

Information Security

Prepared by Dr. Ahmed Omar

IT373

Information Security

Lecture 4



Message Authentication & Digital Signatures

Lecture 4

IT373

Information Security

Prepared by
Dr. Ahmed Omar

Agenda

- ☐ Introduction to Cryptographic Hash Functions
- ☐ Common Hash Algorithms and Their Role
- ☐ Introduction to Message Authentication
- ☐ How MACs Work
- ☐ Real-World Applications of MACs
- ☐ Digital Signatures
- ☐ Real-World Applications of Digital Signatures

Introduction to Cryptographic Hash Functions

- ❑ A **hash function** takes an **input value** of any arbitrary size and converts it into a **fixed-size output**, known as a *hash value*, hash, or **message digest**. Regardless of whether the input is a single word or an entire book, the output will always be of a predetermined, consistent length.

Key Properties of Cryptographic Hashes

- ❑ Understanding the essential traits that make a hash function cryptographically secure.
- ❑ A secure cryptographic hash function must meet three key properties:
 - **Collision Resistance**
 - **Pre-image Resistance**
 - **Second Pre-image Resistance**

Key Properties of Cryptographic Hashes

❑ Collision Resistance

- It should be extremely difficult to find two different inputs that produce the same hash output.
- Also called a "collision-free" hash function.
- **Importance:** Prevents attackers from substituting messages undetected (e.g., tampering with digital documents or signatures).

Key Properties of Cryptographic Hashes

❑ Pre-image Resistance (One-way Property)

- Given a hash value, it should be computationally infeasible to find the original input.
- **Importance:** Protects against reverse-engineering a hash to reveal sensitive data (e.g., passwords).

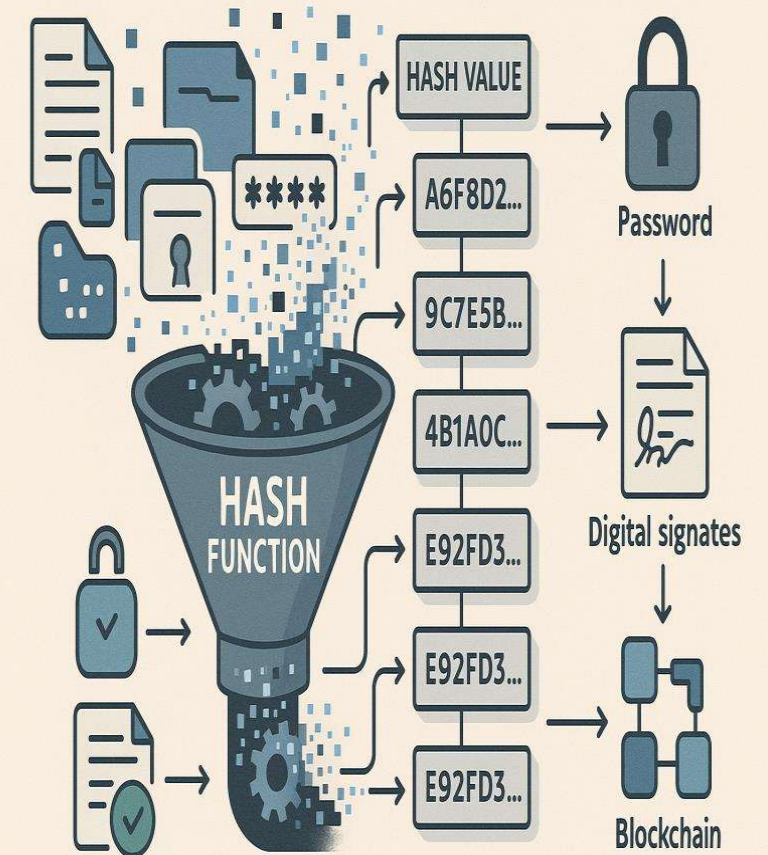
Key Properties of Cryptographic Hashes

❑ Second Pre-image Resistance

- Given an input and its hash, it should be very difficult to find a different input with the same hash.
- **Importance:** Prevents attackers from modifying data while keeping the original hash, enabling stealthy forgeries.

Hash Real world Applications

- Password storage
- Digital signatures
- File verification
- Blockchain



How Hashing is Used in Password Storage

- ❑ User creates a password → "MyPass123"
- ❑ System hashes it → $\text{SHA1}(\text{"MyPass123"}) = 1\text{BAD0E0F36CA8D507ED2A9B}..$
- ❑ Only the hash is stored in the database
- ❑ On login:
 - ✓ User enters password
 - ✓ Hash is computed and compared
 - ✓  Match = allow login
 - ✓  Mismatch → Access Denied

How Hashing is Used in Password Storage

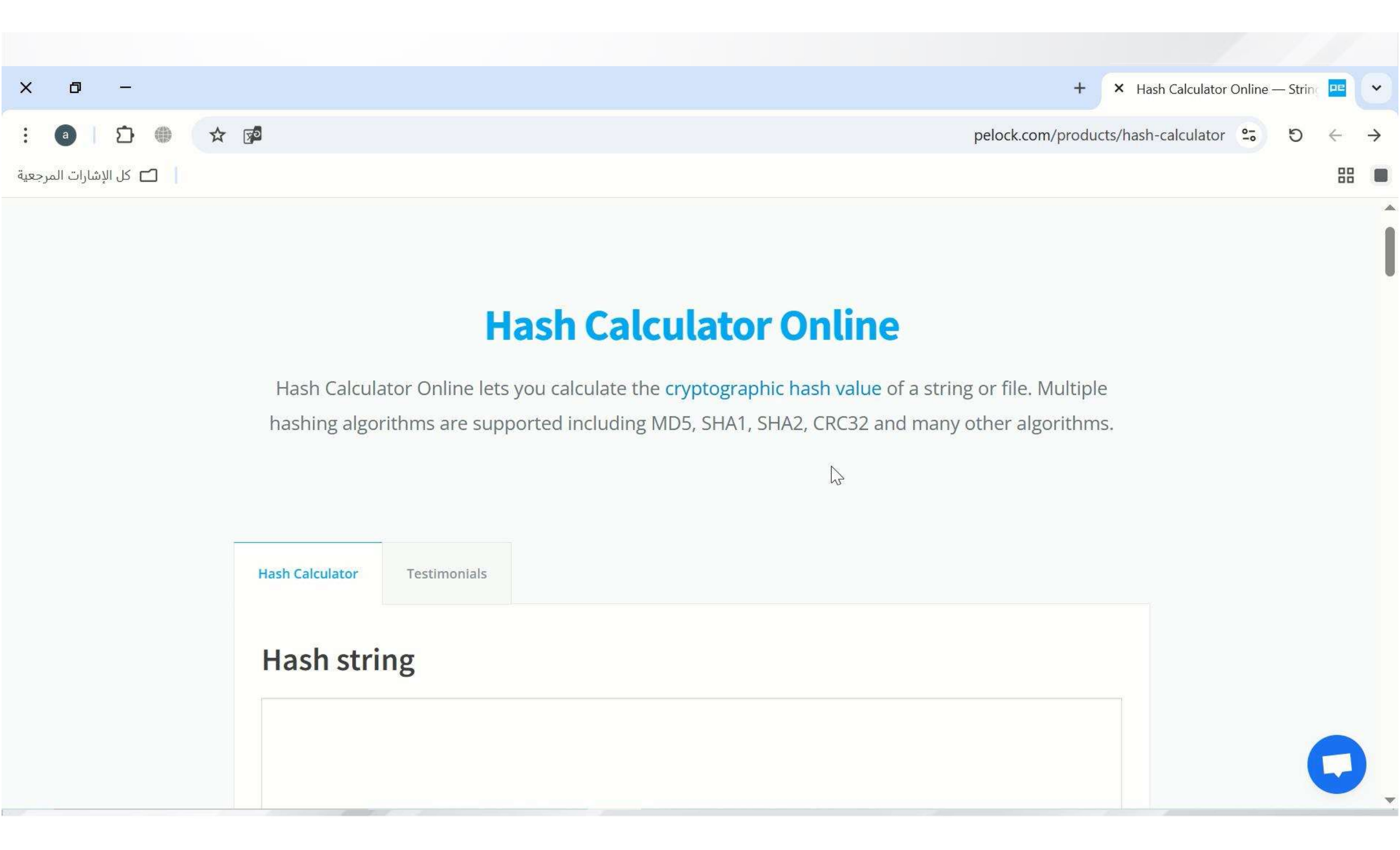
- ❑ **Adding Security with Salting** (adding randomness)
 - Password = 123456
 - Salt = xyz@!
 - Hashed = Hash("123456xyz@!")
- ✓ Prevents attackers from using rainbow tables
- ✓ Ensures even identical passwords have different hashes

Common Hash Algorithms and Their Role

- ❑ **MD (Message Digest) Family**
 - **Breaks message into 512-bit blocks**
 - **Applies a compression function repeatedly**
 - **Produces a 128-bit output (for MD5)**

Common Hash Algorithms and Their Role

- ❑ **Secure Hash Algorithm (SHA) Families**
 - **SHA-1: 160-bit output**
 - **SHA-2: Variable outputs (224, 256, 384, 512)**
 - **SHA-3 :Includes: SHA3-224, SHA3-256, SHA3-384, SHA3-512**



Hash Calculator Online

Hash Calculator Online lets you calculate the [cryptographic hash value](#) of a string or file. Multiple hashing algorithms are supported including MD5, SHA1, SHA2, CRC32 and many other algorithms.



Hash Calculator

Testimonials

Hash string



Introduction to Message Authentication

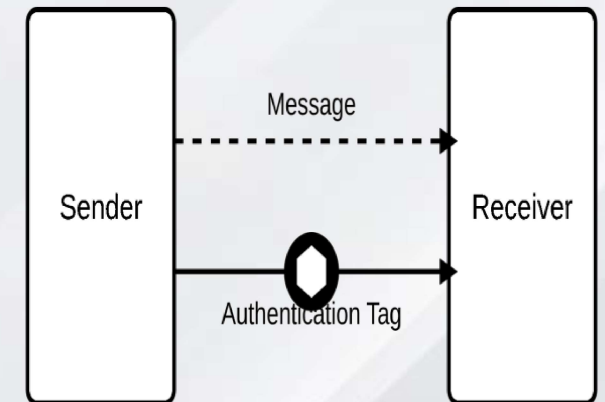
- ❑ A process that provides assurance of the **integrity** of messages, documents, or stored data, and **verification** of the sender's identity.
- ❑ Ensuring a message came from the stated sender and was not altered.
- ❑ **Key Properties:** Verifier (receiver) uses a shared secret to validate the message's authenticity and integrity

Message Authentication Code (MAC)

- ❑ A Message Authentication Code (**MAC**), sometimes referred to as an **authentication tag**, is a concise piece of information used to verify both the authenticity and integrity of a message.
- ❑ Its primary objectives are two fold:
 - first, to enable the detection of any **modifications** to the message content during transmission, thereby ensuring data integrity; and
 - second, to confirm the **identity** of the **sender**, providing authenticity

Introduction to Message Authentication

- ❑ Unlike **encryption**, which focuses on **confidentiality**, MACs do not encrypt the message itself; rather, they provide a cryptographic checksum that depends on both the message and a secret key. This secret key is shared exclusively between the sender and the intended receiver.

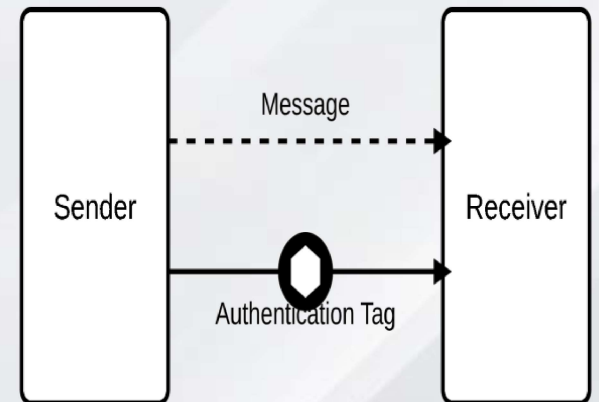


Introduction to Message Authentication

□ Key Concepts

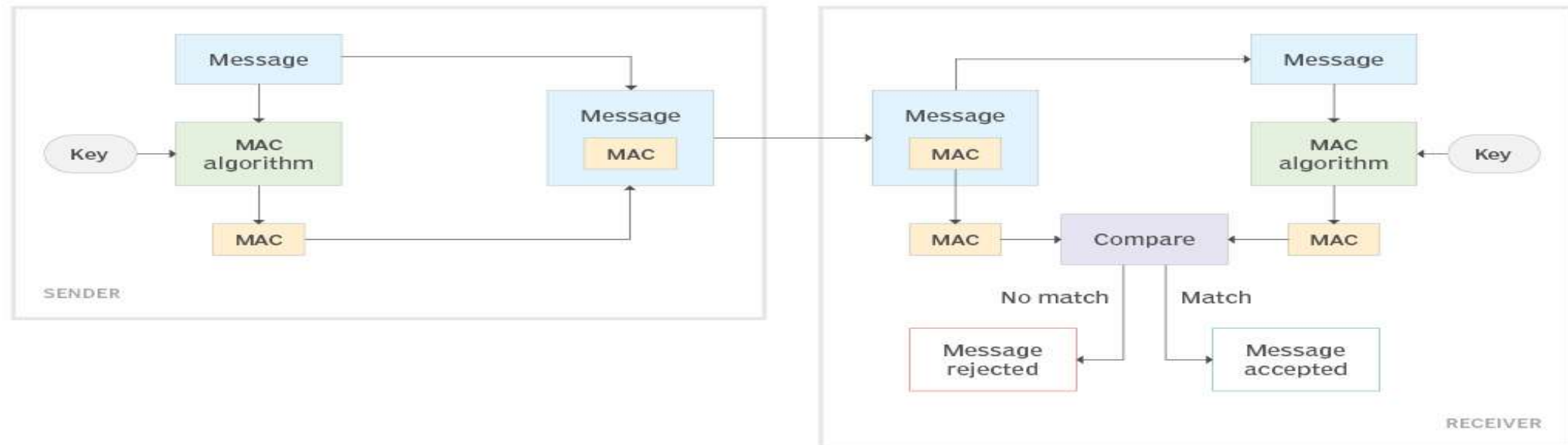
 **Integrity** : Ensuring data has not been altered during transmission

 **Authenticity** : Verifying the identity of the message sender



Message Authentication Codes (MACs)

Generating and verifying message authentication codes



How MACs Work: The Symmetric Key Approach

- ❑ The operation of a MAC relies on a **symmetric** key approach, meaning the same secret key is used for both generating and verifying the MAC tag. The process typically unfolds in **five steps**:
 1. **Key Generation**: Initially, a shared secret key, often denoted as **k**, is securely established and known only to the **sender** (Party A) and the **receiver** (Party B). The security of the MAC entirely hinges on the secrecy of this key.

How MACs Work: The Symmetric Key Approach

2. MAC Generation (Sender A): When **Party A** wishes to send a message ***m*** with integrity and authenticity assurances, **Party A** inputs the message ***m*** and the shared secret key ***k*** into a **MAC algorithm**. This computation produces a fixed-size output called the **MAC tag**, **$t = \text{MAC}(m; k)$** .

How MACs Work: The Symmetric Key Approach

3. Transmission: Party A then transmits the original message m along with the generated MAC tag t to Party B. It is important to emphasize that the message m itself is typically sent in plaintext; the MAC process does not provide confidentiality.

How MACs Work: The Symmetric Key Approach

4. MAC Verification (Receiver B): Upon receiving the message **m** and its accompanying tag **t**, **Party B** independently performs the same MAC computation. Using the received message **m** and their copy of the shared secret key **k**, Party B computes a new **MAC tag**, **$t' = \text{MAC}(m; k)$** .

How MACs Work: The Symmetric Key Approach

5. Comparison: Finally, **Party B** compares the received **MAC tag t** with the newly computed tag **t'** . If t and t' are identical, Party B can confidently conclude that the message has not been altered during transit (**integrity**) and that it originated from Party A (**authenticity**), as only Party A (and Party B) possesses the secret key necessary to produce that specific tag. If the tags do not match, it indicates that the message has been tampered with or that the tag was forged by an unauthorized entity.

Types of MACs

While various MAC algorithms exist, two common types are widely recognized:

- ☐ Hash-based Message Authentication Code (**HMAC**)
- ☐ Cipher-based Message Authentication Code (**CMAC**)

HMAC (Hash-based Message Authentication Code)

A MAC (Message Authentication Code) that combines:



A secret key



A cryptographic hash function



Used to verify message authenticity and integrity

HMAC (Hash-based Message Authentication Code)

How HMAC Works – Two-Pass Construction

- ❑ Combines a cryptographic hash function with a secret key.
- ❑ Produces an authentication code (digest).
- ❑ Designed to address vulnerabilities in simpler key-hashing methods.

HMAC (Hash-based Message Authentication Code)

HMAC: The Two-Pass Computation

1.Key Derivation: The secret key is used to derive an **inner key** and an **outer key**.

2.First Pass (Inner Hash):

1. Generates an internal hash.
2. Inputs: Message + **Inner Key**.

3.Second Pass (Final HMAC):

1. Produces the final HMAC value.
2. Inputs: Result of the inner hash + **Outer Key**.

Real-World Applications of MACs

- ☐ Secure Communication Protocols
- ☐ API Authentication
- ☐ Data Integrity Verification
- ☐ Password Reset and Account Activation Links
- ☐ Internet of Things (IoT) Devices
- ☐ JSON Web Tokens (JWT)

How MACs are Used in Password Reset Links

When you request a password reset, the process often involves these steps, with a MAC securing the critical link:

1. **User Request:** You (the user) go to a website and click "**Forgot Password**," entering your email address.
2. **Server Generates Data:** The server generates a unique, **single-use token** or a set of parameters that identify the user and the request. This data might include:
 - User ID
 - Timestamp of generation
 - A unique, random nonce (number used once)
 - An expiry time for the link

How MACs are Used in Password Reset Links

3. MAC Generation: The server takes all this critical data (user ID, timestamp, nonce, expiry) and combines it with a **secret key** (known only to the server) to generate a Message Authentication Code (**MAC**). A common algorithm for this is **HMAC** (Hash-based Message Authentication Code).

$$\text{MAC} = \text{HMAC}(\text{SecretKey}, \text{UserID} || \text{Timestamp} || \text{Nonce} || \text{ExpiryTime})$$

How MACs are Used in Password Reset Links

4. Link Construction: The server then constructs the password reset link, embedding the user ID, timestamp, nonce, expiry time, and the generated MAC within the URL parameters.

Example:

https://example.com/reset?user_id=123&ts=1719772800&nonce=xyz789&exp=1719776400&mac=ABCDEF123456

5. Email Delivery: This link is sent to the user's registered email address.

6. User Clicks Link: When the user receives and clicks the link, their browser sends the full URL to the server.

How MACs are Used in Password Reset Links

- 7. Server Verification:** The server receives the link and extracts all the parameters (user ID, timestamp, nonce, expiry time, and the received MAC).
- The server then re-generates the MAC using its own secret key and the received parameters (UserID, Timestamp, Nonce, ExpiryTime).
 - **Re-generated_MAC = HMAC(SecretKey, UserID || Timestamp || Nonce || ExpiryTime)**

How MACs are Used in Password Reset Links

8. Comparison and Validation:

- ❑ The server compares the **Re-generated_MAC** with the received **MAC**.
- ❑ If they **match** and the link has not expired: The server trusts the link's integrity and authenticity. It knows the link wasn't tampered with and was legitimately generated by itself. The user is then **allowed** to set a new password.
- ❑ If they **do NOT match** or the link has expired: The server rejects the request, as it indicates either tampering, an expired link, or a malicious attempt to forge a reset link.

Digital Signatures



- ❑ A digital signature is a sophisticated mathematical algorithm routinely employed to **validate** the **authenticity** and integrity of **digital messages** or **documents**.
- ❑ It functions as a unique virtual **fingerprint**, inextricably linked to a specific person or entity, designed to identify users and protect information in digital communications.
- ❑ Digital signatures are often considered stronger and more difficult to forge than traditional handwritten signatures.
- ❑ Digital signatures do not provide confidentiality; the message itself is typically transmitted in plaintext alongside the signature.

Digital Signatures



- ❑ Their key properties are:
- ❑ **Authenticity:** A digital signature provides undeniable proof that the message or document originated from the claimed sender.
- ❑ **Integrity:** It confirms that the original data has remained intact and has not been altered, intentionally or unintentionally, since it was signed. Any modification to the signed document would invalidate the signature.
- ❑ **Non-Repudiation:** This is a critical differentiator from MACs. A digital signature ensures that the **sender** cannot later deny having sent the message, and the **recipient** cannot deny having received it.

How Digital Signatures Work: The Asymmetric Key Approach

The general process for creating and verifying a digital signature is as follows:

- 1. Key Pair Generation:** The sender first generates a unique public/private key pair using a specialized cryptographic algorithm.
- 2. Message Hashing:** The original message or document is processed through a cryptographic hash function to produce a **fixed-size message** digest. This ensures that even a **minor** change in the message will result in a completely different hash.
- 3. Signature Creation (Sender):** The sender then "signs" this message digest by applying a mathematical operation using their private key. This transformed hash value is the digital signature. It is important to clarify that while the mathematical operation often resembles encryption

How Digital Signatures Work: The Asymmetric Key Approach

4. **Transmission:** The sender transmits the original (unencrypted) message along with the newly generated digital signature to the recipient.
5. **Signature Verification (Receiver):** Upon receiving the message and signature, the recipient first obtains the sender's public key. They then use this public key to perform a mathematical operation on the received digital signature to retrieve the original hash value.
6. **Independent Hashing:** Simultaneously, the recipient independently computes a hash of the received message using the exact same cryptographic hash function that the sender used.
7. **Comparison:** Finally, the recipient compares the hash value obtained from decrypting the signature with the hash value they independently computed from the received message.

Real-World Applications of Digital Signatures

- ❑ **SSL/TLS Certificates (HTTPS):** They are fundamental to establishing secure connections over the internet, authenticating web servers and clients, and enabling secure web browsing (HTTPS).
- ❑ **Software Updates and Distribution:** Digital signatures are used to verify the authenticity and integrity of software downloads, patches, and applications, ensuring that they originate from legitimate developers and have not been tampered with by malicious actors
- ❑ **Legal Documents and Contracts:** They provide legally binding proof of acknowledgment and consent for electronic documents, making them equivalent to handwritten signatures in many jurisdictions and ensuring non-repudiation for agreements, such as rental contracts, insurance policies, employment offers, and minutes of board meetings.

Real-World Applications of Digital Signatures

- ❑ **Financial Transactions:** Digital signatures revolutionize digital banking and loan processing by facilitating secure and efficient transactions, ensuring the authenticity and integrity of financial data.
- ❑ **Government Services:** Many governments worldwide adopt digital signatures for e-government services, including the submission of tax declarations, participation in public contracts, and access to administrative records, and enrolling in competitive examinations.
- ❑ **Cryptocurrencies and Blockchain:** Digital signatures are a core component of blockchain technology and cryptocurrencies like Bitcoin and Ethereum, where they are used to sign transactions and verify their integrity, ensuring the immutability of the distributed ledger.

pkitoools.net

- Home
- Cryptographic Primitive
- PKI Services
- Timestamp Services
- Key Tools
- Document Signing
- Document Validator
- Document Encryption
- Miscellaneous

Key parameter: RSASSA-PKCS1-v1_5 - Verifying: Public key validation required Signing: Private key validation required

Algorithm: RSA

Scheme: RSASSA-PKCS1-v1_5

Signature algorithm: SHA256withRSA

Key pair: ☒ Generate new keypair ☐ From file

☐ Import from key store

Key size: 1024

Generate

Signature

String ☒ Hex ☐ Base64 ☐ UInt8Array ☐ Import file

Input bit length: 0

☒ Sign ☐ Verify

Signature bit length: 0

String ☒ Hex ☐ Base64 ☐ UInt8Array ☐ Export file

Summary

- Cryptographic Hash Functions
- Message Authentication Codes (MACs)
- Digital Signatures



Thank You

