UNIT-3

* Application of Cryptographic Hash functions:

→ Cryptographic hash functions are mathematical algorithms that transform input data into a fixed size string of characters.

*→ Data Integrity Verification:

- ensure that data has not been altered during transmission or storage.
- hash value is calculated and sent along with data.

*→ Digital Signatures and Certificates:

- Authenticate the origin & integrity of messages or documents.
- Hash fns are used to create a digest of the data; which is then encrypted with a private key to create digital signature.

*→ Message Authentication Codes:

- Authenticates messages & ensure they haven't been tampered.
- a cryptographic hash fn, combined with secret key, generates a MAC.

* Requirements of cryptographic hash functions:

1. Deterministic O/P
2. Fixed output size
3. Pre-image resistance
4. Second Pre-image instance
5. Collision resistance
6. Avalanche effect
7. Computational efficiency
8. Uniform distribution

* Security of cryptographic hash functions:

1. Pre-image attack.
   - ↳ Finding I/P that matches a specific hash value.
2. Second pre-image attack:
3. Collision attack.
4. Brute force attack
5. Side channel attack
6. Cryptanalysis

Ex: SHA-256 / SHA-512.

* Secure Hash Algorithm: (SHA)

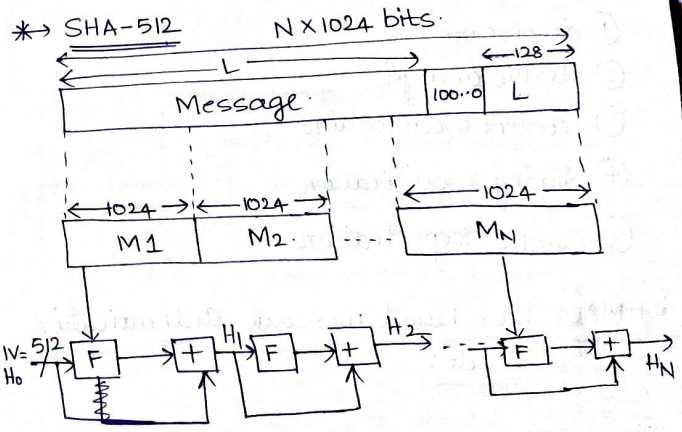
- ↳ US standard for use with DSA signature scheme.
- ↳ produces 160 bit hash values.

Versions of SHA: SHA-1, SHA-256, SHA-384, SHA-512

→ designed for compatibility with ↑sed security provided by AES cipher.

→ Security levels are higher.

* SHA-512



→ processing message in 1024 bit blocks.

→ consists of 80 rounds.

* updates a 512 bit buffer.
* a round constant based on cube root.

* Message Authentication:

→ concerned with:
- protecting the integrity of a message.
- validating identity of originator.

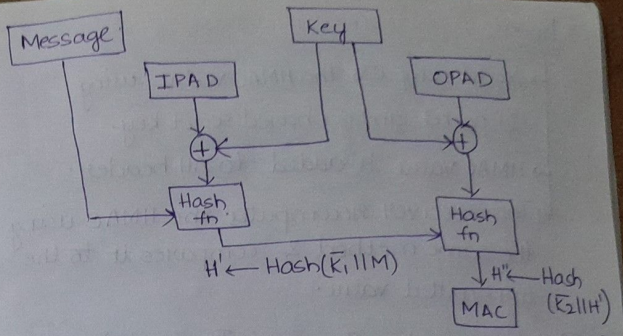
→ 3 alternative functions:
- message encryption
- message authentication code (MAC)
- hash function

* Security Requirements:

1. disclosure
2. traffic analysis
3. content modification
4. timing modification
5. source repudiation

* HMAC: Hash based message authentication code:

Components:
- Hash function: SHA-256
- Secret Key: shared only to the sender & receiver.
- Message.



→ $HMAC(K,M) = H((K \oplus opad) || H((K \oplus ipad) || M)$

K → secret key
M → message
H → cryptographic hash fn
|| : concatenation.

→ Properties:
- Integrity: ensures the message hasn't been altered.
- Authentication: verifying the sender.

→ Applications:
- TLS/SSL for communications.
- APIs for requests.
- Network protocols like IPsec.

* IPsec: Internet Protocol Security:

1. Authentication Header (AH)
   - AH ensures integrity & authenticity of IP packet.

Process:

- Sender computes the HMAC value using IP packet and a shared secret key.
- HMAC value is added to AH header.
- The receiver recomputes the HMAC using the same method & compares it to the transmitted value.

② Encapsulating Security Payload (ESP)

- provides both confidentiality & authentication.

Process:

- payload is encrypted using a symmetric encryption algorithm.
- computed over encrypted ~~packet~~ payload.
- HMAC value is appended to the ESP packet as an Integrity Check Value (ICV)

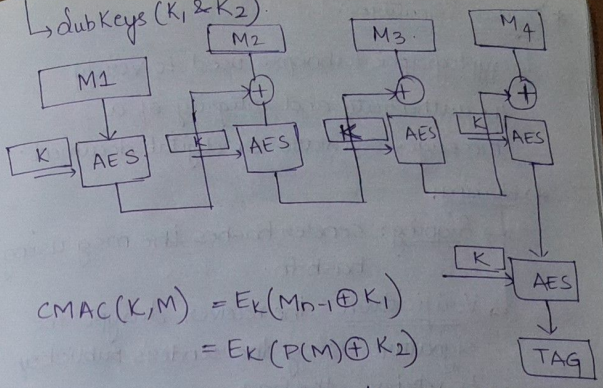
* CMAC: Cipher-based Message Authentication Code:

- based on symmetric block cipher.

Components:

- Block Cipher: symmetric cipher ex: AES.
- Secret Key: used for encryption & MAC generation.

- SubKeys (K_1 & K_2).



$CMAC(K,M) = E_K(M_{n-1} \oplus K_1)$

$= E_K(P(M) \oplus K_2)$

E_K = Encryption using the secret key.

M_{n-1} = 2nd to last msg block.

$P(M)$ = Padded message.

Properties:

- Message integrity
- Security
- Efficiency.

Applications:

- Network security protocols
- Storage of data.
- Embedded systems.

HMAC

- → Hash fn.
- → Fast for hashing
- → High security, due to hashing
- → API authentication

CMAC

- → Symmetric block cipher.
- → Efficient for block ciphers
- → high security due to block cipher
- → Wireless Networks.

* Digital Signatures:

↳ cryptographic technique used to verify the authenticity and integrity of a message, software or digital document.

*↔ Working:

↳ Signing: sender hashes the msg using hash fn.

↳ Verification: the receiver decrypts the signature using the sender's public key to retrieve the hash.

* ElGamal Digital Signature Scheme:

↳ uses modular arithmetic & discrete logarithms.

Steps:

① Key Generation:

→ Choose a large prime p and a generator g.

→ select private key x; compute $y = g^x \bmod p$.

② Signing:

→ Choose random k

→ $r = g^k \bmod p$

→ $s = (H(m) - xr) k^{-1} \bmod (p-1)$.

③ Verification:

$$g^{H(m)} \equiv y^r r^s \bmod p.$$

→ Used in lightweight applications.

* NIST: Digital Signature Algorithm:

Steps:

① Key Generation:

→ generate a prime p and a divisor q and select a generator g.

→ private key x; compute $y = g^x \bmod p$.

② Signing:

$r = (g^k \bmod p) \bmod q$

$s = k^{-1} (H(m) + xr) \bmod q$

③ Verification:

$v = ((g^{u_1} y^{u_2}) \bmod p) \bmod q$.

$u_1 = H(m) s^{-1} \bmod q$

$u_2 = r s^{-1} \bmod q$.

verify $v = r$.

* Key Management and distribution:

① Symmetric Key distribution:

→ Manual Key distribution: physically sharing the key.

→ Key distribution center: A central trusted authority distributes keys.

→ Diffie-Hellman Exchange.

② Distribution of Public Keys:

→ Public Key Infrastructure (PKI):

↳ uses X.509 certificates for trusted public key distribution

→ Public directories stores public keys for retrieval.

→ Key Exchange Protocols.

③ X.509 certificates:

→ Standard for public key certificates.

→ includes: key, identity, certificate issuer, digital signature of CA

→ used for SSL/TLS for secure communication.

* User Authentication:

→ Authenticate remote users securely using methods like passwords, biometrics/tokens.

→ Kerberos:

↳ network authentication protocol using symmetric cryptography & a trusted 3rd party.

Steps:

↳ Client requests a ticket from CDC
↳ KDC issues a ticket encrypted with the user's key.
↳ ticket is used for authentication.

UNIT-4:

* Transport layer Security:

① SSL/TLS: (Secure Sockets layer)

↳ Secure communication over internet.
↳ Features: Encryption, authentication, integrity (MAC).

Process:

① Handshake:

→ client initiates a connection to the server & requests security.
→ server responds with SSL/TLS; contains public key.
→ client verifies using certificate authority
→ client & server negotiate the encryption methods.

② Session Key Exchange:

→ Session key shared b/w client & server is used to encrypt & decrypt the data.

③ Secure Communication:

Encrypted data is exchanged using session key.

Applications: HTTP & FTPS.

* SSH: Secure Shell.
  └ secure remote login & other services over an unsecured network.

Components:
- Encryption
- Authentication
- Integrity

Applications:
- Remote login
- File Transfer

Key features:
- Port forwarding
- Secure Command Execution
- Tunneling.

* Web Security Requirements:
  - Encryption
  - Authentication
  - Message Integrity
  - Non-repudiation.

* Email Security:

→ Internet Mail Architecture:
  - SMTP: Simple Mail Transfer Protocol: (send mails)
  - POP3: Post Office Protocol 3: retrieve mails
  - IMAP: Internet Message Access Protocol: retrieve & manage mails; synchronization.

* Email formats: Plain Text, HTML

* → Email Threats:
  - Phishing
  - Spam
  - Man in middle attacks
  - Spoofing.

* Strategies for Secure Email Communication:

① Encryption:

→ TLS: encrypts mails
→ End to end encryption.
→ PGP: Pretty Good privacy:
  ↳ combination of asymmetric & symmetric encryption for secured email content.
→ S/MIME (Secure/Multipurpose Internet Mail Extensions):
  ↳ X.509 certificates are used for encrypting & signing emails.

② Authentication:

→ SPF: Sender Policy Framework:
  ↳ check for authorized server.
→ DKIM: DomainKeys Identified Mail:
  ↳ adds digital signature to email.
→ DMARC: Domain-based Message authentication, reporting & conformance.
  ↳ specifies how mail servers shd handle mails.

* PGP: Pretty Good Privacy:
  ↳ data encryption & decryption program that provides cryptographic privacy & authentication

Components:
- → Encryption: RSA for public key encryption
- → Signing: digital signatures
- → Hashing: SHA

Process:
- → Sender encrypts the message with recipient's ~~key~~ public key.
- → Recipient decrypts it with their private key.
- ↳ Sender may also sign the email using private key.

* S/MIME:
- → Standard for public key encryption & signing of emails.
- ↳ Uses X.509 certificates.

Features:
- ↳ Encryption & digital signatures.

Process:

1. **Key-Pair**: user generates a public-private key pair & requests a certificate from CA.

2. **Signing & Encrypting**:
   - An email is signed with sender's private key.
   - It is then encrypted with recipient's public key.

3. **Verification**:
   - recipient decrypts the msg with their private key & verifies signature using sender's public key.

Applications:

- Secure email communication
- Personal email security.