Chapter 5
Public Key Cryptography & Message Authentication

I. Authentication

1. What is Authentication
   - It's about verifying the identity of a person or system.
   - For example, when you log in to your email you prove your identity with a password

2. Authentication Requirements
   - → The message must come from a valid source
   - → The message must not be altered
   - → (Sometimes) the message must have been sent at a specific time or in order.

3. Why It's important?
   - Protects agains active attacks (e.g. modifying messages, impersonation)

II. Approaches to Message Authentication

There are 2 main ways to authenticate messages:

1. Authentication using Conventional Encryption
   - Sender encrypts the message using a Shared Secret Key
   - Receiver decrypts it using the same Key

Key idea: Only Sender and receiver should have the key.

2. Message Authentication Without Message Encryp.

. Instead of encrypting the whole message, an authentication tag (small piece of data) is added

. Method : Message Authentication Code (MAC)

→ A small piece of data is generated From:

MAC = F(K, M) (MAC)

. K = Secret Key

M = Message

F = MAC Function

→ The Sender attaches the MAC to the message

→ The receiver recomputes the MAC and checks if it matches the received MAC.

→ IF both match → Message is authentic

. How MAC works (Step-by-Step)

1. Sender Side

. Sender has a message (M) to send

. Uses a secret Key (K_AB) to generate a MAC.

$MAC_M = F(K_{AB}, M)$

. Sends (Message + MAC) to the receiver

2. Receiver Side

. Uses the same secret Key (K_{AB}) to compute a new MAC from the received message

. Compares the received MAC with the computed one.

III- Hash Functions

1- What is a Hash Function?

. It takes any input message (M) and produces a Fixed-size output called a hash value or message digest (H)

. $h = H(M)$

↳ H: Hash Function
M: Input Message
h: Fixed Size hash value

. Purpose: Produce a fingerprint of a file/message

2- Properties of a good Hash Function

. A secure cryptographic hash function should have

→ Fixed Output Size: no matter how big the input is, the hash value is always the same length

→ Fast Compilation: Easy to compute H(M) for any given input M

→ One way property: Given h, it should be impossible to find the original message M

→ Weak Collision Resistance: Should be difficult to find another message with the same hash value

3- One-Way Hash Function

. Alternative to MAC

. Accepts a variable size message M and produces a fixed size digest H(M)

. Doesn't take a secret key as input

. Problem: Since hash functions don't use a secret key, they don't authenticate the sender, they only verify that the message was not modified
. Solution: HMAC (Hash-based Message Authentication Code)

4- Simple HASH Function

One of the simplest hash functions is bitwise XOR
. Divide the input message/file into sequence of n-bit blocks.
. Input is processed one block at a time to produce an n-bit hash function

$$Ci = bi1 \oplus bi2 \oplus \dots \oplus bim$$

↳ Ci : ith bit of the hash function
↳ m: nb. of n-bit blocks in the input
↳ bij: ith bit in the ij block

IV. Secure Hash Algorithm (SHA)

. SHA is a family of cryptographic hash functions designed for data integrity and security.
. Purpose:
→ Detects modification in data (integrity)
→ Produces a unique fixed size hash for any input
→ Used in digital signatures, authentication and encryption.

Types of SHA algorithms

Algorithm	Digest size	Block size	Rounds
SHA_1	160 bits	512 bits	80
SHA_256	256 bits	512 bits	64
SHA_512	512 bits	1024 bits	80

V. HMAC (Hash-based Message Authentication Code)

1. What is HMAC?

. Technique that enhances security by combining a cryptographic hash function (like SHA-1, SHA-256) with a secret key

2. Why HMAC?

. Hash functions alone don't authenticate the sender (no secrete Key)
. MAC alone may not be strong enough for security
. HMAC provides message integrity and authentication

3. Formula:

$$HMAC(K,M) = H\left((K^+ \oplus opad \,||\, H((K^+ \oplus ipad)||M)\right)$$

→ K: Secret Key
→ M: Message
→ H: Hash function
→ K^+ : Key padded to a fixed length
→ ipad: Inner padding
→ opad: Outer padding
→ || : Concatenation

4. How HMAC works?

→ Sender side (Generating HMAC)

1. Pad the secret Key to required length (K^+)
2. Compute the inner hash using the padded Key and the message.
3. Compute the outer hash using the inner hash output
4. Final output is the HMAC.

→ Receiver Side (Verifying HMAC) :

1. Receive (Message + HMAC) from the sender
2. Recompute HMAC using the same Key and hash function.
3. Compare computed HMAC with received HMAC
   ↳ If they match → Message is authentic.

VI. Public_Key Cryptography

1. What is public_key cryptography ?

. An encryption system that uses Two Keys
   → Public Key (K^+) : Shared with everyone
   → Private Key (K^-) : Kept Secret by the owner

. Key Concept:
   → Anything encrypted with one Key can only be decrypted with the other Key

2. Applications of Public_Key Cryptography

. Encryption / Decryption : Protects sensitive data (emails, HTTPS)
. Digital Signatures : Authenticates senders (used in blockchain, certificates)
. Key Exchange : Securely shares encryption key (used in TLS, SSH)

3- Requirements for Public-Key Cryptography

- → Easy Key Generation
- → Efficient Encryption/Decryption
- → Hard to compute private key
- → Hard to reverse encryption
- → Both keys must be interchangeable

VII - Public-Key Cryptographic Algorithms

The two most widely used algorithms are:

1) RSA : used for encryption and digital signatures
2) Diffie-Hellman: Used for key exchange

1) The RSA Algorithm (Rivest-Shamir-Adleman)

- . Asymmetric encryption algorithm
- . Provides confidentiality, integrity, and authentication

. RSA Key Generation (Step-by-Step)

To generate a public-private key pair:

1) Select two large prime numbers p and q
2) Compute $n = p \times q$ (Key size)
3) Compute Euler's Totient Function:
   $\phi(n) = (p-1) \times (q-1)$
4) Choose a public exponent e such that:
   $1 < e < \phi(n)$, $\gcd(e, \phi(n)) = 1$
   (common choices: 3 / 7 / 11 ...)
5) Compute the private key d
   $d = e^{-1} \bmod \phi(n)$
   $d \cdot e = 1 \bmod \phi(n)$

6) Public Key : (e.n) , Private Key : (d.n)

. RSA Encryption and Decryption

→ Encryption (Sender Side)

$C = M^e \bmod n$ M: Message (as a nb.)
C: Encryption message

→ Decryption (Receiver Side)

$M = C^d \bmod n$

Only the receiver (who had d) can decrypt it

I) . Examples

1) $p = 17$, $q = 11$

2) $n = p \times q = 17 \times 11 = 187$

3) $\phi(n) = (p-1)(q-1) = (16 \times 10) = 160$

4) e? $\gcd(\phi(n), e) = 1$ $1 < e < \phi(n)$
*should be prime $\Rightarrow 1 < e < 160$

options : $e = 3$, $e = 7$, $e = 11$

5) $d.e = 1 \bmod (160)$; $d < 160$

$e.d = 1 + K.\phi n$

$3 \times d = 1 + K.160$ } $K = 1$

$3d = 161$

$d = 53.66$ (not integer) $\Rightarrow e = 3$ X

Try $e = 7$

$7 \times d = 161$

$d = 23$ ✓ Then $e = 7$

6) Public Key : $KU\{e, n\} = \{7, 187\}$

Private Key $KR\{d, n\} = \{23, 187\}$

II) 1) $p = 3 \quad q = 11$

2) $n = p \times q = 33$

3) $\phi(n) = (p-1)(q-1) = 2 \times 10 = 20$

4) e ?

$\left.\begin{array}{l} \gcd(\phi(n), e) = 1 \\ \gcd(20, e) = 1 \\ 1 < e < 20 \end{array}\right\}$ e can be 3, 7, 11, ...

5) $d.e = 1 \bmod (\phi(n))$

$d.e = 1 \bmod (20) \Rightarrow d < 20$

$e.d = 1 + K.\phi(n)$

Let $K = 1$ and $e = 3$

$3.d = 1 + 20$

$3.d = 21$

$d = 7$

6) Private Key : $KR = \{d, n\} = \{7, 20\}$

Public Key : $KU = \{e, n\} = \{3, 20\}$

III) 1) $p = 5$, $q = 7$, $M = L$ (12 in alphabetical order)

2) $n = q \times p = 7 \times 5 = 35$

3) $\phi(n) = (p-1)(q-1) = 4 \times 6 = 24$

4) e = ?

$\left.\begin{array}{l} \gcd(24, e) = 1 \\ 1 < e < 24 \end{array}\right\}$ e can be 5, 7, 11 ..

5) $e.d = 1 \bmod (24) \Rightarrow d < 24$

$e.d = 1 + K\phi(n)$

↳ let $K = 1$ and $e = 5$

$5.d = 25$

$d = 5$ and $e = 5$ (e and d are the same : not logical)

↳ let $K = 6$, $e = 5$

$5.d = 1 + 6(24)$

$5.d = 145$

$d = 29$ ✓

⇒ PublicKey : $KU = \{5, 35\}$ → For encryption

Private Key : $KR = \{29, 35\}$ → For decryption

. Encryption :

$C = M^e \bmod n$

$C = 12^5 \bmod 35$

248832	35
248815	7109
17	

⇒ $C = 17$

. Decryption : $M = C^d \bmod n = 17^{29} \bmod 35$