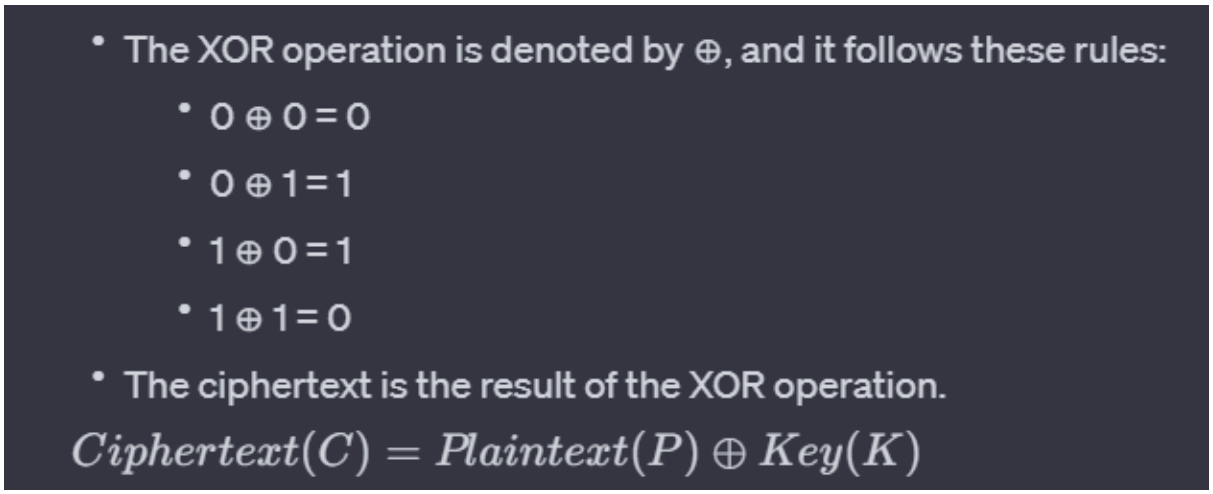
Theory

Prac 1

One-Time Pad

The One-Time Pad (OTP) is a type of encryption algorithm that provides perfect secrecy when used correctly. Here are the key principles and characteristics of the One-Time Pad:

1. **Key Generation:**
    - The key used in the OTP must be as long as the message being encrypted.
    - The key is generated randomly and should be truly unpredictable and used only once.
    - The key is typically a stream of random bits, and each bit is used only once.
2. **Encryption:**
    - The encryption process involves combining each bit of the plaintext message with the corresponding bit of the key using the XOR (exclusive OR) operation.



3. **Decryption:**

- The decryption process is the same as the encryption process. It involves XORing the ciphertext with the key to retrieve the original plaintext.

4. **Perfect Secrecy:**

    - The One-Time Pad provides perfect secrecy when used with a truly random and secret key.
    - Perfect secrecy means that even with unlimited computational resources, an adversary cannot gain any information about the plaintext.

5. **Key Management:**

    - The key must be kept completely secret between the communicating parties.
    - Each key should be used only once. Reusing a key compromises the security of the encryption.

6. **Practical Challenges:**

    - Generating and distributing truly random keys can be challenging.
    - The key must be as long as the message, which can be impractical for long messages.
    - Key distribution and management become significant challenges in practical scenarios.

While the One-Time Pad offers perfect secrecy in theory, its practical implementation faces challenges that make it less

suitable for many real-world scenarios. Key management, key distribution, and the need for long, truly random keys make it cumbersome. In practice, modern cryptographic algorithms like AES are more commonly used for secure communication.

Prac 2

Extended Euclidean Algorithm:

The Extended Euclidean Algorithm is an extension of the Euclidean Algorithm, which is used to find the greatest common divisor (GCD) of two integers.

Key Property:

- At each iteration, the equation *ax+by* remains constant, and the coefficients *x* and *y* are updated accordingly.

Primary purpose

of the Extended Euclidean Algorithm is to compute coefficients x and y such that $ax + by = \mathrm{GCD}(a, b)$ for given integers a and b.

Algorithm Steps:

1. **Initialization:**
   - Given two non-negative integers a and b.
   - Set $x_1 = 1, y_1 = 0, x_2 = 0, y_2 = 1$.
2. **Iterative Step:**
   - While $b \neq 0$:
     - Compute $q = \lfloor a/b \rfloor$ and $r = a \mod b$.
     - Set $a = b, b = r$.
     - Update x_1, y_1, x_2, y_2 as follows:
       - $x_1 = x_2, y_1 = y_2$,
       - $x_2 = x_1 - qx_2, y_2 = y_1 - qy_2$.

$x_2 = x_1 - qx_2, y_2 = y_1 - qy_2$.

3. **Result:**
   - The algorithm terminates when $b = 0$.
   - The values of x_1 and y_1 at this point are the coefficients such that $ax_1 + by_1 = \text{GCD}(a, b)$.

Advantages:

1. **GCD Calculation:**

The primary advantage of the Extended Euclidean Algorithm is its ability to efficiently calculate the greatest common divisor (GCD) of two integers.

2. Cryptographic Applications:

The Extended Euclidean Algorithm is widely used in cryptographic systems, such as RSA, for key generation and modular arithmetic.

Disadvantages:

1. **Computational Complexity:**

While the algorithm is efficient for GCD calculations, it may have higher computational complexity compared to simpler algorithms for specific tasks.

2. **Limited to Integers:**

The algorithm is designed for integer inputs and may not be directly applicable to non-integer or floating-point arithmetic.

3. **Key Management in Cryptography:**

   In cryptographic systems, the secure management of keys generated by the Extended Euclidean Algorithm is crucial.

The algorithm requires careful key handling to ensure the security of cryptographic protocols.

Prac 3

S-DES symmetric Encryption Algorithm

Simplified Data Encryption Standard (S-DES) is a simplified version of the original Data Encryption Standard (DES) algorithm. It was designed for educational purposes to provide a clearer understanding of the basic principles behind DES. S-DES operates on 8-bit blocks of plaintext and uses a 10-bit key for encryption and decryption. Below are the key steps of the S-DES algorithm:

S-DES Key Generation:

1. **Input:**
    - A 10-bit key (**K**) is provided as input.
2. **Key Permutation 1 (P10):**
    - The 10-bit key is permuted using the P10 permutation.
3. **Splitting:**
    - The permuted key is split into two 5-bit halves, left (**L**) and right (**R**).
4. **Key Circular Left Shifts:**
    - Both halves (**L** and **R**) are individually subjected to circular left shifts.

5. **Key Permutation 2 (P8):**
    - The halves (**L** and **R**) are concatenated, and a second permutation (P8) is applied to generate the final 8-bit key (**K1**).

S-DES Encryption:

1. **Initial Permutation (IP):**
    - The 8-bit plaintext block is subjected to an initial permutation.
2. **Splitting:**
    - The permuted block is split into two 4-bit halves, left (**L**) and right (**R**).
3. **Function F:**
    - The right half (**R**) is expanded to 8 bits using an expansion permutation.
    - The expanded half is XORed with the first subkey (**K1**).
    - The result is passed through a substitution (S-box) layer.
    - The output is then permuted using the P4 permutation.
    - The permuted output is XORed with the left half (**L**).
4. **Switching:**
    - The left and right halves are switched.
5. **Function F (Again):**

- The same function F is applied using the second subkey (**K2**).

6. **Inverse Initial Permutation (IP^(-1)):**
    - The final result is subjected to an inverse initial permutation to obtain the ciphertext.

S-DES Decryption:

1. **Key Generation:**
    - The key generation steps are repeated to obtain the subkeys (**K1** and **K2**).
2. **Encryption Steps in Reverse Order:**
    - The encryption steps are applied in reverse order using the subkeys (**K2** and **K1**) to decrypt the ciphertext and obtain the original plaintext.

Prac 4

RSA

The RSA (Rivest-Shamir-Adleman) algorithm is a widely used asymmetric cryptographic algorithm that enables secure data transmission and digital signatures. RSA involves the use of a public key and a private key, and it relies on the mathematical difficulty of factoring the product of two large prime numbers.

Key Generation:

1. **Select Two Large Prime Numbers:**
   - Choose two distinct large prime numbers, p and q.
2. **Compute n:**
   - Compute $n = pq$.
3. **Compute $\phi(n)$:**
   - Compute $\phi(n) = (p-1)(q-1)$, where ϕ is Euler's totient function.
4. **Choose e:**
   - Select a public exponent e such that $1 < e < \phi(n)$ and e is coprime to $\phi(n)$ (i.e., e and $\phi(n)$ share no common factors other than 1).
5. **Compute d:**
   - Calculate the private exponent d such that $ed \equiv 1 \pmod{\phi(n)}$.
6. **Public Key and Private Key:**
   - The public key is (n, e), and the private key is (n, d).

Encryption:

1. **Convert Message to Integer:**
   - Represent the plaintext message as an integer M in the interval $[0, n-1]$.
2. **Compute Ciphertext:**
   - Compute the ciphertext $C \equiv M^e \pmod{n}$.
3. **Send Ciphertext:**
   - Send the ciphertext C to the recipient.

Decryption:

1. **Compute Decryption:**
   - Compute the original message $M \equiv C^d \pmod{n}$.
2. **Convert to Plaintext:**
   - Represent the decrypted integer M as the original plaintext message.

Security Considerations:

1. Key Length:
   - The security of RSA relies on the difficulty of factoring the product *n*. Longer key lengths (e.g.,

2048 or 3072 bits) are recommended for higher security.

2. Randomness:
    - The choice of prime numbers and other parameters should be done with a good source of randomness.
3. Padding:
    - To enhance security, RSA is often used with padding schemes (e.g., PKCS#1 v1.5 or OAEP) to prevent certain attacks.
4. Key Management:
    - Secure key generation, storage, and distribution are crucial for the overall security of RSA.

Prac 5

Diffie – Hellman key exchange

The Diffie-Hellman key exchange is a method for two parties to agree on a shared secret key over an insecure communication channel. It was developed by Whitfield Diffie and Martin Hellman in 1976. The key exchange is based on the mathematical properties of modular exponentiation and the difficulty of the discrete logarithm problem.

1. Key Generation:

- Each party generates a public-private key pair.
- The key pair is based on a mathematical group, typically defined by a large prime number p and a primitive root modulo p denoted as g.
- Each party chooses a private key (a for party A and b for party B) in the range $1 < a, b < p - 1$.

2. Public Key Computation:

- Using the chosen private keys, each party computes its public key:
  - For party A: $A = g^a \mod p$
  - For party B: $B = g^b \mod p$

3. Key Exchange:

- Parties exchange their public keys over the insecure communication channel.

4. Shared Secret Calculation:

- Each party uses its own private key and the received public key to compute the shared secret:
  - For party A: $s = B^a \mod p$
  - For party B: $s = A^b \mod p$
- The shared secret is the same for both parties.

5. Security Considerations:

- **Discrete Logarithm Problem:**
  - The security of Diffie-Hellman relies on the computational difficulty of solving the discrete logarithm problem, where given *g*, *p*, and *A*, it is computationally infeasible to determine *a*.
- **Man-in-the-Middle (MitM) Attack:**

    - Diffie-Hellman is vulnerable to MitM attacks where an attacker intercepts and alters the exchanged public keys. This can be mitigated by incorporating additional authentication mechanisms or digital signatures.
- **Perfect Forward Secrecy (PFS):**
    - Diffie-Hellman provides PFS, meaning that even if the long-term secret keys are compromised, past communications remain secure because each session has its unique shared secret.

6. Key Usage:

- The shared secret can be used as a symmetric key for encrypting subsequent communication using a symmetric encryption algorithm. Symmetric encryption is faster than asymmetric encryption, making it suitable for encrypting large amounts of data.

7. Applications:

- Diffie-Hellman is widely used in various protocols, including TLS (Transport Layer Security) for secure web communication, SSH (Secure Shell) for secure remote access, and IPsec for secure network communication.

8. Parameters:

- The security of Diffie-Hellman depends on the choice of parameters, especially the size of the prime *p*. Larger prime numbers contribute to increased security.

Prac 6

Chinese Remainder Thorem

The Chinese Remainder Theorem (CRT) is a mathematical theorem that provides a way to reconstruct an integer from its remainders when divided by several pairwise coprime (mutually prime) moduli. The theorem is named after its origin in ancient Chinese mathematics. The Chinese Remainder Theorem has various applications in number theory and modular arithmetic and is commonly used in cryptography and computer science.

Statement of the Chinese Remainder Theorem:

Let $n_1, n_2, \ldots, n_k$ be positive integers that are pairwise coprime (i.e., $\gcd(n_i, n_j) = 1$ for all $i \neq j$), and let N be their product ($N = n_1 \cdot n_2 \cdot \ldots \cdot n_k$). For any integers $a_1, a_2, \ldots, a_k$, there exists a unique integer x such that:

$$
\begin{aligned}
x &\equiv a_1 \pmod{n_1} \\
x &\equiv a_2 \pmod{n_2} \\
&\vdots \\
x &\equiv a_k \pmod{n_k}
\end{aligned}
$$

Moreover, this integer x satisfies $0 \leq x < N$.

Solving the System of Congruences:

The Chinese Remainder Theorem provides a systematic way to find the solution x for the given system of congruences. The general approach involves computing a set of constants $c_1, c_2, \ldots, c_k$ and then constructing the solution as follows:

$$x = a_1 \cdot c_1 \cdot M_1 + a_2 \cdot c_2 \cdot M_2 + \ldots + a_k \cdot c_k \cdot M_k$$

where $M_i = \frac{N}{n_i}$ and c_i is the modular inverse of M_i modulo n_i (i.e., $M_i \cdot c_i \equiv 1 \pmod{n_i}$).

Applications:

1. **Modular Arithmetic:**
    - CRT is widely used in modular arithmetic to simplify computations in different modular rings.
2. **Cryptography:**
    - In RSA encryption, CRT is used to speed up the computation of modular exponentiation in the decryption process.
3. **Error Detection and Correction:**
    - CRT is used in certain error-detecting and error-correcting codes.
4. **Parallel Computing:**
    - The CRT is used in parallel algorithms for solving systems of linear congruences.
5. **Computer Algebra Systems:**
    - CRT is a fundamental tool in computer algebra systems for symbolic mathematics.

Prac 7

Digital Signature Algorithm

The Digital Signature Algorithm (DSA) is a widely used asymmetric key algorithm for generating digital signatures. It was proposed by the National Institute of Standards and Technology (NIST) in 1991 and later standardized in FIPS PUB 186. DSA is specifically designed for digital signatures and is part of the Digital Signature Standard (DSS).

Key Components of DSA:

1. **Key Generation:**
   - A pair of keys is generated: a private key (x) and a corresponding public key (y).
   - The private key is randomly chosen in the range $[1, q-1]$, where q is a large prime.
   - The public key is computed as $y = g^x \mod p$, where p is a large prime, and g is a generator of a subgroup of order q in the multiplicative group modulo p.
2. **Signing a Message:**
   - To sign a message M, a per-message secret number k is generated randomly in the range $[1, q-1]$.
   - The digital signature is computed as follows:
     $r \equiv (g^k \mod p) \mod q$
     $s \equiv k^{-1}(H(M) + x \cdot r) \mod q$
   - The signature is (r, s).
3. **Verifying a Signature:**
   - To verify a signature (r, s) on message M, the verifier uses the public key (p, q, g, y) and checks:

$$w \equiv s^{-1} \mod q$$
$$u_1 \equiv H(M) \cdot w \mod q$$
$$u_2 \equiv r \cdot w \mod q$$
$$v \equiv ((g^{u_1} \cdot y^{u_2}) \mod p) \mod q$$

- If v equals r, the signature is valid.

Key Properties and Considerations:

1. **Security:**
    - DSA relies on the difficulty of solving the discrete logarithm problem, specifically in the subgroup of order *q* modulo *p*.
2. **Randomness:**
    - The selection of *k* must be done with sufficient randomness to avoid the exposure of the private key.
3. **Hash Function:**
    - DSA signs a hash of the message ((*H*(*M*)), not the message itself. A secure hash function is crucial for the security of DSA.
4. **Key Lengths:**
    - The security of DSA depends on the choice of key lengths. Common key lengths are 1024, 2048, or 3072 bits.
5. **Applications:**

- DSA is commonly used in digital signatures for ensuring the authenticity and integrity of messages, certificates, and other digital content.

Prac 8

Elliptic Curve Cryptography (ECC)

Elliptic Curve Cryptography (ECC) is a public-key cryptography algorithm that is based on the mathematics of elliptic curves over finite fields. ECC provides strong security with shorter key lengths compared to other public-key algorithms like RSA, making it especially suitable for resource-constrained devices and applications where efficiency is crucial.

ECC Key Generation:

1. **Select an Elliptic Curve:**
   - Choose a specific elliptic curve with its parameters (a, b, p, G, etc.). Common standards define these curves.
2. **Generate a Private Key:**
   - Select a random integer d as the private key, where $1 < d < \text{order of the curve}$.
3. **Compute the Public Key:**
   - Compute the public key Q as $Q = d \cdot G$.

ECC Digital Signature:

1. **Signing:**
   - Generate a random number k (ephemeral key).
   - Compute the point $R = k \cdot G$.
   - Compute $r = x(R) \mod \text{order of the curve}$.
   - Compute $s = k^{-1}(H(m) + dr) \mod \text{order of the curve}$, where $H(m)$ is the hash of the message.
2. **Verifying:**
   - Receive the signature (r, s) and the public key Q.
   - Compute $w = s^{-1} \mod \text{order of the curve}$.
   - Compute $u_1 = H(m) \cdot w \mod \text{order of the curve}$ and $u_2 = r \cdot w \mod \text{order of the curve}$.
   - Compute $X = u_1 \cdot G + u_2 \cdot Q$.
   - If $x(X) \equiv r \mod \text{order of the curve}$, the signature is valid.

ECC Encryption (Elliptic Curve Diffie-Hellman):

1. **Key Exchange:**
   - Two parties generate their private keys d_A and d_B.
   - They compute their respective public keys $Q_A = d_A \cdot G$ and $Q_B = d_B \cdot G$.
   - They exchange public keys and compute the shared secret $S = d_A \cdot Q_B = d_B \cdot Q_A$.
2. **Key Derivation:**
   - The shared secret S is used as a symmetric key for encryption.

Applications:

1. **Digital Signatures:**
   - ECC is widely used for digital signatures in protocols like ECDSA (Elliptic Curve Digital Signature Algorithm).
2. **Key Exchange:**

- ECC-based key exchange is used in protocols like ECDH (Elliptic Curve Diffie-Hellman) for secure key establishment.

3. **Encryption:**
    - ECC-based encryption schemes, like ECIES (Elliptic Curve Integrated Encryption Scheme), are used for secure communication.
4. **IoT and Resource-Constrained Devices:**
    - ECC's efficiency makes it suitable for secure communication in resource-constrained environments, such as Internet of Things (IoT) devices.

Key Concepts:

1. **Elliptic Curve Equation:**
    - An elliptic curve is defined by an equation of the form $y^2 = x^3 + ax + b$, where a and b are constants. The curve is symmetric with respect to the x-axis.
2. **Group Structure:**
    - Points on an elliptic curve form a mathematical group. Operations like addition and doubling are defined on these points.
3. **Base Point (Generator Point):**
    - A specific point G on the elliptic curve is chosen as the base point. Multiplying this base point by an integer produces a set of points known as the elliptic curve group.
4. **Scalar Multiplication:**
    - Scalar multiplication involves multiplying a point on the curve by an integer. For example, $Q = k \cdot G$, where k is an integer and G is the base point.
5. **Elliptic Curve Discrete Logarithm Problem (ECDLP):**
    - The security of ECC relies on the difficulty of solving the ECDLP, which is finding k given Q and G in the equation $Q = k \cdot G$.