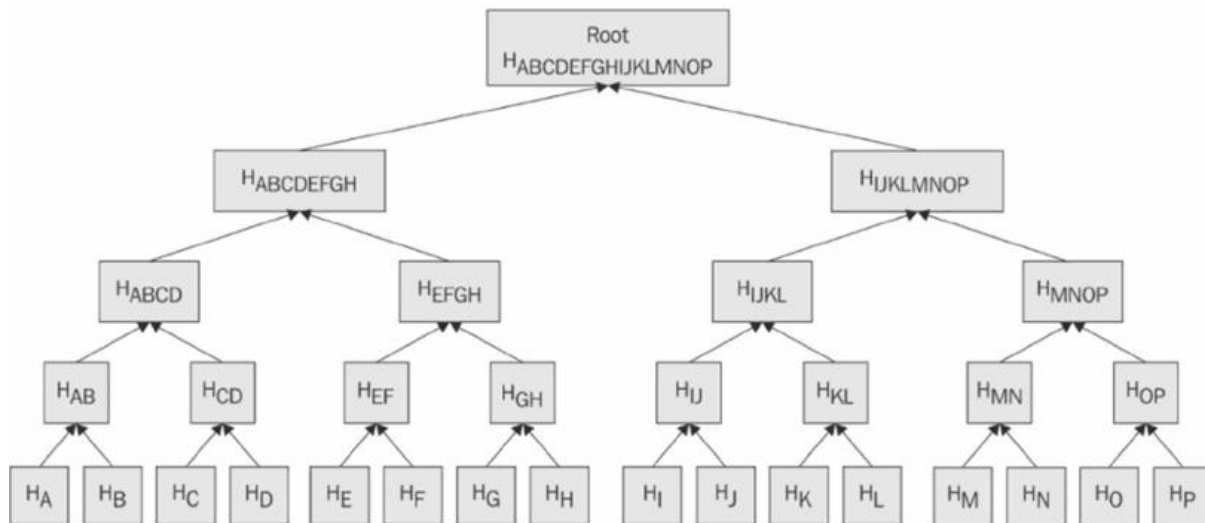
For example, hash functions are used to build Merkle trees, which are used to efficiently and securely verify large amounts of data in distributed systems.

Other applications:

- Hash functions are used in cryptographic puzzles such as the Proof of Work (PoW) mechanism in Bitcoin. Bitcoin's PoW makes use of the SHA-256 cryptographic hash function.
- The generation of addresses in blockchains. For example, in Ethereum, blockchain accounts are represented as addresses. These addresses are obtained by hashing the public key with the Keccak-256 hash algorithm.
- Message digests in digital signatures.
- The creation of Merkle trees to guarantee the integrity of transaction structure in the blockchain. Specifically, this structure is used to quickly verify whether a transaction is included in a block or not.

Merkle trees

The concept of Merkle trees was introduced by Ralph Merkle. Merkle trees enable the secure and efficient verification of large datasets:



A Merkle tree

- A Merkle tree is a binary tree in which the inputs are first placed at the leaves (nodes with no children), and then the values of pairs of child nodes are hashed together to produce a value for the parent node (internal node), until a single hash value known as a Merkle root is achieved.
- This structure helps to quickly verify the integrity of the entire tree (entire dataset), but just by verifying the Merkle root on top of the Merkle tree, because if any change occurs in any of the hashes in the tree, the Merkle root will also change.
- This is the reason why the integrity of the system can be verified quickly by just looking at the Merkle root.
- Another advantage of Merkle trees is that there is no requirement of storing large amounts of data, only the hashes of the data, which are fixed-length digests of the large dataset.
- Due to this property, the storage and management of Merkle trees is easy and efficient as it takes a very small amount of space for storage.
- Also, due to the fact that the tree is storage efficient, the relevant proofs for integrity are also smaller in size and quick to transmit over the network, thus making them bandwidth efficient over the network.

Patricia trees

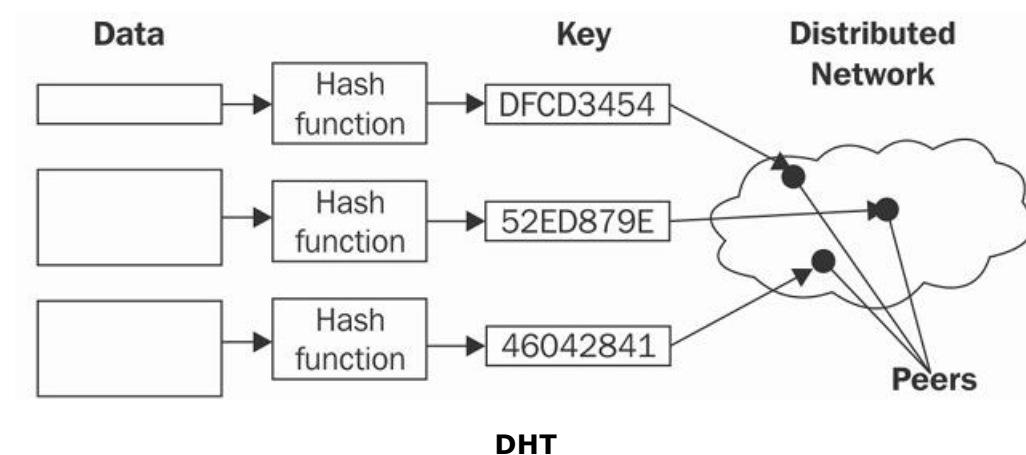
- To understand Patricia trees, we must know the concept of a trie. A trie, or a digital tree, is an ordered tree data structure used to store a dataset.
- The Practical Algorithm to Retrieve Information Coded in Alphanumeric (Patricia) tree, also known as a Radix tree, is a compact representation of a

trie in which a node that is the only child of a parent is merged with its parent.

- A Merkle-Patricia tree, based on the definitions of Patricia and Merkle, is a tree that has a root node that contains the hash value of the entire data structure. Merkle-Patricia trees are used in the Ethereum blockchain.

Distributed hash tables

- A hash table is a data structure that is used to map keys to values. Internally, a hash function is used to calculate an index into an array of buckets from which the required value can be found.
- Buckets have records stored in them using a hash key and are organized into a particular order.
- Distributed Hash Table (DHT) is a data structure where data is spread across various nodes, and nodes are equivalent to buckets in a peer-to-peer network.



- In the preceding diagram, data is passed through a hash function, which then generates a compact key. This key is then linked with the data (values) on the peer-to-peer network.
- When users on the network request the data (via the filename), the filename can be hashed again to produce the same key, and any node on the network can then be requested to find the corresponding data.
- A DHT provides decentralization, fault tolerance, and scalability.