CNS Important Topics

1.Cryptographic attacks.

Cryptographic Attacks

Cryptographic attacks are methods used to undermine the security of cryptographic systems. Here are key types of attacks:

1. **Brute Force Attack**:
   - **Description**: Attempts every possible key until the correct one is found.
   - **Countermeasure**: Use longer key lengths.
2. **Cryptanalysis**:
   - **Description**: Analyzes algorithms to find vulnerabilities and decrypt data without the key.
   - **Example**: Techniques like linear and differential cryptanalysis.
3. **Man-in-the-Middle (MitM) Attack**:
   - **Description**: An attacker intercepts communication between two parties.
   - **Countermeasure**: Implement strong encryption and authentication (e.g., SSL/TLS).
4. **Replay Attack**:
   - **Description**: An attacker captures and reuses valid data transmissions.
   - **Countermeasure**: Use nonces or timestamps to ensure message uniqueness.
5. **Side-Channel Attack**:
   - **Description**: Exploits physical characteristics (e.g., timing, power usage) of a cryptographic system.
   - **Countermeasure**: Use constant-time algorithms and other physical protections.
6. **Chosen Plaintext Attack**:
   - **Description**: The attacker can choose plaintexts to be encrypted and obtain the corresponding ciphertexts.
   - **Countermeasure**: Use strong encryption schemes resistant to such attacks.
7. **Chosen Ciphertext Attack**:
   - **Description**: The attacker can choose ciphertexts to decrypt and learn information about the key or plaintext.
   - **Countermeasure**: Implement secure encryption methods.
8. **Birthday Attack**:
   - **Description**: Exploits the probability of hash collisions in hash functions.
   - **Countermeasure**: Use hash functions with larger output sizes (e.g., SHA-256).

2.Security Services & Mechanisms.

AN). **Security Services & Mechanisms (5 Marks)**

Security services and mechanisms are key to protecting data and ensuring secure communication in information systems. Here's a concise overview:

Security Services:

1. **Confidentiality**:
   - Ensures that sensitive data is only accessible to authorized users.
   - **Mechanism**: Encryption (e.g., AES, RSA).
2. **Integrity**:
   - Guarantees data has not been altered or tampered with.
   - **Mechanism**: Hash functions (e.g., SHA-256).
3. **Authentication**:
   - Verifies the identity of users before granting access.
   - **Mechanism**: Passwords, biometrics, multi-factor authentication.
4. **Non-repudiation**:
   - Ensures that neither the sender nor the recipient can deny the action.
   - **Mechanism**: Digital signatures, transaction logs.
5. **Access Control**:
   - Regulates who can access resources and what actions they can perform.
   - **Mechanism**: Role-based access control (RBAC), access control lists (ACLs).

Security Mechanisms:

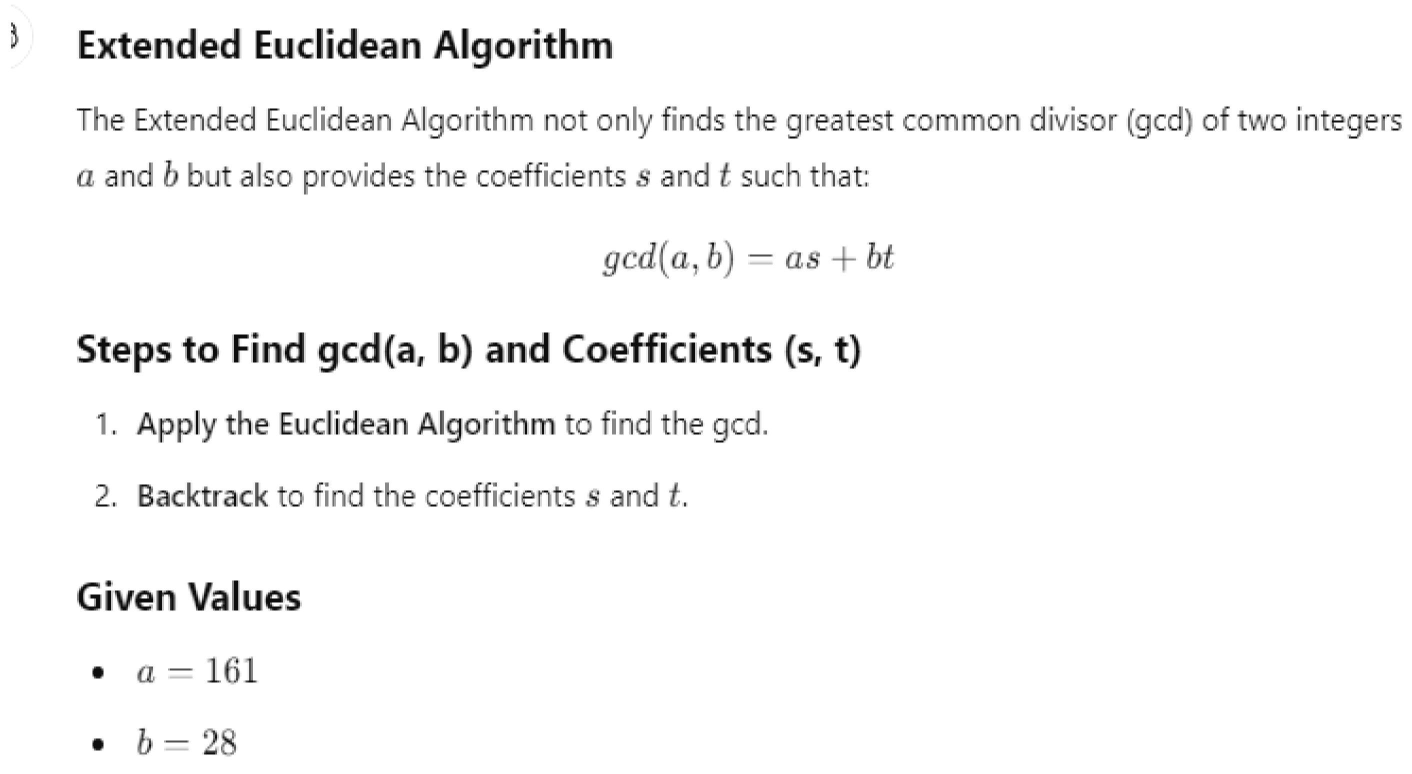
1. **Encryption**:
   - Converts data into unreadable formats to prevent unauthorized access.
2. **Firewalls**:
   - Monitors and controls network traffic based on security rules.
3. **IDS (Intrusion Detection System)**:
   - Detects suspicious activities in a network and alerts administrators.
4. **VPN (Virtual Private Network)**:
   - Creates secure encrypted connections over the internet.
5. **Antivirus Software**:
   - Detects and removes malware from devices.

Diagram of Security Services & Mechanisms

```
+---------------------+
|    Security Services|
+-------------------
```

3.Explain Extended Euclidean Algorithm. Given a=161 and b=28, find gcd(a,b) and the values of (s) and (t) using the Extended Euclidean Algorithm.

AN).



Extended Euclidean Algorithm

The Extended Euclidean Algorithm not only finds the greatest common divisor (gcd) of two integers a and b but also provides the coefficients s and t such that:

$$gcd(a,b) = as + bt$$

Steps to Find gcd(a, b) and Coefficients (s, t)

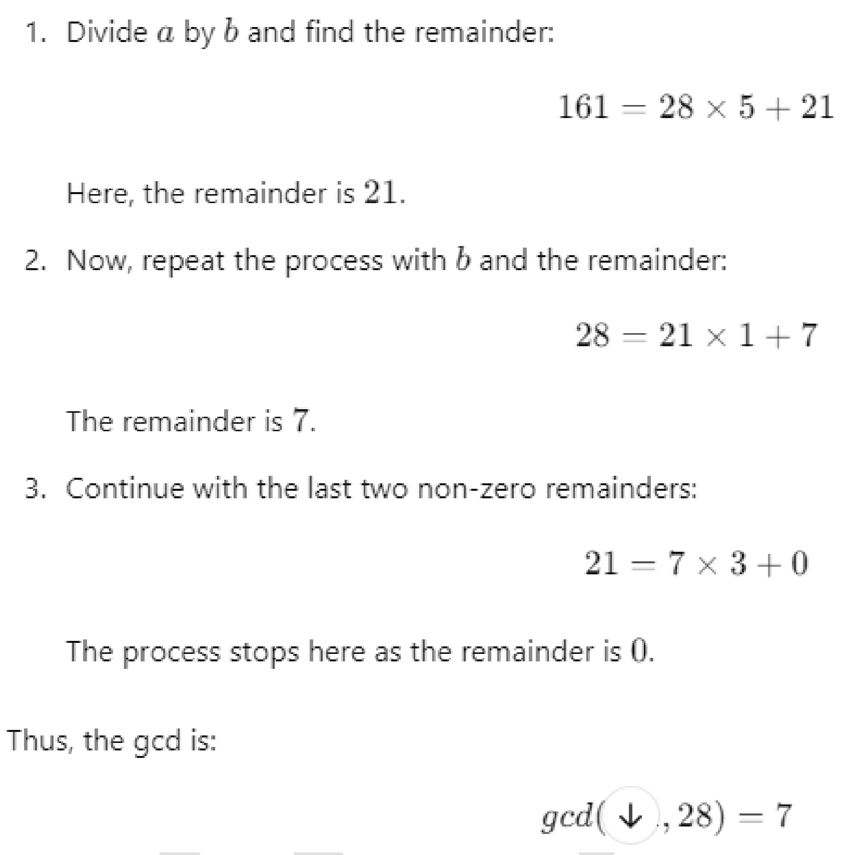
1. **Apply the Euclidean Algorithm** to find the gcd.
2. **Backtrack** to find the coefficients s and t.

Given Values

- $a = 161$
- $b = 28$

Example: Find gcd(161, 28) and Coefficients (s, t)

Step 1: Calculate gcd Using the Euclidean Algorithm



1. Divide a by b and find the remainder:

   $$161 = 28 \times 5 + 21$$

   Here, the remainder is 21.

2. Now, repeat the process with b and the remainder:

   $$28 = 21 \times 1 + 7$$

   The remainder is 7.

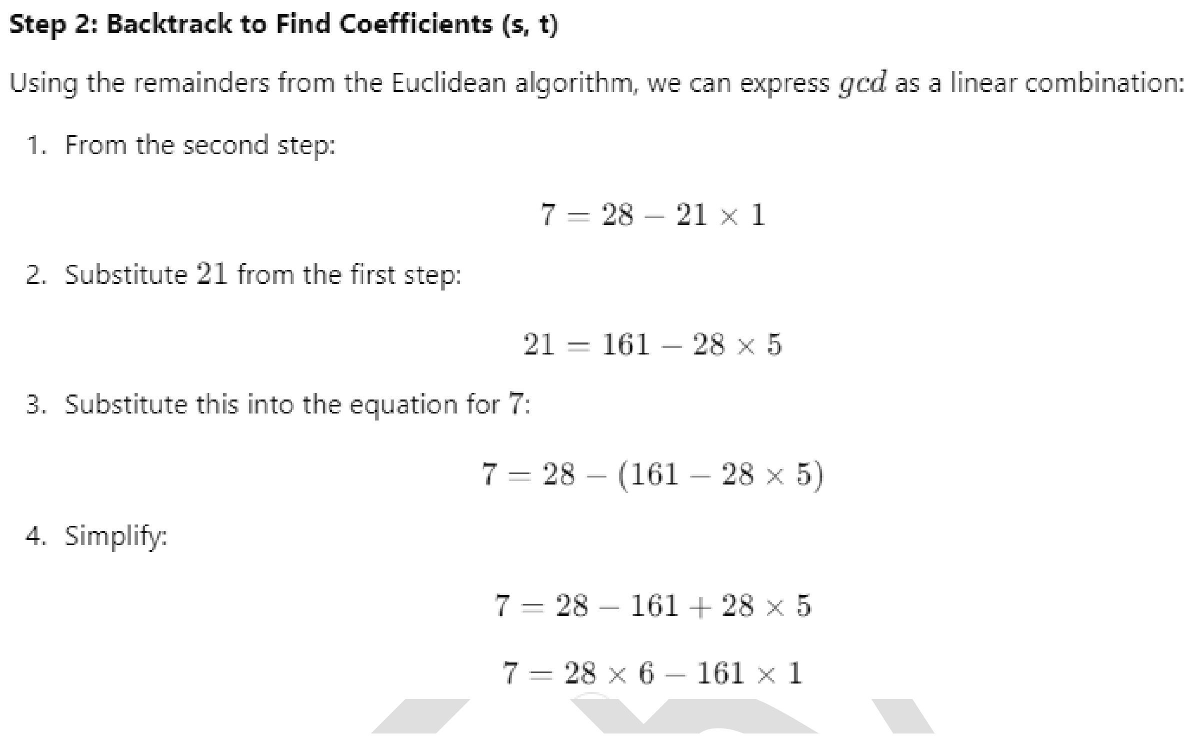
3. Continue with the last two non-zero remainders:

   $$21 = 7 \times 3 + 0$$

   The process stops here as the remainder is 0.

Thus, the gcd is:

$$gcd(\text{[illegible]}, 28) = 7$$



Step 2: Backtrack to Find Coefficients (s, t)

Using the remainders from the Euclidean algorithm, we can express gcd as a linear combination:

1. From the second step:

$$7 = 28 - 21 \times 1$$

2. Substitute 21 from the first step:

$$21 = 161 - 28 \times 5$$

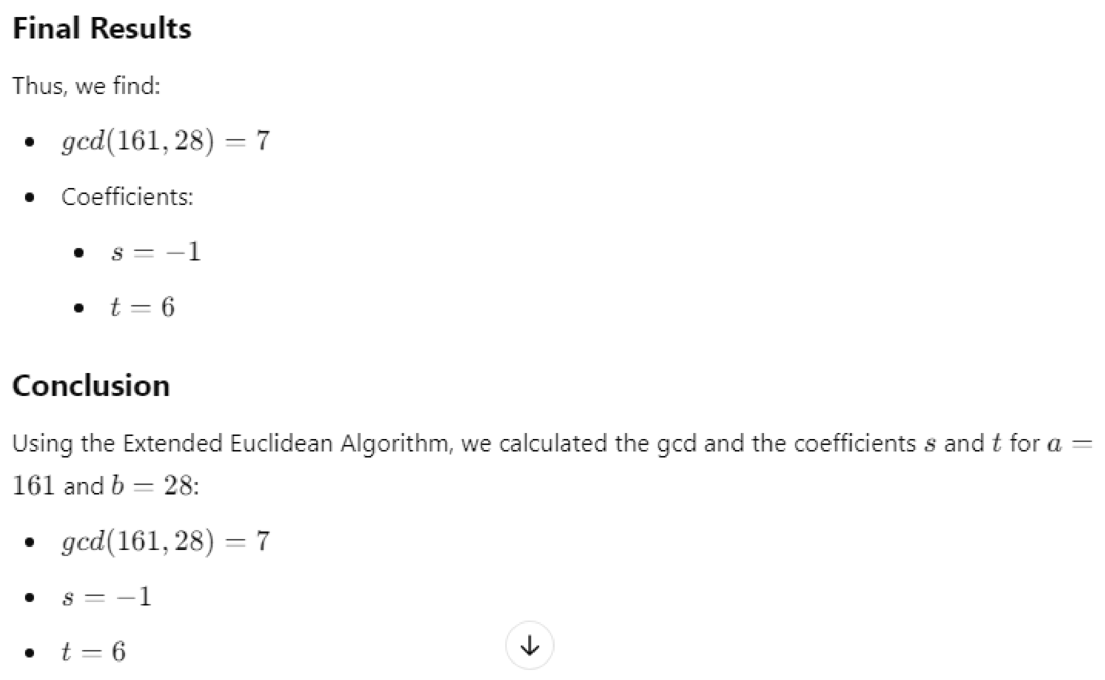
3. Substitute this into the equation for 7:

$$7 = 28 - (161 - 28 \times 5)$$

4. Simplify:

$$7 = 28 - 161 + 28 \times 5$$

$$7 = 28 \times 6 - 161 \times 1$$



Final Results

Thus, we find:

- $gcd(161, 28) = 7$
- Coefficients:
  - $s = -1$
  - $t = 6$

Conclusion

Using the Extended Euclidean Algorithm, we calculated the gcd and the coefficients s and t for $a = 161$ and $b = 28$:

- $gcd(161, 28) = 7$
- $s = -1$
- $t = 6$

4.Algebraic structures properties.

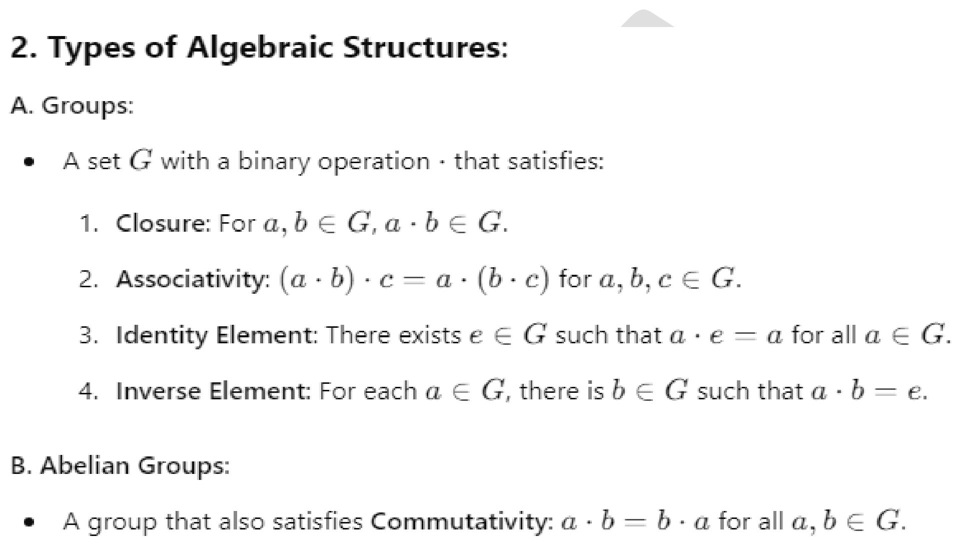
AN). **Algebraic Structures and Their Properties**

Algebraic structures are mathematical entities consisting of a set equipped with one or more operations that satisfy specific properties. Here are the fundamental types of algebraic structures and their key properties:

---

1. Sets and Operations

A set is a collection of distinct objects, and an operation is a rule that combines elements of the set.



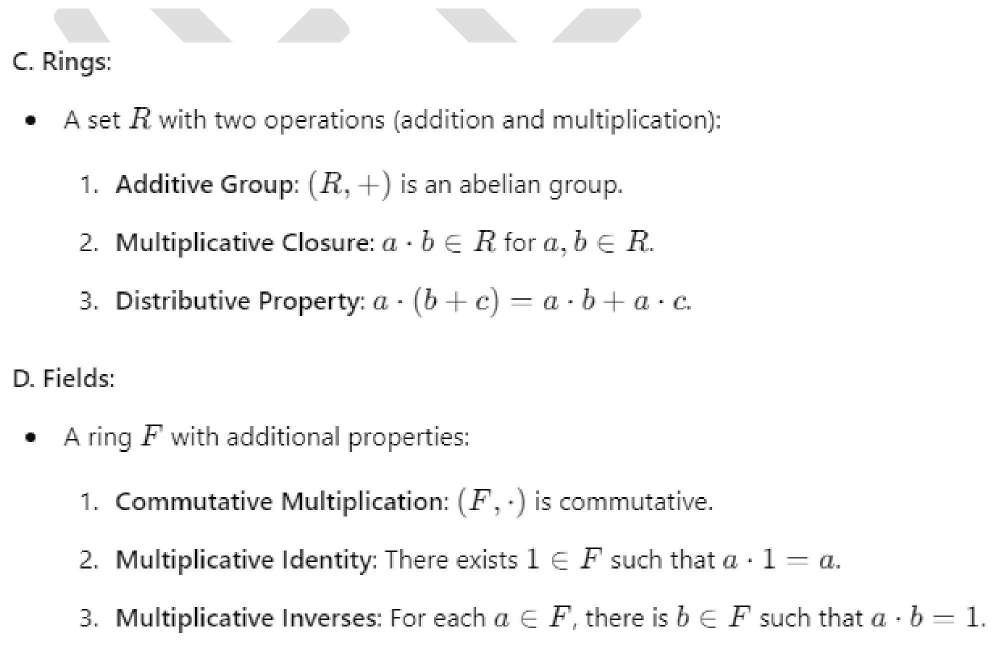
2. Types of Algebraic Structures:

A. Groups:

- A set G with a binary operation $\cdot$ that satisfies:
  1. **Closure**: For $a, b \in G$, $a \cdot b \in G$.
  2. **Associativity**: $(a \cdot b) \cdot c = a \cdot (b \cdot c)$ for $a, b, c \in G$.
  3. **Identity Element**: There exists $e \in G$ such that $a \cdot e = a$ for all $a \in G$.
  4. **Inverse Element**: For each $a \in G$, there is $b \in G$ such that $a \cdot b = e$.

B. Abelian Groups:

- A group that also satisfies **Commutativity**: $a \cdot b = b \cdot a$ for all $a, b \in G$.



C. Rings:

- A set R with two operations (addition and multiplication):
  1. **Additive Group**: $(R, +)$ is an abelian group.
  2. **Multiplicative Closure**: $a \cdot b \in R$ for $a, b \in R$.
  3. **Distributive Property**: $a \cdot (b + c) = a \cdot b + a \cdot c$.

D. Fields:

- A ring F with additional properties:
  1. **Commutative Multiplication**: $(F, \cdot)$ is commutative.
  2. **Multiplicative Identity**: There exists $1 \in F$ such that $a \cdot 1 = a$.
  3. **Multiplicative Inverses**: For each $a \in F$, there is $b \in F$ such that $a \cdot b = 1$.

Structure	**Properties**

Structure	Properties
Group	Closure, Associativity, Identity, Inverses
Abelian Group	All group properties + Commutativity
Ring	Additive Group, Multiplicative Closure, Distribution
Field	All ring properties + Commutative Multiplication, Identity, Inverses

5.DES Permutations and Round Function.

AN). **DES Permutations and Round Function (Short Version)**

1. Key Permutations in DES:

- **Initial Permutation (IP)**: Rearranges the 64-bit input before encryption.
- **Inverse Initial Permutation (IP−1^{-1}−1)**: Reverses the IP at the end of encryption.
- **Permutation (P)**: Reorders 32-bit output from the S-boxes within each round to spread bit influence.

---

2. DES Round Function (Steps):

1. **Expansion (E)**: Expands the 32-bit right half to 48 bits.
2. **XOR with Round Key**: The expanded right half is XORed with the 48-bit round key.
3. **Substitution (S-boxes)**: The 48-bit result is passed through 8 S-boxes, reducing it to 32 bits.
4. **Permutation (P)**: The 32-bit output is permuted to mix bits.
5. **XOR with Left Half (L)**: The permuted output is XORed with the left half of the input.
6. **Swapping**: The left and right halves are swapped for the next round.

Diagram (Simplified)

```
L(i-1)                 R(i-1)
  |                       |
  |--> XOR with Key ---> Expansion (E)
  |                       |
  |       Substitution (S-boxes)
  |       Permutation (P)
  |<-- XOR with L(i-1) ----|
```

Conclusion: DES uses multiple permutations, substitutions, and XOR operations within each round to ensure strong encryption by mixing and diffusing bits effectively.

6.AES Structure

AN). AES Structure (Advanced Encryption Standard)

AES is a symmetric key encryption standard used for secure data encryption. It operates on 128-bit blocks of data and supports key sizes of 128, 192, or 256 bits. AES consists of multiple rounds of transformations, depending on the key size.

Key Features of AES Structure

- **Block Size**: 128 bits (fixed)
- **Key Sizes**: 128 bits (10 rounds), 192 bits (12 rounds), 256 bits (14 rounds)
- **Number of Rounds**: Depends on the key size:
  - 10 rounds for 128-bit key
  - 12 rounds for 192-bit key
  - 14 rounds for 256-bit key

AES Transformation Steps

Each AES round (except the final one) consists of four main transformations:

1. SubBytes (Substitution Layer)

- Each byte in the block is replaced using a substitution table (S-box).
- Provides non-linearity to AES and improves security.

2. ShiftRows (Row Shifting)

- Each row in the 4x4 state matrix is shifted left by a certain number of positions.
- Row 0: No shift
- Row 1: Shift left by 1 byte
- Row 2: Shift left by 2 bytes
- Row 3: Shift left by 3 bytes
- This step introduces diffusion by rearranging the bytes.

3. MixColumns (Column Mixing)

- Each column in the state matrix is transformed by multiplying it with a fixed polynomial in GF(2^8).

- This step ensures that each byte influences the other bytes in its column, spreading out the data across columns.

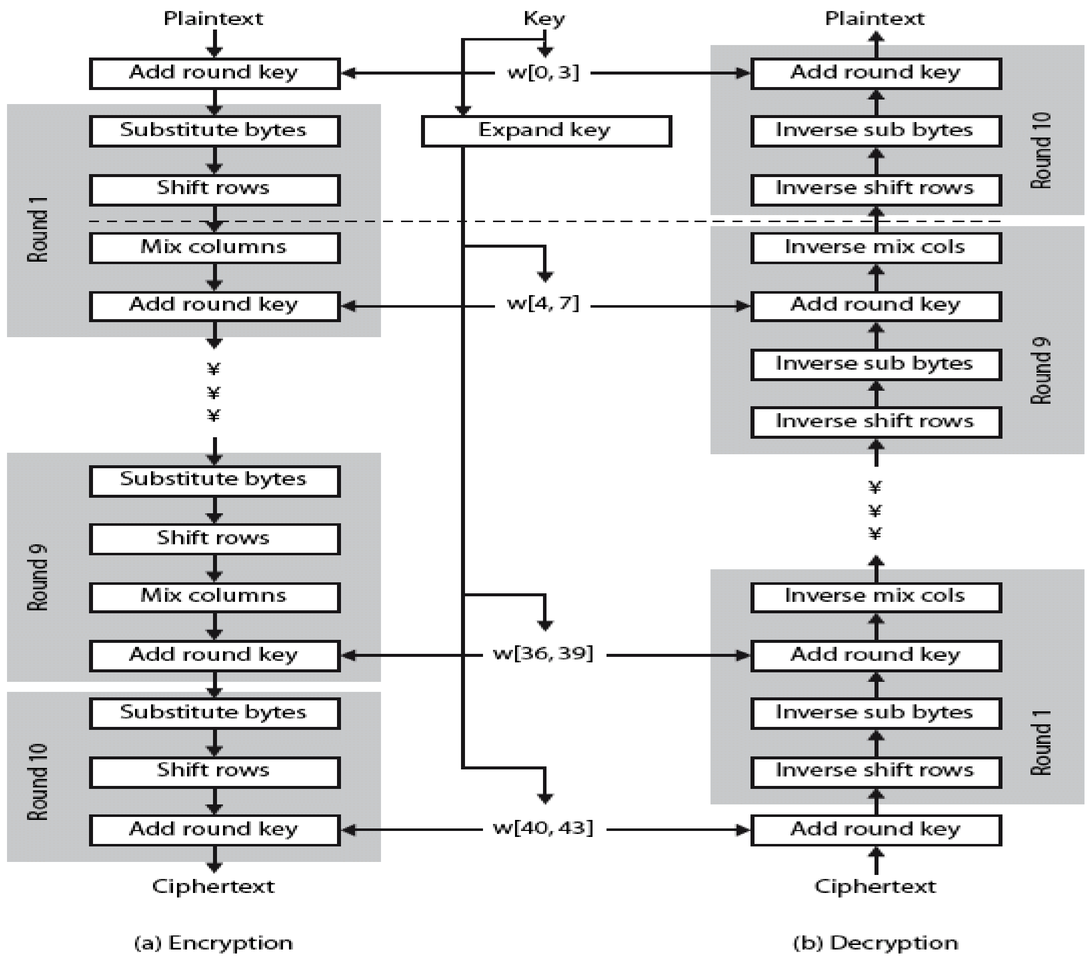
4. AddRoundKey (Key Addition)

- The current state is XORed with a round key derived from the original key using the **Key Expansion** process.
- This is the only step where the encryption key is introduced into the round.

Initial and Final Round

- **Initial Round**: The input block is XORed with the first round key (before the first regular round).
- **Final Round**: The final round excludes the **MixColumns** step and consists of only **SubBytes**, **ShiftRows**, and **AddRoundKey**.

AES Round Structure (Diagram)



7.Euler's theorem and Fermat's theorem with examples.

AN)

Euler's Theorem and Fermat's Little Theorem with Clear Examples

These theorems are fundamental in number theory and widely used in cryptography, particularly in systems like RSA.

---

1. Fermat's Little Theorem

Statement:

If p is a prime number and a is an integer such that a is not divisible by p, then: $a^{p-1} \equiv 1 \pmod{p}$

In simple terms, if p is a prime and a is any number less than p and not divisible by p, raising a to the power $p - 1$ will leave a remainder of 1 when divided by p.

Example:

Let's use $a = 3$ and $p = 7$ (since 7 is prime): $^{1} = 3^6$

Now, calculate 3^6: $3^6 = 729$

Find $729 \pmod{7}$ (i.e., the remainder when 729 is divided by 7): $729 \div 7 = 104 \text{ remainder } 1$

Thus, $729 \equiv 1 \pmod{7}$, meaning: $3^6 \equiv 1 \pmod{7}$

This confirms Fermat's theorem.

---

2. Euler's Theorem

Statement:

For any integer n, if a and n are **coprime** (i.e., $\gcd(a, n) = 1$), then: $a^{\phi(n)} \equiv 1 \pmod{n}$ where $\phi(n)$ is **Euler's totient function**, which counts the number of integers less than n that are coprime with n.

Example:

Let's take $a = 7$ and $n = 10$. First, we calculate $\phi(10)$:

- $\phi(10) = 4$, because the integers less than 10 that are coprime with 10 are 1, 3, 7, and 9.

Now apply Euler's Theorem: $7^{\phi(10)} = 7^4$

Calculate 7^4: $7^4 = 2401$

Now find $2401 \pmod{10}$ (the remainder when 2401 is divided by 10): $2401 \div 10 = 240 \text{ remainder } 1$

Thus, $7^4 \equiv 1 \pmod{10}$.

This confirms Euler's theorem.

3. Relationship Between Euler's Theorem and Fermat's Little Theorem

- Fermat's Little Theorem is a **special case** of Euler's Theorem.
- If p is prime, then $\phi(p) = p - 1$. Therefore, Fermat's theorem becomes a specific version of Euler's theorem where n is a prime number.

Summary of Key Points

- **Fermat's Little Theorem**: For prime p and a not divisible by p, $a^{p-1} \equiv 1 \pmod{p}$.
- **Euler's Theorem**: For any a coprime to n, $a^{\phi(n)} \equiv 1 \pmod{n}$, where $\phi(n)$ is Euler's totient function.

Real-World Application: Cryptography

These theorems are used in **modular exponentiation** to make encryption (like RSA) efficient. Fermat's theorem helps simplify calculations when using prime moduli, while Euler's theorem allows for a more general application with non-prime moduli.

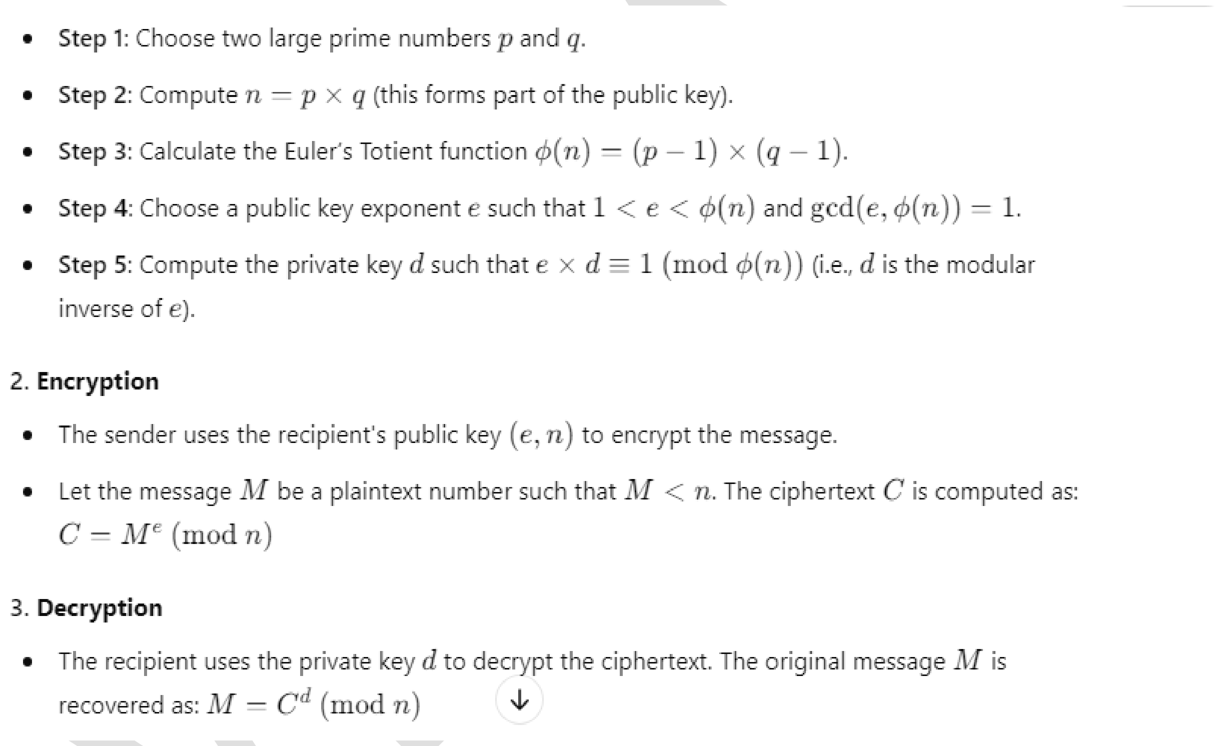
8.RSA Algorithm with a neat diagram.

AN). **RSA Algorithm**

The **RSA Algorithm** is a widely-used public-key cryptosystem for secure data transmission. It is based on the mathematical difficulty of factoring large prime numbers.

Steps in RSA Algorithm

Key Generation:



- **Step 1**: Choose two large prime numbers p and q.
- **Step 2**: Compute $n = p \times q$ (this forms part of the public key).
- **Step 3**: Calculate the Euler's Totient function $\phi(n) = (p-1) \times (q-1)$.
- **Step 4**: Choose a public key exponent e such that $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$.
- **Step 5**: Compute the private key d such that $e \times d \equiv 1 \pmod{\phi(n)}$ (i.e., d is the modular inverse of e).

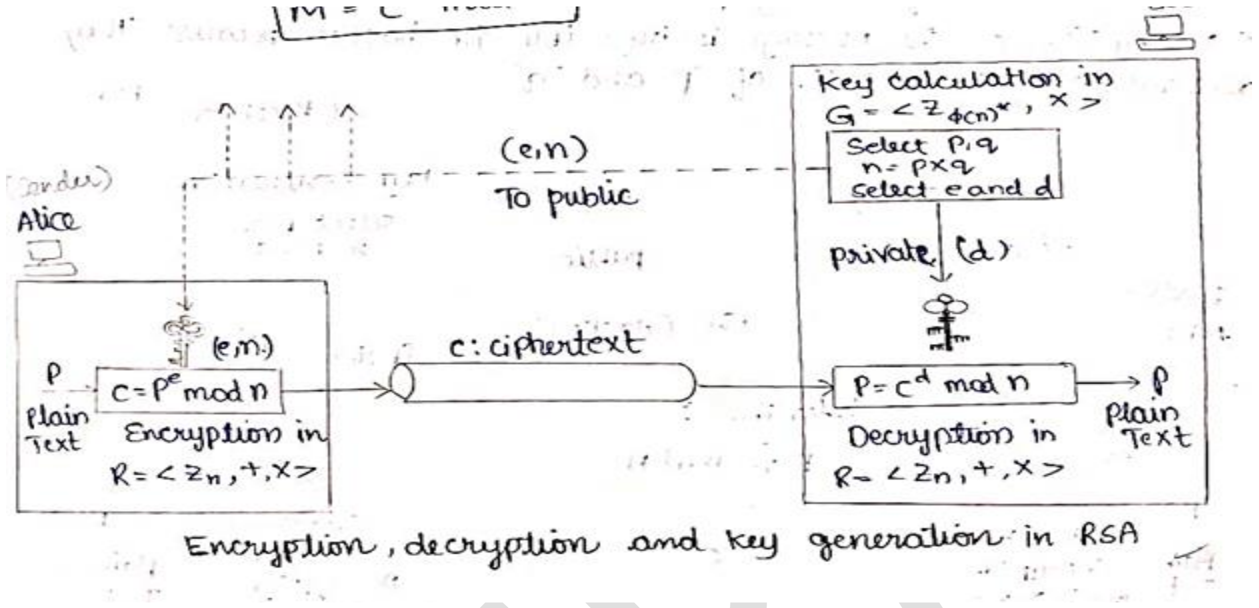
2. **Encryption**

- The sender uses the recipient's public key (e, n) to encrypt the message.
- Let the message M be a plaintext number such that $M < n$. The ciphertext C is computed as: $C = M^e \pmod{n}$

3. **Decryption**

- The recipient uses the private key d to decrypt the ciphertext. The original message M is recovered as: $M = C^d \pmod{n}$

Diagram of RSA Algorithm



(sender)
Alice
(e,n)
To public
Key calculation in $G = < Z_{\phi(n)}^*, \times >$
Select p, q
n = p×q
select e and d
private (d)
(e,n)
P
Plain Text
$C = P^e \bmod n$
Encryption in $R = < Z_n, +, \times >$
c : ciphertext
$P = C^d \bmod n$
P
Plain Text
Decryption in $R = < Z_n, +, \times >$
Encryption, decryption and key generation in RSA

Example of RSA Algorithm

1. Key Generation

- Choose two prime numbers $p = 3$ and $q = 11$.
- Compute $n = p \times q = 3 \times 11 = 33$.
- Compute $\phi(n) = (p-1) \times (q-1) = 2 \times 10 = 20$.
- Choose $e = 3$ (since $\gcd(3, 20) = 1$).
- Compute d such that $e \times d \equiv 1 \pmod{20}$. The value of $d = 7$ because $3 \times 7 = 21 \equiv 1 \pmod{20}$.

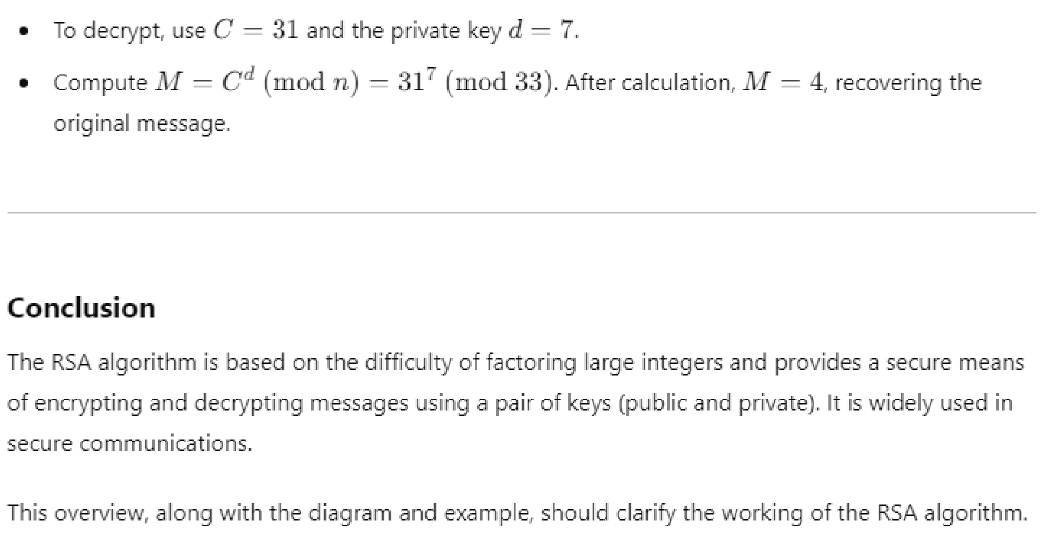
Public key = $(e = 3, n = 33)$
Private key = $(d = 7)$

2. **Encryption**

- Let the message $M = 4$.
- Compute the ciphertext $C = M^e \pmod{n} = 4^3 \pmod{33} = 64 \pmod{33} = 31$.

3. **Decryption**

- To decrypt, use $C = 31$ and the private key $d = 7$.



- To decrypt, use $C = 31$ and the private key $d = 7$.
- Compute $M = C^d \pmod{n} = 31^7 \pmod{33}$. After calculation, $M = 4$, recovering the original message.

Conclusion

The RSA algorithm is based on the difficulty of factoring large integers and provides a secure means of encrypting and decrypting messages using a pair of keys (public and private). It is widely used in secure communications.

This overview, along with the diagram and example, should clarify the working of the RSA algorithm.

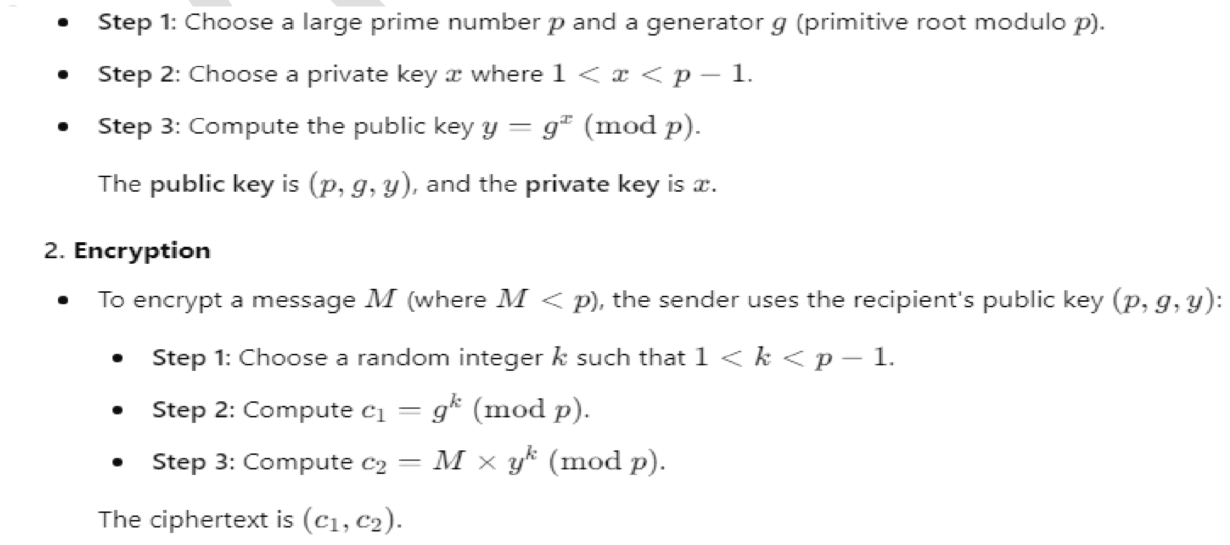
9.ElGamal Cryptosystem.

AN). ElGamal Cryptosystem

The **ElGamal Cryptosystem** is a public-key encryption algorithm based on the Diffie-Hellman key exchange. It provides both encryption and digital signatures and is widely used in cryptographic applications.

Steps in the ElGamal Cryptosystem

1. Key Generation



- **Step 1:** Choose a large prime number p and a generator g (primitive root modulo p).
- **Step 2:** Choose a private key x where $1 < x < p - 1$.
- **Step 3:** Compute the public key $y = g^x \pmod{p}$.

  The **public key** is (p, g, y), and the **private key** is x.

2. **Encryption**

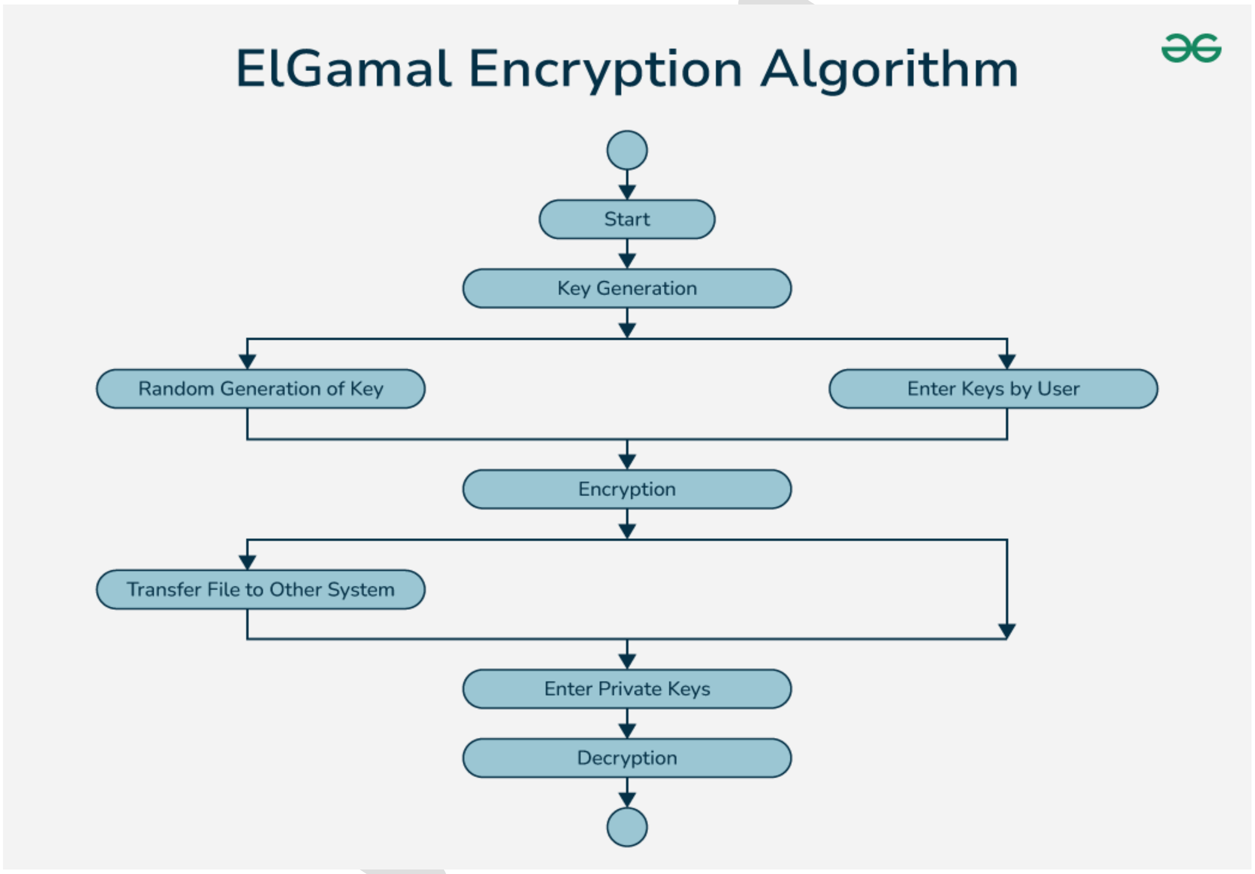
- To encrypt a message M (where $M < p$), the sender uses the recipient's public key (p, g, y):
  - **Step 1:** Choose a random integer k such that $1 < k < p - 1$.
  - **Step 2:** Compute $c_1 = g^k \pmod{p}$.
  - **Step 3:** Compute $c_2 = M \times y^k \pmod{p}$.

  The ciphertext is (c_1, c_2).

3. **Decryption**

- The recipient uses their private key x to decrypt the ciphertext (c_1, c_2):
    - **Step 1**: Compute $s = c_1^x \pmod p$.
    - **Step 2**: Compute the original message $M = c_2 \times s^{-1} \pmod p$, where s^{-1} is the modular inverse of s modulo p.

Diagram of ElGamal Cryptosystem



Key Points

- The ElGamal cryptosystem is based on the difficulty of solving the **Discrete Logarithm Problem**.
- It is used in cryptographic protocols, including digital signatures and key exchanges.
- The encryption and decryption process relies on the modular arithmetic of large prime numbers and generators.

This example simplifies the cryptosystem to show the key concepts without large numbers that are typically used in real-world implementations.

10.Applications of cryptographic hash functions.

AN). **Applications of Cryptographic Hash Functions (Short Version)**

1. **Data Integrity Verification**: Detects alterations in data by comparing hashes.
2. **Digital Signatures**: Ensures message authenticity by signing the hash with a private key.
3. **Password Storage**: Stores hashed passwords instead of plaintext for enhanced security.
4. **Blockchain and Cryptocurrencies**: Secures data blocks, making tampering evident.
5. **Message Authentication Codes (MACs)**: Verifies integrity and authenticity of messages.
6. **Certificate Authorities**: Ensures the integrity of digital certificates.
7. **File Integrity Checking**: Creates checksums for verifying file integrity.
8. **Random Number Generation**: Enhances randomness in secure random number generators.
9. **Deduplication**: Identifies duplicate files using unique hashes.
10. **Secure Multi-Party Computation**: Facilitates secure computations among multiple parties.

These applications highlight the critical role of cryptographic hash functions in ensuring security and integrity in various systems.