Question 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. 1

In an RSA cryptosystem, the public key consists of two components: the modulus (n) and the encryption exponent (e), and the private key consists of the modulus (n) and the decryption exponent (d).

First, let's calculate the modulus (n) for A:

n = p * q

n = 13 * 17

n = 221

So, the modulus (n) is 221.

The public key of A is given as 35. This is the encryption exponent (e).

Now, we need to calculate the private key, which requires finding the decryption exponent (d). In RSA, the decryption exponent is calculated using the modular multiplicative inverse of the encryption exponent (e) modulo the totient (ϕ) of n. The totient (ϕ) of n is given by:

ϕ(n) = (p - 1) * (q - 1)

ϕ(221) = (13 - 1) * (17 - 1)

ϕ(221) = 12 * 16

ϕ(221) = 192

Now, we need to find the modular multiplicative inverse of e (35) modulo ϕ(n) (192). We can use the extended Euclidean algorithm to calculate this:

e * d ≡ 1 (mod ϕ(n))

35 * d ≡ 1 (mod 192)

Using the extended Euclidean algorithm, you can find that d = 83 satisfies this equation. Therefore, the private key of A is (n, d), which is (221, 83).

Question 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.2

In the RSA algorithm, you can calculate the ciphertext (C) for a given plaintext (P) using the following formula:

C = (P^e) mod n

Where:

P is the plaintext.

e is the encryption exponent.

n is the modulus (n = p * q, where p and q are prime numbers).

Given the values you provided:

p = 3

q = 13

d = 7

e = 3

P (plaintext) = 5

First, calculate n:

n = p * q = 3 * 13 = 39

Now, plug these values into the formula to calculate the ciphertext (C):

C = (5^3) mod 39

C = (125) mod 39

Now, calculate the modulus:

C = 125 - (39 * 3)

C = 125 - 117

C = 8

So, the ciphertext (C) for the plaintext 5 using the given values of p, q, d, and e is 8.

Question 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.3

In the Diffie-Hellman key exchange, both parties agree on a modulus (p) and a primitive root (g). Each party chooses its own secret exponent (a for Party A and b for Party B) without sharing them. Then they exchange public values calculated using the modulus and primitive root. Finally, they use each other's public values to compute the shared secret key.

In your scenario:

Modulus (p) = 7

Primitive root (g) = 3

Party A's secret (a) = 2

Party B's secret (b) = 5

Now, both parties calculate their respective public values:

Party A's public value (A):

A = (g^a) mod p

A = (3^2) mod 7

A = (9) mod 7

A = 2

Party B's public value (B):

B = (g^b) mod p

B = (3^5) mod 7

B = (243) mod 7

B = 6

Now, Party A sends its public value (A = 2) to Party B, and Party B sends its public value (B = 6) to Party A.

Now, both parties can compute the shared secret key:

Party A calculates the shared key:

Shared key (K) = (B^a) mod p

K = (6^2) mod 7

K = (36) mod 7

K = 1

Party B calculates the shared key:

Shared key (K) = (A^b) mod p

K = (2^5) mod 7

K = (32) mod 7

K = 1

Both parties now have the same shared Diffie-Hellman key, which is 1. This shared key can be used for secure communication between Party A and Party B.

Question 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?

In the Diffie-Hellman Key Exchange, both Alice and Bob need to perform some calculations to exchange a shared secret key. Here are the given values:

Prime value (q) = 17

Primitive root (g) = 5

Alice's secret key (a) = 4

Bob's secret key (b) = 6

Now, both Alice and Bob will calculate their public values:

Alice's public value (A):

A = (g^a) mod q

A = (5^4) mod 17

To calculate A:

A = (625) mod 17

A = 5

Bob's public value (B):

B = (g^b) mod q

B = (5^6) mod 17

To calculate B:

B = (15625) mod 17

B = 6

Now, Alice and Bob exchange their public values, A and B.

To calculate the shared secret key:

Alice computes the shared secret key (K) using Bob's public value (B) and her secret key (a):

K = (B^a) mod q

K = (6^4) mod 17

To calculate K:

K = (1296) mod 17

K = 11

Bob computes the shared secret key (K) using Alice's public value (A) and his secret key (b):

K = (A^b) mod q

K = (5^6) mod 17

To calculate K:

K = (15625) mod 17

K = 11

Both Alice and Bob have computed the same shared secret key, which is 11. This shared key can now be used for secure communication between them.

Question 5. What is trapdoor one-way function?

A trapdoor one-way function is a mathematical function that is easy to compute in one direction but difficult to compute in the reverse direction without special knowledge or a "trapdoor." The concept of a trapdoor one-way function is essential in many cryptographic systems, as it forms the basis for various encryption and security schemes.

One-Way Function: A one-way function is a function that is easy to compute for any input but computationally difficult to reverse. In other words, given the output of the function, it should be challenging to find the input that produced it.

Trapdoor: A trapdoor is a special piece of information or knowledge that allows one to reverse the one-way function efficiently. It acts as a secret key or additional information that makes it possible to compute the reverse of the function.

In practical cryptography, trapdoor one-way functions are used in various encryption and digital signature algorithms. For example, in public-key encryption systems like RSA, the trapdoor is the private key that allows the recipient to decrypt the message. Without the private key, it should be computationally infeasible for an adversary to reverse the encryption

Question 6. Explain knapsack cryptosystem.

he knapsack cryptosystem is a type of asymmetric cryptosystem that was proposed in the late 1970s. It relies on the difficulty of solving a specific type of knapsack problem, which is an NP-hard problem. Here's how it works:

Key Generation:

The encryption key consists of a super-increasing sequence of positive integers, which is a sequence where each element is larger than the sum of all previous elements.

The decryption key is derived from the super-increasing sequence and a multiplier and modulus.

Encryption:

To encrypt a message, the sender converts the plaintext message into binary form.

The sender then forms a set of numbers by selecting elements from the super-increasing sequence that correspond to the 1s in the binary representation.

The encrypted ciphertext is the sum of the selected numbers.

Decryption:

The recipient uses the decryption key, which includes the super-increasing sequence, multiplier, and modulus, to recover the original set of numbers.

The recipient applies a modulo operation to these numbers to obtain a residue set.

The residue set is converted into a binary form, which represents the original plaintext message.

The security of the knapsack cryptosystem relies on the difficulty of solving the knapsack problem. The standard knapsack problem is NP-hard, meaning it's computationally difficult to find the exact solution for large instances. However, it's important to note that early knapsack cryptosystems were found to be insecure due to specific attacks, so they are not widely used in practice. Modern cryptographic systems, like RSA and elliptic curve cryptography, are generally more secure and efficient for securing communications.

Question 7. Name 7 categories of attacks on RSA. Explain any five in detail.

7 Categories of Attacks on RSA:

Brute Force Attack: In a brute force attack, an attacker systematically tries every possible private key until the correct one is found. This attack is computationally infeasible when long key lengths are used.

Factorization Attack: RSA's security is based on the difficulty of factoring the product of two large prime numbers. In a factorization attack, an attacker attempts to factor the public modulus (n) to obtain the private key. Advanced algorithms like the General Number Field Sieve (GNFS) are used for this purpose.

Timing Attacks: Timing attacks exploit variations in the execution time of cryptographic operations to infer information about the private key. By measuring the time it takes for certain operations to complete, an attacker may deduce information about the key.

Side-Channel Attacks: Side-channel attacks are a class of attacks that exploit physical or implementation vulnerabilities, such as power consumption, electromagnetic radiation, or memory access patterns, to obtain information about the private key.

Chosen Ciphertext Attack (CCA): In a chosen ciphertext attack, the attacker has the ability to choose ciphertexts to be decrypted and observe the corresponding plaintexts. By doing this, they aim to gain information about the private key.

Common Modulus Attack: In a common modulus attack, the same modulus (n) is used for multiple RSA public keys. If an attacker can obtain both the ciphertext and public key for two different users, they may be able to recover the plaintext.

Small Exponent Attack: In a small exponent attack, the attacker takes advantage of the use of small public exponents (e) to recover the plaintext by exploiting properties of modular arithmetic.

Question 8. Discuss the security issues in

a) cipher feedback mode

b) output feedback mode

Security Issues in Cipher Feedback (CFB) Mode:

Error Propagation: If a single bit error occurs in the ciphertext, it can cause incorrect decryption of multiple blocks until the error is detected, leading to potential data corruption.

Initialization Vector (IV) Management: The IV must be unique for each message. Reusing an IV with the same key can lead to security vulnerabilities, as attackers can exploit patterns in the ciphertext.

Synchronization: CFB mode requires the sender and receiver to stay synchronized. If there is any loss of data or if the sender and receiver become out of sync, it can lead to decryption errors.

Security Issues in Output Feedback (OFB) Mode:

Error Propagation: Similar to CFB, errors in the ciphertext can propagate, affecting the decryption of subsequent blocks.

Initialization Vector (IV) Management: Proper IV management is crucial to the security of OFB. Reusing IVs can lead to serious vulnerabilities, as an attacker might be able to predict the keystream and recover the plaintext.

Lack of Data Integrity: OFB mode does not provide data integrity on its own. It only provides confidentiality. To ensure data integrity, additional mechanisms such as message authentication codes (MACs) are needed.

Key Reuse: Reusing the same key with the same IV in OFB mode can result in the same keystream, which is a security risk. Each IV should be unique with the same key to maintain security.

To address these security issues, it's important to properly manage IVs, use error-detection/correction mechanisms, and consider data integrity and synchronization in the design and implementation of cryptographic systems using CFB and OFB modes.

Question 11. Explain why there is no need for ciphertext stealing in CFB, OFB, and CTR modes.

No Need for Ciphertext Stealing in CFB, OFB, and CTR Modes:

Ciphertext stealing is a technique used in some block cipher modes like CBC (Cipher Block Chaining) to ensure that the last ciphertext block has the same length as the plaintext, especially when padding is used. However, in CFB (Cipher Feedback), OFB (Output Feedback), and CTR (Counter) modes, there's no need for ciphertext stealing, and here's why:

CFB (Cipher Feedback) Mode: In CFB mode, the ciphertext is generated by encrypting the feedback of the previous ciphertext block (which is an intermediate result) with the block cipher. This means that the ciphertext length is determined by the block size of the cipher, and there is no padding applied. Therefore, there is no need for ciphertext stealing because the ciphertext length matches the plaintext length without any additional measures.

OFB (Output Feedback) Mode: OFB mode works by encrypting an initialization vector (IV) with the block cipher to create a key stream. This key stream is then XORed with the plaintext to produce the ciphertext. Similar to CFB, there's no need for ciphertext stealing because the ciphertext length is determined solely by the block size of the cipher, and there is no padding.

CTR (Counter) Mode: CTR mode turns a block cipher into a stream cipher by encrypting a counter value with the block cipher to generate a pseudorandom stream of bits. This stream is then XORed with the plaintext to produce the ciphertext. Again, there's no need for ciphertext stealing because the ciphertext length is directly derived from the counter value and the block cipher, and there is no padding.

In summary, CFB, OFB, and CTR modes do not require ciphertext stealing because they generate ciphertext in a manner where the length of the ciphertext matches the plaintext naturally without the need for additional padding or modifications.

Question 10. A) What is the need of S-box? Explain two types of S-boxes.

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

block ciphers?

A) Need for S-Box and Two Types:

S-Box (Substitution Box): S-Boxes are a fundamental component in many cryptographic algorithms, particularly in block ciphers. They introduce non-linearity into the encryption process and provide confusion, making it more difficult for attackers to predict the relationship between the input and output.

Two common types of S-Boxes are:

Simple S-Box: Simple S-Boxes are typically used in symmetric key ciphers. They consist of a fixed table that maps a fixed number of input bits to a corresponding number of output bits. An example is the S-Box used in the AES (Advanced Encryption Standard).

Complex S-Box (Non-linear S-Box): Complex S-Boxes are used in various cryptographic algorithms, including hash functions and block ciphers. They are characterized by more complex mathematical operations, such as substitutions and permutations. These S-Boxes enhance the non-linearity and confusion within the algorithm, increasing its security. One example is the S-Box used in the DES (Data Encryption Standard).

B) Need for D-Box and Types:

D-Box (Diffusion Layer): D-Boxes are used in block ciphers to provide diffusion, which means that small changes in the plaintext or key should result in significant changes in the ciphertext. They help spread the influence of one bit or byte of the plaintext or key to many bits in the intermediate and final ciphertext.

In modern block ciphers, there are typically two types of D-Boxes:

Linear D-Box: Linear diffusion layers use linear operations like XOR and modular addition to achieve diffusion. They are computationally efficient and widely used in many symmetric key ciphers.

Non-linear D-Box: Non-linear diffusion layers use non-linear operations like S-Boxes and bitwise operations to achieve diffusion. Non-linear D-Boxes introduce additional security by increasing the complexity of the transformation.

The choice of D-Box depends on the specific design goals and security requirements of the block cipher.

Question 11. Name any 10 components used in modern block ciphers.

10 Components Used in Modern Block Ciphers:

Substitution (S-Box): Non-linear substitution of plaintext or intermediate values.

Permutation (P-Box): Rearrangement of bits in a block.

Key Schedule: Generation of round keys from the master key.

Round Function: A function applied to the data in each round.

Rounds: The number of iterations of the encryption process.

Diffusion Layer (D-Box): A component that spreads the influence of one bit or byte to many bits in the ciphertext.

Confusion Layer: A component that obscures the relationship between the key and ciphertext.

Feistel Network: A structure used in some block ciphers that splits the data into two halves.

Initialization Vector (IV): A value used to initialize the encryption process.

Block Size: The size of the data blocks processed by the cipher.

Question 12. Differentiate between the two classes of product cipher.

Differentiate Between Two Classes of Product Cipher:

Substitution-Permutation Networks (SPN): SPN ciphers are a type of product cipher that involves substituting and permuting data through multiple rounds. They typically consist of S-Boxes for substitution and P-Boxes for permutation. The Advanced Encryption Standard (AES) is an example of an SPN-based block cipher.

Feistel Networks: Feistel ciphers are another class of product ciphers that split the data into two halves and process them separately in each round. The two halves are often subject to different operations, and the results are combined. The Data Encryption Standard (DES) is a well-known Feistel cipher.

Both SPN and Feistel ciphers provide confusion and diffusion, making them suitable for secure block cipher designs.

Question 13. Distinguish between synchronous and asynchronous stream ciphers.

Distinguish Between Synchronous and Asynchronous Stream Ciphers:

Synchronous Stream Ciphers:

Synchronous stream ciphers generate a keystream based on a synchronized internal state and an initial key.

Keystream bits are generated one at a time and are XORed with the plaintext to produce the ciphertext.

They are suitable for real-time communication and are faster because they do not require additional synchronization information.

Key and initial state need to be securely shared between sender and receiver.

Asynchronous Stream Ciphers:

Asynchronous stream ciphers generate a keystream that is not based on a synchronized clock or internal state.

They are usually slower than synchronous stream ciphers because they require additional bits for synchronization or a longer initialization process.

They are more flexible and can be used in scenarios where strict synchronization is not possible.

They can be more secure against certain types of attacks due to their unpredictable keystream generation.

In practice, the choice between synchronous and asynchronous stream ciphers depends on the specific requirements of the application and the trade-off between speed and security.

14. Name any two block ciphers influenced by DES.

Block Ciphers Influenced by DES:

3DES (Triple Data Encryption Standard): 3DES, also known as Triple DES, is a symmetric key block cipher that was influenced by the original DES algorithm. It applies the Data Encryption Standard algorithm three times to each data block, making it more secure and suitable for modern cryptographic needs.

DESX: DESX is a modified version of DES that combines the original DES algorithm with additional XOR operations to enhance security. It was developed as an alternative to DES and is based on its principles.

15. Comment on the weaknesses in DES due to

a) Design of S-box

b) Design of D-box

c) Key size

a) Design of S-Box:

DES uses fixed S-Boxes (Substitution Boxes), which are not secret. The fixed S-Box design is known, and this makes DES vulnerable to known-plaintext and chosen-plaintext attacks.

There is a lack of diffusion in the S-Box design, meaning that changes in a single bit of the input to the S-Box may not affect multiple bits in the output.

b) Design of D-Box (Diffusion Layer):

DES uses a relatively weak D-Box (Diffusion Layer), which means that the influence of one bit in the plaintext or key does not spread adequately to other bits in the ciphertext. This reduces the overall security and resilience against cryptanalysis.

c) Key Size:

DES uses a relatively small key size of 56 bits. With advances in computing power, DES keys can be brute-forced within a reasonable amount of time using modern hardware. A 56-bit key is no longer considered secure against determined attackers.

16. Explain the steps in 1 round of AES with example.

Steps in 1 Round of AES (Advanced Encryption Standard):

AES operates on data in blocks and consists of multiple rounds. Here are the steps in one round of AES, along with an example:

SubBytes (Substitution):

Each byte in the state is replaced with a corresponding byte from an S-Box (Substitution Box).

Example: If the state is 53 7A 18 FE, after SubBytes, it may become A1 23 45 C9.

ShiftRows (Row Shifting):

Bytes in each row of the state are shifted left by an offset. The first row is not shifted, the second row is shifted one position to the left, the third row is shifted two positions, and the fourth row is shifted three positions.

Example: If the state is A1 23 45 C9, after ShiftRows, it may become A1 23 45 C9.

MixColumns (Column Mixing):

This step applies a mathematical transformation to the columns of the state.

Example: If the state is A1 23 45 C9, after MixColumns, it may become CE B0 53 29.

AddRoundKey (Round Key XOR):

The state is XORed with the round key for the current round.

Example: If the state is CE B0 53 29 and the round key is 11 0F AE E1, the resulting state is the ciphertext for that round.

These steps are repeated for multiple rounds (10, 12, or 14 rounds depending on the key length), providing confusion and diffusion to enhance the security of AES.

17. List the criteria defined by NIST for AES.

Criteria Defined by NIST for AES:

The National Institute of Standards and Technology (NIST) established the following criteria for the selection of the Advanced Encryption Standard (AES):

Security: The selected algorithm should provide a high level of security, making it resistant to various cryptographic attacks, including brute force, differential, and linear cryptanalysis.

Efficiency: The algorithm should be efficient in terms of computational complexity, making it suitable for both software and hardware implementations.

Symmetry: AES is a symmetric key encryption algorithm, meaning the same key is used for both encryption and decryption.

Flexibility: AES should support a range of key sizes, including 128-bit, 192-bit, and 256-bit keys, to address different security requirements.

Algorithm and Implementation Characteristics: The selected algorithm should be well-defined, unpatented, and suitable for implementation in various environments and platforms.

Peer Review: The algorithm should undergo extensive peer review by experts in the field to ensure its security and suitability.

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

Inverse of 550 in GF(1759) using Extended Euclidean Theorem:

To find the inverse of 550 in GF(1759) using the Extended Euclidean Algorithm, we need to compute the greatest common divisor (GCD) of 550 and 1759 and find the Bézout coefficients.

Extended Euclidean Algorithm:

a = 1759, b = 550

Step 1: Initialization

r_0 = a = 1759, r_1 = b = 550

s_0 = 1, s_1 = 0

t_0 = 0, t_1 = 1

Step 2: Iterations

q = r_(i-1) // r_i

r_(i+1) = r_(i-1) - q * r_i

s_(i+1) = s_(i-1) - q * s_i

t_(i+1) = t_(i-1) - q * t_i

Step 3: Continue Iterations

Continue the iterations until r_(i+1) is equal to 1.

Step 4: Result

The Bézout coefficients s_i and t_i when GCD = 1 are the inverse of 550.

In this case, the Bézout coefficients are:

s_i = -124

t_i = 413

To ensure the coefficients are positive, you can compute their positive equivalents:

s = 1759 - 124 = 1635

t = 550 + 413 = 963

So, the inverse of 550 in GF(1759) is 963.

19. Prove the secret exchange of key proposed by Diffie Hellman.

Prove the Secret Exchange of Key Proposed by Diffie-Hellman:

The Diffie-Hellman key exchange protocol allows two parties, traditionally named Alice and Bob, to securely exchange a shared secret key without having to share their private keys. The security of this protocol relies on the difficulty of solving the discrete logarithm problem.

The steps involved in the protocol are as follows:

Initialization: Alice and Bob agree on two public values, a prime number (p) and a primitive root (g) modulo p. These values are known to both parties but not secret.

Key Generation:

Alice selects a secret random number, a, and calculates A = g^a mod p.

Bob selects a secret random number, b, and calculates B = g^b mod p.

Exchange Public Values:

Alice sends A to Bob.

Bob sends B to Alice.

Key Derivation:

Alice computes the shared secret key as S = B^a mod p.

Bob computes the shared secret key as S = A^b mod p.

Both Alice and Bob now have the same shared secret key S, which they can use for secure communication.

20. A) Explain with an example how meet in the middle attack is possible in Diffie

Hellman key exchange.

B) Prove meet in the middle attack in Diffie Hellman key exchange.

. Meet-in-the-Middle Attack in Diffie-Hellman Key Exchange:

A meet-in-the-middle attack is a cryptographic attack that exploits the fact that encryption (or other cryptographic operations) is reversible. In the context of Diffie-Hellman key exchange, it can be used to find the shared secret key by performing calculations in both directions. Here's how it works:

A) Explanation with an Example:

Suppose Alice and Bob are using the Diffie-Hellman key exchange with parameters (p, g) and have exchanged public values A and B. An attacker, Eve, wants to find the shared secret key.

Eve precomputes a table (often called a "meet-in-the-middle table") of all possible values of g^x mod p for various values of x, where x is in the range of possible private keys.

Then, she computes all possible values of (B * g^(-x)) mod p for various values of x, where x is in the range of possible private keys.

If there's a match between the values from the first and second tables, it means she has found the shared secret key, x.

B) Proof of Meet-in-the-Middle Attack:

To prove the attack, consider the following:

The secret key (shared key) is S = (A^x mod p) = (B^y mod p), where x is Alice's private key, y is Bob's private key, and A and B are their public values.

Now, let's rewrite S using the tables generated by Eve:

S = g^(xy) mod p

Eve's tables allow her to compute both g^x and (B * g^(-y)) mod p for various values of x and y.

If a match is found between the two tables for a value, say g^x, and (B * g^(-y)) mod p, it implies g^(xy) mod p, which is the shared secret S, has been found.

21. Describe pseudorandom number generation based on RSA.

Pseudorandom Number Generation Based on RSA:

Pseudorandom number generation based on RSA can be achieved by generating random numbers in the range [1, n-1], where n is the RSA modulus, and checking if the generated number is coprime to n. This process is repeated until a suitable pseudorandom number is found.

Here's how it works:

Choose an RSA modulus n, which is a product of two large prime numbers (p and q).

Calculate the totient ϕ(n) = (p-1) * (q-1).

Generate random integers in the range [1, n-1].

For each random integer, check if it is coprime to n by ensuring that the greatest common divisor (GCD) with n is 1.

If the GCD is 1, the generated integer is considered pseudorandom and can be used.

Pseudorandom number generation based on RSA provides a way to generate pseudorandom numbers that are difficult for an adversary to predict, given the factorization of the RSA modulus n. It can be used for cryptographic applications, such as generating keys or nonces.

22. Illustrate Elgamal cryptographic system.

ElGamal Cryptographic System:

The ElGamal cryptosystem is an asymmetric key encryption algorithm that is based on the Diffie-Hellman key exchange and provides both confidentiality and digital signatures. Here's an illustration of how the ElGamal cryptosystem works:

Key Generation:

Choose a large prime number p and a primitive root g modulo p.

Select a private key x from the range [1, p-2].

Calculate the corresponding public key y = g^x mod p.

Encryption:

To encrypt a message M, the sender selects a random integer k from the range [1, p-2].

Compute c1 = g^k mod p and c2 = (M * y^k) mod p.

The ciphertext is (c1, c2).

Decryption:

To decrypt the ciphertext (c1, c2), the recipient uses their private key x.

Calculate the shared secret s = c1^x mod p.

Compute the multiplicative inverse of s modulo p, denoted as s^(-1).

Decrypt the message as M = (c2 * s^(-1)) mod p.

The ElGamal cryptosystem provides confidentiality through the use of the shared secret s, and it allows digital signatures when used with other algorithms. It is widely used in secure communications and cryptographic applications.

23. On the elliptic curve over the real numbers y

2 = x

3 −

17

12

x + 1 , let P=(0,1) and

Q=(1.5,1.5). Find P+Q and 2P.

24. Solve for the elliptic curve encryption/ decryption. The cryptosystem parameters are

E11(1,6) and G=(2,7). B’s private key is nB=7.

a) Find B’s public key PB

b) A wishes to encrypt the message Pm= (10,9) and chooses the random key k=3.

Determine the ciphertext Cm.

c) Show the calculation by which B recovers Pm from Cm.

Elliptic Curve Operations:

The elliptic curve equation given is y^2 = x^3 - (17/12)x + 1. Let's perform the operations as described:

a) P + Q:

P = (0, 1)

Q = (1.5, 1.5)

We can use the elliptic curve addition formula to find P + Q:

Lambda (λ) = (y2 - y1) / (x2 - x1)

$\lambda = (1.5 - 1) / (1.5 - 0) = 0.5 / 1.5 = 1/3$

$x3 = \lambda^2 - x1 - x2$

$x3 = (1/3)^2 - 0 - 1.5 = 1/9 - 1.5$

Now, we can find y3:

$y3 = \lambda(x1 - x3) - y1$

$y3 = (1/3)(0 - 1/9) - 1 = -1/27 - 1$

So, P + Q = (1/9 - 1.5, -1/27 - 1).

b) 2P:

We can use the same formulas as above with P as both points:

$\lambda = (y2 - y1) / (x2 - x1) = (1 - 1) / (0 - 0) =$ undefined

Since λ is undefined, the line is vertical. Therefore, 2P = (0, 0).

Elliptic Curve Encryption/Decryption:

a) B's Public Key (PB):

B's public key is calculated as PB = nB * G, where nB is B's private key and G is the generator point.

PB = 7 * (2, 7) = (14, 49).

b) Encryption (Cm):

A chooses a random key k = 3.

A calculates k * G = 3 * (2, 7) = (6, 21).

A calculates the ciphertext Cm = Pm + k * PB = (10, 9) + (6, 21) = (16, 30).

c) Decryption (Recovering Pm):

B uses the private key nB = 7 to calculate the shared secret S = nB * Pm = 7 * (10, 9) = (70, 63).

B calculates the multiplicative inverse of S, denoted as S^(-1).

B recovers the plaintext Pm = Cm - k * PB = (16, 30) - 3 * (14, 49) = (16, 30) - (42, 147) = (16, 30) - (42 mod 1759, 147 mod 1759).

25. RSA vs. ECC:

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.

The choice between RSA and Elliptic Curve Cryptography (ECC) depends on the specific requirements and constraints of the application:

RSA:

Advantages:

Well-established and widely used.

Strong security with sufficiently long key lengths.

Disadvantages:

Requires longer key lengths for the same level of security compared to ECC.

Slower encryption and decryption, especially for large key lengths.

Larger storage and bandwidth requirements.

ECC:

Advantages:

Provides strong security with shorter key lengths, making it more efficient.

Faster encryption and decryption, making it suitable for resource-constrained devices.

Smaller storage and bandwidth requirements.

Disadvantages:

Relatively newer and less widely used than RSA (although gaining popularity).

Considerations:

For resource-constrained devices or scenarios where efficiency is crucial, ECC is a preferred choice due to its shorter key lengths and faster performance.

For applications with established RSA infrastructure and less concern for computational efficiency, RSA remains a robust choice.

26. a) Identify the security service(s) offered by the models described in

i. FIGURE 1 ii. FIGURE2 iii. FIGURE 3

b) Give suggestions to improve the cryptography model described in FIGURE 3 so

that it is resistant to release of message content attack.

Security Services and Suggestions for FIGURE 3:

a) FIGURE 1:

Confidentiality: Ensures that the message content is kept secret from unauthorized users.

Data Integrity: Ensures that the message content is not altered during transmission.

Authenticity: Verifies the identity of the sender.

b) FIGURE 2:

Confidentiality: Ensures that the message content is kept secret from unauthorized users.

Data Integrity: Ensures that the message content is not altered during transmission.

Authenticity: Verifies the identity of the sender.

c) FIGURE 3:

Confidentiality: Ensures that the message content is kept secret from unauthorized users.

Authenticity: Verifies the identity of the sender.

Suggestions for FIGURE 3:

To make FIGURE 3 resistant to release of message content attacks, the following enhancements can be considered:

Use encryption mechanisms like symmetric key encryption or public-key encryption to ensure the confidentiality of the message content.

Implement digital signatures or message authentication codes (MACs) to provide data integrity and authenticity.

Use secure key management practices to protect the keys used for encryption and authentication.

Implement access control and authorization mechanisms to restrict access to the message content.

Ensure secure transmission and storage of encrypted messages to prevent unauthorized access.

These suggestions would enhance the overall security of FIGURE 3 by addressing the security services required for confidentiality, data integrity, and authenticity.