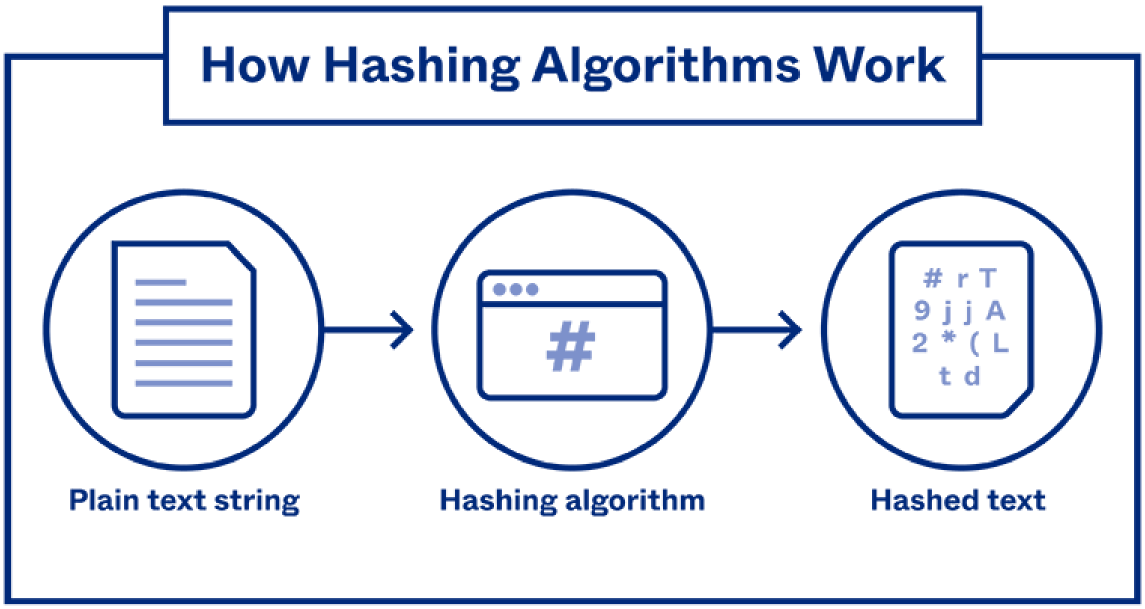
Hashing

- Hashing plays a crucial role in blockchain technology. It is a fundamental concept that contributes to the security, integrity, and immutability of data stored on a blockchain.
- Hashing is a process of converting input data (also known as a message or plaintext) into a fixed-size string of characters, which is typically a hexadecimal number. This output is called a hash value or simply a hash. Hash functions are designed to take an input and produce a unique, seemingly random output, which is based on the input data.



How Hashing Works In Blockchain?

1. Hash Functions:

In the context of blockchain, a hash function is a mathematical algorithm that takes an input (data of arbitrary size) and produces a fixed-size string of characters, which is the hash value. Some commonly used cryptographic hash functions in blockchain include SHA-256 (used in Bitcoin) and Keccak-256 (used in Ethereum).

Er. Harsh Raj
(Assistant Professor, CSE)

2. Data Integrity:

Blockchain uses hashing to ensure the integrity of data within each block. When a block is created, all the transactions within it, along with some additional data (like a timestamp or block number), are concatenated and passed through the hash function. This process generates a unique hash value for that block. If anyone alters even a single bit of data in the block, the resulting hash will be completely different, making it easy to detect tampering.

3. Chaining Blocks:

In a blockchain, each block contains a reference to the hash of the previous block. This reference is called the "previous hash" or "parent hash." When a new block is created, it includes the hash of the previous block in its data. This linking of blocks via their hashes creates a chain of blocks, which is a fundamental feature of blockchain technology. It ensures the chronological order and immutability of transactions.

4. Mining and Proof of Work (PoW):

In PoW-based blockchain networks like Bitcoin, miners compete to solve a complex mathematical puzzle by finding a nonce (a random number). Miners repeatedly change this nonce until they produce a hash value that meets certain criteria. The criteria often involve having a hash that starts with a specific number of leading zeros. This process is known as "finding a proof of work." Once a miner finds a valid proof of work, they can broadcast their block to the network. This adds the block to the blockchain and is the basis for achieving consensus in the network. The PoW system relies on the computational difficulty of finding this proof of work, making it costly and time-consuming to create new blocks. It also secures the network by making it prohibitively difficult for an attacker to control a majority of the network's computational power.

5. Address Generation:

Cryptocurrency addresses in blockchain systems are often derived from a user's public key through a process called "hashing." The user's public key is first hashed to produce a shorter, fixed-length value known as the address. This address is used for sending and receiving cryptocurrencies. Importantly, the original public key remains hidden, preserving user privacy.

Er. Harsh Raj
(Assistant Professor, CSE)

Hashing Algorithm

Hashing algorithms are mathematical functions that take an input (or "message") and return a fixed-size string of characters, which is typically a hexadecimal number.

These algorithms are designed to efficiently transform input data into a unique hash value.

In the context of computer science and cryptography, hashing algorithms have several important properties:

1. **Deterministic:**
   The same input will always produce the same hash value. This property is crucial for data integrity and consistency.

2. **Fast Computation:**
   Hash functions are designed to produce a hash value quickly, even for large inputs.

3. **Fixed Output Length:**
   Regardless of the size or length of the input data, the hash function produces a fixed-size hash value. For example, the SHA-256 algorithm always produces a 256-bit (32-byte) hash value.

4. **Preimage Resistance:**
   Given a hash value, it should be computationally infeasible to determine the original input. In other words, it should be difficult to reverse the hash function.

5. **Collision Resistance:**
   It should be extremely unlikely for two different inputs to produce the same hash value. Collisions can compromise the security of hash functions.

6. **Avalanche Effect:**
   A small change in the input data should result in a significantly different hash value. This property makes it challenging to predict the impact of changes in the input.

Er. Harsh Raj
(Assistant Professor, CSE)

Several widely used hashing algorithms exist, each with its own characteristics and use cases.

A few notable examples are:

1. **SHA-256 (Secure Hash Algorithm 256-bit):**
   This cryptographic hash function is commonly used in blockchain technologies like Bitcoin. It produces a 256-bit (32-byte) hash value.

2. **MD5 (Message Digest Algorithm 5):**
   MD5 was once widely used but is now considered weak due to vulnerabilities to collision attacks. It produces a 128-bit (16-byte) hash value.

3. **SHA-1 (Secure Hash Algorithm 1):**
   SHA-1 is also considered weak due to vulnerabilities, and it is being phased out in favour of stronger hash functions. It produces a 160-bit (20-byte) hash value.

4. **SHA-3 (Secure Hash Algorithm 3):**
   SHA-3 is part of the Keccak family of hash functions and provides a high level of security. It comes in various bit-length versions, such as SHA-3-256, SHA3-512, etc.

5. **bcrypt:**
   Bcrypt is a cryptographic hash function designed for securely hashing passwords. It includes a salt (a random value) to protect against rainbow table attacks.

6. **Scrypt:**
   Like bcrypt, Scrypt is designed for secure password hashing. It is computationally intensive and memory-hard to resist brute-force and dictionary attacks.

Note: The choice of a hashing algorithm depends on the specific use case and security requirements. For secure applications, it is essential to use well-vetted cryptographic hash functions like SHA-256 or SHA-3, while weaker algorithms like MD5 and SHA-1 should be avoided due to their susceptibility to attacks.

Er. Harsh Raj
(Assistant Professor, CSE)