Q1: With the advent of quantum computing, 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 propose possible post-quantum cryptographic algorithms that could replace RSA and ECC. How do these algorithms resist quantum cryptanalysis?

Solution:

Implications of quantum computing on Cryptographic Protocols:

Quantum computing, with its powerful computational abilities, poses a significant threat to the security of many widely used cryptographic protocols. Classical cryptographic systems such as {RSA (Rivest-shamir-Adleman) and ECC (Elliptic Curve Cryptography) rely on the computational difficulty of certain problems. Quantum computers can efficiently solve these problems using algorithms like Shor's Algorithm.

Shor's algorithm and Cryptographic Vulnerabilities:

RSA Vulnerability: RSA encryption relies on the fact that factoring the product of two large primes is computationally difficult for classical computers. However, shor's algorithm allows a quantum computer to factor large numbers in polynomial time, making RSA vulnerable.

2. ECC vulnerability: ECC is based on the hardness of the elliptic curve discrete logarithm problem (ECDLP). Like RSA, this problem can also be solved efficiently by a quantum computer using Shor's algorithm, breaking ECC's security model.

Current encryption schemes based on RSA and ECC will no longer be secure in a world with powerful quantum computers. ~~These~~ This could have wide-reaching implications for data privacy, digital signatures, secure communication, and many other applications that depend on public key cryptography.

☞ Propose post-quantum cryptographic Algorithms that could replace RSA and ECC:

1. Lattice-Based Cryptography:

Lattice-based cryptographic algorithms rely on the hardness of problems in lattice theory, such as the Shortest Vector Problem (SVP) and Learning with Errors (LWE). These problems are believed to be resistant to quantum attacks.

① NTRU: NTRU is a public-key encryption algorithm based on lattice problems. It is efficient and has been proposed as a potential replacement for RSA and ECC.

ii) FrodoKEM: A key exchange protocol based on the hardness of the learning with errors problem.

B. Code-Based Cryptography:

- Based on the difficulty of decoding random linear codes.

→ McEliece: The McEliece cryptosystem uses the hardness of decoding a random linear code to encrypt data. Though it has a relatively large public key size, it is considered secure against quantum attacks.

→ Rainbow: A multivariate public key signature scheme based on error-correcting codes, specifically designed to resist quantum attacks.

C. Hash-Based Cryptography:

Hash-based cryptographic algorithms leverage hash functions, which are believed to be quantum-resistant since the best-known quantum algorithm (Grover's algorithm) can only offer a quadratic speedup for finding hash collisions.

- XMSS (extended Merkle signature scheme):
  - Uses a Merkle tree to provide digital signatures.
  - designed to be secure against quantum attacks and is already standardised in certain areas.
- SPHINCS+: - builds on the Merkle tree. - designed for high security and efficiency.

D. Multivariate Polynomial Cryptography (MQ)

- Based on the difficulty of solving systems of multivariate quadratic equations.
- promising alternative to traditional cryptosystems

• GeMSS: - secure post-quantum signature algorithm
- based on multivariate polynomials.
- proposed for use in PQC standardization.

E. Isogeny-Based Cryptography:

- relies on the difficulty of finding isogenies between elliptic curves.
- a problem that is believed to be difficult for both classical and quantum computers.

• SIDH (Supersingular Isogeny Diffie-Hellman):

- hardness of finding isogenies between supersingular elliptic curves.
- a quantum-resistant alternative to traditional ECC-based protocols.

Quantum Resistance:

→ Lattice-Based, problems like SVP and LWE are hard for both classical and quantum computers, offering robust quantum resistance.

→ Code-Based; Decoding random linear codes remains hard even for quantum computers, ensuring that systems like McEliece are resistant to quantum attacks.

→ Hash-Based, hash functions are resistant to quantum attacks with Grover's algorithm only providing a quadratic speedup, making hash-based signature schemes viable for post quantum cryptography.

- Multivariate Polynomial : The MQ problem is exponentially hard for both classical and quantum algorithms, providing strong resistance to quantum cryptanalysis.

- Isogeny-Based : The problem of finding isogenies between supersingular elliptic curve has no known quantum algorithm that provides a polynomial-time solution, ensuring security.

Q2: 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 modulus operation to constrain the output within a desired range.

Solution:

Design Outline:

1. Timestamp-based seed: We'll use the current timestamp (time.time()) as one of the inputs to generate randomness. The timestamp will be taken in seconds or milliseconds to ensure rapid changes over time.

2. Process ID (PID): The process ID will be a dynamic input for added entropy. The PID changes with each running process, so it can contribute to randomness specific to the running environment.

3. Modulus operation: To constrain the result to a desired range, we'll use a modulus operation to limit the output to specific range (betn 0 and n-1)

Python Code implementation:

```
import time
import os

class NovelPRNG:

    def __init__(self, modulus):
        """ Initializes the PRNG with a given modulus.
        :param modulus: The upper bound for the generated random number (exclusive).
        """
        self.modulus = modulus

    def _generate_seed(self):
        """
        Generates a seed by combining the current timestamp and process ID.
        This ensures a dynamic and unique seed for each run.
        :return: An integer seed.
        """
        timestamp = int(time.time() * 1000)
        pid = os.getpid()
        seed = timestamp ^ pid
        return seed

    def next(self):
        """
        Generates the next pseudo-random number within the specified range.
        :return: A pseudo-random integer within the range [0, modulus).
        """
        seed = self._generate_seed()
        a = 1664525
        c = 1013904223
        m = 2**32
```

[$X_n = (a * X_{(n-1)} + c) \% m$]

```
random_value = (a * seed + c) % m
return random_value % self.modulus
if __name__ == "__main__":
    prng = NovelPRNG(modulus=100)
    for _ in range(10):
        Print(prng.next())
```

Output:

```
68
45
27
33
57
9
72
61
59
94
```

Q3. Compare traditional ciphers (such as Caesar cipher, Vigenere cipher, and playfair cipher) with modern symmetric ciphers like AES and DES. Discuss the strengths and weakness of each type of cipher, including key length, encryption/decryption speed, and security against modern cryptanalysis techniques.

→ Traditional ciphers like the Caesar, Vigenere, and playfair ciphers are educational and have historical significance, but they are vulnerable to modern cryptanalysis and cannot be used for secure communications.

→ Modern ciphers like DES and AES are much stronger. DES, while historically important, is no longer secure and has been replaced by AES, which is widely used for secure communication today.

→ AES is the gold standard for symmetric encryption due to its balance of security, efficiency and key flexibility, making it suitable for a broad range of applications.

IT-23602

Cipher	Key Length	Encryption speed	Security	Strengths (encryption/decryption)	Weaknesses	Encryption technique	Use case
Caesar Cipher	3-25 (fixed shift)	Very fast	Very weak; easy to break with brute-force and frequency analysis	Extremely simple, fast	Easily broken by brute-force, frequency analysis.	substitution cipher, each letter of the plaintext is shifted by fixed no. of positions in the alphabet	not secure for modern communication
Vigenere Cipher	Variable (key word)	Slow	Vulnerable to frequency analysis, kasis kasiski attack.	Harder to break than caesar cipher.	Vulnerable if keyword is short or repeated	• Polyalphabetic substitution cipher	Better than caesar cipher, still not suitable for modern communication system
Playfair Cipher	25 (5 x 5 grid)	Moderate	Relatively weak; vulnerable to frequency analysis.	Better than Caesar and Vigenere.	Vulnerable to cryptanalysis.	digraph substitution cipher	Better than simple substitution ciphers but still not strong enough for modern security needs
DES	56 bits	Fast (especially in hardware)	Insecure by modern standards (Vulnerable to brute force)	Fast, widely used historically	Too weak due to short key length.	A block cipher (substitution and permutation)	de-precated and replaced by strong algorithms like AES. though it remains a historical reference in cryptography
AES	128, 192 or 256 bits	Fast (especially with hardware)	Very strong; resistant to modern attacks	Extremely secure, efficient, flexible	Key management is crucial. Some side-channel attacks possible.	A block cipher	Modern symmetric encryption, widely used for everything from encrypting files to securing internet traffic

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 stabiliser of the subset $\{1,2\}$.

Solution: The symmetric group S_4 consists of all permutations of the set $\{1,2,3,4\}$. We define an action of S_4 on the set of 2 element subsets of $\{1,2,3,4\}$, denoted as:

$$X = \{\{1,2\},\{1,3\},\{1,4\},\{2,3\},\{2,4\},\{3,4\}\}$$

For any permutation $\sigma \in S_4$ and any subset $\{a,b\} \in X$,

$$\sigma \cdot \{a,b\} = \{\sigma(a), \sigma(b)\}$$

since σ is a bijection of $\{1,2,3,4\}$.

→ The action satisfies the identity property: The identity permutation $e \in S_4$ satisfies $e \cdot \{a,b\} = \{a,b\}$

→ The action is compatible with composition:

If $\tau, \sigma \in S_4$ then

$$(\tau\sigma)\cdot\{a,b\} = \tau\cdot(\sigma\cdot\{a,b\}) = \tau\cdot\{\sigma(a),\sigma(b)\}$$
$$= \{\tau(\sigma(a)), \tau(\sigma(b))\}$$
$$= (\tau\sigma)\cdot\{a,b\}.$$

Thus, the action is well defined.

□ Orbit of $\{1,2\}$:

$$Orb(\{1,2\}) = \{\{1,2\},\{1,3\},\{1,4\},\{2,3\},\{2,4\},\{3,4\}\}.$$

Since there are $\binom{4}{2} = 6$ subsets in X, the size of the orbit is :

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

□ Stabilizer of $\{1,2\}$:

The stabilizer of $\{1,2\}$ in S_4 consists of all permutations $\sigma \in S_4$. $\sigma \cdot \{1,2\} = \{1,2\}$.

σ must send 1 to either 1 or 2, and 2 to either 1 or 2, but cannot swap with any of 3 or 4.

→ Stabilizer : permutations that fix $\{1,2\}$ while freely permuting $\{3,4\}$. These are —

→ The identity permutation e.

→ The transposition $(1,2)$, while swaps 1 and 2.

→ The permutations of $\{3,4\}$, while are $(3,4)$, the identity on $\{3,4\}$ and the two transpositions involving 3 and 4.

∴ Total number of permutations :

$$|Stab(\{1,2\})| = 2! \times 2! = 4$$

Verification using the Orbit-stabiliser Theorem:

$$|S_4| = |\text{Orb}(\{1,2\})| \times |\text{Stab}(\{1,2\})|.$$

Since $|S_4| = 4! = 24$ and we computed $|\text{Orb}(\{1,2\})| = 6$

$$24 = 6 \times 4.$$

So, the action is well-defined
→ The orbit of $\{1,2\}$ has size 6.
→ The stabiliser of $\{1,2\}$ has size 4.

6D: 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.

Solution:

Construction of $GF(2^2)$:

The field $GF(2^2)$ is constructed using the irreducible Polynomial:

$$x^2+x+1.$$

over GF(2). This means the $GF(2^2)$ consists of elements of the form.

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

Where α is a root of $x^2+x+1=0$. that is

$$\alpha^2 = \alpha+1.$$

Using this relation, we can compute powers of α:

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

Thus, α has order 3, meaning it satisfies:

$$\alpha^3 = 1.$$

The nonzero elements of GF(2^r) are:

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

i) Show that GF(2^r) Forms a Group under Multiplication:

To be a group under multiplication, the set of non-zero elements $\{1, \alpha, \alpha+1\}$ must satisfy the group:

1. Closure: $\alpha \cdot \alpha = \alpha^2 = \alpha+1$

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

$$(\alpha+1)(\alpha+1) = \alpha^2 + 2\alpha + 1 = \alpha^2 + 1$$

(Since in GF(2), $2\alpha = 0$).

Using $\alpha^2 = \alpha+1$, we get:

$$(\alpha+1)^2 = (\alpha+1) + 1 = \alpha$$

Since all results belong to the set $\{1, \alpha, \alpha+1\}$, closure holds.

2. Associativity: Since field multiplication is associative, it holds automatically.

3. Identity element: $1 \cdot x = x \cdot 1 = x$ for all x.

4. Inverse: $1^{-1} = 1$

α^{-1} should satisfy $\alpha \cdot x = 1$.

Since $\alpha^2 = \alpha + 1$, we check:

$$\alpha(\alpha+1) = 1$$

So, $\alpha^{-1} = \alpha + 1$.

- $(\alpha+1)^{-1}$ should satisfy

$$(\alpha+1)\cdot x = 1 \quad \text{. using } (\alpha+1)^2=\alpha,$$

$$(\alpha+1)\cdot\alpha = 1$$

So, $(\alpha+1)^{-1}=\alpha$

every element has an inverse, so the group property holds.

Thus, the set $\{1, \alpha, \alpha+1\}$ forms a group under multiplication.

ii) Verify wether $\{1,\alpha,\alpha+1\}$ is cyclic:

A group is cyclic if it has an element that generates all nonzero elements through repeated multiplication.

- $\alpha^3=1$ and power cycle through $\{1,\alpha,\alpha+1\}$.

$$\alpha^1=\alpha \quad , \alpha^2=\alpha+1 \quad , \alpha^3=1 .$$

α generates all elements, proving that, $\{1,\alpha,\alpha+1\}$ is cyclic.

→ Similarly, checking $\alpha+1$:

$$(\alpha+1)^1=\alpha+1,$$
$$(\alpha+1)^2=\alpha$$
$$(\alpha+1)^3=1 .$$

(α or $\alpha+1$) that generates all elements, the group is cyclic

- The nonzero elements of $GF(2^2)$ form a multiplicative group of order 3.
- This group is cyclic with α or $\alpha+1$ as a generator.

Thus $GF(2^2)^*=\{1,\alpha,\alpha+1\}$ is a cyclic group of order 3.

IT-23602

Q6: Let $GL(2,R)$ be the general linear group of 2×2 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.

$$GL(2,R)=\{A \in M_2(R) \mid \det(A)\neq 0\}.$$

it is a group under matrix multiplication.

- the set of scalar matrices:

A scalar matrix is a multiple of the identity matrix:

$$S=\{cI \mid c\in R^{\times}\}=\left\{\begin{bmatrix} c & 0 \\ 0 & c \end{bmatrix} \mid c\neq 0\right\}.$$

The set of scalar matrices forms a subgroup of $GL(2,R)$ because:

1. Closure: If $A=cI$ and $B=dI$, then $AB=(cd)I$, which is still a scalar matrix.
2. Identity: The identity matrix $I=1I$ is included.
3. Inverses: The inverse of cI is $(cI)^{-1}=(c^{-1})I$, which is again a scalar matrix.
4. Associativity follows from general matrix multiplication.

Since, every scalar matrix is invertible, S is subgroup of $GL(2,R)$.

Normality of S in GL(2,R).

A subgroup N of a group G is normal if for every $g \in G$ and every $n \in N$, the conjugate gng^{-1} is also N. That is.

$$A(cI)A^{-1} \in S, \quad \forall A \in GL(2,R), \quad \forall cI \in S.$$

We compute.

$$A(cI)A^{-1} = c(AIA^{-1}) = c(AA^{-1}) = cI.$$

Since cI is still a scalar matrix, this shows that S is normal in GL(2,R).

Thus $S \trianglelefteq GL(2,R)$.

Construct the factor group GL(2,R)/S:

The factor group GL(2,R)/S consists of cosets of S, which are equivalence classes of matrices differing by a scalar multiple of the identity. That is, two matrices A, B belong to the same coset if:

$$AB^{-1} \in S.$$

This means that A and B are scalar multiples of each other.

Each coset represents a class of matrices that are equivalent up to a scalar multiple. The set of cosets can be identified with the projective general linear group: $PGL(2,R) = GL(2,R)/S$.

☐ Structure of $GL(2,R)/S$:

The quotient group $GL(2,R)/S$ represents linear transformations up to scalar multiples. This corresponds to transformations of the real projective line RP^1, which consists of lines through the origin in R^2.

* $PGL(2,R)$ acts as the group of Mobius transformations :

$f(z) = \frac{az+b}{cz+d}$, where $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ represents a coset in $GL(2,R)/S$; describs the projective transformations of the real line.

Q.7: Explain the Diffie-Hellman key exchange protocol and its application in secure communication. Discuss the security of the Diffie-Hellman protocol 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?

Diffie-Hellman Key Exchange protocol:

- allows two parties to securely establish a shared secret over an insecure channel. This shared secret can then be used for encrypting futher communication.

Steps of the protocol:

1. Public Parameters selection:
   - A large prime number p (the modulus),
   - A primitive root (generator) g modulo p.

   These parameters are publicly known.

2. Key Exchange:
   - Alice chooses a private key a (a secret random integer) and computes her public key:

     $$A = g^a \bmod p.$$

     she sends A to Bob.
   - Bob chooses a private key b and computes his public key:

     $$B = g^b \bmod p$$

     He sends B to Alice.

3. Shared secret Computation:
   - Alice computes the shared secret using Bob's Public key:

     $$S = B^a \bmod p = (g^b)^a \bmod p.$$

   - Bob computes the same shared secret using Alice's Public key:

     $$S = A^b \bmod p = (g^a)^b \bmod p.$$

• since $(g^b)^a = (g^a)^b$, both compute the same secret key S.

• This secret S can now be used to drive encryption keys for secure communication.

Security of Diffie-Hellman:

The security of the Diffie-Hellman key exchange relies on the difficulty of the Discrete Logarithm Problem (DLP).

Discrete Logarithm Problem (DLP):

Given g, p and $g^a \bmod p$, it is computationally infeasible to determine a when p is large. This problem ensures that even if an attacker sees the exchanged public values A and B, they cannot compute the shared secret S.

Common Attacks on Diffie-Hellman:

1. Man-in-the-Middle (MITM) Attack:

• If an attacker (Eve) intercepts the communication, she can perform separate DH key exchanges with Alice and Bob.

• Eve can establish different shared secrets with each party and act as a relay, decrypting and re-encrypting messages without Alice and Bob realizing it.

• Countermeasure: Authenticated Diffie-Hellman (e.g. with digital signatures or certificates) ensures parties and communicating with the correct entity.

② Brute Force and Precomputed Attacks:

• If p is small, an attack er can precompute logarithms or use brute force to find the secret key.

• Countermeasure: Use large prime numbers (at least ie 2048-bit modulelus).

③ Log jam Attack:

• Exploits precomputed values for common primes to speed up discrete logarithm computation.

• Countermeasure: Use unique, large and strong primes for each session.

④ Side-channel Attacks:

• Attackers may exploit timing or power consumption patterns.

• Countermeasure: Implement constant-time operations to prevent leakage.

☐ Impact of Using a small Prime Modules:

If p is not sufficiently large, the following risks arise:

– Faster Brute Force Attacks: small primes allow attackers to compute discrete logarithms efficiently using brute force or number field sieve methods.

- **Precomputed Attacks:**
  - If widely used small primes are chosen, attackers can precompute tables to break the key exchange quickly.
- **Weakened Logarithm security:**
  - Cryptanalytic advances (index calculus) make solving the discrete logarithm problem easier for small primes.

Prime Sizes:

→ 1024 bit primes: No longer considered secure against state-level adversaries.

→ 2048-bit primes: currently secure for most applications.

→ 4096 bit primes: Used for long-term security.

For enhanced security, Elliptic curve Diffie-Hellman (ECDH) is often preferred, as it provides the same security with smaller key sizes.

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.

Let, G be a group and let, H and K be two subgroups of G. We need to prove that the intersection $H \cap K$ is also a subgroup of G.

Step 1: Non-emptiness:

Since both H and K are subgroups of G, they both contain the identity element e of G. Thus, $e \in H \cap K$, meaning that the intersection is non-empty.

Step 2: Closure under Multiplication:

Let, $a, b \in H \cap K$, since a and b are in both H and K and since both H and K are subgroups, their product must also be in both:

$$ab \in H, \quad ab \in K.$$

Thus, $ab \in H \cap K$, showing closure under multiplication.

Step-3: Inverses:

Let $a \in H \cap K$, since a is in both H and K, and both are subgroups, they must contain a^{-1} as well:

$$a^{-1} \in H, \quad a^{-1} \in K.$$

Thus, $a^{-1} \in H \cap K$, showing closure under inverses

Since, $H \cap K$ is non-empty, closed under multiplication, and closed under inverses, it satisfies the subgroup criteria. Providing that $H \cap K$ is a subgroup of G.

Example: Consider the group $G = \mathbb{Z}$ under addition and the subgroups:

$H = 2\mathbb{Z} = \{\ldots, -4, -2, 0, 2, 4, \ldots\}$ (even integers)

$K = 3\mathbb{Z} = \{\ldots, -3, 0, 3, 6, 9, \ldots\}$ (multiples of 3)

Their intersection is:

$H \cap K = \{x \in \mathbb{Z} \mid x$ is both even and a multiple of 3$\}$

Since $6\mathbb{Z}$ is also a subgroup of $\mathbb{Z}$, $= 6\mathbb{Z}$.

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

Prove: Z_n is Commutative:

The ring Z_n (also called the ring of integers modulo n) consists of the elements $\{0, 1, 2, \ldots, n-1\}$ with addition and multiplication defined modulo n.

1. addition:

For any $a, b \in Z_n$,

$$a + b \equiv b + a \mod n.$$

addition Z and Z_n are commutatitive.

2. Multiplication:

for any $a, b \in Z_n$,

$$a \cdot b \equiv b \cdot a \mod n.$$

Since, Multiplication Z and Z_n is also commutative.

Thus Z_n is a commutative ring.

Zero Divisors in Z_n

A zero divisor in Z_n is a nonzero element a such that there exists a nonzero b where:

$$a \cdot b \equiv 0 \mod n.$$

This happens when a and b are not relatively prime to n, meaning that $\gcd(a, n) > 1$.

Example: Zero Divisors in Z_6:

- 2 and 3 in Z_6
- $2 \cdot 3 = 6 \equiv 0 \mod 6$, so 2 and 3 are zero divisors.

Thus, Z_n has zero divisors whenever n is composite.

Conditions for Z_n to be a Field:

A ring is a field if every nonzero element has a multiplicative inverse.

→ In Z_n, an element a has an inverse if there exists b such that:

$$a \cdot b \equiv 1 \mod n.$$

- This happens if and only if $\gcd(a, n) = 1$, meaning a is coprime to n.

→ When does every nonzero element have an inverse?

- If n is prime, every $a \in Z_n \setminus \{0\}$ satisfies $\gcd(a, n) = 1$, meaning every element has an inverse
- If n is composite, there exist zero divisors (elements without inverses), so Z_n is not a field.

Thus, Z_n is a field if and only if n is prime.

Q10. Explain the vulnerabilities of the DES (Data encryption standard) cipher and why it is considered insecure for modern use. ~~Determine~~ 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?

Vulnerabilities of the DES cipher and why it is insecure today?

It was widly used for decades, it is now considered insecure due to several vulnerabilities:

1. Short key length (56 bit key)

- 56 bit key, which is too small for modern security standards.
- A brute-force attack, which involves trying all possible keys, is feasible with today's computational power.
- the Electronic Frontier Foundation (EFF) built a machine called Deep Crack that broke DES encryption in less than a day.

2. Vulnerability to Brute-Force Attacks:

- Since DES has only 2^{56} possible keys, modern hardware can exhaustively search the entire keyspace.
- Today, a distributed or specialised hardware attack can break DES in a matter of minites to hours.

3. Susceptibility of Cryptanalysis Attacks:

- Linear Cryptanalysis: 2^{43} → plaintexts to break DES.
- Differential Crytanalysis: analyses ciphertext Patterns to recover the key, through DES was designed to resist it.

* Meet-in-the Middle attack:

reduce the complexity of a brute-force attack on double DES from 2^{112} to 2^{57}, making double encryption ineffective.

4. Small Block Size (64 bits):

→ encrypt data - 64 bit blocks, which makes it vulnerable to birthday attacks when encrypting large amounts of data.

→ This increases the likehood of collisions, reducing security.

II Role of Brute-Force Attacks in Breaking DES:

→ Small Keyspace by trying 2^{56} possible keys.

- The main reason for its deprecation, as increasing computational power made breaking DES trival.

- 1998 – Deep Crack broke DES in 22 hours.
- 2017 - A powerful cluster of GPUs could break DES in less than a minute.
- Today - Cloud - based services or distributed networks can crack DES almost instantly.

Since, brute-force attacks are inevitable when key sizes are too small, DES is no longer secure for modern encryption needs.

IT-23602

How AES overcame DES's shortcomings:

1. **Larger key sizes:**
   - 128 bit, 192 bit and 256 bit keys, making brute-force attacks infeasible.
   - 128-bit key has 2^{128} possible combinations, making brute-force attacks astronomically difficult.

2. **Larger Block size:**
   - AES uses a 128-bit block size, which significantly reduces the risk of birthday attacks and collisions.

3. **Stronger security Against Cryptanalysis:**
   - Unlike DES, AES is designed to resist:
     - Differential and linear cryptanalysis.
     - Mut-in-the middle and related-key attacks.
   - No practical attacks exist against AES when implemented correctly.

4. **Efficiency in Software and Hardware:**
   - AES is faster and more efficient than DES, especially when optimised for modern CPUs and hardware.

Thus AES has become the global encryption Standard, ~~efficient~~ effectively replacing DES for secure applications.

Q11: Differential cryptanalysis is a widely known attack against block ciphers,

i. Explain how the Feistel 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 ?

1) Feistel Structure and Resistance in DES :

The fai-faistel structure of DES helps mitigate differential cryptanalysis in the following ways:

1. Avalanche effects:

- The feistel structure ensures that a small change in the ~~output~~ input propagates throughout the cipher.
- After just a few rounds, a single-bit difference in plaintext ~~effects~~ multiple bits in ciphertext.

2. S-Box Design for Non-Linearity:

- DES uses eight S-Boxes (substitution boxes) that introduce non-linearity into the encryption.
- designed to resist differential cryptanalysis, ensuring that differences in input don't follow predictable patterns.

3. 16 Round's of Encryption:

- The effectiveness of differential cryptanalysis depends on tracking input-out differences across multiple rounds.

- DES's 16-round structure makes it significantly harder to predict how differences propagate, increasing resistance.

4. Key Mixing with XOR operations:

- XORing with roundkeys ensures that differential characteristics become less predictable.
- The diffusion properties of the Feistel network further randomise input-output relationships.

Thus, differential cryptanalysis could be used to break DES with 2^{43} known plaintext-ciphertext pairs, making it feasible for powerful adversaries.

□ Comparison of DES vs AES resistance to differential cryptanalysis:

Feature	DES (Feistel Structure)	AES (Substitution-Permutation Network)
S-Boxes	8 small S-Boxes	1 highly non-linear S-Box.
Diffusion	Slow diffusion (Feistel structure)	Fast diffusion (MixColumns + ShiftRows)
Round Key Dependency	Weak key schedule	Strong key schedule
Number of Rounds	16	10, 12 or 14 → 128 / 192 / 256
Resistance to Differential Cryptanalysis	Moderate, but breakable with 2^{43} pairs	Strong no practical attacks.

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

Finding the modular Inverse using the extended Euclidean Algorithm:

The modular inverse of an integer a modulo n is an integer x such that:

$$a \cdot x \equiv 1 \pmod{n}.$$

This modular inverse exists if and only if a and n are coprime (gcd(a,n) = 1).

Step: 1. Use the euclidean algorithm to compute gcd (a,n).

2. If gcd (a,n) ≠ 1, the inverse doesnot exist.

3. If gcd (a,n) = 1, apply the extended Euclidean Algorithm to express 1 as a linear combination:

$ax + ny = 1$, Here, x is the modular inverse of a modulo n.

4. Convert x to a positive value if necessary:

$x \equiv x \pmod{n}$.

Example: Finding $3^{-1} \bmod 26$.

$$3x \equiv 1 \pmod{26}.$$

1. Apply the euclidean Algorithm — Find gcd (3, 26):

$$26 = 3 \times 8 + 2$$
$$3 = 2 \times 1 + 1$$
$$2 = 1 \times 2 + 0$$

$\therefore$ gcd(3, 26) = 1, the modular inverse exists.

2. Apply the extended Euclidean Algorithm:

1 as a linear combination of 3 and 26

by working backward:

$$1 = 3 - 1 \times 2$$

Substituting $2 = 26 - 3 \times 8$:

$$1 = 3 - 1 \times (26 - 3 \times 8)$$
$$1 = 3 - 1 \times 26 + 8 \times 3$$
$$1 = 9 \times 3 - 1 \times 26$$

Thus, $x = 9$ is the modular inverse $\because 3 \times 9 \equiv 1 \pmod{26}$

☐ **Utilization in RSA key generation:**

The extended Euclidean Algorithm plays a crucial role in RSA cryptography, particularly in key generation.

Step: 1. choose two large prime numbers, p and q.

2. Compute $n = p \times q$, the modulus.

3. Compute Euler's totient function:

$$\phi(n) = (p-1)(q-1)$$

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 the modular inverse of e modulo $\phi(n)$:

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

This step is done using the Extended Euclidean Algorithm.

6. The public key is (e, n) and the private key is (d, n).

☐ Importance of the Algorithm's efficiency in cryptography:

1. RSA Relies on Large primes: - 2048 bit or Large Primes
   - Computing the modular inverse of a 2048-bit number requires an efficient algorithm.

2. Extended Euclidean Algorithm Runs in Polynomial Time:
   - It runs in $O(\log n)$ time, making it very fast even for large numbers
   - Brute-force approaches (checking all numbers up to n) would be impractical.

3. Fast Decryption and Signature Verification:
   - RSA decryption and digital signatures rely on exponentiation using d.
   - Efficient computation of d ensures secure and practical cryptographic operations.

6.13: 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 limitted in decryption.

i) Electronic Codebook (ECB) mode encrypts each plaintext block independently using the same key:

$$C_i = E_K(P_i)$$

where, C_i = ith ciphertext block.

P_i = ith plaintext block

E_K = encryption function under key K.

Mathematical Proof of ECB insecurity:

If two plaintext blocks are identical, their corresponding ciphertext blocks are also identical:

$$P_i = P_j \Rightarrow C_i = C_j$$

For highly redundant data (such as images or formatted text), this property allows an attacker to recognise patterns in the cipher text without knowing the key.

Ex: ECB Mode on an image.

If an image has large areas of uniform color, the same plaintext blocks will map to identical ciphertext blocks, leading to an encrypted image that still reveals the structure of the original image.

Thus, ECB mode lacks diffusion and doesn't hide data patterns making it insecure.

ii) ~~Derive~~ CBC Mode (Cipher Block Chaining)

~~Encryption~~:

The recurrence relations for CBC encryption and decryption have been derived as:

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

These recurrence relations illustrate how the chaining machaism in CBC mode makes it more secure than ECB.

1. Encryption:

For the 1st block P_1, the ciphertext C_1 is computed as:

$$C_1 = E(K, P_1 \oplus IV)$$

[where, IV = initialization vector (for the 1st block)]

- For subsequent blocks $P_2, P_3, \ldots P_K$, the ciphertext blocks are calculated as:

$$C_i = E(K, P_i \oplus C_{i-1}) \quad \text{for } i = 2, 3, \ldots k.$$

Thus:

The recurrence relation for CBC encryption is

$$C_i = E(K, P_i \oplus C_{i-1}) \quad \text{for } i = 1, 2, \ldots, K$$

CBC Mode Decryption:

Decryption:

For the 1st ciphertext block C_1, the plaintext P_1 is computed as:

$$P_1 = D(K, C_1) \oplus IV$$

- For subsequent ciphertext blocks $C_2, C_3 \ldots C_K$, the plaintext blocks are:

$$P_i = D(K, C_i) \oplus C_{i-1} \quad \text{for } i = 2, 3, \ldots K.$$

Thus, the recurrence relation for CBC decryption is

$$P_i = D(K, C_i) \oplus C_{i-1} \quad \text{for } i = 1, 2, \ldots$$

Error Propagation in CBC decryption:

The error in C_i will affect the decryption of P_i, because the incorrect C_i will lead to an incorrect P_i:

$$P_i = D(K, C_i) \oplus C_i$$

$$P_{i+1} = D(K, C_{i+1}) \oplus$$

Thus, an error in a single ciphertext block affects both the current plaintext block and the next one.

However, the effect of the error is limited to only two blocks:

* The block with the error itself is corrupted.
* The next block's plaintext is also corrupted.

It is limited to only two blocks. The block with the error and the subsequent block.

But the rest of the plaintext remains unaffected, so the error propagation is limited. After the erroneous block and the next one, the decryption process will continue correctly without further corruption.

Q14 Why the linearity of LFSRs makes them vulnerable to known-plaintext attacks and propose a mathematical method to mitigate this vulnerability.

Linearity of LFSRs:

An LFSR is a shift register whose output bits are fed back into the register through a linear function (typically XORs). The key aspect of LFSRs that makes them vulnerable is their linear nature. The sequence of output bits generated by an LFSR is determined by a linear recurrence relation.

Specifically, if the internal state of the LFSR at time t is denoted as S_t, the the output bit at time t, O_t is a linear combination of previous state bits:

$$\begin{cases} S_t = (S_{t-1}, S_{t-2}, \dots \\ O_t = f(S_t \end{cases}$$

35 ①

Example:

$$O_t = S_{t-3} \oplus S_{t-1}$$

where,
simple 3 bit LFSR,
O_t = the output bit might be the XOR of two bits of the internal state.

Known plaintext attack:

The attacker has access to both the ciphertext and the corresponding plaintext for some part of the communication. With LFSRs, the attacker can exploit the linear nature of the ~~key~~ generated keystream to recover the internal state of the LFSR and consequently, the secret key.

- Keystream generation: The keystream of a stream cipher based on an LFSR is generated by the output sequence of the LFSR.

- Known-plaintext Attack:

$$C_t = P_t \oplus K_t$$

$$K_t = P_t \oplus C_t$$

K_t = keystream
C_t = ciphertext
P_t = plaintext.

Mathematical Method to Mitigate the Vulnerability:

A simple method to increase the security is to use an LFSR-based combination generator or to combine multiple LFSR in a way that introduces non-linearity.

Non-Linear combination generator where multiple LFSRs are combined in a non-linear manner. A practical method is using the feedback from multiple FLFSRs or applying a non-linear function to the outputs of multiple LFSRs.

Example: Combining multiple LFSRs with non-linear feedback:

1. Multiple LFSRs: Let $L_1, L_2 \cdots L_m$ be m independent LFSRs, each generating an output sequence $O_{1,t}, O_{2,t}, O_{m,t}$.

2. Non-Linear Combination: The final keystream bit K_t is generated by applying a non-linear function f to the outputs of these LFSRs. The keystream bit might be:

$$K_t = f(O_{1t}, O_{2,t}, \cdots, O_{m,t}).$$

Ex: $K_t = O_{1,t} \oplus O_{2,t} \oplus O_{3,t} \oplus \cdots \oplus O_{m,t}$.

the combination of multiple LFSRs with a non-linear function (such as an XOR of outputs from multiple LFSRs)

To initigate this vulnerability, non-linear combination

generators can be used to increase the complexity of the keystream, breaking the linearity and making the system more secure against attacks.

11b: 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.

Shannon's definition of perfect Secrecy:

- Indistinguishability bet^n the ciphertext and the plaintext, given that the ciphertext provides no information about the plaintext.

- A cryptographic system achieves perfect secrecy if the probability distribution of the plaintext is the same, regardless of the ciphertext.

Mathematically, perfect secrecy is defined by the following condition:

IT-2360

$P(M=m|C=c) = P(M=m)$ for all $m \in M$, $c \in C$.

Where,

M = the set of all possible plaintext messages.

C = the set of all possible ciphertexts.

$P(M=m|C=c)$ = the posterior probability of the plaintext m given the ciphertext c.

The ciphetext doesnot provide any additional information about the plaintext. knowing the ciphertext c doesnot change the probability distribution of the plaintext m and the plaintext is completely hidden from an adversary.

ii) Prove that, the one-time pad achieves perfect secrecy:

The one-time pad (OTP) is a symmetric key cipher in which the plaintext is XORed with a random key that is the same length as the plaintext.

Let's prove the OTP achieves perfect secrecy under the condition that the key K is uniformly random and $|K| \geq |M|$, the key length is greater than or equal to the plaintext length.

Encryption OTP: $C = M \oplus K$

Where,

M = the set of all possible plaintext

K = the set of all possible key

C = the " " " " ciphertext

$P(M=m)$ = the prior probability distribution of the plaintexts.

$P(C=c)$ = the probability distribution of ciphertexts

Proof of perfect secrecy:

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

Using Baye's' theorem, the posterior probability is

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

In OTP, the encryption function is:

$$C = M \oplus K$$

$$K = M \oplus C$$

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

The probability of $C=c$ is the same regardless of which m is chosen:

$$P(C=c) = \frac{1}{|K|}$$

Therefore,

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

iii) Critically Analyze why perfect secrecy is impractical for large-scale communication systems:

IT-23602

6.16 A Linear Congruential Generator (LCG) is defined by the recurrence relation : $x_{(n+1)} = ax_n + c \bmod m$ 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 aaa, ccc and mmm. Compute the first 5 numbers of an LCG sequence starting with a given seed $x_0 = 7$.

The recurrence relation :

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

Where :
a = multiplier
c = increment
m = modulus
x_n = current value in the sequence.

Suppose, $a = 5$, $c = 3$, $m = 16$

the seed $x_0 = 7$.

the 1st 5 numbers of the sequence by iteratively applying the recurrence relation.

1. $x_1 = (ax_0 + c) \bmod m$
$= (5 \cdot 7 + 3) \bmod 16$
$= (35 + 3) \bmod 16$
$= 38 \bmod 16$
$= 6$

2. $x_2 = (a \cdot x_1 + c) \bmod m$
$= (5 \cdot 6 + 3) \bmod 16$
$= (30 + 3) \bmod 16$
$= 33 \bmod 16$
$= 1$

3. $x_3 = (a \cdot x_2 + c) \bmod m$
$= (5 \cdot 1 + 3) \bmod 16$
$= € 8 \bmod 16$
$= 8$

4. $x_4 = (a \cdot x_3 + c) \bmod m$
$= (5 \cdot 8 + 3) \bmod 16$
$= 43 \bmod 16$
$= 11$

5. $x_5 = (a \cdot x_4 + c) \bmod m$
$= (5 \cdot 11 + 3) \bmod 16$
$= 58 \bmod 16$
$= 10$

The 1st 5 numbers generated by the LCG with the given parameters $x_0 = 7$, $x_1 = 6$, $x_2 = 1$, $x_3 = 8$, $x_4 = 11$

IT-2360

Q18. Given an RSA key pair with the public key (e,n) and the private key ddd, where: P=5, q=11 (two Prime numbers), n=p.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 orginal message.

Let P=7, 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.

RSA encryption and decryption, followed by the RSA signature and verification.

Given that,

→ P=5, q=11, M=2 (the plaintext message to encrypt

→ The public key is (e,n) and the private key is d.

- $n = P \times q = 5 \times 11 = 55$.
- $\phi(n) = (P-1)(q-1) = (5-1)(11-1)$

$= 4 \times 10 = 40$

Step 1: Choose public and Private Exponents:

- $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$.

typical choice for e = 3 or 17, but we need to check that is co-prime to $\phi(n) = 40$.

- Let e=3, gcd (3, 40) = 1, so it is valid.
- the private exponent d,

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

$$d \times 3 \equiv 1 \pmod{40}$$

using extended euclidean Algorithm, we can find d.

- Apply the extended euclidean Algorithm to find the modular inverse of 3 mod 40;

$$3d \equiv 1 \pmod{40}$$

the inverse of 3 modulo 40 is $d = 27$.

∴ the private key, $d = 27$.

Step 2: Encrypt the message:

$$C = M^e \pmod{n} \quad \text{where } e = 3 \text{ and } n = 55$$

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

∴ the ciphertext, $C = 8$.

Step 3: Decrypt the ciphertext:

$$C = 8, \quad M = C^d \pmod{n} \quad \text{where, } d = 27 \text{ and } n = 55$$

$$M = 8^{27} \pmod{55} = 2$$

Thus, after decryption, we recover the original message $M = 2$.

Part 2: RSA signature and Verification:

Given that,

$P = 7$, $q = 3$ (new primes)

$H(m) = 3$ (hash of the message to sign)

Public key (e, n), private key d

Step 1: compute n and $\phi(n)$

$\therefore n = p \times q = 7 \times 3 = 21$

$\phi(n) = (p-1)(q-1) = (7-1)(3-1) = 6 \times 2 = 12$

Step 2: Choose public and Private exponents:

Choose e, $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$.

- when $e = 5$, $\gcd(5, 12) = 1$.
- Private exponent d,

$$d \times e \equiv 1 \pmod{\phi(n)}.$$

$$d \times 5 \equiv 1 \pmod{12}$$

The inverse of 5 modulo 12 is $d = 5$

(since $5 \times 5 = 25 \equiv 1 \pmod{12}$).

The private key, $d = 5$ and public key $(e = 5, n = 21)$.

Step 3: Sign the Hash of the message.

$$H(m) = 3, \quad S = H(m)^d \pmod{n}$$

where, $H(m) = 3$, $d = 5$ and $n = 21$:

$$S = 3^5 \pmod{21} = 243 \pmod{21} = 243 - (11 \times 21)$$

$$= 243 - 231 = 12$$

$\therefore$ signature, $S = 12$

Step 4: Verify the signature:

Use public key ($e=5$, $n=21$).

The verification formula

$$H(m) \equiv S^e \pmod{n}.$$

where, $S=12$, $e=5$ and $n=21$:

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

Compute $12^5 \pmod{21}$ using modular exponentiation:

$$12^2 \pmod{21}$$
$$= 144 \pmod{21}$$
$$= 144 - (6 \times 21)$$
$$= 144 - 126$$
$$= 18.$$

$$12^4 \pmod{21}$$
$$= 18^2 \pmod{21}$$
$$= 324 \pmod{21}$$
$$= 324 - (15 \times 21)$$
$$= 324 - 315 = 9$$

$$12^5 \pmod{21} = 9 \times 12 \pmod{21}$$
$$= 108 \pmod{21}$$
$$= 108 - (5 \times 21)$$
$$= 108 - 105 = 3.$$

Since $12^5 \pmod{21} = 3$, and this matches the original hash $H(m)=3$, the signature is valid.

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.

i) The equation of the elliptic curve is

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

where $p = 23$, $a = 1$ and $b = 1$,

$$\therefore y^2 = x^3 + x + 1 \pmod{23} \quad \text{—①}$$

Substitute the point $P = (3, 10)$ into the curve eqn.

$x = 3$, $y = 10$

from ① the eqn holds for this point:

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

$$10^2 \equiv 3^3 + 3 + 1 \pmod{23}$$

Calculate both sides modulo 23:

$$L.H.S = 10^2 = 100 \equiv 100 \bmod 23$$
$$= 100 - 4 \times 23.$$
$$= 100 - 92$$
$$= 8$$

$$R.H.S = 3^3 = 27 \equiv 27 \bmod 23 = 27 - 23 = 4, 30;$$
$$x^3 + x + 1 = 4 + 3 + 1 = 8 \pmod{23}.$$

L.H.S = R.H.S = 8 modulo 23. $\therefore P = (3, 10)$ does lie on the elliptic curve.

ii) Find the result of doubling the point P=(3,10) using the elliptic curve point doubling formula.

The point doubling the point P=(3,10) using the elliptic curve point doubling

The point doubling formula for elliptic curve is given by:

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

$$x_3 = \lambda^2 - 2x_1 \pmod{p}$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \pmod{p}$$

Given: $P = (x_1, y_1) = (3,10)$

$a = 1$

$P = 23$

First, compute λ: $\lambda = \frac{3x_1^2 + a}{2y_1} \pmod{23}$

Substitute $x_1 = 3$, $y_1 = 10$, and $a = 1$:

$$\lambda = \frac{3 \times 3^2 + 1}{2 \times 10} \pmod{23}$$

$$\lambda = \frac{3 \times 9 + 1}{20} \pmod{23}$$

$$\lambda = \frac{27+1}{20} \pmod{23} = \frac{28}{20} \pmod{23}$$

Now, reduce $\frac{28}{20}$ mod 23, First reduce the numerator and denominator modulo 23:

$28 \equiv 28 - 23 = 5 \pmod{23}$

$20 \equiv 20 \pmod{23}$

Next, find the modular- inverse of 20 modulo 23.
Using the extended Euclidean Algorithm, we find:

$$20^{-1} \equiv 7 \pmod{23}$$

Thus $\lambda = 5 \times 7 = 35 \equiv 35 - 23 = 12 \pmod{23}$

Now calculate x_3 and y_3:

$$x_3 = \lambda^2 - 2x_1 \pmod{23}$$

$$x_3 = 12^2 - 2 \times 3 = 144 - 6$$
$$= 138$$
$$\equiv 138 - 6 \times 23$$
$$= 138 - 138 = 0 \pmod{23}$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \pmod{23}$$

$$y_3 = 12(3 - 0) - 10 = 12 \times 3 - 10 = 36 - 10$$
$$= 26$$
$$\equiv 26 - 23$$
$$= 3 \pmod{23}$$

∴ the result of doubling the point

$$P = (3, 10)$$

$$2P = (x_3, y_3) = (0, 3)$$

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

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

$$x_3 = \lambda^2 - x_1 - x_2 \pmod{p}$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \pmod{p}$$

Given, $P = (x_1, y_1) = (3, 10)$

$Q = (x_2, y_2) = (9, 7)$

$P = 23$

$\therefore \quad \lambda = \frac{y_2 - y_1}{x_2 - x_1} \pmod{23}$

$= \frac{7-10}{9-3} \pmod{23}$

$= \frac{-3}{6} \pmod{23}$

Since, modulo 23, 1st reduce $-3 \bmod 23$:

$-3 \equiv 20 \pmod{23}$

find the modular inverse of 6 modulo 23. Using extended euclidean algorithm, we find:

$6^{-1} \equiv 4 \pmod{23}$

Thus, $\lambda = 20 \times 4 = 80 \equiv 80 - 3 \times 23 = 80 - 69 = 11 \pmod{23}$

Calculate x_3 and y_3:

$\Rightarrow x_3 = \lambda^2 - x_1 - x_2 \pmod{23}$

$\therefore x_3 = 11^2 - 3 - 9 = 121 - 3 - 9 = 109 \equiv 109 - 4 \times 23$

$= 109 - 92$

$= 17 \pmod{23}$

$\Rightarrow y_3 = \lambda (x_1 - x_3) - y_1 \pmod{23}$

$\therefore y_3 = 11(3 - 17) - 10 = 11(-14) - 10 = -154 - 10 = -164$

$\equiv -164 + 8 \times 23$

$= -164 + 184$

$= 20 \pmod{23}$

Thus the result of adding $P = (3, 10)$ and $Q = (9, 7)$ is

$P + Q = (x_3, y_3) = (17, 20)$

Q20. Suppose an elliptic curve $y^2 = x^3 + 7x + 10 \cdot \text{mod } 37$ is used for ECDSA, with the base point $G = (2, 5)$ and order, $n = 19$

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

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

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.

the elliptic curve is

$$y^2 = x^3 + 7x + 10 \quad \text{mod } 37$$

base point, $G = (2, 5)$, order $n = 19$ and private key, $d = 9$.

Public key $Q = dG$. using scalar multiplication.

i) Generate the public key $Q = dG$.

Elliptic curve point Doubling and Addition Formula:

Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be two points on the elliptic curve.

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

$$x_3 = \lambda^2 - 2x_1 \pmod p$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \pmod p$$

IT-1360

- Point Addition (for distinct points P and Q):

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

$$x_3 = \lambda^2 - x_1 - x_2 \pmod p$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \pmod p$$

scalar multiplication to compute $Q = 9G$. It can be broken down using the binary method or the double and add method.

Let, $G = (2, 5)$ and compute $9G$ step by step.

Step 1: compute $2G$ (point doubling)

- Point doubling formula on $G = (2, 5)$.

$$\lambda = \frac{3 \times 2^2 + 7}{2 \times 5} \mod 37$$

$$= \frac{3 \times 4 + 7}{10} \mod 37$$

$$= \frac{19}{10} \mod 37.$$

We need to compute $10^{-1} \mod 37$, using extended Euclidean algorithm, $10^{-1} \equiv 19 \mod 37$.

$$\lambda = 19 \times 19 = 361 \equiv 361 \mod 37$$

$$= 361 - 9 \times 37$$

$$= 361 - 333$$

$$= 28 \mod 37.$$

∴ new point co-ordinates:

$x_3 = \lambda^2 - 2x2 = 28^2 - 4$

$= 784 - 4 = 780 \equiv 780 \bmod 37$

$= 780 - 21 \times 37$

$= 780 - 777$

$= 3 \bmod 37$.

$y_3 = \lambda x(2x3) - 5 = 28 x(-1) - 5 = -28 - 5$

$= -33$

$\equiv -33 \bmod 37$

$= 4 \bmod 37$

∴ $2G = (3, 4)$

Step 2: Compute 4G (Point doubling)

• $2G = (3, 4)$

$$\lambda = \frac{3x3^2+7}{2x4} \bmod 37$$

$$= \frac{3x9+7}{8} \bmod 37$$

$$= \frac{34}{8} \bmod 37.$$

Compute $8^{-1} \bmod 37$. Using extended euclidean algorithm $8^{-1} \equiv 14 \bmod 37$

∴ $\lambda = 34 \times 14 = 476 \equiv 476 \bmod 37$

$= 476 - 12 \times 37$

$= 476 - 444$

$= 32 \bmod 37$.

∴ the new point co-ordinates:

$x_3 = \lambda^2 - 2x3 = 32^2 - 6 = 1024 - 6 = 1018$

$\equiv 1018 \mod 37$

$\equiv 1018 - 27 \times 37$

$= 1018 - 999 = 19 \mod 37.$

$y_3 = \lambda \times (3a - 19) - 4 = 32 \times (-16) - 4$

$= -512 - 4 = -516 \equiv -516 \mod 37$

$= 25 \mod 37$

∴ $4G = (19, 25)$

Step 3. compute 8G (point doubling)

$4G = (19, 25)$:

$$\lambda = \frac{3 \times 19^2 + 7}{2 \times 25} \mod 37$$

$$= \frac{3 \times 361 + 7}{50} \mod 37$$

$$= \frac{1080}{50} \mod 37$$

Find $50^{-1} \mod 37$ → previous step.

continue. 9G.

∴ After performing the necessary point doubling and additions, the final public key $Q = 9G$ will be obtained

11) .

The signing process in ~~Bet~~ ECDSA involves;

1. $R = kG$
2. Extract r from $R = (r, y)$, where r is the x-co-ordinate of the point.
3. compute $s = k^{-1}(H(M) + r \cdot d) \bmod n$.

Given, $H(M) = 8$, $k = 3$, $n = 19$

$\therefore$ $R = 3G$

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

$k^{-1} \bmod 19$,

using the extended euclidean algorithm,

$k^{-1} = 13 \bmod 19$

where,
$d = 9$ Private key
$k^{-1} \bmod n$ = modular inverse of $k \bmod n$

~~$s = k^{-1}$~~ Now compute s using the value.

iii) Verify the signature;

The signature (r, s)

$v = s^{-1} \cdot H(M) \bmod n$

and check if L.H.S = R.H.S.