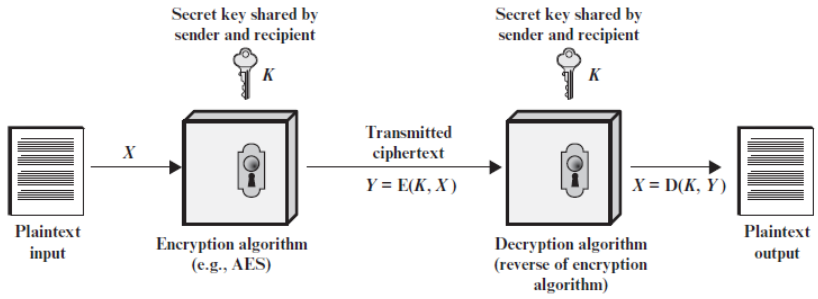
USN										

Internal Assessment Test 1 – Sept. 2025 ANSWER KEY

Sl.	Answer any FIVE FULL Questions	Marks	CO	RBT
1	a) **Explain the symmetric cipher model** b) **Explain one time pad**	5+5	CO1	L2

a)



Symmetric Encryption Scheme –5 Key Ingredients

PlaintextThe original readable message or data.

Encryption Algorithm Performs substitutions and transformations on the plaintext using a key.

Secret KeyA value used by the algorithm; **independent of the message**.

Different keys result in different ciphertexts.

CiphertextThe scrambled, unintelligible output generated from plaintext + key.

Decryption Algorithm Reverses the encryption process using the same key to recover the original plaintext.

b) **One-Time Pad (OTP)**

The *One-Time Pad (OTP)* is a **perfectly secure symmetric encryption technique** in which a **random key** (called the pad) is used **only once** to encrypt and decrypt a message.

Working Principle:

1. The **plaintext** is converted into binary or numeric form.
2. A **random key** of the *same length* as the plaintext is generated.
3. **Encryption** is done using bitwise **XOR (⊕)** operation between the plaintext and key:

C=P⊕KC = P \oplus KC=P⊕K

where

- CCC = Ciphertext
- PPP = Plaintext
- KKK = Key

4. **Decryption** is done by applying XOR again with the same key:

P=C⊕KP = C \oplus KP=C⊕K

	Example:
	Plaintext: **HELLO**
	Convert to binary and use a random key of equal length.
	When XOR is performed bitwise, you get ciphertext.
	Using the same key again decrypts it back to **HELLO**.
	Features / Advantages:
	• Provides **absolute (theoretical) security** if:
	  ○ The key is **truly random**.
	  ○ The key is **as long as the message**.
	  ○ The key is **never reused**.
	  ○ The key is **kept completely secret**.
	Limitations:
	• **Key distribution** is difficult (key must be shared securely).
	• **Key management** is complex since keys cannot be reused.
	• Impractical for long messages or frequent communication.			

2	**Explain Play Fair Cipher & Apply its rules for the following Key = MONARCHY Plaintext = CRYPTOGRAPHY to get the cipher text**	10	CO1	L3
	Definition The **Playfair cipher** is a **digraph substitution** technique. It encrypts pairs of letters using a 5×5 key square built from a keyword (I/J combined). Working on pairs hides single-letter frequency, making it stronger than simple monoalphabetic ciphers. **Constructing the 5×5 Key Square** • Write the **key**; remove duplicate letters. • Fill a 5×5 grid row-wise with key letters, then the rest of the alphabet (merge **I/J**; usually treat J as I). **Key = MONARCHY → unique**: M O N A R C H Y **Alphabet (I/J merged)**: A B C D E F G H I K L M N O P Q R S T U V W X Y Z **Key square:** M O N A R C H Y B D E F G I K L P Q S T U V W X Z **Preparing Plaintext** Rules: • Remove spaces/punctuation; change **J→I**. • Split into **digraphs** (pairs). • If a pair has **double letters**, insert **X** between them. • If odd length, pad a final **X**. **Plaintext = CRYPTOGRAPHY** Pairs: **CR | YP | TO | GR | AP | HY** (no doubles; even length) **Encryption Rules** For each pair (A,B): 1. **Same row**: replace each with the letter **to its right** (wrap at end). 2. **Same column**: replace each with the letter **below** it (wrap at bottom). 3. **Rectangle**: replace each with the letter in the **same row** but the **column of the other** (i.e., take the rectangle corners). **Worked Example (step-by-step)** Using the square above: • **CR**: C(1,0), R(0,4) → rectangle → (1,4)=**D**, (0,0)=**M** → **DM** • **YP**: Y(1,2), P(3,1) → rectangle → (1,1)=**H**, (3,2)=**Q** → **HQ** • **TO**: T(3,4), O(0,1) → rectangle → (3,1)=**P**, (0,4)=**R** → **PR** • **GR**: G(2,2), R(0,4) → rectangle → (2,4)=**K**, (0,2)=**N** → **KN** • **AP**: A(0,3), P(3,1) → rectangle → (0,1)=**O**, (3,3)=**S** → **OS** • **HY**: H(1,1), Y(1,2) → same row → right → **Y**, **B** → **YB**			

	Ciphertext Concatenate: **DMHQPRKNOSYB**			
3	**Explain Feistel structures for encryption and decryption**	10	CO1	L2

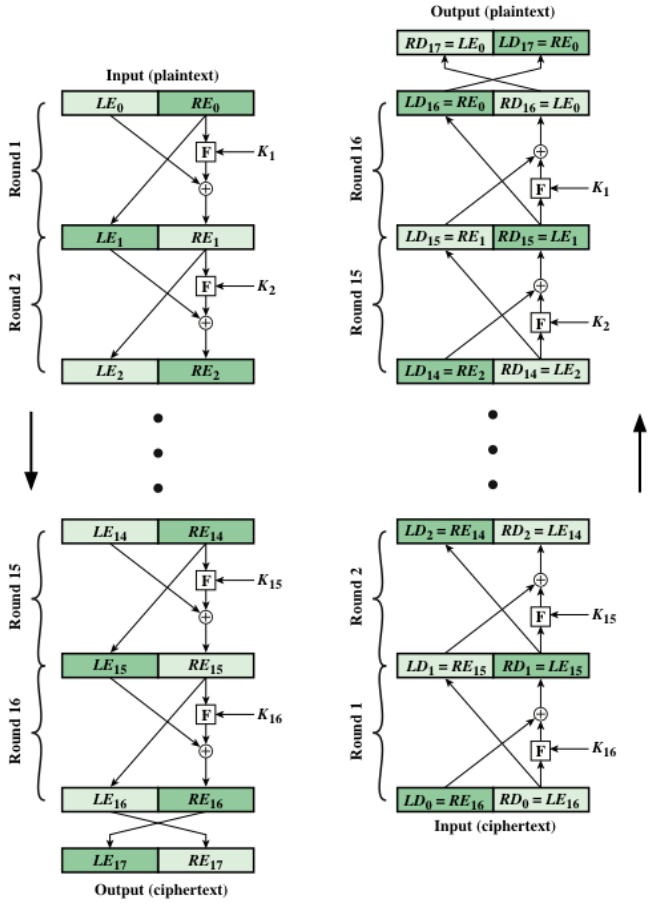


Figure 4.3 Feistel Encryption and Decryption (16 rounds)

The **Feistel structure** is the **fundamental design model** used in many block ciphers like **DES (Data Encryption Standard)**.
It allows the same algorithm to be used for both **encryption and decryption**, making implementation simple and efficient.

- It is a **symmetric block cipher** structure.
- The plaintext block is divided into two equal halves:
  - **Left half (L_0)**
  - **Right half (R_0)**
- The cipher operates in **'n' rounds** using **subkeys (K_1, K_2, …, K_n)** derived from the main key.

Feistel Encryption Process

Let the plaintext block be divided into L0L_0L0 and R0R_0R0.
For each round **i ($1 \le i \le n$):**

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

Where:

- F = **round function** (a complex function involving substitution, permutation, etc.)
- Ki = **subkey** for round *i*
- $\oplus$ = bitwise XOR operation

After the last round (n rounds):

- The final output is **(R_n, L_n)** — sometimes swapped to maintain symmetry.

Feistel Decryption Process

Decryption follows the **same structure** as encryption, except the **subkeys are used in reverse order**.

For each round **i (n → 1):**

$$Ri-1=Li$$
$$Li-1=Ri \oplus F(Li, Ki)$$

Since XOR is reversible, and the structure is symmetric, the same algorithm works both ways.

Key Characteristics

Feature	Description
Symmetric Design	Same process for encryption and decryption (onl
Block Structure	Operates on fixed-size blocks (e.g., 64-bit in DE
Round Function F	Provides confusion and diffusion.
Flexibility	Any function F can be used as long as the struct
Security	Depends on number of rounds and complexity o

4	**Explain RSA Algorithm. Perform encryption and decryption using RSA algorithm with P=3, Q=11, C=3 and M=9.**	10	CO2	L3

RSA is an asymmetric (public-key) cryptosystem based on the hardness of factoring a large composite number n=pq

- **Public key:** (n,e)
- **Private key:** (n,d) with $d \equiv e-1 \pmod{\phi(n)}$

Encryption: $C \equiv M^e \pmod n$

Decryption: $M \equiv C^d \pmod n$

Key Generation (

1. Choose two primes p,q
2. Compute n=pq
3. Compute Euler's totient: $\phi(n)=(p-1)(q-1)$
4. Choose public exponent e such that $1<e<\phi(n)1$ and $\gcd(e,\phi(n))=1$
5. Compute the private exponent d as the modular inverse: $d \equiv e^{-1} \pmod{\phi(n)}$

Worked Example

Given: p=3, q=11, e=3 (they've called it C=3C=3C=3 in the question, but this is the **public exponent**), and plaintext M=9

a) Compute n and ϕ(n)

- $n=pq=3\times11=33$
- $\phi(n)=(p-1)(q-1)=2\times10=20$

b) Check e and find d

- $\gcd(e,\phi(n))=\gcd(3,20)=1$ ✓ valid
- Find d such that $3d \equiv 1 \pmod{20}$
  Try d=7: $3\times7=21 \equiv 1 \pmod{20} \rightarrow$ **d=7**

Public key: (n,e)=(33,3)

Private key: (n,d)=(33,7)

c) Encrypt M=9

$C \equiv M^e \pmod n = 9^3 \bmod 33$

- $9^2=81$
- $9^3=81\times9=729$
- 729 mod 33: $33\times22=726$
- remainder $729-726 \Rightarrow C=3$

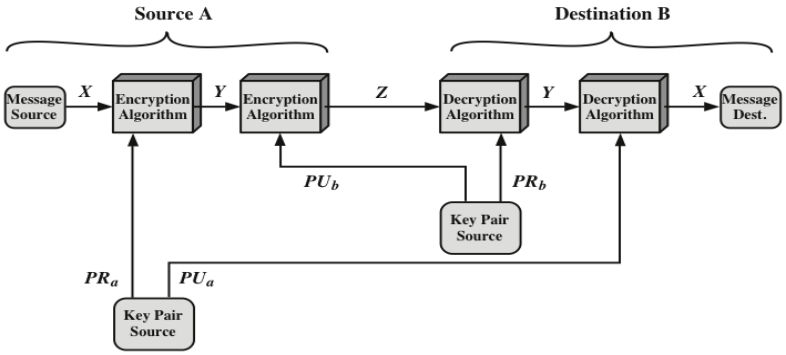
d) Decrypt C=3

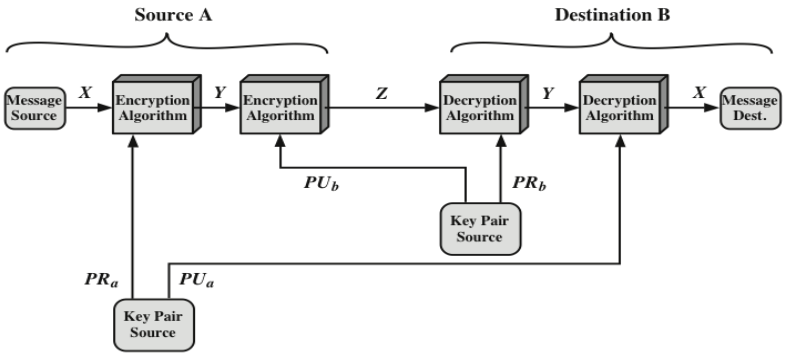
$M \equiv C^d \pmod n = 3^7 \bmod 33$ M \equiv C^{d} \pmod{n} = 3^{7} \bmod 33 $M \equiv Cd \pmod n = 37 \bmod 33$

Compute stepwise (mod 33):

M=9 (original message recovered)

	Final Answer • **Ciphertext:** C=3C=3C=3 • **Decrypted Plaintext:** M=9M=9M=9			
5	**Explain Diffie Hellman key exchange algorithm with example**	10	CO2	L2

	
	
	Figure 9.4 Public-Key Cryptosystem: Authentication and Secrecy
	The **Diffie–Hellman (DH)** algorithm is a **key exchange technique** used to securely share a **secret key** between two parties (say, Alice and Bob) over an **insecure communication channel**.
	It was the **first practical public-key exchange algorithm**, proposed by **Whitfield Diffie and Martin Hellman in 1976**.
	Purpose
	It allows two users to **generate a shared secret key** that can then be used for **symmetric encryption** (e.g., AES, DES) — without ever directly transmitting the key.
	Basic Idea
	• Both users agree on two **public values**:
	  ○ A **prime number** p
	  ○ A **primitive root** g (or generator) modulo p
	• Each user selects a **private (secret) key**, computes a **public key** using g and p, and then exchanges it.
	• Using the received public key, both compute the **same shared secret key** independently.
	Mathematical Steps
	Step 1: Publicly agree on:
	Prime number p and primitive root g
	Step 2: Each selects private key:
	• Alice chooses a private key a
	• Bob chooses a private key b
	Step 3: Compute public keys:
	$A=g^a \bmod p$
	$B=g^b \bmod p$
	Step 4: Exchange public keys (A and B).
	Step 5: Compute shared secret key:
	• Alice computes: $K=B^a \bmod p$
	• Bob computes: $K=A^b \bmod p$
	Both get the same key since:
	$B^a \bmod p=(g^b)^a \bmod p=g^{ab} \bmod p=g^{ba} \bmod p=(g^a)^b \bmod p=A^b \bmod$
	
	Example
	Let:
	• Publicly known values: p=23 g=5
	Step 1: Choose private keys
	• Alice's private key: a=6
	• Bob's private key: b=15			

	Step 2: Compute public keys $A=g^a \bmod p$ $=5^6 \bmod 23$ → **A = 8** $B=g^b \bmod p=$ $5^{15} \bmod 23$ → **B = 19** **Step 3: Exchange public keys (A=8, B=19)** **Step 4: Compute shared secret key** • **Alice computes:** $K=B^a \bmod p = 19^6 \bmod 23$ → **K = 2** • **Bob computes:** $K=A^b \bmod p=8^{15} \bmod 23$ → **K = 2** **Shared Secret Key = 2** **Advantages** • Securely establishes a secret key over a public network. • Key is never transmitted directly. **Limitation** • Vulnerable to **Man-in-the-Middle (MITM)** attack if authentication is not used.			
6	**Explain how authentication and confidentiality is ensured in public key cryptosystem**	10	CO2	L2
	 **Figure 9.4 Public-Key Cryptosystem: Authentication and Secrecy** A **Public Key Cryptosystem (PKC)** uses a **pair of keys**: • A **public key** (shared openly) • A **private key** (kept secret) It provides two major security services: • **Confidentiality** – ensuring that only the intended receiver can read the message. • **Authentication** – ensuring that the sender's identity is genuine. Examples of PKC: **RSA**, **Diffie–Hellman**, **ECC**, etc. **Confidentiality** **Definition:** Confidentiality ensures that the **message content remains secret** and is not disclosed to unauthorized users. **How It Works in PKC:** 1. The **sender encrypts** the plaintext using the **receiver's public key**. $C=E_{KUB}(M)$ where E_{KUB} is the receiver's public key. 2. Only the **receiver**, who has the corresponding **private key** K_{RB}, can decrypt it: $M=D_{KRB}(C)$			

3. Since only the receiver knows their private key, **no one else can decrypt the message → confidentiality achieved**.

Example:

- Alice wants to send a secret message to Bob.
- Bob's public key = K_{UB}, private key = K_{RB}.
- Alice encrypts message M using K_{UB}.
- Bob decrypts it using K_{RB}.
- Even if an attacker intercepts C, it cannot be decrypted without Bob's private key.

Authentication

Definition:

Authentication ensures that the **sender is who they claim to be** and the message is not forged.

How It Works in PKC:

1. The **sender encrypts** the message (or its hash) using their **own private key**:
   $C=E_{KRA}(M)$; K_{RA} is Alice's private key.
2. The **receiver decrypts** the ciphertext using the **sender's public key**:
   $M=D_{KUA}(C)$
3. If decryption is successful and produces a valid message, it proves that:
   - The message was encrypted using **Alice's private key**.
   - Therefore, it must have come from **Alice → Authentication achieved**.

Example:

- Alice signs a message using her **private key**.
- Bob verifies it using Alice's **public key**.
- Since only Alice possesses her private key, Bob is assured that the message is genuinely from Alice.

4) Combining Both: Authentication + Confidentiality

Both services can be combined:

1. **Step 1 – Authentication:** Sender (Alice) encrypts message with her **private key** → ensures identity.
2. **Step 2 – Confidentiality:** Then encrypts the result with the **receiver's public key** → ensures secrecy.

At receiver's end:

1. Receiver decrypts with **own private key** K_{RB}.
2. Then decrypts with **sender's public key** K_{UA}
   $M=D_{KUA(}D_{KRB}(C))$

This guarantees **both**:

- Message **came from Alice** (authentication)
- Message **can be read only by Bob** (confidentiality)

Conclusion

Public Key Cryptography provides:

- **Confidentiality** → using **receiver's public key**
- **Authentication** → using **sender's private key**

  By combining both encryption steps, secure and authenticated communication is achieved.