UNIT – 2 : BASICS

The Double-Spend Problem:

Although Blockchain is secured, still it has some loopholes. Hackers or malicious users take advantage of these loopholes to perform their activities.

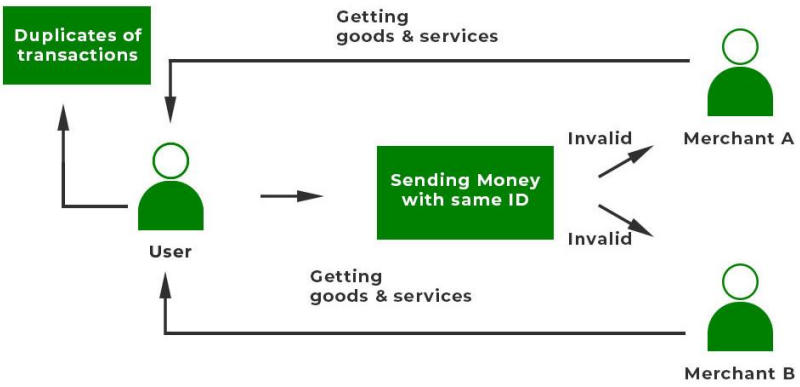
- **Double spending** means the expenditure of the same digital currency twice or more to avail the multiple services. It is a technical flaw that allows users to duplicate money.
- Since digital currencies are nothing but files, a malicious user can create multiple copies of the same currency file and can use it in multiple places.
- This issue can also occur if there is an alteration in the network or copies of the currency are only used and not the original one.
- There are also double spends that allow hackers to reverse transactions so that transaction happens two times.
- By doing this, the user loses money two times one for the fake block created by the hacker and for the original block as well.
- The hacker gets incentives as well for the fake blocks that have been mined and confirmed.

How Does Double Spending Happen?

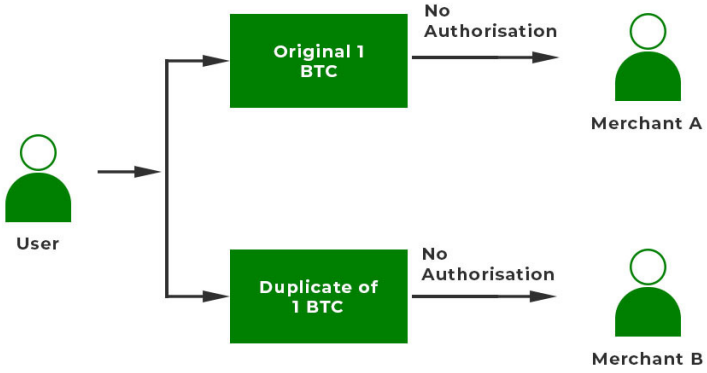
Double spending can never arise physically. It can happen in online transactions. This mostly occurs when there is no authority to verify the transaction. It can also happen if the user's wallet is not secured. Suppose a user wants to avail of services from Merchant 'A' and Merchant 'B'.

- The user first made a digital transaction with Merchant 'A'.
- The copy of the cryptocurrency is stored on the user's computer.
- So, the user uses the same cryptocurrency to pay Merchant 'B'
- Now both the merchants have the illusion that the money has been credited since the transactions were not confirmed by the miners.

This is the case of double spending.



Example: Suppose a user has 1 BTC. He/She wants to avail of services from merchant A and merchant B. The user creates multiple copies of the same BTC and stores it. The user first sends the original BTC to Merchant A and gets the service. Simultaneously, the user sends the copied version of 1 BTC to Merchant B. Since the second transaction was not confirmed by other miners, the merchant accepts the bitcoin and sends the service. But the cryptocurrency that was sent is invalid. This is the case of Double Spending.



Types Of Double Spending Attacks:

There are different types of Double Spending attacks:

- **Finney Attack:** Finney Attack is a type of Double spending Attack. In this, a merchant accepts an unauthorized transaction. The original block is eclipsed by the hacker using an eclipse attack. The transaction is performed on an unauthorized one. After that, the real block shows up and again the transaction is done automatically for the real block. Thus, the merchant loses money two times.
- **Race attack:** is an attack in which there is a 'race' between two transactions. The attacker sends the same money using different machines to two different merchants. The merchants send their goods but transactions get invalid.
- **51% Attack:** This type of attack is prevalent in small blockchains. Hackers usually take over 51% of the mining power of blockchain and therefore can do anything of their own will.

Solutions To Prevent Double Spending:

Double Spending can be prevented using two approaches: Centralized and Decentralized

- **Centralized Approach:** In this case, a secured third party is employed to verify the transactions. The third-party can track each of the user's balances. Suppose a user makes a transaction. The third-party identifies the transaction with a unique identity. Then it verifies the transaction and allows the transaction. The problem is that suppose we want to make transactions with other countries where a third party is not required. So, in such cases, decentralized systems come into play. another drawback is if the whole system fails, the users cannot have access.
- **Decentralized Approach:** This approach is used by Bitcoin. In this, there is no involvement of central authority. Each transaction is verified using powerful algorithms. The decentralized approach proved to be more secure than the centralized approach. Protocols are established and each protocol does its job at each step. Therefore, this also promotes transparency.

How to Combat Double Spending?

Double spending has been minimized to a large extent as companies are using many security features. But we as users also have some responsibility so that such attacks don't happen.

- Any user should wait for a minimum of six confirmations of the transaction before performing another transaction. In the blockchain, more the confirmations by different users, lesser will be double spending attacks.
- Users should keep their hardware resources safe so that hackers do not misuse them for their own purposes. Often hackers target the hardware part because the hardware is costly. If they somehow steal the hardware, they can roll back any transaction or alter information.
- Users should delete spam mails and avoid phishing to avoid unnecessary malware attacks Phishing is a very common attack by hackers as hackers target login credentials.
- Software should be updated regularly with the latest antivirus installed. If the software is not up to date, then the bugs present can cause major damage.

How Successful Double Spending is Administered?

With the increasing dependency on the blockchain, double spending attacks have also become a major problem. Many companies have adopted security features.

- Features like confirmation of the transaction by the nodes have been adopted. A minimum of six confirmations is required to approve the transaction.
- The blocks once created are immutable. They are made irreversible so that no transaction is reverted back.
- The network's distributed ledger of transactions autonomously records each transaction. Each node has a copy of all transactions that are being done in the network.
- Verification of each transaction's authenticity is done by Blockchain protocols to prevent double-spending. The concept of hashing is adopted. Here each block has a unique hash.

Disadvantages of Blockchain Concerning Double Spending:

There are many disadvantages of blockchain concerning Double Spending:

- **Control of the blockchain:** The biggest disadvantage is if the hackers manage to take control of 51% computation power, they can do any transaction of their own will and can steal other users' money. Therefore, there is a threat to security as millions and millions of money are involved in transactions.
- **Alteration of information:** Transaction information can also be altered by hackers. They can mine blocks and hide the original blocks using attacks like Eclipse attack, Finney Attack, etc.
- **No authority:** The third major problem is no central authority is present to verify the transactions. But these problems will be eliminated if companies take proper security measures and users are also aware of the measures.

Byzantine Generals' Computing Problems:

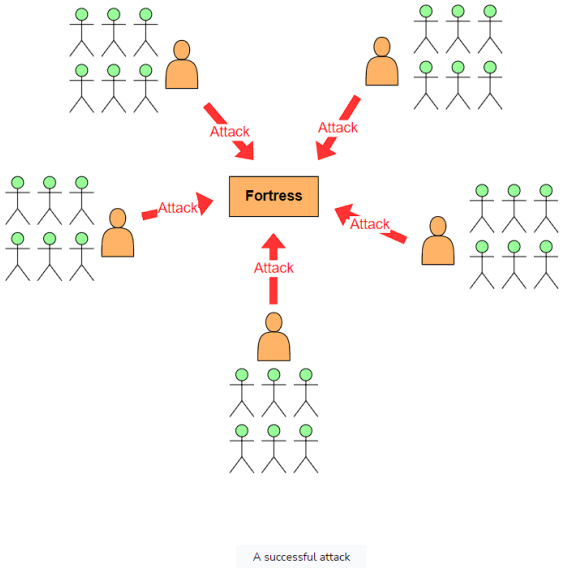
In 1982, The Byzantine General's Problem was invented by Leslie Lamport, Robert Shostak, and Marshall Pease. Byzantine Generals Problem is an impossibility result which means that the solution to this problem has not been found yet as well as helps us to understand the importance of blockchain. It is basically a game theory problem that provides a description of the extent to which decentralized parties experience difficulties in reaching consensus without any trusted central parties.

A group of generals attacks a fortress; every general has an army and surrounds a fort from one side. Every general has a preference about whether to attack or retreat. It has to be a coordinated attack or retreat to incur minimum losses. Thus, a consensus is held, and the majority decision is implemented. This consensus is formed after following the following steps:

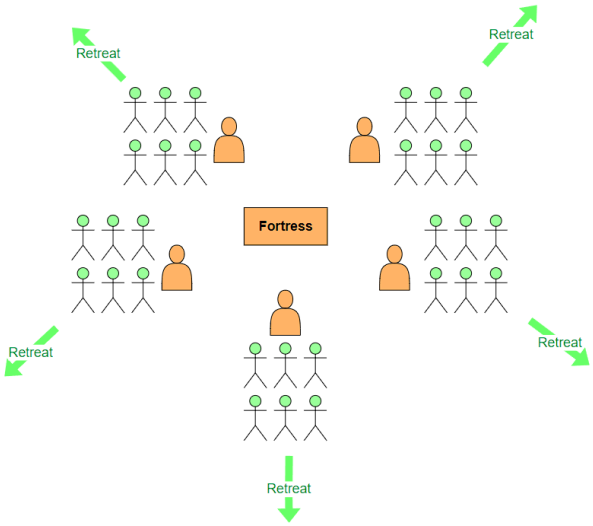
1. Every general sends their own choice to all other generals.
2. After receiving the choice of all generals, every general calculates the votes in favor of attacking and retreating.
3. If the majority is in favor of retreat, then they retreat; otherwise, they attack.

Suppose there is a traitor general who sends a retreat message to half generals and an attack message to the other half generals. Then half of the generals may end up attacking while the other half will retreat, causing the army to lose.

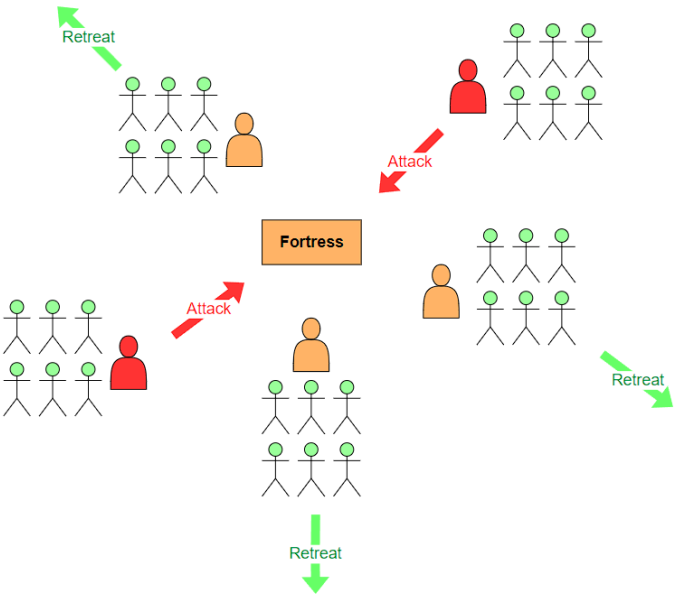
The following slides show all the scenarios of a successful attack, a successful retreat, and an unsuccessful attack.



A successful attack



A successful retreat



An unsuccessful attack

Money and Byzantine General's Problem:

Money is one such commodity whose value should be same throughout the society, that is everyone should agree upon the value of a certain amount of money, despite all the differences therefore in the initial times, precious metals and rare goods were chosen as money because their value was seen equally throughout the society, but in some cases such as precious metals the purity of the metals could not be known for sure or checking the purity was an extremely tedious task which turned out to be very inefficient for the daily transactions, therefore it was decided upon to replace gold with a central party which would be highly trustable chosen by the people in the society to establish and maintain the system of money. But with time it was later realized that those central parties, how much-ever qualified were still not completely trustworthy as it was so simple for them to manipulate the data.

- Centralized systems do not address the Byzantine Generals problem, which requires that truth be verified in an explicitly transparent way, yet centralized systems give no transparency, increasing the likelihood of data corruption.
- They forgo transparency in order to attain efficiency easily and prefer to avoid dealing with the issue entirely.
- The fundamental issue of centralized systems, however, is that they are open to corruption by the central authority, which implies that the data can be manipulated by anyone who has control of the database itself because the centralized system concentrates all power on one central decision maker.

Therefore, Bitcoin was invented to make the system of money decentralized using blockchain to make money verifiable, counterfeit-resistant, trustless, and separate from a central agency.

How Bitcoin Solves the Byzantine General's Problem?

In the Byzantine Generals Problem, the untampered agreement that all the loyal generals need to agree to is the blockchain. Blockchain is a public, distributed ledger that contains the records of all transactions. If all users of the Bitcoin network, known as nodes, could agree on which transactions occurred and in what order, they could verify the ownership and create a functioning, trustless money system without the need for a centralized authority. Due to its decentralized nature, blockchain relies heavily on a consensus technique to validate transactions. It is a peer-to-peer network that offers its users transparency as well as trust. Its distributed ledger is what sets it apart from other systems. Blockchain technology can be applied to any system that requires proper verification.

Proof Of Work: The network would have to be **provable, counterfeit-resistant, and trust-free** in order to solve the Byzantine General's Problem. Bitcoin overcame the Byzantine General's Problem by employing a Proof-of-Work technique to create a clear, objective regulation for the blockchain. Proof of work (PoW) is a method of adding fresh blocks of transactions to the blockchain of a cryptocurrency. In this scenario, the task consists of creating a hash (a long string of characters) that matches the desired hash for the current block.

1. **Counterfeit Resistant:** Proof-of-Work requires network participants to present proof of their work in the form of a valid hash in order for their block, i.e. piece of information, to be regarded as valid. Proof-of-Work requires miners to expend significant amounts of energy and money in order to generate blocks, encouraging them to broadcast accurate information and so protecting the network. Proof-of-Work is one of the only ways for a decentralized network to agree on a single source of truth, which is essential for a monetary system. There can be no disagreement or tampering with the information on the blockchain network because the rules are objective. The ruleset defining which transactions are valid and which are invalid, as well as the system for choosing who can mint new bitcoin, are both objectives.
2. **Provable:** Once a block is uploaded to the blockchain, it is incredibly difficult to erase, rendering Bitcoin's history immutable. As a result, participants of the blockchain network may always agree on the state of the blockchain and all transactions inside it. Each node independently verifies whether blocks satisfy the Proof-of-Work criterion and whether transactions satisfy additional requirements.
3. **Trust-free:** If any network member attempts to broadcast misleading information, all network nodes immediately detect it as objectively invalid and ignore it. Because each node on the Bitcoin network can verify every information on the network, there is no need to trust other network members, making Bitcoin a trustless system.

Byzantine Fault Tolerance (BFT):

The Byzantine Fault Tolerance was developed as inspiration in order to address the Byzantine General's Problem. The Byzantine General's Problem, a logical thought experiment where multiple generals must attack a city, is where the idea for BFT originated.

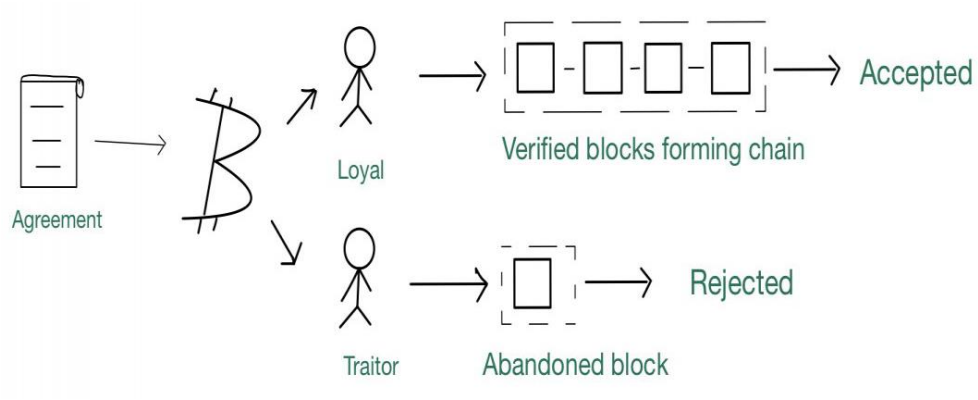
- Byzantine Fault Tolerance is one of the core characteristics of developing trustworthy blockchain rules or features is tolerance.
- When two-thirds of the network can agree or reach a consensus and the system still continues to operate properly, it is said to have BFT.
- Blockchain networks' most popular consensus protocols, such as proof-of-work, proof-of-stake, and proof-of-authority, all have some BFT characteristics.

- In order to create a decentralized network, the BFT is essential.

The consensus method determines the precise network structure. For instance, BFT has a leader as well as peers who can and cannot validate.

In order to maintain the sequence of the Blockchain SC transactions and the consistency of the global state through local transaction replay, consensus messages must pass between the relevant peers.

More inventive approaches to designing BFT systems will be found and put into practice as more individuals and companies investigate distributed and decentralized systems. Systems that use BFT are also employed in sectors outside of blockchains, such as nuclear power, space exploration, and aviation.



Byzantine General's Problem in a Distributed System:

In order to address this issue, honest nodes (such as computers or other physical devices) must be able to establish an agreement in the presence of dishonest nodes.

- In the Byzantine agreement issue, an arbitrary processor initializes a single value that must be agreed upon, and all nonfaulty processes must agree on that value. Every processor has its own beginning value in the consensus issue, and all nonfaulty processors must agree on a single common value.
- The Byzantine army's position can be seen in computer networks.
- The divisions can be viewed as computer nodes in the network, and the commanders as programs running a ledger that records transactions and events in the order that they occur. The ledgers are the same for all systems, and if any of them is changed, the other ledgers are updated as well if the changes are shown to be true, so all distributed ledgers should be in agreement.

Byzantine General's Problem Example:

A basic Byzantine fault is a digital signal that is stuck at "1/2," i.e. a voltage that is anywhere between the voltages for a valid logical "0" and a valid logical "1." Because these voltages are in the region of a gate's transfer function's maximum gain, little quantities of noise on the gate's input become enormous amounts of noise on the gate's output. This is due to the fact that "digital circuits are simply analog circuitry driven to extremes."

This problem is solvable because, with a dominating input, even a Byzantine input has no output impact.

- An excellent composite example is the well-known 3-input majority logic "voter."
- If one of the inputs is "1/2" and the other two are both 0 or both 1, the result is 0 or 1 (due to masking within the voter).
- When one of the inputs is "1/2" and the other two are different values, the output can be 0, "1/2," or 1, depending on the precise gain and threshold voltages of the voter gates and the properties of the "1/2" signal.

Public-Key Cryptography (PKC):

It is also known as **asymmetric cryptography**. It is an encryption technique or a framework that uses a pair of keys (public and private key) for secure data communication.

These keys are related, but not identical keys. Each key performs a unique function, i.e., the public key is used to encrypt, and the private key is used to decrypt. The sender uses the recipient's public key to encrypt a message, and the recipient uses the private key to decrypt this message. The use of two keys enables PKC to solve challenges faced in other cryptographic techniques.

Public-Key Cryptography is different from the symmetric key algorithm, which uses only one key to both encrypt and decrypt.

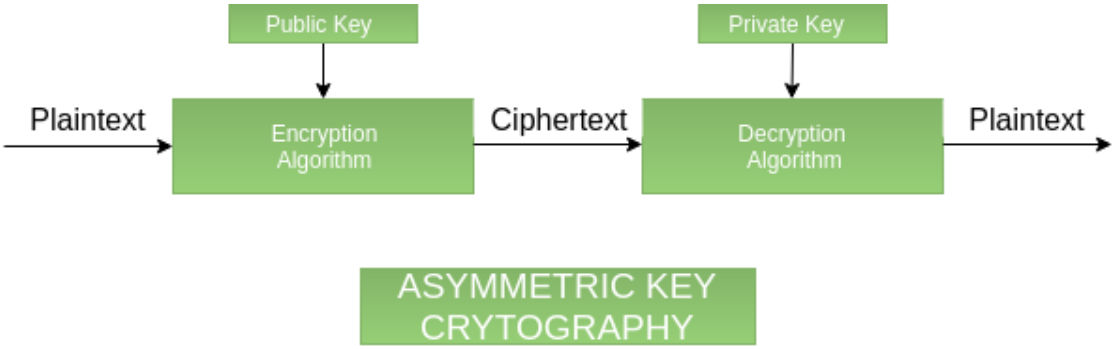
The two types of Public-Key Cryptography algorithms are RSA (Rivest, Shamir, and Adelman) and Digital Signature Algorithm (DSA).

Public-Key Cryptography encryption evolved to meet the growing need for secure communication in multiple sectors such as the military, government offices, etc. This type of cryptography has become an important element of modern computer security and a critical component of the cryptocurrency system.

How Does Public-Key Cryptography Work?

The public key is used by the sender to encrypt information, whereas the private key is used by a recipient to decrypt it. The public key can be shared without compromising the security of the private one. All asymmetric key pairs are unique, so a message encrypted with a public key can only be read by the person who has the corresponding private key.

The keys of a pair are mathematically related, and their length is much longer than those used in symmetric cryptography. So, it is not easy to decipher the private key from its pubic counterpart. RSA is one of the most common algorithms for asymmetric encryption in use today.



Benefits of Public-Key Cryptography:

- One key cannot be derived from another key, and there is no need to exchange the keys
- It allows to establish authentication of the sender by using PKC (digital signature)
- It can be used to create a digital signature in the Operating System software such as Ubuntu, Red Hat Linux packages distribution, etc.

Applications of Public-Key Cryptography:

- Emails can be encrypted using public-key cryptography to keep their content confidential
- Secure socket layer (SSL) protocol also uses asymmetric cryptography to make secure connections to websites
- It is also used in blockchain and cryptography technology. For example, while setting up a new cryptocurrency wallet, a pair of keys is generated.

Difference Between Private Key and Public Key Cryptography:

Feature	Private Key Cryptography (Symmetric)	Public Key Cryptography (Asymmetric)
Key Usage	Same key for encryption and decryption	Different keys for encryption and decryption
Key Sharing	Key must be securely shared between parties	Public key can be openly shared; private key is kept secret
Speed	Faster due to simpler algorithms	Slower due to complex mathematical operations
Security	Security depends on keeping the key secret	Security relies on the computational difficulty of key reversal
Key Management	Difficult to manage keys for large numbers of users	Easier key management as only the private key needs to be kept secret
Scalability	Less scalable for many users	More scalable, ideal for environments with numerous users
Typical Uses	Bulk data encryption, secure communication between known parties	Secure communication, digital signatures, key exchange, authentication
Examples of Algorithms	AES (Advanced Encryption Standard), DES (Data Encryption Standard)	RSA (Rivest-Shamir-Adleman), ECC (Elliptic Curve Cryptography), Diffie-Hellman
Resource Requirements	Lower computational requirements	Higher computational requirements
Encryption/Decryption	Same key used for both processes	Public key for encryption, private key for decryption
Digital Signatures	Not used for digital signatures	Supports digital signatures for authentication and integrity verification

Hashing:

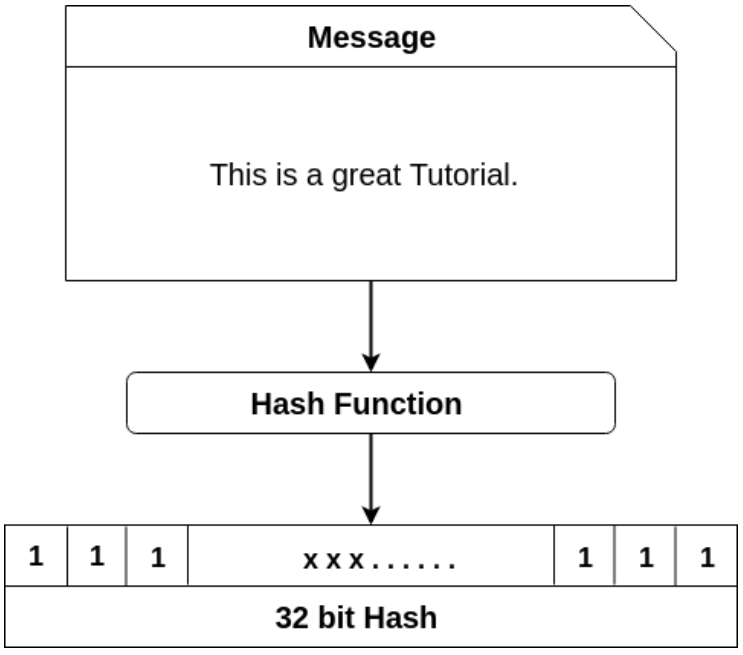
Hashing is a process used to convert an input (or 'message') into a fixed-size string of bytes. This output, typically a sequence of seemingly random characters, is called the **hash value or hash code.**

If we take the example of blockchain use in online transactions (using bitcoins), transactions of different lengths are run through a given hashing algorithm, and all give an output that is of a fixed length. This output is independent of the length of the input transaction.

- **Secure Hashing Algorithm 256 (SHA-256):** This hashing algorithm always gives an output of fixed length 256-bits or 32 bytes, no matter whatever is the size of the input transaction. It means if we hash two different input using SHA-256, let's say one is a movie of 1 gigabyte and another is an image of 5 kilobytes, then in both cases, the output hash will be 256-bits in length. The only difference between the two will be the hash pattern. Currently, this algorithm is used in the Bitcoin network.
- **Keccak-256:** This hashing algorithm always gives an output of fixed length 256-bit; currently it is used in the Ethereum network.

Hash Functions:

Basically, the process of using a given hash function to produce a transaction is called hashing. A hash function, will take any transaction or data input and rehash it to produce an output of a fixed size. The transaction output of that given hash function is what we call a hash.



This complete process is known as Hashing. It consists of two subparts; one is encryption of data (the process of generating a hash from input data) and the second one is the decryption of data (the process of generating output from hash using a cryptographic hash function).

Properties of Hash Function:

Three main properties of cryptographic hash functions are:

- Its Input can be any string of any size.
- It produces a fixed size output. We had already seen the example of SHA-256 regarding this property.
- And the third one is, It is efficiently Computable**,** this means that for a given input string, we can figure out what the output of the hash function is in a reasonable amount of time. If I talk more technically, computing the hash of an n-bit string has a running time that is O(N).

For a hash function to be strong and more secure, it has the following three additional properties:

- **Collision-resistance:** A collision occurs when two distinct inputs produce the same output. A hash function H(x) is collision-resistant if nobody can find a collision. It means for two values x and y, such that $x \neq y$, if h(x) =H(y), it means the function h(.) is collision-resistant. We want to increase data security then it's very important to use a strong hash function that is not collision-resistant.
- **Hiding:** This property of hash functions ensures that if we're given the output of hash function y=H(x), then there's no feasible way to figure out what the input, x, was. In a more technical and mathematical way "A hash function H is hiding if: when a secret value r is chosen from a probability distribution that has high min-entropy, then given H(r // x) it is infeasible to find x.
- **Puzzle friendliness:** This property of the hash function is a bit complicated to understand, but I will try to make you understand in simple words. According to this property, if someone wants to target the hash function to come out to some particular output value y, that if there's part of the input that is chosen in a suitably randomized way, it's very difficult to find another value that hits exactly that target. In a more technical way, this property states that " A hash function H is said to be a puzzle – friendly if for every possible n-bit output value y, if k is chosen from a distribution with high min-entropy, then it is infeasible to find x such that H(k // x)=y in time significantly less than 2^n.

Applications of Hashing:

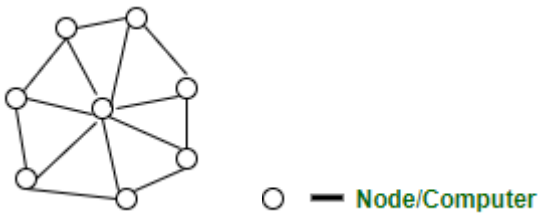
1. Signature generation and verification: Almost all digital signature schemes require a cryptographic hash to be calculated over the message. The message is considered authentic if the signature verification succeeds given the signature and recalculated hash digest over the message. So, the message integrity property of the cryptographic hash is used to create secure and efficient digital signature schemes.

2. Password verification: To authenticate a user, the password presented by the user is hashed and compared with the stored hash. A password hashing is performed; the original password cannot be recalculated from the stored hash value.

3. Verification of files and messages: An important application of secure hashes is the verification of **message integrity.** Comparing messages digests (hash digest over the message) calculated before, and after, transmission can determine whether any changes have been made to the message or file.

Distributed Systems:

A distributed system is a collection of computer programs that utilize computational resources across multiple, separate computation nodes to achieve a common, shared goal. Also known as distributed computing or distributed databases, it relies on separate nodes to communicate and synchronize over a common network. These nodes typically represent separate physical hardware devices but can also represent separate software processes, or other recursive encapsulated systems. Distributed systems aim to remove bottlenecks or central points of failure from a system.



Example: Google search system. Each request is worked upon by hundreds of computers that crawl the web and return the relevant results. To the user, Google appears to be one system, but it actually is multiple computers working together to accomplish one single task (return the results to the search query).

Characteristics of Distributed System:

- **Resource sharing:** A distributed system can share hardware, software, or data but they do not share main memory or disks.
- **Simultaneous processing:** Multiple machines can process the same function simultaneously

- **Scalability:** The computing and processing capacity can scale up as needed when extended to additional machines
- **Error detection:** Failures can be more easily detected
- **Transparency:** A node can access and communicate with other nodes in the system
- The computers in a distributed system communicate with one another through various communication media, such as high-speed private networks or the Internet.
- Data is spread over multiple machines.
- The computers in a distributed system are referred to by a number of different names, such as sites or nodes.
- Network interconnects the machines.
- Data is shared by users on multiple machines.

Types of Distributed Systems:

Homogeneous distributed databases:

- Same software/schema on all sites, data may be partitioned among sites.
- Goal: provide a view of a single database, hiding details of distribution.

Heterogeneous distributed databases:

- Different software/schema on different sites.
- Goal: integrate existing databases to provide useful functionality.

Components of Distributed System:

The components of Distributed System are,

- Node (Computer, Mobile, etc.)
- A communication link (Cables, Wi-Fi, etc.)

The architecture of Distributed System:

- **peer-to-peer** – all nodes are peers of each other and work towards a common goal
- **client-server** – some nodes become server nodes for the role of coordinator, arbiter, etc.

- **n-tier architecture** – different parts of an application are distributed in different nodes of the systems and these nodes work together to function as an application for the user/client

Limitations of Distributed System:

- Difficult to design and debug algorithms for the system. These algorithms are difficult because of the absence of a common clock; so, no temporal ordering of commands/logs can take place. Nodes can have different latencies which have to be kept in mind while designing such algorithms. The complexity increases with the increase in the number of nodes.
- No common clock causes difficulty in the temporal ordering of events/transactions
- Difficult for a node to get the global view of the system and hence take informed decisions based on the state of other nodes in the system

Advantages of Distributed System:

- **Low latency than a centralized system**: Distributed systems have low latency because of high geographical spread, hence leading to less time to get a response.
- **Scalability:** Distributed systems are highly scalable as they can be easily expanded by adding new nodes to the network. This allows the system to handle a large amount of data and traffic without compromising on performance.
- **Fault tolerance:** Distributed systems are fault-tolerant, meaning they can continue to function even if some nodes fail or go offline. This is because the workload is distributed among multiple nodes, so the system can continue to operate even if some nodes are down.
- **Increased reliability:** With multiple nodes in the network, distributed systems are more reliable than centralized systems. Even if one node fails, there are still other nodes that can provide the required service.
- **Cost-effective:** Distributed systems are often more cost-effective than centralized systems, as they can be built using off-the-shelf hardware and software components. This makes them a more affordable option for many organizations.
- **Improved performance:** Distributed systems can achieve higher performance by using parallel processing techniques, where tasks are split among multiple nodes in the network, and each node processes its share of the workload simultaneously. This can significantly reduce processing time and improve overall system performance.
- **Autonomy:** Each site is able to retain a degree of control over data stored locally.

- **Sharing data:** Users at one site are able to access the data residing at some other sites.

Disadvantages of Distributed System:

- Difficult to achieve consensus
- The conventional way of logging events by absolute time they occur is not possible here.
- **Network complexity:** Distributed systems require a complex network infrastructure to operate, which can be difficult to set up and maintain. Network outages and bottlenecks can cause system failures and data loss.
- **Security challenges:** Distributed systems can be vulnerable to security threats such as cyber-attacks, data breaches, and unauthorized access. It can be challenging to implement robust security measures across all nodes and ensure data privacy and integrity.
- **Synchronization issues:** Data consistency and synchronization across all nodes can be a challenge, especially when multiple users are accessing the system concurrently. Ensuring data consistency and avoiding data conflicts requires complex algorithms and protocols.
- **High development and maintenance costs:** Distributed systems require specialized skills and expertise to design, develop, and maintain. This can lead to high development and maintenance costs, especially for large-scale systems.
- **Limited scalability:** Distributed systems can have limited scalability due to network constraints, synchronization challenges, and other technical limitations. Scaling up the system requires significant investment and can be challenging to implement.

Applications of Distributed System:

- **Cluster computing** – a technique in which many computers are coupled together to work so that they achieve global goals. The computer cluster acts as if they were a single computer
- **Grid computing** – All the resources are pooled together for sharing in this kind of computing turning the systems into a powerful supercomputer; essentially.

Use Cases: SOA-based systems, Multiplayer online games

Organizations Using: Apple, Google, Facebook.

Distributed Consensus:

A procedure to reach a common agreement in a distributed or decentralized multi-agent platform. It is important for the message passing system.

Example –

A number of processes in a network decide to elect a leader. Each process begins with a bid for leadership. In traditional or conventional distributed systems, we apply consensus to ensure reliability and fault tolerance. It means, in a decentralized environment when you have multiple individual parties, and they can make their own decision, then it may happen that some node or some parties are working maliciously or working as a faulty individual. So, in those particular cases, it is important to come to a decision or common point of view. So having a common point of view in an environment where people can behave maliciously or people can crash the work in a faulty way, is the main difficulty. So, under this kind of distributed environment, our objective is to ensure reliability which means to ensure correct operation in the presence of faulty individuals.

Features:

- It ensures reliability and fault tolerance in distributed systems.
- In the presence of faulty individuals, it is Ensure correct operations.

Examples –

Commit a transaction in a database, State machine replication, Clock synchronization.

How to Achieve Distributed Consensus:

There are some conditions that need to be followed in order to achieve distributed consensus.

- **Termination –** Every non-faulty process must eventually decide.
- **Agreement –** The final decision of every non-faulty process must be identical.
- **Validity –** Every non-faulty process must begin and ends with the same value.
- **Integrity –** Every correct individual decides at most one value, and the decided value must be proposed by some individual.

Here is one validation criterion, So basically, we should reach a decision with a value that must be the initial value of some process because it is silly to reach an agreement when the agreed value reflects nobody's initial choice.

The Correctness of Distributed Consensus Protocol:

It can be described by the following properties as follows -

- **Safety Property –** It ensures that you will never converge to an incorrect value or correct individuals in a network will never converge to an incorrect value.
- **Liveness Property –** It states that every correct value must be accepted eventually which means something good will eventually happen.
- **Termination Property –** It guarantees that every correct process will eventually decide on a value. This ensures that the protocol will eventually terminate.
- **Agreement Property –** It guarantees that all correct processes will eventually agree on a single value. This ensures that all correct nodes in the network will come to a consensus.
- **Fault Tolerance –** Distributed consensus protocols must be able to handle failures and errors, both in the network and in the participating nodes. This ensures that the system remains correct and functional even in the presence of faults.
- **Byzantine Fault Tolerance –** Some distributed consensus protocols, like PBFT, have the additional property of Byzantine Fault Tolerance (BFT). This means they can tolerate up to a certain number of malicious nodes in the network without compromising safety and liveness properties.
- **Scalability –** The protocol must be able to scale to handle large networks and increasing numbers of nodes without sacrificing safety, liveness, or fault tolerance. This ensures that the protocol can be used in real-world scenarios with a large number of participants.

Application of Distributed Consensus:

- Leader election in a fault-tolerant environment for initiating some global action without introducing a single point of failure.
- Maintaining consistency in a distributed network. Suppose you have different nodes monitoring the same environment. If one of the nodes crashes, a consensus protocol ensures robustness against such faults.
- **Blockchain technology:** Distributed consensus is a fundamental concept in blockchain technology, which allows multiple nodes to agree on a shared database without relying on a central authority.
- **Distributed databases:** Distributed consensus protocols can be used to maintain consistency across multiple replicas of a distributed database.

- **Load balancing:** Consensus protocols can be used to dynamically distribute the workload across multiple nodes in a distributed system to ensure that no node is overloaded.
- **Fault tolerance:** Distributed consensus protocols can provide fault tolerance in distributed systems by allowing nodes to recover from crashes or network partitions.
- **Agreement protocols:** Consensus protocols can be used to achieve agreement among multiple nodes in a distributed system on a particular course of action or decision.