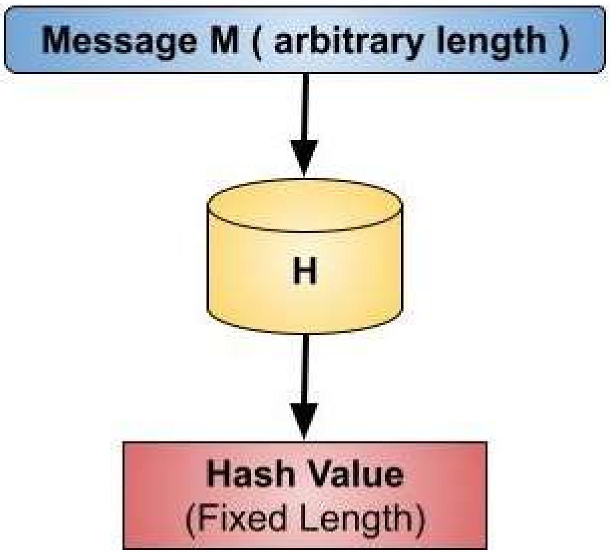
Cryptography - Hash functions

A hash function in cryptography is like a mathematical function that takes various inputs, like messages or data, and transforms them into fixed-length strings of characters. Means the input to the hash function is of any length but output is always of fixed length. This is like compressing a large balloon into a compact ball.

The importance of this process lies in its generation of a unique "fingerprint" for each input. Any minor alteration in the input results in a substantially different fingerprint, a quality known as "collision resistance."

Hash functions play a crucial role in various security applications, including password storage (hash values instead of passwords), digital signatures, and data integrity checks. Hash values, or message digests, are values that a hash function returns. The hash function is shown in the image below −



Key Points of Hash Functions

- Hash functions are mathematical operations that "map" or change a given collection of data into a fixed-length bit string that is referred to as the "hash value."
- Hash functions have a variety of complexity and difficulty levels and are used in cryptography.
- Cryptocurrency, password security, and communication security all use hash functions.

Operation of Cryptographic Hash Functions

In computing systems, hash functions are frequently used data structures for tasks like information authentication and message integrity checks. They are not easily decipherable, but because they can be solved in polynomial time, they are regarded as cryptographically "weak".

Typical hash functions have been improved with security characteristics by cryptographic hash functions, which make it more challenging to decipher message contents or recipient and sender information.

Specifically, cryptographic hash functions display the following three characteristics −

- The hash function are called as "collision-free." As a result, no two input hashes should be equal to the same output hash.
- They are hidden. A hash function's output should make it difficult to figure out the input value from it.
- They should to be friendly to puzzles. The selection of an input that generates a predetermined result needs to be difficult. As such, the input needs to be taken from as wide as possible.

Properties of hash functions

To be a reliable cryptographic tool, the hash function should have the following properties −

Pre-Image Resistance

- According to this feature, reversing a hash function should be computationally difficult.
- In other words, if a hash function h generates a hash value z, it should be difficult to identify an input value x that hashes to z.

- This feature defends against an attacker attempting to locate the input with just the hash value.

Second Pre-Image Resistance

- This property says that given an input and its hash, it should be difficult to find another input with the same hash.
- In other words, it should be challenging to find another input value y such that h(y) equals h(x) if a hash function h for an input x returns the hash value h(x).
- This feature of the hash function protects against an attacker who wants to replace a new value for the original input value and hash, but only holds the input value and its hash.

Collision Resistance

- This feature says that it should be difficult to identify two different inputs of any length that produce the same hash. This characteristic is also known as a collision-free hash function.
- In other words, for a hash function h, it is difficult to identify two distinct inputs x and y such that h(x)=h(y).
- A hash function cannot be free of collisions because it is a compression function with a set hash length. The collision-free condition simply indicates that these collisions should be difficult to locate.
- This characteristic makes it very hard for an attacker to identify two input values that have the same hash.
- Furthermore, a hash function is second pre-image resistant if it is collision-resistant.

Efficiency of Operation

- Computation of h(x) for any hash function h given input x can be an easy process.
- Hash functions are computationally considerably faster than symmetric encryption.

Fixed Output Size

Hashing generates an output of a specific length, regardless of the input size, and heips to make an output of the same size from different input sizes.

Deterministic

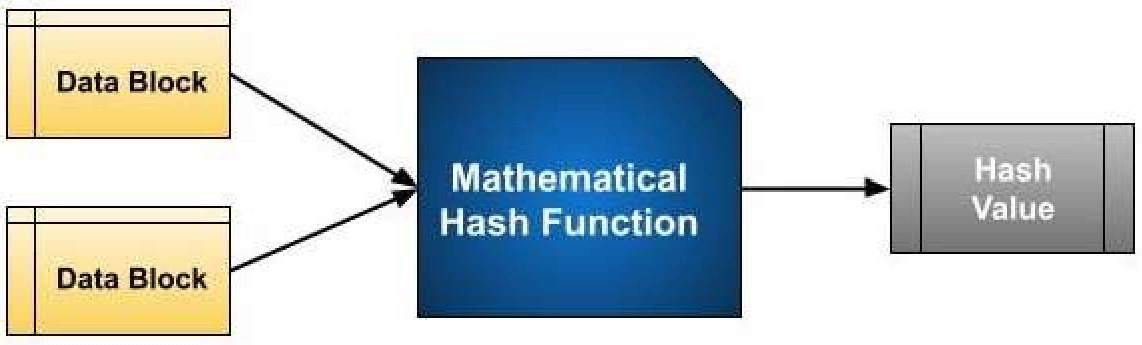
For a given input, the hash function consistently produces the same output, like a recipe that always yields the same dish when followed precisely.

Fast Computation

Hashing operations occur rapidly, even for large amounts of data sets.

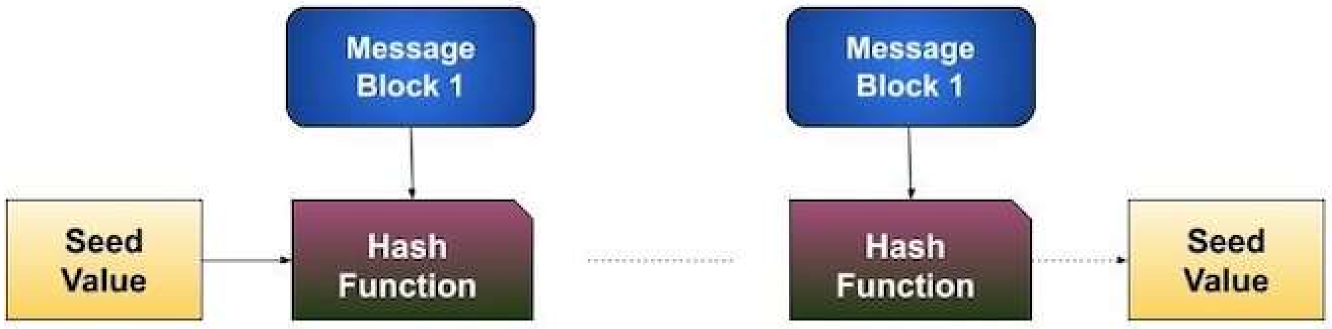
Design of Hashing Algorithms

Hashing essentially involves a mathematical function that takes two data blocks of fixed size and converts them into a hash code. The function is a key part of the hashing algorithm. The length of these data blocks differ according to the algorithm used. Usually, they range from 128 bits to 512 bits. Below is an example of a hash function −



Hashing algorithms use a sequence of rounds, similar to a block cipher, to process a message. In each round, a fixed-size input is used, which usually combines the current message block and the result from the previous round.

This process continues for multiple rounds until the entire message is hashed. A visual representation of this process is provided in the illustration below.



Due to the interconnected nature of hashing, where the output of one operation affects the input of the next, even a minor change (a single bit difference) in the original message can drastically alter the final hash value.

This phenomenon is known as the avalanche effect. Additionally, it's crucial to distinguish between a hash function and a hashing algorithm. The hash function itself takes two fixed-length binary blocks of data and generates a hash code.

A hashing algorithm, on the other hand, establishes how the message is divided into blocks and how the outcomes of multiple hash operations are combined.

Popular Hash Functions

Hash functions play an important role in computing, providing versatile capabilities like: Quick retrieval of data, Secure protection of information (cryptography), Ensuring data remains unaltered (integrity verification). Some commonly used hash functions are −

Message Digest (MD)

For a number of years, MD5 was the most popular and often used hash function.

- The hash functions MD2, MD4, MD5, and MD6 are members of the MD family. It was adopted as the RFC 1321, Internet Standard. It is a 128-bit hash function.
- In the software industry, MD5 digests are frequently used to ensure the integrity of transferred files. To enable users to compare the checksum of the downloaded file with the pre-computed MD5 checksum, file servers frequently provide this feature.
- In 2004, collisions were found in MD5. It was claimed that an analytical attack using a computer cluster was successful in under one hour. Since MD5 was compromised by this collision attack, using it is no longer recommended.

Secure Hash Function (SHA)

The four SHA algorithms which make up the SHA family are SHA-0, SHA-1, SHA-2, and SHA-3. Despite coming from the same family, the structure of it differs.

- The National Institute of Standards and Technology (NIST) released the first iteration of the 160-bit hash algorithm, known as SHA-0, in 1993. It did not gain much popularity and had few drawbacks. SHA-1 was created later in 1995 to address perceived flaws in SHA-0.
- SHA-1 is the most widely used of the existing SHA hash functions. It is used in most of the applications and protocols including Secure Socket Layer (SSL) security.

- In 2005, a technique was discovered for SHA-1 collision detection that can be used in a realistic time frame. So it is doubtful on SHA-1's long-term usability.
- SHA-224, SHA-256, SHA-384, and SHA-512 are the other four SHA variants in the SHA-2 family, which vary based on the number of bits in their hash value. The SHA-2 hash function has not yet been the target of any effective attacks
- Though SHA-2 is a strong hash function. Though significantly different, its basic design still follows the design of SHA-1. NIST thus demanded the creation of new competitive hash function designs.
- The Keccak algorithm was selected by the NIST in October 2012 to replace the SHA-3 standard. Keccak has several advantages, including effective operation and strong attack resistance.

CityHash

CityHash is another non-cryptographic hash function that is designed for fast hashing of large amounts of data. It is optimized for modern processors and offers good performance on both 32-bit and 64-bit architectures.

BLAKE2

BLAKE2 is a fast and secure hash function that improves upon SHA-3. It is widely used in applications like cryptocurrency mining that need fast hashing. There are two types of BLAKE2 −

- **BLAKE2b** − Best for 64-bit computers, it produces hash values up to 512 bits long.
- **BLAKE2s** − Best for smaller computers (8-32 bits), it produces hash values up to 256 bits long.

CRC (Cyclic Redundancy Check)

CRC (Cyclic Redundancy Check) is a technique used to detect errors in data transfer. It involves adding a special value called a checksum to the end of a message. This checksum is calculated based on the message's content and is included during transmission.

When the data is received, the recipient recalculates the checksum using the same method. If the new checksum matches the original one, it's likely that the message was transmitted without errors. While CRC is effective for error detection, it's not a security measure. It is primarily used to ensure the integrity of data during transmission, not to protect it from unauthorized access or modification.

MurmurHash

MurmurHash is a speedy and effective hash function that is not meant for security. It is great for things like hash tables but not for tasks that need protection against collisions (situations where different inputs produce the same hash).

Standard Length

Hashing involves converting a data set of any size into a shorter, fixed-length output using a mathematical formula.

Table I: Different Hash Functions

In table I, the message "CFI" is converted into hash values using three algorithms: MD5, SHA-1, and SHA-256. Each algorithm produces a unique output hash with a fixed length. MD5 generates a hash with 32 hexadecimal characters, SHA-1 with 40 characters, and SHA-256 with 64 characters.

Input Message	Hash Function	Output (Hash Value)
CFI	MD5 (128-bit, 16-byte) 32 characters	3A10 0B15 B943 0B17 11F2 E38F 0593 9A9A
CFI	SHA-1 (160-bit, 20-byte) 40 characters	569D C9F0 7B48 7F58 9241 AD4C 5C28 7DA0 A448 8D08
CFI	SHA-256 (256-bit, 32-byte) 64 characters	F3ED 0867 48FF 3641 3091 0BB6 6293 7080 2958 B5A2 52AF F364 1FC5 07FD E80D 9929

Table II: Using the Same Hash Function (SHA-1) with different Inputs

Besides the data (input) used, a hash function consistently generates a hash value with a fixed number of characters. As shown in Table II, different messages inputted into the same hash function (SHA-1 in this case) consistently produce output values of 40 hexadecimal characters in length.

Input Message	Hash Function	Output (Hash Value)
CFI	SHA-1	569D C9F0 7B48 7F58 9241 AD4C 5C28 7DA0 A448 8D08
Corporate FI	SHA-1	82C0 5EDC 608F AA08 8EE0 BDD8 8E22 3B38 CA38 82CC

CF Input	SHA-1	2013 85FC EEE4 F73D 07F0 4F2A A4CB BOE9 12BF BBB8
CFI 1	SHA-1	C501 23CE 8BB2 A42D 5BB4 4DA7 3FC2 3B3D 62F5 14A5

Applications of Hash Functions

Based on its cryptographic characteristics, the hash function has two direct uses.

Password Storage

Hash functions provide protection to password storage. Instead of storing passwords in clear, mostly all login processes store the hash values of passwords in the file.

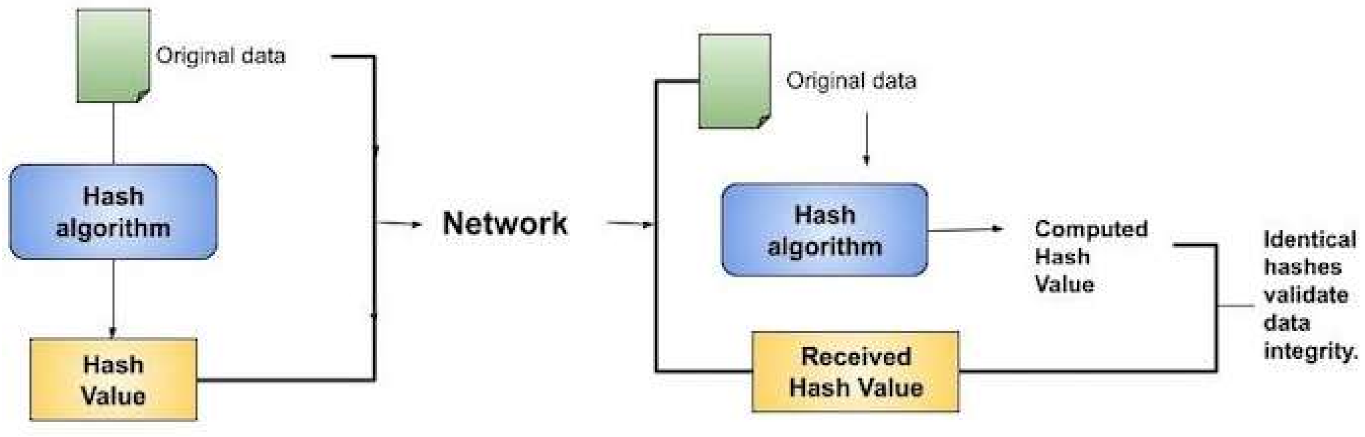
The Password file is a table of pairs in the format (user id, h(P)).

Even if an attacker has access to the password, all they can see is the hashes of the passwords. Because the hash function contains the pre-image resistance feature, he cannot use it to log in or get the password from it.

Data Integrity Check

Data integrity checks, commonly using hash functions, provide assurances about the accuracy of data files by creating checksums. This method allows users to detect any alterations made to the original file.

However, it does not guarantee the authenticity of the file. An attacker could potentially modify the entire file and generate a new hash, sending it to the receiver. This integrity check is only effective if the user trusts the file's original source.



Hashing vs Encryption

Encryption transforms data into a disguised form, requiring a cipher (key) to decipher and read it. Encryption and decryption are reversible processes enabled by the cipher. Encryption is used with the goal of later deciphering the data.

Hashing transforms data of any size into a fixed-length output. Unlike encryption, hashing is typically a one-way function. The high computational effort needed to reverse a hash makes it difficult to retrieve the original data from the hashed output.

Data is protected during transmission by encryption, which stops unwanted access. By comparing the data to a distinct fingerprint (hash) created from the original data, hashing ensures the integrity of the data. Encryption keeps data confidential, while hashing ensures authenticity by detecting any modifications.