BCU BCA 5TH SEM — CRYPTOGRAPHY AND NETWORK SECURITY SOLVED QUESTION BANK

Solved Question Bank by Nikhil P

PART I: SECTION A — 2 MARKS QUESTIONS (PAPERS 1 TO 5)

MODEL QUESTION PAPER – 1

Q1. Define cryptography.

Formal Definition: Cryptography is the science and art of transforming intelligible messages into unintelligible forms and vice-versa, guaranteeing secure communication over insecure channels against adversaries (Stallings/Forouzan).
Key Components / Formula / Mechanism: Comprises Plaintext (P), Ciphertext (C), Secret Key (K), encryption function C = E_K(P), and decryption function P = D_K(C), ensuring confidentiality and integrity.
Example / Use-Case: Transport Layer Security (TLS 1.3) protocol and Advanced Encryption Standard (AES).

Q2. List the three basic security goals of information security.

Formal Definition: The three foundational security goals of information security—collectively termed the CIA Triad—are Confidentiality, Integrity, and Availability, designed to safeguard information assets against unauthorized threats.
Key Components / Formula / Mechanism: Confidentiality prevents unauthorized information disclosure (via encryption); Integrity guarantees protection against unauthorized modification (via hashes/MACs); Availability ensures authorized users uninterrupted access to system resources.
Example / Use-Case: Online banking using AES encryption, SHA-256 integrity checks, and redundant server clusters.

Q3. Differentiate between a threat and a security attack.

Formal Definition: A threat is a potential security violation where a circumstance, capability, or event could cause harm, whereas a security attack is an intentional action executed to exploit a vulnerability (Stallings / RFC 4949).
Key Components / Formula / Mechanism: A threat is a latent danger or capability; an attack is the active or passive operational execution that exploits that vulnerability.
Example / Use-Case: An unencrypted Wi-Fi link is a threat; sniffing credentials via Wireshark is an attack.

Q4. Define symmetric-key cryptography.

Formal Definition: Symmetric-key cryptography (secret-key cryptography) is an encryption scheme where communicating parties share a single identical secret key for both plaintext encryption and ciphertext decryption (Forouzan).
Key Components / Formula / Mechanism: Operates via C = E_K(P) and P = D_K(C). For n communicating parties, n(n - 1) / 2 shared keys are required, requiring secure key distribution.
Example / Use-Case: AES-256 for full-disk encryption (BitLocker) and DES/3DES in legacy payment systems.

Q5. State Kerckhoffs' Principle.

Formal Definition: Kerckhoffs' Principle states that a cryptographic system must remain secure even if everything about the system—including algorithm design and source code—is public knowledge, provided only the key is kept secret.
Key Components / Formula / Mechanism: Security depends solely on key secrecy, length, and randomness, explicitly rejecting "security through obscurity" and enabling rigorous open peer review.
Example / Use-Case: Public cryptographic standards such as AES and RSA, whose mathematical designs are public.

Q6. What is a substitution cipher?

Formal Definition: A substitution cipher is an encryption method where plaintext characters, bits, or symbol groups are systematically replaced with ciphertext characters according to a fixed translation key.
Key Components / Formula / Mechanism: Includes monoalphabetic (fixed mapping, e.g., shift C = (P + k) mod 26) and polyalphabetic (multiple mappings across positions, e.g., Vigenère).
Example / Use-Case: Caesar cipher (monoalphabetic), Playfair cipher (digraphic), and modern AES S-boxes.

Q7. List the five modes of operation of modern block ciphers.

Formal Definition: Block cipher modes of operation are standardized procedures (NIST SP 800-38A) for applying a block cipher algorithm with a symmetric key to securely encrypt arbitrary-length plaintext.
Key Components / Formula / Mechanism: 1. Electronic Codebook (ECB), 2. Cipher Block Chaining (CBC), 3. Cipher Feedback (CFB), 4. Output Feedback (OFB), and 5. Counter (CTR) mode.
Example / Use-Case: CBC mode used in IPsec VPN tunnels and CTR mode used in high-speed TLS 1.3.

Q8. Define asymmetric-key cryptography.

Formal Definition: Asymmetric-key (public-key) cryptography is an encryption system using mathematically linked key pairs: a public key freely distributed for encryption/verification and a private key kept secret for decryption/signing (Stallings).
Key Components / Formula / Mechanism: Based on one-way trapdoor functions (e.g., integer factorization). Requires only 2n keys for n entities. Operates via C = E_Kpub(P) and P = D_Kpriv(C).
Example / Use-Case: RSA and ECC (Elliptic Curve Cryptography) used in TLS handshakes and SSH.

Q9. What is a digital signature?

Formal Definition: A digital signature is a mathematical cryptographic scheme that binds an electronic document or message to the identity of the signer, providing data authentication, integrity, and non-repudiation.
Key Components / Formula / Mechanism: Created by signing a message hash with the sender's private key (S = E_Kpriv(H(M))); verified by any party using the sender's public key (H(M) == D_Kpub(S)).
Example / Use-Case: ECDSA in Bitcoin blockchain transactions and RSA digital signatures in software code signing.

Q10. Define key management.

Formal Definition: Key management is the comprehensive set of policies, protocols, and mechanisms that oversee the secure lifecycle of cryptographic keys within a cryptosystem (NIST SP 800-57 / Forouzan).
Key Components / Formula / Mechanism: Encompasses end-to-end lifecycle phases: secure key generation, key exchange/distribution, protected key storage, periodic key rotation, revocation, and zeroization/destruction.
Example / Use-Case: Public Key Infrastructure (PKI) with Certificate Authorities, and AWS Key Management Service (KMS).

MODEL QUESTION PAPER – 2

Q1. Define data integrity.

Formal Definition: Data integrity is the security service ensuring information is protected against unauthorized modification, insertion, or deletion, preserving complete accuracy during transmission or storage (Stallings).
Key Components / Formula / Mechanism: Enforced via cryptographic hash functions `H(M)` or MACs to verify that `H(M_received) = H(M_sent)`.
Example / Use-Case: HMAC-SHA256 in TLS record protocols to detect unauthorized network tampering.

Q2. What is modular arithmetic?

Formal Definition: Modular arithmetic is an integer arithmetic system where numbers wrap around upon reaching a modulus `n`, written as `a ≡ b (mod n)` if `n` divides `(a - b)`.
Key Components / Formula / Mechanism: Operates on ring `Z_n = {0, 1, ..., n-1}`, preserving addition and

multiplication closure under modulo operations.
Example / Use-Case: Forms the mathematical foundation for RSA modular exponentiation (`C = M^e mod n`) and Diffie-Hellman key exchange.

Q3. Define a transposition cipher.

Formal Definition: A transposition cipher is a classical enciphering technique that rearranges the positions of plaintext characters according to a regular system, without changing their identities.
Key Components / Formula / Mechanism: Preserves single-letter frequency distributions while disrupting n-gram patterns using permutation vectors or columnar matrices.
Example / Use-Case: Rail Fence cipher, Columnar Transposition, and Permutation boxes (P-boxes) in DES and AES block ciphers.

Q4. What is a polyalphabetic substitution cipher?

Formal Definition: A polyalphabetic substitution cipher encrypts plaintext by substituting characters using multiple monoalphabetic cipher alphabets determined by a periodic key.
Key Components / Formula / Mechanism: Shifts characters via `C_i = (P_i + K_(i mod m)) mod 26` , flattening letter-frequency spikes to defeat simple frequency analysis.
Example / Use-Case: Vigenère cipher (using the Vigenère tableau) and the rotor-based WWII Enigma machine.

Q5. Define the Autokey Cipher.

Formal Definition: The Autokey Cipher is a polyalphabetic substitution cipher that eliminates periodic keyword patterns by incorporating the plaintext directly into the keystream.
Key Components / Formula / Mechanism: Uses a short initial seed key `K` followed by plaintext characters: `Key = (k_1, ..., k_m, p_1, p_2, ...)` , encrypting via `C_i = (P_i + K_i) mod 26` .
Example / Use-Case: Classical Autokey cipher; provided the conceptual origin for modern Cipher Feedback (CFB) stream modes.

Q6. What is Electronic Codebook (ECB) mode?

Formal Definition: Electronic Codebook (ECB) is the basic block cipher mode where plaintext is divided into fixed blocks, each encrypted independently with the identical key.
Key Components / Formula / Mechanism: Follows `C_i = E_K(P_i)` ; identical plaintext blocks produce identical ciphertext blocks, preserving structural patterns and leaking statistical data.
Example / Use-Case: Encrypting small single-block items like session keys; insecure for multi-block messages or bitmap images.

Q7. Define an Initialization Vector (IV).

Formal Definition: An Initialization Vector (IV) is a fixed-size, non-secret arbitrary or pseudorandom binary block used with a secret key to initialize cipher execution.
Key Components / Formula / Mechanism: Guarantees semantic security so encrypting identical plaintexts under the same key produces distinct ciphertexts; XORed with block 1: `C_1 = E_K(P_1 ⊕ IV)` .
Example / Use-Case: Required in CBC and GCM modes within IPsec, WPA2, and TLS 1.3.

Q8. What is a trapdoor one-way function?

Formal Definition: A trapdoor one-way function is a mathematical function that is easy to compute in one direction, but computationally infeasible to invert without special secret information called the "trapdoor".
Key Components / Formula / Mechanism: Forward `y = f(x)` is easy; inversion `x = f^{-1}(y)` is hard without key `k` , but efficient `x = f^{-1}_k(y)` with `k` .

Example / Use-Case: Foundational to RSA, where public encryption is modular exponentiation and private decryption uses trapdoor exponent `d`.

Q9. Define a message digest.

Formal Definition: A message digest is a fixed-length cryptographic hash output generated by processing an arbitrary-length message through a deterministic, one-way hash algorithm.
Key Components / Formula / Mechanism: Produces `h = H(M)`; provides preimage resistance, second-preimage resistance, collision resistance, and the strict avalanche effect.
Example / Use-Case: SHA-256 (256-bit output) and SHA-3; used in digital signatures and file integrity verification.

Q10. What is a Key Distribution Center (KDC)?

Formal Definition: A Key Distribution Center (KDC) is a centralized, trusted third-party service that authenticates principals and securely distributes symmetric session keys in large networks.
Key Components / Formula / Mechanism: Holds a master key with each entity; reduces key management complexity from `O(N^2)` to `O(N)` by issuing encrypted tickets.
Example / Use-Case: Needham-Schroeder protocol and Kerberos authentication server (AS/TGS).

MODEL QUESTION PAPER – 3

Q1. Define availability in information security.

Formal Definition: According to RFC 2828 and Stallings, availability is the property ensuring timely, reliable, and uninterrupted access to information, computing systems, and services by authorized users upon demand.
Key Components / Mechanism: Maintained via redundant hardware, failover clustering, load balancing, and data backups; evaluated by `Uptime = [MTBF / (MTBF + MTTR)] × 100%`.
Example / Use-Case: Mitigating DDoS attacks using Cloudflare scrubbing centers and RAID storage arrays.

Q2. What is a masquerading attack?

Formal Definition: A masquerade attack (Stallings) occurs when an unauthorized entity pretends to be a legitimate user by utilizing forged credentials or stolen tokens to gain unauthorized access privileges.
Key Components / Mechanism: Involves credential theft, IP/MAC spoofing, and session hijacking; typically paired with replay attacks to evade authentication controls.
Example / Use-Case: Using stolen Kerberos tickets (pass-the-ticket) or spoofed IP packets to obtain domain administrator access.

Q3. Define the Greatest Common Divisor (GCD).

Formal Definition: In number theory (Stallings), the Greatest Common Divisor `gcd(a, b)` is the largest positive integer dividing non-zero integers `a` and `b` without remainder.
Key Components / Formula: Integers are coprime if `gcd(a, b) = 1`. Computed via Euclidean algorithm: `gcd(a, b) = gcd(b, a mod b)`.
Example / Use-Case: RSA key generation, where public exponent `e` requires `gcd(e, ϕ(n)) = 1` to guarantee modular inverse `d` exists.

Q4. What is a monoalphabetic substitution cipher?

Formal Definition: A monoalphabetic substitution cipher is a symmetric cipher where each letter in the plaintext alphabet is mapped to a unique, fixed replacement character in the ciphertext alphabet.
Key Components / Mechanism: Fixed 1-to-1 bijection with `26!` keyspace; highly vulnerable to frequency analysis because plaintext letter distributions are completely preserved in ciphertext.
Example / Use-Case: Caesar cipher (fixed shift `k = 3`) and general mixed-alphabet substitution ciphers.

Q5. Define a Multiplicative Cipher. State the condition for selecting a valid key.

Formal Definition: A monoalphabetic substitution cipher where plaintext P is multiplied by secret key k modulo 26: $C = (P \times k) \bmod 26$.
Key Condition / Mechanism: Key k must be coprime to 26 ($\gcd(k, 26) = 1$) to ensure inverse k^{-1} exists for decryption; yields $\phi(26) = 12$ valid keys.
Example / Use-Case: For $k = 7$, inverse is $k^{-1} = 15$ since $(7 \times 15) \bmod 26 = 1$.

Q6. What is the A5/1 stream cipher?

Formal Definition: A5/1 is a symmetric stream cipher standardized by ETSI to provide over-the-air voice privacy and communication security in 2G GSM cellular networks.
Key Components / Mechanism: Constructed from three LFSRs (lengths 19, 22, and 23 bits; total 64-bit state) advanced via irregular majority-vote clocking using tapping bits; keyed with a 64-bit session key.
Example / Use-Case: Bit-by-bit XOR encryption of 114-bit speech frames between GSM mobile stations and Base Transceiver Stations (BTS).

Q7. State Fermat's Little Theorem.

Formal Definition: If p is prime and a is any positive integer coprime to p ($\gcd(a, p) = 1$), then $a^{p-1} \equiv 1 \pmod p$.
Key Formula / Mechanism: Alternatively, $a^p \equiv a \pmod p$. Used to compute modular inverses: $a^{-1} \equiv a^{p-2} \pmod p$.
Example / Use-Case: Primality testing (Miller-Rabin) and accelerating modular exponentiations in RSA and Diffie-Hellman cryptosystems.

Q8. What is a session key?

Formal Definition: A session key is a temporary, single-use symmetric cryptographic key generated dynamically to encrypt and authenticate all messages during a single communication session between two endpoints.
Key Components / Mechanism: Short lifetime limits exposure to cryptanalysis; generated via CSPRNG, distributed via asymmetric key exchange (RSA/Diffie-Hellman), and destroyed upon session teardown.
Example / Use-Case: Ephemeral AES-256 session keys negotiated during TLS 1.3 handshakes to encrypt HTTPS web browser traffic.

Q9. Define an X.509 digital certificate.

Formal Definition: An X.509 certificate is an ITU-T standard digital credential that securely binds a public key to an entity's verified identity through the digital signature of a trusted Certificate Authority (CA).
Key Components / Structure: Version, Serial Number, Signature Algorithm, Issuer Name, Validity Period, Subject Name, Subject Public Key Info, and CA Signature.
Example / Use-Case: SSL/TLS server certificates used in HTTPS to authenticate web domains and establish encrypted TLS tunnels.

Q10. What is the Schnorr Digital Signature Scheme?

Formal Definition: An asymmetric cryptographic signature scheme based on the hardness of the discrete logarithm problem (DLP) in cyclic groups.
Key Mechanism: Computes compact signature (e, s) using public key $y = g^{-x} \bmod p$, challenge $e = H(M \,||\, g^k \bmod p)$, and response $s = (k + x \times e) \bmod q$.
Example / Use-Case: Bitcoin Taproot (BIP 340) for efficient multi-signatures and batch verification.

MODEL QUESTION PAPER – 4

Q1. Define a security service. List any two security services.

Formal Definition: In ITU-T X.800, a security service is a processing or communication service provided by a protocol layer to ensure adequate security of systems and data transfers against attacks.
Key Examples / Classification: Two primary security services defined under X.800 are: (1) *Data Confidentiality* (protects transmitted data against unauthorized disclosure) and (2) *Peer Entity Authentication* (assures communicating entities of each other's verified identities).
Example / Use-Case: TLS/HTTPS providing confidentiality and server authentication in secure web browsing.

Q2. What is a multiplicative inverse in modular arithmetic?

Formal Definition: In modular arithmetic, the multiplicative inverse of integer a modulo n is an integer x satisfying $a \cdot x \equiv 1 \pmod{n}$, denoted a^{-1}.
Key Condition / Formula: An inverse exists if and only if $\gcd(a, n) = 1$, computed using the Extended Euclidean Algorithm.
Example / Use-Case: Modulo 26: $3^{-1} \equiv 9$ as $3 \times 9 = 27 \equiv 1 \pmod{26}$; used in RSA key generation.

Q3. Define the Hill Cipher.

Formal Definition: The Hill Cipher is a polygraphic substitution cipher based on linear algebra, encrypting blocks of m plaintext letters simultaneously via matrix multiplication modulo 26.
Key Formula / Mechanism: Encryption computes $C = (P \times K) \bmod 26$ and decryption computes $P = (C \times K^{-1}) \bmod 26$, where key matrix K must satisfy $\gcd(\det(K), 26) = 1$.
Example / Use-Case: Encrypting letter pairs (digrams) using a 2×2 matrix to conceal single-letter frequency patterns.

Q4. What is a One-Time Pad Cipher?

Formal Definition: The One-Time Pad (Vernam Cipher) is a symmetric stream cipher that is mathematically unbreakable (information-theoretically secure), as proven by Claude Shannon.
Key Properties / Formula: Encryption: $C_i = P_i \oplus K_i$; Decryption: $P_i = C_i \oplus K_i$. Requires a truly random key at least as long as the plaintext, used strictly once.
Example / Use-Case: Ultra-secure diplomatic communications, such as the historic Moscow–Washington hotline.

Q5. Define Counter (CTR) mode.

Formal Definition: Counter (CTR) mode is a block cipher mode of operation that turns a block cipher into a stream cipher by encrypting successive counter values combined with a nonce.
Key Mechanism / Formula: Keystream $O_i = E_K(\text{Nonce} \,||\, \text{Counter}_i)$ produces ciphertext $C_i = P_i \oplus O_i$. Supports parallel processing and random-access decryption without error propagation.
Example / Use-Case: High-speed network encryption including IPsec and AES-GCM in TLS 1.3.

Q6. What is Euler's Phi-Function?

Formal Definition: Euler's Phi-Function $\phi(n)$ counts positive integers up to n coprime to n, where $\gcd(k, n) = 1$.
Key Properties / Formula: For prime p, $\phi(p) = p - 1$. For distinct primes p, q, $\phi(pq) = (p - 1)(q - 1)$.
Example / Use-Case: In RSA, for $p = 7, q = 11$, $\phi(77) = 6 \times 10 = 60$, used to calculate decryption exponent d.

Q7. Define the RSA cryptosystem.

Formal Definition: The RSA cryptosystem (Rivest-Shamir-Adleman) is an asymmetric public-key algorithm whose security is based on the computational hardness of factoring large prime products.
Key Mechanism / Formula: Public key (e, n) and private key (d, n) where $n = p \times q$ and $d \equiv e^{-1} \pmod{\phi(n)}$.

Encryption: $C = M^e \bmod n$; Decryption: $M = C^d \bmod n$.
Example / Use-Case: Digital signatures, key exchange, and SSL/TLS certificate authentication.

Q8. What is a Certification Authority (CA)?

Formal Definition: A Certification Authority (CA) is a trusted third-party entity in Public Key Infrastructure (PKI) that verifies user identities and binds them to public keys by issuing digitally signed certificates.
Key Components / Functions: Validates subscriber identity, digitally signs X.509 certificates with its private key, publishes Certificate Revocation Lists (CRLs), and supports Online Certificate Status Protocol (OCSP).
Example / Use-Case: Trusted root CAs such as DigiCert and Let's Encrypt issuing SSL/TLS certificates for HTTPS websites.

Q9. Define Transport Layer Security (TLS).

Formal Definition: Transport Layer Security (TLS), successor to SSL, is an IETF cryptographic protocol providing privacy, mutual authentication, and data integrity over computer networks.
Key Architecture / Layers: Operates above TCP across two sub-protocols: *TLS Handshake Protocol* (negotiates cipher suites, authenticates peers) and *TLS Record Protocol* (provides symmetric encryption and HMAC integrity).
Example / Use-Case: Securing web traffic in HTTPS and email transmission via STARTTLS.

Q10. What is S/MIME?

Formal Definition: Secure/Multipurpose Internet Mail Extensions (S/MIME) is an IETF standard protocol providing end-to-end security services—confidentiality, authentication, integrity, and non-repudiation—for electronic mail.
Key Components / Mechanism: Built on X.509 digital certificates and PKCS #7 / CMS packaging. Employs asymmetric cryptography (RSA/ECDSA) for digital signatures and key distribution, and symmetric ciphers (AES) for bulk email encryption.
Example / Use-Case: Enterprise secure email signing and encryption supported natively in Microsoft Outlook and Apple Mail.

MODEL QUESTION PAPER – 5

Q1. Define steganography.

Formal Definition: Steganography is the cryptographic art of concealing secret messages within innocuous cover media, ensuring unauthorized observers cannot detect covert communication.
Key Components / Formula / Mechanism:

- **Cover Object:** Innocent carrier file (image, audio, or video).
- **LSB Embedding:** Replaces least significant bits of cover bytes with payload bits without perceptible distortion ($\text{Stego} = \text{Cover} \oplus \text{Message}$).

Example / Use-Case: Digital watermarking for copyright protection and covert key exchange in 24-bit PNG images.

Q2. What is a linear congruence? Write its general form.

Formal Definition: A linear congruence is a first-degree modular equation where an unknown multiplied by a constant is congruent to an integer modulo m.
Key Components / Formula / Mechanism:

- **General Form:** $ax \equiv b \pmod m$, with $a, b \in \mathbb{Z}$ and $m > 0$.
- **Solvability:** Yields a unique solution $x \equiv a^{-1}b \pmod m$ when $\gcd(a, m) = 1$.

Example / Use-Case: RSA key computation and Affine ciphers (e.g., $3x \equiv 4 \pmod 7$ gives $x \equiv 6$).

Q3. Define a security mechanism. Give any two examples.

Formal Definition: Under ITU-T X.800, a security mechanism is a technical process, algorithm, or device designed to detect, prevent, or recover from security attacks.

Key Components / Formula / Mechanism: Divided into Specific Mechanisms (layer-specific controls) and Pervasive Mechanisms (system-wide facilities like audit trails).
Example / Two Examples:

- **Encipherment:** Mathematical encryption (AES, RSA) enforcing data confidentiality.
- **Digital Signatures:** Asymmetric cryptographic signing (ECDSA) providing origin authentication and non-repudiation.

Q4. What is the Vigenère Cipher?

Formal Definition: The Vigenère cipher is a polyalphabetic substitution cipher that encrypts alphabetic text using repeated keyword letters to produce varying Caesar shifts.
Key Components / Formula / Mechanism:

- **Encryption / Decryption:** $C_i = (P_i + K_{i \bmod m}) \bmod 26$ and $P_i = (C_i - K_{i \bmod m}) \bmod 26$.
- **Mechanism:** Employs a 26×26 Vigenère Tableau to disguise single-letter frequencies.

Example / Use-Case: Diplomatic message encryption; historically analyzed using Kasiski examination.

Q5. Define a Feistel Cipher.

Formal Definition: A Feistel cipher is a symmetric block cipher framework where plaintext blocks are split into equal halves (L and R) and processed across repeated rounds.
Key Components / Formula / Mechanism:

- **Round Operation:** In round i : $L_i = R_{i-1}$ and $R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$.
- **Symmetry:** Encryption and decryption share the same algorithm, simply reversing round key order (K_n to K_1).

Example / Use-Case: Structural foundation of Data Encryption Standard (DES - 16 rounds) and Blowfish.

Q6. What is AES? List its three standard key sizes.

Formal Definition: Advanced Encryption Standard (AES) is a symmetric block cipher standardized by NIST (FIPS PUB 197) that processes data in fixed 128-bit blocks using a substitution-permutation network.
Key Components / Formula / Mechanism:

- **Round Transformations:** `SubBytes`, `ShiftRows`, `MixColumns`, and `AddRoundKey`.
- **Three Standard Key Sizes:**
  - **AES-128:** 128-bit key (10 rounds).
  - **AES-192:** 192-bit key (12 rounds).
  - **AES-256:** 256-bit key (14 rounds).

Example / Use-Case: Disk encryption (BitLocker) and secure web traffic via TLS 1.3 / HTTPS.

Q7. What is the RC4 stream cipher?

Formal Definition: RC4 is a variable key-size symmetric stream cipher designed by Ron Rivest that generates a pseudo-random keystream XORed byte-by-byte with plaintext.
Key Components / Formula / Mechanism:

- **Internal State:** 256-byte array `S[0..255]` and index pointers `i, j`.
- **Algorithms:** Key Scheduling Algorithm (KSA) permutes `S`; Pseudo-Random Generation Algorithm (PRGA) outputs keystream byte `K`, computing $C = P \oplus K$.

Example / Use-Case: Historically used in IEEE 802.11 WEP/WPA (TKIP) and SSL/TLS; now deprecated.

Q8. Define collision resistance in cryptographic hash functions.

Formal Definition: Collision resistance is a property of a cryptographic hash function `H` making it computationally infeasible to find two distinct inputs producing the identical digest.

Key Components / Formula / Mechanism:

- **Formal Condition:** Infeasible to locate $x \neq y$ such that $H(x) = H(y)$.
- **Security Bound:** Governed by the Birthday Paradox; an n-bit hash function offers $O(2^{n/2})$ resistance against collision attacks.

Example / Use-Case: Critical in SHA-256 digital certificates and blockchain integrity to prevent forgery.

Q9. What is the Digital Signature Standard (DSS)?

Formal Definition: The Digital Signature Standard (DSS) is a federal standard published by NIST (FIPS PUB 186) specifying algorithms for generating and verifying unforgeable digital signatures.
Key Components / Formula / Mechanism:

- **Mechanism:** Hashes plaintext and encrypts the digest with the sender's private key; verified using their public key.
- **Three Approved Algorithms:** DSA (discrete logarithms), RSA, and ECDSA.

Example / Use-Case: Legally binding e-signatures, operating system code signing, and X.509 PKI certificates.

Q10. Define Secure Sockets Layer (SSL).

Formal Definition: Secure Sockets Layer (SSL) is a standard cryptographic protocol suite developed by Netscape to establish an authenticated, encrypted, and tamper-resistant network connection over TCP.
Key Components / Formula / Mechanism:

- **Placement:** Operates between Application Layer (HTTP) and Transport Layer (TCP).
- **Subprotocols:** Handshake Protocol (mutual authentication and key negotiation via X.509 certificates) and Record Protocol (bulk encryption and HMAC integrity).

Example / Use-Case: Securing HTTPS web traffic and financial transactions; modernized and superseded by TLS.

PART II: SECTION B — 6 MARKS QUESTIONS (PAPERS 1 TO 5)

MODEL QUESTION PAPER – 1

Q11. Explain data confidentiality and privacy with suitable examples.

1. Core Concept & Formal Definition

- **Data Confidentiality:** A security service (ISO 7498-2) guaranteeing that sensitive information is inaccessible to unauthorized individuals, entities, or processes via technical controls like encryption.
- **Data Privacy:** A governance and legal framework (e.g., GDPR) governing an individual's right to control how their Personally Identifiable Information (PII) is collected, processed, and shared.

2. Conceptual Architecture Diagram

```
+-------------------------------------------------------------+
|              DATA PRIVACY (Governance & Policy)             |
|   - User Consent   - Purpose Limitation   - Data Retention  |
+-----------------------------+-------------------------------+
                              | Mandates Technical Controls
                              v
+-------------------------------------------------------------+
|         DATA CONFIDENTIALITY (Technical Security)           |
|    Plaintext Data ---> [ AES-256 / TLS 1.3 ] ---> Cipher    |
+-------------------------------------------------------------+
```

3. Core Dimensions & Features

- **Primary Scope:** Confidentiality secures data secrecy; Privacy safeguards individual autonomy and statutory rights.
- **Implementation:** Confidentiality uses encryption and access controls; Privacy enforces consent, anonymization, and data minimization.
- **Threat Vectors:** Confidentiality protects against snooping and breaches; Privacy counters unauthorized tracking and data monetization.
- **Lifecycle Governance:** Confidentiality protects stored and transmitted data; Privacy oversees collection through disposal.

4. Summary Comparison & Practical Examples

Dimension	Data Confidentiality	Data Privacy
Focus	Technical secrecy and access prevention	Ethical and lawful personal data handling
Enforcement	AES-256, TLS 1.3, strict RBAC	Consent notices, minimization policies
Example	Encrypting hospital EHR databases prevents unauthorized access.	Securing patient consent before sharing records with third parties.

Q12. Find gcd(210, 45) using the Euclidean Algorithm.

1. Core Concept & Euclidean Division Theorem

The **Euclidean Algorithm** computes the Greatest Common Divisor (GCD) of two positive integers `a` and `b` (where `a ≥ b`) based on the division algorithm:

> `a = q × b + r,  where 0 ≤ r < b`

The core property states that `gcd(a, b) = gcd(b, r)`. The process repeats until the remainder becomes zero; the last non-zero remainder is the GCD.

2. Step-by-Step Division Algorithm Computation

Given integers: **a = 210**, **b = 45**.

- **Step 1:** Divide 210 by 45:

  > $210 = 45 \times 4 + 30$ (Quotient $q_1 = 4$, Remainder $r_1 = 30$)

- **Step 2:** Divide 45 by 30:

  > $45 = 30 \times 1 + 15$ (Quotient $q_2 = 1$, Remainder $r_2 = 15$)

- **Step 3:** Divide 30 by 15:

  > $30 = 15 \times 2 + 0$ (Quotient $q_3 = 2$, Remainder $r_3 = 0$)

The remainder reaches `0` at Step 3. The last non-zero remainder is **15**.

3. Computation Trace Table & Execution Pipeline

```
[210, 45] ---> 210 = 45(4) + 30 ---> Remainder = 30
                   |
                   v
[45, 30]  ---> 45  = 30(1) + 15 ---> Remainder = 15 (Last non-zero)
                   |
                   v
[30, 15]  ---> 30  = 15(2) + 0  ---> Terminate (r = 0)
```

Iteration (i)	Dividend (a)	Divisor (b)	Quotient (q)	Remainder (r)	Division Relation
1	210	45	4	30	210 = 45(4) + 30
2	45	30	1	15	45 = 30(1) + 15
3	30	15	2	0	30 = 15(2) + 0

4. Mathematical Verification & Final Result

- **Prime Factorization Check:**
  - 210 = 2 × 3 × 5 × 7
  - 45 = 3^2 × 5 = 3 × 3 × 5
  - Common factors = 3 × 5 = **15**. Verified!
- **Bézout's Identity:** 15 = 45 - 30(1) = 45 - [210 - 45(4)] = 210(-1) + 45(5) = -210 + 225 = 15.
- **Divisibility Test:** 210 / 15 = 14 and 45 / 15 = 3, with gcd(14, 3) = 1.

Final Answer: gcd(210, 45) = 15

Q13. Encrypt the plaintext CODE using an Additive Cipher with key K = 9. Decrypt the resulting ciphertext.

1. Core Concept & Mathematical Formulation

The **Additive Cipher** (Shift Cipher) is a monoalphabetic substitution cipher mapping characters in $\mathbb{Z}_{26}$ (A = 0, B = 1, ..., Z = 25) using modular arithmetic:

- **Encryption Formula:** `C = (P + K) mod 26`
- **Decryption Formula:** `P = (C - K) mod 26 = (C - K + 26) mod 26`
- **Given:** Plaintext = `CODE`, Shift Key `K = 9`.

2. Character Mapping & Encryption Process

Plaintext values: **C = 2**, **O = 14**, **D = 3**, **E = 4**.

- **Character C (P = 2):** $C_1 = (2 + 9) \bmod 26 = 11 \Rightarrow$ **L**
- **Character O (P = 14):** $C_2 = (14 + 9) \bmod 26 = 23 \Rightarrow$ **X**
- **Character D (P = 3):** $C_3 = (3 + 9) \bmod 26 = 12 \Rightarrow$ **M**
- **Character E (P = 4):** $C_4 = (4 + 9) \bmod 26 = 13 \Rightarrow$ **N**

Generated Ciphertext: **LXMN**

3. Decryption Process & Verification

Ciphertext values: **L = 11**, **X = 23**, **M = 12**, **N = 13**.

- **Character L (C = 11):** $P_1 = (11 - 9) \bmod 26 = 2 \Rightarrow$ **C**
- **Character X (C = 23):** $P_2 = (23 - 9) \bmod 26 = 14 \Rightarrow$ **O**
- **Character M (C = 12):** $P_3 = (12 - 9) \bmod 26 = 3 \Rightarrow$ **D**
- **Character N (C = 13):** $P_4 = (13 - 9) \bmod 26 = 4 \Rightarrow$ **E**

Recovered Plaintext: **CODE**

4. Transformation Summary Table

Plaintext Char	Value (P)	Encryption: (P + 9) mod 26	Cipher Value (C)	Ciphertext Char	Decryption: (C - 9) mod 26	Recovered Char
C	2	(2 + 9) mod 26 = 11	11	**L**	(11 - 9) mod 26 = 2	**C**
O	14	(14 + 9) mod 26 = 23	23	**X**	(23 - 9) mod 26 = 14	**O**
D	3	(3 + 9) mod 26 = 12	12	**M**	(12 - 9) mod 26 = 3	**D**
E	4	(4 + 9) mod 26 = 13	13	**N**	(13 - 9) mod 26 = 4	**E**

Ciphertext: LXMN | Decrypted Plaintext: CODE

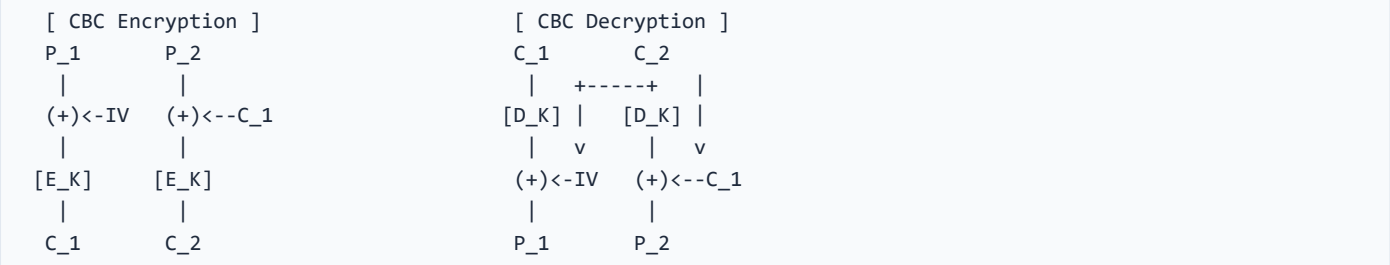
Q14. Explain the working of Cipher Block Chaining (CBC) mode with a neat diagram.

1. Core Concept & Formal Definition

Cipher Block Chaining (CBC) is a block cipher feedback mode (NIST SP 800-38A). Each plaintext block is XORed with the preceding ciphertext block prior to encryption under key K, eliminating ECB mode's pattern-leakage vulnerability.

$$\text{Encryption: } C_1 = E_K(P_1 \oplus IV), \quad C_i = E_K(P_i \oplus C_{i-1}) \quad (i \geq 2)$$
$$\text{Decryption: } P_1 = D_K(C_1) \oplus IV, \quad P_i = D_K(C_i) \oplus C_{i-1} \quad (i \geq 2)$$

2. Architectural Diagram (Encryption & Decryption)



3. Core Operational Features & Dimensions

- **Initialization Vector (IV):** A random nonce ensuring identical plaintexts generate distinct ciphertexts.
- **Sequential Encryption:** Serial execution since C_i requires C_{i-1}, preventing parallel encryption.
- **Parallel Decryption:** Pre-known ciphertext blocks allow concurrent decryption across multi-core processors.
- **Error Propagation:** A 1-bit error in C_i corrupts plaintext P_i and inverts 1 bit in P_{i+1}, self-recovering thereafter.

4. Summary Characteristics & Applications

Feature	Operational Behavior	Impact
Block Alignment	Fixed blocks (e.g., 128-bit AES)	Requires PKCS#7 padding
Parallel Execution	Serial encryption, parallel decryption	Decryption scales with cores
Primary Use	IPsec ESP, BitLocker, TLS suites	Standard bulk symmetric cipher

---

Q15. Explain the three important security requirements of a cryptographic hash function.

1. Core Concept & Formal Definition

A **Cryptographic Hash Function** is a deterministic algorithm `H: {0, 1}* → {0, 1}ⁿ` compressing arbitrary-length input `M` into a fixed `n`-bit digest `h = H(M)` (NIST SP 800-107). It requires three cardinal security properties.

2. Architectural Security Models Diagram

```
1. Pre-image Resistance:        Given h   ---> Infeasible to find M with H(M) = h
2. Second Pre-image Resistance: Given M_1 ---> Infeasible to find M_2 != M_1 with H(M_2) = H(M_1)
3. Collision Resistance:        Given Ø   ---> Infeasible to find ANY M_1 != M_2 with H(M_1) = H(M_2)
```

3. Core Dimensions of the Security Requirements

- **Pre-image Resistance (One-Way):** Given digest `h`, finding input `M` with `H(M) = h` is computationally infeasible. Brute-force complexity is `O(2ⁿ)`, protecting stored passwords.
- **Second Pre-image Resistance (Weak Collision):** Given input `M₁`, finding `M₂ ≠ M₁` such that `H(M₁) = H(M₂)` is computationally infeasible. Complexity is `O(2ⁿ)`, preventing document substitution.
- **Collision Resistance (Strong Collision):** Finding *any* pair `M₁ ≠ M₂` with `H(M₁) = H(M₂)` is computationally infeasible. Birthday Paradox reduces complexity to `O(2ⁿ/²)`, preventing signature forgery.
- **Avalanche Effect:** A 1-bit input difference flips ~50% of output digest bits unpredictably.

4. Comparative Summary of Hash Requirements

Property	Given	Complexity	Attack Mitigated
Pre-image	Digest `h`	$O(2^n)$	Password cracking
Second Pre-image	Message M_1	$O(2^n)$	Document substitution
Collision	None	$O(2^{n/2})$	Signature forgery (Birthday attack)

Q16. Explain the security services provided by digital signatures.

1. Core Concept & Formal Definition

A **Digital Signature** is an asymmetric cryptographic mechanism (NIST FIPS 186-5) binding an entity to digital data using private key `PR` and public key `PU`.

The sender signs digest `S = E_PR(H(M))`. The recipient decrypts `S` with public key `PU` and checks if decrypted hash `h'` matches local hash `H(M)`.

2. Operational Architecture Diagram

```
[ SIGNING ]
  Message (M) ---> [ Hash H ] ---> [ Sign with PR_Alice ] ---> Signature (S)
                                                                    |
+---------------------- Transmit (M + S) --------------------------+
|
v
[ VERIFYING ]
  Message (M) ---> [ Hash H ] ------------------------> Digest H(M)
                                                          | (Compare)
  Signature (S) -> [ Decrypt with PU_Alice ] ---------> Digest H'
                                    Check: H(M) == H' ? Valid : Invalid
```

3. Core Security Services Provided

- **Sender Authentication:** Cryptographically validates origin since only Alice possesses private key `PR_A` needed to produce signature `S`.
- **Data Integrity:** Detects tampering; any modification causes hash mismatch (`H(M) ≠ h'`) during verification.
- **Non-Repudiation:** Prevents signer from denying authorship because private key `PR_A` is held exclusively by them.
- **Tamper Evidence:** Combined with Timestamp Authorities (RFC 3161), signatures certify document state and signing time.

4. Summary Matrix & Real-World Applications

Security Service	Cryptographic Enabler	Primary Application
Authentication	Asymmetric key (`PR_A`)	S/MIME, PGP email
Integrity	Cryptographic hash (SHA-256)	Software code signing
Non-Repudiation	Exclusive private key custody	Banking fund transfers (SWIFT)

MODEL QUESTION PAPER – 2

Q11. Explain any five security mechanisms defined in the X.800 Security Architecture.

1. Core Concept & Formal Definition

Under ITU-T X.800, a **security mechanism** is a technical process, algorithm, or device implementing one or more security services (confidentiality, integrity, authentication). X.800 divides mechanisms into specific protocol-layer mechanisms and pervasive management mechanisms.

2. Architectural Relationship Diagram

```
[X.800 Specific Security Mechanisms]
Encipherment       --> Confers Data Confidentiality
Digital Signature  --> Guarantees Non-Repudiation
Access Control     --> Enforces Entity Authorization
Data Integrity     --> Detects Unauthorized Modification
Auth Exchange      --> Verifies Communicating Identities
```

3. Detailed Explanation of Five Mechanisms

- **Encipherment:** Mathematical ciphers (AES/RSA) transforming plaintext into ciphertext to preserve confidentiality.
- **Digital Signature:** Asymmetric signing with sender private key ensuring origin authenticity and non-repudiation.
- **Access Control:** Enforces authorization via access control lists and capability tickets to prevent unauthorized resource usage.
- **Data Integrity:** Utilizes keyed MACs and cryptographic hash digests to detect message alteration or replay.
- **Authentication Exchange:** Two-way protocol handshakes with nonces and timestamps verifying peer identity.

4. Summary Mapping Table

Security Mechanism	Primary Method	Service Provided	Protocol Example
Encipherment	Symmetric / Asymmetric	Confidentiality	AES in TLS 1.3
Digital Signature	Private key signing	Non-Repudiation	RSA in X.509
Access Control	ACLs & tickets	Access Control	RADIUS / Firewalls
Data Integrity	Hashes & HMACs	Integrity	HMAC in IPsec AH
Auth Exchange	Challenge-response	Entity Authentication	Kerberos / CHAP

Q12. Explain the four major types of cryptanalytic attacks.

1. Core Concept & Formal Classification

Cryptanalysis is the study of mathematical principles for recovering plaintext or key without prior key knowledge. Modern cryptanalysis classifies attacks into an escalating hierarchy based on adversary access to ciphertexts, plaintexts, and cryptographic oracles.

2. Cryptanalytic Attack Capability Hierarchy

```
Attacker Capability & Oracle Access:
[Ciphertext-Only] --> [Known-Plaintext] --> [Chosen-Plaintext] --> [Chosen-Ciphertext]
(Intercepted C)       (Known P-C pairs)     (Encryption Oracle)   (Decryption Oracle)
```

3. Detailed Analysis of the Four Attack Types

- **Ciphertext-Only Attack (COA):** Adversary possesses intercepted ciphertexts only. Recovery relies on statistical frequency analysis or brute-force key search.
- **Known-Plaintext Attack (KPA):** Attacker possesses known plaintext-ciphertext pairs `(P, C)`, analyzing algebraic structures to deduce target key `K`.
- **Chosen-Plaintext Attack (CPA):** Attacker selects arbitrary plaintexts and receives resulting ciphertexts from an encryption oracle (e.g., differential cryptanalysis).
- **Chosen-Ciphertext Attack (CCA):** Attacker submits chosen ciphertexts to a decryption oracle to inspect recovered plaintexts (e.g., Bleichenbacher padding attacks).

4. Summary Comparison Table

Attack Type	Attacker Knowledge	Oracle Access	Primary Defense
Ciphertext-Only	Intercepted `C` only	None (Passive)	Large keyspace & high entropy
Known-Plaintext	Known `(P, C)` pairs	Historical capture	Strong non-linear S-boxes
Chosen-Plaintext	Chosen `P` and ciphertexts	Encryption oracle	Randomized padding & IVs
Chosen-Ciphertext	Chosen `C` and plaintexts	Decryption oracle	Authenticated encryption (AEAD)

Q13. Encrypt the plaintext DATA using a Multiplicative Cipher with key K = 5. Decrypt the resulting ciphertext.

1. Mathematical Formulation

A **Multiplicative Cipher** is a substitution cipher over $\mathbb{Z}_{26}$. Encryption: `C = (P × K) mod 26` with `gcd(K, 26) = 1`. Since `gcd(5, 26) = 1`, key `K = 5` is valid. Decryption: `P = (C × K^-1) mod 26`.

2. Encryption of Plaintext "DATA"

Mapping `A=0, ..., Z=25` yields `D=3, A=0, T=19, A=0` with key `K = 5`:

Plaintext	Value (P)	(P × 5)	Modulo 26	Ciphertext
D	3	3 × 5 = 15	15 mod 26 = 15	**P**
A	0	0 × 5 = 0	0 mod 26 = 0	**A**
T	19	19 × 5 = 95	95 mod 26 = 17	**R**
A	0	0 × 5 = 0	0 mod 26 = 0	**A**

Resulting Ciphertext: `PARA`

3. Multiplicative Inverse Derivation (K^{-1})

Find K^{-1} satisfying $5 \times K^{-1} \equiv 1 \pmod{26}$. Extended Euclidean Algorithm: $26 = 5(5) + 1 \implies 1 = 26 - 5(5) \implies 5(-5) \equiv 1 \pmod{26}$. Since $-5 \equiv 21 \pmod{26}$, verify: $5 \times 21 = 105 = 4(26) + 1 \equiv 1 \pmod{26}$. Hence, $K^{-1} = 21$.

4. Decryption of Ciphertext "PARA"

Decryption mapping for `P=15, A=0, R=17, A=0` using $K^{-1} = 21$:

Ciphertext	Value (C)	(C × 21)	Modulo 26	Plaintext
P	15	15 × 21 = 315	315 mod 26 = 3	**D**
A	0	0 × 21 = 0	0 mod 26 = 0	**A**
R	17	17 × 21 = 357	357 mod 26 = 19	**T**
A	0	0 × 21 = 0	0 mod 26 = 0	**A**

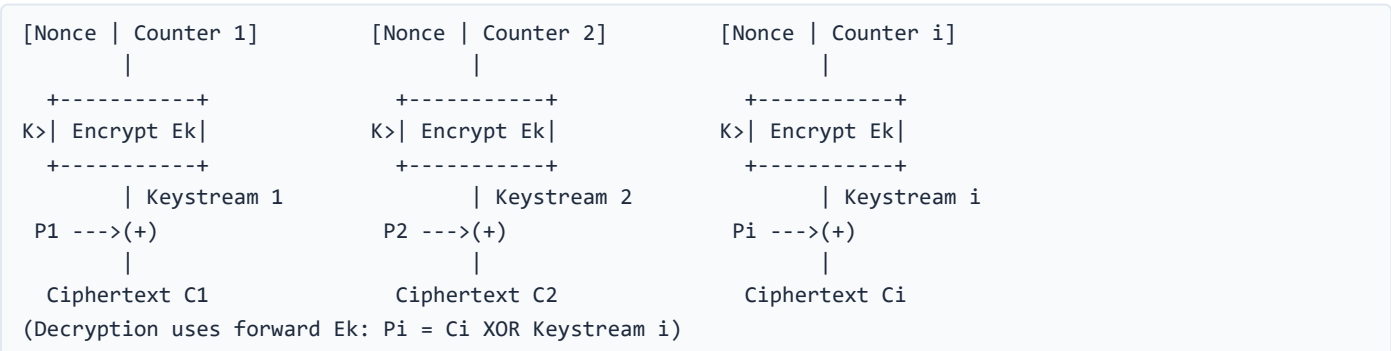
Recovered Plaintext: `DATA` (Decryption verified).

Q14. Explain the working of Counter (CTR) mode with a neat diagram.

1. Core Concept & Formal Definition

Counter (CTR) mode (NIST SP 800-38A) turns a block cipher into a stream cipher. CTR encrypts successive counter values combined with a Nonce to produce a keystream, XORed with plaintext blocks without padding.

2. Architectural Block Diagram



3. Key Operational Principles

- **Keystream Generation:** Encrypts unique block counter $T_i = Nonce \,||\, Counter_i$ using forward block cipher $O_i = E_K(T_i)$.
- **Symmetric XOR Processing:** Computes $C_i = P_i \oplus O_i$ and recovers $P_i = C_i \oplus O_i$ using forward cipher E_K.
- **Parallel Execution:** Counters are predictable, allowing concurrent encryption and decryption across multi-core processors.
- **Random Access:** Any block `i` can be decrypted independently without processing preceding ciphertext blocks.

4. Summary Operational Dimensions

Dimension	Implementation	Advantage
Circuit Area	Forward cipher E_K only	Halves silicon cost
Throughput	Parallel block processing	High pipeline speed
Error Impact	1-bit error affects 1 plaintext bit	No error cascade
Security Rule	Unique (Nonce, Counter) pair	Prevents two-time pad leak

---

Q15. Differentiate between conventional signatures and digital signatures.

1. Core Concept & Formal Definition

A **conventional signature** is a physical, handwritten mark made with ink on paper signifying personal assent. A **digital signature** is a cryptographic construct using asymmetric cryptography that binds the signer's private key to a document hash digest, providing authenticity, integrity, and non-repudiation.

2. Verification Architecture Model

```
Conventional Signature:
[Physical Paper] ---> Written with Ink Pen ---> Subjective Visual Inspection

Digital Signature:
[Message M] ---> Hash H(M) ---> Encrypt with Private Key ---> Signature S
Verification:
Decrypt S with Public Key ---> H'; Compare H(M) == H' (Deterministic Match)
```

3. Core Differentiating Dimensions

- **Content Binding:** Conventional signatures are document-independent. Digital signatures are bound to message bits; altering any bit invalidates the signature.
- **Forgery Resistance:** Handwritten signatures can be traced or photocopied. Digital signatures are computationally infeasible to forge without cracking private keys (RSA/ECDSA).
- **Verification Mode:** Conventional signatures rely on subjective inspection. Digital signatures are verified objectively by algorithms using public keys.
- **Tamper Detection:** Modifying paper text leaves ink signatures intact. In digital signatures, any alteration triggers an immediate hash mismatch.

4. Summary Comparison Table

Parameter	Conventional Signature	Digital Signature
Medium	Physical ink and paper	Asymmetric cryptography & hashing
Content Binding	Independent of document text	Unique to each message bitstring
Verification	Subjective visual inspection	Deterministic algorithmic check
Tamper Proof	Weak; requires forensic tests	Absolute; bit change breaks signature
Reusability	Signature pattern can be copied	Cannot be copied or transferred

---

Q16. Explain the Diffie–Hellman Key Agreement Protocol.

1. Core Concept & Theoretical Basis

The **Diffie–Hellman Key Agreement Protocol** (1976) enables two parties (Alice and Bob) to establish a shared secret key over an insecure channel without transmitting the secret itself. Security relies on the hardness of the Discrete Logarithm Problem in finite cyclic groups $\mathbb{Z}_p^*$.

2. Protocol Exchange Architecture

```
Public Parameters: Large prime p, Primitive root g modulo p
ALICE (Private X_A)                          BOB (Private X_B)
Compute: Y_A = (g ^ X_A) mod p               Compute: Y_B = (g ^ X_B) mod p
         |------------ Public Key Y_A ----------->|
         |<----------- Public Key Y_B ------------|
Shared Secret Derivation:                    Shared Secret Derivation:
K = (Y_B ^ X_A) mod p                        K = (Y_A ^ X_B) mod p
  = g ^ (X_A * X_B) mod p                      = g ^ (X_A * X_B) mod p
```

3. Protocol Execution Steps

- **Parameter Setup:** Both parties agree on public prime p and generator g.
- **Key Pair Generation:** Alice chooses private X_A and publishes $Y_A = g^{X_A} \bmod p$; Bob chooses private X_B and publishes $Y_B = g^{X_B} \bmod p$.
- **Shared Key Derivation:** Alice computes $K = (Y_B)^{X_A} \bmod p$; Bob computes $K = (Y_A)^{X_B} \bmod p$. Both obtain $g^{X_A X_B} \bmod p$.
- **Security Hardness:** Eavesdroppers intercepting public values cannot compute K without solving the Discrete Logarithm Problem.

4. Numerical Demonstration & Security Characteristics

Parameter	Alice Execution	Bob Execution
Domain Setup	Public agreement on prime p = 23, Generator g = 5	
Key Pair	Private $X_A = 6 \rightarrow Y_A = 5^6 \bmod 23 = 8$	Private $X_B = 15 \rightarrow Y_B = 5^{15} \bmod 23 = 19$
Shared Key (K)	$K = 19^6 \bmod 23 = 2$	$K = 8^{15} \bmod 23 = 2$
Vulnerability	Vulnerable to MitM; authenticated using digital certificates (ECDHE)	

MODEL QUESTION PAPER – 3

Q11. Explain the relationship between security goals, security services and security mechanisms with suitable examples. (6 Marks)

1. Core Concept & Formal Definition

Under ITU-T X.800 and RFC 4949, security architecture is organized into three tiers:

- **Security Goals:** CIA triad objectives—*Confidentiality* (protecting against disclosure), *Integrity* (safeguarding against unauthorized alteration), and *Availability* (assuring authorized access).
- **Security Services:** Protocol-layer capabilities enforcing security policies (e.g., Confidentiality, Authentication, Data Integrity, Non-repudiation, Access Control).
- **Security Mechanisms:** Concrete algorithmic tools executing security services (e.g., symmetric/asymmetric ciphers, cryptographic hashes, digital signatures).
- **Interrelationship:** *Goals* define **what** to achieve; *Services* define **which** policies enforce them; *Mechanisms* determine **how** to execute them technically.

2. Architectural Relationship Model

```
Security Goals (CIA Triad: Confidentiality, Integrity, Availability)
       |  (Enforced via Policies)
       v
Security Services (Data Confidentiality, Data Integrity, Authentication)
       |  (Implemented via Algorithms)
       v
Security Mechanisms (AES-256-GCM, SHA-256 HMAC, RSA / ECDSA Signatures)
```

3. Core Dimensions & Features

- **Hierarchical Flow:** Goals formulate mandatory criteria; services define operational protections, and mechanisms execute cryptographic algorithms.
- **Many-to-Many Binding:** One mechanism (e.g., digital signatures) provides multiple services (integrity, authentication, non-repudiation); confidentiality employs AES or RSA.
- **Protocol Placement:** Services reside at network layers (IPsec, TLS), whereas mechanisms execute purely algorithmic functions.
- **Threat Alignment:** Specific mechanisms defeat distinct attacks: encryption counters eavesdropping, hashes defeat tampering, and digital signatures defeat repudiation.

4. Summary Mapping & Concrete Examples

Security Goal	Security Service	Security Mechanism	Practical Example
Confidentiality	Connection Confidentiality	AES-256-GCM	Encrypting HTTPS application payloads.
Integrity	Connectionless Integrity	SHA-256 HMAC	Verifying packet authenticity in TLS.
Non-Repudiation	Origin Non-Repudiation	RSA / ECDSA	Legally signing electronic bank transfers.
Availability	Availability Service	Ingress Rate Limiting	Thwarting SYN flood volumetric DDoS attacks.

Q12. Explain the set of residues Zn and congruence relations with suitable examples. (6 Marks)

1. Core Concept & Formal Definition

In modular arithmetic and abstract algebra:

- **Set of Residues (Z_n):** For modulus $n > 1$, $Z_n = \{0, 1, ..., n-1\}$ represents all least non-negative remainders modulo n. Any integer a satisfies $a = q \cdot n + r$ (with $0 \le r < n$), where $r = a \bmod n$ is the unique residue.
- **Congruence Relation:** Integers a, b are congruent modulo n, written $a \equiv b \pmod n$, if n divides $(a - b)$, meaning both leave identical remainders: $a \bmod n = b \bmod n$.

2. Mathematical Model & Partitioning

```
Integer Partitioning into Residue Classes modulo n = 5:
  [0] = {..., -5, 0, 5, ...}     [3] = {..., -2, 3, 8, ...}
  [1] = {..., -4, 1, 6, ...}     [4] = {..., -1, 4, 9, ...}
  [2] = {..., -3, 2, 7, ...}     Z_5 = {[0], [1], [2], [3], [4]}
Congruence Criterion: a ≡ b (mod n) <===> (a - b) = k·n, k ∈ Z
```

3. Core Dimensions & Features

- **Equivalence Properties:** Congruence is an equivalence relation satisfying reflexivity ($a \equiv a$), symmetry ($a \equiv b \Rightarrow b \equiv a$), and transitivity.
- **Partitioning:** Disjoint residue classes $[r]$ completely partition the infinite integer set Z.
- **Ring Structure:** $(Z_n, +, \times)$ forms a commutative ring with identity; if n is prime, it forms finite field GF(n).
- **Reduced Residues:** Coprime elements $\{a \in Z_n \mid \gcd(a, n)=1\}$ form multiplicative group Z_n^* of order $\varphi(n)$.

4. Summary & Concrete Examples

Operation	Algebraic Rule	Calculation (n = 7)	Result
Residue	$r = a \bmod n$	$-17 \bmod 7$	4, since $-17 = (-3)(7) + 4$
Congruence	$n \mid (a - b)$	$23 \equiv 2 \pmod 7$	Valid: $7 \mid (23 - 2 = 21)$
Addition	$(a + b) \bmod n$	$(5 + 6) \bmod 7$	$11 \bmod 7 = 4 \in Z_7$
Multiplication	$(a \times b) \bmod n$	$(4 \times 5) \bmod 7$	$20 \bmod 7 = 6 \in Z_7$

Q13. Explain the One-Time Pad Cipher and its important requirements. (6 Marks)

1. Core Concept & Formal Definition

Devised by Vernam (1917) and proven by Shannon (1949), the One-Time Pad (OTP) cipher provides information-theoretic security:

- **Formal Definition:** Plaintext bits $P = (p_1, ..., p_m)$ and random key bits $K = (k_1, ..., k_m)$ of equal length are combined via XOR: $c_i = p_i \oplus k_i$. Decryption recovers plaintext via $p_i = c_i \oplus k_i$.
- **Perfect Secrecy:** When key K is truly random and used once, $P(P = m \mid C = c) = P(P = m)$. Ciphertext yields zero statistical data about plaintext, resisting unlimited computing power.

2. Mathematical Model & Flow

```
Encryption:  Plaintext  P : 0 1 0 0 1 0 0 0 ('H')
                            ⊕   ⊕   ⊕   ⊕   ⊕   ⊕   ⊕   ⊕
             Random Key K : 1 1 0 1 0 1 1 0 (Shared Pad, |K| = |P|)
                            =   =   =   =   =   =   =   =
             Ciphertext C : 1 0 0 1 1 1 1 0
Decryption:  C ⊕ K = (P ⊕ K) ⊕ K = P ⊕ (K ⊕ K) = P ⊕ 0 = P ('H')
```

3. Core Dimensions & Key Requirements

- **True Physical Randomness:** Keys must come from physical entropy (Hardware TRNG); algorithmic pseudo-random generators (PRNG) destroy perfect secrecy.
- **Key Length (|K| ≥ |P|):** Pad length must match message length; shorter repeating keys degrade the cipher to a vulnerable polyalphabetic cipher.
- **Strict One-Time Use:** Reusing K across messages exposes $C_1 \oplus C_2 = P_1 \oplus P_2$, allowing two-time pad cryptanalysis.
- **Secure Pad Destruction:** Keys require secure out-of-band exchange and immediate destruction after decryption to guarantee forward secrecy.

4. Summary Table of Requirements

Requirement	Technical Objective	Vulnerability if Breached	Mitigation
True Randomness	Uniform probability across keys	PRNG seed cryptanalysis	Hardware noise sources
Length (|K| ≥ |P|)	Prevents periodic repetitions	Reduces to Vigenere cipher	1:1 pad allocation
Zero Key Reuse	Preserves ciphertext independence	C1 ⊕ C2 leaks plaintext XOR	Single-use pad rule
Secure Destruction	Guarantees forward secrecy	Stored pads expose past traffic	Immediate register wipe

---

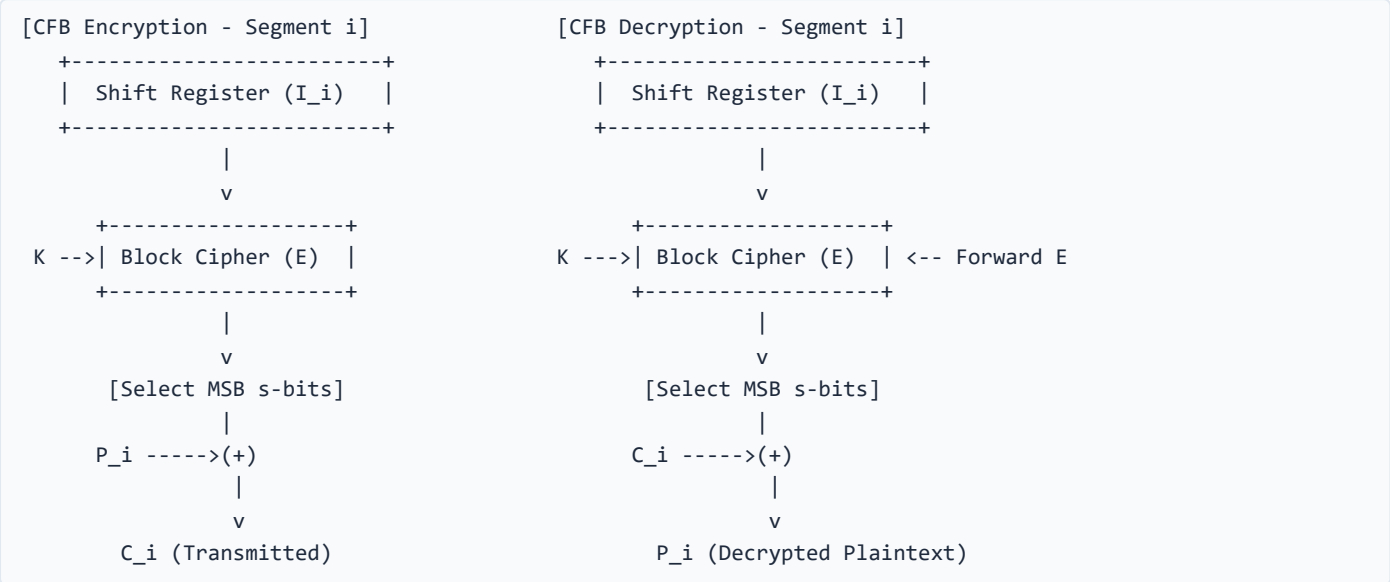
Q14. Explain the working of Cipher Feedback (CFB) mode with a neat diagram. (6 Marks)

1. Core Concept & Formal Definition

Cipher Feedback (CFB) mode, specified in NIST SP 800-38A, converts a symmetric block cipher (e.g., AES) into a self-synchronizing stream cipher:

- **Operational Mechanism:** A b-bit shift register initialized with an IV is encrypted under key K. The s most significant bits (MSBs) of output are XORed with an s-bit plaintext segment P_i to yield ciphertext segment C_i ($1 \le s \le b$; typically $s = 8$ or 128): $C_i = P_i \oplus \text{MSB}_s(E_K(I_i))$.
- **Feedback Loop:** Ciphertext C_i is transmitted and shifted into the register's rightmost s bits for subsequent rounds. Decryption applies identical forward encryption: $P_i = C_i \oplus \text{MSB}_s(E_K(I_i))$.

2. Architecture & Diagram



3. Core Dimensions & Features

- **Forward Encryption in Both Paths:** Both encryption and decryption execute exclusively via forward cipher function (E_K), eliminating reverse decryption circuitry.
- **Stream Transmission Without Padding:** Supports real-time transmission in variable s-bit units (e.g., character-by-character), avoiding padding overhead.
- **Finite Error Propagation:** A 1-bit channel error corrupts current segment C_i and persists in the shift register for $\lceil b/s \rceil$ segments before clearing.
- **Self-Synchronization:** If segments are dropped, decryption automatically resynchronizes once b clean ciphertext bits replenish the shift register.

4. Summary & Comparison

Parameter	CFB Mode Specification	Comparison with CBC	Operational Benefit
Cipher Routine	Forward E_K for Encrypt & Decrypt	CBC requires both E_K and D_K	Reduces hardware footprint
Segment Unit (s)	$s = 8$ or $s = 128$ bits	CBC requires 128-bit blocks	Real-time terminal streams
IV Requirement	Unique, unpredictable nonce	Static IV enables replays	Unique session keystreams
Error Recovery	Self-syncs after b clean bits	Stream ciphers lose sync	Robust over noisy links

---

Q15. State the Chinese Remainder Theorem and explain its applications. (6 Marks)

1. Core Concept & Formal Definition

The Chinese Remainder Theorem (CRT) is a central theorem of modular arithmetic and public-key cryptography:

- **Theorem Statement:** Let $m_1, ..., m_k$ be pairwise coprime positive integers ($\gcd(m_i, m_j) = 1$ for $i \ne j$), and $a_1, ..., a_k$ be arbitrary integers. The system:
  $x \equiv a_i \pmod{m_i}$ for $i = 1, 2, ..., k$
  admits a unique integer solution x modulo M, where $M = \prod m_i$.

- **Constructive Formula:** The solution is evaluated as: $x = [\sum (a_i \cdot M_i \cdot y_i)] \bmod M$, where $M_i = M / m_i$ and $y_i \equiv M_i^{-1} \pmod{m_i}$ (modular inverse via Extended Euclidean Algorithm).

2. Architecture & Solution Model

```
System of Congruences:                Ring Isomorphism:
  x ≡ a_1 (mod m_1)
  x ≡ a_2 (mod m_2)                        Z_M  <========>  Z_m1 × Z_m2 × ... × Z_mk
       :                                    x   <--------->  (a_1,  a_2,  ...,  a_k)
  x ≡ a_k (mod m_k)                   [Single Large Mod]    [Fast Parallel Moduli]
-------------------------------------------------------------------------------
Formula Steps:  1. M = ∏ m_i   2. M_i = M / m_i   3. y_i = (M_i)^(-1) mod m_i
                4. x = [ ∑ (a_i · M_i · y_i) ] mod M
```

3. Core Dimensions & Features

- **Pairwise Coprimality:** Moduli must satisfy $\gcd(m_i, m_j) = 1$; common factors destroy uniqueness or cause inconsistency.
- **Ring Isomorphism ($Z_M \cong \prod Z_{m_i}$):** Maps large operations modulo M into parallel independent arithmetic operations over smaller moduli m_i.
- **Computational Speedup:** Drastically accelerates multi-precision operations by executing calculations concurrently across independent modular units.
- **Uniqueness Modulo M:** Any two solutions x_1, x_2 satisfy $x_1 \equiv x_2 \pmod{M}$, guaranteeing a unique solution in $0 \le x < M$.

4. Summary Table of Applications

Application	Implementation Role	Benefit / Speedup	Numerical Example
RSA Decryption	Computes $c^d \bmod n$ via p and q	Speeds up private key ops ≈ 4×	Decomposes 2048-bit math to 1024-bit
Secret Sharing	Asmuth-Bloom threshold scheme	Any k of n shares recover secret	Modular shares $S \bmod m_i$
RNS Hardware	Residue Number System arithmetic	Carry-free parallel computation	High-speed DSP signal filters
Calculation Example	Solve $x \equiv 2 \pmod 3$, $x \equiv 3 \pmod 5$	M=15; M_1=5,y_1=2; M_2=3,y_2=2	$x = (20 + 18) \bmod 15 = 8$

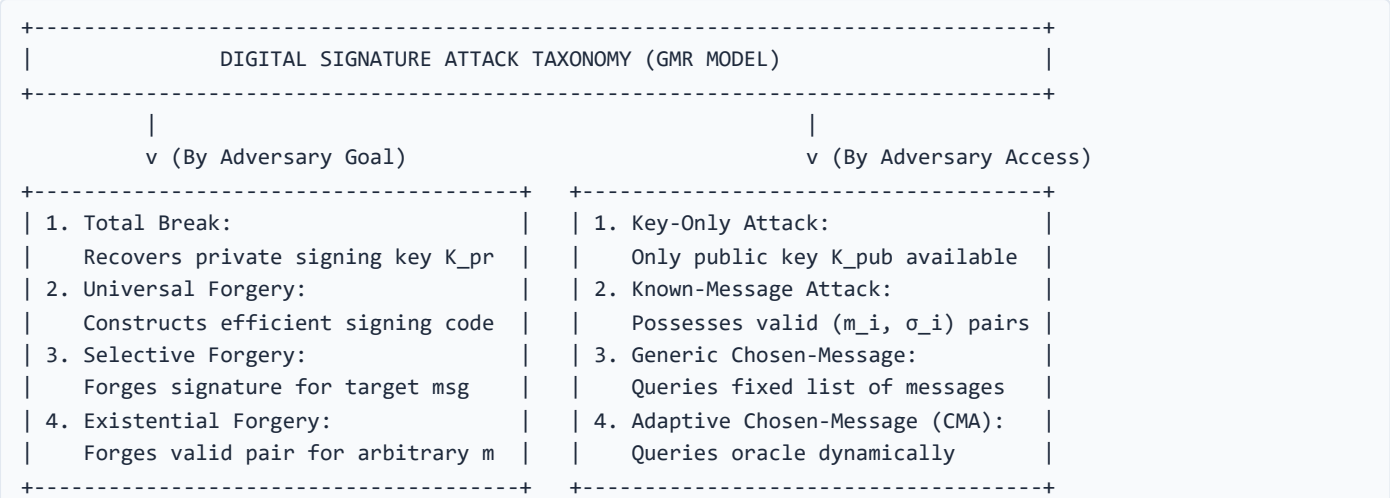
Q16. Explain the different types of attacks and forgery attacks on digital signatures. (6 Marks)

1. Core Concept & Formal Definition

Security analysis of digital signature schemes (KeyGen, Sign, Verify) is formalized under the Goldwasser-Micali-Rivest (GMR) model:

- **Two-Dimensional Attack Matrix:** Attacks are classified along two orthogonal axes: (1) *Adversary's Goal (Severity of Forgery)*, measuring compromised cryptographic capability, and (2) *Adversary's Access Level*, measuring oracle querying opportunities.
- **Security Benchmark:** A signature scheme must achieve **Existential Unforgeability under Adaptive Chosen-Message Attacks (EUF-CMA)**, ensuring no polynomial-time adversary can forge a valid signature for an unqueried message.

2. Attack Taxonomy Architecture



3. Core Dimensions & Features

- **Total Break:** The adversary deduces private key K_{pr} (e.g., factoring RSA modulus $n = pq$), gaining full authority to forge signatures on any document.
- **Universal Forgery:** The adversary constructs an algorithm equivalent to the signing function without discovering K_{pr}, signing arbitrary messages.
- **Selective Forgery:** The adversary produces a valid signature for a specific, pre-selected message chosen prior to the attack.
- **Existential Forgery:** The adversary creates at least one valid pair (m, σ) without signer authorization, even if m is arbitrary or nonsensical (e.g., textbook RSA homomorphic exploit).

4. Summary Table of Attacks

Attack / Forgery	Severity	Mechanism / Attack Vector	Standard Defense
Total Break	Critical	Mathematical factorization ($n = pq$)	Sufficient key length (≥ 2048-bit RSA)
Universal Forgery	Severe	Algorithmic inversion without key	Provably secure reductions
Selective Forgery	High	Collision or preimage exploitation	Cryptographic hashing (SHA-256)
Existential Forgery	Moderate	Homomorphic algebraic relations	Randomized padding (RSA-PSS)
Adaptive CMA	Severe	Iterative signing oracle queries	Deterministic nonces (RFC 6979)

MODEL QUESTION PAPER – 4

Q11. Explain the importance and need for information security.

1. Core Concept

According to NIST SP 800-12, information security preserves the confidentiality, integrity, and availability of information systems. Pervasive connectivity and escalating cyber threats make robust security essential to safeguard organizational assets, maintain regulatory compliance, and ensure stakeholder trust.

2. Information Security Architecture



3. Key Needs and Objectives

- **Protecting Confidentiality:** Prevents unauthorized disclosure of proprietary data, intellectual property, and personal records using encryption algorithms.

- **Preserving Integrity:** Guarantees transmitted and stored data remains unaltered and authentic via cryptographic hashes and digital signatures.
- **Ensuring Availability:** Safeguards network bandwidth, systems, and servers against denial-of-service disruptions and ransomware lockouts.
- **Regulatory Compliance:** Satisfies mandatory legal obligations under GDPR, HIPAA, and IT governance laws, preventing severe financial fines.
- **Business Continuity:** Averts catastrophic operational downtime, financial liability, and brand reputation loss caused by security breaches.

4. Security Dimensions Matrix

Security Dimension	Primary Objective	Threat Mitigated	Standard Control
Confidentiality	Prevent unauthorized disclosure.	Eavesdropping, packet sniffing.	AES encryption, access control.
Integrity	Ensure data authenticity.	Tampering, message alteration.	SHA-256 hashing, HMAC signatures.
Availability	Ensure uninterrupted access.	DoS attacks, server outages.	Redundancy, load balancing.

Q12. Explain the three versions of AES: AES-128, AES-192 and AES-256.

1. Core Concept

Standardized by NIST in FIPS PUB 197, AES is a symmetric block cipher based on Rijndael. It encrypts 128-bit blocks across three versions differing in key length, round count, and expanded key words.

2. AES Encryption Processing Flow

```
128-bit Plaintext ⟶ [Initial Round] ⟶ [Rounds 1 to Nr-1] ⟶ [Final Round] ⟶ 128-bit Ciphertext
```

3. Key Dimensions

- **Block Size:** All versions process 128-bit blocks organized as a 4x4 byte State array.
- **Key Lengths:** Employs keys of 128 bits (16 bytes), 192 bits (24 bytes), or 256 bits (32 bytes).
- **Round Counts:** Number of transformation rounds: 10 (AES-128), 12 (AES-192), and 14 (AES-256).
- **Round Keys:** Generates 44 words (11 keys), 52 words (13 keys), and 60 words (15 keys).
- **Security Level:** AES-128 suits commercial traffic; AES-256 provides top-secret and quantum protection.

4. AES Specification Matrix

Version	Key Size	Rounds	Expanded Words	Application
AES-128	128 bits	10	44 words (11 keys)	Standard web traffic.
AES-192	192 bits	12	52 words (13 keys)	Corporate networks.
AES-256	256 bits	14	60 words (15 keys)	Military defense.

Q13. Explain the round-key generation procedure in DES.

1. Core Concept

In DES (NIST FIPS PUB 46-3), the key schedule derives sixteen 48-bit subkeys (K1 to K16) from a 64-bit master key to parameterize each Feistel round.

2. DES Key Schedule Architecture

```
64-bit Key ⟶ [PC-1] ⟶ (C0, D0: 28b each) ⟶ [Circular Left Shifts] ⟶ [PC-2] ⟶ 48-bit Ki
```

3. Procedural Steps

- **Parity Drop (PC-1):** Strips 8 parity bits, yielding 56 bits split into 28-bit registers C0 and D0.

- **Left Shifts:** Registers C and D shift left by 1 bit (rounds 1, 2, 9, 16) or 2 bits (other 12 rounds).
- **Restoration:** Cumulative shift is 28 bits per register, restoring registers after round 16.
- **Compression (PC-2):** Compresses 56 bits into 48-bit subkey Ki by dropping 8 bits.
- **Decryption Order:** Decryption uses identical subkeys in reverse order (K16 down to K1).

4. Transformation Matrix

Stage	Input Size	Output Size	Operation
PC-1	64 bits	56 bits	Drops 8 parity bits.
Shift-1	28 bits × 2	28 bits × 2	Left shift by 1 bit.
Shift-2	28 bits × 2	28 bits × 2	Left shift by 2 bits.
PC-2	56 bits	48 bits	Selects 48-bit Ki.

Q14. Explain the different methods of integer factorization.

1. Core Concept

Integer factorization decomposes a composite integer n into non-trivial prime factors (n = p * q). In asymmetric cryptography (e.g., RSA), system security depends on the computational hardness of factoring large integers.

2. Factorization Algorithms Taxonomy

```
Composite Modulus (n) ──▶ [Special-Purpose: Trial Div / Fermat / Pollard] OR [General-Purpose: QS / GNFS]
```

3. Key Factorization Algorithms

- **Trial Division:** Tests prime divisors up to sqrt(n); practical only for small prime factors (O(sqrt(p))).
- **Fermat's Method:** Expresses n = x^2 - y^2; factors rapidly when primes p and q are close to sqrt(n).
- **Pollard's Rho:** Uses cycle detection on sequences x_(k+1) = (x_k^2 + c) mod n in expected O(p^(1/2)) time.
- **Pollard's p-1:** Exploits Fermat's Little Theorem when p-1 is smooth, computing gcd(a^(B!) - 1, n).
- **Number Field Sieve:** Fastest general-purpose classical algorithm; runs in sub-exponential time, requiring RSA keys >= 2048 bits.

4. Factorization Methods Matrix

Method	Category	Time Complexity	Optimal Target
Trial Division	Special-purpose	O(sqrt(p))	Small prime divisors (p < 10^6).
Fermat's Method	Special-purpose	O(abs(x - sqrt(n)))	Primes p, q close to sqrt(n).
Pollard's Rho	Special-purpose	O(p^(1/2))	Medium factors (p <= 2^60).
GNFS	General-purpose	Sub-exponential	Standard RSA moduli (>= 2048 bits).

Q15. Explain the structure and important fields of an X.509 digital certificate.

1. Core Concept

Defined in ITU-T X.509 and RFC 5280, an X.509 digital certificate is an ASN.1 structured credential binding an entity's identity to their public key via a trusted Certificate Authority (CA) digital signature.

2. X.509 Certificate Structure

```
Certificate ──▶ [TBSCertificate (Fields & Extensions)] ──▶ [Algorithm ID] ──▶ [CA Digital Signature]
```

3. Core Structural Fields

- **TBSCertificate:** The core To-Be-Signed data block holding subject details, public key, and validity bounds.

- **Version & Serial Number:** Indicates certificate version (v3) and a CA-unique serial integer for revocation tracking.
- **Issuer & Subject DNs:** X.500 Distinguished Names identifying the issuing CA and the certified key holder.
- **Validity Period:** Encapsulates NotBefore and NotAfter timestamps defining the legal operational lifespan.
- **v3 Extensions:** Encapsulates critical attributes including KeyUsage (permitted operations) and SubjectAlternativeName (SAN).

4. Important Fields Matrix

Field	ASN.1 Element	Purpose	Example
Version	`version`	Syntax specification.	v3 (integer 2).
Serial Number	`serialNumber`	Revocation management.	Unique CA identifier.
Issuer DN	`issuer`	Signing CA identity.	CN=DigiCert CA.
Validity	`validity`	Operating timeframe.	NotBefore / NotAfter.
Signature	`signatureValue`	CA digital signature.	Encrypted hash digest.

Q16. Explain the signing and verification procedures of the Elliptic Curve Digital Signature Algorithm (ECDSA).

1. Core Concept

Standardized in ANSI X9.62, ECDSA provides digital signatures matching RSA-3072 security, based on the hardness of the Elliptic Curve Discrete Logarithm Problem (ECDLP).

2. ECDSA Operation Workflow

```
Signing: (m, d) ⟶ Point P = k*G ⟶ Signature (r, s)
Verification: (Signature, m, Q) ⟶ Point P' = u1*G + u2*Q ⟶ Check: x1' mod n == r
```

3. Key Procedural Steps

- **Domain Setup:** Curve E(F_p) with base point G of order n; public key is Q = d*G.
- **Nonce Selection:** Signer picks random integer k in [1, n-1]; reusing k exposes private key d.
- **Signature Generation:** Computes point P = k*G = (x1, y1), r = x1 mod n, and s = k^(-1)*(H(m) + d*r) mod n.
- **Point Reconstruction:** Verifier computes w = s^(-1) mod n and point P' = u1*G + u2*Q.
- **Verification Check:** Accepts signature if x1' mod n equals r, validating authenticity without exposing d.

4. Procedural Summary Matrix

Step	Signer Operation	Verifier Operation
Digest	Compute e = H(m).	Compute e = H(m).
Nonce	Pick random secret k.	Compute w = s^(-1) mod n.
Point	Point P = k * G.	Point P' = u1 * G + u2 * Q.
Tuple	Output (r, s).	Verify if x1' mod n == r.

MODEL QUESTION PAPER – 5

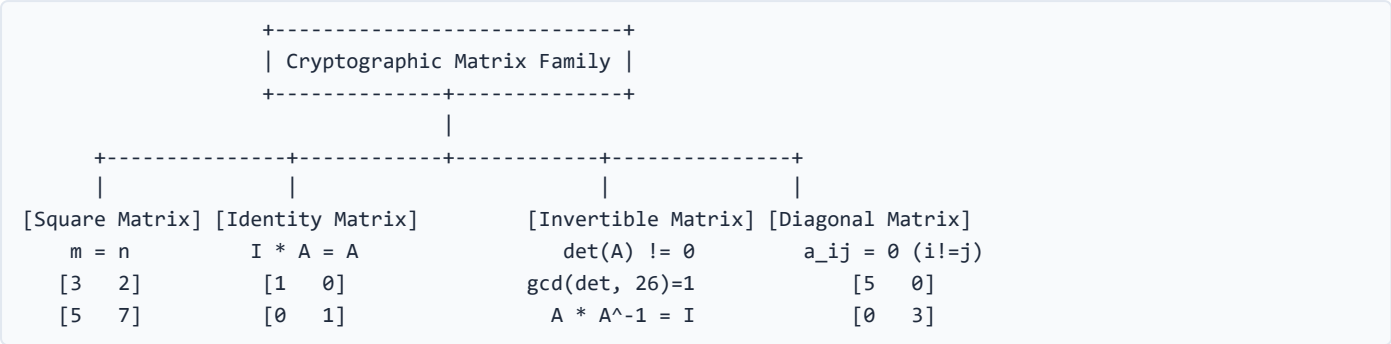
Q11. Explain the different types of matrices with suitable examples.

1. Core Concept & Formal Definition

A **matrix** is a rectangular array of numbers or algebraic expressions arranged in *m* rows and *n* columns (order *m* × *n*). In cryptography, matrices defined over modular rings ($\mathbb{Z}_{26}$) and finite Galois fields (GF(2^8)) are essential for polygraphic

substitution (Hill cipher) and linear diffusion (AES MixColumns).

2. Cryptographic Matrix Taxonomy



3. Core Dimensions & Primary Matrix Types

- **Square Matrix:** Possesses equal row and column counts ($m = n$). Functions as linear transformation operators, such as the Hill cipher key matrix `K = [[3, 2], [5, 7]]`.
- **Identity Matrix (I):** A square matrix having ones along the main diagonal and zeros elsewhere, satisfying `A × I = I × A = A`; serves as the neutral multiplicative baseline.
- **Invertible (Non-Singular) Matrix:** A square matrix whose determinant is coprime to the modular base (`gcd(det(A), 26) = 1` in $\mathbb{Z}_{26}$), guaranteeing an inverse A^{-1} for decryption.
- **Diagonal Matrix:** All off-diagonal elements are zero ($a_{ij} = 0$ for $i \neq j$), enabling decoupled element-wise modular multiplication.
- **Symmetric & Transpose Matrix:** The transpose A^T swaps rows with columns; a symmetric matrix satisfies $A = A^T$, used in reversible key-mixing operations.

4. Summary Comparison & Applications Table

Matrix Type	Defining Mathematical Property	Cryptographic Role	Concrete Example
Square	Row count equals column count ($m = n$)	Block cipher key spaces	`K = [[3, 2], [5, 7]]`
Identity	$a_{ii} = 1$, $a_{ij} = 0$ ($i \neq j$)	Neutral operation verification	$I_2 = [[1, 0], [0, 1]]$
Invertible	`det(A) ≠ 0` and `gcd(det(A), m) = 1`	Decryption key derivation	$K \times K^{-1} \equiv I \pmod{26}$
Diagonal	$a_{ij} = 0$ for all $i \neq j$	Decoupled parallel scaling	`diag(5, 3)` in GF(2^8)

Q12. Solve the following linear congruence using the multiplicative inverse method: 7x ≡ 9 (mod 26).

1. Core Concept & Solvability Criterion

A linear congruence of the form $ax \equiv b \pmod{m}$ admits a unique solution modulo m if and only if $\gcd(a, m) = 1$. Under the **multiplicative inverse method**, we find the modular inverse $a^{-1} \pmod{m}$ such that $a \times a^{-1} \equiv 1 \pmod{m}$, and multiply both sides to obtain $x \equiv a^{-1} \times b \pmod{m}$.

2. Algorithmic Solution Architecture

```
[Given: 7x ≡ 9 (mod 26)]
          |
          v
[Check gcd(7, 26) = 1] --------------> [1 | 9 => Unique Solution Exists]
          |
          v
[Extended Euclidean Algorithm] ------> [Backtracking yields 7^-1 ≡ 15 mod 26]
          |
          v
[Multiply: x ≡ 9 * 15 (mod 26)] -----> [x = 135 mod 26 = 5]
          |
          v
[Verification: 7 * 5 = 35 ≡ 9 mod 26] -> [Confirmed Unique Solution: x = 5]
```

3. Step-by-Step Mathematical Solution

- **Step 1: Check Solvability:** Compute `gcd(7, 26)` using the Euclidean division algorithm:

  ```
  26 = 7(3) + 5
  7 = 5(1) + 2
  5 = 2(2) + 1
  ```

  Since `gcd(7, 26) = 1` and `1 | 9`, a unique incongruent solution exists in $\mathbb{Z}_{26}$.

- **Step 2: Find Multiplicative Inverse (7^{-1} mod 26):** Backtrack using Extended Euclidean relations:

  ```
  1 = 5 - 2(2)
  1 = 5 - 2[7 - 5(1)] = 3(5) - 2(7)
  1 = 3[26 - 7(3)] - 2(7) = 3(26) - 11(7)
  ```

  Reducing modulo 26: `-11(7) ≡ 1 (mod 26)`. Because `-11 ≡ 15 (mod 26)`, we have **$7^{-1} \equiv 15 \pmod{26}$** [check: `7 × 15 = 105 = 4(26) + 1`].

- **Step 3: Multiply Both Sides by Inverse:**

  ```
  x ≡ 9 × 15 (mod 26)  &implies;  x ≡ 135 (mod 26)
  ```

  Dividing 135 by 26: `135 = 26(5) + 5`. Hence, **x ≡ 5 (mod 26)**.

- **Step 4: Formal Verification:** Substitute `x = 5` into the original congruence:

  ```
  7(5) = 35 = 26(1) + 9 ≡ 9 (mod 26)
  ```

Verified Solution: x ≡ 5 (mod 26)

4. Summary Computation Table

Stage	Formula / Operation	Computed Value	Mathematical Role
Solvability Test	`gcd(7, 26)` via Euclidean division	`gcd = 1`	Guarantees unique solution
Modular Inverse	Extended Euclidean back-substitution	$7^{-1} \equiv 15 \pmod{26}$	Cancels coefficient 7
Root Extraction	`x ≡ 9 × 15 = 135 mod 26`	`x = 5`	Primary root in $\mathbb{Z}_{26}$
Verification	`7(5) = 35 mod 26`	`35 ≡ 9 (mod 26)`	Confirms algebraic correctness

> **Q13. Explain the Rail Fence Cipher. Encrypt the plaintext HELLOWORLD using a two-rail Rail Fence**

Cipher.

1. Core Concept & Formal Definition

The **Rail Fence Cipher** (zigzag cipher) is a classical transposition cipher in which plaintext characters are written diagonally downward and upward across *k* parallel imaginary lines ("rails") in an oscillating wave. Ciphertext is subsequently extracted by reading row by row from top to bottom, altering letter positions without changing letter identity.

2. Two-Rail Zigzag Transposition Diagram

```
Plaintext Index:   0   1   2   3   4   5   6   7   8   9
Plaintext Char:    H   E   L   L   O   W   O   R   L   D

Rail 1 (Even):     H . . . L . . . O . . . O . . . L .   -->  H L O O L
                     .   /   .   /   .   /   .   /   .
Rail 2 (Odd):      . E . . . L . . . W . . . R . . . D   -->  E L W R D

Extracted Ciphertext: H L O O L E L W R D
```

3. Encryption & Decryption Execution Steps

- **Plaintext Partitioning:** Plaintext `P = "HELLOWORLD"` contains `N = 10` characters. For depth `k = 2`:
  • Rail 1 receives even index positions (0, 2, 4, 6, 8): **H, L, O, O, L** (5 characters).
  • Rail 2 receives odd index positions (1, 3, 5, 7, 9): **E, L, W, R, D** (5 characters).
- **Ciphertext Generation:** Concatenating Rail 1 followed by Rail 2 yields: **Ciphertext: HLOOLELWRD**
- **Decryption Process:** Total length `N = 10`, depth `k = 2`:
  • Rail 1 length = `⌈10/2⌉ = 5` characters: `[H, L, O, O, L]`.
  • Rail 2 length = `⌊10/2⌋ = 5` characters: `[E, L, W, R, D]`.
  • Interleave alternate characters: R1[0], R2[0], R1[1], R2[1], ... &implies; H-E-L-L-O-W-O-R-L-D = **HELLOWORLD**.
- **Cryptanalytic Vulnerability:** Character frequencies remain unaltered; susceptible to frequency analysis and brute-force key search due to small keyspace ($k \ll N$).

4. Summary Transposition Table

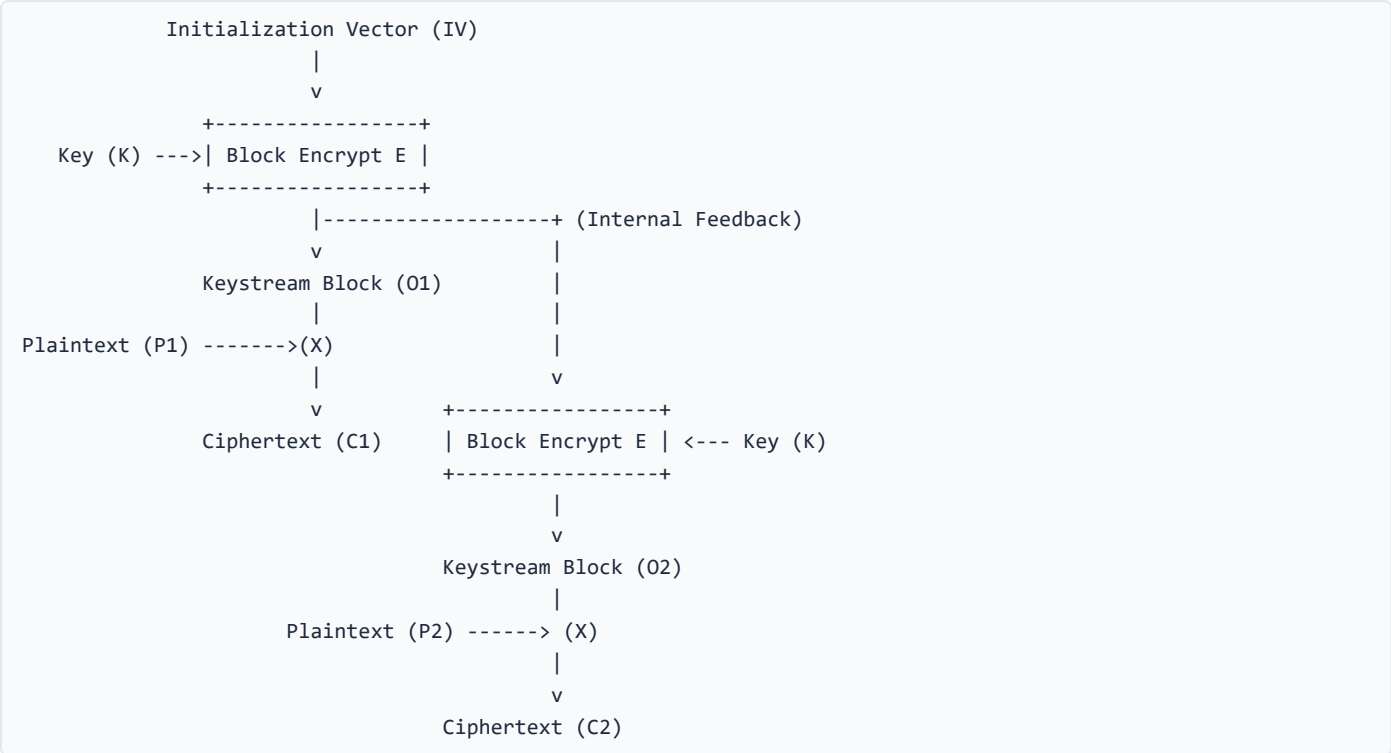
Rail Level	Assigned Index Positions	Character Stream	Segment Length
Rail 1 (Top)	0, 2, 4, 6, 8	H, L, O, O, L	5 characters
Rail 2 (Bottom)	1, 3, 5, 7, 9	E, L, W, R, D	5 characters
Ciphertext	Rail 1 + Rail 2 concatenation	`HLOOLELWRD`	10 characters
Plaintext	Interleaved rail reconstruction	`HELLOWORLD`	10 characters

Q14. Explain the working of Output Feedback (OFB) mode with a neat diagram.

1. Core Concept & Formal Definition

Output Feedback (OFB) mode is an authenticated confidentiality mode of operation (standardized in NIST SP 800-38A) that converts a symmetric block cipher into a synchronous stream cipher. It generates a continuous sequence of pseudo-random keystream blocks by iteratively encrypting the previous cipher output, which is subsequently bitwise XORed with plaintext blocks.

2. Architecture & Execution Flow Diagram



3. Core Dimensions & Operational Features

- **Keystream Generation:** The keystream is created recursively: $O_0 = IV$, $O_j = E_K(O_{j-1})$. Generation is completely decoupled from plaintext and ciphertext streams.
- **Bitwise XOR Symmetry:** Encryption computes $C_j = P_j \oplus O_j$; decryption performs identical XOR operations $P_j = C_j \oplus O_j$ using only the forward encryption function E_K.
- **Zero Error Propagation:** A single-bit transmission corruption in a ciphertext block damages only the corresponding decrypted plaintext bit; subsequent blocks decrypt without distortion.
- **Pre-computation Efficiency:** Keystream blocks can be generated offline prior to message availability, enabling ultra-low encryption latency for real-time traffic (VoIP, streaming).
- **Strict IV Freshness:** The IV must be unique for every transmission; reusing an IV under the same key allows attackers to recover XORed plaintexts ($C_1 \oplus C_2 = P_1 \oplus P_2$).

4. Summary Comparison & Specification Table

Operational Dimension	OFB Mode Specification	Cryptographic Benefit
Cipher Classification	Synchronous stream cipher emulator	Encipher arbitrary bit-lengths without padding
Decryption Engine	Forward block encryption E_K	Eliminates inverse decryption circuitry
Error Extension	Isolated to flipped bit (0-bit propagation)	Ideal for error-prone noisy satellite/RF links
Pre-processing	Full keystream offline generation allowed	Zero encryption delay on data transmission

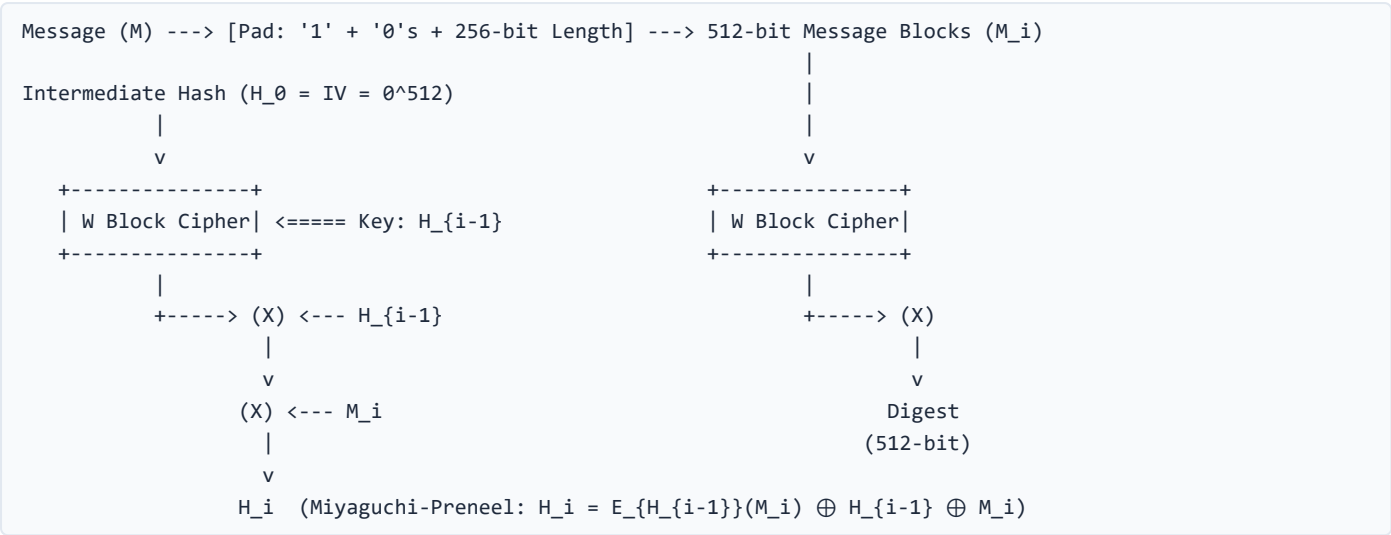
---

Q15. Explain the important features of the Whirlpool hash function.

1. Core Concept & Formal Definition

Whirlpool is an iterated cryptographic hash function designed by Vincent Rijmen and Paulo S. Barreto, standardized in ISO/IEC 10118-3. It operates on arbitrary message inputs of length up to $2^{256} - 1$ bits and produces a fixed 512-bit message digest using a dedicated 512-bit block cipher inside a Miyaguchi-Preneel compression scheme.

2. Architecture & Compression Pipeline Diagram



3. Core Dimensions & Important Features

- **Digest & State Structure:** Produces a 512-bit digest from 512-bit message blocks structured as an 8 × 8 byte matrix state over Galois field GF(2^8).
- **Internal W Block Cipher:** Implements a 10-round block cipher based on AES/Rijndael design principles, executing: `SubBytes` (8 × 8 S-box substitution), `ShiftColumns`, `MixRows` (MDS diffusion), and `AddRoundKey`.
- **Miyaguchi-Preneel Scheme:** Employs a three-way feed-forward compression: $H_i = E_{H_{i-1}}(M_i) \oplus H_{i-1} \oplus M_i$, offering provable security against collision attacks.
- **High Security Margin:** Delivers 256-bit collision resistance and 512-bit preimage resistance, possessing complete immunity against linear and differential cryptanalysis.
- **Optimum MDS Diffusion:** Employs an optimal Maximum Distance Separable circulant matrix ensuring complete avalanche effect across all 64 state bytes within two rounds.

4. Technical Specifications Table

Architectural Parameter	Whirlpool Specification	Cryptographic Significance
Output Digest Size	512 bits (64 bytes)	Equal to SHA-512, standardized under ISO/IEC 10118-3
Message Block Size	512 bits (arranged as 8 × 8 state)	Processed iteratively via Miyaguchi-Preneel compression
Internal Rounds	10 iterative rounds	Square-cipher lineage with optimal MDS matrix diffusion
Security Resistance	256-bit collision, 512-bit preimage	Robust resistance against all known analytical attacks

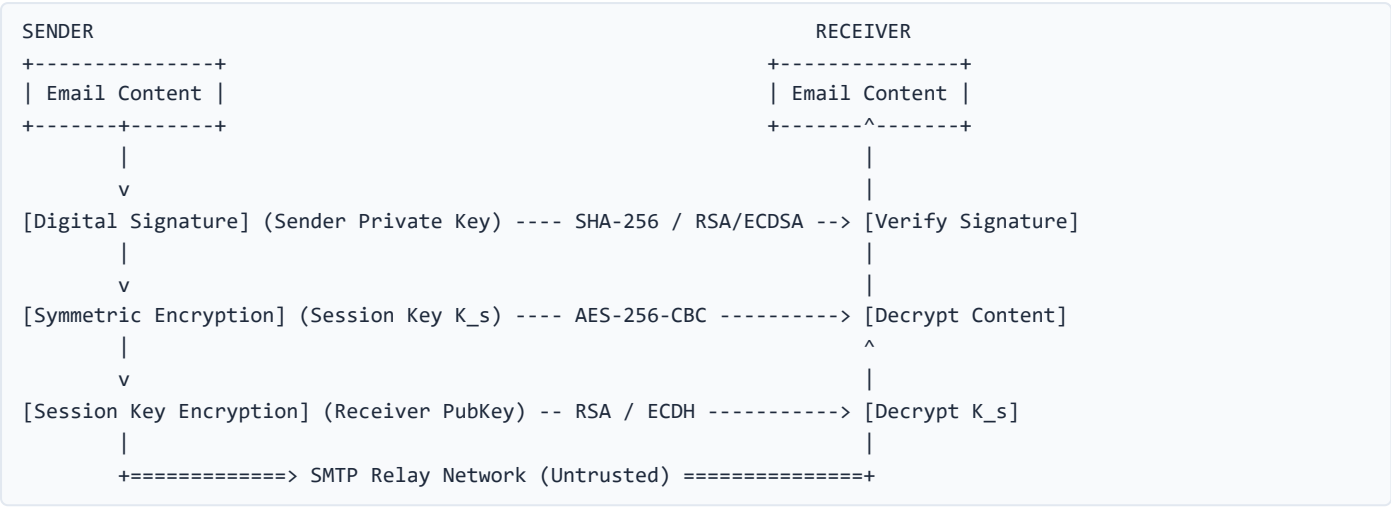
---

Q16. Explain the security services provided by S/MIME.

1. Core Concept & Formal Definition

Secure/Multipurpose Internet Mail Extensions (S/MIME), standardized in RFC 8551, is an IETF industry standard protocol providing end-to-end cryptographic security services for electronic messaging. Built on the Cryptographic Message Syntax (CMS) and X.509 Public Key Infrastructure, it protects MIME message entities across untrusted intermediate SMTP relays.

2. Architecture & Security Processing Flow Diagram



3. Core Dimensions & Security Services

- **Confidentiality (Privacy):** Prevents unauthorized interception by encrypting message content with an ephemeral symmetric session key (AES-128/256), which is itself encrypted with each recipient's public key (CMS `EnvelopedData`).
- **Message Integrity:** Ensures message content cannot be altered or truncated undetected during transit by binding a cryptographic hash (SHA-256/512) to the signed payload.
- **Sender Authentication:** Confirms the sender's verified identity by embedding a digital signature created using the sender's private key and authenticated via an X.509 digital certificate.
- **Non-Repudiation:** Prevents the sender from falsely denying origin or authorship of sent messages, establishing legally enforceable proof through asymmetric signature validation.
- **Attribute Certification & Packaging:** Encapsulates encrypted/signed payloads into standard MIME structures (`multipart/signed`, `application/pkcs7-mime`), securing headers and timestamps.

4. Summary Service Specification Table

Security Service	Cryptographic Mechanism	CMS Object / Algorithm Standard
Confidentiality	Symmetric payload encryption + Key wrapping	`EnvelopedData` / AES-256, RSA-OAEP
Integrity	Cryptographic message hashing	`SignedData` / SHA-256, SHA-512
Authentication	Sender private key digital signature	`SignedData` / RSA, ECDSA, Ed25519
Non-Repudiation	X.509 PKI certificate chain binding	X.509 v3 hierarchical CA verification

---

PART III: SECTION C — 8 MARKS QUESTIONS (PAPERS 1 TO 5)

MODEL QUESTION PAPER – 1

Q17. Explain passive and active attacks with their types and suitable examples.

1. Core Concept & Formal Definition

In network security (RFC 4949 / William Stallings), a **security attack** is any action that compromises the integrity, confidentiality, or availability of organizational information assets.

- **Passive Attack:** An unauthorized entity intercepts or monitors communication channels without modifying system resources or message contents. The primary objective is to obtain transmitted data. Because passive attacks leave no audit trail and do not disrupt operations, detection is exceptionally difficult; defense relies primarily on **prevention** via strong cryptographic encryption.
- **Active Attack:** An unauthorized entity alters, injects, delays, or fabricates data streams or system resources. These attacks violate integrity and availability. Because prevention across open networks is rarely 100% effective, defense focuses on **detection**, **mitigation**, and **recovery**.

2. Architecture & Execution Flowchart

```
[ PASSIVE ATTACK: Eavesdropping & Monitoring ]
Sender Alice ------------ Transmission ------------> Receiver Bob
                               |
                               v (Intercepts / Reads)
                          Attacker Eve (No data change)

[ ACTIVE ATTACK: Tampering & Impersonation ]
Sender Alice ----> [ Attacker Mallory: Alters / Drops / Forges ] ----> Receiver Bob
```

3. Detailed Operational Activities & Processing Phases

- **Release of Message Contents:** Intercepting confidential plaintext transmissions (e.g., e-mails or financial transfers) over vulnerable links, violating confidentiality.
- **Traffic Analysis:** Monitoring encrypted communication patterns, traffic frequency, and packet burst lengths to deduce operational intelligence without decrypting contents.
- **Masquerade:** An attacker fraudulently impersonates an authorized entity using forged credentials to gain unauthorized system access.
- **Replay Attack:** Capturing valid network data transmissions and maliciously retransmitting them later to execute duplicate unauthorized transactions.
- **Modification of Messages:** Intercepting legitimate transit data to alter, delay, reorder, or inject falsified content, destroying data integrity.
- **Denial of Service (DoS):** Flooding communication bandwidth or crashing system services to prevent authorized users from accessing critical resources.

4. Summary Comparison & Specification Parameters

Dimension	Passive Attacks	Active Attacks
Data Impact	Zero modification; data remains untouched	Data altered, injected, deleted, or blocked
Security Goal	Breaches confidentiality and privacy	Breaches integrity, availability, authenticity
Detection	Very difficult (leaves no network traces)	Feasible (detected via logs and hashes)
Primary Defense	Prevention via robust encryption (AES)	Detection via MACs, digital signatures, IDS
Examples	Packet sniffing (Wireshark), wiretapping	Man-in-the-Middle, replay, TCP SYN flood

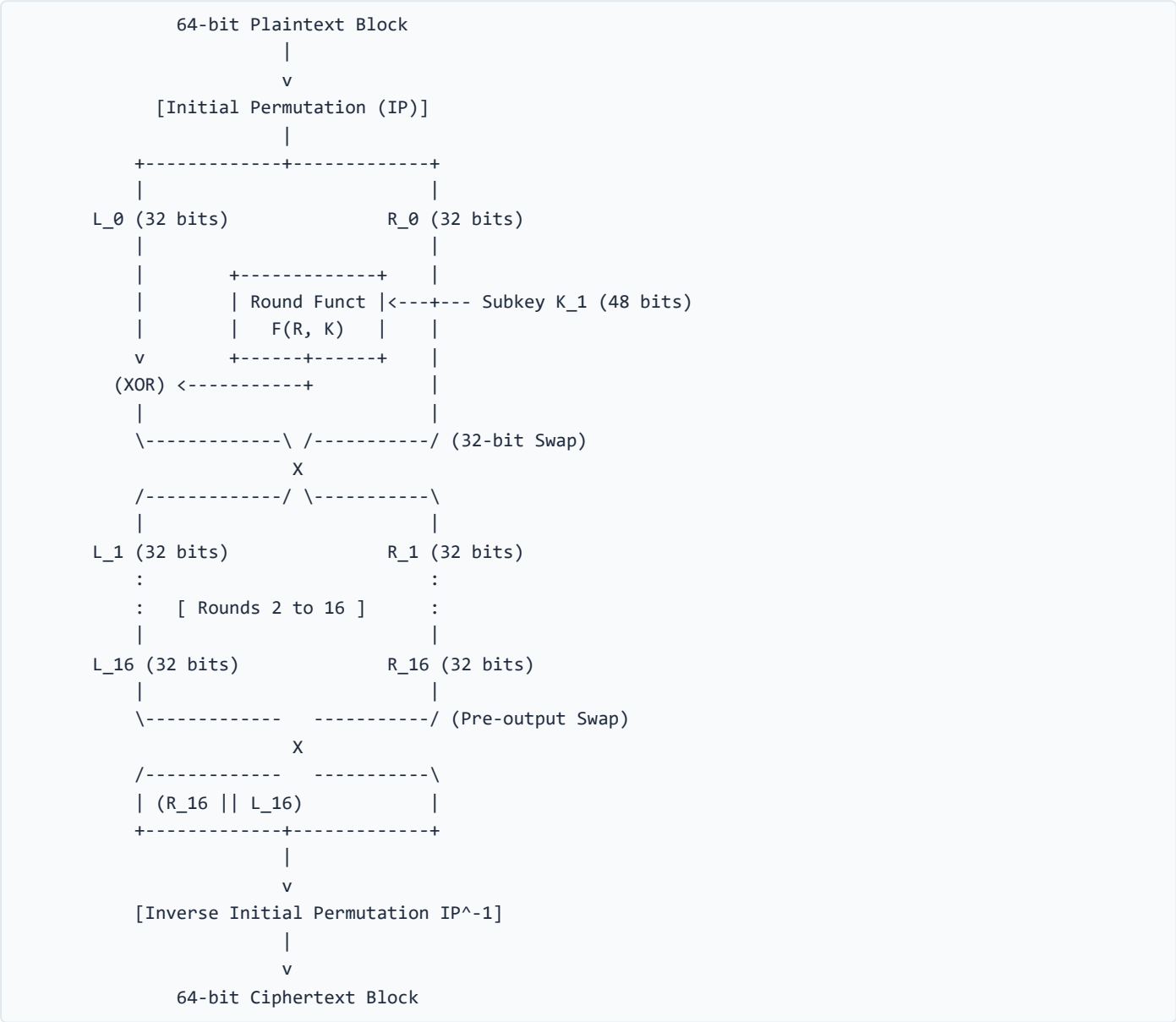
Q18. Explain the DES encryption process with a neat diagram.

1. Core Concept & Formal Definition

The **Data Encryption Standard (DES)** (NIST FIPS PUB 46-3) is a symmetric block cipher that enciphers 64-bit plaintext blocks into 64-bit ciphertext blocks using a 56-bit effective key via a 16-round **Feistel network**.

- **Block & Key Sizes:** Plaintext and ciphertext blocks are fixed at 64 bits. The supplied 64-bit key contains 8 parity bits (bits 8, 16, ..., 64) discarded during initial key permutation (PC-1), leaving an effective 56-bit key.
- **Feistel Property:** Encryption and decryption utilize identical algorithms; decryption merely applies the sixteen 48-bit round subkeys in reverse order (K_{16} down to K_1).

2. Architecture & Execution Flowchart



3. Detailed Operational Activities & Processing Phases

- **Initial and Final Permutation (IP and IP^{-1}):** Fixed bit transpositions that reorder the 64 input bits before Round 1 and reverse the layout after Round 16. These matrices provide no cryptographic secrecy.
- **Round Halving and Swapping:** In round i, the 64-bit block is split into 32-bit halves: $L_i = R_{i-1}$ and $R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$, with halves swapped after each round except Round 16.
- **Expansion Permutation (E-box):** Expands the 32-bit right half R_{i-1} to 48 bits by duplicating 16 bits, matching the length of round key K_i.
- **Key Mixing:** Computes a bitwise exclusive-OR (XOR) between the 48-bit expanded right half and 48-bit subkey K_i: $E(R_{i-1}) \oplus K_i$.

- **S-Box Substitution:** The non-linear core of DES. Eight S-boxes (S_1 to S_8) map 6-bit inputs to 4-bit outputs, compressing 48 bits to 32 bits and injecting confusion.
- **Permutation (P-box):** A fixed 32-bit transposition that diffuses S-box output bits across multiple S-boxes in the subsequent round.

4. Summary Comparison & Specification Parameters

```
Round Function: R_i = L_{i-1} ⊕ P(S(E(R_{i-1}) ⊕ K_i)),   L_i = R_{i-1}
Recombination: Ciphertext = IP^-1(R_16 || L_16)
```

Parameter	DES Standard Specification
Block Size	64 bits (Plaintext and Ciphertext)
Key Size	64-bit raw key (8 parity bits discarded) → 56-bit effective key
Rounds	16 identical Feistel iterations
Subkey Size	48 bits per round (derived via PC-1, circular shifts, PC-2)
S-Boxes	8 non-linear substitution boxes (each maps 6 bits → 4 bits)

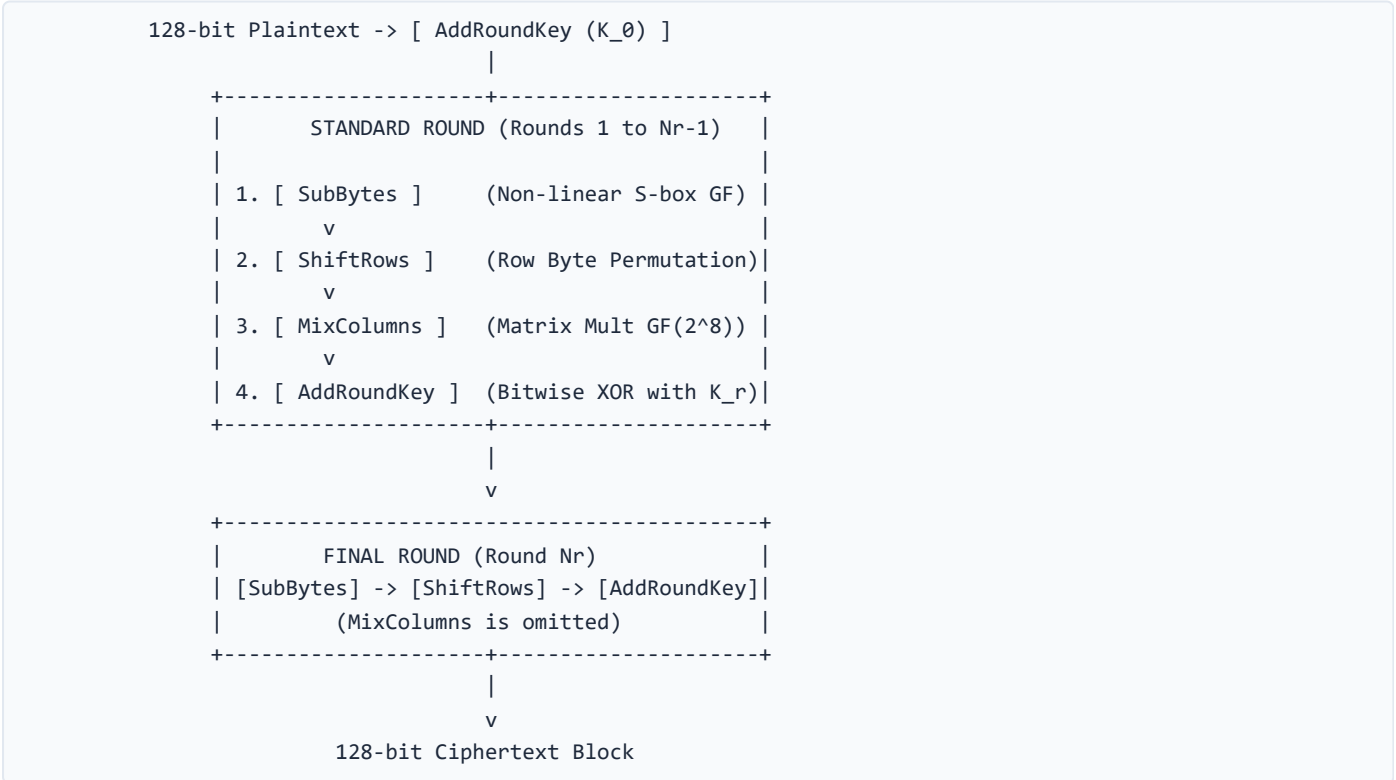
Q19. Explain the four main transformations performed during AES encryption.

1. Core Concept & Formal Definition

The **Advanced Encryption Standard (AES)** (NIST FIPS PUB 197 / Rijndael) is a symmetric block cipher based on a **Substitution-Permutation Network (SPN)**.

- **State Array:** Operates on a fixed 128-bit block size (16 bytes), structured as a 4×4 column-major matrix of bytes termed the **State**.
- **Rounds & Keys:** Executes 10 rounds (128-bit key), 12 rounds (192-bit key), or 14 rounds (256-bit key). Every round (except the final round, which omits MixColumns) applies four distinct algebraic transformations providing Shannon's confusion and diffusion.

2. Architecture & Execution Flowchart



3. Detailed Operational Activities & Processing Phases

- **SubBytes (Byte Substitution):** Replaces each byte of the State independently using an invertible 8-bit S-box constructed via multiplicative inversion in $GF(2^8)$ followed by an affine map over $GF(2)$. This non-linear mapping injects strong **confusion** against linear cryptanalysis.
- **ShiftRows (Row Cyclical Permutation):** Cyclically shifts the rows of the State to the left by differing byte offsets: Row 0 by 0, Row 1 by 1, Row 2 by 2, and Row 3 by 3 bytes. This distributes column bytes across distinct columns, providing intra-column **diffusion**.
- **MixColumns (Galois Field Mixing):** Multiplies each column of the State modulo $(x^4 + 1)$ with a fixed polynomial $c(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$ over $GF(2^8)$. Each output byte becomes a linear combination of all four column bytes, delivering inter-byte diffusion across rows. (Omitted in the final round N_r).
- **AddRoundKey (Key XOR):** Bitwise exclusive-ORs each byte of the State with a corresponding byte from the 128-bit round subkey generated by the AES key expansion schedule. It is self-inverse and incorporates the cipher key directly.

4. Summary Comparison & Specification Parameters

Transformation	Mathematical Basis	Cryptographic Role	Inverse Operation
SubBytes	Inversion in $GF(2^8)$ + Affine transformation	Confusion (Non-linear mapping)	InvSubBytes (Inverse S-box)
ShiftRows	Cyclical byte shift left by row index (r)	Intra-column Diffusion	InvShiftRows (Shift right)
MixColumns	Matrix multiplication modulo $(x^4 + 1)$ in $GF(2^8)$	Inter-byte Diffusion across columns	InvMixColumns (Mult by $d(x)$)
AddRoundKey	Bitwise XOR ($\oplus$) with 128-bit subkey	Key Dependency	AddRoundKey (Self-inverse)

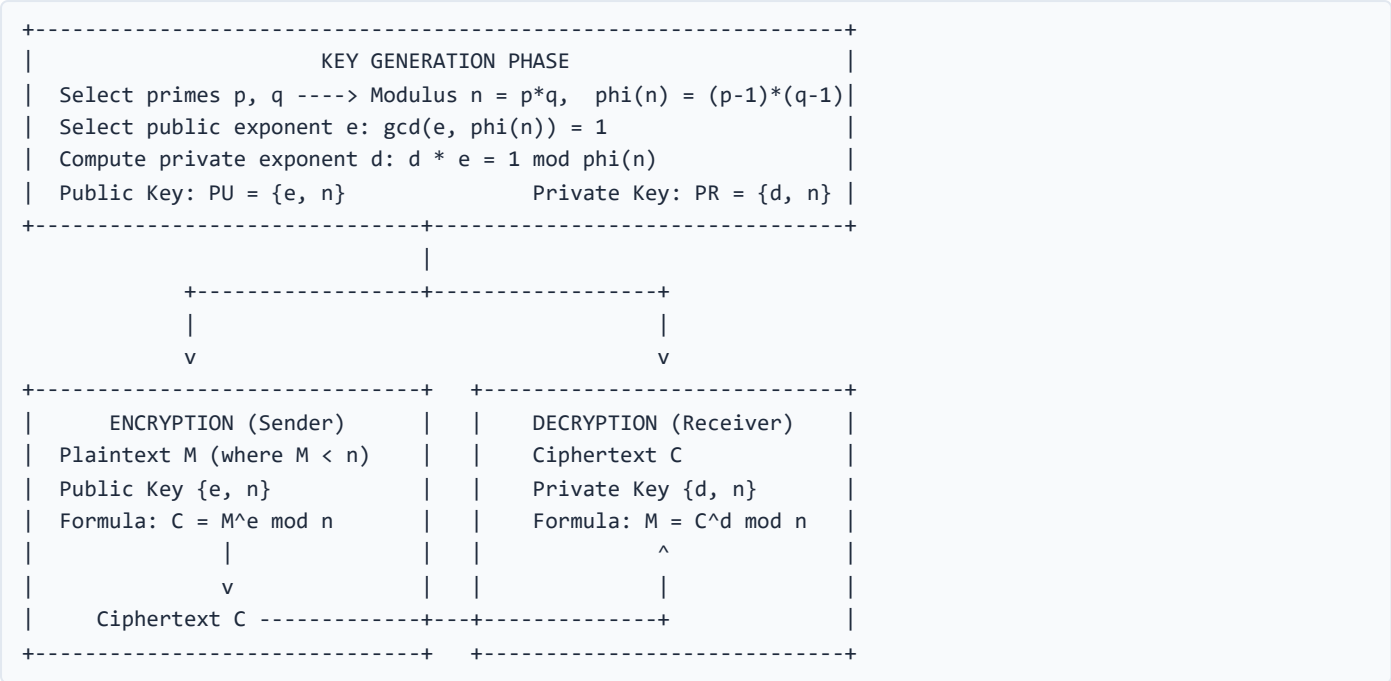
Q20. Explain the RSA cryptosystem with key generation, encryption and decryption procedures.

1. Core Concept & Formal Definition

The **RSA Cryptosystem** (Rivest, Shamir, and Adleman, 1977) is an asymmetric public-key algorithm foundational to modern digital security and e-commerce.

- **Computational Hardness:** Grounded in the **Integer Factorization Problem**: multiplying two large primes p and q to form composite modulus $n = p \times q$ is easy, but factoring n back into p and q is computationally infeasible for large keys (≥ 2048 bits).
- **Euler's Totient Theorem:** States that for any plaintext M coprime to n, $M^{\phi(n)} \equiv 1 \pmod n$, where $\phi(n) = (p - 1)(q - 1)$.

2. Architecture & Execution Flowchart



3. Detailed Operational Activities & Processing Phases

- **Step 1 (Prime Generation):** Select two large distinct primes p and q. Both primes must remain strictly confidential.
- **Step 2 (Modulus Computation):** Calculate composite modulus $n = p \times q$. The bit-length of n represents the RSA key length.
- **Step 3 (Totient Computation):** Compute Euler's totient: $\phi(n) = (p - 1)(q - 1)$.
- **Step 4 (Public Exponent Selection):** Choose integer e coprime to $\phi(n)$ such that $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$.
- **Step 5 (Private Key Inversion):** Calculate private exponent d via the Extended Euclidean Algorithm: $d \equiv e^{-1} \pmod{\phi(n)} \iff d \times e \equiv 1 \pmod{\phi(n)}$.
- **Step 6 (Encryption Execution):** The sender encrypts message M ($0 \le M < n$) into ciphertext C: $C = M^e \pmod n$.
- **Step 7 (Decryption Execution):** The receiver restores plaintext M using private key d: $M = C^d \pmod n$. Correctness: $C^d \equiv (M^e)^d \equiv M^{k\phi(n)+1} \equiv M \pmod n$.

4. Summary Comparison & Specification Parameters

> Walkthrough: Let p = 7, q = 11 ⇒ n = 77, ϕ(n) = 60. Select e = 7 (gcd(7, 60) = 1).
> Calculate d: 7 × 43 = 301 ≡ 1 (mod 60) ⇒ d = 43. Keys: PU = {7, 77}, PR = {43, 77}.
> Encrypt M = 9: C = 9^7 mod 77 = 4,782,969 mod 77 = 37.
> Decrypt C = 37: M = 37^43 mod 77 = 9 (Original plaintext restored).

Component	Notation	Mathematical Definition	Access
Primes	p, q	Two large prime numbers	Secret
Modulus	n	$n = p \times q$	Public
Euler Totient	$\phi(n)$	$\phi(n) = (p - 1)(q - 1)$	Secret
Public Key	PU	$\{e, n\}$ where $\gcd(e, \phi(n)) = 1$	Public
Private Key	PR	$\{d, n\}$ where $d \equiv e^{-1} \pmod{\phi(n)}$	Secret

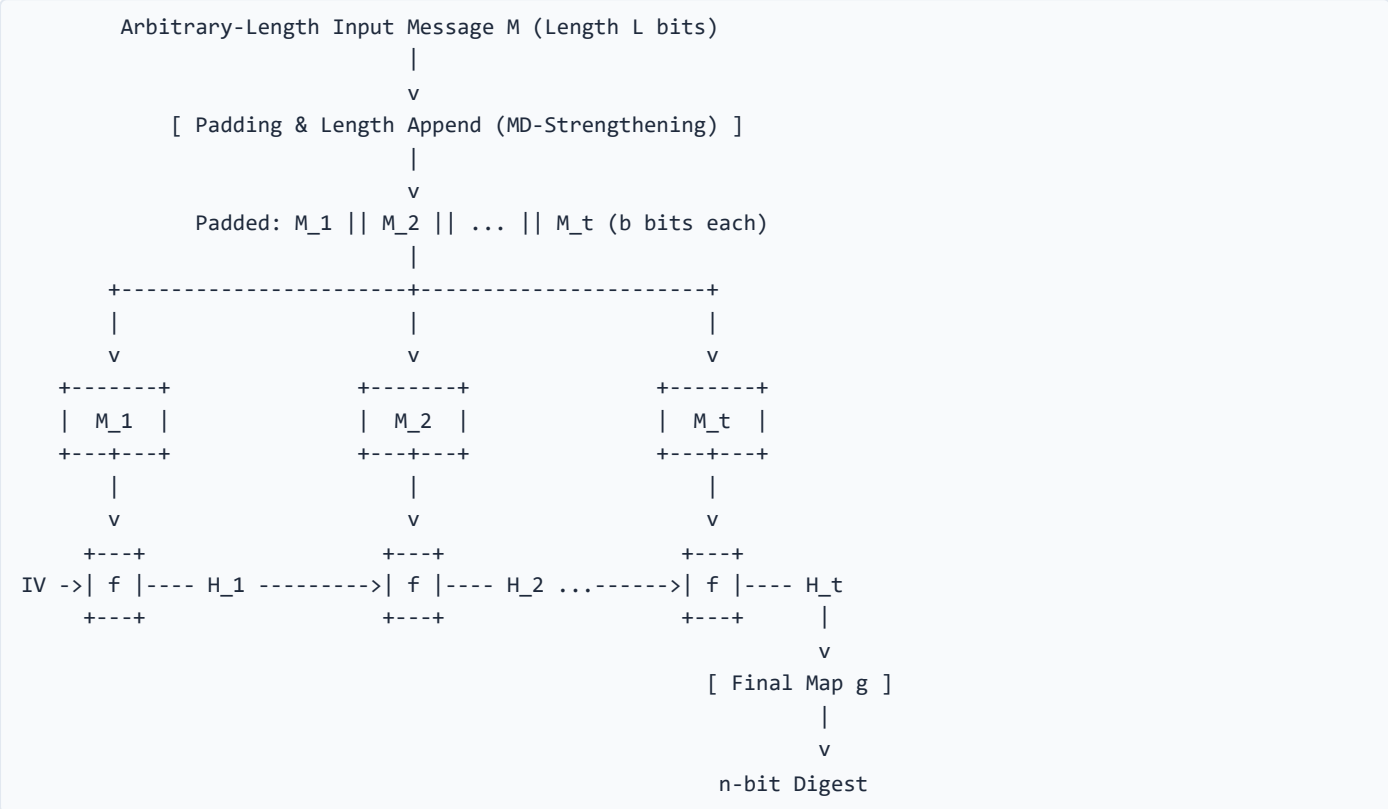
Q21. Explain the Merkle–Damgård scheme with a neat diagram.

1. Core Concept & Formal Definition

The **Merkle–Damgård construction** (Ralph Merkle and Ivan Damgård, 1979/1989) is an iterative framework for building collision-resistant cryptographic hash functions from fixed-size compression functions.

- **Security Reduction:** Proves that if the fixed-size compression function $f: \{0,1\}^n \times \{0,1\}^b \to \{0,1\}^n$ is collision-resistant, the resulting iterated hash function $H: \{0,1\}^* \to \{0,1\}^n$ is provably **collision-resistant**.
- **Standard Adoption:** Serves as the structural blueprint for standard hash functions including MD5, SHA-1, SHA-256, and SHA-512.

2. Architecture & Execution Flowchart



3. Detailed Operational Activities & Processing Phases

- **Padding & Length Hardening:** Appends a '1' bit, k zero bits, and a 64-bit integer encoding original bit-length L. This ensures the message is an exact multiple of block size b and prevents length-extension attacks.
- **Message Partitioning:** The padded bitstream is divided into t equal-sized blocks of b bits each: $M = M_1 \,\|\, M_2 \,\|\, \dots \,\|\, M_t$.
- **Chaining Initialization:** Initializes the recurrence using an Initialization Vector ($H_0 = \text{IV}$), an n-bit constant fixed in the specification.
- **Iterative Compression:** The compression function f takes intermediate state H_{i-1} and current block M_i to compute: $H_i = f(H_{i-1}, M_i)$ for $i = 1, \dots, t$.
- **Output Finalization:** An optional transformation $g(H_t)$ (identity mapping in SHA-256) outputs the final n-bit digest: $\text{Digest} = g(H_t)$.

4. Summary Comparison & Specification Parameters

> `Merkle-Damgård Recurrence: H_0 = IV,  H_i = f(H_{i-1}, M_i)  [1 ≤ i ≤ t],  Digest = g(H_t)`

Metric	SHA-256	SHA-512	MD5
Block Size (b)	512 bits	1024 bits	512 bits
Digest Size (n)	256 bits	512 bits	128 bits
Length Field	64-bit integer	128-bit integer	64-bit integer
Compression Logic	64 rounds of non-linear logic	80 rounds of non-linear logic	64 rounds of modular logic
Collision Resistance	Secure (No collisions)	Secure (No collisions)	Broken (Collisions found)

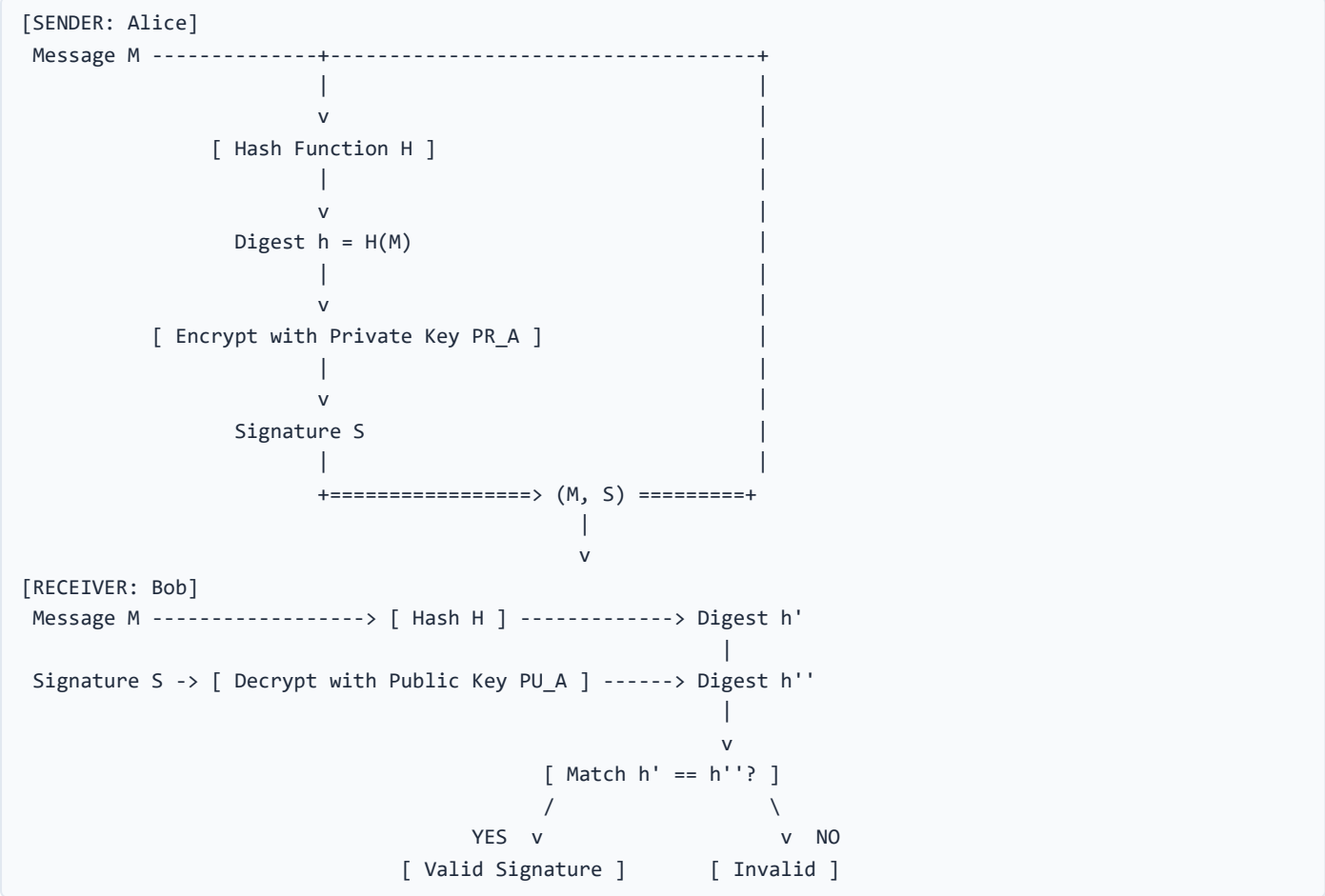
Q22. Explain the digital signature process using a message digest with a neat diagram.

1. Core Concept & Formal Definition

A **Digital Signature** (NIST FIPS PUB 186-4 / RFC 4949) is a cryptographic primitive that provides data authenticity, integrity verification, and non-repudiation.

- **Hash-Then-Sign Scheme:** Asymmetric encryption is computationally slow on large documents. To optimize efficiency, the sender signs a fixed-length cryptographic **message digest** $h = H(M)$ generated by a secure hash function (e.g., SHA-256) rather than the bulk document.
- **Security Guarantees:** Ensures **integrity** (detects transit tampering), **authenticity** (proves signer identity), and **non-repudiation** (signer cannot deny signing).

2. Architecture & Execution Flowchart



3. Detailed Operational Activities & Processing Phases

- **Step 1 (Digest Creation):** The sender passes message M into cryptographic hash function H (e.g., SHA-256) to produce fixed-size digest $h = H(M)$.
- **Step 2 (Signature Generation):** The sender encrypts digest h using their private key PR_A: $S = \text{Sign}_{PR_A}(h) = h^d \pmod n$.

- **Step 3 (Transmission):** The message M and signature S are transmitted together across the network as a bundle (M, S).
- **Step 4 (Local Hash Computation):** Upon receipt, the receiver hashes message M with the identical hash function H to obtain local digest $h' = H(M)$.
- **Step 5 (Signature Decryption):** The receiver decrypts signature S using the sender's public key PU_A: $h'' = \text{Verify}_{PU_A}(S) = S^e \pmod n$.
- **Step 6 (Verification Decision):** The receiver compares h' with h''. If $h' == h''$, the signature is valid, confirming integrity and authenticity.

4. Summary Comparison & Specification Parameters

```
Signing: S = (H(M))^d mod n,   where d ∈ PR_A (Private Key)
Verification: h' = H(M),   h'' = S^e mod n,   where e ∈ PU_A (Public Key)
Rule: Signature is VALID if and only if h' == h''.
```

Service	Cryptographic Enforcement	Mitigated Attack
Data Integrity	Hash comparison ($h' \stackrel{?}{=} h''$) reveals bit changes	Tampering and alteration in transit
Authentication	Sender's public key PU_A validates private key origin	Impersonation and spoofing
Non-Repudiation	Private key PR_A is unique to the sender	False denial of document authorship
Efficiency	Asymmetric operations restricted to short hash digest	Computational overhead on large files

---

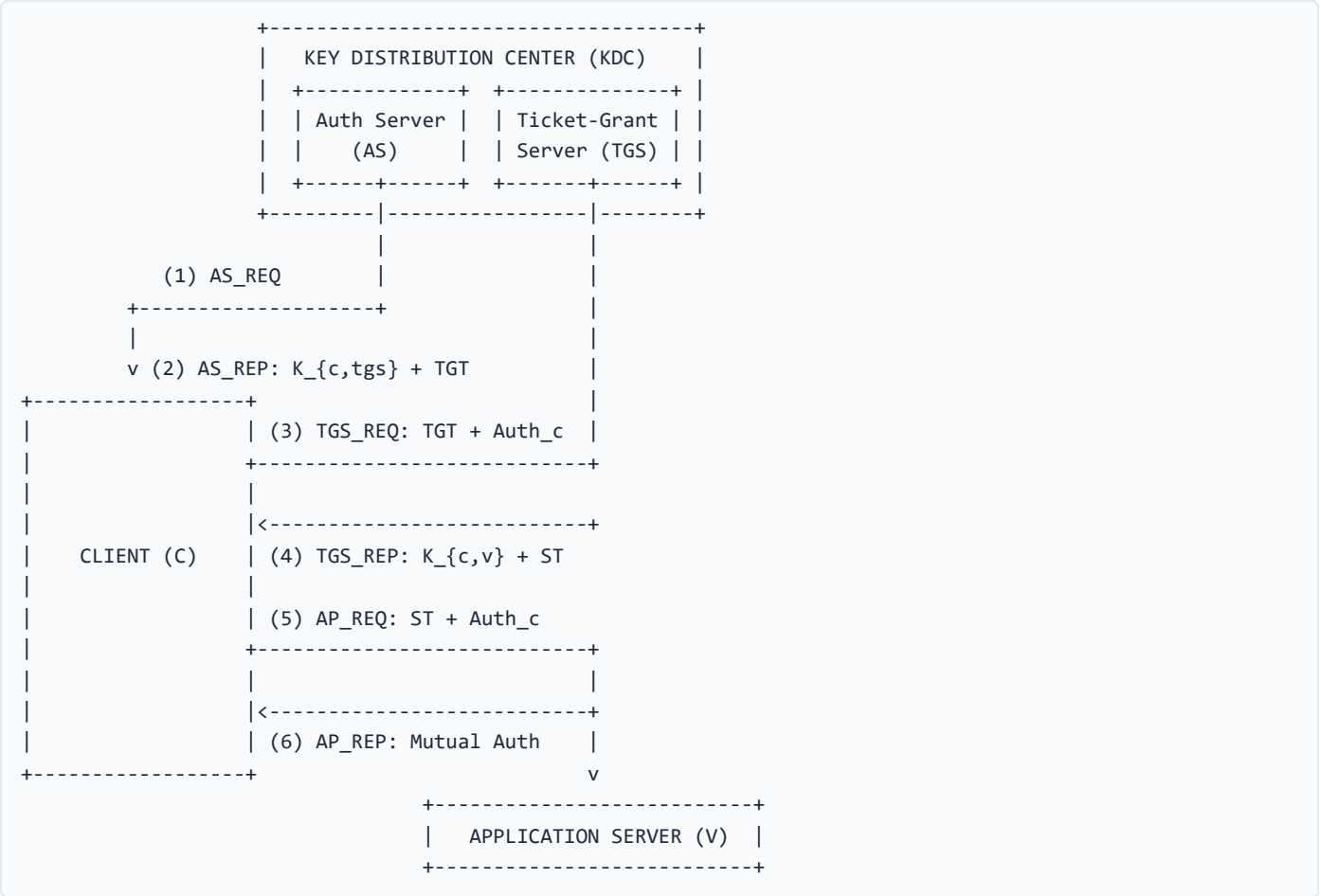
Q23. Explain the Kerberos authentication protocol with a neat diagram.

1. Core Concept & Formal Definition

Kerberos (RFC 4120 / MIT Athena) is a trusted third-party network authentication protocol that enables mutual authentication across insecure networks using symmetric encryption.

- **Needham–Schroeder Basis:** Replaces plaintext password transmissions with time-stamped **Tickets** issued by a centralized Key Distribution Center (KDC).
- **KDC Architecture:** Comprises two servers: an **Authentication Server (AS)** to verify user identity, and a **Ticket-Granting Server (TGS)** to issue application service tickets.

2. Architecture & Execution Flowchart



3. Detailed Operational Activities & Processing Phases

- **Phase 1 (AS Request):** Client sends user ID and requested TGS ID to AS. Passwords are never sent across the network.
- **Phase 2 (AS Response):** AS verifies client identity and responds with $E(K_c, [K_{c,tgs} \,\|\, \text{TS}])$ encrypted with client master key K_c, and Ticket-Granting Ticket $\text{TGT} = E(K_{tgs}, [C \,\|\, K_{c,tgs} \,\|\, \text{TS}])$.
- **Phase 3 (TGS Request):** Client decrypts $K_{c,tgs}$, creates Authenticator $\text{Auth}_c = E(K_{c,tgs}, [C \,\|\, \text{TS}])$, and sends TGT and Authenticator to TGS.
- **Phase 4 (TGS Response):** TGS decrypts TGT, validates timestamp, generates $K_{c,v}$, and returns Service Ticket $\text{ST} = E(K_v, [C \,\|\, K_{c,v} \,\|\, \text{TS}])$ and $E(K_{c,tgs}, [K_{c,v}])$.
- **Phase 5 (AP Request):** Client decrypts $K_{c,v}$, creates Authenticator $\text{Auth}_c = E(K_{c,v}, [C \,\|\, \text{TS}])$, and sends ST and Authenticator to Application Server V.
- **Phase 6 (AP Response - Mutual Auth):** Server V decrypts ST, verifies client Authenticator, and returns $E(K_{c,v}, [\text{TS} + 1])$. Client verifies timestamp, establishing mutual trust.

4. Summary Comparison & Specification Parameters

Token	Encrypted Under	Contents	Security Role
Ticket-Granting Ticket (TGT)	K_{tgs} (TGS Secret Key)	Client ID, $K_{c,tgs}$, IP, Lifetime	Authorizes client to request service tickets without re-prompting password
Service Ticket (ST)	K_v (Server Secret Key)	Client ID, $K_{c,v}$, IP, Lifetime	Authorizes client access to designated network application server V
Authenticator	Session Key ($K_{c,tgs}$ or $K_{c,v}$)	Client ID, Current Timestamp	Proves ticket bearer possesses session key; defeats replay attacks
Mutual Auth Reply	$K_{c,v}$ (Service Session Key)	Timestamp + 1	Proves to client that server legitimately decrypted the Service Ticket

MODEL QUESTION PAPER – 2

Q17. Use the Extended Euclidean Algorithm to express gcd(81, 57) as a linear combination of 81 and 57.

1. Mathematical Foundations & Bézout's Identity

The **Extended Euclidean Algorithm** is an extension of the classical Euclidean division algorithm. In addition to calculating the Greatest Common Divisor (`gcd(a, b)`) of two positive integers `a` and `b`, it computes integers `x` and `y` satisfying **Bézout's Identity**: `a × x + b × y = gcd(a, b)`. In modern asymmetric cryptography, this algorithm is fundamental for computing modular multiplicative inverses during key generation (such as calculating the RSA private key exponent $d \equiv e^{-1} \pmod{\phi(n)}$).

2. Forward Euclidean Division Sequence

```
Forward Division Steps:
Step 1:  81 = 57 * 1 + 24    ==>  Quotient q1 = 1, Remainder r1 = 24
Step 2:  57 = 24 * 2 + 9     ==>  Quotient q2 = 2, Remainder r2 = 9
Step 3:  24 =  9 * 2 + 6     ==>  Quotient q3 = 2, Remainder r3 = 6
Step 4:   9 =  6 * 1 + 3     ==>  Quotient q4 = 1, Remainder r4 = 3
Step 5:   6 =  3 * 2 + 0     ==>  Quotient q5 = 2, Remainder r5 = 0

Last non-zero remainder: r4 = 3  ==>  gcd(81, 57) = 3
```

3. Backtracking & Back-Substitution Derivation

To express `gcd(81, 57) = 3` as a linear combination of 81 and 57, we perform systematic back-substitution starting from Step 4:

- **From Step 4:** Express remainder 3 in terms of 9 and 6:

  ```
  3 = 9 - 1 × 6
  ```

- **Substitute 6 from Step 3:** Since `6 = 24 - 2 × 9`, substitute for 6:

  ```
  3 = 9 - 1 × (24 - 2 × 9) = 3 × 9 - 1 × 24
  ```

- **Substitute 9 from Step 2:** Since `9 = 57 - 2 × 24`, substitute for 9:

  ```
  3 = 3 × (57 - 2 × 24) - 1 × 24 = 3 × 57 - 7 × 24
  ```

- **Substitute 24 from Step 1:** Since `24 = 81 - 1 × 57`, substitute for 24:

  ```
  3 = 3 × 57 - 7 × (81 - 1 × 57) = -7 × 81 + 10 × 57
  ```

Rearranging into standard linear combination form gives:

gcd(81, 57) = 3 = 81(-7) + 57(10)

Thus, the Bézout coefficients are `x = -7` and `y = 10`.

4. Extended Euclidean Tableau & Verification

Index (i)	Quotient (q_i)	Remainder (r_i)	Coefficient (s_i)	Coefficient (t_i)	Linear Combination: $81s_i + 57t_i$
0	—	81	1	0	81(1) + 57(0) = 81
1	—	57	0	1	81(0) + 57(1) = 57
2	1	24	1	-1	81(1) + 57(-1) = 24
3	2	9	-2	3	81(-2) + 57(3) = 9
4	2	6	5	-7	81(5) + 57(-7) = 6
5	1	3	-7	10	81(-7) + 57(10) = 3
6	2	0	19	-27	81(19) + 57(-27) = 0

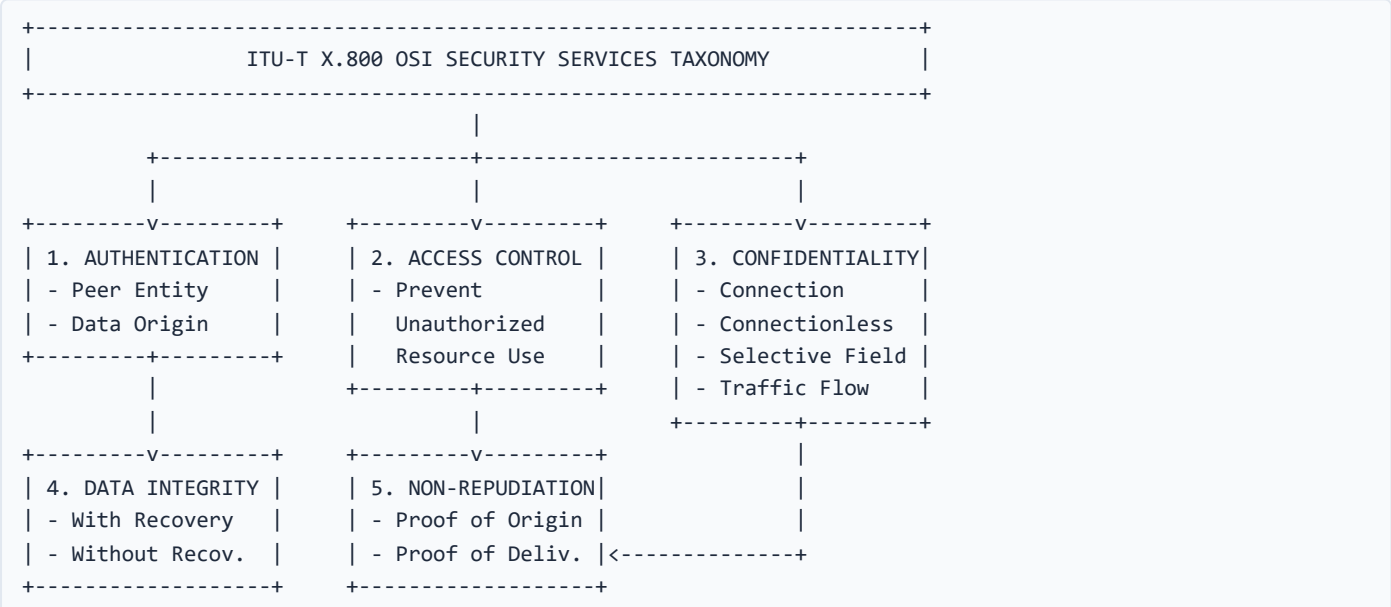
Verification Check: `81(-7) + 57(10) = -567 + 570 = 3`. The linear combination holds with exact mathematical equality.

Q18. Explain the five major security services defined in the X.800 Security Architecture.

1. Core Concept & Architectural Model

The **ITU-T Recommendation X.800** (Security Architecture for OSI) defines a systematic, international framework for identifying security requirements and selecting protocols in networked environments. Under X.800, a **security service** is a processing or communication capability provided by a protocol layer of an open system to ensure adequate security of systems and data transfers. X.800 formally defines five essential categories of security services.

2. X.800 Security Services Taxonomy Diagram



3. Detailed Analysis of the Five Security Services

- **Authentication:** Guarantees that a communicating entity or message source is genuine. It comprises *Peer Entity Authentication* (verifies entity identity during connection establishment to counter masquerade and replay attacks) and *Data Origin Authentication* (verifies the origin of individual datagrams in connectionless communication).
- **Access Control:** Enforces the ability to limit and control access to host systems, communication links, and services. It ensures that only authorized entities can perform read, write, or execute operations, typically managed through access control matrices and capability lists.
- **Data Confidentiality:** Protects transmitted data against passive unauthorized disclosure or eavesdropping. It spans *Connection Confidentiality* (all user data on a connection), *Connectionless Confidentiality* (individual datagrams), *Selective Field Confidentiality* (specific data fields), and *Traffic Flow Confidentiality* (masking traffic volume, rate, and frequency).

- **Data Integrity:** Provides assurance that data is received exactly as transmitted without unauthorized alteration, insertion, deletion, or reordering. Sub-services include connection integrity with recovery, connection integrity without recovery, and connectionless integrity.
- **Non-Repudiation:** Prevents either communicating party from denying having sent or received a message. It provides *Non-Repudiation with Proof of Origin* (receiver possesses proof identifying the sender) and *Non-Repudiation with Proof of Delivery* (sender possesses proof of message delivery).

4. Summary Specification & Mapping Table

Security Service	Primary Objective	Enabling Cryptographic Mechanism	Representative Protocol
Authentication	Confirm entity and message identity	Digital signatures, authentication exchanges, nonces	TLS Handshake, Kerberos
Access Control	Prevent unauthorized resource access	Access control lists, tokens, firewalls	RADIUS, IEEE 802.1X
Data Confidentiality	Prevent passive eavesdropping	Symmetric / asymmetric encipherment, padding	AES in TLS 1.3, IPsec ESP
Data Integrity	Detect unauthorized data modification	Cryptographic hash functions, HMACs	IPsec AH, SHA-256
Non-Repudiation	Prevent denial of transmission/receipt	Digital signatures, trusted timestamping/notarization	X.509 PKI, PGP / S/MIME

Q19. Explain the Playfair Cipher and encrypt the plaintext BALLOON using the given key matrix.

1. Core Concept & Operating Principles

Invented by Charles Wheatstone in 1854 and popularized by Lyon Playfair, the **Playfair Cipher** is a classical polygraphic **digraphic substitution cipher**. Unlike monoalphabetic ciphers that substitute single letters individually, Playfair encrypts pairs of letters (digraphs) simultaneously using a 5×5 grid formed by a keyword. This digraphic substitution expands the alphabet space from 26 letters to `26 × 26 = 676` possible pairs, greatly flattening single-letter frequency distribution. By convention, `I` and `J` share a single cell.

2. Key Matrix Layout & Digraph Preparation

The given 5×5 key matrix constructed with keyword **MONARCHY** is:

```
       Col 1   Col 2   Col 3   Col 4   Col 5
Row 1:   M       O       N       A       R
Row 2:   C       H       Y       B       D
Row 3:   E       F       G      I/J      K
Row 4:   L       P       Q       S       T
Row 5:   U       V       W       X       Z
```

Digraph Preparation Rules:

- Split plaintext `BALLOON` into pairs. The first pair is `BA`.
- The next letters are `LL`. Identical letters within a pair must be separated by inserting a filler letter (typically `X`), yielding `LX`.
- The remaining letters are `OON`. Pairing continues with `LO`, followed by `ON`.
- Resulting plaintext digraphs: **BA**, **LX**, **LO**, **ON**.

3. Step-by-Step Digraph Encryption Walkthrough

Applying the standard Playfair transformation rules to each digraph:

- **Digraph BA (Same Column):** `B` is at (Row 2, Col 4) and `A` is at (Row 1, Col 4). Both lie in Column 4. Under the same-column rule, each letter shifts down cyclically: `B` becomes `I` (Row 3, Col 4) and `A` becomes `B` (Row 2, Col 4). Thus, **BA → IB**.

- **Digraph LX (Rectangle Rule):** `L` is at (Row 4, Col 1) and `X` is at (Row 5, Col 4). Forming the opposite corners of their rectangle, `L` maps to `U` (Row 5, Col 1) and `X` maps to `S` (Row 4, Col 4). Thus, **LX → US**.
- **Digraph LO (Rectangle Rule):** `L` is at (Row 4, Col 1) and `O` is at (Row 1, Col 2). Forming the opposite rectangle corners, `L` maps to `M` (Row 1, Col 1) and `O` maps to `P` (Row 4, Col 2). Thus, **LO → MP**.
- **Digraph ON (Same Row):** `O` is at (Row 1, Col 2) and `N` is at (Row 1, Col 3). Both lie in Row 1. Under the same-row rule, each letter shifts right cyclically: `O` becomes `N` (Row 1, Col 3) and `N` becomes `A` (Row 1, Col 4). Thus, **ON → NA**.

Concatenating the ciphertext digraphs:

Ciphertext: IB + US + MP + NA = IBUSMPNA

4. Decryption Procedure & Summary Mapping Table

Decryption reverses each transformation: shift upward for same column, shift leftward for same row, and reverse corner coordinates for rectangles:

- `IB` (Same Col) → Shift up: `I → B`, `B → A` → **BA**
- `US` (Rectangle) → Corner lookup: `U → L`, `S → X` → **LX**
- `MP` (Rectangle) → Corner lookup: `M → L`, `P → O` → **LO**
- `NA` (Same Row) → Shift left: `N → O`, `A → N` → **ON**

Reconstructed text: `BALXLOON`. Dropping the filler letter `X` between identical letters recovers the original plaintext: **BALLOON**.

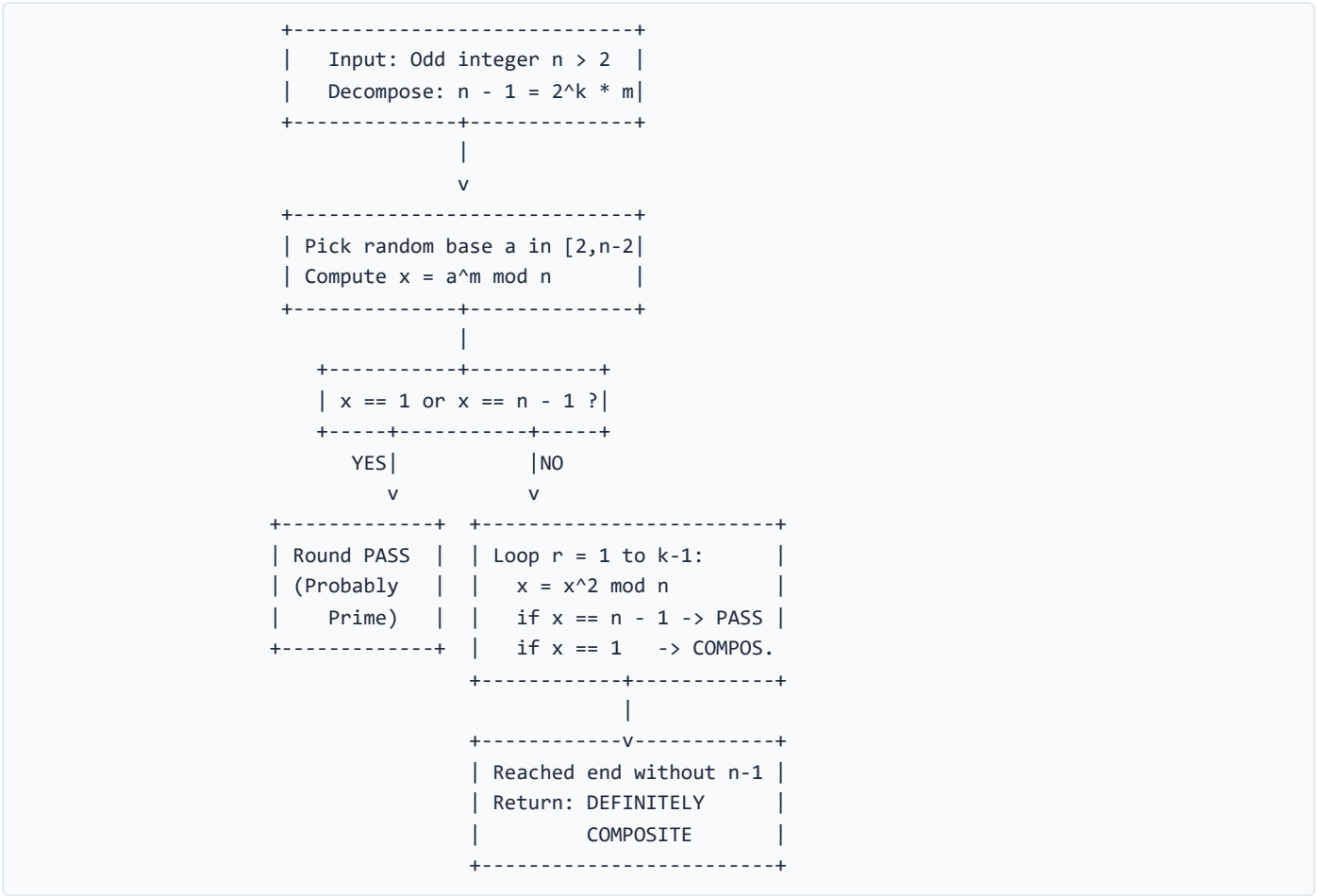
Digraph Pair	Geometric Rule	Coordinates (Row, Col)	Ciphertext Output
BA	Same Column (Col 4)	(2,4) and (1,4) → shift down	**IB**
LX	Rectangle Rule	(4,1) and (5,4) → opposite corners	**US**
LO	Rectangle Rule	(4,1) and (1,2) → opposite corners	**MP**
ON	Same Row (Row 1)	(1,2) and (1,3) → shift right	**NA**

Q20. Explain the working procedure of the Miller–Rabin Primality Test.

1. Core Concept & Mathematical Basis

The **Miller–Rabin Primality Test** is a randomized, probabilistic primality testing algorithm essential for generating large prime numbers in public-key cryptosystems (such as RSA and Diffie–Hellman). It relies on two foundational number-theoretic principles: **Fermat's Little Theorem** ($a^{n-1} \equiv 1 \pmod{n}$ for prime n) and the fact that in a prime field $\mathbb{Z}_n$, the only square roots of 1 modulo n are 1 and $-1 \equiv n - 1$. If any non-trivial square root of 1 is discovered, n is unconditionally composite.

2. Algorithmic Logic & Verification Flowchart



3. Detailed Algorithmic Steps & Working Procedure

- **Step 1 (Decomposition):** Given an odd integer $n > 2$, write $n - 1 = 2^k \times m$, where m is an odd integer and $k \geq 1$ by factoring out all factors of 2.
- **Step 2 (Base Selection):** Choose a random base integer a such that $2 \leq a \leq n - 2$.
- **Step 3 (Initial Exponentiation):** Compute $x \equiv a^m \pmod{n}$ using modular exponentiation. If $x \equiv 1 \pmod{n}$ or $x \equiv n - 1 \pmod{n}$, candidate n passes this round (it is **probably prime**).
- **Step 4 (Successive Squaring Loop):** If $x \neq 1$ and $x \neq n - 1$, square x repeatedly for $r = 1, 2, \ldots, k - 1$:
  Calculate $x \equiv x^2 \pmod{n}$.
  • If $x \equiv n - 1 \pmod{n}$, candidate n passes this round.
  • If $x \equiv 1 \pmod{n}$, then the previous value of x was a non-trivial square root of 1, proving n is **definitely composite**.
- **Step 5 (Compositeness Witness):** If the loop finishes without ever reaching $n - 1$, n is declared **definitely composite** (base a is a compositeness witness).
- **Step 6 (Error Bound & Repetition):** If n is composite, the probability of passing a round is at most $1/4$. Testing across t independent random bases reduces the false prime probability to at most $(1/4)^t = 2^{-2t}$, which is negligible for $t = 64$.

4. Numerical Example & Comparative Summary Table

Example: Test $n = 61$ with base $a = 2$:
Decompose: $61 - 1 = 60 = 2^2 \times 15$ &implies; $k = 2, m = 15$.
Compute: $x = 2^{15} \pmod{61}$. Since $2^6 = 64 \equiv 3$, $2^{12} \equiv 9$, $2^{15} = 9 \times 8 = 72 \equiv 11 \pmod{61}$.
Next square: $x = 11^2 = 121 \equiv 60 \equiv -1 \equiv n - 1 \pmod{61}$.
Since $x \equiv n - 1$, $n = 61$ passes the test and is certified probably prime.

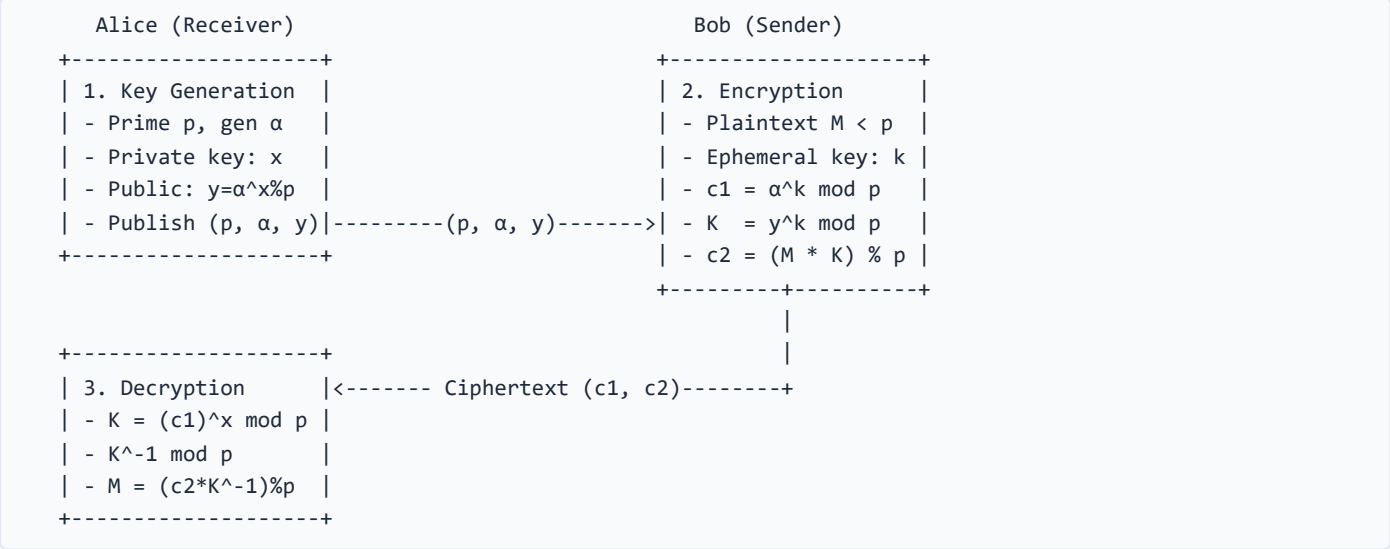
Evaluation Metric	Miller–Rabin Primality Test	Fermat Primality Test
Core Foundation	Fermat's Theorem + Non-trivial square roots of 1	Fermat's Little Theorem alone ($a^{n-1} \equiv 1$)
Carmichael Numbers	**Immune:** reliably detects all Carmichael composites	**Fails completely:** Carmichael numbers pass all coprime bases
Worst-Case Error Bound	At most $(1/4)^t$ for t rounds	Can approach 1 for Carmichael numbers
Time Complexity	$O(t \times \log^3 n)$ bit operations	$O(t \times \log^3 n)$ bit operations

Q21. Explain the ElGamal cryptosystem with its key generation, encryption and decryption procedures.

1. Core Concept & Mathematical Security Basis

Formulated by Taher Elgamal in 1985, the **ElGamal Cryptosystem** is an asymmetric public-key encryption scheme based on the **Diffie–Hellman key exchange** principle. Its mathematical security relies on the intractability of the **Discrete Logarithm Problem (DLP)** in finite cyclic groups: given a large prime p, a generator α, and $y \equiv \alpha^x \pmod p$, it is computationally infeasible to compute private key $x = \log_\alpha y$. ElGamal is a **probabilistic cryptosystem**: encrypting identical plaintexts produces completely different ciphertexts due to an ephemeral random key k.

2. ElGamal Architectural Flowchart



3. Formal Mathematical Procedures

- **Key Generation:**
  • Select a large prime p and a primitive root (generator) α modulo p.
  • Select a private integer x uniformly at random such that $1 < x < p - 1$.
  • Calculate public value $y \equiv \alpha^x \pmod p$.
  • Public Key is $PK = (p, \alpha, y)$; Private Key is $SK = x$.
- **Encryption Procedure:**
  • Plaintext message is encoded as integer M in the range $0 \le M \le p - 1$.
  • Sender selects a random ephemeral key k such that $1 < k < p - 1$ and $\gcd(k, p - 1) = 1$.
  • Computes ephemeral public key component: $c_1 \equiv \alpha^k \pmod p$.
  • Computes shared masking key: $K \equiv y^k \equiv (\alpha^x)^k \equiv \alpha^{xk} \pmod p$.
  • Computes masked ciphertext component: $c_2 \equiv (M \times K) \pmod p$.
  • Transmits ciphertext tuple: $C = (c_1, c_2)$.

- **Decryption Procedure:**
  • Receiver Alice uses private key x and received c_1 to recover shared secret: $K \equiv (c_1)^x \equiv (\alpha^k)^x \equiv \alpha^{kx} \pmod p$.
  • Computes modular multiplicative inverse $K^{-1} \pmod p$ using the Extended Euclidean Algorithm.
  • Recovers message: $M \equiv (c_2 \times K^{-1}) \pmod p \equiv (M \times K \times K^{-1}) \equiv M \pmod p$.

4. Numerical Example & Cryptographic Characteristics Table

Example: Let $p = 11, \alpha = 2$. Private key $x = 3 \implies y = 2^3 \pmod{11} = 8$. Public key is $(11, 2, 8)$.
• *Encrypt:* Plaintext $M = 5$, choose ephemeral $k = 4$.
$c_1 = 2^4 \pmod{11} = 16 \equiv 5 \pmod{11}$.
$K = 8^4 \pmod{11} = 4096 \equiv 4 \pmod{11}$.
$c_2 = (5 \times 4) \pmod{11} = 20 \equiv 9 \pmod{11}$. Ciphertext is $(c_1, c_2) = (5, 9)$.
• *Decrypt:* $K = 5^3 \pmod{11} = 125 \equiv 4 \pmod{11}$. Inverse $4^{-1} \pmod{11} = 3$ (since $4 \times 3 = 12 \equiv 1$).
$M = (9 \times 3) \pmod{11} = 27 \equiv 5 \pmod{11}$. Plaintext 5 is recovered exactly.

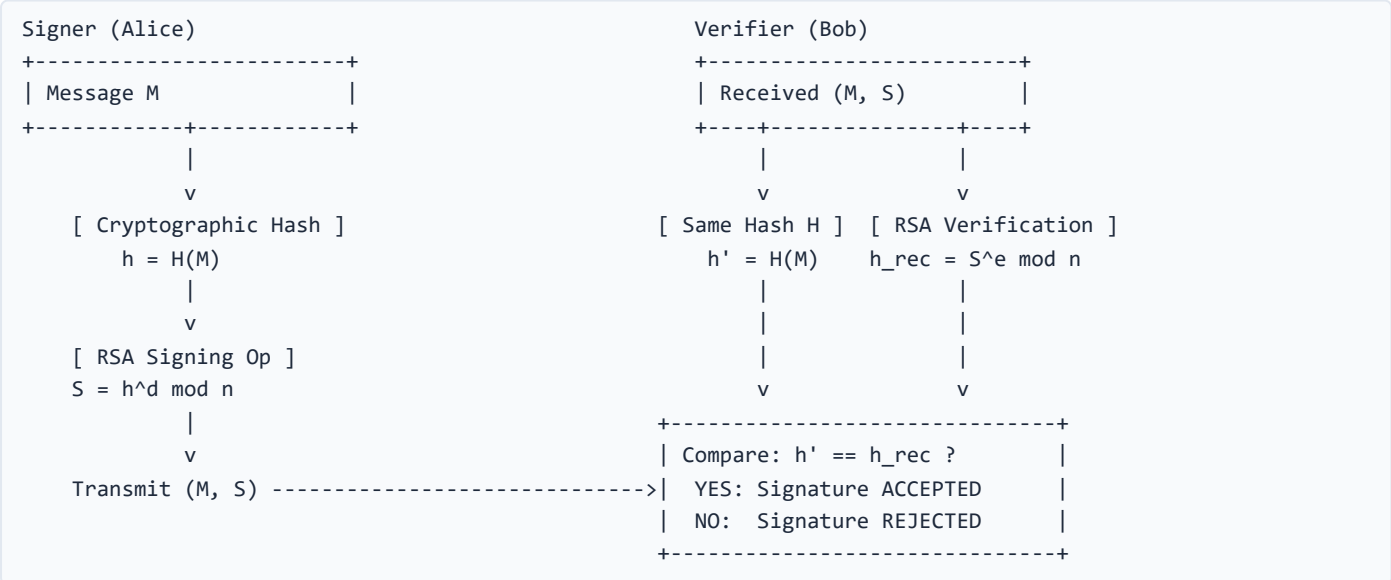
Characteristic	ElGamal Cryptosystem	RSA Cryptosystem
Hardness Problem	Discrete Logarithm Problem (DLP)	Integer Factorization Problem (IFP)
Ciphertext Expansion	**2× Expansion:** 2 ciphertext integers (c_1, c_2) per block	**1× Expansion:** 1 ciphertext integer per block
Cipher Nature	Randomized / Probabilistic (ephemeral k)	Deterministic (requires OAEP padding)
Ephemeral Key Hazard	Reusing k completely compromises plaintext	No ephemeral key in standard RSA

Q22. Explain the RSA Digital Signature Scheme with signing and verification procedures.

1. Core Concept & Dual Trapdoor Architecture

The **RSA Digital Signature Scheme** is an asymmetric cryptographic mechanism providing **authenticity**, **data integrity**, and **non-repudiation**. Built on the mathematical foundations of the RSA algorithm, it inverts the traditional encryption paradigm: only the private key holder can create a signature, whereas any party holding the signer's public key can verify its validity. Under the modern **Hash-then-Sign paradigm**, the signer signs a fixed-length cryptographic hash digest $h = H(M)$ rather than the raw message, eliminating existential forgery and ensuring computational efficiency.

2. RSA Signature Signing & Verification Flowchart



3. Formal Mathematical Procedures

- **Key Generation:**
  • Select two large distinct prime numbers p and q.
  • Compute modulus $n = p \times q$ and Euler's totient $\phi(n) = (p - 1)(q - 1)$.
  • Select public verification exponent e such that $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$ (typically $e = 65537$).
  • Calculate private signing exponent $d \equiv e^{-1} \pmod{\phi(n)}$ using the Extended Euclidean Algorithm.
  • Public Verification Key is $VK = (e, n)$; Private Signing Key is $SK = (d, n)$.
- **Signing Procedure:**
  • Signer hashes message M using a secure hash function (e.g., SHA-256): $h = H(M)$.
  • Signer computes the digital signature S using private exponent d:
  $S \equiv h^d \pmod{n}$.
  • The signature pair (M, S) is transmitted to the verifier.
- **Verification Procedure:**
  • Verifier receives (M', S') and obtains authentic public key (e, n).
  • Verifier independently recomputes message digest: $h' = H(M')$.
  • Verifier recovers signer's digest using public exponent e:
  $h_{rec} \equiv (S')^e \equiv (h^d)^e \equiv h^{ed} \equiv h^{k\phi(n)+1} \equiv h \pmod{n}$.
  • If $h_{rec} == h'$, signature is authenticated (valid); otherwise, it is rejected.

4. Security Dimensions & Standards Comparison Table

Raw ("textbook") RSA signatures are vulnerable to mathematical attacks such as *multiplicative forgery* (where $S_1 \times S_2 = (M_1 \times M_2)^d$). Modern implementations mandate **RSA-PSS** (Probabilistic Signature Scheme, PKCS#1 v2.2), which adds randomized salt and masking to achieve provable cryptographic security.

Security Dimension	Textbook RSA Signature	Standard RSA-PSS (RFC 8017)
Input Formulation	Raw unpadded message / digest	Cryptographic hash with randomized salt & mask
Homomorphic Attack	Vulnerable to algebraic manipulation	**Provably secure:** immune to chosen-message forgery
Verification Speed	Fast (small public exponent $e = 65537$)	Fast (small public exponent $e = 65537$)
Standard Deployment	Insecure / Deprecated	Industry standard in TLS 1.3, S/MIME, X.509 PKI

Q23. Explain the TLS Handshake Protocol with a neat diagram.

1. Core Concept & Architectural Role

The **Transport Layer Security (TLS)** Handshake Protocol (RFC 5246 / RFC 8446) is the primary control protocol in the TLS framework, executing above TCP before any application data is transferred. It is responsible for three core security objectives: **algorithm negotiation** (agreeing on protocol version and Cipher Suite), **mutual or server authentication** (validating X.509 public-key certificates), and **secure key establishment** (generating shared session secrets while resisting eavesdropping and man-in-the-middle attacks).

2. TLS Handshake Protocol Message Flow Diagram

```
Client                                                          Server
  |                                                               |
  | --------------------- 1. ClientHello ---------------------------> | Phase 1:
  |                                                               | Capabilities
  | <-------------------- 2. ServerHello ---------------------------- | Negotiation
  | <-------------------- 3. Certificate ---------------------------- |
  | <-------------------- 4. ServerKeyExchange (Optional) ----------- | Phase 2:
  | <-------------------- 5. CertificateRequest (Optional) ---------- | Server Auth &
  | <-------------------- 6. ServerHelloDone ----------------------- | Key Exchange
  |                                                               |
  | --------------------- 7. Certificate (Optional) ----------------> |
  | --------------------- 8. ClientKeyExchange ---------------------> | Phase 3:
  | --------------------- 9. CertificateVerify (Optional) ----------> | Client Auth &
  | -------------------- 10. ChangeCipherSpec ----------------------> | Pre-Master Secret
  | -------------------- 11. Finished (Encrypted) ------------------> |
  |                                                               |
  | <------------------- 12. ChangeCipherSpec ----------------------- | Phase 4:
  | <------------------- 13. Finished (Encrypted) ------------------- | Switch to Encrypted
  |                                                               | Session
  |<=================== Secure Application Data Flow ================>|
```

3. Four-Phase Detailed Handshake Activities

- **Phase 1 (Negotiate Security Capabilities):**
  - `ClientHello` : Client transmits highest supported TLS version, 32-byte `ClientRandom` (R_C), Session ID, and supported Cipher Suites (e.g., `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`).
  - `ServerHello` : Server selects protocol version, sends 32-byte `ServerRandom` (R_S), and picks the active Cipher Suite.
- **Phase 2 (Server Authentication & Key Exchange):**
  - `Certificate` : Server provides its X.509 digital certificate chain containing its public key.
  - `ServerKeyExchange` : Sent if ephemeral Diffie–Hellman key exchange is chosen, containing signed DH parameters.
  - `CertificateRequest` : (Optional) Sent if server requires mutual client authentication.
  - `ServerHelloDone` : Signals completion of the server's hello phase.
- **Phase 3 (Client Authentication & Key Exchange):**
  - `Certificate` : (Optional) Client sends its X.509 certificate if requested.
  - `ClientKeyExchange` : Client sends key material. For RSA, client generates a 48-byte Pre-Master Secret (PMS) encrypted with server's public key; for ECDHE, it sends ephemeral public DH parameters.
  - `CertificateVerify` : (Optional) Signs previous handshake messages with client private key to prove ownership.
  - `ChangeCipherSpec` : Notifies server that subsequent packets will be encrypted under negotiated session keys.
  - `Finished` : First encrypted verification message containing a MAC of all prior exchanges.
- **Phase 4 (Session Finalization):**
  - Both parties compute the 48-byte Master Secret: `MS = PRF(PMS, "master secret", R_C || R_S)` .
  - From `MS` , encryption keys, MAC keys, and IVs are derived.
  - Server responds with `ChangeCipherSpec` and an encrypted `Finished` message, after which secure application traffic begins.

4. Summary Specification & Parameter Mapping Table

Handshake Phase	Primary Messages Exchanged	Cryptographic Objective	Generated Parameter
Phase 1: Capabilities	`ClientHello`, `ServerHello`	Cipher suite & protocol version negotiation	`ClientRandom`, `ServerRandom`
Phase 2: Server Auth	`Certificate`, `ServerKeyExchange`, `ServerHelloDone`	Authenticate server identity & provide DH parameters	Server public key, Server DH parameters
Phase 3: Client Key	`ClientKeyExchange`, `ChangeCipherSpec`, `Finished`	Deliver Pre-Master Secret (PMS) & verify key derivation	Pre-Master Secret (PMS), Client Handshake Hash
Phase 4: Completion	`ChangeCipherSpec`, `Finished`	Switch to symmetric authenticated encryption	Master Secret (MS), Symmetric Session Keys

---

MODEL QUESTION PAPER – 3

Q17. Find the inverse of the matrix [[3, 3], [2, 5]] modulo 26. (8 Marks)

1. Core Concept & Invertibility Criterion

In modular matrix arithmetic and polygraphic ciphers (such as the Hill Cipher), computing the modular inverse of a key matrix M modulo 26 is required to decrypt ciphertext blocks into plaintext ($P = C \cdot M^{-1} \bmod 26$).

For a 2 × 2 matrix $M = [[a, b], [c, d]]$, the modular inverse M^{-1} exists if and only if:

- **Non-Zero Determinant:** $\det(M) = (ad - bc) \bmod 26 \neq 0$.
- **Coprimality Condition:** $\gcd(\det(M), 26) = 1$, ensuring a unique modular multiplicative inverse $\det(M)^{-1} \bmod 26$ in $\mathbb{Z}_{26}$.

The matrix inverse formula is: $\mathbf{M^{-1} = \det(M)^{-1} \cdot \text{adj}(M) \bmod 26}$, where $\text{adj}(M) = [[d, -b], [-c, a]] \bmod 26$.

2. Systematic Inversion & Verification Workflow

```
Given Matrix M = [[3, 3], [2, 5]]
                 |
                 v
[Step 1: Compute Determinant] -----> det(M) = (3*5) - (3*2) = 15 - 6 = 9
                 |
                 v
[Step 2: Test Invertibility] ------> gcd(9, 26) = 1 (Coprime, Inverse Exists)
                 |
                 v
[Step 3: Multiplicative Inverse] --> 9 * d = 1 (mod 26)  ==>  det(M)^-1 = 3
                 |
                 v
[Step 4: Formulate Adjugate] ------> adj(M) = [[5, -3], [-2, 3]] = [[5, 23], [24, 3]] (mod 26)
                 |
                 v
[Step 5: Scalar Modulo Product] ---> M^-1 = 3 * adj(M) = [[15, 69], [72, 9]] = [[15, 17], [20, 9]] (mod 26)
                 |
                 v
[Step 6: Mathematical Proof] ------> M * M^-1 = [[105, 78], [130, 79]] = [[1, 0], [0, 1]] = I (mod 26)
```

3. Step-by-Step Computational Execution

- **Step 1: Compute Determinant:**
  $\det(M) = (a \cdot d) - (b \cdot c) = (3 \times 5) - (3 \times 2) = 15 - 6 = \mathbf{9} \equiv 9 \bmod 26$.

- **Step 2: Verify Coprimality and Invertibility:**
  The prime factors of 26 are 2 and 13. The prime factor of 9 is 3. Since gcd(9, 26) = 1, 9 is an invertible unit in $\mathbb{Z}_{26}$, confirming that M^{-1} exists.
- **Step 3: Calculate Multiplicative Inverse $\det(M)^{-1}$ mod 26:**
  We solve $9 \cdot x \equiv 1 \bmod 26$ using the Extended Euclidean Algorithm:
  $26 = 2(9) + 8$ &implies; $8 = 26 - 2(9)$
  $9 = 1(8) + 1$ &implies; $1 = 9 - 1(26 - 2(9)) = 3(9) - 1(26)$
  Taking modulo 26 yields: $3 \times 9 \equiv 1 \bmod 26$ &implies; **$\det(M)^{-1} = 9^{-1} \equiv 3 \bmod 26$**. *(Check: $9 \times 3 = 27 = 26(1) + 1 \equiv 1 \bmod 26$).*
- **Step 4: Construct Adjugate Matrix modulo 26:**
  $\text{adj}(M) = [[d, -b], [-c, a]] = [[5, -3], [-2, 3]]$
  Mapping negative values to standard residues in $\mathbb{Z}_{26}$:
  $-3 \equiv 26 - 3 = 23 \bmod 26$, $\quad -2 \equiv 26 - 2 = 24 \bmod 26$
  $\text{adj}(M) \equiv [[5, 23], [24, 3]] \bmod 26$.
- **Step 5: Multiply by $\det(M)^{-1}$ and Reduce Modulo 26:**
  $M^{-1} \equiv 3 \times [[5, 23], [24, 3]] = [[15, 69], [72, 9]] \bmod 26$
  • Entry (1,1): $15 \bmod 26 =$ **15**
  • Entry (1,2): $69 = (2 \times 26) + 17 \equiv$ **17** mod 26
  • Entry (2,1): $72 = (2 \times 26) + 20 \equiv$ **20** mod 26
  • Entry (2,2): $9 \bmod 26 =$ **9**
  $M^{-1} = [[15, 17], [20, 9]] \bmod 26$.
- **Step 6: Mathematical Proof & Verification ($M \cdot M^{-1} \equiv I_2 \bmod 26$):**
  $M \cdot M^{-1} = [[3, 3], [2, 5]] \times [[15, 17], [20, 9]] = [[(3 \cdot 15 + 3 \cdot 20), (3 \cdot 17 + 3 \cdot 9)], [(2 \cdot 15 + 5 \cdot 20), (2 \cdot 17 + 5 \cdot 9)]] = [[105, 78], [130, 79]]$
  • Row 1: $105 = 4(26) + 1 \equiv$ **1** mod 26, $\quad 78 = 3(26) + 0 \equiv$ **0** mod 26
  • Row 2: $130 = 5(26) + 0 \equiv$ **0** mod 26, $\quad 79 = 3(26) + 1 \equiv$ **1** mod 26
  $M \cdot M^{-1} \equiv$ **$[[1, 0], [0, 1]] = I_2 \bmod 26$**. The inverse is rigorously verified.

4. Summary of Computational Stages

Calculation Stage	Mathematical Formula	Intermediate Value	Modulo 26 Result	Cryptographic Role
Determinant $\det(M)$	$ad - bc = (3)(5) - (3)(2)$	$15 - 6 = 9$	9	Validates non-singularity
Coprimality Test	$\gcd(\det(M), 26)$	$\gcd(9, 26)$	1	Confirms matrix is invertible in $\mathbb{Z}_{26}$
Modular Inverse $\det^{-1}$	$9 \cdot x \equiv 1 \bmod 26$	Extended Euclid	3	Scalar factor for adjugate matrix
Adjugate Matrix $\text{adj}(M)$	$[[d, -b], [-c, a]]$	$[[5, -3], [-2, 3]]$	$[[5, 23], [24, 3]]$	Transpose of cofactor matrix
Inverse Matrix M^{-1}	$\det^{-1} \cdot \text{adj}(M)$	$[[15, 69], [72, 9]]$	**$[[15, 17], [20, 9]]$**	Decryption key for Hill cipher blocks
Identity Verification	$M \cdot M^{-1} \bmod 26$	$[[105, 78], [130, 79]]$	$[[1, 0], [0, 1]]$	Confirms exact decipherability

---

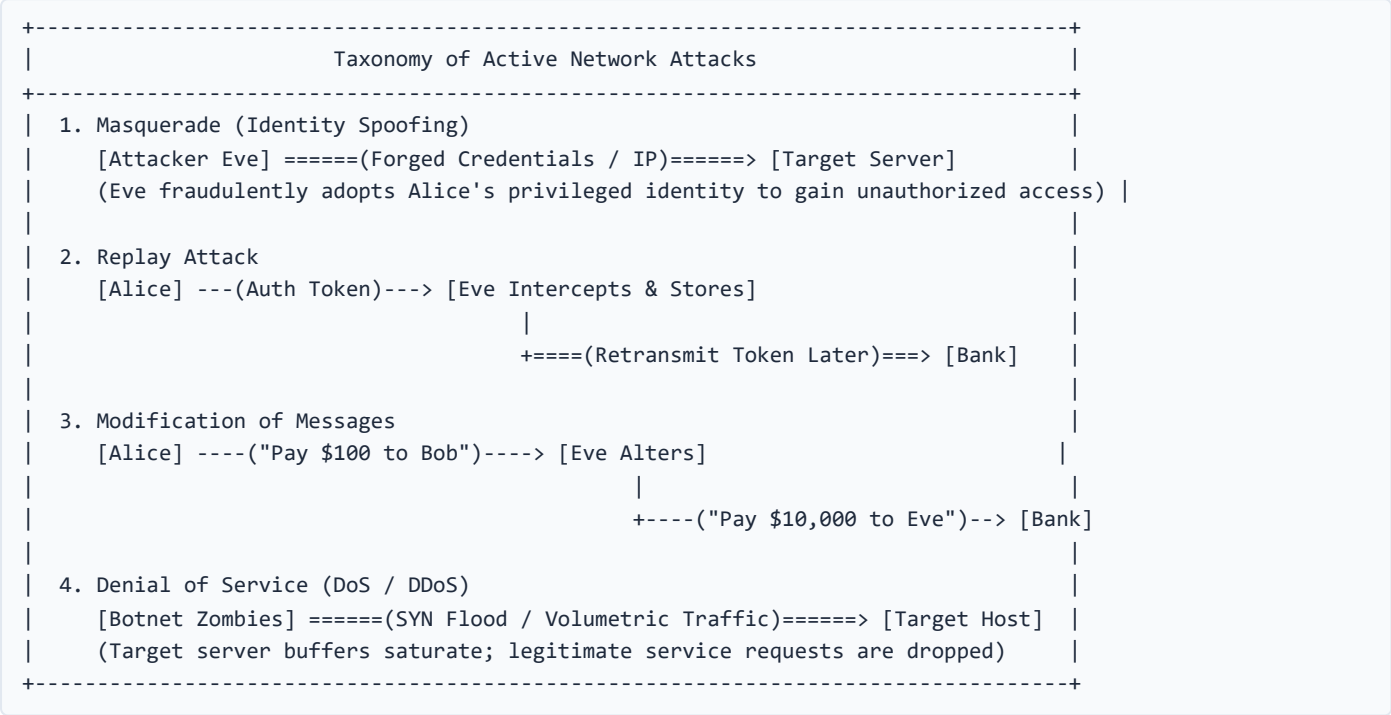
Q18. Explain the different types of active attacks with suitable examples. (8 Marks)

1. Core Threat Concept & Taxonomy

In network security architecture (formalized under ITU-T Recommendation X.800 and RFC 4949), security threats are categorized into **Passive Attacks** and **Active Attacks** based on their interaction with communications channels and system resources:

- **Active Attack Definition:** An attack that involves the unauthorized modification of data streams, fabrication of illegitimate messages, unauthorized alteration of system state, or deliberate disruption of computing and network availability.
- **Targeted Security Goals:** While passive attacks (eavesdropping, traffic analysis) compromise *Confidentiality* without altering data, active attacks directly threaten **Data Integrity**, **System Availability**, and **Entity Authenticity**.
- **Detection vs Prevention:** Passive attacks are difficult to detect because they leave no audit trail, but they can be prevented using strong encryption (e.g., AES, TLS). Conversely, active attacks cannot be completely prevented across open networks; therefore, security frameworks prioritize **rapid real-time detection**, **session termination**, and **resilient system recovery**.

2. Architecture & Modus Operandi of Active Attacks



3. Detailed Dimensions & Practical Real-World Examples

- **1. Masquerade (Impersonation / Spoofing):**
  Mechanism: An attacker assumes the verified identity of a legitimate entity by forging authentication credentials, forging network addresses (IP/MAC/ARP spoofing), or hijacking active session tokens.
  Concrete Example: **IP Address Spoofing & Session Hijacking:** An external attacker crafts TCP packets with the spoofed source IP address of an internal admin machine to bypass perimeter packet-filtering firewalls and issue unauthorized SQL commands.
- **2. Replay Attack:**
  Mechanism: The passive or active interception of a valid, authenticated message or token, followed by its unauthorized, delayed retransmission to duplicate privileged operations.
  Concrete Example: **Financial Wire Transfer Replay:** An adversary intercepts an encrypted transaction packet authorizing a $5,000 transfer. Without decrypting the payload, the adversary captures the raw ciphertext packets and injects them repeatedly into the banking gateway, causing duplicate debits before detection. Countered using nonces, sequence numbers, and timestamps.
- **3. Modification of Messages:**
  Mechanism: An adversary intercepts in-transit message packets and tampers with their content by altering data values, deleting fields, reordering packet sequences, or injecting false commands, violating message integrity.
  Concrete Example: **Man-in-the-Middle (MitM) Purchase Order Alteration:** An attacker intercepting unauthenticated hospital supply communications alters a digital medicine order from `{"Item": "Insulin", "Qty": 10}` to `{"Item": "Insulin", "Qty": 500}` before forwarding it. Countered via HMAC and digital signatures.

- **4. Denial of Service (DoS & DDoS):**
  Mechanism: Prevents or inhibits the normal use or management of communications facilities, networks, or computing servers by exhausting bandwidth, CPU cycles, socket connections, or server memory.
  Concrete Example: **TCP SYN Flood Attack:** An attacker floods a server with millions of spoofed TCP SYN connection requests without completing the three-way handshake. The server's Transmission Control Block (TCB) queue saturates, causing it to reject all legitimate incoming connections.

4. Active Attack Classification & Mitigation Matrix

Attack Category	Compromised Goal	Modus Operandi	Real-World Scenario	Primary Countermeasure
Masquerade	Authenticity & Access Control	Faking credentials, ARP/IP headers, or tokens	IP spoofing to bypass perimeter firewall access lists	Multi-Factor Authentication (MFA), Mutual TLS, 802.1X
Replay Attack	Integrity & Freshness	Sniffing valid auth packets and retransmitting later	Retransmitting captured e-banking fund transfer packets	Cryptographic Nonces, Session Timestamps, Sequence Numbers
Modification	Data Integrity & Authenticity	Splicing, altering, or delaying payload data	Man-in-the-Middle alteration of transaction values	Hash-based Message Authentication Codes (HMAC), Digital Signatures
Denial of Service	System Availability	Flooding system resources, buffers, or bandwidth	TCP SYN Flooding or DNS Amplification DDoS attack	SYN Cookies, Rate Limiting, Anycast Scrubbing Centers, WAF

Q19. Explain the Vigenère Cipher. Encrypt the plaintext DATASECURE using the keyword MATH and decrypt the resulting ciphertext. (8 Marks)

1. Core Concept & Theoretical Foundations

The Vigenère Cipher, formulated by Giovan Battista Bellaso in 1553 and later attributed to Blaise de Vigenère, is a classic **polyalphabetic substitution cipher**:

- **Vulnerability of Monoalphabetic Ciphers:** Monoalphabetic substitution ciphers (e.g., Caesar cipher) maintain a static 1-to-1 mapping between plaintext and ciphertext characters. Consequently, natural letter frequency distributions (such as 'E', 'T', 'A' in English) are preserved directly in the ciphertext, making them trivial to crack via statistical frequency analysis.
- **Polyalphabetic Design Principle:** The Vigenère cipher mitigates this vulnerability by using multiple varying Caesar shifts across the message. The shift value applied to each character is determined cyclically by the corresponding letter of a repeating secret keyword. Because identical plaintext letters map to multiple different ciphertext letters, unigram letter frequencies are effectively flattened.
- **Mathematical Modular Formulations:** Mapping letters numerically where A = 0, B = 1, ..., Z = 25:
  Encryption: $C_i = (P_i + K_i) \bmod 26$
  Decryption: $P_i = (C_i - K_i + 26) \bmod 26$
  where P_i is the plaintext letter index, K_i is the keystream letter index, and C_i is the resulting ciphertext letter index.

2. Keystream Expansion & Architectural Flow

```
Plaintext Block (L = 10)  :  D   A   T   A   S   E   C   U   R   E
Plaintext Numeric (0..25) :  3   0  19   0  18   4   2  20  17   4
                             +   +   +   +   +   +   +   +   +   +
Repeating Key "MATH"      :  M   A   T   H   M   A   T   H   M   A
Key Numeric (0..25)       : 12   0  19   7  12   0  19   7  12   0
                             =   =   =   =   =   =   =   =   =   =
Sum Modulo 26             : 15   0  12   7   4   4  21   1   3   4
Ciphertext Letters        :  P   A   M   H   E   E   V   B   D   E
```

3. Step-by-Step Computational Encryption & Decryption

Given Plaintext P = `DATASECURE` (length $N = 10$), Keyword K = `MATH` (length $m = 4$). Repeating keyword generates keystream: **MATHMATHMA**.

Encryption Process ($C_i = (P_i + K_i) \bmod 26$):

- $i = 0$: P_0 = 'D'(3), K_0 = 'M'(12) &implies; $C_0 = (3 + 12) \bmod 26 = 15$ &implies; **P**
- $i = 1$: P_1 = 'A'(0), K_1 = 'A'(0) &implies; $C_1 = (0 + 0) \bmod 26 = 0$ &implies; **A**
- $i = 2$: P_2 = 'T'(19), K_2 = 'T'(19) &implies; $C_2 = (19 + 19) \bmod 26 = 38 \bmod 26 = 12$ &implies; **M**
- $i = 3$: P_3 = 'A'(0), K_3 = 'H'(7) &implies; $C_3 = (0 + 7) \bmod 26 = 7$ &implies; **H**
- $i = 4$: P_4 = 'S'(18), K_4 = 'M'(12) &implies; $C_4 = (18 + 12) \bmod 26 = 30 \bmod 26 = 4$ &implies; **E**
- $i = 5$: P_5 = 'E'(4), K_5 = 'A'(0) &implies; $C_5 = (4 + 0) \bmod 26 = 4$ &implies; **E**
- $i = 6$: P_6 = 'C'(2), K_6 = 'T'(19) &implies; $C_6 = (2 + 19) \bmod 26 = 21$ &implies; **V**
- $i = 7$: P_7 = 'U'(20), K_7 = 'H'(7) &implies; $C_7 = (20 + 7) \bmod 26 = 27 \bmod 26 = 1$ &implies; **B**
- $i = 8$: P_8 = 'R'(17), K_8 = 'M'(12) &implies; $C_8 = (17 + 12) \bmod 26 = 29 \bmod 26 = 3$ &implies; **D**
- $i = 9$: P_9 = 'E'(4), K_9 = 'A'(0) &implies; $C_9 = (4 + 0) \bmod 26 = 4$ &implies; **E**

Generated Ciphertext: PAMHEEVBDE

Decryption Process ($P_i = (C_i - K_i + 26) \bmod 26$):

- $i = 0$: C_0 = 'P'(15), K_0 = 'M'(12) &implies; $P_0 = (15 - 12) \bmod 26 = 3$ &implies; **D**
- $i = 1$: C_1 = 'A'(0), K_1 = 'A'(0) &implies; $P_1 = (0 - 0) \bmod 26 = 0$ &implies; **A**
- $i = 2$: C_2 = 'M'(12), K_2 = 'T'(19) &implies; $P_2 = (12 - 19 + 26) \bmod 26 = 19$ &implies; **T**
- $i = 3$: C_3 = 'H'(7), K_3 = 'H'(7) &implies; $P_3 = (7 - 7) \bmod 26 = 0$ &implies; **A**
- $i = 4$: C_4 = 'E'(4), K_4 = 'M'(12) &implies; $P_4 = (4 - 12 + 26) \bmod 26 = 18$ &implies; **S**
- $i = 5$: C_5 = 'E'(4), K_5 = 'A'(0) &implies; $P_5 = (4 - 0) \bmod 26 = 4$ &implies; **E**
- $i = 6$: C_6 = 'V'(21), K_6 = 'T'(19) &implies; $P_6 = (21 - 19) \bmod 26 = 2$ &implies; **C**
- $i = 7$: C_7 = 'B'(1), K_7 = 'H'(7) &implies; $P_7 = (1 - 7 + 26) \bmod 26 = 20$ &implies; **U**
- $i = 8$: C_8 = 'D'(3), K_8 = 'M'(12) &implies; $P_8 = (3 - 12 + 26) \bmod 26 = 17$ &implies; **R**
- $i = 9$: C_9 = 'E'(4), K_9 = 'A'(0) &implies; $P_9 = (4 - 0) \bmod 26 = 4$ &implies; **E**

Recovered Plaintext: DATASECURE

4. Cryptographic Execution & Verification Table

Index i	Plaintext P_i	P_i Val	Key K_i	K_i Val	Encryption $(P_i + K_i) \bmod 26$	Ciphertext C_i	Decryption $(C_i - K_i + 26) \bmod 26$	Recovered P_i
0	D	3	M	12	(3 + 12) = 15	**P**	(15 − 12) = 3	**D**
1	A	0	A	0	(0 + 0) = 0	**A**	(0 − 0) = 0	**A**
2	T	19	T	19	(19 + 19) mod 26 = 12	**M**	(12 − 19 + 26) = 19	**T**
3	A	0	H	7	(0 + 7) = 7	**H**	(7 − 7) = 0	**A**
4	S	18	M	12	(18 + 12) mod 26 = 4	**E**	(4 − 12 + 26) = 18	**S**
5	E	4	A	0	(4 + 0) = 4	**E**	(4 − 0) = 4	**E**
6	C	2	T	19	(2 + 19) = 21	**V**	(21 − 19) = 2	**C**
7	U	20	H	7	(20 + 7) mod 26 = 1	**B**	(1 − 7 + 26) = 20	**U**
8	R	17	M	12	(17 + 12) mod 26 = 3	**D**	(3 − 12 + 26) = 17	**R**
9	E	4	A	0	(4 + 0) = 4	**E**	(4 − 0) = 4	**E**

Cryptanalytic Note: Although historically considered unbreakable (*le chiffre indéchiffrable*), Charles Babbage and Friedrich Kasiski (1863) cracked Vigenère by identifying repeated ciphertext n-grams to deduce keyword length m, reducing it to m monoalphabetic Caesar ciphers solved via frequency analysis. William Friedman later formalized key-length deduction using the Index of Coincidence (I_C).

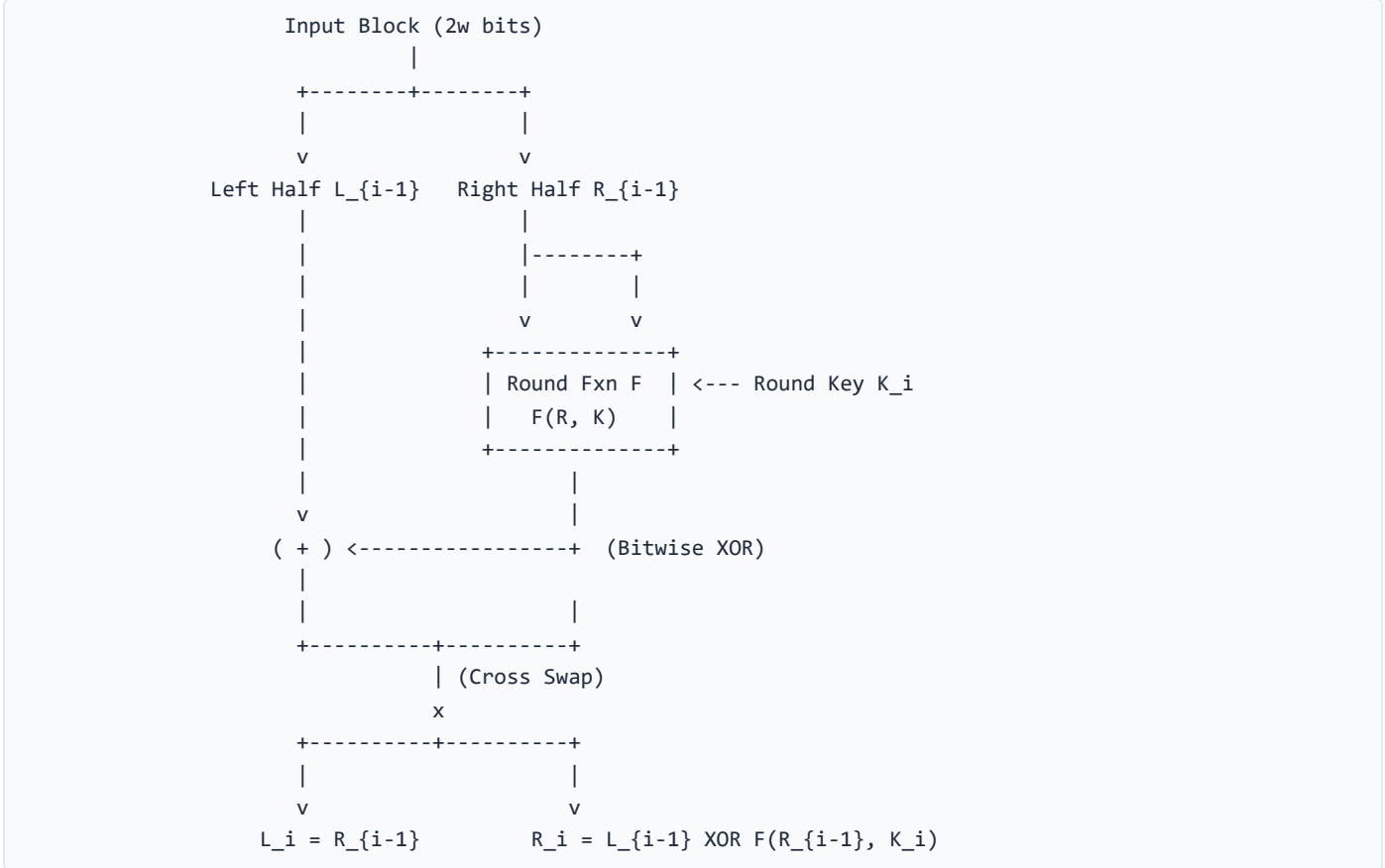
Q20. Explain the Feistel Cipher Structure with a neat diagram. Perform one Feistel round for input block 1010 0110, round key 0011 and function F(R, K) = R XOR K. (8 Marks)

1. Core Concept & Design Principles

The Feistel Cipher Structure, designed by IBM cryptographer Horst Feistel in the early 1970s (originating in Lucifer and standardized in DES), is an architectural framework for designing symmetric block ciphers:

- **Shannon's Fundamental Cryptographic Principles:** The Feistel structure operationalizes Claude Shannon's 1949 criteria for robust block cipher design:
  - **Diffusion:** Spreading statistical patterns of the plaintext across the entire ciphertext so that changing a single plaintext bit alters approximately 50% of ciphertext bits (the avalanche effect).
  - **Confusion:** Making the relationship between the key and resulting ciphertext as complex and non-linear as possible.
- **Invertibility Guarantee:** A critical strength of the Feistel design is that the internal round function *F(R, K)* **does not need to be mathematically invertible**. Because the round structure relies on the self-inverting property of bitwise XOR ($A \oplus B \oplus B = A$), decryption uses the identical algorithm in reverse by applying the round keys in backwards sequence ($K_n, K_{n-1}, \ldots, K_1$).
- **Standard Operational Flow:** An input block of $2w$ bits is split into two halves of length w bits: Left (L_0) and Right (R_0). In each round i:
  $L_i = R_{i-1}, \quad R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$

2. Architecture & Feistel Round Diagram



3. Execution of One Feistel Round

Input Parameters: Input Block = `1010 0110` (8 bits total); Left Half L_0 = `1010`, Right Half R_0 = `0110`; Round Subkey K_1 = `0011`; Round Function $F(R, K) = R \oplus K$.

Step-by-Step Computational Round Execution:

- **Step 1: Compute Round Function Output $F(R_0, K_1)$:**
  $F(R_0, K_1)$ = `0110` $\oplus$ `0011`
  • Bit 3: $0 \oplus 0 = 0$, Bit 2: $1 \oplus 0 = 1$, Bit 1: $1 \oplus 1 = 0$, Bit 0: $0 \oplus 1 = 1$
  $\boldsymbol{F(R_0, K_1)}$ = **`0101`**$_2$
- **Step 2: Assign Next Left Half L_1:**
  By Feistel structural definition: $\boldsymbol{L_1 = R_0}$ = **`0110`**$_2$
- **Step 3: Compute Next Right Half R_1:**
  $R_1 = L_0 \oplus F(R_0, K_1)$ = `1010` $\oplus$ `0101`
  • Bit 3: $1 \oplus 0 = 1$, Bit 2: $0 \oplus 1 = 1$, Bit 1: $1 \oplus 0 = 1$, Bit 0: $0 \oplus 1 = 1$
  $\boldsymbol{R_1}$ = **`1111`**$_2$
- **Step 4: Concatenate Halves for Round Output:**
  Round 1 Output Block = $L_1 \,||\, R_1$ = **`0110 1111`**$_2$
- **Step 5: Decryption Round Verification (Proof of Invertibility):**
  Given (L_1 = `0110`, R_1 = `1111`) with subkey K_1 = `0011`:
  • $R_0 = L_1$ = `0110`$_2$
  • $L_0 = R_1 \oplus F(L_1, K_1)$ = `1111` $\oplus$ (`0110` $\oplus$ `0011`) = `1111` $\oplus$ `0101` = `1010`$_2$
  Restores original input block `1010 0110`$_2$ perfectly.

4. Feistel Structural Parameters & Design Trade-offs Table

Design Parameter	Cryptographic Significance	Standard Practical Value (e.g., DES)	Impact on Security and Performance
Block Size ($2w$)	Granularity of data processed per cycle	64 bits (DES), 128 bits (Twofish)	Larger block sizes prevent codebook and birthday attacks, slightly slowing execution.
Key Size	Cryptographic strength against brute force	56 bits (DES), 128/256 bits (Modern)	Larger key size increases work factor (2^k) to prevent exhaustive search attacks.
Number of Rounds (n)	Multiplicity of confusion/diffusion iterations	16 rounds (DES), 16–32 rounds (Blowfish)	More rounds resist differential and linear cryptanalysis; 16 rounds is standard.
Subkey Generation	Key scheduling complexity	48-bit subkeys generated per round	Highly complex key schedules increase resistance against slide and related-key attacks.
Round Function F	Engine of non-linearity and bit permutation	S-boxes + P-boxes	Greater complexity and avalanche properties yield higher cryptanalytic resistance.
Symmetric HW/SW	Execution efficiency across hardware/software	Identical encryption/decryption logic	Minimizes gate count on VLSI chips and instruction footprint in software.

Q21. Explain the basic working of Elliptic Curve Cryptography (ECC) with a neat diagram. (8 Marks)

1. Core Concept & Mathematical Basis

Elliptic Curve Cryptography (ECC), proposed independently by Neal Koblitz and Victor S. Miller in 1985, is an advanced public-key cryptosystem based on the algebraic structure of elliptic curves defined over finite fields:

- **Finite Field Equation (Weierstrass Form):** Over a prime Galois field $\mathbb{F}_p$ ($p > 3$), an elliptic curve $E(\mathbb{F}_p)$ consists of all points (x, y) satisfying the cubic congruence:

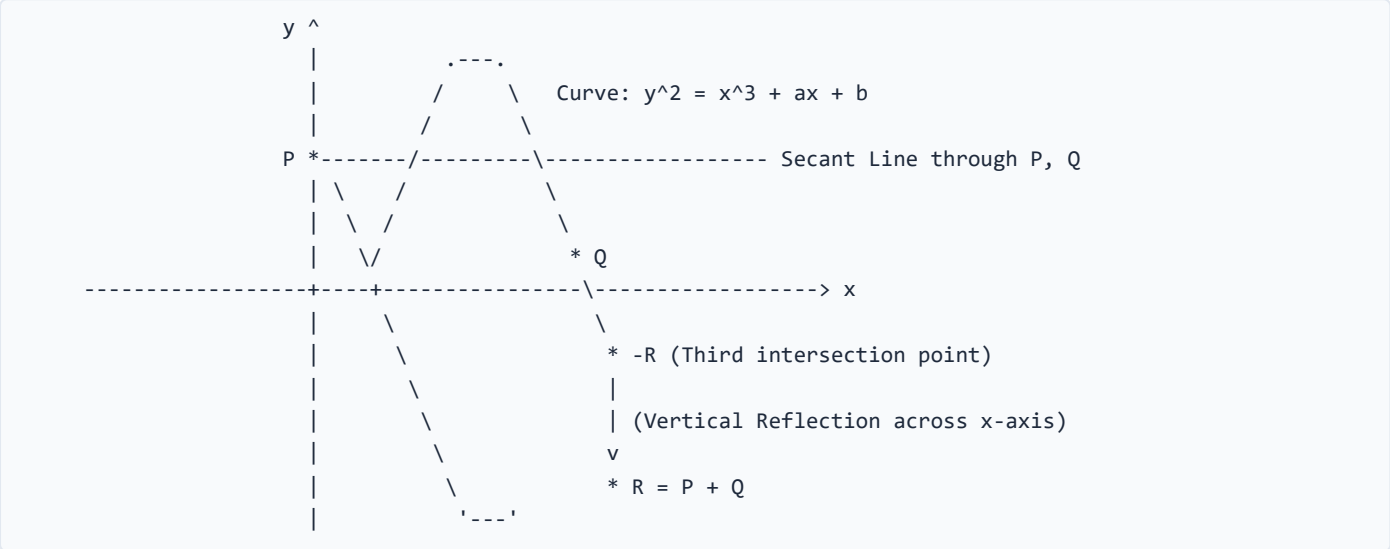
$$y^2 \equiv x^3 + ax + b \bmod p$$

  together with an ideal point at infinity $\mathcal{O}$ serving as the group additive identity element.
- **Non-Singularity Criterion:** The curve coefficients a, b must satisfy the non-singularity condition:

$$4a^3 + 27b^2 \not\equiv 0 \bmod p$$

  This ensures the cubic polynomial has distinct roots, guaranteeing the curve has no cusps or self-intersections, so a unique tangent line exists at every point.
- **Hard Problem (ECDLP):** The cryptographic security of ECC relies on the **Elliptic Curve Discrete Logarithm Problem (ECDLP)**. Given base point G and public key $Q = k \cdot G$ (computed by adding G to itself k times), finding the scalar private key k is computationally infeasible for large primes p (e.g., 256 bits). Unlike RSA, no sub-exponential algorithm (such as index calculus) exists to solve ECDLP on general curves.

2. Geometric Point Arithmetic & Curve Diagram



- **Point Addition ($P + Q = R$ where $P \neq Q$):** Draw a secant line through $P(x_1, y_1)$ and $Q(x_2, y_2)$. The line intersects the cubic curve at a third point $-R(x_3, y'&sub3;)$. Reflecting $-R$ vertically across the x-axis gives group sum $R(x_3, y_3) = P + Q$.
- **Point Doubling ($P + P = 2P$):** Draw the tangent line to the curve at $P(x_1, y_1)$. The tangent intersects at $-R$, which when reflected across the x-axis produces $2P$.
- **Point at Infinity & Inverses:** For point $P(x, y)$, its additive inverse is $-P = (x, -y \bmod p)$. A vertical line through P and $-P$ meets at $\mathcal{O}$, giving $P + (-P) = \mathcal{O}$.

3. Cryptographic Operations: Key Generation & ECDH Key Exchange

- **Domain Parameters:** Parties agree on curve parameters: (p, a, b, G, n, h), where p is field prime, a, b are curve constants, G is base generator point, n is prime order of G ($n \cdot G = \mathcal{O}$), and h is cofactor.
- **Key Pair Generation:** Select private key d_A uniformly from $[1, n - 1]$. Compute public key point $Q_A = d_A \cdot G$ using point multiplication (double-and-add algorithm).
- **ECDH Key Exchange Protocol:**
  - Alice selects d_A, computes public key $Q_A = d_A \cdot G$, and transmits Q_A to Bob.
  - Bob selects d_B, computes public key $Q_B = d_B \cdot G$, and transmits Q_B to Alice.
  - Shared secret calculation: Alice computes $K = d_A \cdot Q_B = d_A(d_B \cdot G)$. Bob computes $K = d_B \cdot Q_A = d_B(d_A \cdot G)$. Both arrive at identical secret point $K = d_A\, d_B\, G$, deriving a shared AES key from its x-coordinate.

4. Security & Performance Comparison: ECC vs RSA

Feature Dimension	Elliptic Curve Cryptography (ECC)	Rivest-Shamir-Adleman (RSA)	Architectural Advantage of ECC
Mathematical Problem	Elliptic Curve Discrete Logarithm (ECDLP)	Integer Factorization Problem (IFP)	ECDLP has no sub-exponential attack algorithms
NIST Equivalent Key Size	**256 bits**	**3072 bits**	12x reduction in key length for identical security
High Security Key Size	**384 bits**	**7680 bits**	20x key size savings at ultra-high security levels
Computational Overhead	Low processing power; fast scalar ops	High processing power; large modular exp	Ideal for constrained IoT, smartcards, and mobile
Bandwidth & Storage	Extremely compact signatures and keys	Large certificates, signatures, and keys	Reduces network payload and handshake latency
Power Consumption	Minimal battery drain	Significant battery and CPU utilization	Essential for battery-operated wireless devices

---

Q22. Explain the working and organisation of PGP Key Rings. (8 Marks)

1. Core Concept of PGP & Decentralized Key Management

Pretty Good Privacy (PGP), created by Phil Zimmermann in 1991 and standardized by the IETF under RFC 4880 (OpenPGP), is a hybrid cryptosystem providing end-to-end confidentiality, data integrity, non-repudiation, and authentication for email and file storage:

- **The Web of Trust (WoT) Paradigm:** Unlike traditional X.509 architectures that mandate a hierarchical, centralized Public Key Infrastructure (PKI) governed by commercial Certificate Authorities (CAs), PGP pioneered a decentralized **Web of Trust**. Individual users validate and digitally sign each other's public keys, establishing transitive trust networks without single points of failure.
- **Dual Key Ring Architecture:** To manage asymmetric cryptographic keys efficiently on a local host, every PGP installation maintains two distinct local database structures:
  1. **Private Key Ring:** Stores the public/private key pairs owned by the local user.
  2. **Public Key Ring:** Stores the public keys of all external correspondents and contacts known to the local user, along with cryptographic trust indicators.

2. Structural Schema & Organisation of PGP Key Rings

```
+-------------------------------------------------------------------------------------------------+
|                                   PGP PRIVATE KEY RING                                          |
+---------------------+-------------+------------------+-----------------------------+--------------+
| Timestamp           | Key ID      | Public Key       | Encrypted Private Key       | User ID      |
| (Creation Date/Time)| (Low 64-bit)| (RSA/DSA/ECDSA)  | (Symmetric Encrypted via PW)| (Name & Mail)|
+---------------------+-------------+------------------+-----------------------------+--------------+

+-------------------------------------------------------------------------------------------------+
|                                    PGP PUBLIC KEY RING                                          |
+-------------+--------+------------+-------------+---------+---------------+-------------+---------+
| Timestamp   | Key ID | Public Key | Owner Trust | User ID | Key Signatures| Sig Trust   | Key Leg |
| (Entry date)| (64-bit|(Public Key | (Assigned by|(Email & | (Certifying   | (Trust in   | (Trust  |
|             | of Pub)| parameters)| local user) | Name)   | signatures)   | signer)     | Score)  |
+-------------+--------+------------+-------------+---------+---------------+-------------+---------+
```

3. Field Specifications & Internal Working Mechanics

- **Organisation of the Private Key Ring:**
  - **Timestamp:** Records the exact date and time the key pair was generated.
  - **Key ID:** A 64-bit identifier formed from the least significant 64 bits of the public key ($K_{pub} \bmod 2^{64}$). This allows message headers to identify the required decryption key without attaching bulky public keys.
  - **Public Key Component:** Contains complete public parameters (e.g., modulus n and exponent e).
  - **Encrypted Private Key Component:** The private key parameter (d) is stored securely. The user enters a secret passphrase, hashed via SHA-256 to derive a symmetric key. The private key is encrypted using CAST-128 or AES-256 in CBC mode, protecting it against disk theft.
  - **User ID:** Typically the user's legal name and primary email address (e.g., `Alice <alice@example.com>`).
- **Organisation of the Public Key Ring:**
  - **Timestamp, Key ID, Public Key, User ID:** Identify and store external contacts' public cryptographic material.
  - **Owner Trust (Trust Flag):** Assigned manually by the user to reflect confidence in this contact as an introducer who signs third-party keys: *Undefined/Unknown*, *Untrusted*, *Marginal Trust*, or *Complete Trust*.
  - **Key Signatures:** Digital signatures attached by third-party endorsers certifying the binding between User ID and public key.
  - **Signature Trust:** Cached trust level derived from the Owner Trust of the signer.
  - **Key Legitimacy Field:** Computed automatically by PGP: A key is legitimate if signed by ≥ 1 Completely Trusted introducer, OR ≥ 2 Marginally Trusted introducers.

4. Operational Message Workflow & Key Ring Comparison

- **Sending an Encrypted & Signed Email:**
  1. The sender signs the plaintext using their private key (retrieved from Private Key Ring via passphrase).
  2. A random one-time symmetric session key K_s encrypts the message via AES.
  3. The recipient's public key from the Public Key Ring encrypts session key K_s. The recipient's Key ID is attached.
- **Receiving & Verifying an Email:**
  1. The recipient extracts Key ID from packet header, finds matching private key in Private Key Ring, decrypts it with passphrase, and recovers session key K_s.
  2. The message is decrypted using K_s. Sender's Key ID fetches their public key from Public Key Ring to verify the digital signature.

Architectural Dimension	PGP Private Key Ring	PGP Public Key Ring
Purpose	Stores local user's own key pairs	Stores public keys of communicating associates
Confidentiality Level	High sensitivity; private keys are passphrase-encrypted	Public data; keys are unencrypted and shareable
Trust Evaluation	Not applicable (local owner is implicitly trusted)	Dynamic calculation of Key Legitimacy via Web of Trust
Lookup Mechanism	Indexed by 64-bit Key ID to decrypt incoming session keys	Indexed by User ID or Key ID to encrypt and verify
Database Operations	Rare additions (only when generating new key pairs)	Frequent additions (importing contacts' public keys)

Q23. Explain IPSec Transport Mode and Tunnel Mode with neat diagrams. (8 Marks)

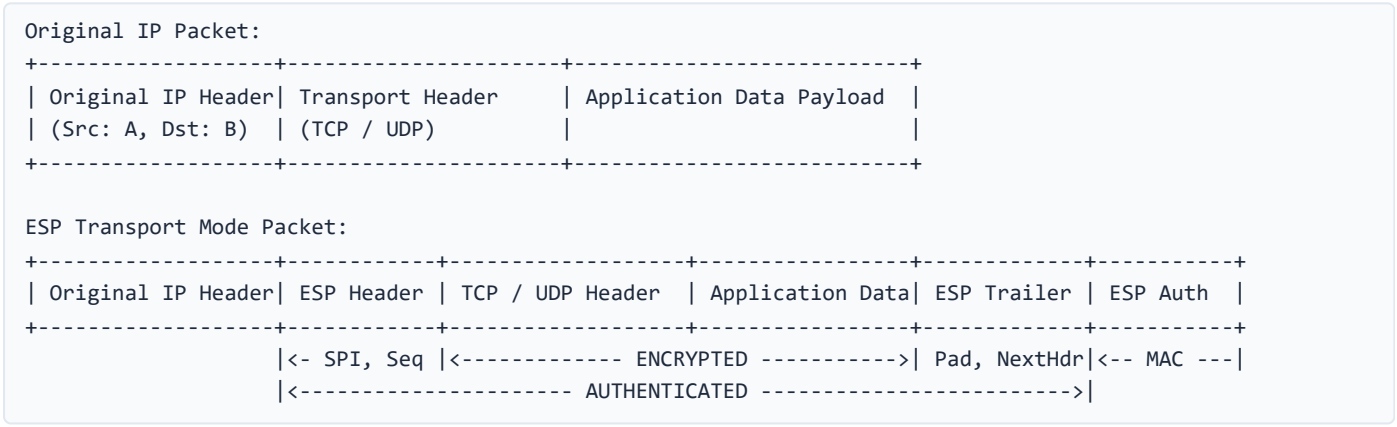
1. Core Concept & IPsec Architectural Framework

Standardized by the IETF in RFC 4301, Internet Protocol Security (IPsec) is a comprehensive suite of cryptographic protocols operating at the **Network Layer (OSI Layer 3)** of the TCP/IP protocol stack:

- **Layer 3 Architectural Advantage:** Because IPsec operates transparently below the Transport Layer (TCP/UDP), it provides robust, application-independent security services (confidentiality, authentication, data integrity, and anti-replay) for all network traffic without necessitating alterations to end-user software, applications, or upper-layer protocols.
- **Core Security Protocols:**
  - **Authentication Header (AH, IP Protocol 51):** Provides connectionless data integrity, data origin authentication, and anti-replay protection across the IP packet, but provides **zero confidentiality** (no encryption).
  - **Encapsulating Security Payload (ESP, IP Protocol 50):** Provides data confidentiality through symmetric block encryption (e.g., AES-CBC, AES-GCM), optional data origin authentication, message integrity (via HMAC-SHA-256), and anti-replay protection.
- **Two Operational Modes:** Depending on network topology and endpoint termination, IPsec encapsulates packets using either **Transport Mode** or **Tunnel Mode**.

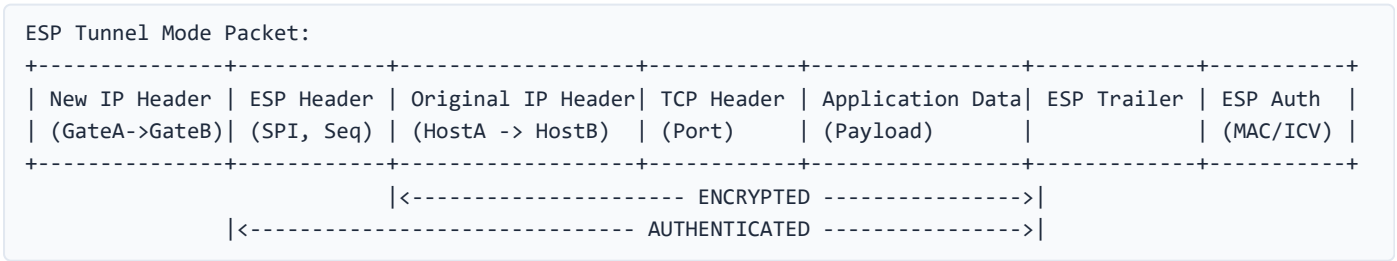
2. Transport Mode Architecture & Packet Format

- **Scope & Topology:** Designed exclusively for **host-to-host (end-to-end)** communications between two dedicated endpoints (e.g., workstation connecting directly to a database server). Both participating endpoints must support, configure, and execute IPsec.
- **Operational Mechanism:** Protects only the **upper-layer payload** (the transport protocol header such as TCP/UDP and the application data). The original IP header is retained as the outermost header without encryption, allowing intermediate routers across the Internet to inspect destination IP addresses and route packets normally.



3. Tunnel Mode Architecture & Packet Format

- **Scope & Topology:** Designed for **gateway-to-gateway (site-to-site VPNs)** or **host-to-gateway (remote-access VPNs)** between security appliances (firewalls, routers, or VPN concentrators). Internal client hosts do not require IPsec software.
- **Operational Mechanism:** Protects the **entire original IP packet**, including the original IP header. The complete original IP packet is encrypted and encapsulated as the payload of a brand-new IP packet. A **New IP Header** is prepended to the packet, specifying the external gateway addresses as source and destination. This completely hides internal network IP topologies and routing structures from intermediate public networks.



4. Comprehensive Architectural Comparison Matrix

Evaluation Dimension	IPsec Transport Mode	IPsec Tunnel Mode
Typical Deployment	End-to-end host-to-host communications	Site-to-site VPNs & Remote access client-to-gateway
Protected Portion	Upper-layer payload only (TCP/UDP + Data)	Entire original IP packet (Header + Payload)
IP Header Handling	Preserves original IP header intact	Encapsulates original header; adds New Outer IP Header
IP Address Privacy	None; source & destination IPs exposed to routers	High; internal IP addresses hidden behind gateway IPs
Packet Overhead	Minimal (only ESP header, trailer, and ICV added)	Higher (adds 20-byte New IP Header plus ESP fields)
Host Configuration	Every participating host must run IPsec software	Executed entirely by boundary security gateways
NAT Traversal (NAT-T)	Challenging; original TCP/IP checksums break	Handled seamlessly via UDP encapsulation (port 4500)
Processing Overhead	Lower CPU overhead on endpoints	Centralized cryptographic processing on VPN gateways

---

MODEL QUESTION PAPER – 4

Q17. Find the particular solution and general solution of the linear Diophantine equation: 21x + 14y = 35.

1. Solvability Criterion

A linear Diophantine equation `ax + by = c` with integer coefficients has integer solutions if and only if `d = gcd(a, b)` divides `c` (`d | c`).

For `21x + 14y = 35`, we compute `gcd(21, 14)` via Euclidean algorithm:

```
21 = 1 × 14 + 7
14 = 2 × 7 + 0  ⇒  d = gcd(21, 14) = 7
```

Since `35 / 7 = 5`, `7 | 35` holds true. Thus, the equation possesses infinitely many integer solutions.

2. Particular Solution Derivation

Dividing by `d = 7` yields coprime equation `3x + 2y = 5`, where `gcd(3, 2) = 1`. Extended Euclidean Algorithm gives: `3(1) + 2(-1) = 1`. Multiplying by 5 gives `3(5) + 2(-5) = 5`. By inspection, setting `x = 1` yields `3(1) + 2(1) = 5`.

- **Base Point:** `(x0, y0) = (1, 1)`
- **Check:** `21(1) + 14(1) = 21 + 14 = 35`

3. General Solution Formulation

For `3x + 2y = 5` and base solution `(1, 1)`, the general solution for parameter `t ∈ ℤ` is:

- `x(t) = x0 + (b / d)t = 1 + 2t`
- `y(t) = y0 - (a / d)t = 1 - 3t`

Proof: `21(1 + 2t) + 14(1 - 3t) = 21 + 42t + 14 - 42t = 35`. Parameter `t` cancels identically, confirming all integer pairs are valid solutions.

4. Solution Space Matrix

Parameter	x = 1 + 2t	y = 1 - 3t	Evaluation: 21x + 14y
t = -1	-1	4	21(-1) + 14(4) = 35
t = 0	1	1	21(1) + 14(1) = 35 (Base Solution)
t = 1	3	-2	21(3) + 14(-2) = 35
t = 2	5	-5	21(5) + 14(-5) = 35

Q18. Find the inverse of the matrix A = [[2, 1], [1, 1]] and verify the result.

1. Invertibility Criterion

A square matrix `A` has a multiplicative inverse A^{-1} if and only if its determinant is non-zero (`det(A) ≠ 0`). When `det(A) = 1` , the matrix is unimodular, ensuring its inverse contains purely integer entries without modular fraction division.

```
A = [ [2, 1],
      [1, 1] ]
```

Determinant Calculation: `det(A) = (ad - bc) = (2 × 1) - (1 × 1) = 2 - 1 = 1` . Since `det(A) = 1 ≠ 0` , matrix `A` is non-singular and strictly invertible across $\mathbb{R}$, $\mathbb{Z}$, and $\mathbb{Z}_{26}$.

2. Inverse Matrix Derivation

By classical matrix inversion theory: $A^{-1} = (1 / det(A)) \times adj(A)$. For a 2×2 matrix, the adjugate matrix swaps main diagonal elements and negates off-diagonal elements:

```
adj(A) = [ [ d, -b],
           [-c,  a] ] = [ [ 1, -1],
                          [-1,  2] ]
```

Substituting `det(A) = 1` : $A^{-1} = (1 / 1) \times [[1, -1], [-1, 2]] = [[1, -1], [-1, 2]]$.

3. Two-Sided Matrix Verification

Correctness requires confirming that both products yield identity matrix I_2 :

- **Right Product ($A \times A^{-1}$):** `[[2, 1], [1, 1]] × [[1, -1], [-1, 2]] = [[2 - 1, -2 + 2], [1 - 1, -1 + 2]] = [[1, 0], [0, 1]] =` I_2 .
- **Left Product ($A^{-1} \times A$):** `[[1, -1], [-1, 2]] × [[2, 1], [1, 1]] = [[2 - 1, 1 - 1], [-2 + 2, -1 + 2]] = [[1, 0], [0, 1]] =` I_2 .

Both products strictly evaluate to I_2 , validating the calculated inverse.

4. Cryptographic Significance Matrix

Property	Value	Cryptographic Role
Determinant	det(A) = 1	Unit determinant avoids fraction reduction; preserves discrete lattice volume.
Inverse Matrix	[[1, -1], [-1, 2]]	Acts as the exact decryption transformation in polygraphic block ciphers.
Modulo 26 Role	gcd(1, 26) = 1	Invertible in Z26; enables Hill cipher decipherment without scaling factors.

Q19. Explain Two-Key and Three-Key Triple DES with neat diagrams.

1. Core Concept & Motivation

Triple DES (TDEA / 3DES, ANSI X9.52, NIST SP 800-67) was developed to remediate standard DES vulnerabilities caused by its 56-bit key space. Double DES was rejected because Diffie and Hellman proved it vulnerable to Meet-in-the-Middle (MitM) attacks, reducing security from 2^{112} to 2^{57} operations.

To defeat MitM attacks while maintaining backward compatibility, 3DES uses an **Encrypt-Decrypt-Encrypt (EDE)** sequence:

- **Encryption:** $C = E_{K3}(D_{K2}(E_{K1}(P)))$
- **Decryption:** $P = D_{K1}(E_{K2}(D_{K3}(C)))$
- **Backward Compatibility:** When $K1 = K2 = K3$, the middle decryption cancels the first encryption ($E_K(D_K(E_K(P))) = E_K(P)$), executing single DES.

2. Triple DES Architecture Diagrams

```
Two-Key Triple DES (EDE2 - 112 bits, Keys: K1, K2, K1):
P ──▶ [DES Encrypt K1] ──▶ [DES Decrypt K2] ──▶ [DES Encrypt K1] ──▶ C
C ──▶ [DES Decrypt K1] ──▶ [DES Encrypt K2] ──▶ [DES Decrypt K1] ──▶ P

Three-Key Triple DES (EDE3 - 168 bits, Keys: K1, K2, K3):
P ──▶ [DES Encrypt K1] ──▶ [DES Decrypt K2] ──▶ [DES Encrypt K3] ──▶ C
C ──▶ [DES Decrypt K3] ──▶ [DES Encrypt K2] ──▶ [DES Decrypt K1] ──▶ P
```

3. Keying Options & Security Characteristics

- **Two-Key 3DES (Option 2):** Employs two independent keys with $K3 = K1$ (112 key bits). Advanced chosen-plaintext attacks reduce effective security to ~80 bits. Deprecated by NIST after 2015.
- **Three-Key 3DES (Option 1):** Employs three mutually independent keys $K1 \neq K2 \neq K3$ (168 key bits). MitM attacks require 2^{112} operations, providing 112-bit effective security.
- **Block Size Vulnerability:** Both options retain 64-bit blocks. Under the birthday paradox, encrypting 2^{32} blocks (~32 GB) risks Sweet32 collision attacks, prompting replacement by AES.

4. Specification Comparison Matrix

Parameter	Single DES	Two-Key 3DES (EDE2)	Three-Key 3DES (EDE3)	AES-128
Key Length	56 bits	112 bits	168 bits	128 bits
Effective Security	56 bits (Insecure)	~80 bits (Deprecated)	112 bits (Legacy)	128 bits (Secure)
Block Size	64 bits	64 bits	64 bits	128 bits
Internal Rounds	16 Feistel rounds	48 Feistel rounds	48 Feistel rounds	10 SPN rounds
NIST Status	Withdrawn (1999)	Deprecated (2015)	Retired (2023)	Active Standard

---

Q20. Explain the Hill Cipher. Encrypt the plaintext HI using key matrix [[3, 3], [2, 5]] and decrypt the resulting ciphertext.

1. Core Concept & Mathematical Formulation

Invented by Lester S. Hill in 1929, the Hill Cipher is a polygraphic substitution cipher that encrypts blocks of letters simultaneously using modular linear algebra over $\mathbb{Z}_{26}$. Letters are mapped using standard indexing: $A = 0, B = 1, C = 2, \ldots, Z = 25$.

- **Encryption Formula:** $C = (K \times P) \bmod 26$, where P is a plaintext vector and K is an invertible key matrix.
- **Decryption Formula:** $P = (K^{-1} \times C) \bmod 26$, requiring modular inverse matrix K^{-1}.
- **Invertibility Condition:** Matrix K is invertible modulo 26 if and only if $\gcd(\det(K), 26) = 1$.

2. Key Matrix Inversion Modulo 26

Given key matrix $K = [[3, 3], [2, 5]]$. Evaluating determinant: $\det(K) = (3 \times 5) - (3 \times 2) = 15 - 6 = 9$. Since $\gcd(9, 26) = 1$, K is invertible in $\mathbb{Z}_{26}$. Finding modular multiplicative inverse of 9 modulo 26: $9d \equiv 1 \pmod{26} \Rightarrow 9 \times 3 = 27 = (1 \times 26) + 1 \equiv 1 \pmod{26} \Rightarrow \det(K)^{-1} = 3 \pmod{26}$.

```
adj(K) = [ [ 5, -3],
           [-2,  3] ] ≡ [ [ 5, 23],
                         [24,  3] ] (mod 26)

K^-1 = 3 × [ [ 5, 23],
             [24,  3] ] = [ [15, 69],
                            [72,  9] ] ≡ [ [15, 17],
                                           [20,  9] ] (mod 26)
```

Verification: $K \times K^{-1} = [[3, 3], [2, 5]] \times [[15, 17], [20, 9]] = [[105, 78], [130, 79]] \equiv [[1, 0], [0, 1]] \pmod{26}$.

3. Step-by-Step Encryption of Plaintext "HI"

Numerical mapping: $H \to 7$, $I \to 8$ ⇒ $P = [[7], [8]]$.

```
C = (K × P) mod 26 = [ [3, 3],  ×  [ [7],
                       [2, 5] ]       [8] ] (mod 26)

c1 = (3 × 7) + (3 × 8) = 21 + 24 = 45 ≡ 45 - 26 = 19 (mod 26) → T
c2 = (2 × 7) + (5 × 8) = 14 + 40 = 54 ≡ 54 - 52 = 2  (mod 26) → C
```

Ciphertext Output: **"TC"**.

4. Step-by-Step Decryption & Verification

Numerical mapping: $T \to 19$, $C \to 2$ ⇒ $C = [[19], [2]]$.

```
P = (K^-1 × C) mod 26 = [ [15, 17],  ×  [ [19],
                          [20,  9] ]       [ 2] ] (mod 26)

p1 = (15 × 19) + (17 × 2) = 285 + 34 = 319 ≡ 319 - 312 = 7 (mod 26) → H
p2 = (20 × 19) + (9 × 2)  = 380 + 18 = 398 ≡ 398 - 390 = 8 (mod 26) → I
```

Recovered Plaintext: **"HI"** (exact polygraphic reconstruction verified).

Q21. Explain the Rabin cryptosystem with its key generation, encryption and decryption procedures.

1. Core Concept & Provable Security

Introduced by Michael O. Rabin in 1979, the Rabin cryptosystem is an asymmetric scheme based on modular squaring (quadratic residues). It is **provably equivalent to integer factorization**: solving $c \equiv m^2 \pmod{n}$ without prime factors is computationally identical to factoring $n = pq$. Unlike RSA, Rabin's factoring equivalence is mathematically proven. Encryption requires a single modular squaring, but decryption produces four ambiguous roots requiring disambiguation.

2. Architecture & Operational Flow

```
KeyGen: Primes p, q ≡ 3 (mod 4) ──▶ Public: n = pq, Private: (p, q)
Encryption: Message m (with redundancy) ──▶ c = m^2 mod n ──▶ Ciphertext c
Decryption: c ──▶ Roots mod p, q ──▶ Chinese Remainder Theorem ──▶ 4 Roots ──▶ Plaintext m
```

3. Cryptographic Procedures

- **Key Generation:**
  1. Select large Blum primes p, q with $p \equiv 3 \pmod 4$ and $q \equiv 3 \pmod 4$.
  2. Compute modulus $n = p \times q$.
  3. Public Key is n; Private Key is (p, q).

- **Encryption:**
  1. Append redundancy bits to message `m` (`m = data || redundancy`).
  2. Compute ciphertext: `c = m^2 mod n` .
  3. Encryption requires only one modular multiplication (`O(k^2)`).
- **Decryption:**
  1. Compute square roots mod `p` and `q` : `m_p = c^((p+1)/4) mod p` , `m_q = c^((q+1)/4) mod q` .
  2. Find Bézout integers: `y_p × p + y_q × q = 1` .
  3. Construct four roots modulo `n` via Chinese Remainder Theorem: `r_1 = (y_p × p × m_q + y_q × q × m_p) mod n` , `r_2 = n - r_1` , `r_3 = (y_p × p × m_q - y_q × q × m_p) mod n` , `r_4 = n - r_3` .
  4. Select the candidate matching the redundancy pattern to recover `m` .

4. Cryptosystem Comparison Matrix

Feature	Rabin	RSA	ElGamal
Hard Problem	Quadratic Residues / Factoring	RSA Problem	Discrete Logarithm
Factoring Equivalence	Provably equivalent	Not proven	Unrelated
Encryption Speed	Ultrafast (1 squaring)	Fast (e = 65537)	Slow (2 exponentiations)
Decryption Mapping	4-to-1 (requires padding)	1-to-1 bijection	1-to-1 bijection

---

Q22. Explain the SHA family of hash functions: SHA-0, SHA-1, SHA-2 and SHA-3.

1. Core Concept & Hash Principles

The Secure Hash Algorithm (SHA) family includes cryptographic hash standards developed by NIST under FIPS PUB 180 and FIPS PUB 202. Hash functions compress arbitrary-length inputs into fixed-length digests, satisfying three properties:

- **Preimage Resistance:** Infeasible to invert digest `h` to find original message `m` .
- **Second Preimage Resistance:** Infeasible to find `m2 ≠ m1` producing identical digest `H(m1)` .
- **Collision Resistance:** Infeasible to find any distinct pair `m1, m2` with identical digest.

2. Architecture Comparison (Merkle-Damgård vs. Sponge)

```
Merkle-Damgård (SHA-0, SHA-1, SHA-2):
Message ⟶ [Pad & Split Blocks] ⟶ [Compression f (IV)] ⟶ [f] ⟶ Fixed Digest

Sponge Construction (SHA-3 / Keccak):
Message ⟶ [Pad & Split Blocks] ⟶ [XOR with Rate r] ⟶ [Keccak-f] ⟶ [Extract Digest]
```

3. Evolution of SHA Generations

- **SHA-0 (1993, FIPS 180):** Produced a 160-bit digest from 512-bit blocks across 80 rounds. Withdrawn in 1995 due to an undisclosed message expansion flaw permitting collision attacks.
- **SHA-1 (1995, FIPS 180-1):** Added a 1-bit circular shift to message expansion. Broken theoretically in 2005 and practically by Google's "SHAttered" attack in 2017 (`2^63.1` operations). Completely deprecated.
- **SHA-2 (2001, FIPS 180-4):** Uses Merkle-Damgård with Davies-Meyer compression. Includes SHA-224, SHA-256 (64 rounds), SHA-384, and SHA-512 (80 rounds). Employs bitwise logic (`Ch, Maj`) and rotations (`Σ, σ`). Fully secure.
- **SHA-3 (2015, FIPS 202):** Implements the **Keccak Sponge Construction** over a 1600-bit state. Native immunity to length-extension attacks without HMAC wrappers. Includes SHA3-224 to SHA3-512 and SHAKE XOFs.

4. Specification Comparison Matrix

Algorithm	Digest	Block Size	Word Size	Rounds	Collision Status
SHA-0	160 bits	512 bits	32 bits	80	Broken (Withdrawn 1995)
SHA-1	160 bits	512 bits	32 bits	80	Broken (Collided 2017)
SHA-256	256 bits	512 bits	32 bits	64	Fully Secure (Standard)
SHA-512	512 bits	1024 bits	64 bits	80	Fully Secure (High Assurance)
SHA3-256	256 bits	1088 bits (rate)	64 bits	24	Fully Secure (Sponge)

Q23. Explain the major components and working of Public-Key Infrastructure (PKI) with a neat diagram.

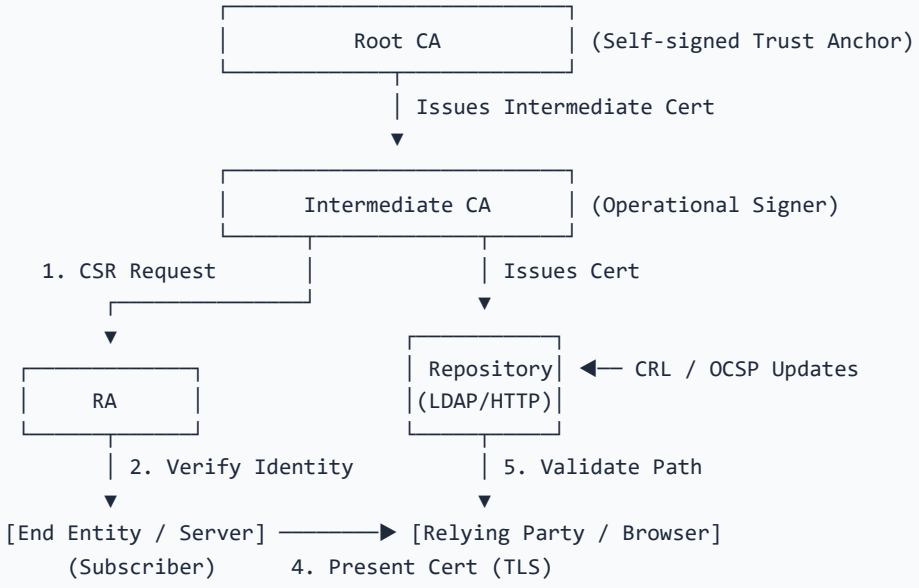
1. Core Concept & Identity Binding

Public-Key Infrastructure (PKI), defined under ITU-T Recommendation X.509 and RFC 5280, is an integrated system of hardware, software, policies, and cryptographic protocols that creates, manages, distributes, and revokes digital certificates. Asymmetric cryptography lacks inherent ownership proof, risking man-in-the-middle impersonation. PKI solves this by binding public keys to verified entities (users, servers, organizations) through digitally signed X.509 certificates.

2. Major Components of PKI

- **Certificate Authority (CA):** Trusted third-party anchor that digitally signs and issues certificates, managing cryptographic key hierarchies.
- **Registration Authority (RA):** Front-end verifier that authenticates applicant identities (domain control, identity proof) before approving CSRs for CA signing.
- **Certificate Repository:** Public directory (LDAP, HTTP) storing active certificates and Certificate Revocation Lists for retrieval.
- **Revocation Infrastructure (CRL & OCSP):** Mechanism to invalidate certificates prior to expiration. CRL is a periodic signed blacklist; OCSP provides real-time status validation.
- **End Entities:** Includes *Subscribers* (entities owning private keys and presenting certificates) and *Relying Parties* (browsers validating trust chains).

3. PKI Architecture & Trust Hierarchy Diagram



4. PKI Operational Lifecycle & Security Matrix

Lifecycle Stage	Responsible Entity	Core Procedure	Security Control
1. Key Generation	Subscriber	Generates asymmetric key pair; creates CSR (PKCS#10).	Private key never leaves secure local storage (HSM/TPM).
2. Identity Vetting	Registration Authority	Vets applicant identity and domain control.	Prevents unauthorized spoofing and fraudulent issuance.
3. Certificate Signing	Certificate Authority	Binds identity with public key and signs with CA private key.	CA signature guarantees tamper-proof authenticity.
4. Chain Validation	Relying Party	Validates hierarchical signatures recursively up to Root CA.	Cryptographic signature chain establishes trust.
5. Revocation Check	Relying Party / OCSP	Queries OCSP or checks CRL for revoked status.	Immediate termination of trust for compromised keys.

MODEL QUESTION PAPER – 5

Q17. Solve the system of linear congruences using the elimination method: x + y = 1 (mod 5), 2x + y = 4 (mod 5).

1. Mathematical Foundations & Solvability Criteria

A **system of linear congruences** comprises simultaneous modular equations over a shared residue ring $\mathbb{Z}_m$. When the modulus is prime (here $p = 5$), $\mathbb{Z}_5$ forms a Galois Field GF(5), ensuring every non-zero element has a unique multiplicative inverse.

- **Matrix Formulation:** The system is represented as $A\mathbf{x} \equiv \mathbf{b} \pmod 5$, where $A = [[1, 1], [2, 1]]$, $\mathbf{x} = [x, y]^T$, and $\mathbf{b} = [1, 4]^T$.
- **Solvability Criterion:** The determinant $\det(A) = (1)(1) - (1)(2) = -1 \equiv 4 \pmod 5$. Because $\gcd(\det(A), 5) = \gcd(4, 5) = 1$, A is non-singular over $\mathbb{Z}_5$, guaranteeing a unique residue solution modulo 5.

2. Algorithmic Elimination Workflow

```
[Given: (1) x + y = 1 (mod 5)  and  (2) 2x + y = 4 (mod 5)]
                            |
                            v
[Subtract Eq (1) from Eq (2)] ----> (2x + y) - (x + y) = 4 - 1 mod 5
                            |
                            v
              [Eliminates y] ------> x = 3 mod 5  ===>  x ≡ 3 (mod 5)
                            |
                            v
[Back-Substitute x=3 in Eq (1)] --> 3 + y = 1 mod 5
                            |
                            v
    [Solve for Residue y] --------> y = -2 = 3 mod 5  ===>  y ≡ 3 (mod 5)
                            |
                            v
[Verification in System] ---------> 3 + 3 = 6 ≡ 1 (mod 5)  [TRUE]
                                    2(3) + 3 = 9 ≡ 4 (mod 5)  [TRUE]
```

3. Step-by-Step Algebraic Elimination & Calculation

- **Step 1: System Definition:** Identify the given simultaneous congruences:

> (1) x + y ≡ 1 (mod 5)
> (2) 2x + y ≡ 4 (mod 5)

- **Step 2: Elimination of Variable y:** Subtract congruence (1) directly from congruence (2):

  ```
  (2x + y) − (x + y) ≡ (4 − 1) (mod 5) ⇒ x ≡ 3 (mod 5)
  ```

  Thus, **x = 3 mod 5**.
- **Step 3: Substitution to Find y:** Substitute $x = 3$ into congruence (1):

  ```
  3 + y ≡ 1 (mod 5) ⇒ y ≡ 1 − 3 ≡ −2 (mod 5)
  ```

  Converting negative residue −2 to its standard least positive residue modulo 5:

  ```
  y = −2 + 5 = 3 (mod 5) ⇒ y ≡ 3 (mod 5)
  ```

  Thus, **y = -2 = 3 mod 5**.
- **Step 4: Verification in Original Equations:**

  ```
  Check Eq (1): 3 + 3 = 6 = 1 mod 5 (Since 6 ≡ 1 mod 5)
  Check Eq (2): 2(3) + 3 = 6 + 3 = 9 = 4 mod 5 (Since 9 ≡ 4 mod 5)
  ```

  Solution: x = 3 (mod 5), y = 3 (mod 5) ⇒ x ≡ 3 (mod 5), y ≡ 3 (mod 5)

4. Summary Verification & Parametric Solution Set

The complete integer solution family across $\mathbb{Z}$ is given parametrically by $x = 3 + 5k_1$ and $y = 3 + 5k_2$ for all $k_1, k_2 \in \mathbb{Z}$.

Equation	Substitution (x = 3, y = 3)	Modular Evaluation	Target Value	Status
(1) x + y ≡ 1 (mod 5)	3 + 3 = 6	6 mod 5 = 1 ⇒ 6 ≡ 1	1	Verified
(2) 2x + y ≡ 4 (mod 5)	2(3) + 3 = 9	9 mod 5 = 4 ⇒ 9 ≡ 4	4	Verified
Inverse Check: $A^{-1}b$	[4, 1; 2, 4] × [1; 4] mod 5	[8; 18] mod 5 = [3; 3]	[3; 3]	Unique

---

Q18. Explain the Affine Cipher. Encrypt the plaintext DATA using multiplicative key K1 = 5 and additive key K2 = 8. Decrypt the resulting ciphertext.

1. Mathematical Principles & Solvability Criteria

The **Affine Cipher** is a monoalphabetic substitution cryptosystem combining modular multiplication and addition over $\mathbb{Z}_{26}$. Numerical character assignments follow A = 0, B = 1, ..., Z = 25.

- **Encryption Function:** $C = (K_1 \times P + K_2) \bmod 26$, here $C = (5P + 8) \bmod 26$.
- **Decryption Function:** $P = K_1^{-1} \times (C - K_2) \bmod 26$, where K_1^{-1} is the modular multiplicative inverse of K_1.
- **Solvability Criterion:** Invertibility requires $\gcd(K_1, 26) = 1$. The key space has $\varphi(26) \times 26 = 12 \times 26 = 312$ keys. Because $\gcd(5, 26) = 1$, $K_1 = 5$ is valid.

2. Encryption & Decryption Workflow Architecture

```
Plaintext:           D(3)         A(0)         T(19)        A(0)
                       |            |            |            |
Multiply (x 5):       15            0           95            0
Add Shift (+ 8):      23            8          103            8
Modulo 26:            23            8           25            8
Ciphertext:          X(23)        I(8)         Z(25)        I(8)  ===> "XIZI"
                      ========================================
Inverse Key Derivation: 5^-1 = 21 mod 26  (since 5 * 21 = 105 = 4*26 + 1 ≡ 1)
Subtract Shift (- 8):  15            0           17            0
Multiply (x 21):      315            0          357            0
Modulo 26:              3            0           19            0
Decrypted Plaintext:  D(3)         A(0)         T(19)        A(0)  ===> "DATA"
```

3. Step-by-Step Encryption and Decryption Procedures

- **Character Numerical Values:** D(3), A(0), T(19), A(0).
- **Encryption using C = (5P + 8) mod 26:**
  - For D(3): C = (5 × 3 + 8) mod 26 = 23 mod 26 = 23 → **X(23)**
  - For A(0): C = (5 × 0 + 8) mod 26 = 8 mod 26 = 8 → **I(8)**
  - For T(19): C = (5 × 19 + 8) mod 26 = 103 mod 26 = 25 → **Z(25)** (103 = 3 × 26 + 25)
  - For A(0): C = (5 × 0 + 8) mod 26 = 8 mod 26 = 8 → **I(8)**

  Transformation: D(3), A(0), T(19), A(0) → X(23), I(8), Z(25), I(8) → **XIZI**.
- **Multiplicative Inverse Derivation:** Find 5^{-1} mod 26.

  > `5 × 21 = 105 = 4 × 26 + 1 ≡ 1 (mod 26) ⇒ Inverse 5^-1 = 21 mod 26`
- **Decryption using P = 21(C − 8) mod 26:**
  - For X(23): P = 21 × (23 − 8) mod 26 = 21 × 15 mod 26 = 315 mod 26 = 3 → **D(3)** (315 = 12 × 26 + 3)
  - For I(8): P = 21 × (8 − 8) mod 26 = 21 × 0 mod 26 = 0 → **A(0)**
  - For Z(25): P = 21 × (25 − 8) mod 26 = 21 × 17 mod 26 = 357 mod 26 = 19 → **T(19)** (357 = 13 × 26 + 19)
  - For I(8): P = 21 × (8 − 8) mod 26 = 0 → **A(0)**

  Decryption P = 21(C - 8) mod 26 ⇒ "DATA" (Exact Match)

4. Transformation Trace & Cipher Evaluation Table

Plaintext	P	(5P + 8)	C = Mod 26	Ciphertext	(C − 8)	21(C − 8)	Mod 26	Decrypted
D	3	23	23	**X**	15	315	3	**D**
A	0	8	8	**I**	0	0	0	**A**
T	19	103	25	**Z**	17	357	19	**T**
A	0	8	8	**I**	0	0	0	**A**

Q19. Explain the AES Key Expansion (Key Schedule) procedure.

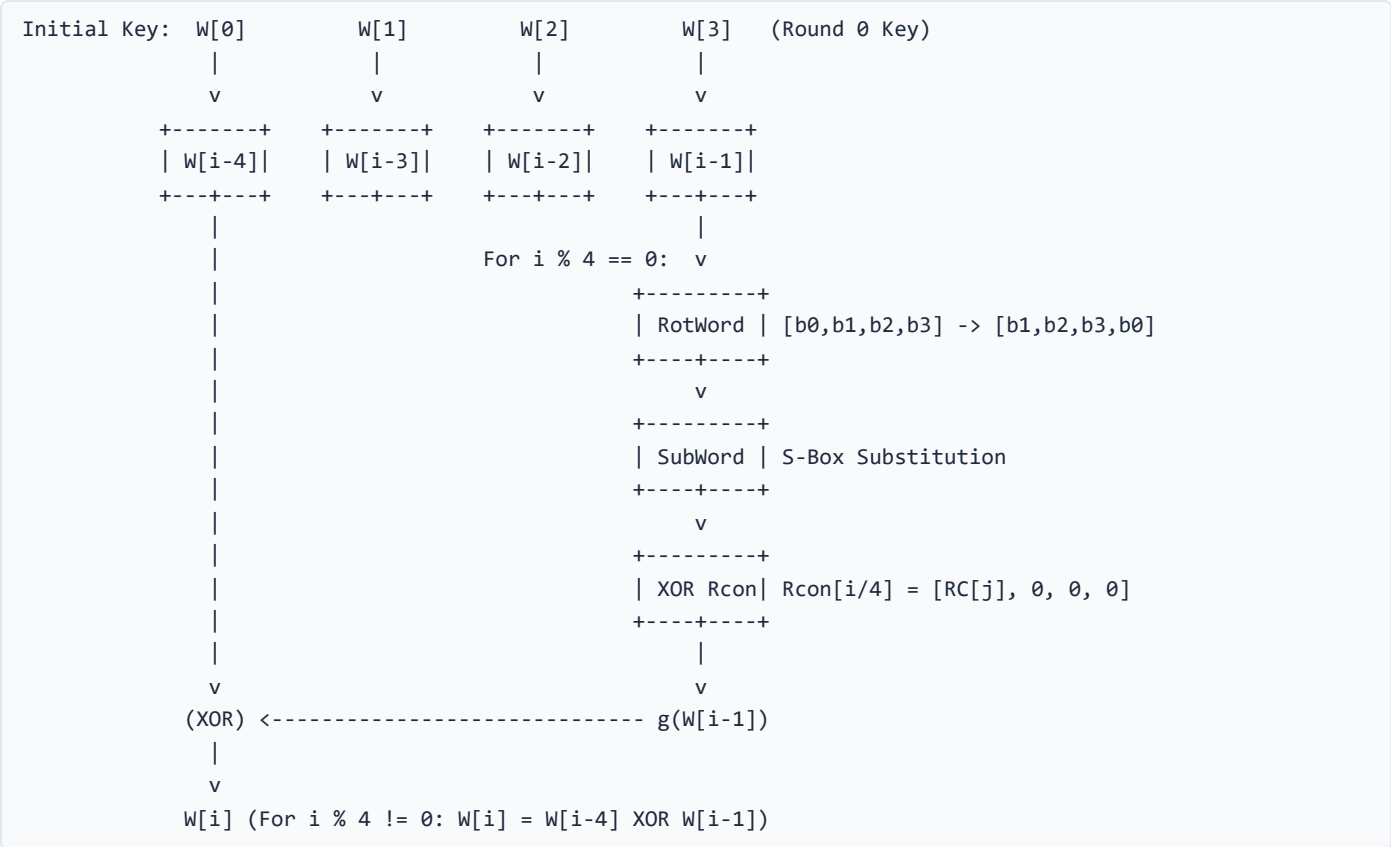
1. Core Concept and Architectural Principles

The **AES Key Expansion** (NIST FIPS 197) converts a master key of N_k 32-bit words (128, 192, or 256 bits) into an expanded array of 4(N_r + 1) 32-bit words denoted W[0 ... 4(N_r + 1) − 1].

- **Round Keys:** Each round consumes a 128-bit subkey (4 words). For AES-128 (N_k = 4, N_r = 10), the routine generates 44 words (W[0 ... 43]), supplying 1 pre-round key and 10 round keys.

- **Design Criteria:** Guarantees strict avalanche diffusion (inverting one key bit alters many subkey bits), introduces non-linearity via S-box bytes to thwart slide attacks, and enables fast bi-directional hardware evaluation.

2. AES Word Expansion Architecture



3. Step-by-Step Expansion Routine & Function g(W)

- **Initial Key Assignment:** The first N_k words ($W[0 \dots N_k-1]$) are copied directly from the master cipher key columns.
- **Linear Word XOR (i % N_k ≠ 0):** Intermediate words are generated by linear recurrence: $W[i] = W[i - N_k] \oplus W[i - 1]$.
- **Non-Linear Core Function g(W) (i % N_k == 0):** When i is a multiple of N_k, word $W[i - 1]$ is transformed by:
  1. **RotWord:** Cyclically rotates bytes left by 1 position: $[b_0, b_1, b_2, b_3] \rightarrow [b_1, b_2, b_3, b_0]$.
  2. **SubWord:** Maps each byte through the 8-bit non-linear AES S-Box using GF(2^8) multiplicative inversion.
  3. **Rcon XOR:** XORs the leading byte with Round Constant $RC[j]$, where $j = i/N_k$. Round constants are powers of x in GF(2^8): $RC[1..10] = \{01, 02, 04, 08, 10, 20, 40, 80, 1B, 36\}_{16}$.
- **AES-256 Extra Rule:** For $N_k = 8$ and $i \equiv 4 \pmod 8$, $W[i - 1]$ undergoes *SubWord* without rotation or Rcon XOR.

4. AES Key Schedule Parameters Comparison Table

AES Cipher Variant	Key Size (Bits / Words)	Rounds (N_r)	Expanded Words / Bytes	Words per Subkey	Rcon Consumed
AES-128	128 bits / 4 words	10 rounds	44 words / 176 bytes	4 words (128 bits)	10 constants (RC[1..10])
AES-192	192 bits / 6 words	12 rounds	52 words / 208 bytes	4 words (128 bits)	8 constants (RC[1..8])
AES-256	256 bits / 8 words	14 rounds	60 words / 240 bytes	4 words (128 bits)	7 constants (RC[1..7])

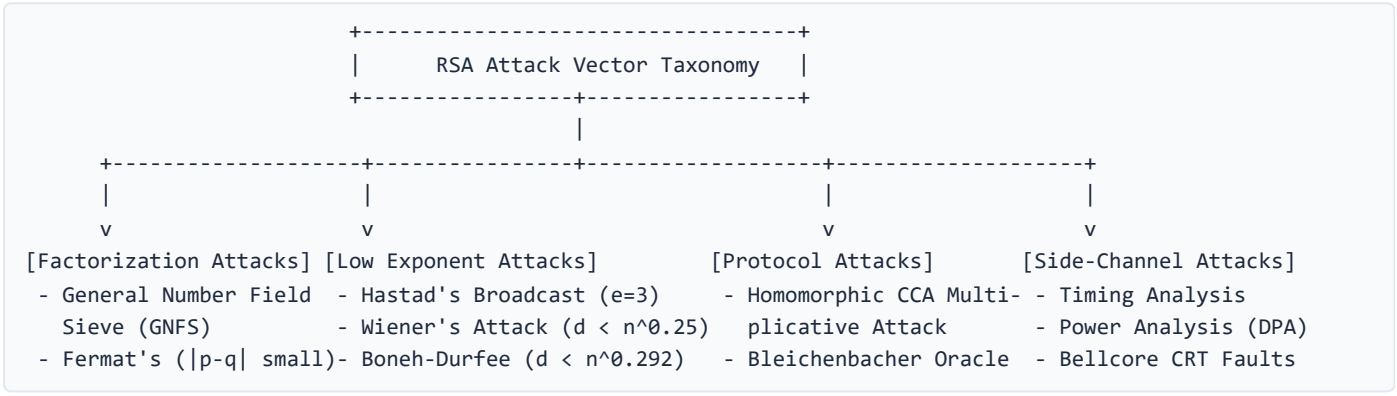
Q20. Explain the major types of attacks on the RSA cryptosystem.

1. Core Concept and Theoretical Vulnerability Surface

The **RSA cryptosystem** (Rivest-Shamir-Adleman) relies on the intractability of factoring large composite moduli $n = p \times q$ and computing modular e-th roots.

- **Theoretical Foundations:** Secure RSA requires large random primes, full-length private exponents, randomized padding, and constant-time execution.
- **Vulnerability Surface:** Failures in parameter selection, deterministic padding, or hardware leakage expose RSA to mathematical factoring, algebraic low-exponent attacks, protocol chosen-ciphertext attacks, and side-channel compromises.

2. RSA Attack Vector Taxonomy



3. Detailed Breakdown of Major Attack Classes

- **Mathematical Factorization Attacks:**
  - *General Number Field Sieve (GNFS):* The fastest general factoring algorithm, operating in sub-exponential time $O(\exp((c+o(1))(\ln n)^{1/3}(\ln \ln n)^{2/3}))$, necessitating ≥ 2048-bit moduli.
  - *Fermat's Factorization:* If prime factors are chosen too close together ($|p - q| < 2n^{1/4}$), n factors efficiently via difference of squares.
- **Small Exponent & Algebraic Attacks:**
  - *Hastad's Broadcast Attack (Small e):* When identical plaintext M is sent to 3 recipients with $e = 3$ and coprime moduli n_1, n_2, n_3, the Chinese Remainder Theorem solves for $M^3 \bmod (n_1 n_2 n_3)$. Because $M^3 < n_1 n_2 n_3$, taking the regular integer cube root over $\mathbb{R}$ recovers M directly.
  - *Wiener's Continued Fraction Attack (Small d):* If private exponent $d < (1/3)n^{1/4}$, continued fraction convergents of e/n reveal d in polynomial time (extended to $d < n^{0.292}$ by Boneh-Durfee).
- **Protocol & Chosen-Ciphertext Attacks:**
  - *Homomorphic Multiplicative Attack:* Because $E(m_1)E(m_2) \equiv (m_1 m_2)^e \pmod n$, an attacker blinds target C as $C' = C \times r^e \bmod n$ to extract $M = M' \times r^{-1} \bmod n$.
  - *Bleichenbacher Padding Oracle:* Targets PKCS#1 v1.5 error messages to iteratively decrypt ciphertexts via adaptive chosen queries.
- **Side-Channel & Hardware Fault Attacks:** Exploits timing discrepancies during square-and-multiply loops, or injects faults into RSA-CRT modular exponentiation to factor n via $p = \gcd(s' - s, n)$ (Bellcore attack).

4. Attack Mitigation & Countermeasure Matrix Table

Attack Class	Exploit Mechanism	Vulnerability Root Cause	Standard Countermeasure
Factorization	GNFS / Fermat	Modulus < 2048 bits; $\|p - q\|$ too small	2048/4096-bit keys; strong prime spacing
Small Public Exponent	Hastad's Broadcast / CRT	e = 3 across multiple recipients	Set e = 65537 ($2^{16} + 1$); OAEP padding
Small Private Exponent	Wiener's / Boneh-Durfee	$d < n^{0.292}$ chosen for speed	Enforce full-length private exponents ($d > n^{0.5}$)
Chosen-Ciphertext	Bleichenbacher / Malleability	Deterministic or PKCS#1 v1.5 padding	Enforce RSA-OAEP (RFC 8017)
Side-Channel / Fault	Timing / Bellcore CRT fault	Data-dependent runtime; unverified CRT	Cryptographic blinding; constant-time ALUs

---

Q21. Explain the main stages of the SHA-512 algorithm with a neat diagram.

1. Core Concept & Cryptographic Foundations

SHA-512 (NIST FIPS 180-4) is an iterative cryptographic hash function based on the Merkle-Damgård framework with a Davies-Meyer compression core.

- **Structural Parameters:** Operates on 1024-bit message blocks (sixteen 64-bit words) and generates a 512-bit (64-byte) digest across 80 round iterations over eight 64-bit working registers (*a, b, c, d, e, f, g, h*).
- **Security Strength:** Supports input messages up to $2^{128} - 1$ bits, providing 256 bits of collision resistance against birthday attacks.

2. Architectural Block Diagram & Compression Round

```
[Input Message M] ---> [Padding & Length Append] ---> 1024-bit Blocks M_1, M_2...
                                                          |
Initial Hash H0 (512-bit)                                 v
         |                                    +--------------------+
         +------------------------------------>| 80-Round Engine    |<-- Wt, Kt
                                              +---------+----------+
                                                        |
                                                        v
                                        Hash H1 = H0 + Regs (mod 2^64)
--------------------------------------------------------------------------------
Single Round Logic:
  T1 = h + Sigma1(e) + Ch(e, f, g) + Kt + Wt (mod 2^64)
  T2 = Sigma0(a) + Maj(a, b, c) (mod 2^64)
  New Regs: a = T1 + T2,  b = a,  c = b,  d = c,  e = d + T1,  f = e,  g = f,  h = g
```

3. Step-by-Step Operational Stages of SHA-512

- **Stage 1: Message Padding:** Append a '1' bit followed by k '0' bits such that the bit length satisfies $(L + 1 + k) \equiv 896 \pmod{1024}$.
- **Stage 2: Length Field Append:** Append a 128-bit unsigned integer representing original message length L, making the total length an exact multiple of 1024 bits.
- **Stage 3: Buffer Initialization:** Pre-load registers (H_0 ... H_7) with the fractional square root constants of the first 8 primes (2 to 19).
- **Stage 4: Message Schedule Expansion (W_0 to W_{79}):** Block words W_0 ... W_{15} are taken directly; W_{16} ... W_{79} are derived via:

> $$W_t = \sigma_1(W_{t-2}) + W_{t-7} + \sigma_0(W_{t-15}) + W_{t-16} \pmod{2^{64}}$$

- **Stage 5: 80-Round Compression Execution:** For t = 0 to 79, update state variables using logical functions:

  $$Ch(e,f,g) = (e \wedge f) \oplus (\neg e \wedge g), \quad Maj(a,b,c) = (a \wedge b) \oplus (a \wedge c) \oplus (b \wedge c)$$
  $$T_1 = h + \Sigma_1(e) + Ch(e,f,g) + K_t + W_t, \quad T_2 = \Sigma_0(a) + Maj(a,b,c)$$

  Update: $h=g$, $g=f$, $f=e$, $e=d+T_1$, $d=c$, $c=b$, $b=a$, $a=T_1+T_2$.

- **Stage 6: Intermediate Accumulation & Output:** Add register outputs to previous hash vectors: $H_i^{(j)} = H_{i-1}^{(j)} + \text{Reg}^{(j)} \mod 2^{64}$. Concatenating $H_N^{(0..7)}$ gives the 512-bit digest.

4. SHA-512 Operational Specification Table

Parameter	Specification Value	Cryptographic Role
Input Message Size	$< 2^{128}$ bits	Arbitrary length payload capacity
Block Size	1024 bits (16 × 64-bit words)	Iterative compression processing unit
Word Size	64 bits	Native 64-bit register arithmetic operations
Rounds per Block	80 rounds	Diffusion and confusion iterations
Output Digest Size	512 bits (64 bytes)	256-bit collision security bound

---

Q22. Explain the ElGamal Digital Signature Scheme with signing and verification procedures.

1. Core Concept & Mathematical Foundations

The **ElGamal Digital Signature Scheme** (1985) is an asymmetric, probabilistic digital signature algorithm based on the Discrete Logarithm Problem (DLP) over $\mathbb{Z}_p^*$.

- **Probabilistic Signatures:** Unlike RSA, signing the same message twice generates distinct signature pairs (r, s) due to a fresh random secret nonce k chosen per signing execution.
- **Security Hardness:** Forging signatures requires computing discrete logarithms (finding private key x from public key $y \equiv g^x \mod p$), which is computationally infeasible for large primes.

2. Architectural Protocol Flow Diagram

```
KEY GENERATION:
  Select prime p, primitive root g mod p, private key x in [2, p-2]
  Compute public key y = g^x mod p  ---> Public: (p, g, y), Private: x
                  |
SIGNING (Alice): |                        VERIFICATION (Bob):
  Input: Hash m = H(M), Private Key x       Input: m = H(M), Signature (r, s), (p, g, y)
  1. Pick random secret nonce k:            1. Verify bounds:
     gcd(k, p-1) = 1                           0 < r < p  AND  0 < s < p-1
  2. Compute r = g^k mod p                  2. Compute test values:
  3. Compute k^-1 mod (p-1)                    v1 = g^m mod p
  4. Compute signature s:                      v2 = (y^r * r^s) mod p
     s = k^-1 * (m - x*r) mod (p-1)         3. Accept if v1 ≡ v2 (mod p),
  Output Signature: (r, s)                     Else Reject
```

3. Detailed Algorithmic Procedures & Correctness Proof

- **Phase 1: Key Generation:** Choose large prime p and primitive root g mod p. Pick private key $x \in [2, p-2]$. Compute public verification key $y = g^x \mod p$.

- **Phase 2: Signature Generation:**
  1. Compute message digest $m = H(M)$, where $0 \leq m \leq p - 2$.
  2. Select random ephemeral key $k \in \mathbb{Z}_{p-1}^*$ with $\gcd(k, p - 1) = 1$.
  3. Compute $r = g^k \bmod p$.
  4. Compute $s = k^{-1} \times (m - x \times r) \bmod (p - 1)$. If $s = 0$, select a fresh k.
  5. Output signature pair (r, s). *Security Warning:* Nonce k must never be reused; reuse exposes x via linear congruence subtraction.
- **Phase 3: Signature Verification:** Verify that $0 < r < p$ and $0 < s < p - 1$. Compute $v_1 = g^m \bmod p$ and $v_2 = (y^r \times r^s) \bmod p$. Accept if and only if $v_1 \equiv v_2 \pmod p$.
- **Mathematical Proof of Correctness:**

  $$k \times s \equiv m - x \times r \bmod (p - 1) \Rightarrow m \equiv x \times r + k \times s \bmod (p - 1)$$
  $$g^m \equiv g^{x \times r + k \times s} \equiv (g^x)^r (g^k)^s \equiv y^r \times r^s \pmod p$$

  Thus $v_1 \equiv v_2 \pmod p$, confirming validity.

4. Comparative Evaluation: ElGamal vs. RSA Digital Signatures

Metric	ElGamal Digital Signature	RSA Digital Signature
Hardness Problem	Discrete Logarithm Problem (DLP) over $\mathbb{Z}_p^*$	Integer Factorization Problem ($n = p \times q$)
Signature Output	Pair (r, s), 2 × modulus size	Single integer $s = m^d \bmod n$, 1 × modulus size
Nature	Probabilistic (randomized via nonce k)	Deterministic (requires randomized PSS padding)
Nonce Vulnerability	Catastrophic: nonce reuse reveals private key x	Not applicable (no per-message nonce used)
Successor Standard	NIST DSA and ECDSA (FIPS 186)	RSA-PSS (RFC 8017)

Q23. Explain the Authentication Header (AH) and Encapsulating Security Payload (ESP) protocols in IPSec.

1. Core Concept & IPSec Architecture

Internet Protocol Security (IPSec) (RFC 4301) operates at the Network Layer (OSI Layer 3) to deliver transparent security services for all IPv4 and IPv6 upper-layer protocols.

- **Authentication Header (AH - RFC 4302):** Provides connectionless integrity, data origin authentication, and anti-replay protection. AH provides *zero confidentiality* (payload remains plaintext).
- **Encapsulating Security Payload (ESP - RFC 4303):** Provides confidentiality (symmetric encryption), connectionless integrity, data origin authentication, and anti-replay.
- **Operational Modes:** *Transport Mode* secures upper-layer payloads while retaining original IP headers; *Tunnel Mode* encapsulates entire IP datagrams within new outer IP headers for VPN gateways.

2. Protocol Header Formats & Packet Structures

```
AH Header Format (RFC 4302 - IP Protocol 51):
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Next Header(8)| Payload Len(8)|           RESERVED(16)        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Security Parameters Index (SPI) (32)          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Sequence Number (Anti-Replay) (32)            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Integrity Check Value (ICV) (Variable length)       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

ESP Packet Format (RFC 4303 - IP Protocol 50):
[ESP Header:  SPI (32) | Sequence Number (32)]           (Plaintext)
[Payload Data (TCP + App Data)]               --- ENCRYPTED
[Padding (0-255B) | Pad Len (8) | Next Hdr (8)] --- SCOPE (ESP Trailer)
[Integrity Check Value (ICV)]                 --- AUTHENTICATED SCOPE (Header to Trailer)
```

3. Detailed Technical Analysis: AH vs. ESP Mechanisms

- **AH Operation & Scope:** Authenticates the entire IP payload and outer IP header. Intermediate routers mutate certain header fields (TTL, Checksum, DSCP/TOS, Flags); AH zeroes out these mutable fields prior to HMAC calculation. However, NAT alters immutable IP addresses, causing ICV validation to fail; thus AH is **strictly incompatible with NAT**.
- **ESP Operation & Scope:** Encrypts the payload and ESP trailer using AES-GCM or AES-CBC. Authenticates the ESP header, ciphertext, and trailer. Crucially, ESP does *not* protect the outer IP header, making ESP **fully compatible with NAT traversal** (NAT-T via UDP port 4500).
- **Anti-Replay Protection:** Both protocols employ a monotonically increasing 32-bit (or 64-bit Extended) Sequence Number and a sliding verification window (typically 64 packets) to silently drop duplicate packets.

4. Comprehensive Technical Comparison Table: AH vs. ESP

Feature	Authentication Header (AH)	Encapsulating Security Payload (ESP)
RFC Standard	RFC 4302	RFC 4303
IP Protocol Number	51	50
Confidentiality (Encryption)	No (Plaintext payload)	Yes (AES-GCM, AES-CBC, ChaCha20)
Data Origin Authentication	Yes (HMAC over payload + immutable IP header)	Yes (HMAC over ESP Header, Payload, Trailer)
Anti-Replay Protection	Yes (Sliding window sequence numbers)	Yes (Sliding window sequence numbers)
Outer IP Header Protection	Yes (Authenticates immutable fields)	No (Outer IP header unprotected)
NAT Traversal	Incompatible (NAT breaks ICV)	Compatible (NAT-T over UDP 4500)
Primary Use Case	Legacy router integrity checks	Modern IPsec VPNs, Remote Access

---