① With the ~~event~~ advent of quantum computing, ~~traditional~~ traditional public-key cryptosystems such as RSA and ECC are potentially vulnerable to Shor's algorithm. Discuss the implications of quantum computing on the security of cryptographic protocols, and purpose possible post-quantum cryptographic ~~protocols,~~ algorithms that could replace RSA and ECC. How do these algorithms resist quantum cryptanalysis?

Answer:

Quantum Computing and Post-Quantum Cryptography (PQC)

The Implications Classical Public-key systems (RSA and ECC) rely on the computational difficulty of specific mechanical problems: Integer Factorization (RSA) and the discrete Logarithm problem (ECC).

* Shor's Algorithm: This quantum algorithm can solve both problems in polynomial time. A ~~sufficient~~ sufficiently powerful quantum computer (CRQC) could derive a private key from a public key effectively ~~instantaneous~~, instantaneously, decrypting all past and future data.

* Harvest Now, Decrypt Later (HNDL): Adversaries are currently intercepting encrypted data to store it until quantum computers are available to decrypt it.

Proposed Post-Quantum Algorithm NIST has recently ~~the~~ standardized the following algorithms to replace RSA / ECC. These run on classical computers but rely on math that quantum computers cannot solve efficiently.

Algorithm	~~Type~~ Type	Use Case	Math Family
ML-KEM (Kyber)	Key Encryption	General Encryption	Lattice-based
ML-DSA (Dilithium)	Digital Signature	Autentication	Lattice-based
SLH-DSA (SPHINCS+)	Digital Signature	Backup Signature	Hash-based

Resistance Mechanism:

⊛ Latic-Based (Kyber / Dilithium): These rely on the ~~Short~~ Vector Problem (SVP) in high dimensional lattices. Imagine a grid in 500 dimensions with "noise" added to a point. Finding the nearest grid point is geometrically chaotic. Unlike factoring, this problem has no hidden "periodicity" for Shor's algorithm to exploit.

⊛ Hash-Based (SPHINCS+): These rely on the security of hash functions (like SHA-3). While Grover's ~~algorithm~~ algorithm provides a slight speedup against hashes,

simply increasing the output size (e.g. to 256 or 512 bits) negates the threat entirely.

(2) Design and implement a novel Pseudo-Random Number Generator (PRNG) algorithm in Python using the current timestamp, The process ID (os.pid) for added randomness, a module operation to constrain the output within a desired range.

Ans:

This implementation uses a Linear Congruential Generator (LCG) structure, seeded by a combination of the system time and the process ID to ensure unique starting states across different run instances.

code in python:

```
import time
import os

class SystemEntropyPRNG:
    def __init__(self, seed=None):
        """Initialization of PRNG state.
        If no seed is provided, generates one using current
                                        time and PID.
        """
        if seed is None:
            # combine current time (ns) and process ID using bitwise XOR
            # to create a high-entropy initial seed.
            current_time = int(time.time() * 1_000_000)
```

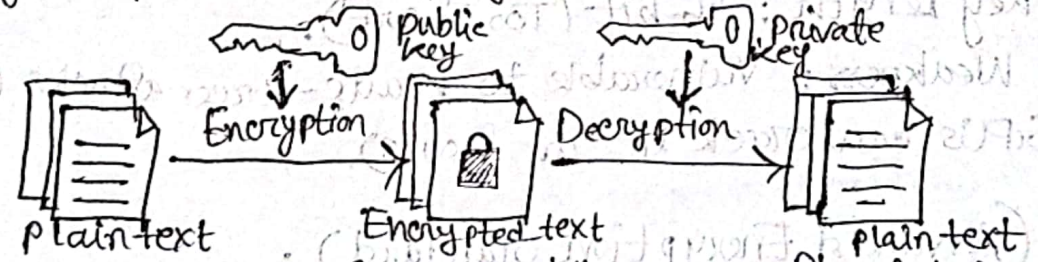
```
            pid = os.getpid()
            self.state = current_time ^ (pid << 16)
        else:
            self.state = seed
    #constant for LCG (Linear Congruential Generator)
    #values from standard implementations (e.g. glibc)
    for good distribution

    self.a = 1103515245
    self.c = 12345
    self.m = 2 ** 31
    def next_int (self, min_val, max_val):
        """generates the next pseudo-random number
           within [min_val, max_val]."""
        if min_val > max_val:
            raise ValueError("min_val must be less than
                              or equal to max_val")
        # Update state: X_{n+1} = (a * X_n + c) mod m
        self.state.
        self.state = (self.a * self.state + self.c) %
                                            self.m
```

```
#constrain output to desired range using modulus
range_width = max_val + 1
return min_val + (self.state % range_width)

#Usage Example ---
prng = SystemEntropyPRNG()
print(f"Generator Seeded with PID: {os.getpid()}")
print("Random Sequence (1-100):")
for _ in range(5):
    print(prng.next_int(1,100))
```

Q2 Compare the ~~traditional etc~~ ciphers (such as the Caesar cipher, vigenere cipher and playfair cipher) with modern symmetric ciphers ~~as~~ like AES and DES. Discuss the strength and weakness of each cipher, including key ~~strength~~, length, encryption/decryption speed and security against modern cryptanalysis techniques.



Traditional Ciphers (Caesar, Vigenere, Playfair)

* Mechanism: Substitution (replacing letters) and Transposition (rearranging).
* Strengths: Simple to understand; can be performed manually with pen and paper.
* Weaknesses:
  - Key space: Extremely small (Caesar has only 25 possible keys).

• Cryptanalysis : Vulnerable to frequency Analysis. In English, 'E' is the most common letter; if 'X' appears most often in the ciphertext 'X' is likely 'E'.

• Structure : They preserve the statistical structure of the language.

Modern Symmetric Ciphers (DES, AES) :

⊛ Mechanism: Repeated rounds of substitution-permutation Networks (SPN) operating on bits rather than letters, creating high "Confusion" and "Diffusion".

DES (Data Encryption Standard):

- Status : obsolete.
- Key length : 56-bit (Too short).
- Weakness : Vulnerable to brute-force attacks (modern GPUs can crack it in hours).

AES (Advanced Encryption Standard) :

- Status : The global standard.
- Key Length : 128, 192, or 256 bits.
- Strength : Resistant to all known attacks (Linear and Differential Cryptanalysis). A 128-bit key takes billions of years to ~~brut~~ brute force.

(Q4) Let S_4 be the symmetric group on the set $\{1,2,3,4\}$. Define an action of S_4 on the set of 2-element subsets of $\{1,2,3,4\}$.
Prove that this action is well-defined, and compute the size of the orbit and the stabilizer of the subset $\{1,2\}$.

Ans:

Group Action of S_4:

Action definition

Let S_4 be the symmetric group of permutations on $\{1,2,3,4\}$.
Let X be the set of 2-element subsets of $\{1,2,3,4\}$.
$X = \{\{1,2\},\{1,3\},\{1,4\},\{2,3\},\{2,4\},\{3,4\}\}$.

The action is defined as $\sigma \cdot \{a,b\} = \{\sigma(a), \sigma(b)\}$.

1. Proof of Well-definedness

For the action to be well-defined, the result must be a set in X.
Since σ is a bijection (permutation), if $a \neq b$, then $\sigma(a) \neq \sigma(b)$. Thus, the image $\{\sigma(a), \sigma(b)\}$ is a set of exactly two distinct elements from $\{1,2,3,4\}$.
Therefore, $\sigma \cdot \{a,b\} \in X$.

2. The Orbit of $\{1,2\}$

The orbit is the set of all subsets that $\{1,2\}$ can be transformed into by permutations in S_4.

Since S_4 is a k-transitive for all $k \leq n$, it can map any pair of distinct elements to any other pair of distinct elements.

Therefore, the orbit is the entire set X.
size of orbit = 6

3. The stabilizer of {1,2}

The stabilizer is the set of permutation $\sigma \in S_4$ such that $\{\sigma(1), \sigma(2)\} = \{1,2\}$.

The elements inside the set {1,2} can be permuted ~~along~~ among themselves, and the elements outside the set {3,4} can be permuted among themselves.

⊛ Internal Permutations (on {1,2}): Identity or swap (1 2). (2 options).

⊛ External Permutations (on {3,4}): Identity or swap(3 4). (2 options).

The Stabilizer consists of combinations of these swaps:

1. e (Identity).
2. (1 2)
3. (3 4)
4. (1 2) (3 4).

Size of Stabilizer = 4

Verification using Orbit-Stabilizer Theorem:

$|\text{Orbit}| \times |\text{Stabilizer}| = 6 \times 4 = 24$

$|S_4| = 4! = 24$. The calculation holds.

~~(Q5) Finite Field GF(2^2).~~

(Q5) Let GF(2^2) be the finite field of order 4, constructed using the irreducible polynomial x^2+x+1 over GF(2).

i) Show that GF(2^2) forms a group under multiplication.

ii) Verify whether the set of all nonzero elements of GF(2^2) is cyclic.

Ans:

Finite Field GF(2^2)

We construct GF(2^2) (order 4) using the irreducible polynomial $P(x) = x^2+x+1$ over GF(2). In this field, coefficients are modulo 2 (so $1+1=0$ and $-1=1$). From $x^2+x+1=0$, we have the relation: $x^2 = x+1$

The elements are: $\{0, 1, x, x+1\}$.

(i) Show it is a Group under Multiplication

We consider the set of Non-zero elements $G = \{1, x, x+1\}$

⊛ Closure:

- $1 \cdot a = a$ (Identity holds).
- $x \cdot x = x^2 = x+1$ (In set).
- $x \cdot (x+1) = x^2 + x = (x+1) + x = 2x+1 = 1$ (In set).
- $(x+1)(x+1) = x^2 + 2x + 1 = x^2 + 1 = (x+1) + 1 = x$ (In set).

⊛ Identity: 1 is the identity.

⊛ Inverses:

- Inverse of 1 is 1.
- Inverse of x is $x+1$ (since product is 1)
- Inverse of $x+1$ is x.

⊛ Associativity: Multiplication of polynomials is associati

Since the set is closed, has an identity, has inverses, and is associative, it is a group.

(ii) Verify if the set of non-zero elements is cyclic

The group has order 3 ($|G| = 4-1 = 3$). A group of prime order is always cyclic.

Let's test the powers of the element x:

1. $x^1 = x$
2. $x^2 = x+1$.
3. $x^3 = x \cdot x^2 = x(x+1) = x^2 + x = (x+1) + x = 1$.

Since x generates all non-zero elements $\{x, x+1, 1\}$, the group is cyclic with generator x (and also generator

(Q6) Let $GL(2,R)$ be the general linear group of 2x2 invertible matrices over R. Show that the set of scalar matrices forms a normal subgroup of $GL(2,R)$. Construct the corresponding factor group, and interpret its structure.

Ans:

GL(2,R) and Scalar Matrices

The Setup

* $G = GL(2, \mathbb{R})$: The General Linear Group of 2x2 invertible matrices with real entries.
* Z: The set of scalar matrices, defined as matrices of the form $kI = \begin{pmatrix} k & 0 \\ 0 & k \end{pmatrix}$, where $k \in \mathbb{R} \setminus \{0\}$.

(i) Proof that Z is a Normal Subgroup

To prove Z is a normal subgroup, we must show it is a subgroup and that ~~is~~ it is invariant under conjugation.

1. Subgroup Test:

- Identity: If $k=1$, we get the identity matrix I, Thus $I \in Z$.
- Closure: $(aI)(bI) = (ab)I$, Since ~~a≠1~~ $a, b \neq 0$, $ab \neq 0$, so the result is in Z.
- Inverses: The inverse of kI is $(1/k)I$. Since $k \neq 0$, $1/k$ exists.

2. Normality Test:

We must ~~know~~ show that for any matrix $A \in G$ and any scalar matrix $S \in Z$, the value ASA^{-1} is in Z.

Because scalar matrices commute with all matrices (they form the Center of the group):

$$ASA^{-1} = A(kI)A^{-1} = k(AIA^{-1}) = k(AA^{-1}) = KI = S$$

since the result is S (which is in Z), Z is a normal subgroup.

(ii) The Factor Group and Interpretation

The factor group (quotient group) is G/Z.

⊛ Construction: The elements of this group are cosets AZ. Two matrices A and B are in the same coset if $A = cB$ for some scalar c.

Essentially, matrices that differ only by a scalar multiple are considered "equal" in this group.

⊛ Structure: This group is isomorphic to $PGL(2, \mathbb{R})$ (the Projective General Linear Group).

⊛ Interpretation: Geometrically, $GL(2, \mathbb{R})$ represents linear transformations. $PGL(2, \mathbb{R})$ represents projective transformations. In projective geometry, ~~scalt~~ scaling a vector doesn't change the point it represents

(just as the point (2,4) and (1,2) lie on the same line through the origin). The factor group "mods out" the concept of magnitude/scaling, leaving only the directional transformation.

(Q.7)

Explain the Diffie-Hellman key exchange protocol and its application in ~~se~~ secure communication. Discuss the security of the Diffie-Hellman protol against common attacks such as man-in-the-middle and the role of the discrete logarithm problem in ensuring its security. What would be the impact on the protocol's security if the prime modulus used is not sufficiently large?

Ans:

The protocol

Diffie-Hellman allows two parties (Alice and Bob) to establish a shared secret over an insecure channel. It does not encrypt messages itself; it generates the key for symmetric encryption (like AES).

1. Public Parameters: Alice and Bob agree on a large prime p and a generator g.
2. Private keys: Alice picks secret integer a. Bob picks secret integer b.
3. Public Exchange:
   - Alice computes $A = g^a \pmod p$ and sends it to Bob.

- Bob computes $B = g^b \pmod p$ and sends it to Alice.

4. Shared Secret:

- Alice calculates $S = B^a \pmod p$.
- Bob calculates $s = A^b \pmod p$.
- Mathematically:
$(g^b)^a = g^{ba} = g^{ab} = (g^a)^b \pmod p$. They now possess the same numbers.

Security and the Discrete Logarithm Problem (DLP)

The security relies on the ~~assy~~ asymmetry of modular exponentiation.

- Easy: Computing $g^a \pmod p$.
- Hard: Given $g^a \pmod p$, determining a. This is the Discrete Logarithm Problem. As long as p is large enough, no classical algorithm can solve this in reasonable time.

Vulnerabilities:

1. Man-in-the-Middle (MITM): Diffie-Hellman does not authenticate users. An attacker (Mallory) can intercept Alice's signal, send her own public key to Alice, and do the same to Bob. Alice thinks she has a secure key with Bob, but she actually has one with Mallory.

This is solved by using Digital Signatures (PKI) to sign the public keys.

2. Small Prime Modulus:

If p is small (e.g., 512 bits), attackers can use the Number Field Sieve (NFS) algorithm to solve the discrete log. Modern standards require p to be at least 2048 bits.

(Q8) Let G be a group, and let H be a subgroup of G. Prove that the intersection of any two subgroups of G is also a subgroup of G. Provide an example using specific groups.

Ans:

Theorem: Let G be a group and let H, K be subgroups of G. The intersection $H \cap K$ is a subgroup of G.

proof:

Let $D = H \cap K$. We use the subgroup test.

1. Identity: Since H and K are subgroups, the identity element e must be in both H and K. Therefore, $e \in D$.

2. Closure: Let $x, y \in D$. By definition, $x, y \in H$ and $x, y \in K$.

- Since H is a subgroup, $xy \in H$.
- Since K is a subgroup, $xy \in K$.
- Therefore, $xy \in D$.

3. Inverses: Let $x \in D$

- Since $x \in H$ and H is a subgroup, $x^{-1} \in H$.
- Since, $x \in K$ and K is a subgroup, $x^{-1} \in K$.
- Therefore, $x^{-1} \in D$.

Example:
Let $G = \mathbb{Z}$ (Integer under addition).

- Let $H = 2\mathbb{Z} = \{\ldots, -6, -4, -2, 0, 2, 4, 6, \ldots\}$ (Multiples of 2)
- Let $K = 3\mathbb{Z} = \{\ldots, -6, -3, 0, 3, 6, \ldots\}$ (Multiples of 3).
- $H \cap K = \{\ldots, -6, 0, 6, 12, \ldots\} = 6\mathbb{Z}$,

Multiples of 6 form a valid subgroup of the integers.

(Q9) Prove that Z_n is commutative and identify whether it has zero divisors.
Further, determine the conditions under which Z_n is a field.

Ans:
Ring Z_n: Commutative and Fields
Commutativity the ring Z_n consists of integers $\{0, 1, \ldots, n-1\}$ under addition and multiplication modulo n. Since standard multiplication is commutative ($ab = ba$) modular multiplication is also commutative:

$$a \cdot b \equiv ab \equiv ba \equiv b \cdot a \pmod{n}.$$

Thus, Z_n is always a commutative ring with unity (1).

Zero Divisors A zero divisor is a non-zero element a such that $a \cdot b = 0$.

* $\mathbb{Z}_n$ has zero divisor is a non-zero element a such that $a \cdot b = 0$.
* proof: If n is composite, we can write $n = a \cdot b$ where $1 < a, b < n$. In modulo n, this means $a \cdot b \equiv 0 \pmod{n}$. Both a and b are zero divisors.
  - Example: In $\mathbb{Z}_6$, $2 \cdot 3 = 6 \equiv 0$. 2 and 3 are zero divisors.

Conditions to be a Field: A field is a commutative ring where every non-zero element has a multiplicative inverse.

* $\mathbb{Z}_n$ is a field if and only if n is prime.
* If n is prime, for any non-zero $a < n$, the $\gcd(a, n) = 1$.

By Bezout's identity, there exist integers x, y such that $ax + ny = 1$. Taking modulo n, we get $ax \equiv 1 \pmod{n}$. Thus, x is the inverse of a.

(Q10) Explain the vulnerabilities of the DES (Data Encryption Standard) cipher and why it is considered insecure for modern use. Discuss the role of brute-force attacks in breaking DES and the impact of key length on the security of the algorithm. How did the development of AES address the shortcomings of DES, particularly in terms of key size and resistance to cryptanalytic attacks?

Ans: Vulnerabilities of DES (Data Encryption Standard)

DES was the standard from 1977 until the early 2000s.

If failed due to advancements in raw computing power.

1. Key length: DES uses a 56-bit key. This creates a key space of $2^{56} \approx 7.2 \times 10^{16}$. While massive in 1977, modern hardware can brute-force check every possible key in less than a day.

2. Block Size: DES uses 64-bit blocks. This makes it vulnerable to "collision" attacks (like the sweet32 attack) when encrypting large amounts of data. (e.g., hundreds of GBs).

How AES (Advanced Encryption Standard) addressed this

AES (Rijndael) replaced DES in 2001.

1. Key size
   - AES supports 128, 192 and 256-bit keys.
   - AES-128 has 3.4×10^{38} possible keys. Every computers combined on earth would take billion years to brute-force

2. Structure:
   - DES used a Feistel Network (splitting bits into halves)
   - AES uses a "Substitution-Permutation Network" (SPN).

This provides stronger mathematical resistance (confusion and diffusion) against analytic attacks like Linear and Differential Cryptanalysis.

3. Efficiency: AES was designed to be efficient in both software and hardware (chips), making it ubiquitous in everything from WIFI to banking.

Q.11) Differential Cryptanalysis is a widely known attack against block ciphers.

(i) Explain how the feistal structure of DES handles differential cryptanalysis.

(ii) How AES with its subBytes, ShiftRows, MixColumns and AddRoundKey operations, is more resistant to such attacks compared to DES?

Ans:

Differential Cryptanalysis analyzes how differences in plaintext (usually XOR differences) propagate to differences in ciphertext

(i) The Feistal Structure of DES: DES uses a Feistal Network, which divides the data block into two halves (L, R). In each round, the right half goes through a function F and is XORed with the left half: $L_{i+1} = R_i$ and $R_{i+1} = L_i \oplus F(R_i, K_i)$.

* Handling Differential Attacks:

The resistance of DES relies almost entirely on its S-boxes (Substitution Boxes) inside the function F.

* Historical Context:

When DES was designed (mid-1970s), the IBM team and the NSA were already aware of differential cryptanalysis (calling it the "T-attack"). They specifically tuned the S-box entries to minimize the probability that specific input differences would lead to specific output differences.

⊛ Limitation:
While the S-boxes are optimized, the Feistal structure itself essentially just "shuffles" the problem. If the S-boxes were linear, the whole system would be trivially breakable.

(ii) AES and the Wide Trail Strategy
AES is a ~~Subpos~~ Substitution-Permutation Network (SPN) not a Feistel Network. It was designed specifically to resist differential and linear cryptanalysis ~~the~~ using the Wide Trail Strategy.

⊛ SubBytes (Non-linearity):
This is the only ~~linear~~ non-linear step. It uses an S-box based on multiplicative ~~inversion~~ inversion in the finite field GF(2^8). This specific algebraic structure makes the differential probability of the S-box very low and uniform (close to ideal).

⊛ MixColumns and ShiftRows (Diffusion):
This is the core of the Wide trail strategy.

- ShiftRows moves bytes to different columns.
- MixColumns mixes the bytes within a column vertically.
- Result: These two steps guarantee a high "branch number".

A difference in just one byte of the state is guaranteed to spread to at at least 4 bytes after one round, and to all 16 bytes after just two rounds.

This rapid diffusion makes tracking a "differential characteristic" (a specific path of differences) ~~probability~~ probabilistically impossible over 10+ rounds.

(Q12) Using the Extended Euclidean Algorithm, ~~demono~~ demonstrate how to find the modular inverse of an integer 'a' modulo 'n' (where 'a' and 'n' are coprime). How is this Algorithm utilized in RSA key generation, and why is the efficiency of this algorithm important for large-scale cryptographic systems?

Ans:

Extended Euclidean Algorithm (EEA) and RSA

(a) Demonstration

To find the modular inverse of a modulo n.

Let $a=47$ and $n=9$. We want x such that $47x \equiv 1 \pmod 9$

Note: normally $a<n$, but the math holds regardless.

Let, $a=9$, $n=47$ for standard RSA context where we find $d = e^{-1} \pmod \phi$.

Goal: Find $9^{-1} \pmod{47}$.

step 1: Euclidean algorithm to find GCD.

$$47 = 5(9) + 2$$

$$9 = 4(2) + 1$$

$$2 = 2(1) + 0$$

GCD is 1 (coprime).

step 2: ~~Back-sub~~

Back-substitution (Extended step) to Express 1 as linear combo of 9 and 47. From line 2: $1 = 9 - 4(2)$ substitute "2" from line 1: $1 = 9 - 4(47 - 5(9))$ Distribute: $1 = 9 - 4(47) + 20(9)$ combine terms: $1 = 21(9) - 4(47)$.

Thus, $21(9) \equiv 1 \pmod{47}$. The modular inverse is 21.

(ii) Utilization in RSA in RSA key Generation:

1. Compute $N = pq$ and $\phi(N) = (p-1)(q-1)$,
2. choose public exponent e,
3. Compute private exponent d using EEA: $d = e^{-1} \pmod{\phi(N)}$

The algorithm guarantees d exists (since e and $\phi(N)$ are coprime) and finds it directly.

(iii) Importance of Efficiency: In modern RSA, the modulus N is 2048 or 4096 bits.

* Naive Approach: Checking every number to see if it's the inverse would take $O(n)$ time (exponential in terms of bit-length), which is impossible.

* EEA Efficiency: EEA runs in $O(\log(\phi(N))^2)$ time. It scales linearly with the number of bits. Without this logarithmic efficiency, generating keys for secure systems would be computationally infeasible.

(Q13) Consider the following modes of operation for block ciphers. ECB (Electronic Codebook), CBC (Cipher Block Chaining), and CTR (Counter Mode).

(i) For a block cipher with block size n, mathematically prove why ECB mode is insecure for encrypting highly redundant data.

(ii) Derive the recurrence relation for CBC mode encryption and decryption and prove that error propagation is limited in decryption.

Ans:

Block Cipher Modes: ECB vs. CBC

(i) Security of ECB (Electronic Codebook) Let E_k be the encryption function. In ECB the message is divided into blocks $P_1, P_2, \ldots$ and encrypted independently: $C_i = E_k(P_i)$

Mathematical Proof of Insecurity: ~~tet~~ Let P be a highly ~~tet~~ redundant data ~~stem~~ stream (e.g., a bitmap image of a black logo on a white background). Assume two blocks at indices i and j are identical: $P_i = P_j$.

$$C_i = E_k(P_i) = E_k(P_j) = C_j$$

$$\therefore P_i = P_j$$

$$\Rightarrow C_i = C_j$$

This creates a deterministic leakage of patterns. The ciphertext preserves the statistical structure of the plaintext. An attacker can "see" the image despite the

encryption (the famous "Linux Penguin" ECB visualization).

ii) CBC (Cipher Block chaining) Recurrence & Error Propagation Recurrence Relations:

* Encryption: $C_i = E_K(P_i \oplus C_{i-1})$ (where $C_0 = IV$).
* Decryption: $P_i = D_K(C_i) \oplus C_{i-1}$

Proof of limited Error Propagation: Suppose a transmission ~~Suppo~~ error corrupts ciphertext block C_i (a bit flip). We analyze the effect on decryption:

1) Block i Decryption:

$$P_i' = D_k(C_i') \oplus C_{i-1}$$

since C_i' is input into the non-linear decryption function D_k, the output is completely randomized (avalanche effect). Block P_i is totally garbled.

2) Block i+1 Decryption:

$$P_{i+1}' = D_k(C_{i+1}) \oplus C_i'$$

Here, C_i' is only used in the final XOR operation. The error in C_i flips exactly the corresponding bits in P_{i+1}. Block P_{i+1} has bit-specific errors but is recoverable.

3) Block i+2 Decryption:

$$P_{i+2} = D_k(C_{i+2}) \oplus C_{i+1}$$

This depends only on C_{i+2} and C_{i+1}. Neither was corrupted. Block P_{i+2} is perfectly correct.

Conclusion: The error propagates to the current block and next block, then stops. It does not corrupt the entire stream.

(Q14) Why the linearity of LSFRs makes them vulnerable to known plaintext attacks, and purpose a mathematical method to mitigate this vulnerability.

Ans:
Vulnerabilities of LSFRs (Linear Feedback Shift Registers)

An LSFR generates a stream based on a linear recurrence relation over a finite field (usually GF(2)):

$$S_{i+L} = C_{L-1}S_{i+L-1} + \cdots + C_1 S_{i+1} + C_0 S_i \pmod 2.$$

Because this operation is entirely linear, it can be solved using linear algebra.

⊛ Attack: If an attacker knows 2L bits of the plaintext (known-plaintext attack), they can derive the corresponding keystream bits.

⊛ Algorithm: They can use the Berlekamp-Massey Algorithm. This algorithm takes the output stream and efficiently reconstructs the "connection polynomial" (the feedback taps).

Once the polynomial and the current states are known, the attacker can predict the entire future sequence and reverse engineer the past.

Proposed Mitigation: To make LSFRs secures, we must destroy the linearity. We use non-linear combiners

1. Run multiple LSFRs of different lengths (relatively prime lengths) in parallel.

2. Feed their outputs into non-linear Boolean function $f(x_1, x_2, \ldots, x_n)$.

⊛ Example: The Geffe Generator or the A5/1 cipher (used in GSM)

⊛ Condition: The function f must have high "algebraic immunity" so that the output cannot be approximated by a linear function.

(Q15) Let M be the set of all possible plaintexts, K the set of keys, and C the set of ciphertexts in a cryptographic system.

(i) State Shannon's definition of perfect secrecy mathematically.

(ii) Prove that the One-Time Pad achieves perfect secrecy under the condition that the key K is uniformly random and $|K| \geq |M|$.

(iii) Critically analyze why perfect secrecy is impractical for large-scale communication systems.

Ans:

Shannon's Perfect Secrecy

(i) Mathematical Definition

A cryptosystem has a perfect secrecy if the ciphertext yields no information about the plaintext. For all plaintexts $m \in M$ and ciphertexts $c \in C$:

$$P(M=m \mid C=c) = P(M=m).$$

Alternatively: $P(C=c \mid M=m) = P(C=c)$

(ii) Proof for One-Time ~~ma~~ Pad (OTP) conditions:

1. key K is chosen uniformly at random: $P(K=k)=\frac{1}{|K|}$
2. $|K| \geq |M|$ (Usually $|K|=|M|=|C|$).
3. Encryption is $c = m \oplus k$.

proof: We want to find $P(C=c \mid M=m)$. For a fixed message m and a target ciphertext c, there exists exactly one key $k = m \oplus c$ that produces c. Since keys are uniform:

$$P(C=c \mid M=m) = P(K=m \oplus c) = \frac{1}{|K|}$$

Since this probability $\frac{1}{|K|}$ is constant regardless of ~~M~~ m, the condition for perefect secrecy holds. The interceptor sees every possible plaintext as equally likely to have produced the ciphertext.

(iii) Critical Analysis of Impracticality

While theoretically unbreakable, perfect secrecy is impractical for modern networks (like the internet) because of the Key Distribution Problem.

- Constraint: $|K| \geq |M|$.
- Implication: To send a 1 GB file securely, you must physically pre-share a 1 GB random key.
- Paradox: If you have a secure channel capable of transmitting the 1 GB key secretly, you could have just sent the message itself over that channel.
- Modern Solution: We trade "perfect secrecy" for "Computational Security" (AES/RSA), where it is theoretically possible to break the code, but computationally infeasible.

(Q16) A linear Congruential Generator (LCG) is defined by the recurrence relation: $x_{n+1} = ax_n + c \pmod m$ where x_n is the sequence of pseudo-random numbers, and a, c, and m are integer parameters representing the multiplier, increment and modulus respectively. Using specific values for a, m and c, compute the first 5 numbers of an LCG sequence starting with a given seed $x_0 = 7$.

16 | 38 | 2
32
6

Ans:

Sequence computation:

Parameters: Let's choose typical small parameters for demonstration.

Multiplier $a = 5$

Increment $c = 3$

Modulus $m = 16$

seed $x_0 = 7$

Formula: $x_{n+1} = (5x_n + 3) \pmod{16}$

Sequence computation:

1. $n = 0$, $x_1 = (5x_0 + 3) \pmod{16} = (5(7) + 3) \pmod{16} = 38 \pmod{16} = 6$
2. $n = 1$, $x_2 = (5x_1 + 3) \pmod{16} = (5(6) + 3) \pmod{16} = 33 \pmod{16} = 1$
3. $n = 2$, $x_3 = (5(1) + 3) \pmod{16} = 8 \pmod{16} = 8$
4. $n = 3$, $x_4 = (5(8) + 3) \pmod{16} = 43 \pmod{16} = 11$
5. $n = 4$, $x_5 = (5(11) + 3) \pmod{16} = 58 \pmod{16} = 10$.

Resulting sequence: 6, 1, 8, 11, 10.

(Q17) Define a ring in abstract algebra and explain its key properties. Provide an example of commutative ring. How does a concept of a ring relate to the construction of a finite fields, and what roles do rings play in cryptographic algorithms like RSA?

Ans:

Ring Theory and Cryptography:

Definition of a Ring: In abstract algebra, a Ring $(R, +, \cdot)$ is a set equipped with two binary ~~operations~~ operations: addition $(+)$ and multiplication $(\cdot)$. It must satisfy three key properties:

1. Abelian Group under Addition:
It is closed, associative, has an identity (0), and every element has an additive inverse. It is also ~~com~~ commutative $(a+b = b+a)$.

2. Monoid under Multiplication:
It is closed and associative (If it has a multiplicative identity 1, it is a "Ring with Unity").

3. Distributive Laws:
Multiplication distributes over addition: $a \cdot (b+c) = (a \cdot b) + (a \cdot c)$

Examples:

- Commutative Ring: The set of integers $(\mathbb{Z})$. Order of multiplication does not matter $(2 \times 3 = 3 \times 2)$.

• Non-Commutative Ring: The set of 2×2 Matrices. Matrix multiplication is generally not commutative. ($AB \neq BA$).

Relation to Finite Fields & RSA:

⊛ Finite Fields: A finite field (Galois field) is a special type of commutative ring where every non-zero element has a multiplicative inverse (division is possible).
We construct fields like GF(p) by taking the ring of integers modulo a prime p.

⊛ Role in RSA: RSA operates in the ring $\mathbb{Z}_n$ (integers modulo n, where $n=pq$).

- $\mathbb{Z}_n$ is a commutative ring, not a field (because it has zero divisors and non-invertible elements.
- RSA relies on the Euler's Theorem property within this ring structure: $a^{\phi(n)} \equiv 1 \pmod{n}$ for units in the ring.

(Q18) Given an RSA key pair with the public key (e,n) and the private key d, where $p=5$, $q=11$ (two prime numbers), $n=p \cdot q$ and $\phi(n)=(p-1)(q-1)$. Use the RSA algorithm to encrypt a message, $M=2$ and decrypt the ciphertext to recover the original message.

Let, $p=7$ and $q=3$, sign the hash of a message $H(m)=3$ using the private key d. Verify the signature using the public key (e,n). Explain how the signature

ensures the integrity and authenticity of the message.

Ans:

RSA - Encryption and Signing

(i) Encryption:

setup:

* $p=5, q=11$

$n = p \cdot q = 5 \cdot 11 = 55$

* $\phi(n) = (5-1)(11-1) = 4 \cdot 10 = 40.$

* choosing public Exponet e:

e must be coprime to 40. Let $e=3$.

* calculating private key d: $d \equiv e^{-1} \pmod{40}$.

Using Extended Euclidean: $3 \times 27 = 81 = 2(40)+1$

so, $d = 27$.

Encryption ($M=2$):

$C = M^e \pmod{n} = 2^3 \pmod{55} = 8$

Decryption ($C=8$):

$M = C^d \pmod{n} = 8^{27} \pmod{55}$

Calculation Tricks:

$8^{27} = (2^3)^{27} = 2^{81}$

By Euler's theorem, $2^{40} \equiv 1 \pmod{55}$

$2^{81} = 2^{40}$.

$2^{40} \cdot 2^1 = 1 \cdot 1 \cdot 2 \pmod{55}$

(ii) Digital Signature

Setup:

- $p=7, q=3$

$\Rightarrow n = p \cdot q = 7 \cdot 3 = 21$

- $\phi(n) = (6)(2) = 12$.
- Choose e : coprime to 12, let $e=5$
- Calculate d : $5d \equiv 1 \pmod{12}$

Since $5 \times 5 = 25 = 2(12)+1$

$\therefore d = 5$

Signing $(H(m) = 3)$:

$S = H(m)^d \pmod{n} = 3^5 \pmod{21}$.

$3^3 = 27 \equiv 6$

$3^5 = 3^3 \cdot 3^2 = 6 \cdot 9 = 54$

$54 \pmod{21} = 54 - 42 = 12$

Signature $S = 12$.

Verification:

$V = S^e \pmod{n} = 12^5 \pmod{21}$.

$12 \equiv -9 \pmod{21}$.

Calculated manually: $12^2 = 144 = 6(21) + 18 = -3$.

$12^4 \equiv (-3)^2 = 9$

$12^5 = 12^4 \cdot 12 = 9 \cdot 12 = 108$.

$108 = 5(21) + 3$

$V = 3$. Since $V = H(m)$, the signature is valid.

Explanation:

* Integrity: If the message changes, the hash $H(m)$ changes. The signature verification would fail because $S^e \neq H(m')$
* Authenticity: Only the holder of the private key d could have calculated an S such that $S^e = H(m)$.

(Q19) Given the elliptic curve equation $y^2 = (x^3 + ax + b) \pmod p$ where $p = 23$, $a = 1$ and $b = 1$:

(i) Verify if the point $P = (3, 10)$ lies on the curve.

(ii) Find the result of doubling the point $P(2P)$ using the elliptic curve point doubling formula.

(iii) Compute the addition of $P = (3, 10)$ and $Q = (9, 7)$ on the curve.

Ans:

Elliptic Curve Arithmetic

$y^2 = x^3 + x + 1 \pmod{23}$

$(a = 1, b = 1, p = 23)$

(i) Verify Point P(3,10):

- LHS: $y^2 = 10^2 = 100$

  $100 \div 23 = 4$ rem 8. LHS = 8

- RHS: $x^3 + x + 1 = 3^3 + 3 + 1 = 27 + 4 = 31$

  $31 \div 23 = 1$ rem 8 RHS = 8.

- Result: $8 = 8$, so P lies on the curve.

(ii) Point Doubling (2P) :

Slope $\lambda = \frac{3x_1^2 + a}{2y_1} \pmod p$

1. Numerator:
$3(3)^2 + 1 = 27 + 1 = 28 \equiv 5 \pmod{23}$.

2. Denominator:
$2(10) = 20$.

3. Inverse of 20 mod 23: $20 \equiv -3$. Inverse of -3 is -8.
(since $-3 \times -8 = 24 \equiv 1$). $-8 \equiv 15$.

4. $\lambda = 5 \times 15 = 75 \equiv 6 \pmod{23}$.

$x_3 = \lambda^2 - 2x_1 = 6^2 - 2(3) = 36 - 6 = 30 \equiv 7$

$y_3 = \lambda(x_1 - x_3) - y_1 = 6(3-7) - 10$
$= 6(-4) - 10$
$= -24 - 10$
$= -34$

$-34 \equiv 12 \pmod{23}$ $2P = (7, 12)$.

(iii) Point Addition (P+Q): $P = (3, 10)$, $Q = (9, 7)$.

Slope. $\lambda = \frac{y_2 - y_1}{x_2 - x_1} \pmod p$

1. Numerator: $7 - 10 = -3 \equiv 20$
2. Denominator: $9 - 3 = 6$
3. Inverse of 6 mod 23: $6 \times 4 = 24 \equiv 1$.
Inverse is 4.

4. $\lambda = 20 \times 4 = 80$.

$80 = 3(23) + 11$

$\lambda = 11$.

$x_3 = \lambda^2 - x_1 - x_2 = 11^2 - 3 - 9 = 121 - 12 = 109$.

$109 = 4(23) + 17$

$x_3 = 17$

$y_3 = \lambda(x_1 - x_3) - y_1 = 11(3 - 17) - 10$

$= 11(-14) - 10$

$= -154 - 10 = -164$

$-164 \pmod{23}$: $23 \times 8 = 184$.

$-164 + 184 = 20$

$P + Q = (17, 20)$.

(Q2) Suppose an elliptic curve $y^2 = (x^2 + 7x + 10) \pmod{37}$ is used for ECDSA, with the base point $G = (2,5)$ and order $n = 19$.

Generate a private key $d = 9$ and compute the ~~corres~~ corresponding public key $Q = dG$ using scalar multiplication

(i) Sign the hash of a message $H(M) = 8$ using a random

(ii) Compute the signature pair (r,s) using ECDSA formulas.

(iii) Verify the signature (r,s) using the public key Q and demonstrate that the signature is valid.

Ans:

ECDSA Signature Generation & Verification

parameters:

Curve: $y^2 = x^2 + 7x + 10 \pmod{37}$

$G = (2,5)$, order $n = 19$.

Private key $d = 9$.

(i) Sign Hash $H(M) = 8$ with nonce $k = 3$, first, we must calculate the point $R = kG = 3G$.

- Find $2G$: $\lambda = \frac{3(2)^2 + 7}{2(5)} = \frac{19}{10} \pmod{37}$.

$$10^{-1} \pmod{37} = 26$$

$$\lambda = 19 \times 26 = 494 \equiv 13 \pmod{37}.$$

$$x_{2G} = 13^2 - 2(2) = 169 - 4 = 165 \equiv 17.$$

~~$= 169 - 4$~~

$$y_{2G} = 13(2 - 17) - 5 = 13(-15) - 5$$

$$= -200 \equiv 22$$

$$2G = (17, 22).$$

- Find $3G$ $(2G + G)$:

$$\lambda = \frac{5 - 22}{2 - 17} = \frac{-17}{-15} = 17 \cdot 15^{-1} \pmod{37}$$

$$\Rightarrow 15^{-1} \pmod{37} = 5$$

(since $15 \times 5 = 75 = 2(37) + 1$)

$$\lambda = 17 \times 5 = 85 \equiv 11$$

$x_{3G} = 11^2 - 17 - 2$
$= 121 - 19 = 102 \equiv 28$.

y_{3G} = ... (we only need the x-coordinate for the signature).

(ii) Compute Signature (r,s).

1. Calculate r: $r = x_{3G} \pmod n$. r
$= 28 \pmod{19} = 9$

2. calculate s:

$s = k^{-1}(H(M) + d \cdot r) \pmod n$.

$k = 3$.

Inverse of 3 mod 19 is 13 (since $3 \times 13 = 39 = 2(19)+1$

$s = 13(8 + 9 \cdot 9) \pmod{19}$.

$\Rightarrow s = 13(8+81) = 13(89)$

~~13(89)~~ $89 \pmod{19} = 13$ (since $19 \times 4 = 76$)

$\Rightarrow s = 13 \times 13 = 169$ ~~169~~

$169 \pmod{19} = 17$ (since $19 \times 8 = 152$)

Signature Pair : $(r,s) = (9,17)$

(iii) Verification Public Key Q

$Q = dG = 9G$ (Assume valid point on curve).

We verify using $(r,s) = (9,17)$.

1. Calculate w : $w = s^{-1} \pmod n$
$= 17^{-1} \pmod{19}$

$17 \equiv -2$

Inverse of $-2 \bmod 19$ is 9 $(-2 \times 9 = -18 \equiv 1)$.

$\therefore w = 9$.

2. Calculate u_1 and u_2:

$u_1 = H(M) \cdot w \pmod n$

$= 8 \times 9 = 72 \equiv 15$

$u_2 = r \cdot w \pmod n$

$= 9 \times 9 = 81 \equiv 5$

3. Calculate point X:

$X = u_1 G + u_2 Q_A = 15G + 5(9G)$.

$= 15G + 45G$

$= 60G$.

since, the order is $n = 19$

$60G \equiv (60 \bmod 19)\, G = 3G$.

Q21 Explain the key properties of cryptographic hash functions such as those in the Secure Hash Algorithm (SHA) family (e.g., SHA-256). Specifically:

(i) What are the essential characteristics of a secure hash function (e.g., pre-image resistance, collision resistance)?

(ii) How does the length of the output hash (e.g., 256bits in SHA-256) impact the security of the algorithm?

(iii) Discuss how SHA is utilized in real-world applications, such as digital signatures and blockchain systems.

Ans: Cryptographic Hash functions (SHA family)

(i) Essential Characteristics

A cryptographic hash function $H(x)$ takes an input of arbitary length and produces a fixed-size string.
To be considered secure, it must satisfy:

1. Pre-image Resistance (One-Wayness):

Given a hash value h, it should be computationally infeasible to find any message m such that $H(m)=h$.

2. Second Pre-image Resistance:

Given a specific input m_1, it should be infeasible to find a different input m_2 such that $H(m_1)=H(m_2)$

3. Collision Resistance:
It should be infeasible to find any two different inputs m_1 and m_2 such that $H(m_1) = H(m_2)$.

4. Avalanche Effect:
A change in just one bit of the input should change roughly half the bits in the output hash.

(ii) Impact of Output Length
The output length (n bits) directly determines the difficulty of brute-force attacks:

⊛ Pre-image Resistance:
Depends on 2^n. For SHA-256, an attacker must try 2^{256} operations to reverse a hash.

⊛ Collision Resistance:
Depends on $2^{n/2}$ due to the Birthday Paradox.

- The Birthday Paradox states that in a group of just 23 people, there is a 50% chance two share a birthday. Similarly, finding any collision is much easier than finding a specific one.
- For SHA-256, collision resistance is 2^{128}, which is currently considered unbreakable. If the hash length were too short (e.g., 64 bits), collisions could be found in 2^{32} operations (mere seconds).

(iii) Real-World Applications :

⊛ Digital Signatures : Encrypting a large file with private key is slow. Instead, the system hashes the file (e.g., to 256 bits) and signs only the hash. This ensures integrity because if the file changes, the hash changes, invalidating the signature.

⊛ Blockchain :

- Merkle Tree: Transactions are hashed in pairs up to a single "Root Hash". This allows efficient verification of data integrity without downloading the whole chain.
- Proof of work : Miners expend energy trying to find a nonce that, when hashed with the block header, results in a hash with a specific number of leading zeros.

Q22 Explain the concept of Galois fields (also known as finite fields), focusing on GF(p) and GF(2^n). How are Galois fields used in the construction of cryptographic primitives, such as in elliptic curve cryptography and the AES ~~encryption~~ encryption algorithm? Discuss the importance of field arithmetic in these cryptographic systems.

Ans:

Galois Fields (Finite Fields)

concept: GF(p) and GF(2^n) A Galois Field is a finite set of elements where addition, subtraction, multiplication, and division (except by zero) are well-defined and satify specific algebraic laws.

1. Prime Fields GF(p): The elements are integers $\{0,1,\ldots,p-1\}$. Arithmetic is performed modulo ~~on irreducible polynomial of degree n. This is essential for computers because elements map perfectly to binar~~ a prime p.

2. Extension Fields GF(2^n):
The elements are polynomials with binary coefficients (0 or 1) of degree less than n. Arithmetic is performed ~~mw~~ modulo an irreducible polynomial of degree n. This is essential for computers because elements map perfectly to binary bytes.

Usage in Cryptography:

⊛ Elliptic Curve Cryptography (ECC):
ECC ~~replies~~ relies on finding points (x,y) that satify an equation like $y^2 = x^2 + ax + b$ over a finite field.

- Standard curves (e.g., NIST P-256) use GF(p) where p is a very large prime. The discrete

nature of the field turns the curve into a cloud of scattered points, making the discrete logarithm problem hard.

* AES Encryption:

AES treats every bytes as an element of $GF(2^8)$.

- The S-Box (SubBytes): This is the core non-linear component. It replaces a byte with its multiplicative inverse in $GF(2^8)$ (followed by an ~~in~~ affine transformation
- Importance: Using a field allows for inversion. In real numbers, dividing by 7 yields an infinite decimal. In $GF(2^8)$, every non-zero byte has an unique inverse. This allows the encryption to be perfectly reversible (decryption) without rounding errors.

Q23 Lattice-based cryptography is considered a promising candidate for post-quantum cryptography due to its resistance to attacks ~~qx~~ by quantum computers.

(i) Explain the Shorest Vector Problem (SVP) and its role in the security of lattice-based cryptographic schemes.

(ii) Compare the security assumptions of lattice-based cryptography with traditional cryptographic schemes like RSA and ECC in the context of Shor's algorithm.

(iii) Discuss how quantum cryptography differs from lattice-based cryptography, particularly in terms of their goals and underlying principles.

Ans:

Lattice-Based Cryptography

(i) The shortest Vector Problem (SVP)

A lattice is a grid of points in n-dimensional space generated by a set of basis vectors.

⊛ The problem:

Given a basis for a lattice (which might be "bad" long and skew), find the shortest non-zero vector in the grid.

⊛ Role in Security:

In high dimension (e.g., n=500), this problem is exponentially hard. Cryptosystems like Kyber (ML-KEM) rely on variants like "Learning with Errors" (LWE), which effectively asks an attacker to solve approximate SVP to recover the secret key.

(ii) Security Assumptions vs. RSA/ECC (Shor's Algorithm)

⊛ RSA/ECC: Their security relies on the Hidden Subgroup problem (specially periodicity). Shor's algorithm utilizes quantum superposition to find the "period" of a function efficiently ($O(n^3)$), breaking these systems

completely.

~~⊛ Lattice-Based :~~

⊛ Lattice-Based :

SVP is a geometric optimization problem. It has no known periodic structure for Shor's algorithm to exploit. The best known quantum attacks (like Grover's search) only offer a quadratic speed up ($O(\sqrt{N})$), which can be countered by simply ~~sig~~ slightly increasing the key size.

(iii) Quantum Crypto (QKD) vs. Post-Quantum Crypto (PQC)

⊛ Post-Quantum Cryptography (PQC) : Software ~~algorithm~~ algorithms (math) running on classical computers that are designed to resist quantum attacks. (Example: Lattice, Hash-based).

⊛ Quantum Cryptography (e.g., QKD- Quantum Key Distribution) : Hardware systems that use the physics of quantum mechanics (photons) to exchange keys. If an eavesdropper observes the photons, the quantum state collapses, ~~alt~~ alerting the users.

• Difference : PQC protects the data mathematically, QKD protects the transmission channel physically.

Q.24 In a stream cipher, let the keystream $K = (k_1, k_2, k_3, \ldots)$ be generated using a linear Feedback Shift Register (LFSR) defined by the recurrence relation:

$$k_t = c_1 k_{t-1} \oplus c_2 k_{t-2} \oplus \cdots \oplus c_m k_{t-m} \text{ over } GF(2)$$

Prove that the maximum period of the keystream is $(2^m - 1)$ if the characteristic polynomial of the LFSR is primitive.

Ans:

LSFR Maximum Period Proof:

Theorem: An LFSR of length m has maximum period of $2^m - 1$ if its characteristic polynomial is primitive.

proof:

1. State Space: An LSFR with m flip-flops has 2^m possible states, represented by binary vectors $(s_0, s_1, \ldots, s_{m-1})$. The states of all zeros $(0, 0, \ldots, 0)$ is a "lock-up" state. If the register is all zeros, the XOR feedback sum will also be zero, and the register will remain zero forever. Therefore, the maximum number of non-zero states is $2^m - 1$.

2. Polynomial Representation: The recurrence relation $k_t = \sum_{i=1}^{m} c_i k_{t-i}$ corresponds to a characteristic

polynomial $P(x) = x^m + c_1 x^{m-1} + \cdots + c_m$ over GF(2).
The LFSR state transitions can be viewed as multiplication by x in the quotient ring $GF(2)[x]/P(x)$

3. Primitive Polynomials :

A polynomial of degree m is primitive if it is irreducible and is a generator of the field $GF(2^m)$. Specifically, the roots of a primitive polynomial generate the multiplicative group $GF(2^m)^*$. Since the order of this multiplicative group is exactly $2^m - 1$, the LFSR sequence will cycle through every possible non-zero element of the field (every state) before repeating.

Conclusion:

Thus, if the polynomial is primitive, the LFSR generates a Maximal Length Sequence (m-sequence) with period

$$T = 2^m - 1 .$$

(Q25) In lattice-based cryptography, particularly using the Learning With Errors (LWE) problem and Shortest Vector Problem (SVP), sign a message as follows:

(i) Explain the process of signing a message using an LWE-based signature scheme, including the key generation, signing and verification steps.

(ii) Given a message M, demonstrate how to sign the message using an LWE-based scheme with a public key P_k and private key S_k. Outline the steps of message signing and explain the role of the LWE problem in ensuring security.

Ans:

LWE-Based Signature scheme:

This describes a high-level "Fiat-Shamir with Abort" scheme (like NIST standard Dilithium).

(i) The Process

1. Key Generation:

- Create a matrix A of random polynomials.
- Generate secret vectors s_1, s_2 with small coefficients.
- Compute public key $t = As_1 + s_2$ (This is the LWE instance: it is hard to find s given A and t).

2. Signing (The Prover):

- Sample a random "masking" vector y.
- Compute $w = Ay$.
- Create a challenge c by hashing the message M and w: $c = H(M \| w)$.
- Compute potential signature $z = y + cs_1$.

3. Verification:

- The verifier computes $w' = Az - ct$.
- If the signature is valid, w' should differ from the original w only by small error terms (related to s_2).

The verifier checks if the norm $\|z\|$ is small and if the hash c matches.

(ii) Demonstration steps for Message M ~~et~~ Setup:

- Public key: (A, t)
- Private key: s (the small vector).

Step-by-Step Signing:

1. Commitment: Signer picks random vector y. Calculates, $w = Ay$.
2. Challenge: Signer calculates $c = Hash(M, w)$.
3. Response: Signer calculates $z = y + c \cdot s$.
4. Security Check: Signer checks if z is within a

safe range (essentially checking if z looks like random noise ~~at~~ rather than a shifted secret key). If unsafe, restart loop.

5. Output: The signature is the pair (z, c).

Role of LWE: The security relies on the fact that t looks indistinguishable from uniform randomness to anyone who doesn't know s (the LWE assumption). An attacker cannot forge a signature (z, c) that satisfies the linear relation $Az \approx ct$ without knowing the secret s.