CASE STUDY ON HASH FUNTIONS

Saran Shabu

MY.SC.I5MCA21039

Int. MCA Sem 6

Arjun U Menon

MY.SC.I5MCA21010

Int. MCA Sem 6

1. **Introduction**

**1.1. Importance of Cryptographic Hash Functions:**

* Data Integrity: Cryptographic hash functions act like digital fingerprints for data. Any change to the data, however small, will result in a completely different hash value. This allows us to verify if data has been tampered with during storage or transmission.
* Authentication: Hash functions are used to create message authentication codes (MACs) that combine a message with a secret key. The receiver can then recalculate the MAC using the same function and key and compare it to the received MAC. This ensures the message originated from a trusted source and hasn't been altered.
* Confidentiality (indirectly): While hash functions don't directly encrypt data, they play a role in securing passwords. Passwords are never stored in plain text. Instead, a hash of the password is stored. If a hacker steals the stored password hashes, they cannot easily reverse the hash to retrieve the original password.
* Non-repudiation (indirectly): Similar to confidentiality, hash functions are used with digital signatures to achieve non-repudiation. A digital signature is created by signing a message digest (hash) with a private key. Anyone with the corresponding public key can verify the signature, proving the message originated from a specific holder of the private key and hasn't been tampered with.

**1.2. Building Blocks of Cryptographic Systems:**

* Random Generators: Cryptographic systems rely heavily on randomness. Random generators produce unpredictable sequences of bits that are crucial for creating strong encryption keys, salts (random values added to passwords for extra security), and other cryptographic elements.
* One-way Functions: These are mathematical functions that are easy to compute in one direction (from input to output) but infeasible to reverse (from output back to input). Cryptographic hash functions are a specific type of one-way function with additional security properties.

**1.3. One-way Functions and Security Objectives:**

One-way functions are versatile tools in cryptography and are used to achieve various security objectives:

* Confidentiality: One-way functions are used to create encryption algorithms. The encryption key acts as a secret "password" to scramble the data. It's easy to encrypt data with the key, but extremely difficult to decrypt it without knowing the key (which is essentially reversing a one-way function).
* Authentication: As mentioned earlier, one-way functions are the foundation for creating message authentication codes (MACs) used for message integrity and sender verification.
* Integrity: By relying on the avalanche effect (small changes in the input lead to significant changes in the output) of one-way hash functions, we can ensure data hasn't been modified.
* Non-repudiation: Digital signatures, which utilize one-way functions, provide non-repudiation by mathematically linking a message to its signer.

1. **Classification of Cryptographic Systems:**

Cryptographic systems are classified based on the level of security they offer against attackers. Here's an explanation of the three categories you mentioned:

* 1. **Unconditional Security:**

This is the strongest form of security, but unfortunately, it's very rare in practical applications. A cryptosystem is considered unconditionally secure if:

* No matter how much computational power an attacker possesses, they cannot break the system and recover the plaintext from the ciphertext.
* The attacker gains no advantage from having access to both the plaintext and ciphertext.
* The secrecy of the key is irrelevant; even if the attacker knows the key, they still cannot decrypt the message.

The only known unconditionally secure cryptosystems are based on one-time pads. In a one-time pad, the key is a random string of bits that is the same length as the message. The key is used only once and then discarded. However, one-time pads are impractical for most applications due to the difficulty of securely generating and distributing long random keys.

* 1. **Provable Security:**

A cryptosystem is considered provably secure if the security of the system can be mathematically proven to be related to the difficulty of a well-understood mathematical problem. In other words, breaking the cryptosystem is demonstrably as hard as solving a specific hard mathematical problem. For example, the security of many public-key cryptosystems (like RSA) is based on the difficulty of factoring large prime numbers. If an attacker can efficiently factor large prime numbers, they can break the cryptosystem. However, the current state of mathematics suggests that factoring large primes is computationally expensive.

Provable security provides a strong theoretical guarantee of security, but it doesn't guarantee that the system is unbreakable. New mathematical discoveries or advancements in computing power could potentially break the system in the future.

* 1. **Computational Security:**

This is the most common category for cryptographic systems in real-world applications. A cryptosystem is considered computationally secure if:

* An attacker with current computational resources (processing power, time, etc.) cannot break the system within a reasonable timeframe.
* The effort required to break the system is significantly greater than the value of the information being protected.
* The security of computationally secure systems relies on the assumption that attackers don't have unlimited resources. Advancements in computing power could potentially make these systems vulnerable in the future.

Therefore, it's crucial to choose cryptographic systems with appropriate security levels based on the sensitivity of the information being protected and the potential resources an attacker might have.

1. **Need for One-way functions**

* ﻿﻿Confidentiality is one of the 'must have features of cryptographic systems.
* ﻿﻿We need functions that transform the message to make it unreadable, and, if somebody hears' the message, should not able to understand or interpret it.
* ﻿﻿This means that we need functions that can be computed one way but not the inverse:One-way functions.
* ﻿﻿Cryptography needs one way functions because of the main property of being hard to compute the inverse of the function.
* ﻿﻿We have to be very careful with because it involves mathematical issues which are not fully solved or known.

1. **Hard Problems**

* There are the problems that can be solved in polynomial time (or sub-exponential), it means, that the time it would take to solve them is known, and this time grows in a polynomial way as the problem becomes more complicated (called P problems).
* ﻿﻿And there are the other problems, called NP that are problems whose solutions can be checked in polynomial time.
* ﻿﻿In order to solve the big question P = NP? We should find a NP complete problem that can be solved in polynomial time.
* ﻿﻿Then we would be able to reduce all other NP problems to our solution and solve them.

1. **Hash Functions**

A hash function is a function of the form:

**h(x) = y where, x € Z and y E Zn**

* ﻿﻿The calculation is easy and fast
* ﻿﻿The output of the functions is usually smaller than the input (Zn)
* ﻿﻿The input can be any size while the output is usually of a fixed size The result of the hash function is also called the digest

1. **Properties of Hash Functions**

**Non-reversibility or One-way Function:**

This is a fundamental property. It means that given a hash value (digest), it should be computationally infeasible to determine the original input data (password) that generated it. This prevents attackers from easily retrieving passwords even if they steal the stored hash values.

**Diffusion or Avalanche Effect:**

This property ensures high sensitivity to changes in the input. A small change in the original password (even a single bit) should result in a significant change in the resulting hash value. Ideally, this change should affect roughly half the bits in the hash. This makes it extremely difficult for attackers to guess passwords by making slight variations based on leaked password hashes.

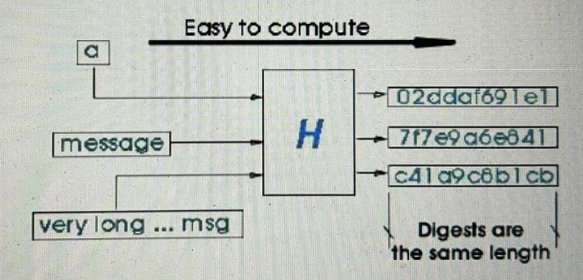
**Determinism:**

This property ensures consistency. For a given input (password), the hash function should always produce the same output (hash value). This allows for reliable verification. When a user enters their password, the system recalculates the hash and compares it to the stored hash. If they match, the password is verified.

**Collision Resistance:**

This property ensures uniqueness. It's very difficult, if not impossible, to find two different inputs (passwords) that produce the same hash value (collision). This is crucial to prevent attackers from creating a fake password that hashes to the same value as a legitimate password, granting them unauthorized access.

**Non-predictable:**

This property ensures the hash value cannot be easily guessed based on the password itself. The hash function should be designed to make it hard to predict the output even with some knowledge of the input. This prevents attackers from exploiting patterns in passwords to generate potential matches for stored hash values.

1. **Variations in Hash Function** 
   1. **Salted hashes**

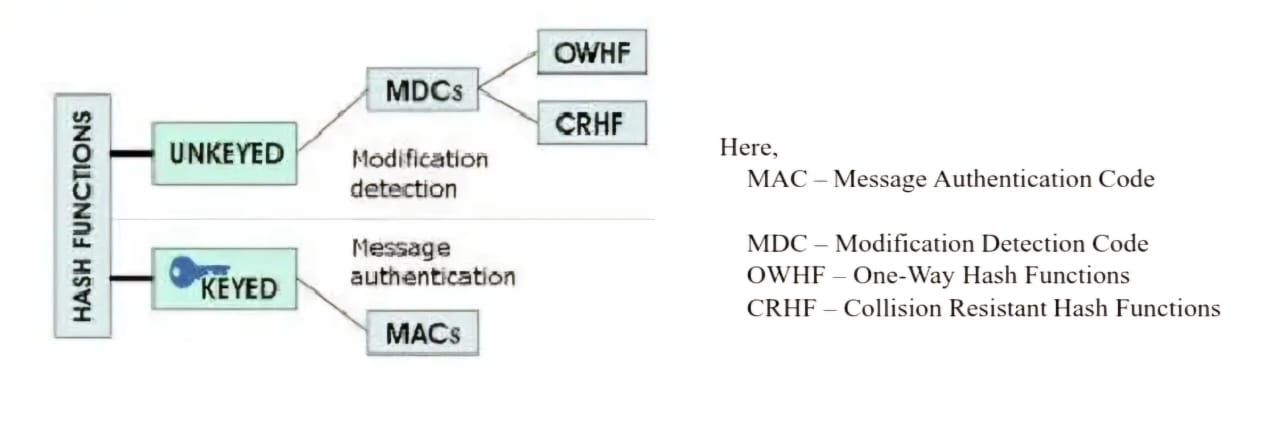
Salting adds random data to each plaintext credential. The result: two identical plaintext passwords are now differentiated in enciphered text form so that duplicates cannot be detected.

**7.2. Keyed hash functions**

A keyed hash function (HMAC) is an algorithm that uses a cryptographic key AND a cryptographic hash function to produce a message authentication code that is keyed and hashed.

**7.3. Adaptive hash functions**

An adaptive one-way function is any function that is designed to iterate on its inner workings, feeding the output back as input, in a manner that causes it to-ultimately-take longer to execute.

1. **Classification of Hash Functions**

**Modification Detection Code (MDC)**

A MDC is a message digest that can prove the integrity of the message that message that it has not been changed.

Example: If Alice needs to send a message to bob, he can create a message digest, MDC and send both the message and MDC to Bob.

Bob can create a new MDC from the message and compare the received message and new MDC, If they are same, the message is not changes.

**Message Authentication Code (MAC)**

MAC involves the use of a secret key to generate a small fixed size block of data that is appended to message

* ﻿﻿It assumes that the two communicating parties A and B share a common secret key K
* ﻿﻿When A send message to B it calculates the MAC of the message and the key,

MAC = C (K, M)

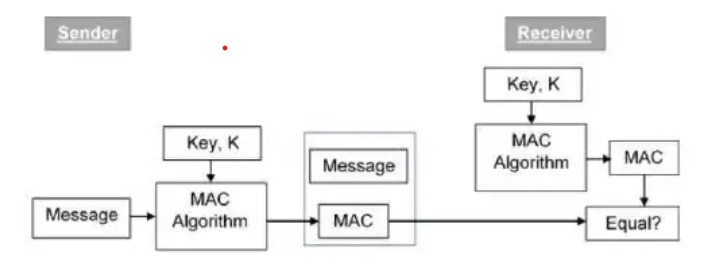
Where,

M= Input message, K= Secret Key, C= MAC function

• The Message + MAC are transmitted to receiver

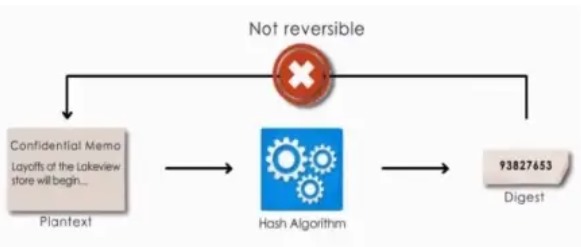
• The receiver performs calculation on received

• The received MAC is compared to calculated



1. **Working of Hash Functions**

Hash functions or hash algorithms creates a unique digital fingerprint of data i.e. message digest or hash



Hash Algorithm has 3 basic characteristics

• Secure - Non reversible function

• Fixed Size - Size is fixed to the digest

• Unique - Cannot be same



1. **Working of SHA -256**

SHA-256 is a hash function with digest length of 256 bits.

* 1. Boolean Operations:

• Boolean operations AND, XOR and OR, denoted by ^, ® and V, respectively

• Bitwise complement, denoted by ~

• Integer addition modulo 232, denoted by A + B

• RotRA, n) denotes the circular right shift of n bits of binary word A

• ShRA, n) denotes the right shift of n bits of the binary word A

• AkB denotes the concatenation of the binary words A and B

* 1. Functions and constants:

The algorithm uses the functions,

* ﻿﻿Ch(X, Y, Z) = (X ^ Y) Đ (X ^Z)
* ﻿﻿Maj(X, Y, Z) = (X M Y) Đ (X ^Z) Đ (Y ^Z)
* ﻿﻿E0(X) = RotR(X, 2) \* RotR(X, 13) \* RotR(X, 22)
* ﻿﻿o0(X) = RotR(X, 7) © RotR(X, 18) ® ShR(X, 3)
  1. 3)Padding

4)Block Decomposition

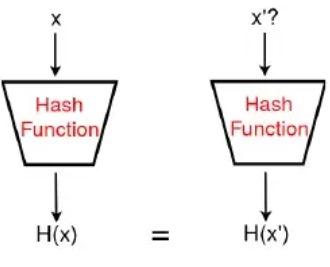
5)Hash Computation

6)Implementation

**Example for Secure Hash Algorithm (SHA-256)**

1. **Possible Attacks on Hash Functions** 
   1. **Weak collision resistance**

The problem is that given a digest and the message that produced it, find another message that will produce the same digest. We will try to compute the number of times we should try, to have a good probability to find the answer to the problem. Good probability will be given by a probability of 0.5.



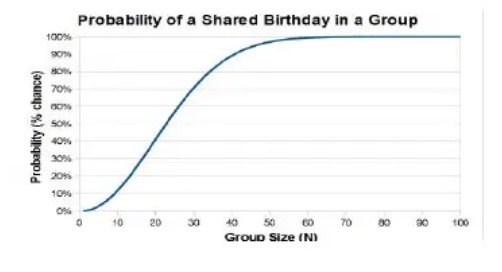
* 1. **Birthday paradox**

It's a simple principle that comes from a simple problem. How many people have to be in a room, to have a probability bigger than 0.5 that two persons have been born the same day? The calculation, after some mathematic work, gives us the formula:

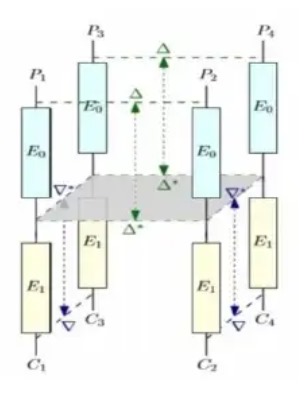
m = V2 (In2) n = 1.17Vn

where n is the number of results in the image (for the birthday paradox, n would be 365). And m would be the

number of persons we need. This gives us m = 22.3. So, if we have 23 people in a room there is a good chance

that we find two born the same day.

* 1. **Boomerang attack**
* The boomerang attack is a chosen plaintext and adaptive chosen ciphertext attack discovered by Wagner.
* The attack may use characteristics, differentials, as well as truncated differentials.
* The attack breaks constructions in which there are high-probability differential patters propagating halfway through the cipher both from the top and from the bottom, but there are no good patterns that propagate through the full cipher.



1. **Applications of Hash Function**

**• Digital signatures**

These are used for signing a document with the good properties that the signer can tell when something has not been signed by him AND that he can not deny his signature in a signed document.

**• Virus checking**

To know if a file has been infected with a virus means to know if it has been, we can use the digest of the unmodified and safe file to compare it to the digest of the possible infected file. If it turns that both digest are equal then we can say that the file has not been modified, then that it is virus-free.

**• Secure Socket Layer connections**

Secure Socket Layer (SSL) provides server authentication to clients. It's the facto standard for communication on the Internet. During the SSL handshake once the parts have negotiated the protocol, these have to select also the hash methods to use for authentication.

**• Password tables**

A common method of client authentication is to require the client to present a password previously registered with the server. Storing passwords of all users on the server poses an obvious security risk.

