**Hashemite University**

**Prince Al-Hussein bin Abdullah II Faculty for Information Technology**

**Department of Computer Information Systems**

**Assignment: Information Systems Security**

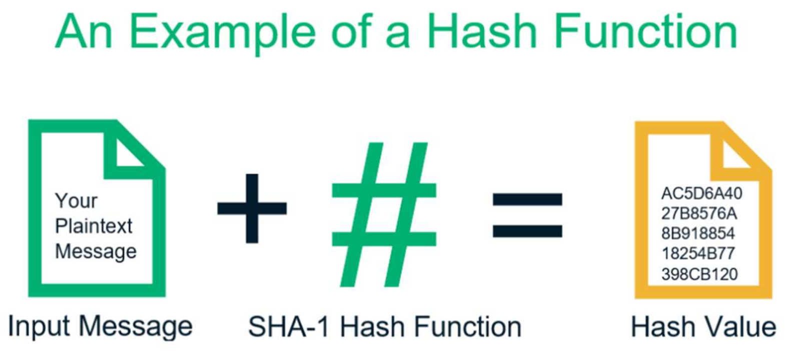
# Topic title: cryptographic hash function (CHF)

**Date: 25/5/2022**

| | | | | |
|---|---|---|---|---|
| **Student Name** | **Yousef Hisham Ahmad Sha'ban** | **Ahmad Mustafa Taher Khamis** | **Abdelrahman Ihssan Ahmad AbdelBagi** | **Mohammad Salah Mousa AlMari** |
| **Student ID** | **1932107** | **1934759** | **1934408** | **1939117** |
| **Instructor Name** | **Dr. Ala' Said Mohammad Mughaid** | | | |
| **Section ID** | **Section 1** | | | |

## Table of content:

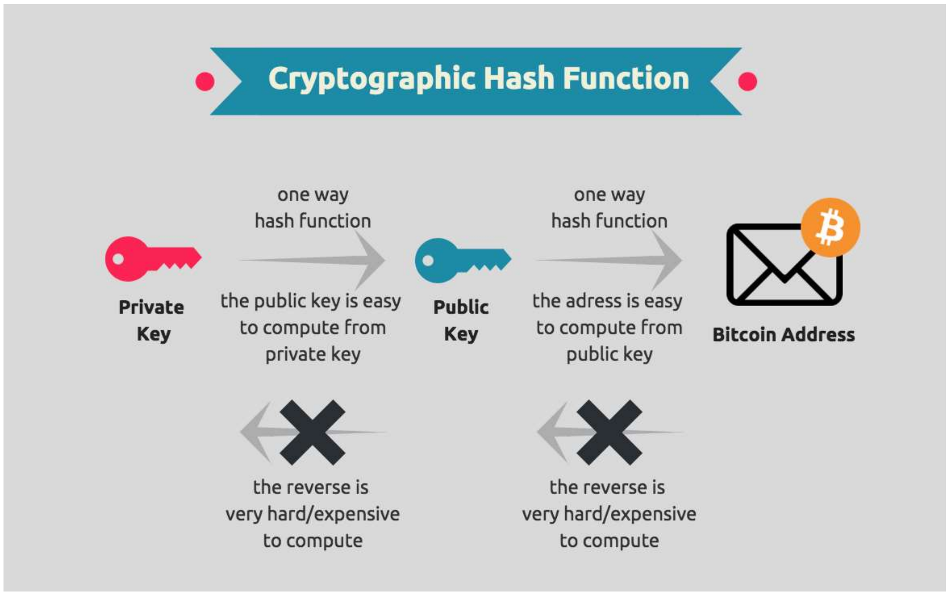
## 1. Introduction:


An Example of a Hash Function
Your Plaintext Message
AC5D6A40 27B8576A 8B918854 18254B77 398CB120
Input Message
SHA-1 Hash Function
Hash Value

**Cryptographic hash function (CHF): is a function that maps a message of any length into a fixed-length hash value that serves as the authenticator.**

**In other words, a cryptographic hash function (CHF): is an algorithm that takes an arbitrary amount of data input—a credential—, and produces a fixed-size output of enciphered text called a "hash value", or just "hash".**

**It's a single direction work, making it extraordinarily difficult to reverse in order to recreate the information used to make it, it's providing message integrity.**


Cryptographic Hash Function
one way hash function
the public key is easy to compute from private key
Private Key
Public Key
one way hash function
the adress is easy to compute from public key
Bitcoin Address
the reverse is very hard/expensive to compute
the reverse is very hard/expensive to compute

## 2. Applications of cryptographic hash function

**A cryptographic hash function (CHF) is an equation used to verify the validity of data. It has many applications, notably in information security (e.g. user authentication), Here we will give some examples:**

### 2.1. Message Authentication

**It is a mechanism or service used to verify the integrity of a message; it assures that data received is exactly as sent (i.e., contains no modification, insertion, deletion, or replay).**

### 2.2 Digital Signatures

**The operation of the digital signature is similar to that of the MAC. In the case of the digital signature, the hash value of a message is encrypted with a user's private key. Anyone who knows the user's public key can verify the integrity of the message that is associated with the digital signature. In this case, an attacker who wishes to alter the message would need to know the user's private key.**

### 2.3 Data Integrity Check

**To maintain data integrity, the sender sends both the message and its hash value to the user. The receiver then checks whether the hash value of the message is the same as the hash value sent by the sender. This ensures that no modification has been done to the data while being transmitted. This process of integrity checks can be seen in email systems, messaging applications, etc.**

### 2.4 Other applications

**Such as: pseudo number generation, digital steganography, digital time stamping, Password Verification, Compiler Operation, etc.**

## 3. Why we used cryptographic hash function?

**- It combines the message-passing capabilities of hash functions with security properties.**

**- It has variable levels of complexity and difficulty.**

**- It's used for cryptocurrency, password security, and message security.**

**- It checks the integrity of messages and authenticates information.**

**- It adds security features to typical hash functions, making it more difficult to detect the contents of a message or information about recipients and senders.**

## 4. Differences among preimage resistant, second preimage resistant, and collision resistant?

### 4.1 Pre-Image Resistance

**- This property means that it should be computationally hard to reverse a hash function.**

**- In other words, if a hash function h produced a hash value z, then it should be a difficult process to find any input value x that hashes to z.**

**- This property protects against an attacker who only has a hash value and is trying to find the input.**

### 4.2 Second Pre-Image Resistance

**- This property means that given an input and its hash, it should be hard to find a different input with the same hash.**

**- In other words, if a hash function h for an input x produces a hash value h(x), then it should be difficult to find any other input value y such that h(y) = h(x).**

**- This property of the hash function protects against an attacker who has an input value and its hash, and wants to substitute a different value as a legitimate value in place of the original input value.**

### 4.3 Collision Resistance

**- This property means it should be hard to find two different inputs of any length that result in the same hash. This property is also referred to as a collision-free hash function.**

**- In other words, for a hash function h, it is hard to find any two different inputs x and y such that h(x) = h(y).**

**- Since, a hash function is a compressing function with a fixed hash length, it is impossible for a hash function not to have collisions. This property of collision-free only confirms that these collisions should be hard to find.**

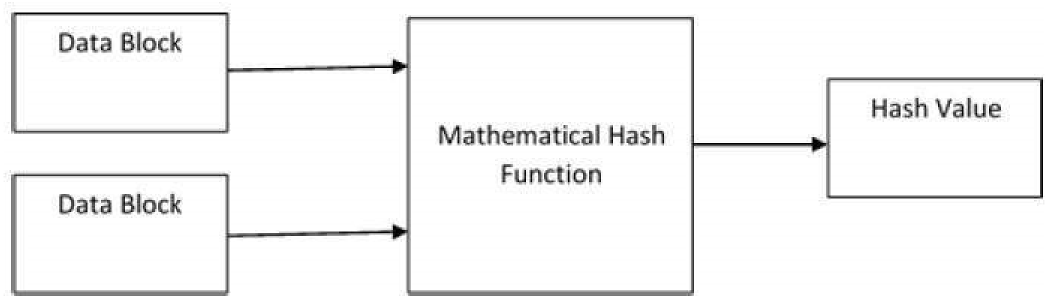
**- This property makes it very difficult for an attacker to find two input values with the same hash.**

**- Also, if a hash function is collision-resistant then it is second pre-image resistant.**

## 5. Structure and Design of Hashing Algorithms

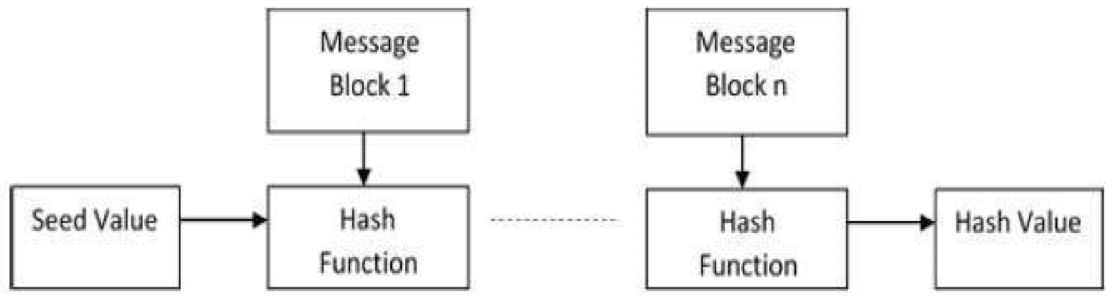
**At the heart of hashing is a mathematical function that operates on two fixed-size blocks of data to create a hash code. This hash function forms part of the hashing algorithm.**

**The size of each data block varies depending on the algorithm. Typically, the block sizes are from 128 bits to 512 bits. The following image explains the hash function:**



**The hashing algorithm involves rounds of the above hash function as a block cypher. Each round takes an input of a fixed size, typically a combination of the most recent message block and the output of the last round.**

**This process is repeated for as many rounds as are required to hash the entire message. A schematic of a hashing algorithm is explained in the following image:**



**Since, the hash value of the first message block becomes an input to the second hash operation, the output of which alters the result of the third operation, and so on.**

**This effect is known as the "avalanche effect of hashing"**

## 6. How ciphers block chaining to construct a cryptographic hash function?

**A number of proposals have been made for hash functions based on using a cipher block chaining technique, but without using the secret key. One of the first such proposals was that of Rabin [RABI78]. Divide a message M into fixed-size blocks M1, M2, Á, MN and use asymmetric encryption system such :as DES to compute the hash code G as**

$$H_0 = \text{initial value}$$
$$H_i = \mathrm{E}(M_i, H_{i-1})$$
$$G = H_N$$

**This is similar to the CBC technique, but in this case, there is no secret key. As with any hash code, this scheme is subject to the birthday attack, and if the encryption algorithm is DES and only a 64-bit hash code is produced, then the system is vulnerable.**

**Furthermore, another version of the birthday attack can be used even if the opponent has access to only one message and its valid signature and cannot obtain multiple signatures. Here is the scenario: We assume that the opponent intercepts a message with a signature in the form of an encrypted hash code and that the unencrypted hash code is m bits long.**

**1- Use the algorithm defined at the beginning of this subsection to calculate the unencrypted hash code G.**

**2- Construct any desired message in the form of Q1, Q2 … $Q_{N-2}$.**

**3- Compute Hi = E($Q_i$, $H_{i-1}$) for 1 … i … (N – 2).**

**4- Generate 2^(m/2) random blocks; for each block X, compute E(X, $H_{N-2}$). Generate an additional 2^(m/2) random blocks; for each block Y, compute D(Y, G), where D is the decryption function corresponding to E.**

**5- Based on the birthday paradox, with a high probability there will be an X and Y such that E(X, $H_{N-2}$) = D(Y, G).**

**6- Form the message Q1, Q2, Á, $Q_{N-2}$, X, Y. This message has the hash code G and therefore can be used with the intercepted encrypted signature.**

This form of attack is known as a "meet-in-the-middle" attack. A number of researchers have proposed refinements intended to strengthen the basic block chaining approach. For example, Davies and Price [DAVI89] describe the variation:

$$H_i = E(M_i, H_{i-1}) \oplus H_{i-1}$$

Another variation, proposed in [MEYE88], is

$$H_i = E(H_{i-1}, M_i) \oplus M_i$$

However, both of these schemes have been shown to be vulnerable to a variety of attacks [MIYA90]. More generally, it can be shown that some form of birthday attack will succeed against any hash scheme involving the use of cipher block chaining without a secret key, provided that either the resulting hash code is small enough (e.g., 64 bits or less) or that a larger hash code can be decomposed into independent sub codes [JUEN87].

Thus, attention has been directed at finding other approaches to hashing.

## 7. The operation of SHA-512?

### 7.1 Append: Padding bits

The first step is to carry out the padding function, in which we append a certain number of bits to the plaintext message to increase its length, which should be exactly 128 bits less than an exact multiple of 1024.

When we are appending these bits at the end of the message, we start with a ‘1’ and then keep on adding ‘0’ till the moment we reach the last bit that needs to be appended under padding and leave the 128 bits after that.

### 7.2 Append: Length bits

Now, we add the remaining 128 bits left to this entire block to make it an exact multiple of 1024, so that the whole thing can be broken down into ’n’ number of 1024 blocks of message that we will apply our operation on. The way to calculate the rest of the 128 bits is by calculating the modulo with $2^{64}$.

Once done, we append it to the padded bit and the original message to make the entire length of the block to be “n * 1024” in length.

## 7.3 Initialize the buffers

**Now, that we have "n x 1024" length bit message that we need to hash, let us focus on the parts of the hashing function itself. To carry on the hash and the computations required, we need to have some default values initialized.**

**These are the values of the buffer that we will need. There are other default values that we need to initialize as well. These are the values for the 'k' variable which we will be using.**

**The reason for initiating these values will be clear to you in the very next step that we will explore.**

## 7.4 Compression function

**Now we need to take a closer look at the hash function to see what's going on. To begin, we partition(divide) a 1024-bit message into 'n' bits.**

**We'll do 80 rounds, and the input, W, will be derived from 1024 bits, which will be divided into 16 pieces. The message in plaintext is the value of W from 0 to 15, whereas the values of W from 16 to 79 are calculated from the previous 16 blocks.**

**Now that we know the methodology to obtain the values of W for 0 to 79 and already have the values for K as well for all the rounds from 0 to 79 we can proceed ahead and see where and how we do put in these values for the computation of the hash.**

**We have the values and formulas for each of the functions carried out we can perform the entire hashing process. These are the functions that are performed in each of the 80 rounds that are performed over and over for 'n' number of times.**

## 7.5 Output

**The output from every round act as an input for the next round, and this process keeps on continuing till the last bit of the message goes through one of the rounds, and the output from the last block of the last message is the hash code. The length of the output is 512 bits.**

## 7.6 Conclusion

**The SHA-512 hashing algorithm is currently one of the best and most secure hashing algorithms after hashes like MD5 and SHA-1 have been broken down. Due to its complicated nature, it is not widely accepted that SHA-256 is a general standard, but the industry is slowly moving towards this hashing algorithm.**

## 8. Overview of birthday paradox and birthday attack?

### 8.1 Birthday attack

**The Birthday attack is a type of cryptographic attack that cracks the algorithms of mathematics by finding matches in the hash function. The method relies upon the birthday paradox, through which the chance of sharing one birthday by two people is quite higher than it appears. In the same way, the chance of collision detection is also higher within the target hash function, which thereby enables the attacker to find similar fragments by the use of a few iterations. The attack is mostly used to abuse communication between the two parties. The nature of the attack is dependent upon the collisions found among random attacks as well as the degree of the permutation.**

### 8.2 Birthday paradox

**The probability theory describes the birthday paradox problem as if you carry the 'n' number of people in the room; there is the probability that some of them might have their birthdays on a similar day. But, one important point to consider is that we do not focus on the matching birth dates but rather two people sharing one birthday is the major point.**

**So, the birthday attacks utilize the approach of probability to minimize the difficulty in the matched collision and to get the approximate hash collision risk in a certain number. It also shows that discovering a particular hash collision is the most problematic thing instead of finding the matched hash collisions with similar values.**

### 8.3 Calculating birthday paradox

**Most people assume 183 as it is actually half of every possible birthday and it seems intuitive as well. However, it is not the game of intuition so, for the calculation, there are few assumptions to be made.**

**First of all, we will not consider the leap year as it will simplify the math and doesn't change results. We will also consider that every birthday has an change of occurring.**

**So, start with one person and then keep on adding people one at a time to show how this calculation is working. For the calculations, this is easy to calculate the probability that no other person will share the birthday.**

**This probability will be subtracted from the other one derived that includes at least two people will share one birthday.**

**The chance of one match at least = 1 – the probability of no match.**

**For the first person, there is a 365/365 chance that there is no other shared birthday.**

$$1 - \left(\frac{365}{365}\right) = 0$$

**Now, add the second person. The first person has all chances of one birthday, but the second person will have only a 364/365 chance of not sharing a similar day birthday. This goes:**

$$1 - \left(\frac{365}{365}\right) \times (\frac{364}{365}) = 0.0027$$

**Similarly, for the third person, the probability will be 363/365 of not sharing a similar day birthday.**

$$1 - \left(\frac{365}{365}\right) \times \left(\frac{364}{365}\right) \times (\frac{363}{365}) = 0.0082$$

**So, the pattern for the entire population will be:**

$$1 - \left(\frac{365}{365}\right) \times \left(\frac{364}{365}\right) \times \left(\frac{363}{365}\right) \times .. \times (\frac{365 - n\ + 1}{365})$$

**When the probabilities are known, the answer to the birthday problem becomes a 50.7% chance of people sharing people in a total of 23 people group.**

**If the group size is increased, the probability will be reduced. Varying group size will have a different effect on the probability.**

## 8.4 How to prevent the Birthday attack?

**To prevent the birthday attack, there is the possibility that the length of the output for the hash function of the signature scheme can be selected to be large enough such that the chance of a birthday attack becomes computationally impossible.**

**Along with using the extended bit length, the signer can also prevent the attack if make some inoffensive but random changes to the document before it is signed and keep the contract copy under possession. Such that he can demonstrate within the court that the signature matches the contract.**

## 8.5 Final Thoughts

**The birthday attack cracks the mathematics algorithm by its matching in the hash function. The birthday attack is best calculated with the probability theory. However, the attack can be prevented by increasing the bit length and if the signer makes some random changes within the document.**

## 9. References:

### 9.1 Introduction:

https://www.synopsys.com/blogs/software-security/cryptographic-hash-functions/

https://www.hypr.com/cryptographic-hash-function/

### 9.2 Applications of cryptographic hash function:

https://www.includehelp.com/cryptography/applications-of-hash-function.aspx

https://www.brainkart.com/article/Applications-of-Cryptographic-Hash-Functions_8446/

### 9.3 Why we used cryptographic hash function:

https://www.investopedia.com/news/cryptographic-hash-functions/

### 9.4 Differences among preimage resistant, second preimage resistant, and collision resistant:

https://www.tutorialspoint.com/cryptography/cryptography_hash_functions.htm

### 9.5 Structure and Design of Hashing Algorithms:

https://www.tutorialspoint.com/cryptography/cryptography_hash_functions.htm

### 9.6 How ciphers block chaining to construct a cryptographic hash function:

https://www.brainkart.com/article/Hash-Functions-Based-on-Cipher-Block-Chaining_8449/

### 9.7 The operation of SHA-512:

https://infosecwriteups.com/breaking-down-sha-512-algorithm-1fdb9cc9413a

### 9.8 Overview of birthday paradox and birthday attack:

https://www.chubbydeveloper.com/birthday-attack/