Q1 With the advancement of quantum computing, traditional public key cryptosystems like RSA and ECC are at risk due to Shor's Algorithm, which can efficiently factor large numbers and compute discrete algorithms in polynomial time. This presents significant security threats to modern cryptoraphic systems, paticularly those used in secure communication financial transactions & digital signatures.

Lets discuss How Quantum Computing Threatens Cryptography

Shor's algorithm can break widely used encryption schemes:

① RSA : RSA relies on factoring large numbers, which Shor's algorithm solves in polynomial time.

② ECC : ECC is based on elliptic curve discrete logarithm problem, which quantum computers can solve efficiently.

③ Diffie-Hellman : Uses discrete logarithms, which is also vulnerable to Shor's Algorithm.

Impacts of These Threates:

① Future quantum computers could decrypt stored encrypted data.
② Digital Signature can be forged, enabling identity theft.
③ HTTPs, VPNs, and blockchain security will be compromised.

To counteract quantum attacks, new encryption methods are being developed, we can say them post-Quantum Cryptographic Algorithms. The PQC algorithms include:

① Lattice-Based Cryptography : kyber, Dilithium, NTRU, FrodoKEM are some example of this -type algorithm. They are secure because, hard lattice problems remain difficult for quantum computers.

② Code-Based Cryptography includes McEliece, this is secure because no known quantum algorithm efficiently decodes random linear codes.

③ Multivariative Polynomial Cryptography includes Rainbow which is secure because solving nonlinear polynomial equations is hard even for quantum computers.

④ Hash Based Cryptrography includes SPHINCS+, XMSS which are secure because they are Resistant to Shor's algorithm; larger hashes mitigate Grover's attack.

⑤ Isogeny-Based Cryptography include SIKE, which has resently broken.

These algorithms Resist Quantum Attacks by:
(i) Mathematical Handness (ii) Lack of Quantum speedup
(iii) Diversity of Approaches.

Q2] PRNG Generation :

A Pseudo-Random Number Generator (PRNG) is an algorith that generates a sequence of numbers that appear to be random but are actually deterministic and generated using an initial value called a seed.

Here, A PRNG algorithm using python and considering The Current Timestamp, The Process ID (os.pid) for added randomness, a modulus operation to contrain the output.

```
import random
from datetime import datetime
import time
import os
random.seed( (time.time_ns() + os.getpid()) % 10)

for i in range(5):
    print(random.randint(0, 100), end = "\t")
```

Q3] Comparision of traditional ciphers and Modern Symmetric Cipher :

Cryptography has evolved significantly from classical ciphers such as Caesar, Vigenère, and Playfair ciphers to modern symmetric encryption algorithms like AES (Advanced Encryption standard) and DES (Data Encryption Standard).

Let's explain each ciphers with their characteristics.

① Caesar Cipher :

→ Method: A simple substitution cipher that shifts each letter in the plain-text by a fixed number (e.g. shifting "A" by 3 results in "D").

→ key Length: 1 value (shift amount)

→ Security : Weak, easily broken using frequency analysis.

→ Encryption Speed : Very fast due to simple operations.

→ Weakness : Easily breakable with brute force (only 25 possible keys in English). No resistance to modern cryptanalysis.

② Vigenère Cipher :

→ Method : Uses a keyword to apply multiple Caesar shifts (polyalphabetic substitution).

→ key Length : Varies according to the length of keyword.

→ Security : Stronger than Caesar cipher but vulnerable to frequency analysis (kasiski examination, Friedman test).

→ Encryption Speed : Fast but less than a simple shift cipher.

→ Weakness : Repeating key patterns make it vulnerable to modern cryptanalysis and not secure against brute-force attacks if the key is short.

③ Playfair Cipher :

- Method : Uses a 5×5 matrix of letters for encryption by encrypting digraphs (pairs of letters).
- key Length : Varies Based on the matrix.
- Security : More secure than monoalphabetic ciphers but still vulnerable to frequency analysis of digraphs.
- Encryption Speed : Moderate, as letter-pair substitution require more processing.
- Weakness : Patterns in letter pairs can be exploited by attackers & still not resistant to modern cryptoanalysis techniques.

Modern Symmetric Ciphers

① Data Encryption Standard (DES) :

- Method : A block cipher that processes 64-bit blocks using 16 rounds of Feistel Network transformations.
- key Length : 56 bits.
- Security : Weak due to short key length (brute-force attack possible)
- Encryption Speed : Faster than traditional cipher but slow compared to AES.
- Weakness :
  - Easily broken by brute-force attacks (can be craked in hours with modern hardware).
  - Triple DES (3DES) was introduced to improve security but still not efficient.

(ii) Advanced Encryption Standard (AES) :

- Method : A block cipher that processes 128-bit block with 10, 12, or 14 rounds depending on the key length.
- Key Length : 128-bit, 192 bit, or 256 bit.
- Security : Extremely strong; resistant to brute-fore and differential cryptanalysis.
- Encryption Speed : Faster than DES, highly optimized for hardware and software.
- Strength : Used worldwide
  - Efficient on modern hardware (supports parallel processing).
  - secure against all know cryptanalysis techniques.

In conclusion, we can say, traditional ciphers are no longer secure and can be broken using modern cryptanalysis. DES is outdated due to bruteforce vulnerabilities and AES is the most secure and widely used modern symmetric cipher.

94 Action of S_4 on 2-Element Subsets of $\{1,2,3,4\}$

→ Define 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 group S_4 acts on X by permuting elements:

$$\sigma \cdot \{a,b\} = \{\sigma(a), \sigma(b)\}, \quad \forall \sigma \in S_4.$$

→ Since S_4 permutes elements of $\{1,2,3,4\}$, applying any $\sigma \in S_4$ to a subset $\{a,b\}$, produces another 2-element subset in X, ensuring the action is well defined.

→ Orbit of $\{1,2\}$:
since S_4 acts transitively on all pairs $\{a,b\}$, the orbit of $\{1,2\}$. includes all 6 subsets in X. Thus.

$$|\text{Orbit}(\{1,2\})| = 6.$$

→ Stabilizer of $\{1,2\}$
The stabilizer of $\{1,2\}$ consists all permutations $\sigma \in S_4$ that leave $\{1,2\}$ unchanged. The permulation can either swap 1 and 2, or Permute 3 and 4 freely.

→ Thus the stabilizer contains permutation of the form ① identity e ② Transposition $(1\,2)$, ③ Permutation within $\{3,4\}$, → $(3\,4)$, $(1\,2)(3\,4)$. This gives a subgroup isomorphic to $S_2 \times S_2$ (which has 4 elements)

$$\text{Stab}(\{1,2\}) = \{e, (12), (34), (12)(34)\}$$

So, $|\text{Stab}(\{1,2\})| = 4$.

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

$$\{0, 1, \alpha, \alpha+1\}$$

where, α is a root of x^2+x+1, meaning that, $\alpha^2 = \alpha+1$.

(i) $GF(2^2)$ forms a Group under Multiplication:

The nonzero elements are $\{1, \alpha, \alpha+1\}$. To show they form a group:

① Closure: Multiplication of any two elements remain in the set.

② Identity: 1 is the multiplicative identity.

③ Inverse: $\alpha \cdot (\alpha+1) = 1$, so, $\alpha^{-1} = \alpha+1$.
$(\alpha+1)^{-1} = \alpha$ since multiplication is commutative.

④ Associativity: Follows from field properties.

Thus, $\{1, \alpha, \alpha+1\}$ forms a group.

(ii) Nonzero Elements form a Cyclic Group:

we check if $\{1, \alpha, \alpha+1\}$ is cyclic:

→ $\alpha^1 = \alpha$

→ $\alpha^2 = \alpha+1$

→ $\alpha^3 = (\alpha+1)\cdot\alpha = \alpha^2+\alpha = (\alpha+1)+\alpha = 1$

Since $\alpha^3 = 1$ and it generates all elements, α is a generator.

Thus, the set $\{1, \alpha, \alpha+1\}$ is cyclic.

Q6 To show that the set of scalar matrices forms a normal subgroup of $GL(2,\mathbb{R})$, and to construct the corresponding factor group, we follow some steps.

① A scalar matrix in $GL(2,\mathbb{R})$ is a matrix of the form λI, where $\lambda \in \mathbb{R}^*$ (non-zero real numbers) and I is the identity matrix. The set of all scalar matrices is denoted as $S = \{\lambda I \mid \lambda \in \mathbb{R}^*\}$.

② Subgroup Check:

ⓐ Closure: If we multiply two scalar matrices λI and μI, we get $(\lambda I)(\mu I) = \lambda\mu I$. which is also a scalar matrix. Hence, S is closed under multiplication.

ⓑ Identity: The identity matrix I is a scalar matrix, so it is in S.

ⓒ Inverse: The inverse of a scalar matrix λI is $\frac{1}{\lambda} I$ which is also scalar matrix, so every element in S has an inverse in S.

③ Normality: To show S is a normal subgroup, we need to check that for any matrix $A \in GL(2,\mathbb{R})$ and any scalar matrix $\lambda I \in S$, the conjugate $A(\lambda I)A^{-1}$ is still a scalar matrix.

We calculate:

$$A(\lambda I)A^{-1} = \lambda(AIA^{-1}) = \lambda AA^{-1} = \lambda I.$$

This shows that the conjugate of a scalar matrix by any matrix. So, S is normal in $GL(2,\mathbb{R})$.

④ Factor Group : Now, we can form the factor group $GL(2,\mathbb{R})/S$, which represents the general linear group modulo scalar matrices. This factor group describes the "non-scalar" transformations in $GL(2,\mathbb{R})$, and it turns out to be isomorphic to the projective general linear group $PGL(2,\mathbb{R})$.

⑤ Interpretation of the Structure : The factor group $GL(2,\mathbb{R})/S$ represents the group of linear transformations in $\mathbb{R}^2$, ignoring scalar scaling. This is the group of projective transformation on $\mathbb{R}^2$ which is exactly $PGL(2,\mathbb{R})$.

Q7 Diffie-Hellman key exchange photocol :

Diffie-Hellman (DH) protocol is a method of assymmetric exchange of the cryptographic key for a group of two or more participants. Unlike the symmetric key exchange, the DH protocol eliminates the direct transfer of the shared secret between the participants so that each participant computes a shared secret with his own private-public key pair. The DH protocol is based on a function

$$A = G^a \bmod P \quad \text{where,}$$

→ A is the user's public key

→ a is the users private key

→ $P = 2Q+1$ is modulus such that at least 2048 bits safe-prime because Q is also prime.

→ G is generator such that G is primitive root modulo P.

If, Given, 2048 bits public prime modulus P and Generator G such that G is primitive root modulo P then.

① → Alice chooses her secret a

② → Alice sends to Bob her public key $A = G^a \bmod P$

③ → Bob chooses his secret b

④ → Bob sends to Alice his public key $B = G^b \bmod P$.

⑤ → Alice computes common secret $s = B^a \bmod P$

⑥ → Bob computes common secret $s = A^b \bmod P$

⑦ Alice and Bob have arrived to the same value that is common secret s:

$$s = A^b \bmod P = G^{ab} \bmod P$$
$$s = B^a \bmod P = G^{ba} \bmod P.$$

Security of Diffie-Hellman :

DH protocol is comparatively more secure. An eavesdropper would see g^a and g^b mod P but can't easily compute a or b, so they can't find $g^{(ab)}$. That's the security of DH protocol. It is computationally hard to solve the discrete log, so the protocol is secure. But in case of man-in-midle attack it has vulnerabilities. Since the initial exchange in not authenticated, an attacker could intercept and replace the public keys with their own. Then they can establish separate keys with each party and decrypt/encrypt messages in between. So, MITM is possible unless there's authentication via certificates or something else.

Besides the protocol has small prime Modulus Weakness. Because the security relies on the difficulty of solving discrete logs for large primes. If p is small, an attacker could compute the discrete log using algorithm like Pohlig-Hellman, which works better when p-1 has small factors. So using a large, safe prime. (where (p-1)/2 is also prime) is important to prevent such attacks. Small p would make the protocol vulnerable to these computations, breaking the security.

Q8 Proof: Let H_1 and H_2 be subgroups of G. We we need to show that $H_1 \cap H_2$ is a subgroup of G.

Since H_1 and H_2 are subgroups, they both contain the identity element e of G.

Therefore, $e \in H_1 \cap H_2$

Let $a, b \in H_1 \cap H_2$. This means $a \in H_1$, $a \in H_2$, $b \in H_1$ & $b \in H_2$. Since, Since H_1 and H_2 are subgroups $ab \in H_1$ and $ab \in H_2$, so, $ab \in H_1 \cap H_2$.

Let $a \in H_1 \cap H_2$. Since $a \in H_1$ and $a \in H_2$ both $a^{-1} \in H_1$ and $a^{-1} \in H_2$.

Therefore $a^{-1} \in H_1 \cap H_2$.

since all three conditions (identity, closure, and inverses) are satisfied, $H_1 \cap H_2$ is a subgroup of G.

Example:

Let $G = Z_6$, the group of integers modulo 6. and Let $H_1 = \{0, 2, 4\}$ and $H_2 = \{0, 3\}$, be subgroups of G, Their intersection is:

$$H_1 \cap H_2 = \{0\}.$$

Since, $\{0\}$, contains the identity, is closed under addition modulo 6, and contains the inverse of its only element 0, it is subgroup of G.

9] The ring Z_n is commutative:

To prove, we need to show that for all $a, b \in \mathbb{Z}_n$ the operation $a \cdot b = b \cdot a$.

In $\mathbb{Z}_n$, multiplication is defined modulo n. Since multiplication of integers is commutative in $\mathbb{Z}$ for any $a, b \in \mathbb{Z}_n$,

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

Thus Z_n is commutative.

Finding Zero Divisions in $\mathbb{Z}_n$:

A zero divisor in a ring is an element $a \neq 0$ such that there exists some $b \neq 0$ where $a \cdot b = 0$ in the ring.

In $\mathbb{Z}_n$, $a \cdot b = 0 \pmod{n}$ if and only if n divides $a \cdot b$. Therefore zero divisors exists if there are non zero elements a and b such that n divides a, b. This happens when a, or b shares a factor with n, so, $\mathbb{Z}_n$ has zero divisors when n is not prime.

Conditions for $\mathbb{Z}_n$ to be a field:

(i) Every non-zero element must have a multiplicative inverse. This occurs only when n is a prime number.

(ii) If n is prime, then for any $a \neq 0 \in \mathbb{Z}_n$, there exists an integer b such that $a \cdot b \equiv 1 \pmod{n}$. i.e a has an inverse.

(iii) If n is not prime, $\mathbb{Z}_n$ has zero divisions therefore it is not a field.

Q10 Vulnerabilities of DES :

The Data Encryption Standard (DES), developed in the 1970s, has several key vulnerabilities that make it insecure for modern use :

(1) Short key length: DES uses a 56-bit key, which is considered too short by today's standards. This makes it vulnerable to brute-force attacks, where an attacker systematically tries all possible keys until the correct one is found.

(2) Brute-force Attacks: A brute-force attack on DES involves trying all 2^56 possible keys. With modern computing power, this is feasible, and in 1997, the EFF's Deep Crack machince was able to break DES in about 56 hours. The short key length thus drastically reduces the security of DES in the face of modern computational power.

(3) Cryptanalytic Attacks: DES is also susceptible to various cryptanalytic attacks, such as differential and linear cryptanalysis. These attacks exploit patterns in the encryption process to recover the key faster than brute-force methods.

Impacts of key Length on Security :

The key length directly impacts the security of a cipher. With a 56-bit key DES only offers 2^{56} possible keys. As computational power has

increased, the feasibility of breaking DES through brute force has become much higher.

The Advanced Encryption Standard (AES) was introduced to address the shortcomings of DES:

(i) key size: AES supports key size of 128, 192, & 256 bits. This increases the key space to 2^{128}, 2^{192}, 2^{256} making brute-force attacks on AES infeasible with current technology

(2) Resistance to Cryptanalysis: AES was designed to be resistant to differential and linear cryptanalysis, which were vulnerabilities of DES The structure of AES is more robust against these types of attacks.

9 11

(i) How the Feistel structure of DES handles differential Cryptanalysis:

The feistel structure in DES helps mitigate differential cryptanalysis by using multiple rounds of encryption with the data being split into two halves. During each round, one half of the data is modified based on the other half and a round key, followed by a swapping operation. This design ensures that even small differences in input propagate through the rounds, making it difficult for attackers to find consitent patterns between plaintext difference and ciphetext differences. The complexity added by the multiple rounds and no-linear transformations in each round increases the difficulty of differential cryptanalysis.

(ii) How AES is more Resistant to cryptanalysis:

(1) Subbytes: Non-linear substitution step that obseures any linear relationship between plaintext and ciphertext.

(2) Shiftrows: Transposition step that diffuses the data and makes it harder for patterns to emerge.

③ mixcolumns: A matrix multiplication step that mixes the data in a way that makes it difficult for attackers to predict how differences propagate.

④ Add Roundkey: XORs the data with a round key adding. furthur complexity.

These operations create significant diffusion and confusion making it harder for differtial cryptanalysis to find predictable differences in ciphertext unlike DES, where the feistal structure might allow some patterns to remain.

9.12 The extended Euclidian Algorithm can calculate the ged(a,b) at the same time it can calculate the values of s and t, in an equation such that, $s \times a + t \times b = ged(a, b)$.

Using this EEA algorithm we can find modular Inverse which is essential in cryptography in RSA key generation.

We can find MI in two ways;

(i) Backward substitution

(ii) Through itteration.

To clearly understand let us take an example to find the multiplicative inverse of 7 upon mod 13.

First Approach: MI of b upon mod n, (Zn)

step 01: Finding of first few coprimes of n such that, coprimes of n > n.

for 13, we find 14, 15, 16, 17, 18- - - - .

step 02: Find such coprime which can be divisible by b.

For, b = 7, we get,

$2 * 7 = 14$, so, we consider 14.

$2 * 7 \equiv 1 \bmod 13$

There for, the multiplicative inverse of 7 is 2.

Second Approach : Directly using the concept of EEA.
Here $gcd(r_1, r_2) = 1$ and, multiplicative inverse is equal the updated t_1. The values of $t_1 = 0, t_2 = 1$ are standard assumptions and, $t = t_1 - t_2 * q$.
q is the quotient, r is the remainder.

MI of 7 in Z_{13} :

q	r_1	r_2	r	t_1	t_2	t
1	13	7	6	0	1	-1
1	7	6	1	1	-1	2
6	6	1	0	-1	2	-13
	1	0		2	-13	

So, multiplicative inverse of 7 in Z_{13} is 2.

Application of Modular Inverse in RSA key Generation :

In RSA, we need to compute the modular inverse while generating the private key.

The key Generation steps are:

1. Choose two large prime numbers p and q.
2. Compute $n = p \times q$
3. Calculate Euler's totient function $\phi(n) = (p-1)(q-1)$.

If there are more than two prime numbers use this following equation:

$$\phi(n) = n * \left(1 - \frac{1}{P_1}\right)\left(1 - \frac{1}{P_2}\right) \cdots \left(1 - \frac{1}{P_i}\right)$$

when $(i \in Z_n)$

4. Choose a public exponent e such that $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$.

5. Compute the private exponent d as:

$$d \equiv e^{-1} \pmod{\phi(n)}$$

So, while computing, we need to find multiplicative inverse of e to find private key, using EEA.

The efficiency of this algorithm is important in cryptography, specialy in large-scale cryptographic systems. Because:

① Fast Computation for large Numbers:
- RSA involves 2048 bit or 4096 bit numbers.
- Brute force methods to find modular inverse would be very slow.
- The EEA runs in O(logn) time, making it efficient even for large scales.

② Used in Digital signature and Encryption:
- Digital Signature Algorithm and Elliptic Curve Cryptograph also use modular inverse.
- Efficient computation ensures secure and fast key gene

③ Security and Practicality
- Efficient modular inversion is critical for real-ti encryption and authentication.

Q13

(i) Why ECB mode is insecure for Encrypting Highly Redundant Data:

In ECB (Electronic Codebook) mode, the plaintext is divided into blocks, and each block is encrypted independently with the same key. If the plaintext is split into Blocks $P_1, P_2, \ldots P_m$, then each block is encrypted as:

$C_i = E(k, P_i)$ where, $E(k, P_i)$ is the encryption function and k is the key.

For highly redundant data, such as repeated patterns or identical blocks, identical plaintext blocks will always produce the same ciphertext blocks. If $P_i = P_j$, then $C_i = C_j$, revealing patterns in the ciphertext. This predictability can be exploited, as an attacker can infer that identical ciphertext blocks correspond to identical plaintext blocks.

Therefore, ECB mode is insecure for encrypting highly redundant data, because it does not hide block-level repetitions, making it vulnerable to cryptanalysis.

(ii) CBC Mode Encryption and Decryption:

CBC Mode Encryption:

In CBC mode, each plaintext block P_i, is XORed with the previous ciphertext block C_{i-1} before encryption.

Mathematically:

$$C_i = E(K, P_i \oplus C_{i-1})$$

where C_0 is the initialization vector (IV) and $E(k, \cdot)$ is the encryption function using key k.

CBC Mode Decryption:

To decrypt the ciphertext, the process is reserved. The decryption of the i-th ciphertext block C_i is:

$$P_i = D(k, C_i) \oplus C_{i-1}$$

where $D(k, \cdot)$ is the decryption function, and C_0 is the IV.

Error Propagation in CBC Decryption:

In CBC mode, if an error occurs in ciphertext block C_i, it affects both the decryption of P_i and P_{i+1}:

↳ The error in C_i causes incorrect decryption of P_i due to the XOR with the corrupted C_{i-1}.

↳ The next block P_{i+1} is also affected by the corrupted C_i.

Thus an error in one block affects the current and next plaintext block, but only one subsequent block is impacted, limiting error propagation.

9|4

LFSRs (Linear feedback shift Registers) are vulnerable to known-plaintext attacks because they produce a keystream using a linear process. In a known plaintext attack, the attacker knows both the plaintext and the ciphertext. Since the key stream is just the XOR of plaintext and ciphertext, the attacker can easily obtain the keystream. Given the linear nature of the LFSR, an attacker can use this key stream to reverse-engineer the internal state of the LFSR and recover the key.

⇒ To protect against this vulnerability, nonlinearity can be added to the system. LFSRs by themselves are linear, so adding a nonlinear function to the process makes it much harder to break using algebric methods.

⇒ One way to introduce nonlinearity is to combine the LFSR, output with a nonlinear function like an XOR of multiple bits. This makes it harder for attackers to exploit the linear relationships in the LFSR.

$$k_i = f(LFSR_i, LFSR_{i+1}, \ldots, LFSR_{i+k})$$

where f is a nonlinear function such as an XOR of multiple bits, or a Boolean function.

Let, two LFSRs L_1 and L_2,

then, $k_t = f(L_1(t), L_2(t))$.

9|15

(i) Shannon's Definition :

A cryptosystem achieves perfect secrecy if knowing the ciphertext provides no additional information about the plaintext. Mathematically, this means :

$$P(M|C) = P(M)$$

for all paintexts $m \in M$ and ciphertext $c \in C$.

This implies that the probability of any plaintext message remains unchanged even after observing the ciphertext.

(ii) Proof of the One-Time Pad Achieves Perfect Secrecy :

The One-Time-Pad (OTP) encryption is defined as :

$$c = M \oplus k$$

where k is a uniformly random key of the same length as m.

To show perfect secrecy, we calculate :

$$P(C|M) = P(k \oplus M | M) = P(k)$$

Since, k is choosen uniformly at random and independent of M, we have :

$$P(M|C) = \frac{P(C|M)P(M)}{P(C)} = P(M)$$

Thus, OTP satisfies shannon's definition of perfect secrecy.

(iii) This proves that knowing C give no additional information about M, acheiving perfect secrecy. The one-time pad requires $|k| \geq |M|$, ensuring a unique key for each message.

(iii) Why Perfect Secrecy is impractical:

(1) key size Requirement: The key must be as long as the message ($|k| \geq |M|$), making storage and distribution highly inefficient.

(2) key Distribution Problem: Securely sharing large, unique keys for each communication session is impractical for large scale systems.

(3) One Time Use: keys must never be reused, otherwise, encryption becomes vulnerable to attacks.

Q16

The first five numbers in the Linear Congruential Generator (LCG) sequence using the recurrence relation:

$$x_{n+1} = (ax_n + c) \bmod m.$$

Given parameters : a, c, m, $x_0 = 7$,

lets assume $a = 5$
$c = 3$
$m = 16$

① $x_1 = (5 \times 7 + 3) \bmod 16 = 6$

② $x_2 = (5 \times 6 + 3) \bmod 16 = 1$

③ $x_3 = (5 \times 1 + 3) \bmod 16 = 8$

④ $x_4 = (5 \times 8 + 3) \bmod 16 = 11$

⑤ $x_5 = (5 \times 11 + 3) \bmod 16 = 10$

So, $x_1, x_2, x_3, x_4, x_5 = \{6, 1, 8, 11, 10\}$.

9.17 Ring in Abstract Algebra:

A ring R is an algebraic structure consisting of a set equipped with two binary operations: adding and multiplication. Ring satisfies the following properties:

(1) Additive closure: If $a, b \in R$, then $a+b \in R$.

(2) Associativity of Addition and Multiplication:
- $(a+b)+c = a+(b+c)$
- $(a \cdot b) \cdot c = a \cdot (b \cdot c)$

(3) Additive Identity: There exists an element $0 \in R$ such that $a+0 = a$ for all $a \in R$.

(4) Additive Inverse: for every $a \in R$, there exists $-a \in R$ such that $a+(-a)=0$.

(5) Distributive Property:
- $a \cdot (b+c) = a \cdot b + a \cdot c$
- $(a+b) \cdot c = a \cdot c + b \cdot c$

(6) Multiplicative Closure: If $a, b \in R$, then $a \cdot b \in R$

A ~~rig~~ ring with unity has also a multiplicative identity $1 \neq 0$, meaning $a \cdot 1 = a$ for all $a \in R$

Example of Rings:

(1) Commutative Ring: A ring is commutative if multiplication is commutative: $a \cdot b = b \cdot a$.

Example: The set of integers $\mathbb{Z}$ with usual addition & multiplication.

② Non-Commutative Ring: If multification does not satisfy $a \cdot b = b \cdot a$, the ring is non-commutative.

Example: The set of 2×2 matrices $M_2(\mathbb{R})$ under matrix addition and multiplication.

The concept of a ring is related to the construction of finite fields. Because, a field is a commutative ring where every nonzero element has a multiplicative inverse. Finite fields ($GF(p^n)$) are built from rings by adding division (except by zero). for example, the integers modulo a prime p from finite field $\mathbb{Z}_p$ used in cryptography.

⇒ Role of Rings in Cryptographic Algorithms like (RSA):

① Modular Arithmetic: RSA operations in the ring $\mathbb{Z}_n$ (integers modulo n), where $n = pq$ (product of two primes)

② Multiplicative Inverse: The RSA decryption exponent is computed using the modular inverse in the ring $\mathbb{Z}_{\varphi(n)}$.

③ Security: The difficulty of factoring n into its prime components (related to the ring structures) ensures RSA's security

Thus ring, provide the foundational algebric framework for secure encryption & decryption in modern cryptography.

Q18] RSA Encryption & Decryption:

Given, $P = 5$, $q = 11$, $M = 2$

To encrypt/Decrypt first we need to follow some steps:

① key Generation:

→ $(P, q) = (5, 11)$.

→ $n = P \times q = 5 \times 11 = 55$.

→ $\phi(n) = (5-1)(11-1) = 40$

→ Choose e = coprime of $\phi(n)$ [55] such as,

$1 < e \leq \phi(n)$ & $\gcd(e, \phi(n) = 1)$

let $e = 3$ where $\gcd(3, 40) = 1$

e, is the public key.

→ choose d such that $e.d \equiv 1 \pmod{40}$

that is d = multiplicative inverse of e upon mod $\phi(n)$

d = multiplicative inverse of 3 upon mod 40.

d, is the private key.

So, & let's find d,

① Few coprimes of 40 > 40 are: 41, 81, 121, ...

② ~~40~~ $3 * 27 = 81$, so consider 81.

Hence, $3 * 27 \equiv 1 \bmod 40$.

Therefore $d = 27$.

So, Public key, $P_k = \{e, n\}$

Private key, $P_r = \{d, P, q\}$.

(2) Encryption :

Given, $m = 2$

So, $c = m^e \bmod n$

$= 2^3 \bmod 55$

$= 8 \bmod 55$

$= 8$

$e = 3$
$n = 55$

$\therefore c = 8$

(3) Deryption:

$c = 8$

$m = c^d \bmod n$

$= 8^{27} \bmod 55$

$d = 27$
$n = 55$

As, 8^{27} is langer we need to compute it using Euiler equation. If $a = 8$,

so, $8^{\phi(n)} \equiv 1 \bmod n$.

$\phi(n) = 40$,

so, $8^{40} \equiv 1 \bmod 55$

since, $27 < 40$, we compute $8^{27} \bmod 55$ directly.

$8^2 = 64 \bmod 55 = 9$

$8^4 = (8^2)^2 \bmod 55 = 9^2 \bmod 55 = 81 \bmod 55 = 26$

$8^8 = (8^4)^2 \bmod 55 = 676 \bmod 55 = 16$

$8^{16} = (8^8)^2 \bmod 55 = 256 \bmod 55 = 36$

$8^{27} = 8^{16} \times 8^8 \times 8^2 \times 8 = (36 \times 16 \times 9 \times 8) \bmod 55$

$\hookrightarrow (576 \bmod 55) = 26$

$\hookrightarrow (26 \times 9 \bmod 55) = 14$

$\hookrightarrow (14 \times 8 \bmod 55) = 2$

So, $m = 2$.

Digital signature :

① key Generation :

→ Prime numbers: $p = 7,\ q = 3$

→ Calculate $g = h^{(p-1)/q} \bmod p$

choose $h = 2$

$\therefore g = 2^{(7-1)/3} \bmod 7$

$g = 2^2 \bmod 7 = 4$

→ private key x such that,

~~$0 < x < q$.~~

~~$0 < x < 3$~~

$x \bmod p < q$

Let $x = 5$

$x \bmod p < q$

$x \neq 0$

→ y = public key.

$y = g^x \bmod p$

$y = 4^5 \bmod 7 = 1024 \bmod 7$

$\therefore y = 2$

keys: public key $\{p, q, g, y\} = \{7, 3, 4, 2\}$

private key $\{p, q, g, x\} = \{7, 3, 4, 5\}$.

③ Signature Generation:

↳ message digest : $H(m) = 3$

↳ Random integer $k = 2$

↳ calculate $r = (g^k \bmod p) \bmod q$

$r = (4^2 \bmod 7) \bmod 3$

$r = (16 \bmod 7) \bmod 3$

$r = 2 \bmod 3 = 2$

↳ calculate $s = k^{-1}(H(m) + x.r) \bmod q$

$= 2^{-1}(3 + 5.2) \bmod 3$

$= 2^{-1}(13) \bmod 3$

$= 2.13 \bmod 3$

$= 26 \bmod 3$

$= 2$

2^{-1} = 2 (multiplicative inverse of 2 upon 3)

∴ signature $\{r, s\} = \{2, 2\}$.

③ Verification: Using public key.

↳ compute $w = s^{-1} \bmod q$

$= 2^{-1} \bmod 3$

∴ $w = 2 \bmod 3 = 2$.

↳ compute $u_1 = H(m) . w \bmod q$

$= 3 . 2 \bmod 3$

$u_1 = 6 \bmod 3 = 0$

↳ compute $u_2 = r . w \bmod q$

$= 2 . 2 \bmod 3$

$u_2 = 4 \bmod 3 = 1$

↳ verify, $v = (g^{u_1} . y^{u_2} \bmod p) \bmod q = (4^0 . 2^1 \bmod 7) \bmod 3$

$v = 2 \bmod 3 = 2$

if $v = r$, signature is valide, so, $v = r = 2$, so valid.

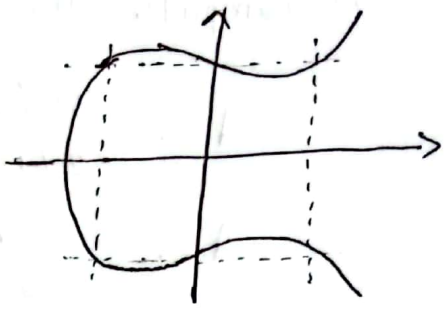
Digital signature ensures the integrity and authenticity of the message. by,

(i) Ensuring Authenticity: The sender signs the message using private key. The receiver verifies the signature using the sender's public key. Since only the sender knows the private key, the signature confirms the senders identity.

(ii) Ensuring Integrity: The message is hashed using a cryptographic hash function (SHA-256). The has is encrypted with the senders private key to generate the signature. The receiver decrypts the signature using the senders public key to obtain the original hash. The receiver also hashes the received message independently. If both hashes match, the message is unaltered otherwise it has been tampered with.

9(2) The Elliptic Curve given:

$$y^2 = x^3 + ax + b \mod p.$$



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

So the equation becomes:

$$y^2 = x^3 + x + 1 \mod 23$$

Check, $P = (3, 10)$ satisfies the equation

$x = 3$, $y = 10$ into the equation.

L.H.S:

$$y^2 = 10^2 = 100 \mod 23 = 8$$

R.H.S:

$$x^3 + x + 1 = (3^3 + 3 + 1) \mod 23$$
$$= (27 + 4) \mod 23$$
$$= 31 \mod 23$$
$$= 8$$

So, L.H.S = R.H.S.

So, point $P(3,10)$ lies on the curve.

(ii) Find the result of doubling the point $P = (3, 10)$ (i.e compute $2P$).

The formula for point doubling is:

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

where, (x_1, y_1) is the $P = (3, 10)$, and $a = 1$.

substitute $x_1 = 3$, $y_1 = 10$ $a = 1$ into the formula.

① Compute the slop λ:

$$\lambda = \frac{3(3^2) + 1}{2(10)} \mod 23$$

$$\lambda = \frac{3(9) + 1}{20} \mod 23$$

$$\lambda = \frac{28}{20} \mod 23$$

Now reduce $\frac{28}{20} \mod 23$. first compute the modular invers of 20 modulo 23. We use the Extended fuclidean Algorithm to find the inverse.

$$20^{-1} \mod 23 = 7$$

Now, compute λ:

$$\lambda = 28 \times 7 \mod 23 = 196 \mod 23 = 12$$

② Now compute new coordinates of 2P:

$$x_2 = \lambda^2 - 2x_1 \mod 23.$$

$$x_2 = 12^2 - 2(3) \mod 23 = 138 \mod 23 = 0$$

$$y_2 = \lambda(x_1 - x_2) - y_1 \mod 23$$

$$y_2 = 3$$

So the doubling of point $P = (3, 10)$ is

$$2P = (0, 3).$$

920]

(i) Signing the hash of a message $H(M) = 8$ using a random nonce $k = 3$.

Given:

→ Elliptic curve equation: $y^2 = x^3 + 7x + 10 \bmod 37$

↳ $y^2 = x^3 + ax + c \bmod p$.

→ Base Point: $G = (2, 5)$

→ Order of the curve: $n = 19$

→ Private key: $d = 9$

→ Random nonce: $k = 3$

→ message hash: $H(M) = 8$.

① Compute kG

$k = 3, \quad G = (2, 5)$

$\Rightarrow 3G = G + G + G$

→ first $2G$: $\lambda = \frac{3x_1^2 + a}{2y_1^2} \bmod p$

where, $(x_1, y_1) = (2, 5)$, $a = 7$, $p = 37$.

$$\lambda = \frac{3(2^2) + 7}{5} \bmod 37$$

$$\lambda = \frac{19}{10} \bmod 37$$

$\therefore 10^{-1} \bmod 37 = 10$

$\therefore \lambda = (19 \times 10) \bmod 37 = 5$

Now; $2G$:

$x_2 = \lambda^2 - 2x_1 \bmod 37 = 21$

$y_2 = \lambda(x_1 - x_2) - y_1 \bmod 37 = 11$

Thus $2G = (21, 11)$

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

Formula for point addition is:

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} \mod p$$

① $\lambda = \frac{7-10}{9-3} \mod 23$

$\lambda = \frac{-3}{6} \mod 23$

$\because 6^{-1} \mod 23 = 4$

$\lambda = -3 \times 4 \mod 23 = 11$

② $P + Q$:

$x_3 = \lambda^2 - x_1 - x_2 \mod 23$

$x_3 = 11^2 - 3 - 9 \mod 23$

$x_3 = 17$

$y_3 = \lambda (x_1 - x_3) - y_1 \mod 23$

$y_3 = -164 \mod 23 = 20$

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

Now compute $3G = 2G + G$

Using the point addition formula

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} \bmod p$$

$(x_1, y_1) = (2,5)$ & $(x_2, y_2) = (21,11)$

$$\lambda = \frac{11-5}{21-2} \bmod 37$$

$$\lambda = \frac{6}{19} \bmod 37$$

$19^{-1} \bmod 37 = 17$

$\therefore \lambda = 6 \times 17 \bmod 37 = 28$

Now $3G$:

$x_3 = \lambda^2 - x_1 - x_2 \bmod 37 = 21$

$y_3 = \lambda(x_1 - x_3) - y_1 \bmod 37 = 18$

Thus $3G = (21,18)$

(2) Compute signature components r & s.

$$r = x_3 \bmod n$$

$$s = k^{-1}(H(M) + rd) \bmod n$$

$H(M) = 8$, $d = 9$, $k = 3$, $n = 19$.

$\therefore r = 21 \bmod 19 = 2$

$s = 3^{-1} \bmod 19 = 13$

$\therefore s = 13(8 + 2 \times 9) \bmod 19 = 15$

$\therefore (r, s) = (2, 15)$

Verify the signature $(r, s) = (2, 15)$

① check $1 \leq r, s < n$:

→ $r = 2$, $s = 15$ both valid since $n = 19$.

② compute $w = s^{-1} \bmod 19$:

$$w = 15^{-1} \bmod 19 = 14$$

③ Compute u_1 and u_2:

$$u_1 = H(M) \cdot w \bmod 19 = 8 \times 14 \bmod 19$$
$$u_1 = 17$$

$$u_2 = r \cdot w \bmod 19$$
$$= (2 \times 14) \bmod 19$$
$$= 9$$

④ Compute: $u_1 G + u_2 Q$:

$$17G + 9Q$$
$$17G + 9 * 9G$$
$$(17G + 81G) \bmod 19$$
$$= 17G + 5G$$
$$= (21, 26) + \alpha$$

∴

⑤ verify $r \equiv x\text{-coordinate} \bmod n$:

$$2 \equiv 21 \bmod 19$$

$$2 = 2$$

So, valid.

9.2.1 key Properties of Cryptographic Hash function (SHA-256)

(i) Essential Characteristics of a Secure Hash Function

① Pre-image Resistance: Given a hash output h, it should be infeasible to find an input x such that $H(x)=h$.

② Second Pre-image Resistance: Given an input x, it should be hard to find another input x', where $H(x) = H(x')$

③ Collision Resistance: It should be computationally infeasible to find two distinct inputs x and y such that $H(x) = H(y)$.

④ Deterministic & Fast Computation: The same input always produces the same hash efficiently.

⑤ Avalanche Effect: A small change in input should produce a drastically different output.

(ii) Impact of Hash Length on Security:

- The hash length determines resistance against brute-force attacks.
- In SHA-256 the 256 bit output provides, 2^{256} possible hashes, making collisions highly improbable.
- Longer Hashes (eg SHA-512) offer even stronger security but may require more computational resources.

(iii) Real World Application of SHA:

① Digital Signatures :- Used in RSA, ECDSA, and other cryptographic protocols to ensure message integrity and authentication.

② Blockchain & Cryptocurrencies :- SHA-256 secures transactions in Bitcoin and blockchain systems, ensuring data integrity and proof-of-work consensus.

③ Password Hashing :- Hashes stored passwords securely (though salted hashes are preferred for added security.

④ Data Integrity Verification: Used in integrity checks to detect tampering.

Q22 Galois Fields / Finite fields:

① Galois Fields : GF(P) and GF(2^n)

↳ GF(P): A finite field with p elements, where p is a prime number. Arithmetic follows modulo P, ensuring closure, associativity, commutativity, distributivity, identity elements, and inverses.

↳ GF(2^n) : A finite field with 2^n elements, used in binary arithmetic. Operations (addition and multiplication) are performed using polynomial arithmetic modulo an irreducible polynomial of degree n.

② Galois Fields in Cryptographic Primitives:

→ Elliptic Curve Cryptography (ECC): ECC operates over GF(p) or $GF(2^n)$, using field arithmetic to define secure elliptic curve groups for encryption, key exchange, and digital signatures.

→ AES (Advanced Encryption Standard): AES operates in $GF(2^8)$ where byte wise transformations such as the SubByte step use inversion in $GF(2^8)$ ensuring non-linearity and strong security properties.

③ Importance of Field Arithmetic:

(i) Esures Security: Provides structures for secure key exchange, encryption and digital signatures.

(ii) Efficient Computations: Field arithmetic enables fast modular operations, crucial for cryptographic performance.

(iii) Error Resistance: Used in error correction codes.

P23 Lattice-Based Cryptography & its Post-Quantum securit

① Shortest Vector Problem and Its Role in Security
- Given a lattice (a discrete set of points in n-dimensional space), SVP seeks the shortest nonzero vector in the lattice.
- Security Basis: The hardness of SVP underlies lattice based cryptographic schemes, as no efficient classical or quantum algorithm can solve SVP in high dimensions, making it resistant to quantum attacks.

② Comparision with Tradifional Cryptographic Assumptions:
- RSA & ECC: Their security relies on the integer factorization and discrete logarithmic problems, which Shor's Algorithm can efficiently solve, breaking these cryptosystems.
- Lattice Based Cryptography: Based on hard lattice problems like SVP and learning with Errors, which remain intractable even for quantum computers; making them strong post-quantum candidates.

③ Comparision with Quantum Cryptography:-

Lattice based cryptograpy uses mathematical problems in lattices to build clasical but quantum resistant cryptosystems. Quantum cryptography uses quantum mechanics (e.g. Quantum key Distribution QKD) to achieve unconditional security based on quantum principles & rather than computational assumptions.

9.24 Proof of Maximum Period of LFSR keystream

A Linear feedback Shift Register (LFSR) over GF(2) is defined by the recurrence relation:

$$k_t = c_1 k_{t-1} \oplus c_2 k_{t-2} \oplus \cdots \oplus c_m k_{t-m}$$

where the sequence is generated modulo 2.

The sequence generated by an LFSR is periodic, meaning it eventually repeates after some cycles. The period depends on the characteristic polynomial:

$$P(x) = x^m - c_1 x^{m-1} - c_2 x^{m-2} - \cdots - c_m$$

If $P(x)$ is a primitive polynomial of degree m over GF(2), then the LFSR generates all nonzero m-bit sequences before repeating.

The number of possible nonzero states in an m-bit LFSR is $2^m - 1$ (excluding the all-zero state, which causes the sequence to terminate)

Since a primitive polynomial ensures that every nonzero state appears exactly once in the sequence the maximum period of the keystream is $2^m - 1$

This follows from the fact that a primitive polynomial generates a maximum-length sequence (MLS); cycling through all $2^m - 1$ possible states before repeating.

Thus, if the characteristic polynomial of the LFSR is primitive, the keystream acheives the maximum period of $2^m - 1$.

925 LWE- Based Signature Scheme:

① key Generation :-

→ Private key (s_k) : A small seret vector s sampled from an error distribution.

→ Public key :- A matrix A and vector $t = As + e$, where e is a small error vector.

② Signing :-

→ Hash the message M to produce a challenge c.

→ Generate a random masking vector y.

→ Compute $w = Ay$..

→ Derive $c = H(M \| w)$ (hash function)

→ Compute response $z = y + cs$.

→ Apply rejection sampling to ensure z dose not leak s.

→ signature : (z, c)

③ verification :-

→ Recompute $w' = Az - ct$.

→ check $c = H(M \| w')$ and z has small norm.

Example of message Signing with LME.

→ Input message M, $P_K = (A, t)$, $S_K = S$.

→ Steps :

① Hash M to get c.

② sample y from a Gaussian distribution

③ compute $w = Ay$.

④ Generate $c = H(M \| w)$.

⑤ Compute $z = y + cs$.

⑥ Output signature (z, c).

LWE problem in ensuring Security :-

① The LWE problem ensures that an attacker can not efficiently recover s_k from P_k.

② Finding a short z without a trapdoor is as hard as solving the Shortest Vector Problem in high dimensional lattices, which is believed to be quantum-secure.

③ Forging a signature requires solving $t = As + e$ for s, which is computationally hard under LWE.

④ Rejection sampling ensures signatures do not Leak s, and the hash function binds M to c, preventing replay attacks.