Cryptographic Algorithms - Simple Implementations

DES - Key Generation, Encryption, Decryption

DES (Data Encryption Standard) operates on 64-bit blocks and uses a 56-bit key.
Key Generation:
1. The key is initially 64 bits. Parity bits are removed, leaving 56 bits.
2. The key is split into two 28-bit halves and shifted according to predefined tables.
3. Subkeys are generated for each round of DES (16 rounds).

Encryption/Decryption:
1. The 64-bit plaintext is permuted using the initial permutation table.
2. The block is split into two halves: Left (L) and Right (R).
3. For each round:
   a. A function f(R, subkey) is applied using the subkey and the Right half.
   b. XOR the output of f with the Left half to generate the new Right half.
   c. Swap the halves for the next round.
4. After 16 rounds, the final permutation is applied to get the ciphertext.

Decryption is the reverse process of encryption.

AES - Mix Column

In AES, MixColumn is one of the four operations in a round of encryption.
MixColumn transforms each column of the state matrix by multiplying it with a fixed matrix.

Formula:
[b0] = [02 03 01 01] [a0]

[b1] = [01 02 03 01] [a1]

[b2] = [01 01 02 03] [a2]

[b3] = [03 01 01 02] [a3]

b0, b1, b2, b3: Output bytes of a column.

- a0, a1, a2, a3: Input bytes of a column.

The result is obtained by polynomial multiplication followed by modulo operation.

Extended Euclidean Algorithm

The Extended Euclidean Algorithm finds the GCD of two numbers a and b and also computes coefficients x and y such that:
gcd(a, b) = ax + by

Symbols:

- gcd(a, b): Greatest common divisor of a and b.

- x, y: Coefficients satisfying the equation.

Steps:
1. Start with a and b.
2. Perform the Euclidean Algorithm to compute the GCD using repeated division.
3. Trace back the steps to compute x and y by substituting remainders.

Chinese Remainder Theorem

The Chinese Remainder Theorem solves systems of congruences:
Given:

$$x \equiv a_1 \mod n_1$$

$$x \equiv a_2 \mod n_2$$

...

Where n1, n2 are coprime, the solution is:

$$x = \sum (a_i M_i y_i) \mod N$$

Here:
- $M_i = \frac{N}{n_i}$, where N is the product of all n_i.
- y_i is the modular inverse of $M_i \mod n_i$.

MD5 - Padding

MD5 processes messages in 512-bit blocks. Padding ensures the message length is a multiple of 512 bits.
Steps:
1. Append a '1' bit to the message.
2. Append enough '0' bits to make the message length 448 mod 512.
3. Append the original message length as a 64-bit value.

SHA - Padding

SHA uses a similar padding approach to MD5.
Steps:
1. Append a '1' bit to the message.
2. Append '0' bits until the length is congruent to 448 mod 512.
3. Append the message length as a 64-bit value.

RSA

RSA is a public-key cryptosystem based on number theory.
Steps:
Choose two primes p, q. Compute $n = p * q$, $\varphi(n) = (p-1)(q-1)$.

Choose e such that $1 < e < \varphi(n)$ and $\gcd(e, \varphi(n)) = 1$.

Compute d as the modular inverse of e mod $\varphi(n)$.

Formulas:

- Encryption: c = m^e mod n

- Decryption: m = c^d mod n

Symbols:

- c: Ciphertext.

- m: Plaintext.

- e: Public key exponent.

- d: Private key exponent.

- n: Modulus, product of two primes p and q.

ElGamal

ElGamal is an asymmetric encryption algorithm based on the Diffie-Hellman key exchange.
Steps:
1. Key Generation:
  a. Choose a large prime p and a generator g of the multiplicative group of integers modulo p.
  b. Select a private key x randomly, where 1 < x < p-1.
  c. Compute the public key y = g^x mod p.
2. Encryption:
  a. Represent the plaintext message m as an integer, 1 < m < p.
  b. Choose a random integer k, where 1 < k < p-1.
  c. Compute c1 = g^k mod p and c2 = (m * y^k) mod p.
  d. The ciphertext is (c1, c2).
3. Decryption:
  a. Compute s = c1^x mod p.
  b. Recover the plaintext: m = (c2 * s^(-1)) mod p, where s^(-1) is the modular inverse.

DSS (Digital Signature Standard)

DSS uses the DSA (Digital Signature Algorithm) for generating digital signatures.
Steps:

1. Key Generation:
   a. Select a large prime p and a 160-bit prime q such that q divides $(p-1)$.
   b. Choose a generator $g = h^{((p-1)/q)} \bmod p$, where h is any integer satisfying $1 < h < p-1$.
   c. Select a private key x, where $0 < x < q$.
   d. Compute the public key $y = g^x \bmod p$.
2. Signing:
   a. Choose a random integer k, where $0 < k < q$.
   b. Compute $r = (g^k \bmod p) \bmod q$.
   c. Compute $s = (k^{(-1)} * (H(m) + x * r)) \bmod q$, where $H(m)$ is the hash of the message.
   d. The signature is (r, s).
3. Verification:
   a. Compute $w = s^{(-1)} \bmod q$.
   b. Compute $u1 = (H(m) * w) \bmod q$ and $u2 = (r * w) \bmod q$.
   c. Verify $r = ((g^{u1} * y^{u2}) \bmod p) \bmod q$.

DSS-RSA

DSS-RSA is a variation of DSS that uses RSA for digital signatures.
Steps:
1. Key Generation: Generate RSA keys (n, e) and (n, d).
2. Signing:
   a. Hash the message using a cryptographic hash function (e.g., SHA-1).
   b. Encrypt the hash using the private key to generate the signature.
3. Verification:
   a. Decrypt the signature using the public key to obtain the hash.
   b. Recompute the hash of the message and compare.

ElGamal DSS

ElGamal DSS adapts the ElGamal algorithm for digital signatures.
Steps:
1. Key Generation: Same as ElGamal encryption.
2. Signing:
   a. Choose a random integer k, where $1 < k < q$.
   b. Compute $r = g^k \bmod p$.
   c. Compute $s = (H(m) - x * r) * k^{(-1)} \bmod q$.
   d. The signature is (r, s).
3. Verification:
   a. Verify r and s satisfy $0 < r, s < q$.
   b. Compute $v1 = g^{(H(m))} \bmod p$ and $v2 = (y^r * r^s) \bmod p$.
   c. Signature is valid if $v1 \equiv v2 \bmod p$.

Diffie-Hellman Key Exchange (DH)

DH is a key exchange protocol for secure key agreement.
Steps:
1. Both parties agree on a prime p and a generator g.

2. Each party chooses a private key (a for Alice, b for Bob) and computes their public keys:
   a. $A = g^a \bmod p$
   b. $B = g^b \bmod p$
3. The public keys are exchanged.
4. Each party computes the shared secret:
   a. Alice computes $S = B^a \bmod p$.
   b. Bob computes $S = A^b \bmod p$.

The shared secret S is identical for both parties.

MITM - Diffie-Hellman Key Exchange

A Man-in-the-Middle (MITM) attack on DH intercepts and alters public key exchanges.
Steps:
1. Attacker intercepts public keys A and B.
2. Attacker generates its own private keys (e.g., m) and computes:
   a. $A' = g^m \bmod p$ (sent to Alice as Bob's key).
   b. $B' = g^m \bmod p$ (sent to Bob as Alice's key).
3. Attacker computes shared secrets with both Alice and Bob.
   a. With Alice: $S1 = A^m \bmod p$.
   b. With Bob: $S2 = B^m \bmod p$.
4. Attacker decrypts and re-encrypts messages between Alice and Bob.