UNIT - 1

Background Theories: Cryptographic Hash Functions (SHA), Cryptographically Secured, Digital Signature, Public Key Cryptography (RSA), Chain of Blocks, Merkle Trees, Smart Contract, Centralized Vs. Distributed network. Actors and components in Blockchain solution.

Cryptography Hash Functions

Hash functions in cryptography are extremely valuable and are found in practically every information security application. A hash function transforms one numerical input value into another compressed numerical value. It is also a process that turns plaintext data of any size into a unique ciphertext of a predetermined length.

What is Cryptography Hash Function?

A cryptographic hash function (CHF) is an equation that is widely used to verify the validity of data. It has many applications, particularly in information security (e.g. user authentication). A CHF translates data of various lengths of the message into a fixed-size numerical string the hash. A cryptographic hash function is a single-directional work, making it extremely difficult to reverse to recreate the information used to make it.

How Does a Cryptography Hash Function Work?

- The hash function accepts data of a fixed length. The data block size varies between algorithms.
- If the blocks are too small, padding may be used to fill the space. However, regardless of the kind of hashing used, the output, or hash value, always has the same set length.
- The hash function is then applied as many times as the number of data blocks.

What Does a Cryptography Hash Function Do?

A hash function in cryptography takes a plaintext input and produces a hashed value output of a particular size that cannot be reversed. However, from a high-level viewpoint, they do more.

- Secure against unauthorized alterations: It assists you in even minor changes to a message that will result in the generation of a whole new hash value.
- Protect passwords and operate at various speeds: Many websites allow you to save your passwords so that you don't have to remember them each time you log in. However, keeping plaintext passwords on a public-facing server is risky since it exposes the information to thieves. Websites commonly use hash passwords to create hash values, which they then store.

Applications of Cryptographic Hash Functions

Below are some applications of cryptography hash functions

Message Authentication

- Message authentication is a system or service that verifies the integrity of a communication.
- It ensures data is received precisely as transmitted, with no modifications, insertions, or deletions, a hash function is used for message authentication, and the value is sometimes referred to as a message digest.
- Message authentication often involves employing a message authentication code (MAC).
- MACs are widely used between two parties that share a secret key for authentication purposes. A MAC function uses a secret key and data block to generate a hash value that identifies the protected communication.

Data Integrity Check

- Hash functions are most commonly used to create checksums for data files.
- This program offers the user with assurance that the data is correct.
- The integrity check allows the user to detect any modifications to the original file.
- It does not assure uniqueness. Instead of altering file data, the attacker can update the entire file, compute a new hash, and deliver it to the recipient.

Digital Signatures

- The digital signature application is comparable to message authentication.
- Digital signatures operate similarly to MACs.
- Digital signatures encrypt message hash values using a user's private key.

- The digital signature may be verified by anybody who knows the user's public key.

What are the Characteristics of Cryptographic Hash functions?

1. Deterministic

    - **Definition:** For a given input, the output (hash) is always the same.
    - **Example:**
        - SHA-256("Hello") will **always** produce the same hash.
          Ensures consistency in verification and authentication.

2. Fast Computation

    - **Definition:** The hash function should compute the output quickly, even for large inputs
    - Useful for real-time applications like blockchain mining, authentication, and digital signatures.

3. Preimage Resistance (One-way property)

    - **Definition:** Given a hash output H(x), it should be **computationally infeasible** to find the original input x.
    - **Use:** Ensures that attackers cannot reverse-engineer the input.
      Prevents password recovery and data leakage.

4. Second Pre-image Resistance

    - **Definition:** Given an input x1, it should be **hard to find another input x2** such that hash(x1) = hash(x2) and x1 ≠ x2.
      Use: Prevents substitution attacks (changing one message to another with the same hash).

5. Collision Resistance

    - **Definition:** It should be **infeasible to find two different inputs** x and y such that hash(x) = hash(y).
    - Important to prevent data tampering.
    - A good hash function has a negligible probability of collision.

6. Avalanche Effect

- **Definition:** A **small change** in the input (even one bit) should result in a **drastically different** hash output.
- **Example:**SHA-256("hello") and SHA-256("Hello") produce totally different hashes.This makes the output unpredictable and helps detect modifications.

7. Fixed Output Length

- **Definition:** The hash value should always be of a **fixed length**, regardless of the input size.
- **Example:**
SHA-256 always gives a 256-bit (32-byte) hash.

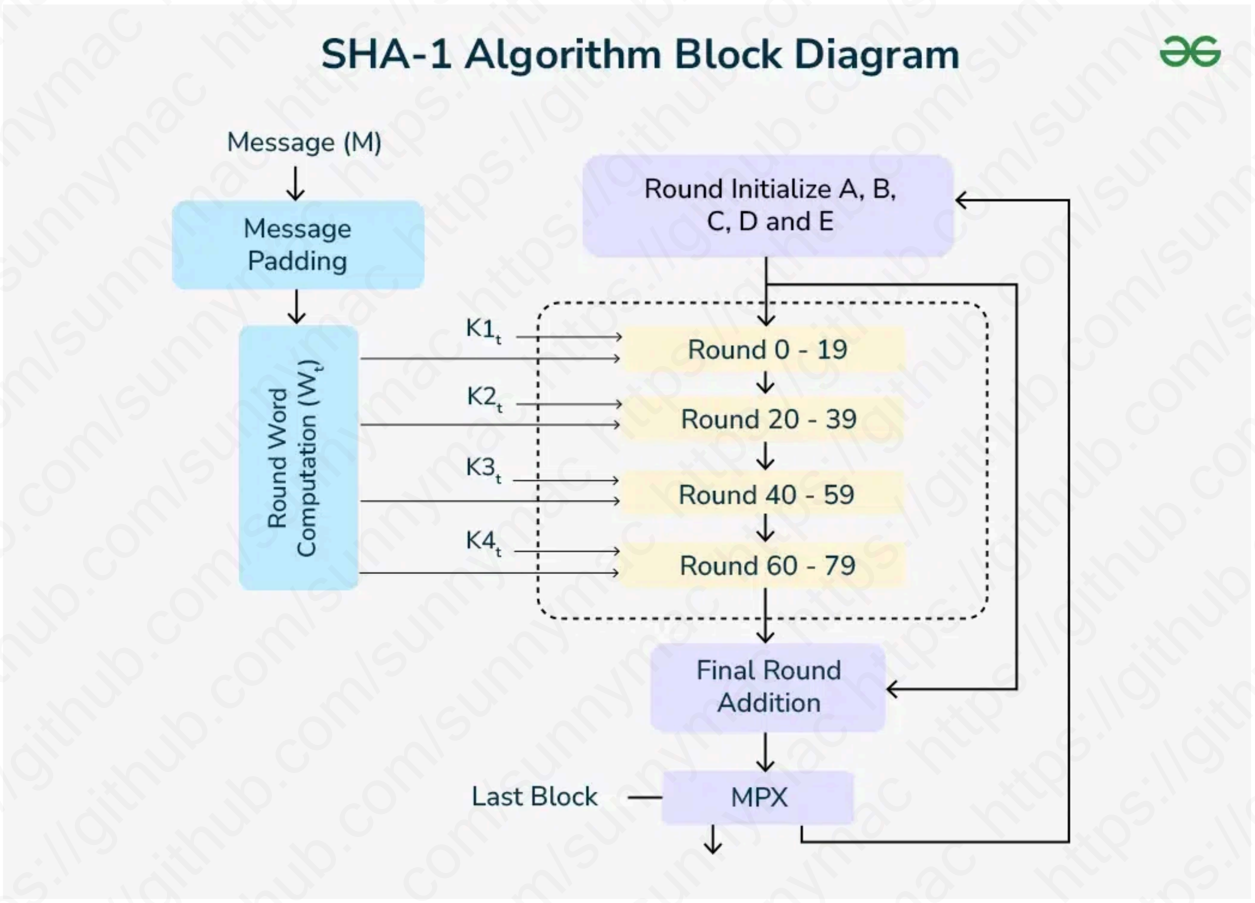
8. Pseudorandomness

- **Definition:** Output should appear random and have no detectable patterns.
- Helps resist statistical or pattern-based attacks.

Youtube Tutorial :
What is a Cryptographic Hash Function? | Where & How It Is Used In Crypto (A…
What is a Cryptographic Hashing Function? (Example + Purpose)
Hashing in Blockchain

SHA-1 Hash

SHA-1 or Secure Hash Algorithm 1 is a cryptographic algorithm that takes an input and produces a 160-bit (20-byte) hash value. This hash value is known as a message digest. This message digest is usually then rendered as a hexadecimal number which is 40 digits long. It is a U.S. Federal Information Processing Standard and was designed by the United States National Security Agency. SHA-1 has been considered insecure since 2005. Major tech giants browsers like Microsoft, Google, Apple, and Mozilla have stopped accepting SHA-1 SSL certificates by 2017.



How SHA-1 Works

Components and Process Flow:

1. **Message (M)**:
    - The original input message that needs to be hashed.
2. **Message Padding**:
    - The initial step where the message is padded to ensure its length is congruent to 448 modulo 512. This step prepares the message for processing in 512-bit blocks.
3. **Round Word Computation (WtW_tWt)**:
    - After padding, the message is divided into blocks of 512 bits, and each block is further divided into 16 words of 32 bits. These

words are then expanded into 80 32-bit words, which are used in the subsequent rounds.

4. **Round Initialize (A, B, C, D, and E)**:
    - Initialization of five working variables (A, B, C, D, and E) with specific constant values. These variables are used to compute the hash value iteratively.
5. **Round Constants (KtK_tKt)**:
    - SHA-1 uses four constant values (K1K_1K1, K2K_2K2, K3K_3K3, K4K_4K4), each applied in a specific range of rounds:
        - K1K_1K1 for rounds 0-19
        - K2K_2K2 for rounds 20-39
        - K3K_3K3 for rounds 40-59
        - K4K_4K4 for rounds 60-79
6. **Rounds (0-79)**:
    - The main computation loop of SHA-1, divided into four stages (each corresponding to one of the constants K1K_1K1 to K4K_4K4). In each round, a combination of logical functions and operations is performed on the working variables (A, B, C, D, and E) using the words generated in the previous step.
7. **Final Round Addition**:
    - After all 80 rounds, the resulting values of A, B, C, D, and E are added to the original hash values to produce the final hash.
8. **MPX (Multiplexing)**:
    - Combines the results from the final round addition to form the final message digest.

Youtube Tutorial :

SHA-1 (Secure hash Algorithm) working in English | CSS series

Cryptographically Secured

"Cryptographically secured" means that a system, data, or communication is protected using cryptographic methods to ensure confidentiality, integrity, authenticity, and non-repudiation.

Cryptography is the science of securing information using mathematical techniques, typically involving encryption and decryption.When something is cryptographically secured, it means that security relies on cryptographic algorithms to prevent unauthorized access or tampering.

Key Aspects of Cryptographic Security

Aspect	Purpose
Confidentiality	Ensures that only authorized parties can access the information (e.g., using **encryption**)
Integrity	Ensures the data hasn't been altered (e.g., using **hash functions** like SHA-256)
Authentication	Confirms the identity of users or systems (e.g., using **digital signatures**)
Non-repudiation	Ensures that a party **cannot deny** the authenticity of a message they sent (e.g., using **public key cryptography**)

Examples

- Encrypted Messaging Apps (e.g., Signal, WhatsApp) - Use cryptographic protocols like end-to-end encryption to ensure no one else can read the conversation.
- Blockchain- Transactions are cryptographically signed to prove ownership and ensure immutability.
- Secure Websites (HTTPS)- Use SSL/TLS encryption to secure data between your browser and the server.

Why It's Important

- Prevents data breaches
- Protects user privacy
- Enables secure online transactions
- Maintains trust in digital systems

What is a Digital Signature?

A digital signature is a cryptographic technique used to prove the authenticity and integrity of a message, document, or software. It's the digital equivalent of a handwritten signature or a stamped seal, but much more secure because it uses mathematical algorithms.

How Digital Signatures Work (Step-by-Step)

Let's understand how a digital signature is created and verified using public key cryptography (specifically asymmetric encryption like RSA, DSA, or ECDSA):

1. Key Generation

- A person (say, Alice) generates:
    - A private key (kept secret)
    - A public key (shared with others)

2. Signing the Document

To send a signed document to Bob, Alice does the following:

a. Hash the Document

- A hash function (like SHA-256) converts the document into a fixed-length digest.
- This digest is a unique fingerprint of the document.

b. Encrypt the Hash

- Alice uses her private key to encrypt the hash.
- The result is the digital signature.
- She sends both the original document and the signature to Bob.

3. Verifying the Signature

When Bob receives the document, he:

a. Hashes the Received Document

- Bob applies the same hash function to the document to get a new digest.

b. Decrypts the Signature

- Bob uses Alice's public key to decrypt the signature, which gives the original hash that Alice created.

c. Compares the Hashes

- If both hashes match:
  ✅ The document is authentic and untampered.
  ✅ The signature is valid.

- If the hashes do not match:
  ❌ The document has been altered or the signature is fake.

Cryptographic Algorithms Used

Algorithm	Description
RSA	Uses large prime numbers and modular arithmetic
DSA	Digital Signature Algorithm (uses discrete logarithms)
ECDSA	Elliptic Curve version of DSA (more efficient)

Applications of Digital Signatures

Application	Purpose
E-Government Forms	Verifying citizen identity (e.g., Aadhaar e-sign)
Software Distribution	Ensures the software hasn't been tampered
Blockchain & Crypto	Signs transactions to ensure authenticity
Secure Email (PGP, S/MIME)	Confirms sender and message integrity
Legal Documents	Digital contracts and e-signatures

Why Digital Signatures Matter

Feature	Benefit
Authenticity	Confirms who sent it
Integrity	Confirms nothing was changed
Non-repudiation	Sender **can't deny** having signed it

Efficiency	Quick to verify, scalable

Youtube Tutorial :

▶ Digital Signatures and Digital Certificates

▶ What are Digital Signatures and How Do They Work?

▶ What is digital signature?

Public Key Cryptography (Asymmetric Key Cryptography)

What is Public Key Cryptography?

Public Key Cryptography is a method of encrypting and securing data using two different but mathematically related keys:

- A public key (shared with everyone)
- A private key (kept secret)

Unlike symmetric cryptography (where the same key is used for both encryption and decryption), public key cryptography uses One key to encrypt, Another key to decrypt.
This enables secure communication, digital signatures, and identity verification even over insecure channels like the internet.

How It Works (Step-by-Step)

Step 1: Key Pair Generation

The user generates a **key pair**: A **public key**: shared openly, A **private key**: kept secure and secret.

Step 2: Secure Communication (Encryption/Decryption)

Let's say **Alice** wants to send a secure message to **Bob**.

a. Alice gets Bob's public key
b. Alice encrypts the message with Bob's public key

c. Bob receives the encrypted message and decrypts it with his private key

Only Bob's private key can decrypt what was encrypted using his public key — **this ensures confidentiality**.

Step 3: Authentication and Digital Signature

Let's say **Bob** wants to prove he sent a message (authenticity):

a. Bob hashes the message
b. He encrypts the hash using his private key → this becomes the digital signature
c. Alice receives the message and signature
d. Alice decrypts the signature using Bob's public key, and compares the hash with the message's own hash

If they match → message is **authentic and untampered**

Only Bob could have created a signature that matches his public key → **authenticity and non-repudiation**

Summary Table

Task	Key Used	Purpose
Encrypt message	Public Key	So only the private key owner can read it
Decrypt message	Private Key	Read message securely
Sign message (hash)	Private Key	Prove sender's identity
Verify signature	Public Key	Confirm message came from the sender

Algorithms Used

Algorithm	Description
RSA	Based on large prime numbers
ECC	Uses elliptic curves (more efficient)

ElGamal	Based on discrete logarithms
DSA	For digital signatures only

Real-Life Applications

Application	Use of Public Key Cryptography
HTTPS (Web Security)	Secure data exchange in browsers
Email encryption (PGP)	Private, secure email communication
Digital Signatures	Document authenticity
Cryptocurrency wallets	Signing blockchain transactions
SSL/TLS Certificates	Server identity verification

Comparison: Symmetric vs. Asymmetric

Feature	Symmetric Encryption	Public Key Encryption (Asymmetric)
Keys	1 (same key)	2 (public + private)
Speed	Faster	Slower
Key sharing	Must be kept secret	Only private key needs to be secret
Use case	Bulk data encryption	Authentication, key exchange, digital signatures

Youtube Tutorial :

Public Key Encryption (Asymmetric Key Encryption)

Asymmetric Encryption - Simply explained

RSA (Rivest–Shamir–Adleman)

RSA(Rivest-Shamir-Adleman) Algorithm is an asymmetric or public-key cryptography algorithm which means it works on two different keys: Public Key and Private Key. The Public Key is used for encryption and is known to everyone, while the Private Key is used for decryption and must be kept secret by the receiver. RSA Algorithm is named after Ron Rivest, Adi Shamir and Leonard Adleman, who published the algorithm in 1977.

Example of Asymmetric Cryptography:

If Person A wants to send a message securely to Person B;Person A encrypts the message using Person B's Public Key.Person B decrypts the message using their Private Key.

RSA Algorithm

RSA Algorithm is based on factorization of large numbers and modular arithmetic for encrypting and decrypting data. It consists of three main stages:

1.Key Generation: Creating Public and Private Keys
2.Encryption: Sender encrypts the data using Public Key to get cipher text.
3.Decryption: Decrypting the cipher text using Private Key to get the original data.

1. Key Generation

- Choose two large prime numbers, say p and q. These prime numbers should be kept secret.
- Calculate the product of primes, $n = p * q$. This product is part of the public as well as the private key.
- Calculate Euler Totient Function$\Phi(n)$ as $\Phi(n) = \Phi(p * q) = \Phi(p) * \Phi(q) = (p - 1) * (q - 1)$.
- Choose encryption exponent e, such that
    - $1 < e < \Phi(n)$, and
    - $\gcd(e, \Phi(n)) = 1$, that is e should be co-prime with $\Phi(n)$.

- Calculate decryption exponent d, such that
    - (d * e) ≡ 1 mod Φ(n), that is d is modular multiplicative inverse of e mod Φ(n). Some common methods to calculate multiplicative inverse are: Extended Euclidean Algorithm, Fermat's Little Theorem, etc.
    - We can have multiple values of d satisfying (d * e) ≡ 1 mod Φ(n) but it does not matter which value we choose as all of them are valid keys and will result into same message on decryption.
- Finally, the Public Key = (n, e) and the Private Key = (n, d).

2. Encryption

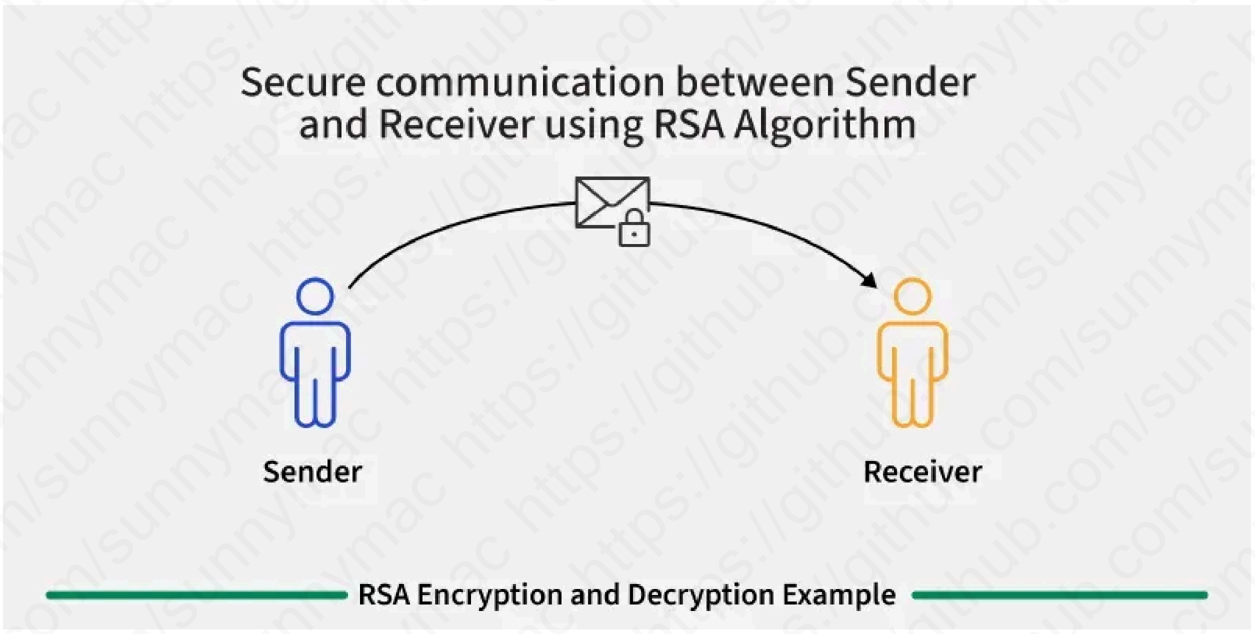
To encrypt a message M, it is first converted to numerical representation using ASCII and other encoding schemes. Now, use the public key (n, e) to encrypt the message and get the cipher text using the formula:

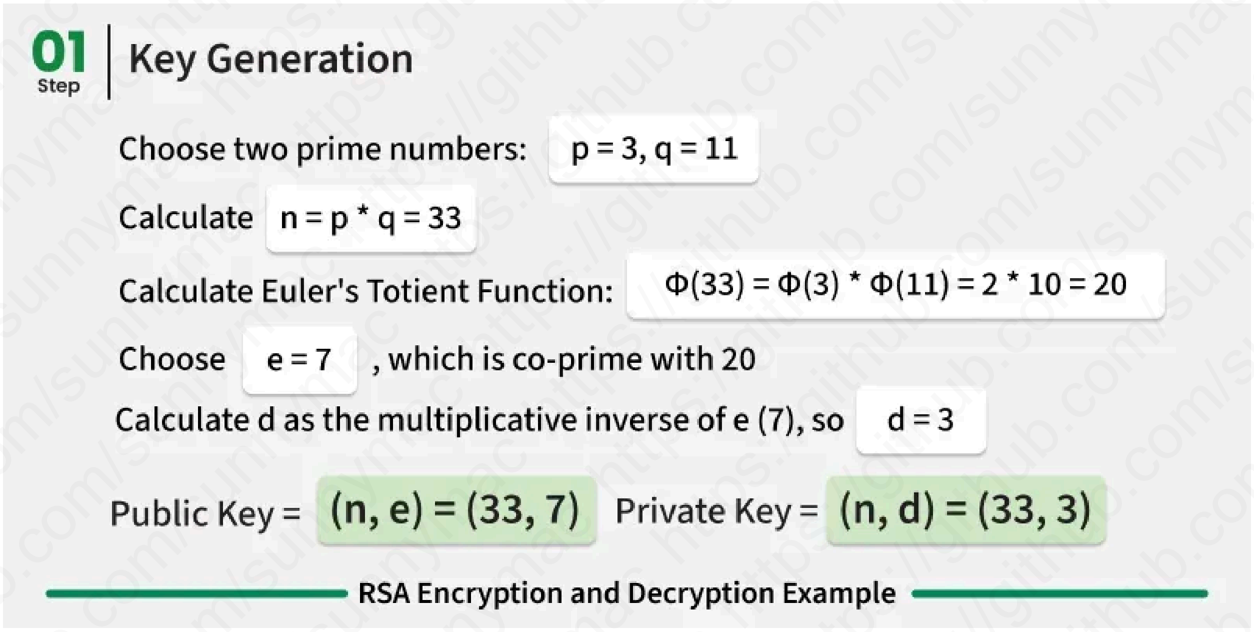
C = Me mod n, where C is the Cipher text and e and n are parts of the public key.

3. Decryption

To decrypt the ciphertext C, use the private key (n, d) and get the original data using the formula:

M = Cd mod n, where M is the message and d and n are parts of the private key.


Secure communication between Sender and Receiver using RSA Algorithm
Sender
Receiver
RSA Encryption and Decryption Example



01 Step | Key Generation

Choose two prime numbers: $p = 3, q = 11$

Calculate $n = p * q = 33$

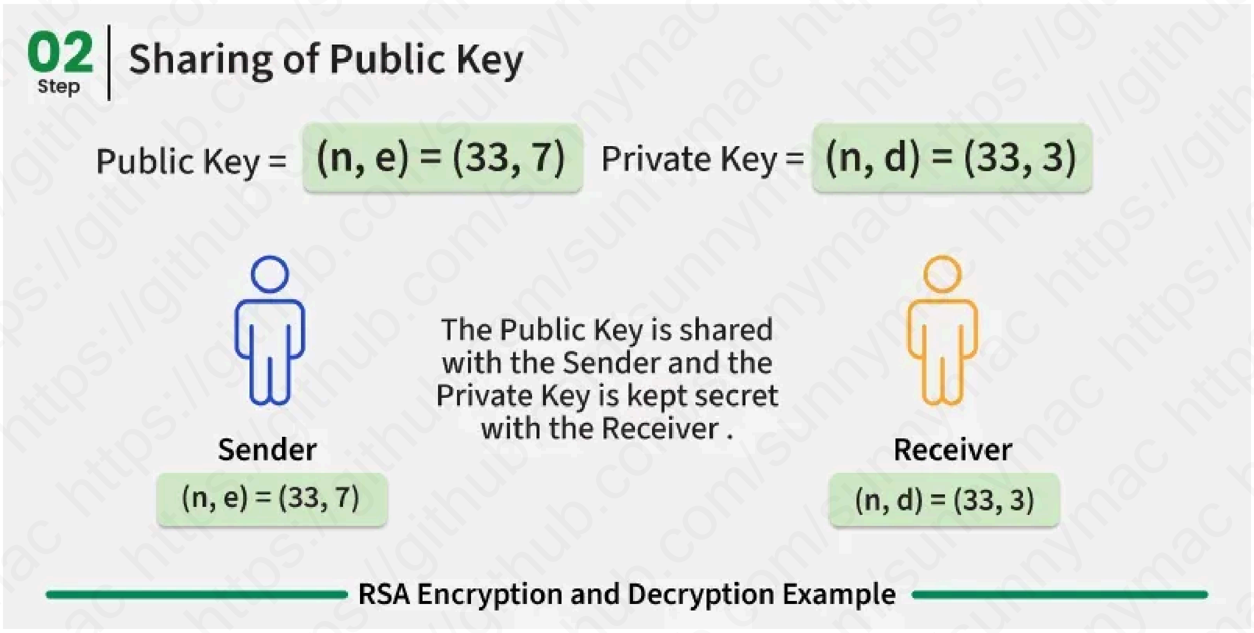
Calculate Euler's Totient Function: $\Phi(33) = \Phi(3) * \Phi(11) = 2 * 10 = 20$

Choose $e = 7$, which is co-prime with 20

Calculate d as the multiplicative inverse of e (7), so $d = 3$

Public Key = $(n, e) = (33, 7)$ Private Key = $(n, d) = (33, 3)$

RSA Encryption and Decryption Example



02 Step | Sharing of Public Key

Public Key = $(n, e) = (33, 7)$ Private Key = $(n, d) = (33, 3)$

The Public Key is shared with the Sender and the Private Key is kept secret with the Receiver .

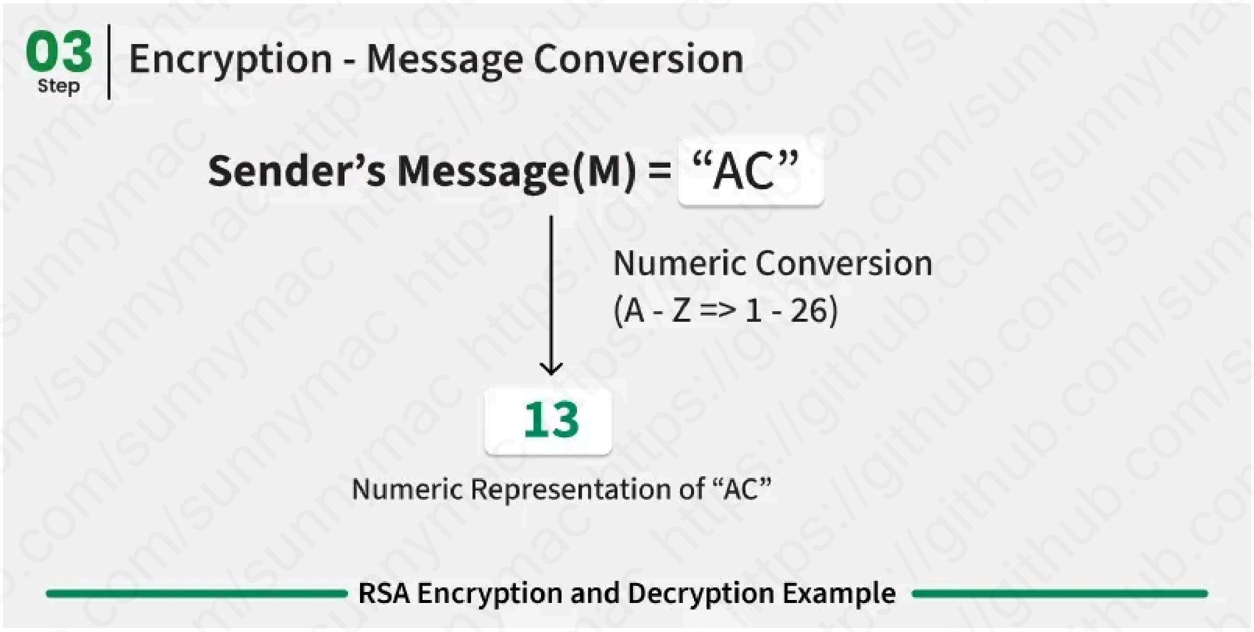
Sender

$(n, e) = (33, 7)$

Receiver

$(n, d) = (33, 3)$

RSA Encryption and Decryption Example



03 Step | Encryption - Message Conversion

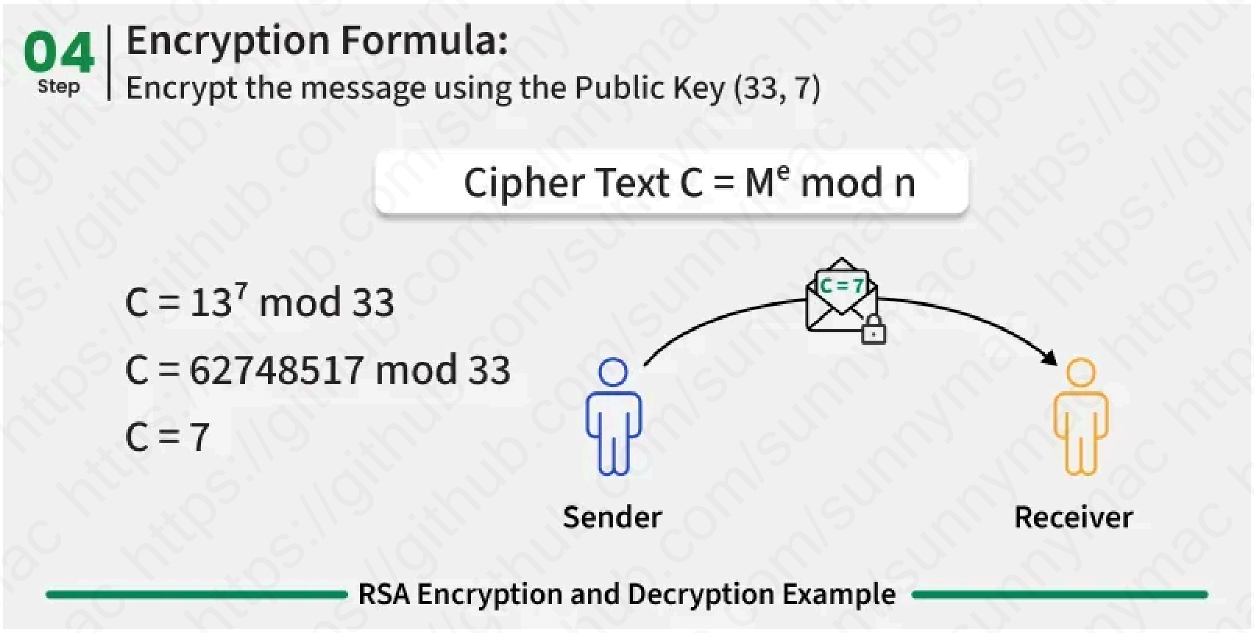
Sender's Message(M) = "AC"

Numeric Conversion
(A - Z => 1 - 26)

13

Numeric Representation of "AC"

RSA Encryption and Decryption Example



04 Step | Encryption Formula:

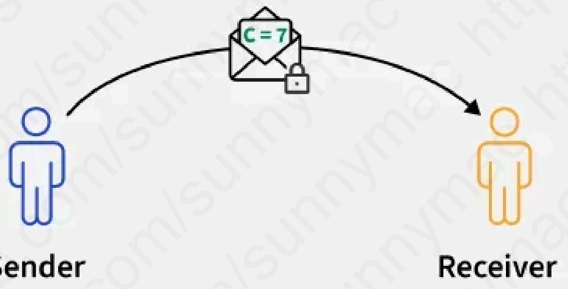
Encrypt the message using the Public Key (33, 7)

Cipher Text $C = M^e \bmod n$

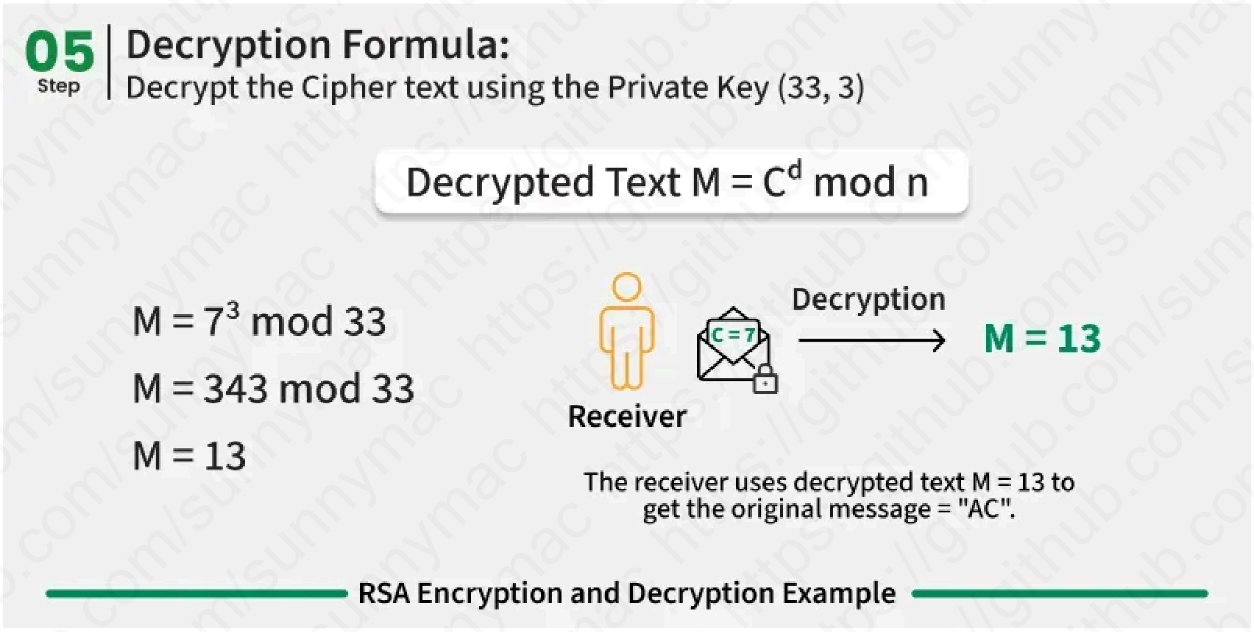
$C = 13^7 \bmod 33$

$C = 62748517 \bmod 33$

$C = 7$



RSA Encryption and Decryption Example


05
Step
Decryption Formula:
Decrypt the Cipher text using the Private Key (33, 3)
Decrypted Text $M = C^d \bmod n$
$M = 7^3 \bmod 33$
$M = 343 \bmod 33$
$M = 13$
Receiver
C = 7
Decryption
M = 13
The receiver uses decrypted text M = 13 to get the original message = "AC".
RSA Encryption and Decryption Example

Advantages

- Security: RSA algorithm is considered to be very secure and is widely used for secure data transmission.
- Public-key cryptography: RSA algorithm is a public-key cryptography algorithm, which means that it uses two different keys for encryption and decryption. The public key is used to encrypt the data, while the private key is used to decrypt the data.
- Key exchange: RSA algorithm can be used for secure key exchange, which means that two parties can exchange a secret key without actually sending the key over the network.
- Digital signatures: RSA algorithm can be used for digital signatures, which means that a sender can sign a message using their private key, and the receiver can verify the signature using the sender's public key.
- Widely used: Online banking, e-commerce, and secure communications are just a few fields and applications where the RSA algorithm is extensively developed.

Disadvantages

- Slow processing speed: RSA algorithm is slower than other encryption algorithms, especially when dealing with large amounts of data.
- Large key size: RSA algorithm requires large key sizes to be secure, which means that it requires more computational resources and storage space.

- Vulnerability to side-channel attacks: RSA algorithm is vulnerable to side-channel attacks, which means an attacker can use information leaked through side channels such as power consumption, electromagnetic radiation, and timing analysis to extract the private key.
- Limited use in some applications: RSA algorithm is not suitable for some applications, such as those that require constant encryption and decryption of large amounts of data, due to its slow processing speed.
- Complexity: The RSA algorithm is a sophisticated mathematical technique that some individuals may find challenging to comprehend and use.
- Key Management: The secure administration of the private key is necessary for the RSA algorithm, although in some cases this can be difficult.
- Vulnerability to Quantum Computing: Quantum computers have the ability to attack the RSA algorithm, potentially decrypting the data.

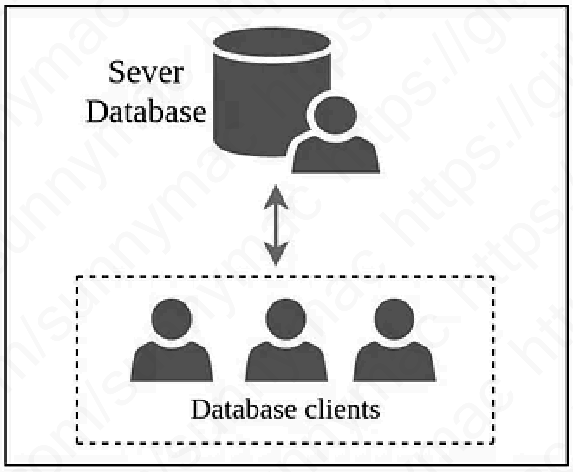
Youtube Tutorial :
Public Key Cryptography: RSA Encryption

Chain of Blocks

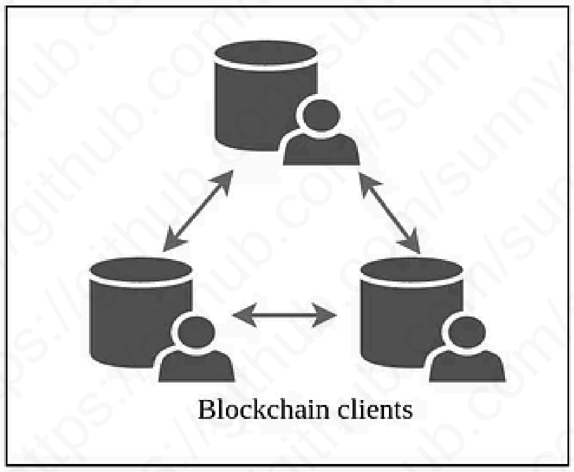
Blockchain, as the term suggests, is a chain of blocks. In more technical terms, Blockchain is a decentralized and distributed database. Think of it like a google doc that stores information which is shared and controlled by everyone.

Decentralized & Distributed: This means that no singular individual or entity has control over the Blockchain. It's entirely decentralized, meaning that it's controlled by multiple participants and distributed across the network. This technology is known as the Distributed Ledger Technology (DLT).

Database: A database is an organized collection of data, typically stored electronically in a computer system. Although, while Blockchain is considered a database, a database is not a blockchain. The primary difference between a Blockchain and a database is centralization. All records on a database are centralized, while as mentioned above, a Blockchain is entirely decentralized.

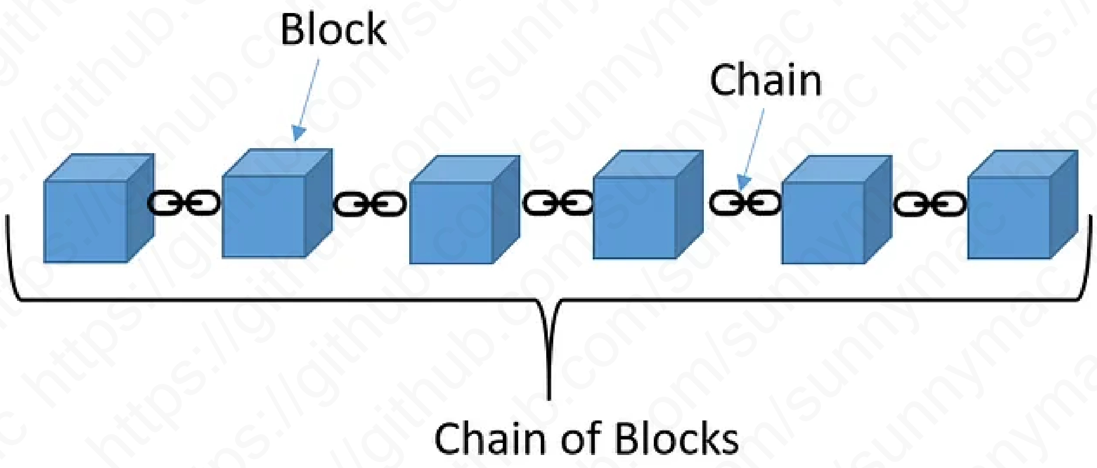


Centralized Database Architecture

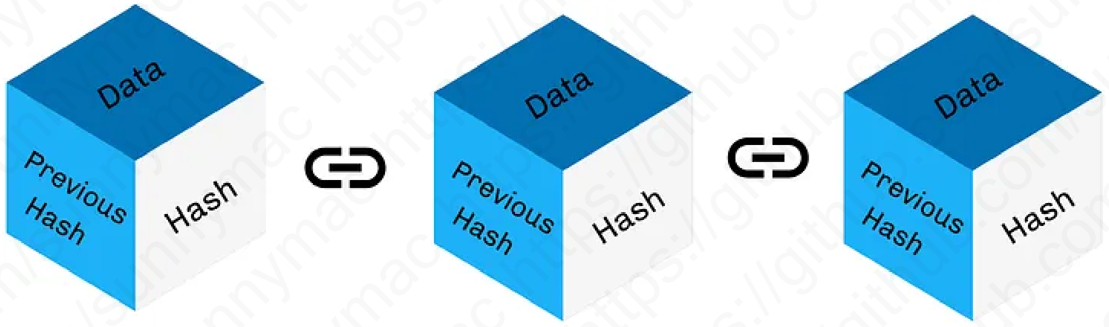


Blockchain Architecture

Each piece of data on the blockchain is stored in a block, and all the blocks of data are chained together.



Each block contains 3 key things.



Data: This is the information that will be permanently stored on the public ledger. For example: When a transaction takes place, the block will contain the information about the sender, the receiver, the timestamp etc.

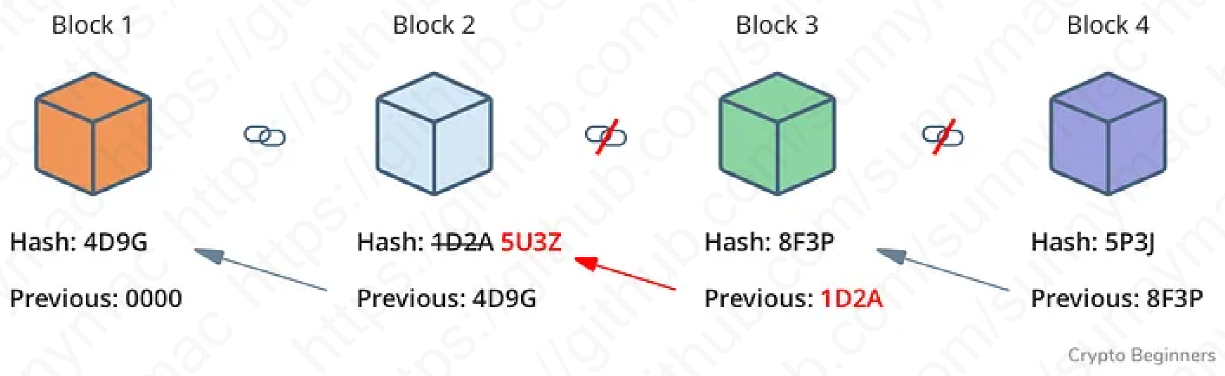
Hash: Cryptographic Hashing, in very simple terms is converting the big data stored on the block into a unique set of letters and numbers that has a fixed length. It's like a mathematical fingerprint of that specific data.

Hash of the previous block: The next block in the chain will also contain the hash of the previous block.

Now you might wonder, why do we need a hash for the data in the first place? The reason being, if someone fiddles with any data on a block, anywhere in the chain, it immediately makes the recorded hash result for that block not match the data.

Remember, the hash for data that has been tampered with is also present on the next block.

This makes it impossible to tamper with the data without it becoming immediately obvious to everyone else, because the sequential hashing "doesn't make sense".



If the hash of the previous block is changed, the sequential hashing won't add up

Once the transaction is agreed between the users, it needs to first be authorized before it is added on the block.

For a public blockchain, the decision to add a transaction to the chain is made by consensus. This means that the majority of “nodes” (computers in the network) must agree that the transaction is valid.

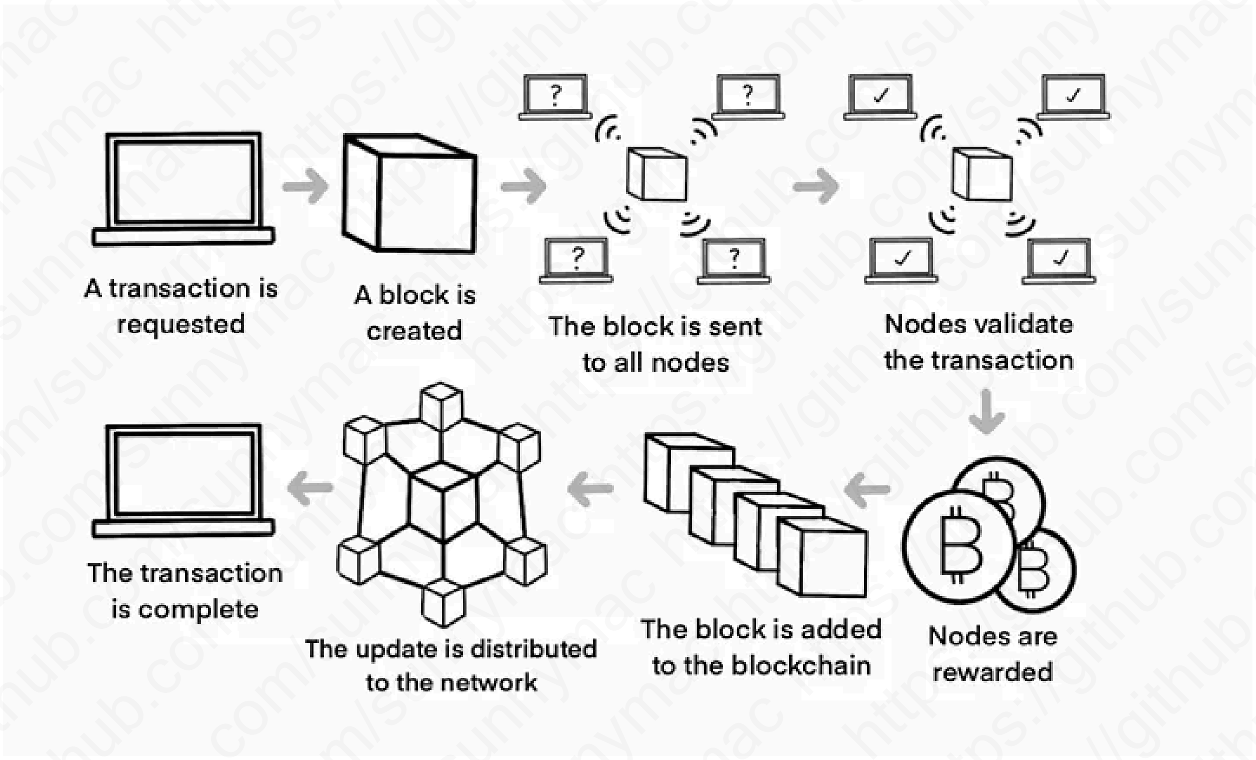
The purpose of the consensus algorithms in blockchain is to maintain security and integrity within the system.

Consensus Algorithm: a process in computer science used to achieve agreement on a single data value among distributed processes or systems.

Two main consensus algorithms used in the Blockchain: Proof- of -Work and Proof-of- Stake.

Proof of Work (PoW): Proof of Work requires the people who own the computers in the network to solve a complex mathematical problem to be able to add a block to the chain. These ‘miners’ compete against each other to see who can solve this problem first. The winning ‘miner’ receives a reward, typically an amount of crypto. With this mechanism, the network requires a huge amount of processing power. Currently, cryptocurrencies like Bitcoin and Ethereum use this mechanism.

Proof of Stake (PoS): In Proof of Stake, participants must have a stake in the blockchain, usually by owning some of the cryptocurrency — in exchange for a chance of getting to validate a new transaction, update the blockchain, and earn a reward. This saves computing power resources because no mining is required. Currently, cryptocurrencies like Cardano, Ethereum 2,Avalanche, and Solana use this mechanism.


A transaction is requested
A block is created
The block is sent to all nodes
Nodes validate the transaction
Nodes are rewarded
The block is added to the blockchain
The update is distributed to the network
The transaction is complete

Blockchain transaction in a nutshell

Youtube Tutorial :

Blockchain In 7 Minutes | What Is Blockchain | Blockchain Explained|How Blo…

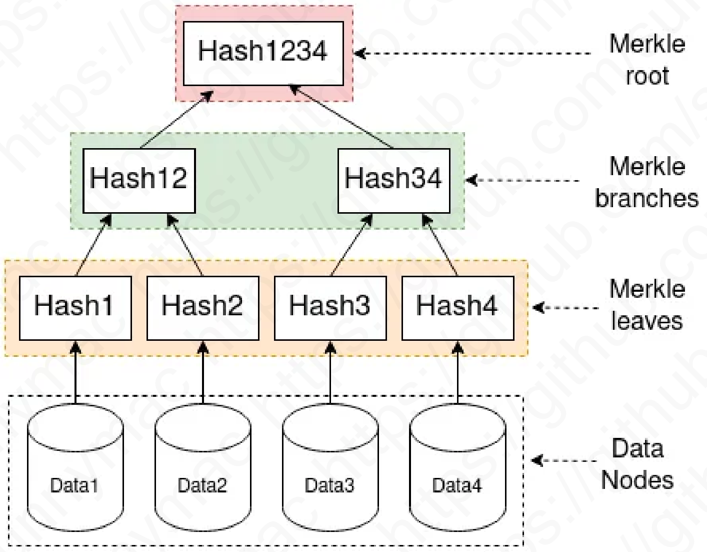
How does a blockchain work - Simply Explained

All about Blockchain | Simply Explained

Make hashes for your data Click Here.

Merkle Trees

A Merkle tree is fundamentally just a hierarchical set of hash values, building from a set of actual data (Merkle leaf) to intermediate hashes (Merkle branches) and up to the Merkle root that summarizes all the data in one hash value.



In this figure, the bottom nodes (Data1-Data4) are the actual data processed by the application. Each of these is summarized by their respective hash value (Hash1-Hash4), as a Merkle leaf. From these, the Merkle tree builds a hierarchy, combining hashes together until only one is left. The nodes combining other hash nodes are called Merkle branches (here Hash12 and Hash34). When there is only one left (here Hash1234), this is called the Merkle root.

Youtube Tutorial :
▶ Merkle Tree | Merkle Root | Blockchain
▶ Merkle Trees Are Efficient! - Alchemy University

Smart Contract

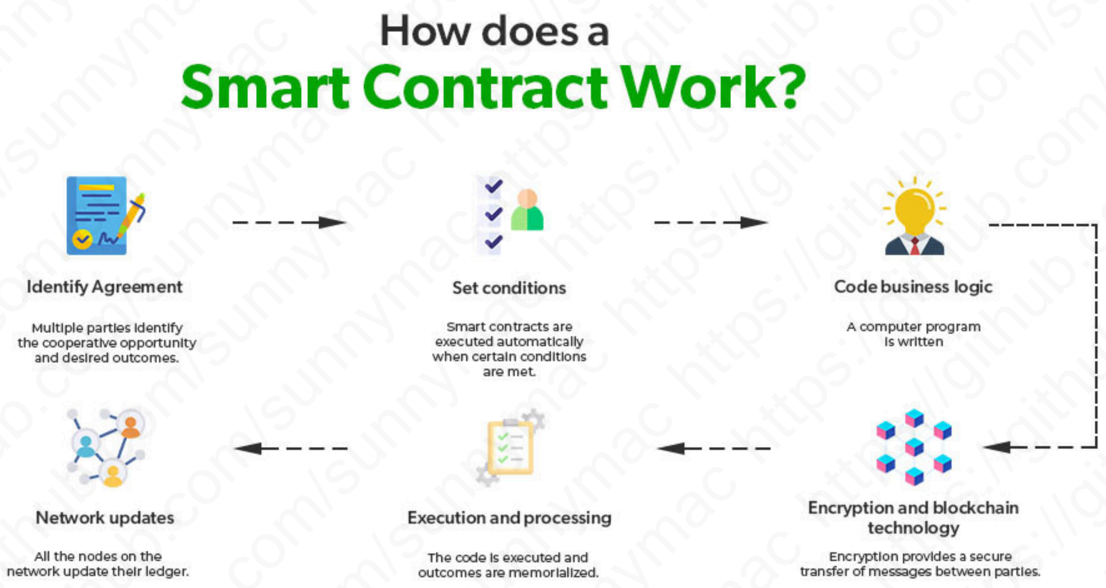
A Smart Contract (or crypto contract) is a computer program that directly and automatically controls the transfer of digital assets between the parties under certain conditions. A smart contract works in the same way as a traditional contract while also automatically enforcing the contract. Smart contracts are programs that execute exactly as they are set up(coded, programmed) by their creators. Just like a traditional contract is enforceable by law, smart contracts are enforceable by code.

Features of Smart Contracts

- Distributed: Everyone on the network is guaranteed to have a copy of all the conditions of the smart contract and they cannot be changed by one of the parties. A smart contract is replicated and distributed by all the nodes connected to the network.
- Deterministic: Smart contracts can only perform functions for which they are designed only when the required conditions are met. The final outcome will not vary, no matter who executes the smart contract.
- Immutable: Once deployed smart contract cannot be changed, it can only be removed as long as the functionality is implemented previously.
- Autonomy: There is no third party involved. The contract is made by you and shared between the parties. No intermediaries are involved which minimizes bullying and grants full authority to the dealing parties. Also, the smart contract is maintained and executed by all the nodes on the network, thus removing all the controlling power from any one party's hand.
- Customizable: Smart contracts have the ability for modification or we can say customization before being launched to do what the user wants it to do.
- Transparent: Smart contracts are always stored on a public distributed ledger called blockchain due to which the code is visible to everyone, whether or not they are participants in the smart contract.
- Trustless: These are not required by third parties to verify the integrity of the process or to check whether the required conditions are met.

- Self-verifying: These are self-verifying due to automated possibilities.
- Self-enforcing: These are self-enforcing when the conditions and rules are met at all stages.

Smart Contract Working


How does a
Smart Contract Work?
Identify Agreement
Multiple parties identify the cooperative opportunity and desired outcomes.
Set conditions
Smart contracts are executed automatically when certain conditions are met.
Code business logic
A computer program is written
Encryption and blockchain technology
Encryption provides a secure transfer of messages between parties.
Execution and processing
The code is executed and outcomes are memorialized.
Network updates
All the nodes on the network update their ledger.

- Identify Agreement: Multiple parties identify the cooperative opportunity and desired outcomes and agreements could include business processes, asset swaps, etc.
- Set conditions: Smart contracts could be initiated by parties themselves or when certain conditions are met like financial market indices, events like GPS locations, etc.
- Code business logic: A computer program is written that will be executed automatically when the conditional parameters are met.
- Encryption and blockchain technology: Encryption provides secure authentication and transfer of messages between parties relating to smart contracts.
- Execution and processing: In blockchain iteration, whenever consensus is reached between the parties regarding authentication and verification then the code is executed and the outcomes are memorialized for compliance and verification.
- Network updates: After smart contracts are executed, all the nodes on the network update their ledger to reflect the new state. Once the record is posted

and verified on the blockchain network, it cannot be modified, it is in append mode only.

Advantages of Smart Contracts

- Recordkeeping: All contract transactions are stored in chronological order in the blockchain and can be accessed along with the complete audit trail. However, the parties involved can be secured cryptographically for full privacy.
- Autonomy: There are direct dealings between parties. Smart contracts remove the need for intermediaries and allow for transparent, direct relationships with customers.
- Reduce fraud: Fraudulent activity detection and reduction. Smart contracts are stored in the blockchain. Forcefully modifying the blockchain is very difficult as it's computation-intensive. Also, a violation of the smart contract can be detected by the nodes in the network and such a violation attempt is marked invalid and not stored in the blockchain.
- Fault-tolerance: Since no single person or entity is in control of the digital assets, one-party domination and the situation of one part backing out do not happen as the platform is decentralized and so even if one node detaches itself from the network, the contract remains intact.
- Enhanced trust: Business agreements are automatically executed and enforced. Plus, these agreements are immutable and therefore unbreakable and undeniable.
- Cost-efficiency: The application of smart contracts eliminates the need for intermediaries(brokers, lawyers, notaries, witnesses, etc.) leading to reduced costs. Also eliminates paperwork leading to paper saving and money-saving.

Challenges of Smart Contracts

- No regulations: A lack of international regulations focusing on blockchain technology(and related technology like smart contracts, mining, and use cases like cryptocurrency) makes these technologies difficult to oversee.
- Difficult to implement: Smart contracts are also complicated to implement because it's still a relatively new concept and research is still going on to understand the smart contract and its implications fully.

- Immutable: They are practically immutable. Whenever there is a change that has to be incorporated into the contract, a new contract has to be made and implemented in the blockchain.

Youtube Video :
Smart contracts - Simply Explained

Centralized Vs. Distributed network

Centralized systems have a single, central point of control, like a hub controlling all the activities. For instance, a school administration office manages all the classrooms and teachers. On the other hand, distributed systems are like a team of equals, with no single point of control. Each part of the system can operate independently, yet they work together seamlessly, like computers connected to each other.

What is a Centralized System?
A centralized system is a type of system where all the important tasks like processing data, storing information, and making decisions are done by a single main computer or server. This means that there is one central place that controls and manages all the resources and important choices for the whole system. In such systems, all resources, data, and functionalities are managed and controlled from this central point.

Characteristics of Centralized Systems

- Single Point of Control: In a centralized system, there is a single point of control and authority. This central entity typically makes all decisions and manages all resources.
- Centralized Data Management: All data and resources are stored and managed centrally. This means that all data processing, storage, and retrieval activities occur within the central system.
- Hierarchical Structure: Centralized systems often have a hierarchical structure, with lower-level nodes or entities reporting to and receiving instructions from the central authority.
- Communication Flow: Communication within a centralized system typically flows from peripheral nodes or entities to the central node.

- Simplicity in Management: Centralized systems are relatively simpler to manage and administer since all control and decision-making are centralized. This can lead to efficient coordination and streamlined operations.

Use Cases of Centralized Systems

- Small Office Network: Many offices use one main computer to run things. This main computer stores files for all workers. It also helps computers access the network. The main computer checks workers are who they say. Using one main computer makes it simpler to manage everything. It also allows all workers to use things the same way.
- Traditional Client-Server Architecture: A lot of older programs like email, websites, and databases work one way. Clients talk to one main server to get what they need. This setup has a center. Computers connect to the main spot to get services or info.
- Standalone Applications: Apps running on one machine do everything locally. They process and store things without needing other machines. This is a centralized system. All the work happens on the single machine you are using.

What is a Distributed System?

In a distributed system, different parts of a computer system are located on different computers or devices that are connected together. Each computer or device can work by itself, but they all work together to do things like process information, store data, or provide services.

It's kind of like having a team of people working on the same project, but each person is in a different place and has their own task to do.But they all communicate and share information with each other to make sure the whole project gets done correctly and efficiently.

Characteristics of Distributed Systems

- Decentralized Control: In a distributed system, control and decision-making authority are decentralized. Each node in the system has a degree of autonomy and can make decisions independently based on local information.

- Distributed Data Management: Data and resources are distributed across multiple nodes in the system. Each node may store a subset of the data or perform specific tasks, contributing to the overall functionality of the system.
- Peer-to-Peer Communication: Communication in a distributed system can occur directly between nodes without the need for a central intermediary. Nodes can exchange information, coordinate actions, and collaborate to achieve shared objectives.
- Fault Tolerance: Distributed systems are often designed to be resilient to failures. Since there is no single point of failure, the system can continue to operate even if individual nodes experience issues or failures.
- Scalability: Distributed systems can be highly scalable, allowing for the addition of new nodes to accommodate increased workload or user demand. This scalability is achieved through the parallelization of tasks across multiple nodes.

Use Cases of Distributed Systems

- Cloud Computing Platforms: Cloud services share resources over data centers. This lets them offer computing on demand. AWS, Azure, GCP are examples.
- Peer-to-Peer (P2P) Networks: Peer-to-peer networks are special kinds of computer networks that allow direct communication between computers without needing a central server. These networks work by letting each computer share its resources and services with other computers on the network
- Distributed Databases: Distributed databases are computer systems that store information across several computers or nodes linked together in a network. Having data spread out like this makes the database highly available, fault-tolerant, and scalable. This means the database can handle many users and requests without crashing or slowing down. It also allows the system to keep working even if some parts fail.

Youtube Video :
Distributed Systems Explained | System Design Interview Basics

Aspect	Centralized System	Distributed System
Control	Centralized control and authority	Decentralized control and authority
Resource Management	All resources managed centrally	Resources distributed across multiple nodes
Communication	Communication flows to central node	Direct communication between nodes
Fault Tolerance	Single point of failure	Redundancy, less vulnerable to single points of failure
Scalability	Limited scalability due to centralization	Highly scalable, new nodes can be added easily
Complexity	Relatively simpler to manage	More complex to manage

Actors and components in Blockchain solution

Actors are the participants who interact with the blockchain network in various roles.

1. Users / Clients
2. Nodes / Peers
3. Miners / Validators
4. Developers
5. Administrators / Network Operators
6. Smart Contracts

1. **Users / Clients**

   Role: End-users who initiate transactions or use blockchain-based applications.

   Example: A person sending cryptocurrency or using a dApp (Decentralized App).
2. **Nodes / Peers**

   Role: Computers connected to the blockchain network that maintain copies of the ledger and participate in consensus.

   Types of Nodes:

   i. Full Node: Stores the entire blockchain and validates transactions and blocks.

   ii. Light Node: Stores only necessary parts of the blockchain, relies on full nodes for validation.
3. **Miners / Validators**

   Role: Participants who validate transactions and add new blocks to the blockchain.

   Mechanisms:

   i. Proof of Work (PoW): Used in Bitcoin, miners solve puzzles.

   ii. Proof of Stake (PoS): Used in Ethereum 2.0, validators are selected based on stake.
4. **Developers**

   Role: Build and maintain blockchain platforms, smart contracts, and applications.

   Tools Used: Solidity (Ethereum), Rust (Solana), Go (Hyperledger), etc.

5. **Administrators / Network Operators**
   Role: Manage permissions, monitor the network, and ensure security (mainly in permissioned blockchains like Hyperledger Fabric).
6. **Smart Contracts**
   Role: Self-executing programs stored on the blockchain that run when predefined conditions are met.
   Used For: Automating workflows (e.g., payments, asset transfers, governance rules).

Components of a Blockchain Solution

1. Distributed Ledger
2. Consensus Mechanism
3. Cryptography
4. Smart Contracts
5. Wallets
6. Blockchain Protocol
7. Peer-to-Peer (P2P) Network
8. Tokens / Cryptocurrencies

1. Distributed Ledger
   Definition: A replicated, synchronized database shared across nodes.
   Purpose: Ensure transparency and data immutability.

2. Consensus Mechanism
   Definition: Protocol to agree on the validity of transactions.
   Common Types:
   i. Proof of Work (PoW)
   ii. Proof of Stake (PoS)
   iii. Practical Byzantine Fault Tolerance (PBFT)
   iv. Delegated Proof of Stake (DPoS)

3. Cryptography
   Role: Ensures security and privacy.
   Types Used:
   i. Hashing (SHA-256): Converts input into a fixed-size hash.

    ii. Public/Private Keys: Used for digital signatures and transaction authorization.

4. Smart Contracts

   Functionality: Automates and enforces rules without intermediaries.

   Languages: Solidity (Ethereum), Chaincode (Hyperledger Fabric).

5. Wallets

   Purpose: Store private/public keys and allow users to send/receive crypto.

   Types:

   i. Hot Wallets (online)
   ii. Cold Wallets (offline, more secure)

6. Blockchain Protocol

   Definition: Set of rules defining how data is structured, added, and communicated.

   Examples: Bitcoin protocol, Ethereum protocol, Hyperledger Fabric protocol.

7. Peer-to-Peer (P2P) Network

   Function: Enables direct communication between nodes without a central server.

   Benefit: Decentralization and fault tolerance.

8. Tokens / Cryptocurrencies

   Purpose: Represent value or utility on the blockchain.

   Example: ETH on Ethereum, BTC on Bitcoin.

Summary

Category	**Element**	**Function**
Actor	User	Initiates transactions or uses dApps
Actor	Node	Maintains blockchain ledger
Actor	Miner / Validator	Validates and adds blocks
Actor	Developer	Builds blockchain systems and smart contracts
Component	Ledger	Stores transaction records
Component	Consensus Mechanism	Ensures agreement on network state
Component	Smart Contracts	Automates execution of logic
Component	Cryptography	Secures data and ensures identity
Component	Wallet	Manages keys and signs transactions

Que. - Compare Centralised, Decentralised and Distributed Systems.

Feature / Aspect	Centralized System	Decentralized System	Distributed System
Control	Single central authority	Multiple authorities (partial control)	No single point of control
Architecture	Hub-and-spoke	Tree or mesh	Fully connected mesh
Decision Making	Centralized	Shared among several entities	Coordinated among independent nodes
Failure Point	Single point of failure (SPOF)	Less prone to total failure	Very fault-tolerant
Performance	Can be fast under light load	Depends on structure	Scalable and resilient
Security	Easy to manage but vulnerable to attacks	Improved by reducing central risks	High, but more complex to secure
Data Storage	Stored in one place (central server)	Multiple data sources	Data distributed across nodes
Example	Traditional Bank Server	Blockchain networks like Ethereum	Torrent, Bitcoin, Cloud Storage (CDN)
Cost of Maintenance	Lower infrastructure, higher central management	Moderate	High (initial setup + maintenance)
Scalability	Limited	Moderate	High