Unit 3

Message Authentication and Hash Functions:

Message authentication is a procedure to verify that received messages come from the alleged source and have not been altered. Message authentication may also verify sequencing and timeliness.

AUTHENTICATION REQUIREMENTS:

Message authentication can be achieved using cryptographic methods which further make use of keys.:

1. Disclosure: It means releasing the content of the message to someone who does not have an appropriate cryptographic key.
2. Traffic analysis: Determination of the pattern of traffic through the **duration of connection** and **frequency of connections** between different parties.
3. Masquerade: Adding out of context messages from a fraudulent source into a communication network. This will lead to mistrust between the parties communicating and may also cause loss of critical data.
4. Content modification: Changes to the contents of a message, including insertion, deletion, transposition, and modification.
5. Sequence modification: Any modification to a sequence of messages between parties, including insertion, deletion, and reordering.
6. Timing modification: This includes **replay** and **delay** of messages sent between different parties. This way **session tracking is also disrupted.**
7. Source repudiation: Denial of transmission of message by source. When the source denies being the originator of a message.
8. Destination repudiation: Denial of receipt of message by destination. When the receiver of the message denies the reception.

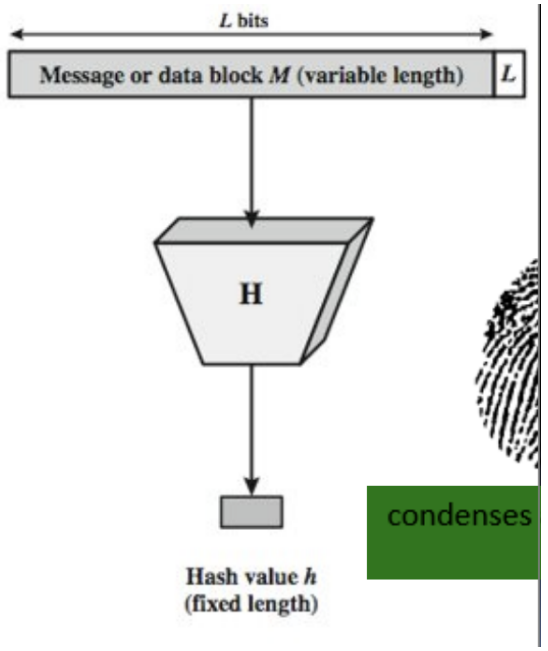
AUTHENTICATION FUNCTIONS:

Any message authentication or digital signature mechanism has **two levels of functionality**.

- **Lower level:** At this level, there is a need for a function that produces an authenticator, which is the value that is used to authenticate a message.
- **Higher-level:**The lower level function is used here in order to help receivers verify the authenticity of messages.

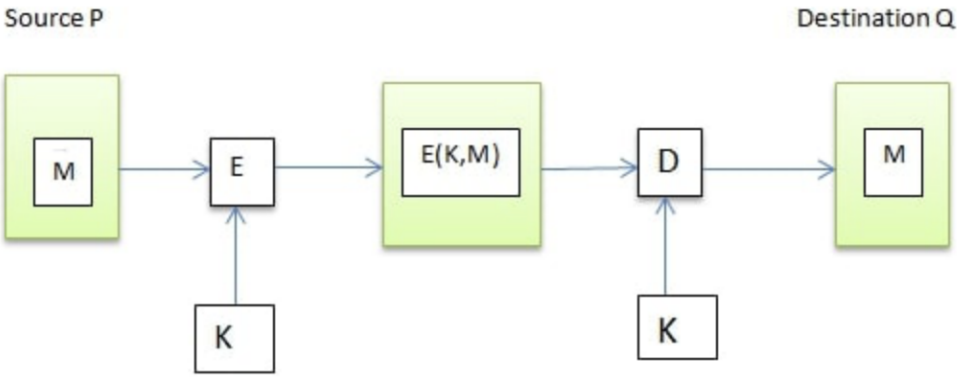
functions that may be used to produce an authenticator :

- **Hash function:** A hash function is nothing but a **mathematical function** that can convert a numeric value into another numeric value that is **compressed**.
  The **input** to this hash function can be of **any length** but the **output** is always of **fixed length.**
  The values that a hash function returns are called the **message digest or hash values.**



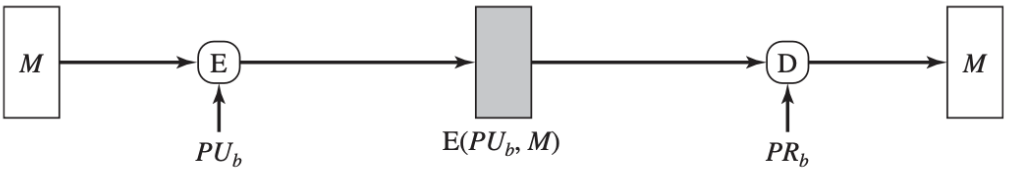
- **Message encryption:** The **cipher text** of the entire message serves as its **authenticator**
  We use Message Encryption to mitigate the risk of Man In The Middle Attack(MITM).
  In message encryption, the data is first converted to a ciphertext and then sent any further.
  Message encryption can be done in two ways:
  1. **Symmetric Encryption:**
     - provides **authentication** as well as **confidentiality**.
     - Say we have to send the message M from a source P to destination Q. This message M can be encrypted using a **secret key K that both P and Q share.**
     - Without this key K, no other person can get the plain text from the ciphertext. This maintains **confidentiality**.
     - Further, Q can be sure that P has sent the message. This is because other than Q, P is the only party who possesses the key K and thus the ciphertext can be decrypted only by Q and no one else. This maintains **authenticity**.

▪

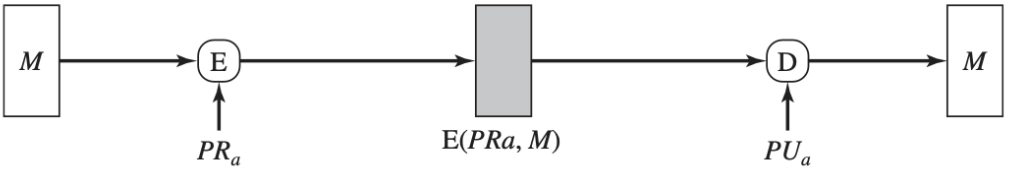


2. **Public Key Encryption:**
    - not as advanced as symmetric encryption as it provides **confidentiality** but **not authentication. To provide both** authentication and confidentiality, the **private key** is used.
    - The source (A) uses the public key *PUb* of the destination (B) to encrypt *M*. Because only B has the corresponding private key *PRb*, only B can decrypt the message.
    - This scheme provides no **authentication**, because any opponent could also use B's public key to encrypt a message and claim to be A.
    - To provide authentication, A uses its private key to encrypt the message, and B uses A's public key to decrypt . The message must have come from A because A is the only party that possesses *PRa* and therefore the only party with the information necessary to construct ciphertext that can be decrypted with *PUa*.

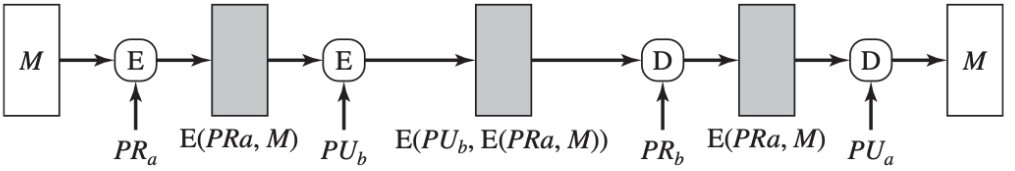
      Source A Dest. B

- 

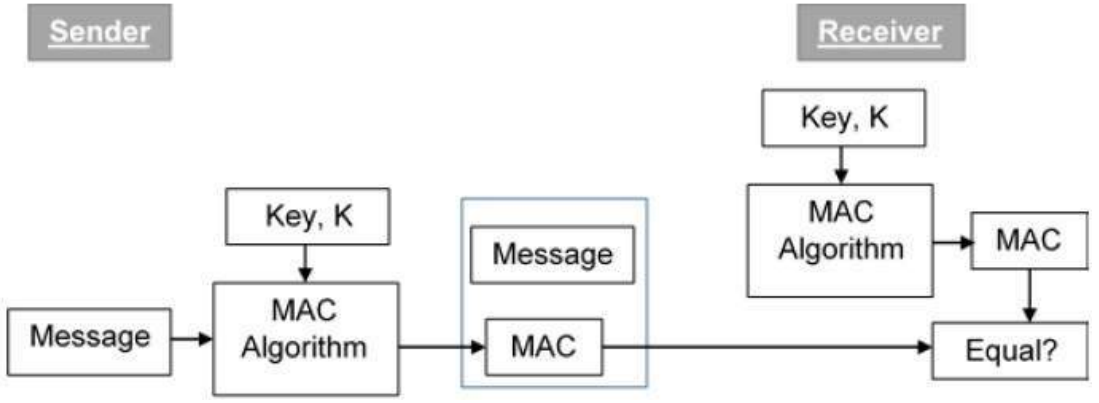
(b) Public-key encryption: confidentiality



(c) Public-key encryption: authentication and signature



(d) Public-key encryption: confidentiality, authentication, and signature

- **Message authentication code (MAC):**
  - MAC algorithm is a **symmetric key cryptographic technique** to provide message authentication. For establishing MAC process, the sender and receiver share a symmetric key K.
  - Essentially, a MAC is an encrypted checksum generated on the underlying message that is sent along with a message to ensure message authentication.
  - 

MESSAGE AUTHENTICATION CODES (MAC):

- An authentication technique involves the **use of a secret key to generate** a small fixed-size block of data, known as a **cryptographic checksum or MAC**, that is appended to the message.

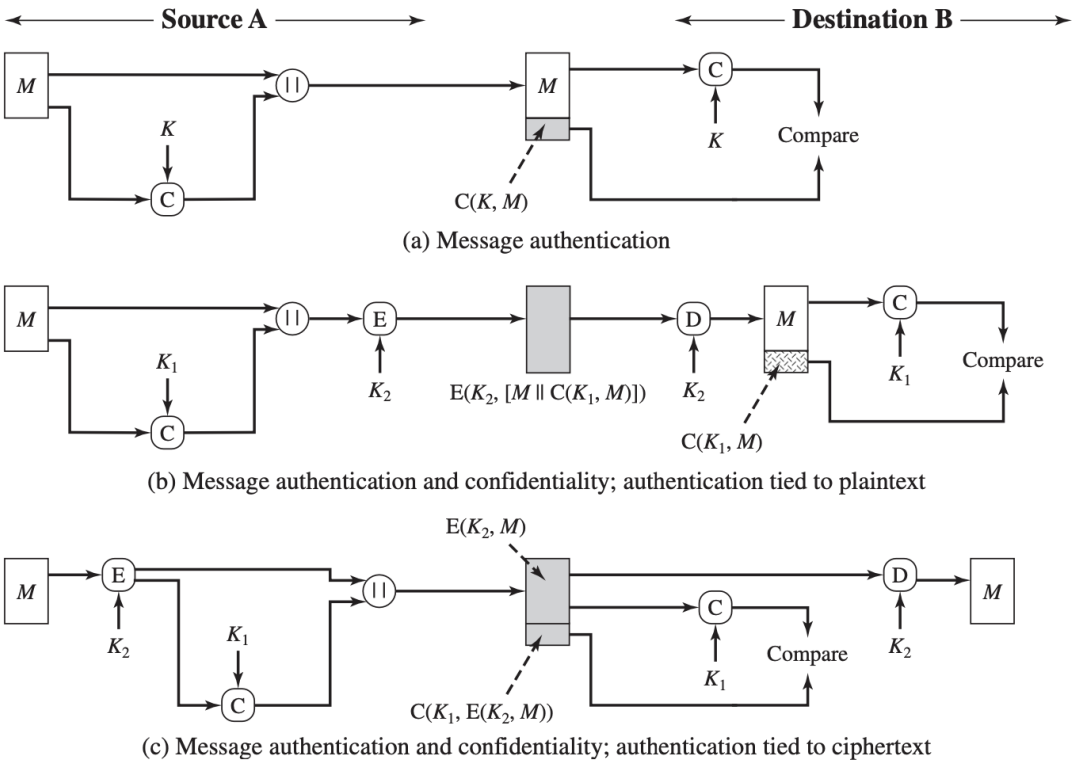
- This technique assumes that two communicating parties, say A and B, share a common secret key *K*. When A has a message to send to B, it calculates the MAC as a function of the message and the key:
  MAC = MAC(K, M) where
  M = input message
  C = MAC function
  K = shared secret key
  MAC = message authentication code
- The message plus MAC are transmitted to the intended recipient.
- The recipient performs the same calculation on the received message, using the same secret key, to generate a new MAC. The received MAC is compared to the calculated MAC.
- If the received MAC matches the calculated MAC, then:
  - The receiver is assured that the message has not been altered. - confidentiality
  - The receiver is assured that the message is from the alleged sender. - authenticity
- 

Figure 12.4 Basic Uses of Message Authentication code (MAC)

- A MAC function is similar to encryption. **One difference is that the MAC algorithm need not be reversible, as it must be for decryption**

SECURITY OF HASH FUNCTIONS AND MAC

We can group attacks on hash functions and MACs into two categories: brute-force attacks and cryptanalysis.

1. **Brute-force attacks:**

- Hash functions:
  - The strength of a hash function against brute-force attacks depends solely on the **length of the hash code** produced by the algorithm.
  - there are three desirable properties:
    - **One-way**: For any given code h, it is computationally infeasible to find x such that H(x) = h.
    - **Weak collision resistance**: For any given block x, it is computationally infeasible to find y x with H(y) = H(x).
    - **Strong collision resistance**: It is computationally infeasible to find any pair (x, y) such that H(x) = H(y).

    For a hash code of length n, the level of effort required, is proportional to the following:

One way	$_2$n
Weak collision resistance	$_2$n
Strong collision resistance	$_2$n/2

- Message Auth Code (MAC):
  - A brute-force attack on a MAC is a more difficult undertaking because it requires known message-MAC pairs.
  - the desired security property of a MAC algorithm:

    Computation resistance:
    - Given one or more text-MAC pairs (x_i, $C_K[x_i]$), it is computationally infeasible to compute any text-MAC pair (x, C_K(x)) for any new input x ≠x_i.
    - In other words, the attacker would like to come up with the valid MAC code for a given message x. There are two lines of attack possible: Attack the key space and attack the MAC value.
  - the level of effort for brute-force attack on a MAC algorithm can be expressed as min(2^k, 2^n).

2. **Cryptanalysis:**

- cryptanalytic attacks on hash functions and MAC algorithms seek to exploit some property of the algorithm to perform some attack other than an exhaustive search.
- The way to measure the resistance of a MAC algorithm to cryptanalysis is to compare its strength to the effort required for a brute-force attack.
- That is, an ideal MAC algorithm will require a cryptanalytic effort greater than or equal to the brute-force effort.

SECURE HASH ALGORITHMS (SHA):

SHA-512

- The algorithm takes as input a message with a maximum length of less than 2128 bits and produces as output a 512-bit message digest. The input is processed in 1024-bit blocks.
-

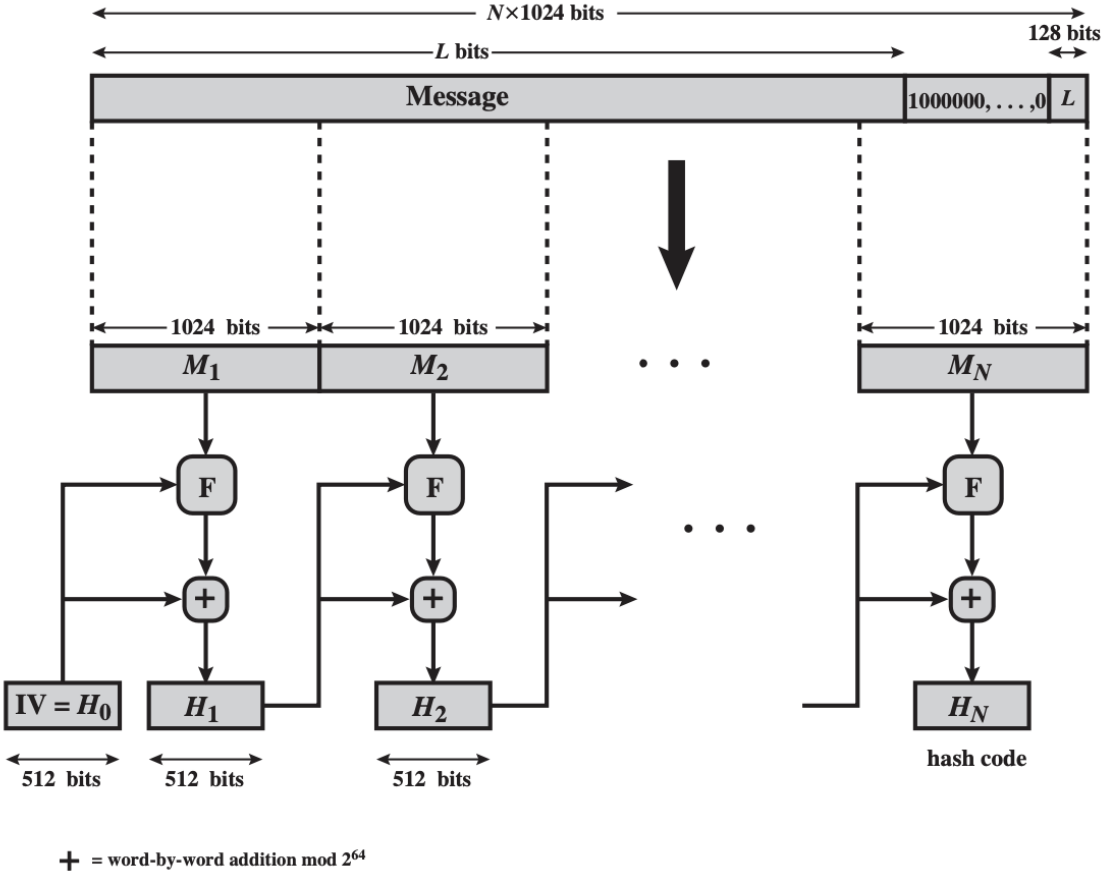


Figure 11.8 Message Digest Generation Using SHA-512

- STEP 1 : **Append padding bits**:
  Padding is always added, even if the message is already of the desired length.The padding **consists** of a single 1 bit followed by the necessary number of 0 bits.
- STEP 2: **Append Length:**
  A block of **128 bits** is appended to the message as L.
- STEP 3: **Initialize Hash Buffer:**
  A **512-bit buffer** is used to hold intermediate and final results of the hash function. The buffer can be represented as **eight 64-bit registers** (a, b, c, d, e, f, g, h). These registers are initialized to **64-bit integers** (hexadecimal values). These values are stored in **big-endian** format

- STEP 4: **Process message in 1024-bit blocks:**
  The heart of the algorithm is a module that consists of **80 rounds**; this module is labeled **F** . Each round **takes as input the 512-bit buffer** value, abcdefgh, and updates the contents of the buffer.
- STEP 5: **Output**:
  After all ***N* 1024-bit blocks have been processed**, the **output from the *N*th stage** is the **512-bit message digest.**

HMAC : MAC BASED ON HASH FUNCTION

Hash-based message authentication code (HMAC) is a cryptographic technique that uses a secret key and a hash function to verify the authenticity and integrity of a message.

K+ *K* padded with zeros on the left so that the result is *b* bits in length.

1. Append zeros to the left end of *K* to create a *b*-bit string *K +*

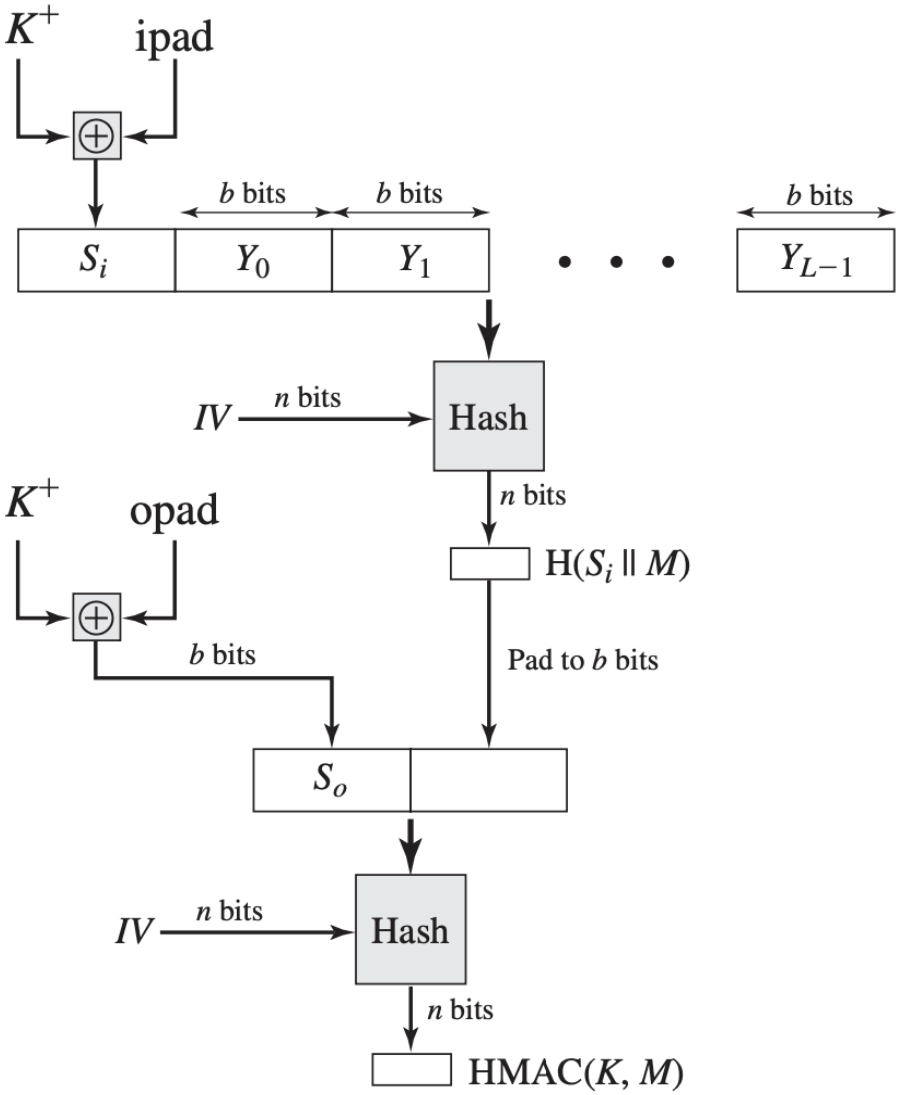


Figure 12.5 HMAC Structure

Digital Signatures and Authentication

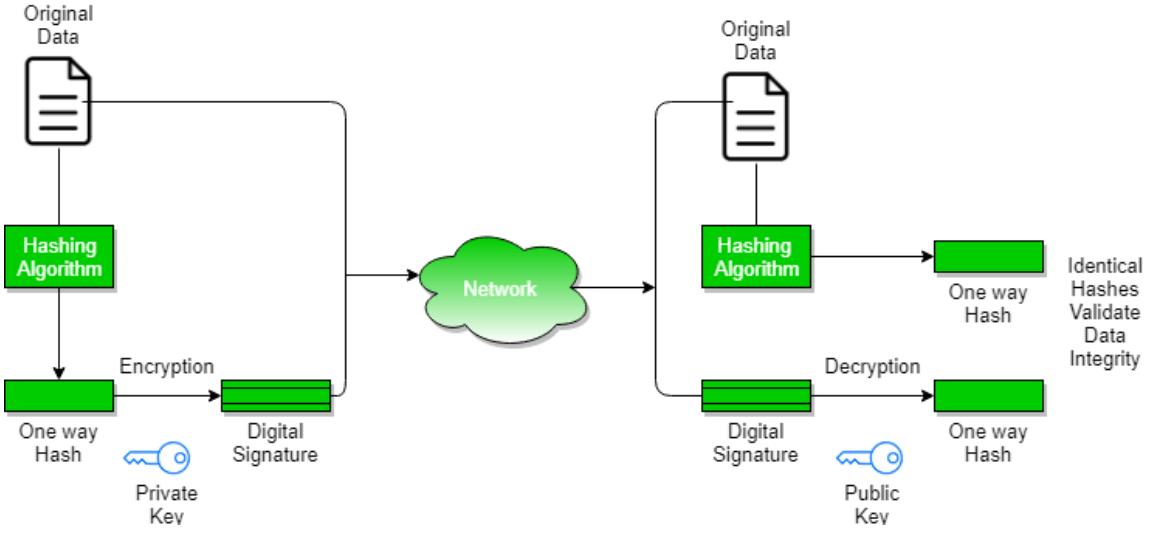
DIGITAL SIGNATURES:

- A digital signature is an **authentication mechanism that enables the creator of a message to attach a code that acts as a signature.**
- Typically the signature is formed by taking the **hash of the message** and **encrypting** the **message with the creator's private key.** The signature guarantees the source and integrity of the message.

A digital signature includes the following components:

- Public key

  Encrypts and secures the document at the time of **authorization**. The public key is signed by a trusted **Certificate Authority (CA)** and must match the private key.

- Private key
  Required to decrypt the digital signature.
- Digital certificate
  A cryptographic file that contains the digital signature and binds the signing key to the entity.The CA issues the digital certificate, which includes the public key and the identity associated with the key.
- Expiration date
  Every digital signature has an expiration date, which is determined by the public key.
- 

DIGITAL CERTIFICATES:

Digital certificate is issued by a trusted third party which proves sender's identity to the receiver and receiver's identity to the sender.

A digital certificate is a certificate issued by a Certificate Authority (CA) to verify the identity of the certificate holder. Digital certificate is used to attach public key with a particular individual or an entity.

Digital certificate contains

A digital certificate is an **electronic document** used to **prove the ownership of a public key.** It contains information that helps **verify the authenticity** of the certificate and the **identity** of the entity (person, organization, or device) to whom it is issued.

The contents of a digital certificate typically include:

1. **Certificate Authority (CA) Information**:
   - The name of the **trusted entity** (Certificate Authority) that issued the certificate.
   - The **CA's digital signature**, which validates the certificate.
2. **Certificate Holder Information**:
   - The **name or identity** of the certificate holder (individual, organization, or device).

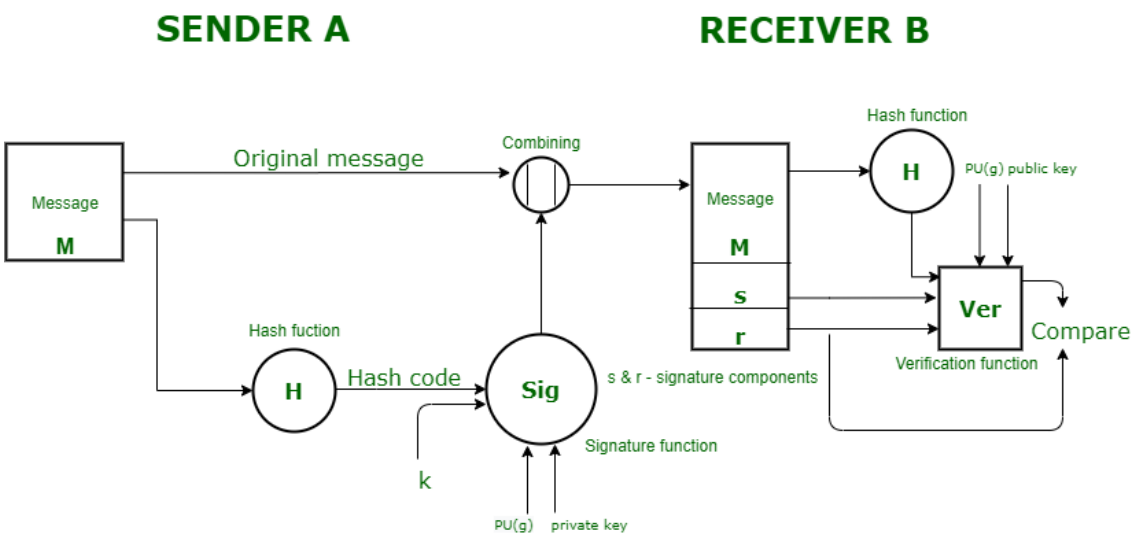
- Information about the entity's **organizational unit**, such as department or company.
- **Geographic details**, including country and location (optional).

3. **Public Key**:
   - The public key associated with the **certificate holder**.
   - This key is used for **encryption** or **verifying** digital signatures.
4. **Serial Number**:
   - A **unique identifier** assigned by the CA to the certificate for **identification** purposes.
5. **Validity Period**:
   - The **start date** and **expiration date,** indicating when the certificate is valid.
6. **Signature Algorithm**:
   - The algorithm used by the CA to **create the digital signature** of the certificate, e.g., RSA, ECDSA.
7. **Version**:
   - The version of the **X.509 standard** used for the certificate (commonly version 3).

Features	Digital Signature	Digital Certificate
Definition	It is similar to a **fingerprint** or an attachment to a digital document that verifies its **validity** and **integrity**.	It is a file that verifies the **identity** of the holder and offers **security**.
Security	It offers **non-repudiation**, **authentication**, and **integrity**.	It offers **security** and **authentication**.
Purpose	To ensure that the document sent between the sender and the receiver has not been **altered**.	The primary function of digital certificates is to increase **trust** between the client and the site owner.
Works on	Its working is based on the **Digital Signature Standard (DSS).**	Its working is based on **encryption securities** and **cryptographic keys.**
Use of Security	It utilizes the **hashing function**.	It utilizes **cryptographic keys.**
Creation	The **SHA-1** or **SHA-2** algorithms are utilized to generate digital signatures.	The **X.509** format is utilized to create digital certificates.

DIGITAL SIGNATURE STANDARDS

- **Digital Signature Standard (DSS)** is a **Federal Information Processing Standard(FIPS)** which **defines algorithms** that are used to generate digital signatures with the help of Secure Hash Algorithm(SHA) for the authentication of electronic documents.
- DSS **only provides us with the digital signature function** and not with any encryption or key exchanging strategies.

- 

- **Sender Side:** In DSS Approach, a hash code is generated out of the message and following inputs are given to the signature function –
  - The **hash code.**
  - The **random number 'k'** generated for that particular signature.
  - The **private key** of the sender i.e., PR(a).
  - A **global public key**(which is a set of parameters for the communicating principles) i.e., PU(g).
- **Receiver Side :** At the receiver end, **verification** of the sender is done. The **hash code** of the sent message is generated. There is a **verification function** which takes the following **inputs** –
  - The **hash code** generated by the receiver.
  - **Signature components** 's' and 'r'.
  - **Public key** of the sender.
  - **Global public key.**
- The output of the verification function is compared with the signature component 'r'.

Q.What is Kerberos? How Kerberos authenticates the users for authorized service access?
Q. Summarze the kerbos authentication system
Q. Explain kerbos in detail

Kerberos:
Kerberos is a **network authentication protocol** designed to provide strong authentication for client-server applications by using **secret-key cryptography.** It was developed by the Massachusetts Institute of Technology (**MIT**) and is widely used in **systems** that require secure, **mutual authentication**, such as in enterprise networks.

Key Features of Kerberos:

- Centralized Authentication: It uses a trusted third party (the Key Distribution Center, or KDC) to verify the identity of both the user and the service.
- Ticket-Based Authentication: Authentication and access control are based on tickets rather than reusable passwords.
- Mutual Authentication: Both the client and server verify each other's identity.
- Time-Sensitive: Kerberos relies on timestamps to prevent replay attacks, ensuring that credentials are only valid for a short period.

Components of Kerberos:

- Key Distribution Center (KDC): A trusted server that provides authentication and ticket-granting services.
- Authentication Server (AS): Verifies the user's identity and issues a Ticket Granting Ticket (TGT).
- Ticket Granting Server (TGS): Issues service-specific tickets based on the TGT.
- Client/User: The user or entity requesting access to a service.
- Service Server (SS): The server hosting the service that the client wants to access.
- Tickets: Time-limited cryptographic tokens that verify identity and grant access.

How Kerberos Authenticates Users for Authorized Service Access:

1. **Client Authentication Request:**

- The client sends an authentication request to the Authentication Server (AS) with their **username**.
- This request is sent in **plaintext**.

2. **Authentication Server(AS) Verifies User and Issues Ticket Granting Ticket (TGT):**

- The AS verifies the user's credentials by looking up the user in its **database**.
- If valid, the AS sends back an **encrypted** Ticket Granting Ticket (**TGT**) and a **session key** to the client.
- The **client's** copy of the **TGT** is encrypted using the user's **password-derived key.**

3. **Client Requests Service Ticket:**

- The **client decrypts the TGT** using their **password**, **retrieves** the **session key**, and sends the TGT to the **Ticket Granting Server** (TGS) along with a request for access to a specific service.

4. **TGS Issues a Service Ticket:**

- The TGS decrypts the TGT using its own secret key, validates it, and issues a Service Ticket.

5. **Client Presents Service Ticket:**

- The client sends the Service Ticket to the target **Service Server (SS)**, requesting access to the service.
- The service server ensures the **ticket is still valid.**

6. **Access Granted:**

- If the Service Server successfully validates the Service Ticket, it grants the client access to the requested service.
- Optionally, the service server may respond to the client, confirming **mutual authentication.**

Q. Describe X.509 authentication service

Describe X.509 authentication service.

X.509 is a widely-used **standard for public key infrastructure (PKI)** that **defines the structure of digital certificates** used in authentication services.

It is the basis for securing communications over the internet, using public key cryptography to authenticate entities (users, servers, or devices) in systems such as SSL/TLS, S/MIME, and more.

Authentication Process:

1. **Certificate Issuance**:
   - The entity (client or server) generates a public-private key pair.
   - The entity creates a **Certificate Signing Request (CSR)** containing its public key and identity details and sends it to a trusted **Certificate Authority (CA)**.
   - The CA verifies the entity's identity, and issues the X.509 digital certificate.
2. **Authentication Using X.509 Certificates**: When a client or server uses an X.509 certificate for authentication, the process follows these steps:
   a. **Client/Server Certificate Presentation**:
      - During the **authentication handshake** (such as an SSL/TLS connection), the entity presents its X.509 certificate to the peer (e.g., client to server, or server to client).
   b. **Certificate Validation**:
      - The peer verifies the certificate by checking:
        - The **issuer's signature** to ensure the certificate was signed by a trusted CA.
        - The **validity period** to confirm the certificate is not expired.
        - The **revocation status** to ensure the certificate hasn't been revoked.
   c. **Trust Chain Verification**:
      - If the presented certificate was signed by an intermediate CA, the verifier checks the **certificate chain**, ultimately verifying that the **Root CA** is trusted.

d. **Secure Communication**:
   - Once the certificate is validated, the peer can securely exchange information using the public key in the certificate to encrypt data or verify digital signatures.

3. **Mutual Authentication** (Optional):
   - In some cases, both parties (client and server) will authenticate each other by exchanging X.509 certificates. This is known as **mutual authentication** and is used in applications requiring a high level of security.

PUBLIC KEY INFRASTRUCTURE

Public key infrastructure or PKI is the **governing body behind issuing digital certificates**. It helps to **protect confidential data** and **gives unique identities to users and systems.**

The public key infrastructure uses a pair of keys: the public key and the private key to achieve security.

Architecture:

- **End entity:** A generic term used to denote end users, devices (e.g., servers, routers)
- **Certification authority(CA):**The issuer of certificates and(usually)certificate revocation lists (CRLs).
- **Registration authority (RA):** An optional component that can assume a number of administrative functions from the CA.
- **CRL(Certification Revocation Lists) issuer:** An optional component that a CA can delegate to publish CRLs.
- **Repository:** A generic term used to denote any method for storing certificates and CRLs so that they can be retrieved by end entities.

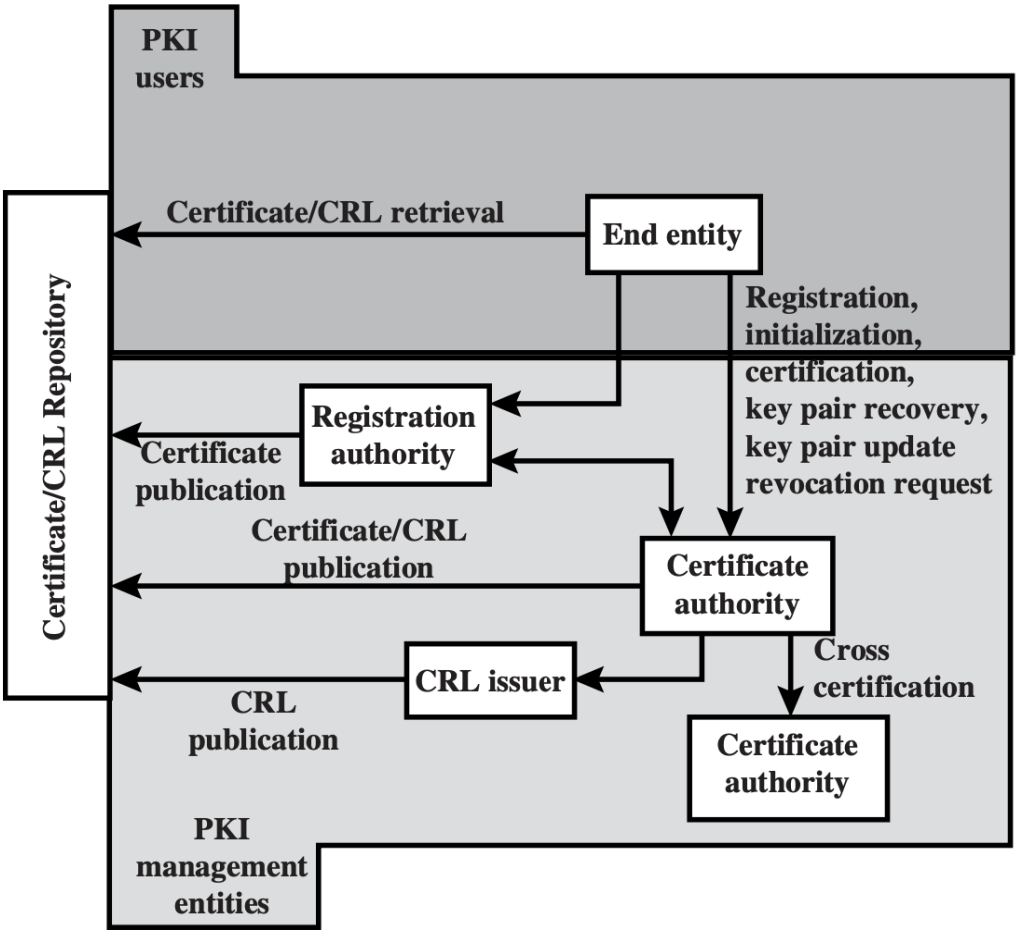
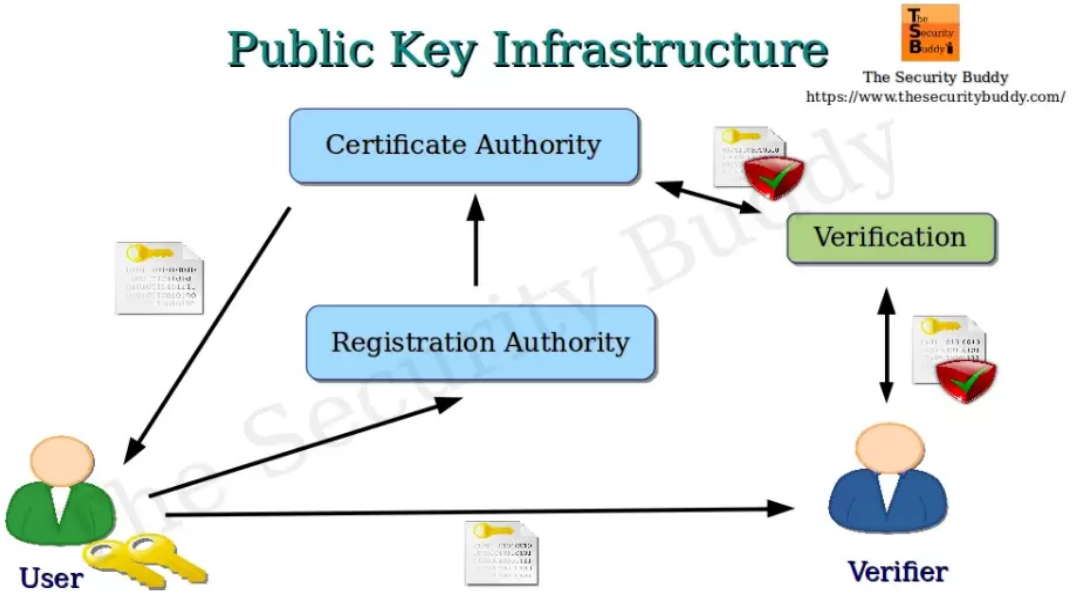


Figure 14.16 PKIX Architectural Model



Network Access Control

Q. Describe IEEE 802.11 Wireless Security with Wi-Fi Protected Access (WPA).

Wireless Network Security

MOBILE DEVICE SECURITY:

Mobile device security involves implementing various measures to protect mobile devices (smartphones, tablets, etc.) from threats like **data breaches, malware, unauthorized access, and theft.**

Below is a detailed explanation of key mobile security practices:

1. **Enable Auto-Lock:** Automatically lock the device **after a period of inactivity** to prevent unauthorized access. This ensures that unattended devices stay secure.
2. **Use SSL/TLS**: SSL/TLS ensures **secure browsing** and app interactions.
3. **Set Strong Passwords/PINs**: Use complex passwords or PINs to **strengthen device authentication.** This prevents unauthorized users from easily accessing the device.
4. **Disable Auto-Completion for Passwords**: Turn off automatic password filling to **avoid storing credentials** that attackers could exploit. Manual entry ensures higher password security.
5. **Enable Remote Wipe:** Allow administrators or users **to remotely erase all data on the device** in case of loss or **theft**. This prevents sensitive information from being accessed by unauthorized individuals.
6. **Install Anti-Virus Software**: Use trusted anti-virus software **to detect and remove malware** on the device.
7. **Encrypt Sensitive Data**: **Apply encryption** to protect personal and business data stored on the device.
8. **Prohibit Third-Party App Installations**: Restrict app downloads to trusted app stores to avoid installing **malicious software.**

WIRELESS LAN SECURITY

WLAN stands for wireless local area network, meaning a type of network that **enables devices** like laptops, smartphones, and tablets**to communicate** with each other **and access the Internet wirelessly** (i.e., **without physical cable connections**).

WLANs are commonly used in homes, offices, schools, and other public spaces like coffee shops and hotels to connect devices to the Internet.

Some of the technologies employed in WLAN security include:

- **Wired Equivalent Privacy (WEP):**
  - An **old encryption standard** used to **overcome security threats.**

    - WEP provides security to WLAN by **encrypting the information transmitted** over the air so that only the receivers with the correct encryption key can decrypt the information.
- **WPA/WPA2 (WI-FI Protected Access):**
    - Improved on WEP by **introducing Temporal Key Integrity Protocol** (TKIP).
    - TKIP uses a **temporal encryption key** that is **regularly renewed**, making it more difficult to steal.
- **Wireless Intrusion Prevention Systems/Intrusion Detection Systems:**
    - Intrusion detection and prevention f**ocuses on radio frequency (RF) levels.**
    - **Advanced implementations** are able to **visually represent the network area** along with potential threats, and have **automatic classification capabilities** so that threats can be easily identified.