Assignment 17

Q1. Explain Data Encryption Standard (DES) and Rivest-Shamir-Adleman (RSA) Algorithms.

Data Encryption Standard (DES)
DES is a symmetric-key algorithm for the encryption of electronic data. It was developed in the early 1970s at IBM and adopted as a federal standard in the United States in 1977.

Key Size: 56 bits
Block Size: 64 bits
Structure: DES uses a Feistel network with 16 rounds of processing.
Subkey Generation: DES generates 16 subkeys, one for each round, from the original key.
Encryption Process:

1. Initial Permutation (IP): The 64-bit plaintext block is permuted.
2. 16 Rounds of Processing: Each round consists of:
   Expansion: Expanding 32 bits to 48 bits.
   Subkey Mixing: XOR with the subkey.
   Substitution: Using S-boxes to transform 48 bits back to 32 bits.
   Permutation: Rearranging the bits.
3. Final Permutation (FP): Inverse of the initial permutation.

Security: While DES was secure for a time, its 56-bit key size is now considered too small, making it vulnerable to brute-force attacks.

Rivest-Shamir-Adleman (RSA)
RSA is a widely-used public-key cryptosystem for secure data transmission. It was invented by Ron Rivest, Adi Shamir, and Leonard Adleman in 1977.

Key Generation:

1. Choose two large prime numbers, ppp and qqq.
2. Compute n=pqn = pqn=pq (the modulus).
3. Compute φ(n)=(p−1)(q−1)\phi(n) = (p-1)(q-1)φ(n)=(p−1)(q−1).
4. Choose an encryption key eee such that 1<e<φ(n)1 < e < \phi(n)1<e<φ(n) and gcd(e,φ(n))=1\text{gcd}(e, \phi(n)) = 1gcd(e,φ(n))=1.
5. Compute the decryption key ddd such that ed≡1 (mod φ(n))ed \equiv 1 \ (\text{mod} \ \phi(n))ed≡1 (mod φ(n)).
   Public Key: (e,n)(e, n)(e,n)
   Private Key: (d,n)(d, n)(d,n)

Encryption:
Ciphertext C is computed as C=Me (mod n)C = M^e \ (\text{mod} \ n)C=Me (mod n), where MMM is the plaintext.

Decryption:
Plaintext M is recovered as M=Cd (mod n)M = C^d \ (\text{mod} \ n)M=Cd (mod n).

Security: RSA's security relies on the difficulty of factoring large integers. Key sizes of 2048 bits or higher are considered secure.

Q2. Explain Diffie-Hellman Key Exchange Algorithm With an Example.

The Diffie-Hellman Key Exchange algorithm allows two parties to establish a shared secret over an insecure communication channel. It was proposed by Whitfield Diffie and Martin Hellman in 1976.

Steps:

1. Agree on a large prime ppp and a primitive root ggg.
2. Each party generates a private key:
   Alice chooses a private key aaa and computes A=ga (mod p)A = g^a \ (\text{mod} \ p)A=ga (mod p).
   Bob chooses a private key bbb and computes B=gb (mod p)B = g^b \ (\text{mod} \ p)B=gb (mod p).
3. Exchange public keys AAA and BBB.
4. Compute the shared secret:
   Alice computes S=Ba (mod p)S = B^a \ (\text{mod} \ p)S=Ba (mod p).
   Bob computes S=Ab (mod p)S = A^b \ (\text{mod} \ p)S=Ab (mod p).

Example:

p=23p = 23p=23, g=5g = 5g=5
Alice chooses a=6a = 6a=6, computes A=56 (mod 23)=8A = 5^6 \ (\text{mod} \ 23) = 8A=56 (mod 23)=8.
Bob chooses b=15b = 15b=15, computes B=515 (mod 23)=19B = 5^{15} \ (\text{mod} \ 23) = 19B=515 (mod 23)=19.
They exchange A=8A = 8A=8 and B=19B = 19B=19.
Alice computes S=196 (mod 23)=2S = 19^6 \ (\text{mod} \ 23) = 2S=196 (mod 23)=2.
Bob computes S=815 (mod 23)=2S = 8^{15} \ (\text{mod} \ 23) = 2S=815 (mod 23)=2.
Shared secret S=2S = 2S=2.

Q3. Explain Digital Signature Algorithm (DSA) With an Example.

DSA is a federal standard for digital signatures that was proposed by the National Institute of Standards and Technology (NIST) in 1991.

Key Generation:

1. Choose a prime q and a prime p such that p−1 is a multiple of q.
2. Choose g where g is a number whose order modulo p is q.
3. Choose a private key xxx such that 0<x<q.
4. Compute the public key y=g^x (mod p).
   Public Key: (p,q,g,y)
   Private Key: x

Signing:

1. Choose a random number k such that 0<k<q.
2. Compute r=(g^k (mod p)) (mod q).
3. Compute s=(k−1(H(m)+xr)) (mod q)s = (k^{-1} (H(m) + xr)) \ (\text{mod} \ q)s=(k−1(H(m)+xr)) (mod q), where H(m)H(m)H(m) is the hash of the message mmm.
   The signature is (r,s).

Verification:

1. Compute w=s−1 (mod q)w = s^{-1} \ (\text{mod} \ q)w=s−1 (mod q).
2. Compute u1=H(m)w (mod q)u_1 = H(m)w \ (\text{mod} \ q)u1=H(m)w (mod q) and u2=rw (mod q)u_2 = rw \ (\text{mod} \ q)u2=rw (mod q).
3. Compute v=((gu1yu2) (mod p)) (mod q)v = ((g^{u_1} y^{u_2}) \ (\text{mod} \ p)) \ (\text{mod} \ q)v=((gu1yu2) (mod p)) (mod q).
   The signature (r,s) is valid if and only if v=r.

Example:

Choose p=23p = 23p=23, q=11q = 11q=11, g=4g = 4g=4.
Private key x=6x = 6x=6, public key y=46 (mod 23)=9y = 4^6 \ (\text{mod} \ 23) = 9y=46 (mod 23)=9.
Signing:

Random k=3k = 3k=3, message m="Hello"m = "Hello"m="Hello", H(m)=2H(m) = 2H(m)=2.
Compute r=(43 (mod 23)) (mod 11)=2r = (4^3 \ (\text{mod} \ 23)) \ (\text{mod} \ 11) = 2r=(43 (mod 23)) (mod 11)=2.
Compute s=(3−1(2+6· 2)) (mod 11)=7s = (3^{-1} (2 + 6 \cdot 2)) \ (\text{mod} \ 11) = 7s=(3−1(2+6· 2)) (mod 11)=7.
Signature is (2,7)(2, 7)(2,7).

Verification:

Compute w=7−1 (mod 11)=8w = 7^{-1} \ (\text{mod} \ 11) = 8w=7−1 (mod 11)=8.
Compute u1=2· 8 (mod 11)=5u_1 = 2 \cdot 8 \ (\text{mod} \ 11) = 5u1 =2· 8 (mod 11)=5, u2=2· 8 (mod 11)=5u_2 = 2 \cdot 8 \ (\text{mod} \ 11) = 5u2=2· 8 (mod 11)=5.
Compute v=(45· 95 (mod 23)) (mod 11)=2v = (4^5 \cdot 9^5 \ (\text{mod} \ 23)) \ (\text{mod} \ 11) = 2v=(45· 95 (mod 23)) (mod 11)=2.
Since v=r, the signature is valid.