Mathematics in Information Security

UNIT 3

Security requirements for Hash functions

Pre-image **Resistance :**

Definition: Given a hash output H(x)H(x)H(x), it should be computationally infeasible to find any input xxx that hashes to that output.
Security implication: This prevents attackers from reversing the hash to discover the original input data.

Second Pre-image Resistance :

Definition: It should be infeasible to find a second distinct input x2x_2x2 that produces the same hash value as a given input x1x_1x1.
Security implication: This prevents attackers from finding different inputs that result in the same hash (collision).

Security requirements for Hash functions

Collision Resistance **:**

Definition: It should be computationally infeasible to find any two distinct inputs x1x_1x1 and x2x_2x2 such that H(x1)=H(x2)H(x_1) = H(x_2)H(x1)=H(x2).
Importance: Collision resistance ensures that no two different inputs will hash to the same output. If collisions are possible, an attacker could substitute one message for another, undermining the integrity of the hash.

Avalanche Effect :

Definition: A small change in the input (even a single bit) should produce a significantly different hash value.
Importance: This ensures that the output appears random and unrelated to the input. It prevents attackers from making small changes to the input data that would lead to predictable changes in the hash output.

Security requirements for Hash functions

Fast Computation :

Definition: The hash function should be efficient to compute, even for large inputs.
Importance: Speed is important for practical applications, such as real-time data verification, digital signatures, and blockchain. However, efficiency should not come at the cost of security.

Uniform Distribution :

Definition: The hash function should produce hash values that are uniformly distributed across the entire output space.
Importance: This property ensures that small changes in input lead to large, unpredictable changes in the hash output. It prevents attackers from identifying patterns in the hash values that could make attacks easier.

Security requirements for Hash functions

Fast Computation :

Definition: The hash function should be efficient and fast to compute, even for large inputs.
Importance: Efficiency is important for real-world applications, especially where speed is necessary, such as in blockchain technology, digital signatures, or data verification. However, this should not compromise security.

Resistance to Known Cryptanalytic Attacks :

Definition: The hash function should be resistant to attacks such as birthday attacks, differential cryptanalysis, and pre-image attacks.
Importance: Without resistance to cryptanalytic attacks, hash functions can be broken, allowing attackers to find collisions, pre-images, or second pre-images more easily.

Security requirements for Hash functions

Keyed Hashing (for HMAC) :
Definition: When used in HMAC (Hashed Message Authentication Code) constructions, the hash function should incorporate a secret key along with the message, ensuring that both the message and the key influence the hash value.
Importance: Keyed hashing prevents certain attacks on the hash function, such as attacks that target the hash function's structural weaknesses, and ensures data integrity and authenticity.

No Length Extension Attacks
Definition: The hash function should not be susceptible to length extension attacks, where an attacker can modify the message by appending data without knowing the original message.
Importance: This property is important for functions like MD5 or SHA-1, which are vulnerable to such attacks. A secure hash function, like SHA-256 or SHA-3, avoids this vulnerability.

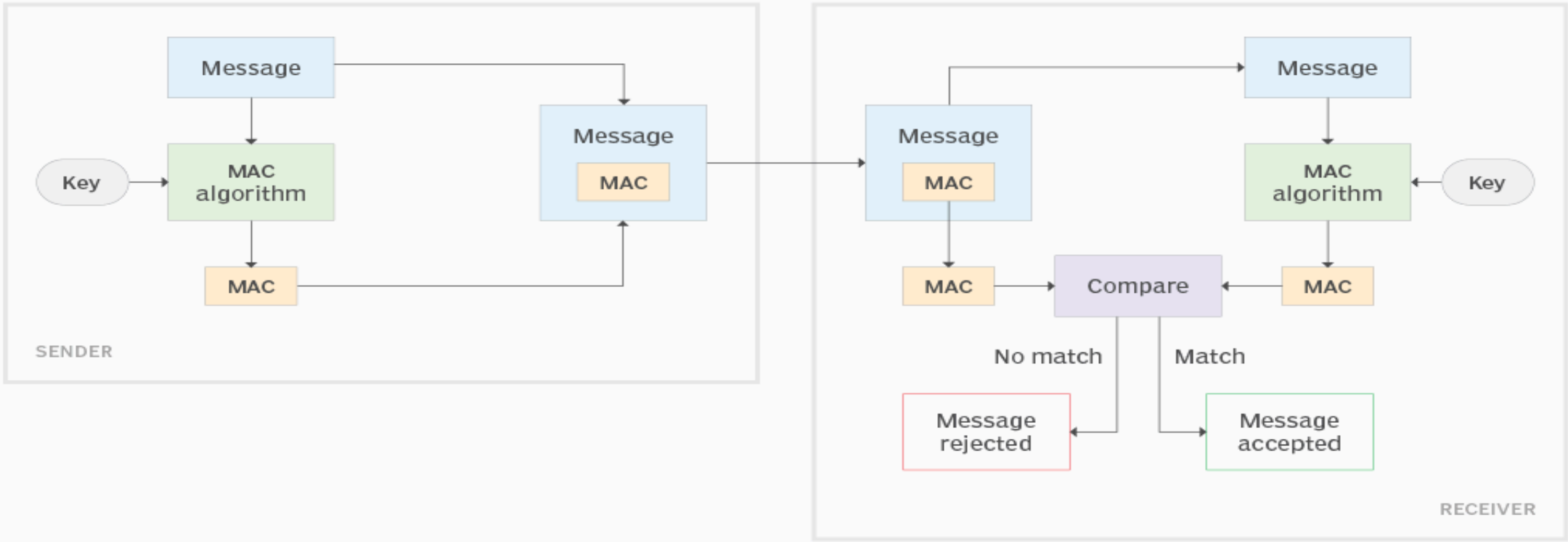
Message Authentication Code

A message authentication code (MAC) is a cryptographic checksum applied to a message in network communication to guarantee its integrity and authenticity.

Symmetric key cryptographic techniques are used to generate MACs for individual messages. The process requires a standard MAC algorithm that takes two inputs: the original message and a secret key known only to the message originator and its intended recipient. The following figure provides an overview of how a sender generates a MAC and how it is verified by the receiver.

Message Authentication Code

Generating and verifying message authentication codes



Message Authentication Code

MAC algorithms :

Keyed-Hash-based Message Authentication Code :
HMAC is a cryptographic authentication technique that uses a secret key in conjunction with a hash function approved by the Federal Information Processing Standards (FIPS).

Keccak Message Authentication Code :
KMAC is a cryptographic hash function that can also be used for authentication, encryption and pseudo-random number generation. There are two variants of Keccak, KMAC128 and KMAC256.

CMAC Mode for Authentication :
The cipher-based message authentication code (CMAC) standard defines a block cipher-based MAC algorithm for ensuring authenticity and integrity.

Data Integrity

Data integrity in cryptography ensures that a message has not been tampered with or changed. The most typical method is to use a hash function, which combines all the bytes in the message with a secret key to generate a message digest that is difficult to reverse. Integrity verification is a component of an information security program.

Different Kinds of data integrity :

Physical and logical data integrity are the two forms of data integrity. Both are a collection of procedures and methods for maintaining data integrity in hierarchical and relational databases.

Data Integrity

Physical integrity :

Physical integrity refers to the safeguarding of data's completeness and precision during storage and retrieval. Physical integrity is jeopardized when natural disasters occur, electricity goes out, or hackers interrupt database functions.

Data processing administrators, device programmers, applications programmers, and internal auditors may be unable to obtain reliable data due to human error, storage degradation, and a variety of other issues.

Data Integrity

Logical Integrity :

Logical integrity, like physical integrity, defends data from human error and hackers, but in a different way. There are four different forms of logical consistency.

Entity integrity :

It's a characteristic of relational databases, which store information in tables that can be connected and used in a number of ways.

Referential integrity :

The term "referential integrity" refers to a set of procedures that ensure that data is stored and used consistently. Only necessary modifications, additions, or deletions of data are made.

Data Integrity

Domain integrity :

A domain is a collection of suitable values that a column may contain in this context. Constraints and other steps that restrict the format, sort, and amount of data entered can be used.

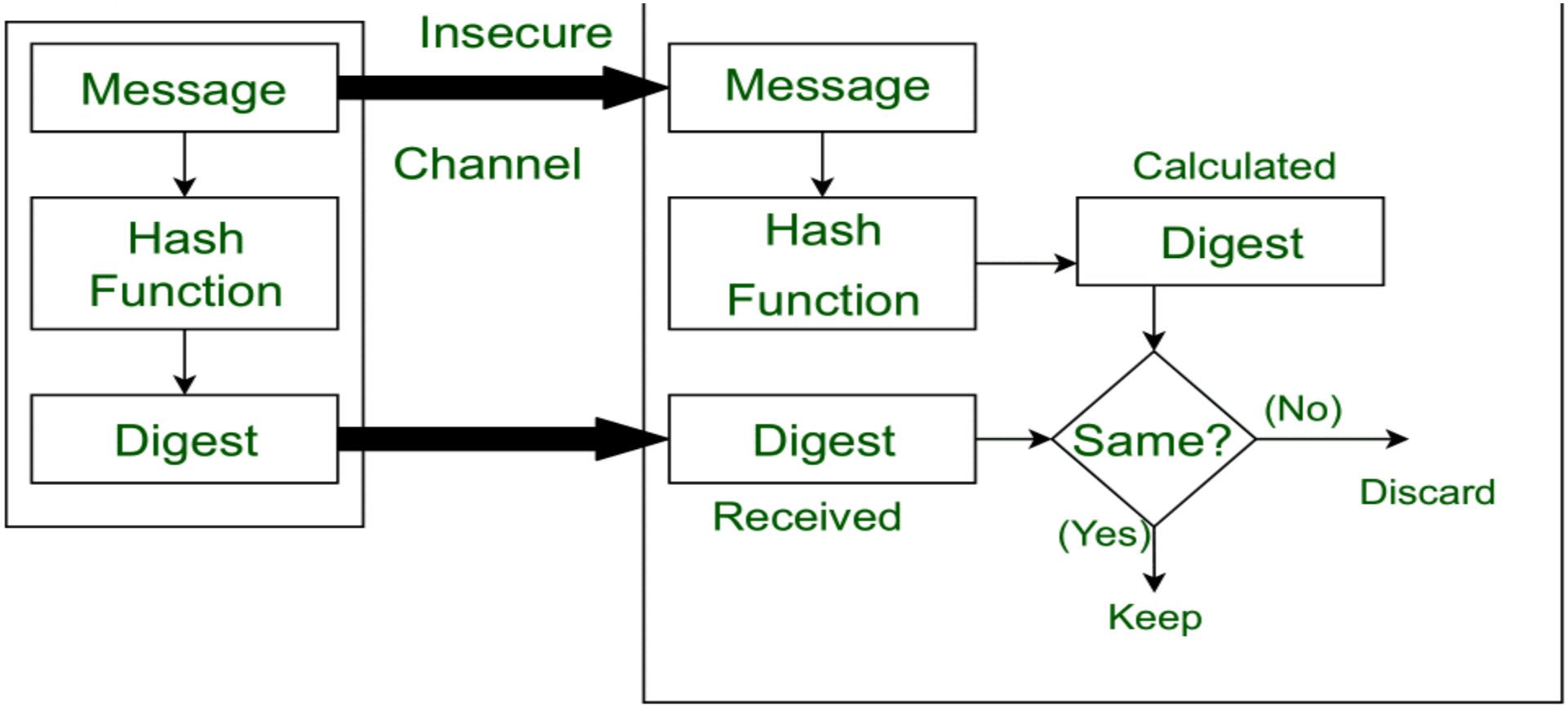
User-defined integrity :

User-defined integrity refers to the rules and restrictions that the user creates to meet their specific requirements. When it comes to data security, person, referential, and domain integrity aren't always enough.

Message Digest

Message Digest is used to ensure the integrity of a message transmitted over an insecure channel (where the content of the message can be changed).

The message is passed through a Cryptographic hash function. This function creates a compressed image of the message called Digest.



Message Digest

This message and digest pair is equivalent to a physical document and fingerprint of a person on that document. Unlike the physical document and the fingerprint, the message and the digest can be sent separately.

Most importantly, the digest should be unchanged during the transmission.
The cryptographic hash function is a one way function, that is, a function which is practically infeasible to invert.

This cryptographic hash function takes a message of variable length as input and creates a digest / hash / fingerprint of fixed length, which is used to verify the integrity of the message.

Message Digest

Message digest ensures the integrity of the document. To provide authenticity of the message, digest is encrypted with sender’s private key.

Now this digest is called digital signature, which can be only decrypted by the receiver who has sender’s public key.

Now the receiver can authenticate the sender and also verify the integrity of the sent message.

Message Digest

Use Cases of Message Digests:

Data Integrity: Verifying that data has not been altered. If the digest of the received data matches the expected digest, the data is considered unchanged.

Digital Signatures: Ensuring that a document or message comes from a verified source. The document's message digest is encrypted with the sender's private key.

Password Storage: Instead of storing plaintext passwords, systems store the message digest of the password. When users log in, their input is hashed and compared to the stored digest.

Checksums: Used to validate the integrity of files during transfer, ensuring they haven't been corrupted or tampered with.

Message Digest 5

MD5 is used as a checksum to verify the integrity of files and data by comparing the hash of the original file with the file received to check if the files or data has been altered.

MD5 is used for data security and encryption e.g. Secure password of users in database and non-sensitive data.

It is used in version control systems to manage different versions of files.

It was earlier used in digital signatures and certificate but due it’s vulnerabilities, it has been replaced by more secure algorithms like SHA-256.

Message Digest 5

Advantages of MD5 Algorithm

- MD5 is faster and simple to understand.
- MD5 algorithm generates a strong password in 16 bytes format. All developers like web developers, etc. use the MD5 algorithm to secure the password of users.
- To integrate the MD5 algorithm, relatively low memory is necessary.
- It is very easy and faster to generate a digest message of the original message.

Message Digest 5

Disadvantages of MD5 Algorithm :

- MD5 generates the same hash function for different inputs (hash collision).
- MD5 provides poor security over SHA1, SHA256 and other modern cryptographic algorithms.
- MD5 has been considered an insecure algorithm. So now we are using SHA256 instead of MD5.
- MD5 is neither a symmetric nor asymmetric algorithm.

Secure Hashing Algorithm

SHA stands for Secure Hash Algorithm, which is a family of cryptographic hash functions designed by the National Security Agency (NSA). These hash functions are widely used in various security applications, such as digital signatures, message authentication codes, and data integrity verification.

Secure Hashing Algorithm

There are several versions of SHA, each offering a different level of security and performance:

SHA-1:

This is an older version that produces a 160-bit hash value. It was widely used in the past but has since been considered insecure due to vulnerabilities discovered over time. It is now deprecated in favor of more secure algorithms.

SHA-2:

This is a more secure version of the algorithm and includes several variants that generate hash values of different sizes:

SHA-224: 224-bit hash
SHA-256: 256-bit hash
SHA-384: 384-bit hash
SHA-512: 512-bit hash SHA-2 is widely used and considered secure for most applications.

Secure Hashing Algorithm

SHA-3:

This is the latest version in the SHA family, developed as a part of the Keccak cryptographic hash family. SHA-3 offers different hash lengths (224, 256, 384, and 512 bits), and it is designed to be more resistant to potential vulnerabilities that SHA-2 might face in the future. SHA-3 is not a replacement for SHA-2 but rather an alternative with a different internal structure.

Secure Hashing Algorithm

Common Uses of SHA:

Data Integrity: Ensuring that the data hasn't been tampered with during transmission or storage.

Digital Signatures: Verifying the authenticity and integrity of digital messages or documents.

Password Hashing: Storing passwords securely in databases by hashing them before storing them.

Blockchain: In blockchain technology (such as Bitcoin), SHA-256 is used to secure transactions and ensure the integrity of the chain.

Completion of Unit-3

Your Questions

THANKS