Data Security And Privacy ANSWERS

1)Explain Playfair cipher algorithm. Find the ciphertext for plaintext =” INSTRUMENTS” with key =” MONARCHY”.

A) **1) Quick description**

Playfair is a digraphic substitution cipher that encrypts pairs of letters (digraphs) using a 5×5 key square. It was invented in the 19th century and is stronger than simple monoalphabetic ciphers because it encrypts letter *pairs*, introducing dependency between adjacent letters. For modern security it is **not** secure (vulnerable to frequency analysis of digraphs and known-plaintext attacks), but it’s a useful historical/classroom cipher for learning classical cryptography.

Basic rules (common conventions used below):

- Combine I and J into a single cell (treat J as I).
- Build a 5×5 key square from the key (left-to-right, top-to-bottom) using only unique letters, then fill with remaining alphabet letters (skipping J).
- Prepare plaintext: remove spaces, convert J→I, form digraphs; if a pair has identical letters insert an X between them; if final letter is single, append X.
- For each digraph:
    1. If both letters are in the same row: replace each by the letter to its right (wrap to start).
    2. If both letters are in the same column: replace each by the letter below it (wrap to top).
    3. Otherwise (rectangle): each letter is replaced by the letter in the same row but in the column of the other letter (i.e., corner swap).

2) Key square construction (key = MONARCHY)

Key: M O N A R C H Y

Remove duplicates, then append remaining letters of alphabet (combine I/J). The completed 5×5 square:

mathematica Copy code

```
Row\Col  0  1  2  3  4
0        M  O  N  A  R
1        C  H  Y  B  D
2        E  F  G  I  K
3        L  P  Q  S  T
4        U  V  W  X  Z
```

(We used the convention I/J combined and placed `I` in the square.)

3) Plaintext preprocessing

Plaintext: `INSTRUMENTS`

Convert J→I (none), remove spaces → `INSTRUMENTS`.

Split into digraphs, inserting X if needed and padding final single letter:

`IN | ST | RU | ME | NT | S` → last `S` is single → append `X` → `SX`

Final digraphs: **IN, ST, RU, ME, NT, SX**

---

4) Encrypt each digraph (using coordinates from the key square)

Coordinates (row, col):

- M(0,0) O(0,1) N(0,2) A(0,3) R(0,4)
- C(1,0) H(1,1) Y(1,2) B(1,3) D(1,4)
- E(2,0) F(2,1) G(2,2) I(2,3) K(2,4)
- L(3,0) P(3,1) Q(3,2) S(3,3) T(3,4)
- U(4,0) V(4,1) W(4,2) X(4,3) Z(4,4)

Now apply rules:

1. **IN**: I = (2,3), N = (0,2) → rectangle → I → (2,2) = **G**, N → (0,3) = **A**
   → **GA**
2. **ST**: S = (3,3), T = (3,4) → same row → replace each by letter to right (wrap): S→T, T→L
   → **TL**
3. **RU**: R = (0,4), U = (4,0) → rectangle → R→(0,0)=**M**, U→(4,4)=**Z**
   → **MZ**
4. **ME**: M = (0,0), E = (2,0) → same column → replace each by letter below: M→C, E→L

2)Encrypt the plaintext “CRYPTOGRAPHY” using Hill Cipher algorithm with key and decrypt the same.

[9 4 5 7]

A)

Step 1: Preprocess the plaintext

Plaintext: **CRYPTOGRAPHY**

1. Convert to numbers using A=0, B=1, ..., Z=25.

   C(2) R(17) Y(24) P(15) T(19) O(14) G(6) R(17) A(0) P(15) H(7) Y(24)

So numeric plaintext sequence:

[2, 17, 24, 15, 19, 14, 6, 17, 0, 15, 7, 24]

2. Group into column vectors of size 2 (since 2×2 key):

$$P_1 = \begin{bmatrix} 2 \\ 17 \end{bmatrix}, P_2 = \begin{bmatrix} 24 \\ 15 \end{bmatrix}, P_3 = \begin{bmatrix} 19 \\ 14 \end{bmatrix}, P_4 = \begin{bmatrix} 6 \\ 17 \end{bmatrix}, P_5 = \begin{bmatrix} 0 \\ 15 \end{bmatrix}, P_6 = \begin{bmatrix} 7 \\ 24 \end{bmatrix}$$

Step 2: Encryption

Encryption rule (mod 26):

$$C_i = K \cdot P_i \pmod{26}$$

Work through each pair

Key:

$$K = \begin{bmatrix} 9 & 4 \\ 5 & 7 \end{bmatrix}$$

Pair 1: $P_1 = [2, 17]^T$

$$C_1 = \begin{bmatrix} 9 & 4 \\ 5 & 7 \end{bmatrix} \begin{bmatrix} 2 \\ 17 \end{bmatrix} = \begin{bmatrix} 9*2+4*17 \\ 5*2+7*17 \end{bmatrix} = \begin{bmatrix} 18+68 \\ 10+119 \end{bmatrix} = \begin{bmatrix} 86 \\ 129 \end{bmatrix} \pmod{26}$$

Compute mod 26:

- 86 ÷ 26 = 3 remainder **8**
- 129 ÷ 26 = 4 remainder **25**

3)Explain Fiestel encryption and decryption algorithm, with neat diagram.

A) The **Feistel network** (or Feistel cipher) is the building block for many symmetric block ciphers such as **DES, Blowfish, and Camellia**. Its key feature is that **encryption and decryption use almost the same algorithm** — only the order of subkeys is reversed during decryption.

3) Neat Diagram of Feistel Cipher

Here's a simple clean diagram you can draw in exams:

```mathematica
             ┌─────────────┐
Plaintext →  │ Split into  │ → L0, R0
             │ L0 and R0   │
             └─────────────┘
                   │
          ┌──────────┴──────────┐
          │       Round 1       │
          │  L1 = R0            │
          │  R1 = L0 ⊕ F(R0,K1) │
          └──────────┬──────────┘
                     │
          ┌──────────┴──────────┐
          │       Round 2       │
          │  L2 = R1            │
          │  R2 = L1 ⊕ F(R1,K2) │
          └──────────┬──────────┘
                     │
                  ......
                     │
          ┌──────────┴──────────┐
          │      Round n        │
          │  Ln = Rn-1          │
          │  Rn = Ln-1⊕F(Rn-1,Kn)│
          └──────────┬──────────┘
                     │
          ┌──────────┴──────────┐
          │    Combine Ln,Rn    │
          │     → Ciphertext    │
          └─────────────────────┘
```

Encryption Algorithm (Step-by-Step)

1. **Input:** Plaintext block (P), Key (K)
2. **Divide:** P → L_0, R_0
3. **Key Scheduling:** Generate round keys $K_1, K_2, \ldots, K_n$
4. **Repeat for i = 1 to n:**
   - $L_i = R_{i-1}$
   - $R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$
5. **Output:** Ciphertext = $L_n || R_n$

Decryption Algorithm

Decryption simply uses the **same structure** but applies the round keys in **reverse order**:

1. **Input:** Ciphertext block L_n, R_n
2. **For i = n down to 1:**
   - $R_{i-1} = L_i$
   - $L_{i-1} = R_i \oplus F(L_i, K_i)$
3. **Output:** Plaintext = $L_0 || R_0$

4)Explain with neat diagram DES encryption and decryption algorithm.

A)

The algorithm itself is referred to as the Data Encryption Algorithm (DEA). For DES, data are encrypted in **64-bit blocks using a 56-bit key.** The algorithm transforms 64-bit input in a series of steps into a 64-bit output. The same steps, with the same key, are used to reverse the encryption.

DES Encryption

As with any encryption scheme, there are two inputs to the encryption function: the **plaintext** to be encrypted and **the key.** In this case, the plaintext must be 64 bits in length and the key is 56 bits in length.

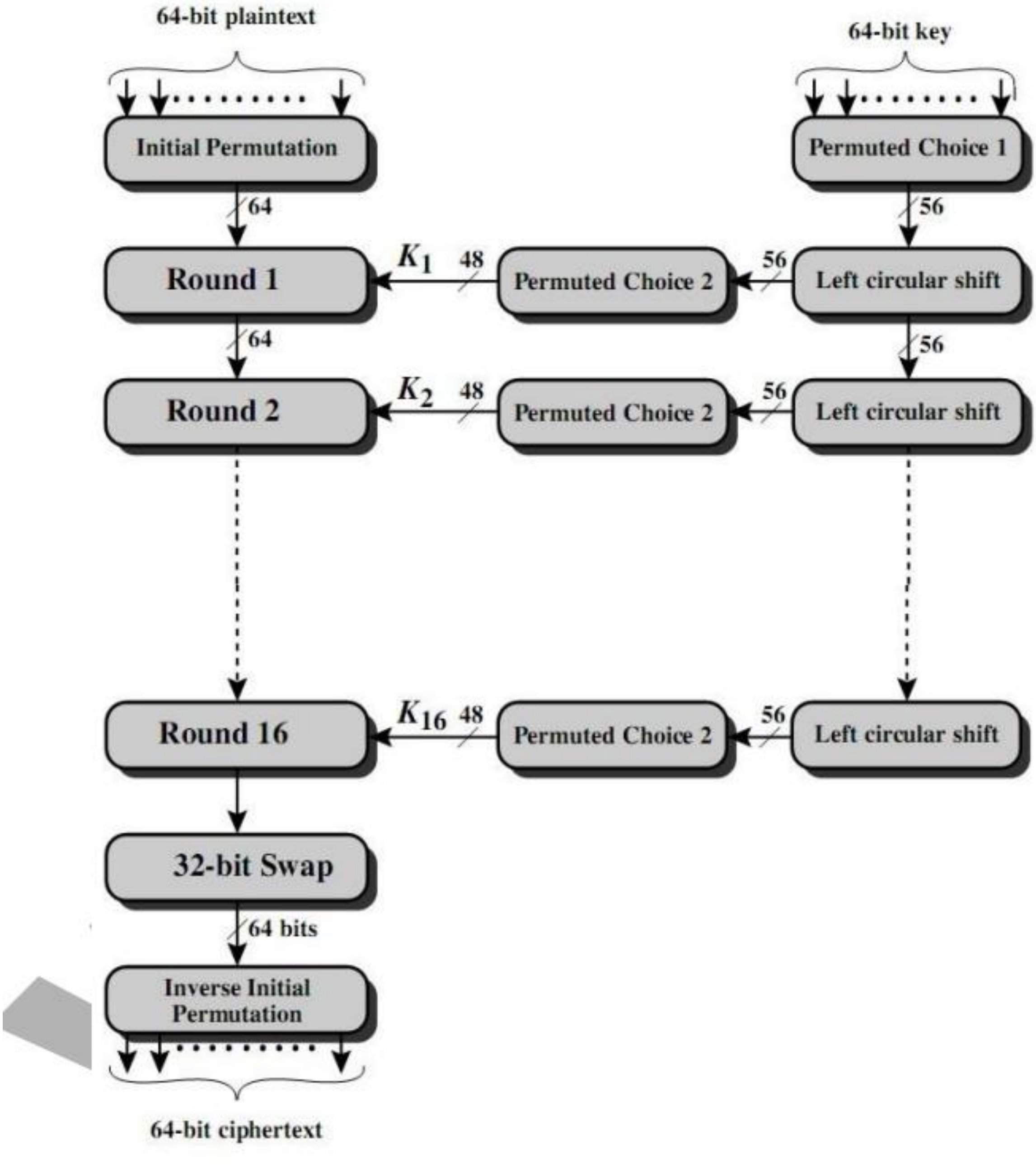

Figure 3.4 General Depiction of DES Encryption Algorithm

5) Explain RSA Algorithm operation in detail. Perform an encryption of plain text and decryption of cipher text using RSA algorithm for p=3, q=11, e=7 and M=5.

A)

1) RSA — high-level idea

RSA is an asymmetric (public-key) cryptosystem. Each user has:

- a **public key** (n, e) used to **encrypt** messages, and
- a **private key** d used to **decrypt**.

Security relies on difficulty of factoring $n = p \cdot q$ when p, q are large primes. The math uses modular exponentiation and Euler's theorem.

2) RSA steps (in detail)

Key generation

1. Choose two distinct large primes p and q.
2. Compute $n = p \cdot q$. n is modulus for both keys.
3. Compute Euler's totient $\varphi(n) = (p-1)(q-1)$.
4. Choose integer e such that $1 < e < \varphi(n)$ and $\gcd(e, \varphi(n)) = 1$. e is the public exponent.
5. Compute the modular inverse d of e modulo $\varphi(n)$, i.e. find d such that $e \cdot d \equiv 1 \pmod{\varphi(n)}$. d is the private exponent.

Public key: (n, e).
Private key: d (and keep p, q secret).

Encryption: Given plaintext numeric message M with $0 \le M < n$, compute

$$C \equiv M^e \pmod{n}.$$

Decryption: Given ciphertext C, compute

$$M \equiv C^d \pmod{n}.$$

3) Worked example (given values)

Given: $p = 3,\ q = 11,\ e = 7,\ M = 5.$

Step A — compute n and φ(n):

- $n = p \cdot q = 3 \cdot 11 = 33.$
- $\varphi(n) = (p-1)(q-1) = 2 \cdot 10 = 20.$

Step B — check e and compute d:

- $e = 7$. Check $\gcd(7, 20) = 1$ → ok.
- Find d such that $7d \equiv 1 \pmod{20}$.

Solve: $7 \cdot 3 = 21 \equiv 1 \pmod{20}$.
So $d = 3$.

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

Step C — Encryption

$$C \equiv M^e \pmod{n} = 5^7 \bmod 33.$$

Compute modularly:

- $5^2 = 25.$
- $5^4 = 25^2 = 625 \equiv 625 - 33 \cdot 18 = 625 - 594 = 31 \pmod{33}.$
- $5^7 = 5^4 \cdot 5^2 \cdot 5 = 31 \cdot 25 \cdot 5.$
  - $31 \cdot 25 = 775 \equiv 775 - 33 \cdot 23 = 775 - 759 = 16.$
  - $16 \cdot 5 = 80 \equiv 80 - 33 \cdot 2 = 80 - 66 = 14.$

So $C = 14$.

Ciphertext = 14.

Step D — Decryption

$$M' \equiv C^d \pmod{n} = 14^3 \bmod 33.$$

Compute:

- $14^2 = 196 \equiv 196 - 33 \cdot 5 = 196 - 165 = 31.$
- $14^3 = 31 \cdot 14 = 434 \equiv 434 - 33 \cdot 13 = 434 - 429 = 5.$

So $M' = 5$, which matches the original message.

4) Final results (concise)

- $n = 33,\ \varphi(n) = 20.$
- Public key: $(33, 7)$. Private key: $d = 3$.
- Encryption of $M = 5$: $C = 14$.
- Decryption of $C = 14$: $M' = 5$ (original recovered).

Secrecy (Confidentiality)

Goal: **Only receiver should be able to read the message.**

- Sender **encrypts message using receiver's public key (PU_B)**.
- Only receiver can decrypt it using **their private key (PR_B)**.
- Even if attackers intercept ciphertext, they cannot decrypt because they don't know PR_B.

3) Authentication (and Integrity)

Goal: **Receiver should be sure about the identity of the sender.**

- Sender **signs message using their own private key (PR_A)**.
- Receiver **verifies signature using sender's public key (PU_A)**.
- If verification is successful, it confirms the message really came from sender (authentication) and is unchanged (integrity).

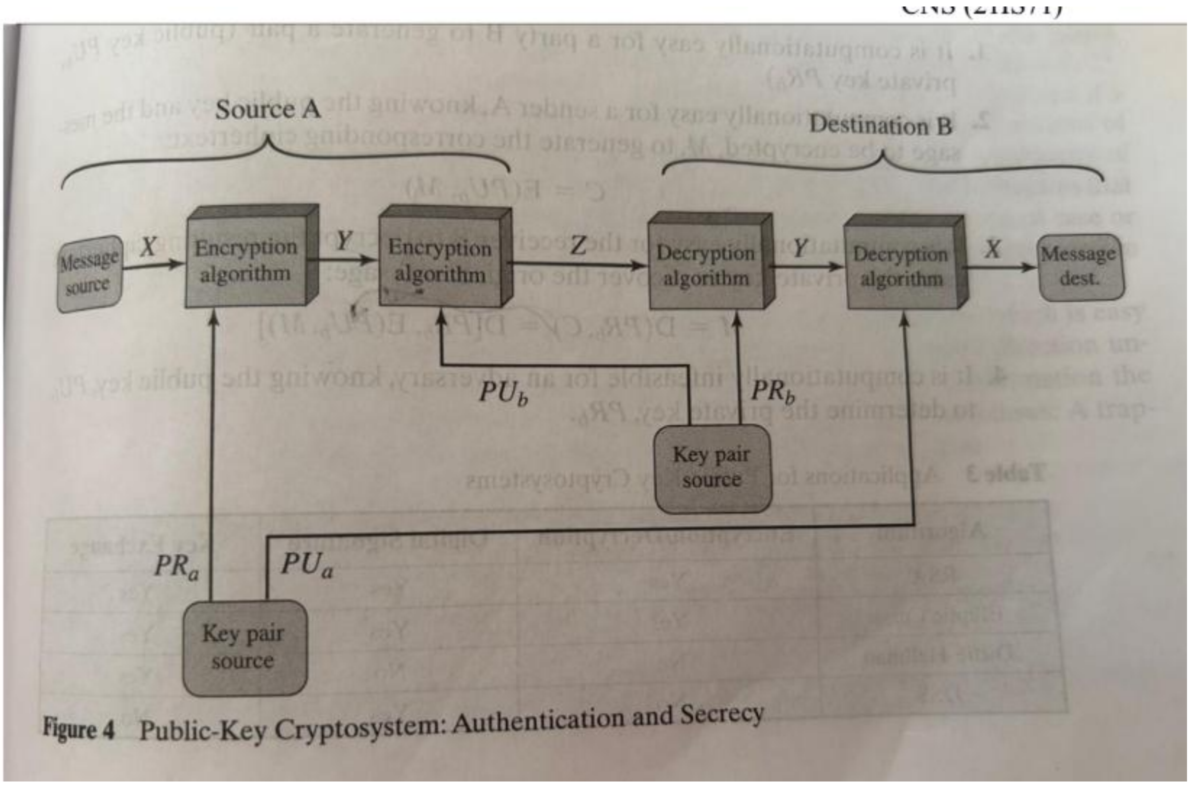

Figure 4 Public-Key Cryptosystem: Authentication and Secrecy

6)Explain Symmetric key distribution using Symmetric encryption.

A)Symmetric Key Distribution using Symmetric Encryption

In symmetric cryptography, the **same secret key** is used for both encryption and decryption.
The challenge is **securely distributing this key** over an insecure network.
Solution: Use a **Key Distribution Center (KDC)** with symmetric encryption.

Explanation

- Each user has a **long-term master key** shared only with the trusted KDC.
- When two users (A and B) wish to communicate securely, the KDC:
    - **Generates a fresh session key (K_AB)**.
    - Sends this session key to A and B, **encrypted under their respective master keys** so that only they can decrypt it.
- Both A and B now share **K_AB**, which they use for **secure encryption and decryption** of all messages during that session.
- After the session ends, the key is discarded — a **new key is generated for the next session**.

7) Explain:

i) Hierarchical key control

ii) Session key lifetime

iii) A transparent key control scheme

iv) Decentralized key control

A) i) Hierarchical Key Control

- **In hierarchical key control, keys are organized in layers (hierarchy).**
- **A Key Distribution Center (KDC) manages keys for a group of users.**
- **Structure:**
    - **Master Key: Shared between each user and KDC (long-term).**
    - **Session Key: Generated for communication between users and distributed by KDC.**
- **The hierarchy allows a single KDC to manage many users and multiple session keys at once.**

---

ii) Session Key Lifetime

- **Definition: The period of time during which a session key is valid and used for encryption/decryption.**
- **A short session key lifetime increases security because:**
    - **Limits damage if the key is compromised.**
    - **Reduces risk of cryptanalysis attacks on repeated ciphertext.**
- **Long session key lifetime can be efficient (less frequent key generation) but less secure.**

---

iii) Transparent Key Control Scheme

- **In a transparent key control scheme, the user does not manually manage keys — the key management process is invisible to end users.**
- **Keys are automatically:**
  - **Generated**
  - **Distributed**
  - **Installed**
  - **Destroyed**
- **The user just encrypts/decrypts data without worrying about key generation or exchange.**

---

iv) Decentralized Key Control

- **In decentralized key control, there is no single KDC for the entire network.**
- **Each local node or group has its own Key Distribution Center (local KDC).**
- **Users within a group get keys from their local KDC.**
- **For inter-group communication, local KDCs coordinate and exchange keys.**