**NAMEn : ADITYA JETHANI**

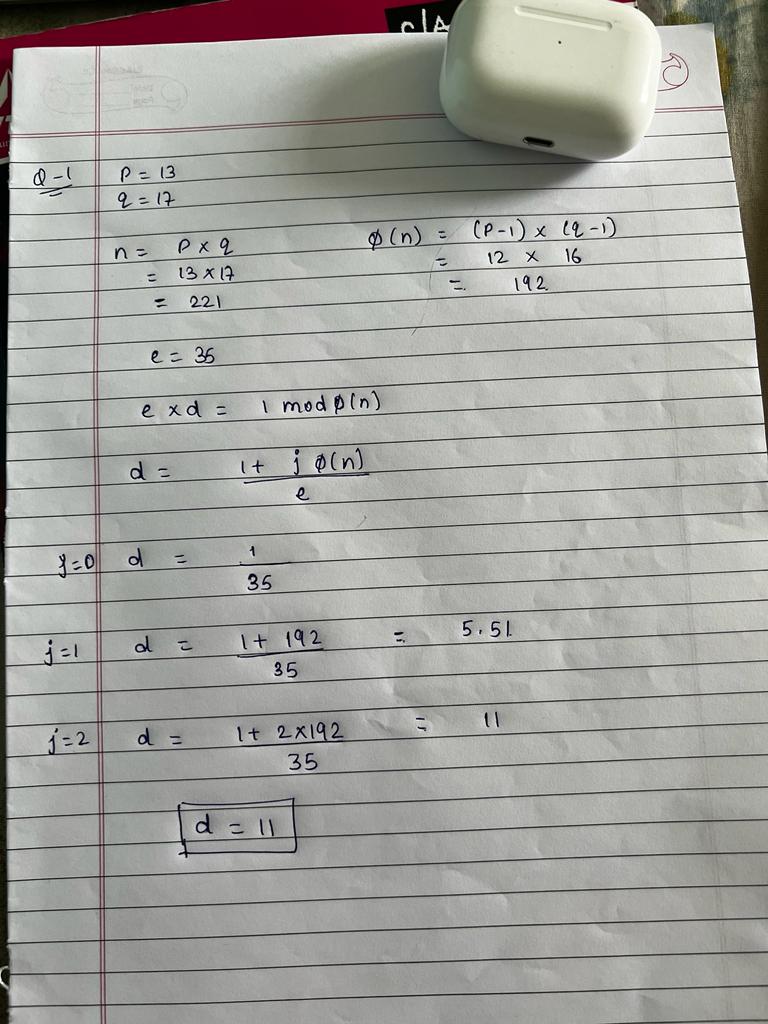
**ROLL NO: 21BCP166**

**DIV-3 G-5**

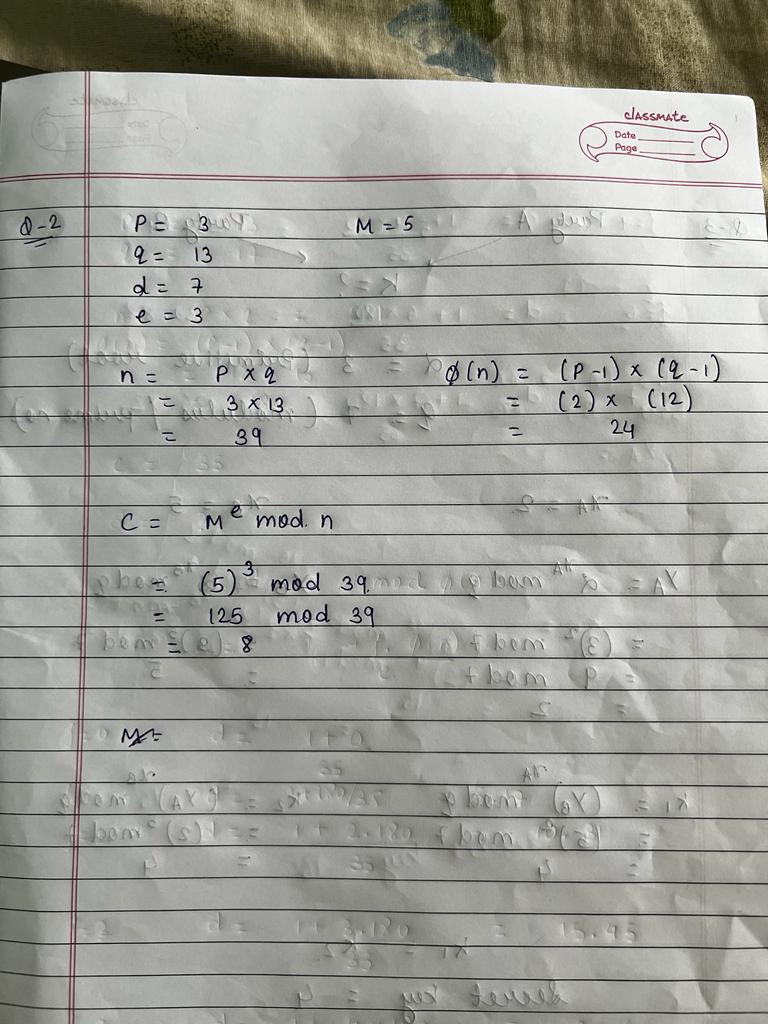
**Information Security Assignment**

**Q-1) In an RSA cryptosystem, a particular A uses two prime numbers p = 13 and q =17 to generate her public and private keys. If the public key of A is 35. Then the private key of A is?**

**Ans :**

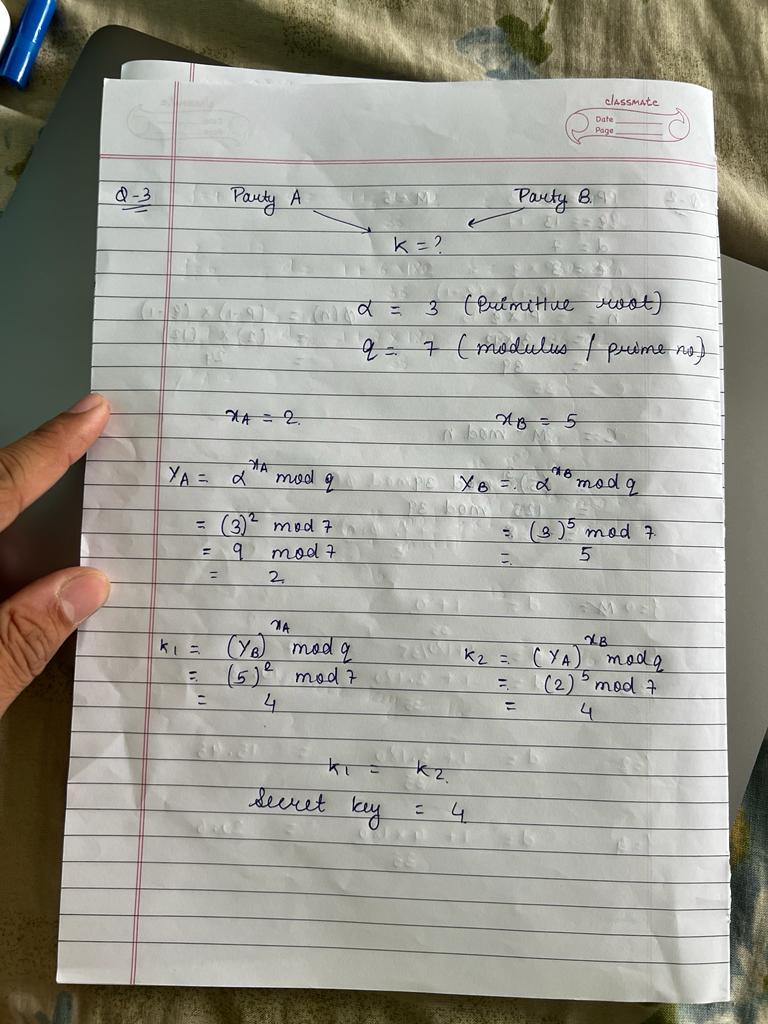
****

**Q-2) Using p=3, q=13, d=7 and e=3 in the RSA algorithm, what is the value of cipher text for a plain text 5?**

**Ans :   
**

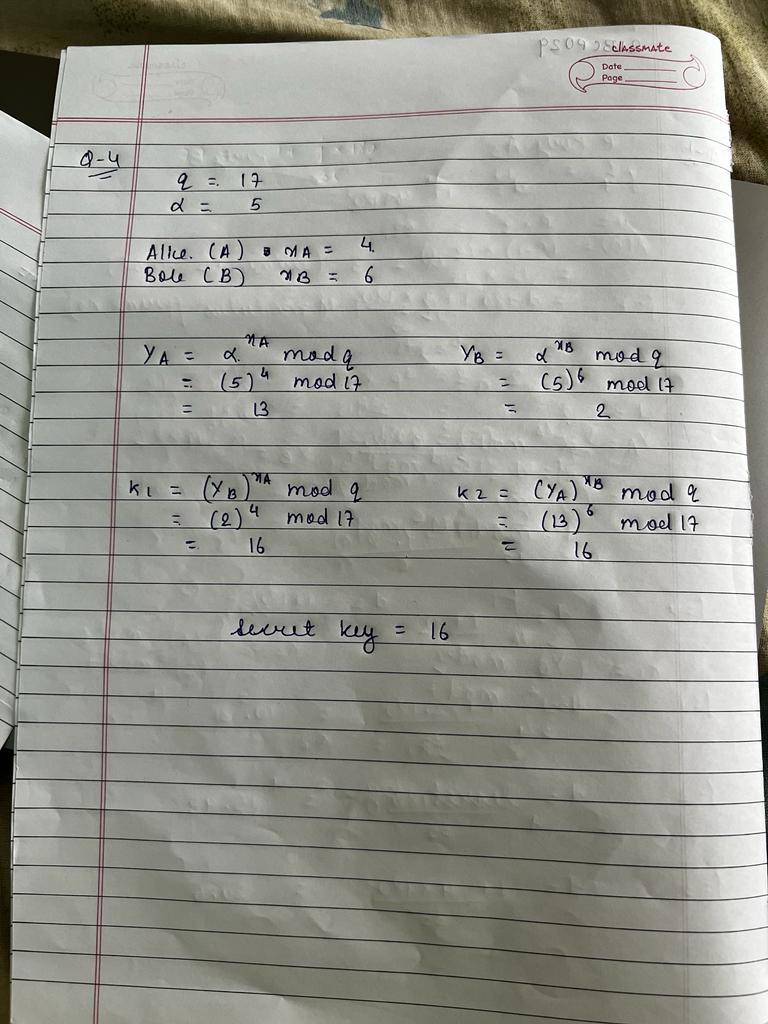
**Q-3) Suppose that two parties A and B wish to set up a common secret key (D-H key) between themselves using the Diffie Hellman key exchange technique. They agree on 7 as the modulus and 3 as the primitive root. Party A chooses 2 and party B chooses 5 as their respective secrets. What is their shared D-H key?**

**Ans :**

****

**Q-4) In a Diffie-Hellman Key Exchange, Alice and Bob have chosen prime value q = 17 and primitive root = 5. If Alice’s secret key is 4 and Bob’s secret key is 6, what is the secret key they exchanged?**

**Ans :**

****

**Q-5) What is a trapdoor one-way function?**

**Ans :**

A trapdoor one-way function is a cryptographic function that is easy to compute in one direction but extremely difficult to reverse, except for individuals who possess a secret "trap door" or key that allows for efficient reversal. This property is crucial for secure encryption, digital signatures, and other cryptographic operations, as it ensures that authorized parties can easily perform reverse computations while keeping unauthorized parties from doing so, thus safeguarding the security of sensitive information and digital transactions.

**Q-6) Explain knapsack cryptosystem.**

**Ans :**

The knapsack cryptosystem is an early form of public-key cryptography. It involves creating a private key consisting of a sequence of superincreasing positive integers and a corresponding public key. Messages are encrypted using the public key, and decryption is performed using the private key. The security relies on the difficulty of solving the knapsack problem, but early knapsack cryptosystems were found to be vulnerable to attacks. Modern cryptography has largely replaced it with more secure and efficient systems like RSA and ECC.

**Q-7) Name 7 categories of attacks on RSA. Explain any five in detail.**

**Ans :**

There are several categories of attacks on the RSA (Rivest-Shamir-Adleman) encryption algorithm, which is widely used in public-key cryptography. Here are seven categories of attacks on RSA, along with explanations of five of them in detail:

1. Brute Force Attack:
2. Factorization Attacks:
3. Timing Attacks:
4. Chosen-Ciphertext Attacks (CCA):
5. Padding Oracle Attacks:
6. Wiener's Attack:
7. Low-Exponent Attacks:

Explanation of five attacks in detail:

1. Brute Force Attack: This is the simplest but least practical attack. It involves trying all possible private key combinations until the correct one is found. With sufficiently long key lengths, it would take an infeasible amount of time to complete.
2. Factorization Attacks: These attacks target the weakness of RSA when the prime factors of the modulus (N) can be efficiently found. The General Number Field Sieve (GNFS) is the most effective method for factoring large N values, but it's computationally intensive and time-consuming for sufficiently large key sizes.
3. Timing Attacks: These attacks take advantage of small variations in execution times to infer information about the private key. By carefully measuring these variations, an attacker might extract bits of the private key over time.
4. Chosen-Ciphertext Attacks (CCA): In CCA attacks, the attacker actively selects ciphertexts for decryption and observes the corresponding plaintexts. This can reveal critical information about the private key, enabling further attacks or decryption of other ciphertexts.
5. Padding Oracle Attacks: These attacks exploit vulnerabilities in the padding schemes used with RSA encryption. By submitting specially crafted ciphertexts and analyzing the server's responses (e.g., error messages or timing variations), an attacker can deduce information about the private key and potentially decrypt other ciphertexts.

**Q-8) Discuss the security issues in**

1. **cipher feedback mode**

**Ans :**

1. Error Propagation: If an error occurs in one block during encryption or decryption, the error propagates through the following blocks. This can result in the incorrect decryption of multiple blocks, potentially causing data corruption.
2. Malleability: CFB mode is malleable, meaning an attacker can manipulate the ciphertext to change the corresponding plaintext. This is a concern if data integrity is a primary requirement.
3. Block Dependency: CFB mode processes the plaintext in blocks that depend on the previous ciphertext block, making it unsuitable for parallel processing. This sequential nature can lead to performance issues in some scenarios.
4. Initialization Vector (IV) Sensitivity: CFB mode requires a unique IV for each encryption session, and the IV should be kept secret. Reusing the IV or revealing it can lead to security vulnerabilities.
5. **output feedback mode**

**Ans :**

1. Lack of Malleability Protection: Similar to CFB, OFB does not provide inherent protection against malleability attacks. Attackers can manipulate the ciphertext to modify the corresponding plaintext.
2. Ciphertext Reuse: If the same IV is used with the same key for multiple encryptions, the same keystream is generated. Reusing the keystream compromises security and allows attackers to deduce parts of the plaintext.
3. Block Dependency: OFB, like CFB, processes the plaintext in blocks, which can lead to performance issues when encryption or decryption operations require parallel processing.
4. Failure to Detect Tampering: Unlike some other modes like Cipher Block Chaining (CBC) with a MAC (Message Authentication Code), OFB does not provide built-in data integrity or authentication. It only focuses on confidentiality, and therefore, it's crucial to combine it with a separate integrity verification mechanism.

**Q-9) Explain why there is no need for ciphertext stealing in CFB, OFB, and CTR modes.**

**Ans :**

Ciphertext stealing is a technique used in some block cipher modes to handle the encryption of the last block of plaintext when it's shorter than the block size. This technique essentially modifies the way the final incomplete block is encrypted and provides a way to ensure that the ciphertext is the same length as the plaintext. However, in certain block cipher modes, specifically CFB (Cipher Feedback), OFB (Output Feedback), and CTR (Counter) modes, there is no need for ciphertext stealing. Here's why:

* CFB (Cipher Feedback) Mode:

In CFB mode, the block cipher is applied directly to the ciphertext from the previous encryption operation (initially, the IV is used). The output of the block cipher is then XORed with the plaintext to produce the ciphertext. Since CFB mode doesn't require padding to handle incomplete blocks, there's no need for ciphertext stealing. The block cipher operates in a stream-like fashion, and the ciphertext length is exactly the same as the plaintext length.

* OFB (Output Feedback) Mode:

OFB mode is similar to CFB mode in that it operates in a stream-like manner. The block cipher is applied to the IV or the previous ciphertext block to produce a keystream, which is then XORed with the plaintext to generate the ciphertext. Like CFB mode, there's no need for ciphertext stealing in OFB mode because it doesn't involve padding to accommodate incomplete blocks. The ciphertext length matches the plaintext length.

* CTR (Counter) Mode:

In CTR mode, a counter value is encrypted by the block cipher to generate a keystream. This keystream is then XORed with the plaintext to produce the ciphertext. The counter value typically starts at a specific number and increments for each block. CTR mode does not require ciphertext stealing because it doesn't depend on the padding of incomplete blocks. It generates the same amount of keystream as there is plaintext data, ensuring that the ciphertext length matches the plaintext length.

**Q-10 )**

1. **What is the need of S-box? Explain two types of S-boxes.**

**Ans :**

Substitution boxes, commonly referred to as S-boxes, play a crucial role in modern symmetric key cryptography, especially in block ciphers. They are used to introduce non-linearity into the encryption process, making it more resistant to various cryptographic attacks. S-boxes are an integral part of the confusion layer in cryptographic algorithms, and they help achieve the desired security properties. Here, I'll explain the need for S-boxes and introduce two types of S-boxes:

Need for S-boxes:

1. Non-linearity: S-boxes introduce non-linearity into the encryption process, ensuring that small changes in the input (plaintext or ciphertext) result in significant and unpredictable changes in the output. This nonlinearity helps prevent linear cryptanalysis, differential cryptanalysis, and other attacks that exploit predictable mathematical relationships.
2. Confusion: S-boxes enhance the confusion property of a cryptographic algorithm. They make it difficult for an attacker to deduce the relationship between the plaintext and ciphertext by substituting values in a complex and non-linear manner.
3. Avalanche Effect: S-boxes contribute to the avalanche effect, where a minor change in the input (even a single bit) results in a cascade of changes throughout the output. This property makes it extremely challenging for attackers to infer information about the key or plaintext from the ciphertext.
4. Security: S-boxes are an essential component in the security of many symmetric key ciphers, including the Advanced Encryption Standard (AES), where the design and properties of S-boxes are carefully analyzed to resist a wide range of cryptographic attacks.

Two Types of S-boxes:

1. Substitution-Permutation Network (SPN) S-box:

* This type of S-box is commonly used in block ciphers like AES. It performs both substitution and permutation operations. In an SPN S-box, each input byte (or a small block) is replaced by another byte according to a predefined substitution table. This substitution step introduces non-linearity and confusion. After substitution, a permutation operation is often applied to the data to further increase confusion and spread the influence of each input byte across multiple output bytes.

1. Feistel Network S-box:

* Feistel ciphers, such as DES (Data Encryption Standard), use a specific type of S-box within their structure. In a Feistel network, the data is divided into two halves, and the S-box is applied separately to one of the halves. The result is used to modify the other half. This design allows for easy and efficient key expansion and the application of the S-box in a controlled and balanced manner.

**B) What is the need of D-box? How many types of D-boxes can be used in modern block ciphers?**

**Ans :**

The term "D-box" is not a standard term in modern block ciphers, and its usage is not commonly found in cryptographic literature. Instead, what you might be referring to is the "Diffusion" or "Data Diffusion" layer in block ciphers. The Diffusion layer is responsible for spreading the influence of individual bits or blocks of data throughout the entire ciphertext, which is crucial for achieving security properties in symmetric key cryptography.

In modern block ciphers, the diffusion layer helps fulfill the following needs:

1. Avalanche Effect: The diffusion layer ensures that a small change in the input (plaintext or key) results in a significant and unpredictable change in the output (ciphertext). This property is known as the avalanche effect and is vital for the security of the cipher.
2. Confusion: The diffusion layer adds complexity and confusion to the encryption process. It makes it challenging for attackers to deduce the relationship between the plaintext, key, and ciphertext.
3. Security: The design and properties of the diffusion layer are carefully analyzed to resist a wide range of cryptographic attacks, including linear and differential cryptanalysis.

The diffusion layer in modern block ciphers typically consists of various operations, such as bitwise XOR, modular addition, substitution-permutation networks, and other operations that help achieve the desired diffusion and confusion effects.

Modern block ciphers like the Advanced Encryption Standard (AES) use diffusion layers to ensure that cryptographic properties such as non-linearity, avalanche effect, and confusion are maintained. These diffusion layers are an integral part of the overall cipher design and contribute significantly to its security.

There isn't a standardized categorization of "types of D-boxes" in modern block ciphers, as the design of the diffusion layer varies from one cipher to another. The specific operations used and the order in which they are applied depend on the design choices made by the cipher's developers. Each block cipher may have its unique approach to achieving diffusion and security.

**Q-11) Name any 10 components used in modern block ciphers.**

**Ans :**

Modern block ciphers are cryptographic algorithms that operate on fixed-size blocks of data and are widely used to provide data confidentiality and integrity. They consist of various components and operations to achieve their security objectives. Here are 10 components commonly found in modern block ciphers:

1. Substitution-Permutation Network (SPN): SPN is a common structure in modern block ciphers, dividing the data into blocks and applying substitution and permutation operations iteratively.
2. S-boxes (Substitution Boxes): S-boxes perform non-linear substitution, replacing plaintext or intermediate data with values from predefined tables. They introduce confusion and non-linearity into the cipher.
3. P-boxes (Permutation Boxes): P-boxes are used to permute bits within a data block, further increasing confusion and the avalanche effect.
4. Round Keys: Modern block ciphers use a key schedule to generate round keys from the original encryption key. These round keys are used in each round of the encryption process.
5. Rounds: A round is a sequence of operations applied to the data. Multiple rounds are used to ensure the diffusion and confusion of the data.
6. Feistel Structure: Some block ciphers, such as DES (Data Encryption Standard), use a Feistel network, which divides data into halves and applies a series of operations to each half, including the use of S-boxes.
7. Data Diffusion Layer: The diffusion layer spreads the influence of individual bits or blocks of data throughout the ciphertext, contributing to the avalanche effect.
8. Bitwise XOR Operations: XOR (exclusive OR) operations are commonly used to combine data with round keys or other intermediate results.
9. Key Whitening: Key whitening involves XORing the plaintext with a value derived from the encryption key before the actual encryption process begins, enhancing security.
10. Confusion and Diffusion Layers: These layers are composed of a combination of operations, such as substitution, permutation, and XOR, which are applied iteratively in each round to achieve the desired cryptographic properties.

**Q-12) Differentiate between the two classes of product cipher.**

**Ans :**

|  |  |
| --- | --- |
| **Substitution-Permutation Network (SPN)** | **Feistel Network** |
| Structure: SPN ciphers have a specific block structure where the plaintext is divided into blocks that undergo multiple rounds of substitution and permutation operations. Each round usually consists of a substitution (S-box) layer followed by a permutation (P-box) layer. | Structure: Feistel network ciphers divide the data into two halves and apply a series of operations to one half, followed by a mixing operation involving the other half. The output of one half becomes the input to the next round, and this process repeats for multiple rounds. |
| Substitution: In SPN, the substitution layer uses S-boxes to replace bits or blocks of data with values from predefined tables. S-boxes introduce non-linearity and confusion into the encryption process. | Key Scheduling: In Feistel networks, key scheduling is often more straightforward compared to SPN ciphers, as the same subkey can be applied to both halves of the data in each round. |
| Permutation: The permutation layer rearranges bits within the data block, further increasing confusion and the avalanche effect. | Rounds: Feistel ciphers use multiple rounds, and the number of rounds may vary depending on the cipher. The Feistel structure ensures that each round maintains the same structure and can be efficiently implemented. |
| Rounds: SPN ciphers typically use multiple rounds, and the number of rounds may vary depending on the cipher. Each round involves applying S-boxes, P-boxes, and other operations. | Examples: DES (Data Encryption Standard) is a classic example of a Feistel network-based block cipher. |
| Examples: AES (Advanced Encryption Standard) is a well-known example of an SPN-based block cipher. | S-boxes: Feistel networks also use S-boxes, but they apply them in the operation on one half of the data block. S-boxes introduce non-linearity and confusion, similar to SPN ciphers. |

**Q-13) Distinguish between synchronous and asynchronous stream ciphers.**

**Ans :**

|  |  |
| --- | --- |
| **Synchronous Stream Ciphers:** | Asynchronous Stream Ciphers |
| Keystream Generation:Synchronous stream ciphers generate the keystream based on the encryption key and an initialization vector (IV) or a nonce. The keystream is typically generated independently of the plaintext and ciphertext data, and it is generated in advance of the data transmission. | Keystream Generation: Asynchronous stream ciphers generate the keystream based on the plaintext data itself, often in a self-synchronizing manner. The keystream depends on the previous ciphertext and plaintext bits, making them more adaptable to variations in data. |
| Synchronization: The sender and receiver must be synchronized regarding the keystream generation. Both parties must use the same key and IV or nonce to generate the same keystream. If synchronization is lost, data can be lost or corrupted. | Synchronization:Asynchronous stream ciphers do not require explicit synchronization between the sender and receiver. They can automatically adapt to changes in the data, such as packet loss or transmission errors. |
| Parallel Processing: Synchronous stream ciphers are generally efficient for parallel processing. Multiple bits of the keystream can be generated simultaneously, allowing for high-speed encryption and decryption. | Sequential Processing:Asynchronous stream ciphers process data sequentially, which may be less efficient for parallel processing compared to synchronous stream ciphers. |
| Examples: Common examples of synchronous stream ciphers include RC4 and the A5/1 algorithm used in GSM (Global System for Mobile Communications). | Examples:An example of an asynchronous stream cipher is the CMEA (Cryptographically Enhanced Miniature Algorithm) used in Bluetooth. |

**Q-14) Name any two block ciphers influenced by DES.**

**Ans :**

The Data Encryption Standard (DES) has had a significant influence on the development of subsequent block ciphers. Two block ciphers that were influenced by DES are:

Triple DES (3DES): Triple DES, also known as 3DES or TDEA, is a symmetric key block cipher that uses the Data Encryption Standard (DES) algorithm three times in a cascade. It was developed to provide increased security compared to the original DES, mainly by using multiple rounds of encryption. While 3DES is much more secure than single DES, it is considered relatively slow and has been largely replaced by more efficient and secure ciphers like the Advanced Encryption Standard (AES).

DESX: DESX is another block cipher that builds upon the original DES algorithm. It was designed to provide additional security by incorporating an XOR operation with an additional subkey into the encryption process. The inclusion of this XOR operation aims to strengthen DES against certain attacks, making it more resistant to differential and linear cryptanalysis.

**Q-15) Comment on the weaknesses in DES due to**

**a) Design of S-box**

**Ans :**

` Weaknesses: The design of the S-boxes in DES has several weaknesses:

1. Small S-box Size: Each S-box in DES is relatively small, consisting of only 6 bits in, 4 bits out. This limited size means that the S-boxes do not provide strong non-linearity and are susceptible to differential and linear cryptanalysis attacks.
2. Deterministic: The S-boxes are fixed and publicly known, which means their behavior is deterministic. This property makes DES vulnerable to known-plaintext attacks and attacks that can exploit the specific characteristics of these fixed S-boxes.
3. Fixed Permutation: The S-boxes' fixed permutation provides limited diffusion in the encryption process, potentially weakening DES's security against various attacks.

**b) Design of D-box**

**Ans :**

Weaknesses: The diffusion layer in DES also has some limitations:

1. Fixed Structure: The diffusion layer has a fixed and known structure, which means that attackers can exploit this predictability to launch attacks.
2. Limited Diffusion: The diffusion layer's structure does not provide optimal diffusion of data bits across the ciphertext, which can make it vulnerable to certain attacks.
3. Complexity: The complexity of the diffusion layer is less than that of modern block ciphers, which makes DES more susceptible to differential and linear cryptanalysis.

**c) Key size**

**Ans :**

Weaknesses: The key size in DES is considered one of its most significant weaknesses:

1. Small Key Size:DES uses a 56-bit key, which is relatively small by today's standards. This limited key size makes brute force attacks feasible using modern computing power, and it significantly reduces the resistance to key search attacks.
2. Key Management: The 56-bit key requires effective key management practices to ensure security. If keys are not properly managed, it can lead to vulnerabilities.

**Q-16) Explain the steps in 1 round of AES with example.**

**Ans :**

The Advanced Encryption Standard (AES) is a widely used symmetric key block cipher that operates on fixed-size blocks of data. AES consists of a series of rounds, and each round involves several steps to encrypt or decrypt data. Here, I'll explain the steps in one round of AES, which is common to both encryption and decryption.

AES operates on 128-bit blocks (16 bytes) of data and supports key sizes of 128, 192, or 256 bits. A round in AES consists of four main steps: SubBytes, ShiftRows, MixColumns, and AddRoundKey. I'll provide an example for a single round of AES-128:

AES-128 Round Example:

Assume we have a 128-bit block of plaintext (input) and a 128-bit round key.

1. SubBytes:

* In this step, each byte of the input block is replaced with a corresponding byte from an S-box (Substitution Box). The S-box is a fixed lookup table.
* Example: If a byte in the input block is 0x53, the SubBytes step would replace it with the corresponding value from the S-box, let's say, 0x7C.

1. ShiftRows:

* In this step, the bytes in the rows of the block are shifted to the left by different amounts.
* Example: For the first row, no shifts are performed. For the second row, the bytes are shifted one position to the left. For the third row, the bytes are shifted two positions to the left. For the fourth row, the bytes are shifted three positions to the left.

1. MixColumns:

* In this step, each column of the block is mixed using a mathematical transformation called the Galois Field operation, which provides diffusion and non-linearity.
* Example: If we have a column [0x2A, 0x7B, 0xD5, 0x83], MixColumns applies a mathematical transformation to mix these values.

1. AddRoundKey:

* In this step, the round key is XORed with the output of the previous steps. The round key is derived from the original encryption key during the key expansion process.
* Example: The round key is XORed with the result of the MixColumns step to produce the output for this round.

The output of this round becomes the input for the next round (or for the final round in the case of encryption). The process is repeated for the specified number of rounds depending on the key size (10 rounds for AES-128, 12 rounds for AES-192, and 14 rounds for AES-256).

Each round ensures that the plaintext is transformed in a complex and secure manner, providing data diffusion and confusion while adding the key's influence to achieve encryption or decryption.

**Q-17) List the criteria defined by NIST for AES.**

**Ans :**

The National Institute of Standards and Technology (NIST) established a set of criteria for selecting the Advanced Encryption Standard (AES), which would replace the aging Data Encryption Standard (DES). The criteria for selecting AES were designed to ensure the new encryption standard's security, efficiency, and suitability for a wide range of applications. Here are the criteria defined by NIST for AES:

1. Security: AES candidates had to demonstrate a high level of security and resistance to various cryptographic attacks, such as differential and linear cryptanalysis. The algorithm had to withstand known types of attacks and demonstrate strong security properties.
2. Efficiency: AES was required to be efficient in terms of both hardware and software implementations. It should be suitable for a broad range of applications, from low-power embedded systems to high-performance servers.
3. Adaptability: The selected algorithm needed to support different key sizes, including 128, 192, and 256 bits. This allowed for customization based on the specific security requirements of various applications.
4. Ease of Implementation: AES candidates were expected to have a clear and efficient design that would make it easy to implement in both hardware and software.
5. Avalanche Effect: AES needed to exhibit a strong avalanche effect, meaning that a small change in the plaintext or key would result in a significantly different ciphertext. This property enhances security.
6. Algorithm and Implementation Characteristics: The algorithm had to be suitable for both block and stream cipher modes, and it needed to work well in a wide variety of platforms and environments.
7. Flexibility: The selected algorithm should be flexible enough to accommodate future changes in technology and cryptographic requirements.
8. Public Scrutiny: NIST encouraged public scrutiny and peer review of the AES candidates. It sought input and feedback from the cryptographic community to ensure the highest level of security and confidence in the selected algorithm.

The AES competition, which was open to public participation, led to the selection of Rijndael as the Advanced Encryption Standard. Rijndael met all these criteria and has since become one of the most widely used encryption standards in the world, offering strong security and efficiency across various applications and platforms.

**Q-18) Find the inverse of 550 in GF(1759) using extended Euclidean Theorem.**

**Ans**

**Q-19) Prove the secret exchange of key proposed by Diffie Hellman.**

**Ans**

**Q-20)**

1. **Explain with an example how meet in the middle attack is possible in Diffie Hellman key exchange.**

**Ans :**

A meet-in-the-middle attack is a cryptographic attack that targets symmetric key ciphers and certain types of cryptographic protocols, including the Diffie-Hellman key exchange. It involves trying all possible key combinations to find a match, effectively "meeting in the middle" of the search space. In the context of Diffie-Hellman, this attack is known as a meet-in-the-middle attack on Diffie-Hellman. Here's an explanation with an example of how this attack can be possible:

Diffie-Hellman Key Exchange:

The Diffie-Hellman key exchange is a public-key cryptographic protocol that allows two parties to securely establish a shared secret key over an insecure communication channel. It is based on modular exponentiation and the difficulty of the discrete logarithm problem.

The key exchange works as follows:

1. Alice and Bob publicly agree on a large prime number (p) and a primitive root modulo p (g).
2. Alice chooses a secret random integer (a) and computes A = g^a mod p, which she sends to Bob.
3. Bob chooses a secret random integer (b) and computes B = g^b mod p, which he sends to Alice.
4. Alice and Bob each compute the shared secret key as follows:

* Alice: K = B^a mod p
* Bob: K = A^b mod p

Meet-in-the-Middle Attack on Diffie-Hellman:

In a meet-in-the-middle attack on Diffie-Hellman, an attacker intercepts the public values (p, g, A, B) exchanged between Alice and Bob and tries to find the secret values (a, b) that correspond to these public values. This attack assumes that the secret values (a, b) are relatively small, and it is feasible to try all possible values.

Here's a simplified example to illustrate the attack:

1. Alice and Bob agree on p = 23 and g = 5.
2. Alice chooses a = 7 and computes A = 5^7 mod 23 = 10.
3. Bob chooses b = 4 and computes B = 5^4 mod 23 = 18.

The attacker intercepts A = 10 and B = 18 and proceeds with the meet-in-the-middle attack:

1. The attacker precomputes a list of possible values of A^x mod p for all potential values of x. This list is stored.
2. The attacker then computes all possible values of B \* g^y mod p for all potential values of y and checks if any of these values match the values in the precomputed list.

In this simplified example, the attacker would find that A^x = B \* g^y for x = 3 and y = 5. Therefore, the attacker has found a match, which corresponds to Alice's secret (a) and Bob's secret (b).

In practice, this attack becomes computationally infeasible as the size of the prime number (p) and the secret values (a, b) increase. Modern cryptographic protocols using Diffie-Hellman use sufficiently large prime numbers and random secret values to resist meet-in-the-middle attacks.

1. **Prove meet in the middle attack in Diffie Hellman key exchange.**

**Ans :**

A meet-in-the-middle attack on the Diffie-Hellman key exchange can be proven by demonstrating how an attacker can successfully find the shared secret key by trying all possible combinations of secret values for Alice and Bob. I'll provide a step-by-step proof of the meet-in-the-middle attack:

Step 1: Setup

* Alice and Bob agree on a prime number (p) and a primitive root modulo p (g) as public parameters.
* Alice chooses a secret random integer "a."
* Bob chooses a secret random integer "b."
* Alice computes A = g^a mod p.
* Bob computes B = g^b mod p.
* Both Alice and Bob send their public values A and B to each other.

Step 2: Attacker's Intercept

* The attacker intercepts the public values A and B exchanged between Alice and Bob, as well as the known prime number "p" and primitive root "g."

Step 3: Meet-in-the-Middle Attack

* The attacker proceeds with the meet-in-the-middle attack by trying all possible values of "a" and "b" to find a matching pair (a, b) such that A = B.

1. Generate a list of all possible values of A^x mod p for all potential values of x, and store this list.
2. Compute all possible values of B \* g^y mod p for all potential values of y and check if any of these values match the values in the precomputed list.

* When a match is found (i.e., A^x = B \* g^y for some values of x and y), the attacker has successfully identified the secret values a and b that correspond to the public values A and B.

Step 4: Recover the Shared Secret Key

* Now that the attacker has found a matching pair of secret values (a, b), they can compute the shared secret key as follows:
* Alice's secret key: K\_A = B^a mod p
* Bob's secret key: K\_B = A^b mod p

Since A = g^a mod p and B = g^b mod p, the shared secret key computed by both Alice and Bob will match:

K\_A = (g^b mod p)^a mod p = (g^a mod p)^b mod p = K\_B

Conclusion:

The meet-in-the-middle attack on the Diffie-Hellman key exchange has been successfully demonstrated. By trying all possible combinations of secret values for Alice and Bob, the attacker can identify the shared secret key used by both parties. However, it is important to note that the feasibility of this attack depends on the size of the prime number "p" and the secret values "a" and "b." Using sufficiently large values for these parameters makes the attack computationally infeasible.

**Q-21) Describe pseudorandom number generation based on RSA.**

**Ans :**

Pseudorandom number generation based on RSA involves using the mathematical properties of the RSA algorithm to generate cryptographically secure pseudorandom numbers. The process typically leverages the modular exponentiation operation in RSA. Here's an overview of how this can be done:

1. Key Generation:

* Start by generating an RSA key pair, consisting of a public key (n, e) and a private key (n, d). The public key is used for encryption, while the private key is used for decryption.

1. Seed Value:

* Begin with a random seed value (an initial value), which serves as the starting point for the pseudorandom number generation process.

1. Modular Exponentiation:

* To generate pseudorandom numbers, repeatedly perform modular exponentiation operations using the seed value, exponent, and modulus from the public key.
* The basic operation is to raise the seed value to the power of the public exponent (e) modulo the public modulus (n). In mathematical terms:

|  |
| --- |
| pseudorandom\_number = seed\_value^e mod n |

* Here, "pseudorandom\_number" is the result of the modular exponentiation.

1. Next Seed Value:

* After obtaining the pseudorandom number, it becomes the new seed value for the next iteration.

1. Repeat:

* Repeat the modular exponentiation process as many times as needed to generate the desired number of pseudorandom numbers.

1. Pseudorandom Number Output:

* The output of each modular exponentiation operation is considered a pseudorandom number. These numbers are generated in a deterministic and predictable manner, based on the initial seed value and the RSA key pair.

The key security property here is the difficulty of factoring the public modulus "n" to derive the private key (d) from the public key (e, n). Without knowledge of the private key, it is computationally infeasible to predict or reverse-engineer the pseudorandom numbers generated through this method.

It's important to note that using RSA for pseudorandom number generation is not a common or recommended practice. Pseudorandom number generators (PRNGs) specifically designed for cryptographic purposes, such as the ones based on block ciphers or hash functions, are generally preferred. These dedicated PRNGs have been extensively analyzed and tested for security, and they provide strong pseudorandom output with known and proven properties.

**Q-22) Illustrate Elgamal cryptographic system.**

**Ans :**

The ElGamal cryptographic system is a public-key encryption scheme that provides confidentiality and authentication for secure communication. It is based on the mathematical properties of the Diffie-Hellman key exchange and uses the concept of a trapdoor one-way function. ElGamal is typically used for secure communication, digital signatures, and other cryptographic applications. Here's an illustration of how the ElGamal cryptographic system works:

Key Generation:

1. Key Pair Generation:

* Alice generates her key pair: a private key (x) and a corresponding public key (y). The private key is kept secret, while the public key is made available to anyone who wishes to send her encrypted messages or verify her digital signatures.

1. Public Parameters:

* Alice also selects public parameters that include a prime number (p) and a generator (g) of the multiplicative group modulo p. These parameters are used by both Alice and Bob and are typically publicly known.

Encryption (Sender's Side):

1. Message Preparation:

* Bob wants to send a secure message to Alice. He begins by converting his plaintext message (M) into a numerical value.

1. Random Number Generation:

* Bob generates a random number (k) that is relatively prime to (p - 1), which ensures the security of the encryption.

1. Key Agreement:

* Bob calculates the shared secret (K) using the Diffie-Hellman key exchange: K = (y^k) mod p.

1. Ciphertext Generation:

* Bob computes the ciphertext components:
* C1 = (g^k) mod p
* C2 = (M \* K) mod p
* The ciphertext is (C1, C2).

1. Message Transmission:

* Bob sends the ciphertext (C1, C2) to Alice.

Decryption (Receiver's Side):

1. Ciphertext Processing:

* Alice receives the ciphertext (C1, C2).

1. Shared Secret Calculation:

* Alice calculates the shared secret (K) using the private key: K = (C1^x) mod p.

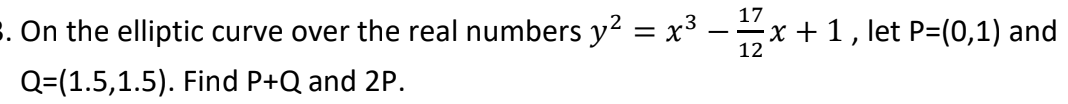
1. Plaintext Recovery:

* Alice calculates the modular multiplicative inverse of K modulo p, denoted as K⁻¹, which allows her to recover the original message: M = (C2 \* K⁻¹) mod p.

Illustration Summary:

* Bob encrypts a message M using Alice's public key (y) and public parameters (p, g).
* Alice decrypts the message using her private key (x) to recover the original plaintext.
* The security of ElGamal relies on the difficulty of solving the discrete logarithm problem in a multiplicative group modulo a prime (finding x given y, g, and p).

ElGamal provides confidentiality through encryption and can be used for digital signatures to achieve authentication and non-repudiation. It is a widely used public-key cryptographic system that offers strong security when implemented properly.

**Q-23)** 

Ans :

Q-24)

**Q-25) You want to secretly send a message to your friend using public key cryptography. Which one would you prefer: RSA or ECC? Justify your choice.**

**Ans :**

The choice between RSA (Rivest-Shamir-Adleman) and ECC (Elliptic Curve Cryptography) for sending a secret message using public key cryptography depends on various factors, including security, efficiency, and the specific use case. Here's a justification for each option:

RSA:

1. Security: RSA is a well-established public key cryptography algorithm with a long history of analysis and scrutiny. When used with appropriate key sizes, it provides strong security against known attacks, including factoring large numbers, which is the basis of RSA's security.
2. Compatibility: RSA is widely supported in cryptographic libraries and systems, making it a reliable choice for interoperability with a variety of platforms and applications.
3. Key Length: RSA typically requires longer key lengths to achieve the same level of security as shorter ECC key lengths. This can make RSA less efficient, especially when using large keys.
4. Performance: RSA encryption and decryption operations can be relatively slow compared to ECC, especially for longer key lengths. This can be a concern for resource-constrained devices and applications.

ECC:

1. Security: ECC offers strong security with shorter key lengths compared to RSA. It is based on the difficulty of the elliptic curve discrete logarithm problem, which makes it more computationally efficient.
2. Efficiency: ECC is known for its efficiency in terms of both computation and bandwidth usage. It requires shorter key lengths to achieve the same level of security, making it well-suited for resource-constrained devices, such as mobile phones and IoT devices.
3. Key Length: Shorter key lengths mean that ECC can generate keys faster and use fewer computational resources for encryption and decryption.
4. Size: ECC keys and signatures are shorter, making them more compact and faster to transmit. This is particularly important for data transmission and storage efficiency.

Justification:

The choice between RSA and ECC depends on your specific use case and requirements:

* If you prioritize security and need to ensure that your messages are resistant to the most advanced cryptographic attacks, and if you have the computational resources to handle longer key lengths, RSA may be a suitable choice.
* If you prioritize efficiency, especially in resource-constrained environments or where bandwidth and computational resources are limited, ECC is a more efficient choice due to shorter key lengths and smaller message sizes.

In many modern applications, ECC is preferred for its efficiency, and it is often recommended for use in scenarios like mobile devices, secure messaging apps, and IoT devices. However, for legacy or compatibility reasons, RSA is still widely used and remains a valid choice in certain contexts. Ultimately, the choice should be based on your specific needs, the computational resources available, and your security requirements.

Q-26)