①

Question 1: Shor's Algorithm Threat to RSA and ECC

• Threat: Shor's algorithm efficiently factors large elliptic curve discrete logarithm problem ~~(breakin~~ (breaking ECC) on quantum computers.

Consequences:

-> RSA and ECC-based encryption, key exchange, and digital signatures become insecure.

-> critical systems (e.g., banking, TLS, crypto-currencies) relying on these algorithm would need post-quantum replacements (e.g., lattice-based crytography).

Question 2: Quantum Key Distribution (QKD)

• Role: QKD uses quantum mechanics (e.g., Photon Polarization) to securely distribute keys, Providing unconditional security against eavesdropping (detected via quantum state disturbances)

• Difference from classical Encryption.

-> Classical PKI relies on Physical laws (e.g., no-cloning theorem).

②

Question 3: Lattice-Based vs Traditional Encryption:

- Lattice-Based: uses hard problem like Learning With Errors (LWE) or Shortest Vector Problem (SVP). Resistant to quantum attacks.

- Traditional (RSA/ECC): Relies on factoring or discrete logarithms, Vulnerable to shor's algorithm.

- Advantage: Lattice-based schemes offer quantum resistance and simpler operations. (e.g., matrix multiplications).

Question 4: Python PRNG with System Time and Seed.

Python

```
import time
def custom_prng(seed):
    current_time = int(time.time())
    combined = (current_time * seed) % 2**32
    return combined
```

③

seed = 42
random_number = custom_Prng(seed)
Print(f"Generated number: {random_number}")

Output:
text
Generated number: 123456789 # Example output (varies with time)

Question 5: Sive of Eratosthenees

- Algorithm: Iteratively marks multiples of Primes starting from 2.
- Primes < 50:
  2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47
- Time complexity: O(nlog fo log fo n) O(nloglogn) (faster than trial division's O(nn) O(nn)).

4.1

Question6: Carmichael Numbers

- Conditions: A composite nn is carmichael if:

1. nn is square-free.

2. For all a coprime to nn, $a^{n-1} \equiv 1 \bmod n$ $a^{n-1} \equiv$ mod n.

- Verification:

-> 561: Meet conditions (smallest carmichael number).

-> 1105: 1105 = 5×13×17 1105 = 5×13×17; Passes tests.

-> 1729: 1729 = 7×13×19 1729 = 7×13×19; Passes tests.

Question7: Algebraic structures

1. Z11Z11 with (+,·)(+,·):

   -> yes, a ring (modular arithmetic satisfies ring axioms).

2. (Z37,+)(Z37,+):

   - yes, Abelian group (inverses exist, commutative).

3. (Z35,×)(Z35,×):

-> NO, not a group (element without inverses e.g., 5).

Question 8: Remainder of $-52 \bmod 31$ $-52 \bmod 31$

- Calculation:

$-52 \bmod 31 = (-52+2\times31) \bmod 31 = 10 \bmod 31$ $-52 \bmod 31 = (-52+2\times31) \bmod 31 = 10 \bmod 31$.

Answer: 15.

Question 9: Multiplicative Inverse of 7 mod 26

- Extended Euclidean steps:

$26 = 3\times7+5$ $26 = 3\times7+5$

$7 = 1\times5+2$ $7 = 1\times5+2$

$5 = 2\times2+1$ $5 = 2\times2+1$

Back-substitute: $1 = 5-2\times2 = 5-2\times(7-1\times5) = 3\times5-2\times7$ $1 = 5-2\times2 = 5-2\times(7-1\times5) = 3\times5-2\times7$.

$1 = 3\times(26-3\times7)-2\times7 = 3\times26-11\times7$ $1 = 3\times(26-3\times7)-2\times7 = 3\times26-11\times7$.

Inverse: $-11 \bmod 26 = 15$ $-11 \bmod 26 = 15$.

Answer: 15

Question 10: Negative Modular Multiplication

: Calculation:

$(-8\times5) \bmod 17 = (-40) \bmod 17 = 40+3\times17 =$

$11\ (-8\times5) \bmod 7 = (-40) \bmod 17 = -40+3\times17 = 11.$

· Simplification Rule: convert negatives to positives by adding multiples of the modulus.

Question 11: Bézout's Theorem

· Theorem: For integers a, ba, b, there exist x, yx, y such that $ax+by = \gcd(a,b)\ ax+by = \gcd(a,b)$.

: Proof: uses Euclidean algorithm back-substitution.

: Inverse of 97 mod 385:

$\gcd(97, 385) = 1$, so inverse exists.

Solution: $x = 317\ x = 317$ (or 68 mod 385 68 mod 385).

Question 12: Bezout's Identity and solution

· Proof: Follows from Euclidean algorithm.

· Solve $43x \equiv 1 \mod 240 43x \equiv 1 \mod 240$

gcd fo 43, 240) = 1 gcd (43; 240) = 1.

so $x = 67$ $x = 67$ (since $43 \times 67 = 2881 \equiv 1 \mod 240 43 \times 67 = 2881 \equiv 1 \mod 240$).

Question 13: Fermat's Little Theorem

· Theorem: If p p is prime, $ap = 1 \equiv \mod p$ $a^{p-1} \equiv 1 \mod p$.

· Primality test for 561:

Test 2560 mod 5612560 mod 561. Fails (561 is carmichael, not prime).

· compute 5123 mod 1755123 mod 175:

$175 = 25 \times 7 175 = 25 \times 7$. Use CRT:

5123 mod 25 = 05123 mod 25 = 0.

5123 mod 7 = 5 (123 mod 6) mod 7 = 53 mod 7 = 65123 mod 7 = 5(123 mod 6) mod 7 = 53 mod 7 = 6.

Solve $x \equiv 0 \bmod 25x \equiv 0 \bmod 25$, $x \equiv 6 \bmod 7x \equiv 6 \bmod 7$:

$x = 25kx = 25k \equiv 6 \bmod 7 \Rightarrow k \equiv 4 \bmod 7\ 25k \equiv 6 \bmod 7 \Rightarrow k \equiv 4 \bmod 7$.

$x = 25 \times 4 = 100x = 25 \times 4 = 100$.

Answer: 100.

Question: 14: Chinese Remainder The rem (CRT).

· CRT statement: If moduli are coprime, a unique solution exists modulo their product.

· Solution:

1. Solve $x \equiv 2 \bmod 3x \equiv 2 \bmod 3$, $x \equiv 3 \bmod 5x \equiv 3 \bmod 5$:

$x = 5k + 3x = 5k + 3$. Substitute into first: $5k + 3 \equiv 2 \bmod 3 \Rightarrow k \equiv 1 \bmod 3\ 5k + 3 \equiv 2 \bmod 3 \Rightarrow k \equiv 1 \bmod 3$.

$k = 3m + 1k = 3m + 1$, so $x = 15 + 8x = 15m + 8$.

2. Solve $x \equiv 8 \bmod 15 x \equiv 8 \bmod 15$ and $x \equiv 2 \bmod 7 x \equiv 2 \bmod 7$:

· $x = 15n + 8 x = 15n + 8$.

Substitute: $15n + 8 \equiv 2 \bmod 7 \Rightarrow n \equiv 1 \bmod 7 15n + 8 \equiv 2 \bmod 7 \Rightarrow n \equiv 1 \bmod 7$.

Final Answer: $x \equiv 23 \bmod 105 x \equiv 23 \bmod 105$.

Question 15: CIA Triad

- Confidentiality: Encryption (e.g., AES).
- Integrity: Hashing/MACs (e.g., SHA-256).
- Availability: Redundancy/DoS Protection.

Question 16: Steganography vs. Crytography.

- Steganography: Hides data existence (e.g., in images, LSB manipulation).
- Crytography: Scrambles data (e.g., AES).
- Techniques: LSB, frequency domain masking (DCT).

Question 17: Phishing, Malware, DoS

- Phishing: Deception (e.g., fake emails).
- Malware: Software harm (e.g., ransomware).
- DoS: Overloads resources (e.g., SYN floods).

Question 18: GDPR and Cyber Attacks

- Role: Mandates data Protection (e.g., encryption, breach reporting), reducing attack impact via accountability.

Question 19: DES Algorithm Overview

1. Initial Permutation (IP): Rearranges 64-bit Plaintext.
2. 16 Rounds: Feistel network with XOR

S-boxes and Permutations.

3. Final Permutation (FP): Inverse of IP.

Example: Plaintext 0x0123456789ABCDEF.

key 0x133457799BBCDFF1.

Question 20: DES Round Function

: Given:

R0=0x FOFOFOFORO = 0xFOFOFOFO, K1=

0x OFOFOFOFOF

=0x0FOFOFOFOF, L0 = 0x AAAAAAAAL0=0x

AAAAAAAA.

· Steps:

1. f (R0, K1) = R0⊕K1 = 0x FFFFFFFF f(R0,

K1)=R0⊕K1=0x FFFFFFFF.

2. $L1 = R0 = 0xF0F0F0F0 L1 = R0 = 0xF0F0F0F0$.

3. $R1 = L0 \oplus f(R0, K1) = 0xAAAAAAAA \oplus 0xFFFFFFFF = 0x55555555$ $R1 = L0 \oplus f(R0, K1) = 0xAAAAAAAA \oplus 0xFFFFFFFF = 0x55555555$

Question 21: AES SubBytes

- Input: [0x23, 0xA7, 0x4C, 0x19][0x23, 0xA7, 0x4C, 0x19].
- S-box Lookup:

-> 0x23 -> 0x23 -> Row 2, Col 3 -> -> D4 (from table).

-> 0xA7 -> 0xA7 -> Row A, Col 7 -> -> 0x63.

-> 0x4C -> 0x4C -> Not in table (hypothetical: assume 0x4C maps to 0x2E).

-> 0x19 → 0x19 → Not in table (hypothetical: assume 0x19 maps to 0xC6).

- Output: [0xD4, 0x63, 0x2E, 0xC6][0xD4, 0x63, 0x2E, 0xC6].

Question 22: AES AddRoundKey

- Input: [0x14, 0x2B, 0x3C, 0x4D][0x14, 0x2B, 0x3C, 0x4D]
- Round Key: [0x55, 0x66, 0x77, 0x88][0x55, 0x66, 0x77, 0x88]
- XOR Result:

$0x14 \oplus 0x55 = 0x41$ $0x14 \oplus 0x55 = 0x41$

$0x2B \oplus 0x66 = 0x4D$ $0x2B \oplus 0x66 = 0x4D$

$0x3C \oplus 0x77 = 0x4B$ $0x3C \oplus 0x77 = 0x4B$

$0x4D \oplus 0x88 = 0xC5$ $0x4D \oplus 0x88 = 0xC5$

- Output: [0x41, 0x4D, 0x 4B, 0x C5] [0x 41, 0x 4D, 0x 4B, 0x C5].

Question 23: AES Mix Columns

- Input column: [0x01, 0x 03, 0x 04] [0x 01, 0x 02, 0x 03, 0x 04].

- Matrix Multiplication (GF (2^8)):
- 0x01 x 02 = 0x 02 0x 01 x 02 = 0x 02, 0x 02 x 03 = 0x 06 0x 02 x 03 = 0x 06, etc.
- Sum: 0x 02 ⊕ 0x 06 ⊕ 0x 03 ⊕ 0x 04 = 0x 07 0x 02 ⊕ 0x 06 ⊕ 0x 03 ⊕ 0x 04 = 0x 07.

- Output column: [0x 07, ...] [0x 07, ...] (complete similarly for other rows).

Question 24: AES - OFB Mode

- working: Encrypts an IV to produce a

keystream, XORed with Plaintext. Self-Synchronizing (errors in ciphertext affect only corresponding Plaintext bits).

- Synchronization: IV must match; keystream regenerates identically.

Question 25: Error Propagation in AES Modes

- CBC: Errors Propagate to subsequent blocks (corrupts entire block + next block's same bit).
- CFB: Affects current and next block's same bit position.
- ECB: No Propagation (independent blocks).

Question 26: AES Mode for Large Files

• Recommendation: CTR mode.

• Why: Parallelizable (nonce + counter), no error propagation, efficient for bulk data.

Question 27: RSA Encryption / Decryption

• Encrypt $M=1M=1$:

$C = Me \bmod n = 15 \bmod 14 = 1C = Me \bmod n = 15 \bmod 14 = 1$.

• Decrypt $C=1C=1$:

$M = Cd \bmod n = 111 \bmod 14 = 1M = Cd \bmod n = 111 \bmod 14 = 1$.

Question 28: RSA Digital Signature

• Given: $H(M)=5H(M)=5$, $d=3d=3$, $n=33n=33$.

• Signature: $S = H(M)^d \bmod n = 5^3 \bmod 33 = 125 \bmod 33 = 26$ $S = H(M)^d \bmod n = 5^3 \bmod 33 = 125 \bmod 33 = 26$.

Question 29: Diffie-Hellmen key Exchange

• Aleya's Public key: $A = g^a \bmod p = 3^4 \bmod 17 = 81 \bmod 17 = 13$ $A = g^a \bmod 17 = 81 \bmod 17 = 13$.

• Badol's Public key: $B = g^b \bmod p = 3^5 \bmod 17 = 243 \bmod 17 = 5$ $B = g^b \bmod p = 3^5 \bmod 17 = 243 \bmod 17 = 5$.

Question 30: Simple Hash Function

• Hash of "AB":

ASCII of 'A' = 65, 'B' = 66

Sum $= 66 + 65 = 131$

$H("BA") = 131 \mod 100 = 31$ $H("BA") = 131 \mod 100 = 31$

• Collision: Both messages produce the same hash (31), demonstrating weak collision resistence in this hash function.

Question 31: Simple MAC

• MAC Calculation:

$MAC = (15+7) \mod 17 = 22 \mod 17 = 5$ $MAC = (15+7) \mod 17 = 22 \mod 17 = 5$

• Forgery Attempt:

Attacker changes message to 10 but doesn't know the key.

without the key, they cannot compute the correct MAC (which would be $(10+7) \bmod 17 = 0$ $(10+7) \bmod 17 = 0$).

Conclusion: Forgery is not possible without the key.

Question 32: TLS Hand Shake Process

1. ClientHello: Client sends supported cipher and a random number.

2. ServerHello: Server chooses cipher suite, sends its certificate, and a random number.

3. Key Exchange: Client verifies the certificate, generates a pre-master secret, encrypts it with the servers

Public key and sends it.

4. Session keys: Both sides derive symmetric keys from the pre-master secret and random numbers.

5. Finished: Encrypted "Finished" messages confirm handshake success.

Symmetric key Establishment: Asymetric cryptography (e.g., RSA or ECDH) securely exchanges the pre-master secret, which is used to derive symmetric keys.

Question 33: SSH Protocal stack

1. Transport Layer: Handles encryption, Integrity and host authentication (e.g., Diffie-Hellmen key exchange).

2. Authentication Layer: Manages user authentication (e.g., password, public key).

3. Connection Layer: Multiplexes channels (e.g., shells, file transfers).

Question 34: TLS Handshake (Repeated)

Same as Question 32.

Question 35: Elliptic curve Equation

• General Form:

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

Use in cryptography: Provides hard problems (e.g., ECDLP) for secure key exchange with smaller key sizes than RSA.

Question 36: ECC vs. RSA

. Smaller key Sizes: ECC relies on the hardness of the Elliptic curve Discrete Logarithm Problem (ECDLP), which is computationally harder than factoring (RSA). A 256-bit ECC key ≈ 3072-bit RSA key in security.

Question 37: Point on Elliptic curve ?

· Check : substitute $P=(3,6)P=(3,6)$ into $y^2 \equiv x^3+2x+3 \bmod 97y^2 \equiv x^3+2x+3 \bmod 97$:

$6^2 = 36 \bmod 976^2 = 36 \bmod 97$

$3^3+2(3)+3 = 27+6+3 = 36 \bmod 973^3+2(3)+3 = 27+6+3 = 36 \bmod 97$.

Result: $36 \equiv 3636 = 36$, So PP lies on the curve.

Question 38: ElGamal Ciphertext

- Given: $p=23$, $g=5$, $h=8$, $m=10$, $k=6$ $p=23$, $g=5$; $h=8$; $m=10$, $k=6$.

- steps:

1. $c1=g^k \bmod p=5^6 \bmod 23=15625 \bmod 23=8$ $c1=g^k \bmod p=5^6 \bmod 23=15625 \bmod 23=8$.

2. $c^2=m \cdot h^k \bmod p=10 \cdot 8^6 \bmod 23$ $c^2=m \cdot h^k \bmod p=10 \cdot 8^6 \bmod 23$.

$8^6 \bmod 23=262144 \bmod 23=12$ $8^6 \bmod 23=262144 \bmod 23=12$.

$c^2=10 \cdot 12 \bmod 23=120 \bmod 23=5$ $c^2=10 \cdot 12 \bmod 23=120 \bmod 23=5$.

- ciphertext: $(c1, c2)=(8,5)$ $(c1, c2)=(8,5)$.

Question 39: Lightweight crytography for IoT

• Importance: IoT devices have limited resources (CPU, memory, Power). Lighweight algorithms minimize overhead while maintaining security.

• Example: Cha Cha20 (stream cipher) or PRESENT (block cipher).

Question 40: Common IoT-specific Attacks and Mitigations

1. Firmware Hijacking

• Explanation: Attackers exploit vulnerabilities in IoT device firmware to inject malicious code (e.g., via unsecured OTA updates).

- Mitigations:
  - Code signing: Require cryptographically signd firmware updates.
  - Secure Boot: Ensure only trusted firmware executes.
  - Regular Patching: Maintain firmware updates from vendors.

2. Physical Tampering

- Explanation: Attackers gain physical access to extract data, modify hardware, or bypass security (e.g., UART Pin access, JTAG debugging).

- Mitigation:

-> Tamper-Proof Enclosures : Use anti-tampering hardware seals.

-> Secure Storage: Encrypt sensitive data (e.g. TPM/HSM modules).

-> Zeroize on Tamper: Wipe memory if tampering is detected.

3. Botnets (e.g., Mirai)

· Explanation: compromised IoT devices are enlisted in DDos botnets via default credentials or exploits.

· Mitigations:

-> Change Default credetials : Enforce strong unique passwords.

-> Network segmentation: Isolate IoT devices from critical networks.

Traffic Monitoring: Detect abnormal traffic (e.g., sudden spikes in outbound requests).