Answer to the question no 01:

Quantum computing poses a significant threat to traditional public key cryptosystems such as RSA and Elliptic Curve Cryptography (ECC) due to the development of Shor's algorithm. Shor's algorithm allows a sufficiently powerful quantum computer to efficiently factor large integers and compute discrete logarithms, which are the foundational problems that ensure the security of RSA and ECC respectively. This renders these crypto-systems obsolete in a post-quantum era, necessitating the development and deployment of quantum-resistant cryptographic protocols.

Implications of quantum computing on Cryptographic Security:

① Breaking public key encryption: RSA, ECC and other number theoretic cryptosystems will be rendered insecure once quantum computers reach the necessary scales. This means encrypted communications, digital signatures

P.T.O.

Scanned with CamScanner

and secure key exchanges replying on these algorithms will be compromised.

② Threat to Digital Signatures: Many authentication mechanisms, including SSL/TLS use ECC-based signatures. A quantum attack could allow an adversary to forge signatures, undermining the integrity of digital certificates and authentication systems.

3. Key exchange Vulnerabilities: Protocols as Diffie Hellman key exchange (both integer based and ECC-based) would be insecure, allowing an attacker ~~in~~ to intercept and decrypt confidential communications.

4. Impact on Stored Encrypted Data: Encrypted data intercepted today could be stored and later decryped when quantum computers become viable, posing a risk known as "harvest now, decrypt Later".

Scanned with CamScanner

Post quantum cryptographic Algorithms:

① Lattice Based Cryptography

→ examples: CRYSTALS-Kyber (key exchange), CRYSTALS-Dilithium (digital signatures), NTRU.

→ Security Basis: The hardness of problems like Learning with Errors (LWE) and Shortest vector problem (SVP), which are believed to be resistant to both classical and quantum attacks.

→ Quantum Resistance: These problems remain hard even for quantum computers because they don't map efficiently to Shor's Algorithm.

② Code Based Cryptograph

examples: Rainbow (digital signature scheme)

Security Basis: The complexity of solving a system of multivariate quadratic equations over a finite field.

Quantum Resistance: Solving such equations is NP hard, and quantum algorithms do not provide

Scanned with CamScanner

significant speedups.

④ Hash Based cryptography:

examples: SPHINCS + (stateless hash based signature scheme)

Security bases: Security relies on the pre-image and second preimage resistance of cryptographic hash functions.

Quantum resistance: Grover's algorithm, which speeds up brute force attack, reduces security by only a square root factor, making it possible to adjust key sizes accordingly

Answer to the question no 2:

```
import time
import os
def custom_prng(seed = None, min_val = 0, max_val = 100)
"""
A simple pseudo-random number generator (PRNG)
based on:
- current timestamp (nanoseconds precision)
→ Process ID (os.getpid())
```

Scanned with CamScanner

- A linear congruential transformation with a modulus

Args:
seed (int, optional): An optional seed value for deterministic behavior.

min_val (int): Minimum value of the output range.

max_val (int): Maximum value of the output range

Returns:
int: A pseudo-random number within the specified range.
""
if seed is None:
timestamp = int(time.time() * 1e9 # Nanosecond precision
pid = os.getpid()
seed = (timestamp ^ pid) & 0xFFFFFFFF # combine timestamp and PID

# LCG parameters (chosen to ensure good randomness properties)
a = 1664525
c = 1013904223
m = 2**32 # 32 bit modulus
# Generate a pseudo-random number
seed = (a * seed + c) % m
# map the result into the desired range

P.T.O.

Scanned with CamScanner

```
return min_val + (seed % (max_val - min_val + 1))
# example usage
for _ in range(5):
    print(custom_prng(min_val = 1, max_val = 100))
```

Answer to the question no 3:

Comparison of Traditional ciphers vs Modern Symmetric Ciphers:

Cryptographic ciphers have evolved significantly from classical techniques to modern encryption standards. Below is a detailed comparison of traditional ciphers (Caesar, Vigenere, playfair) and modern symmetric ciphers (AES, DES) based on key factors such as security, key length, speed and resistance to cryptanalysis.

Cipher	Key length
Caesar cipher	1 character shift (5 bits)
Vigenere cipher	Varies, typically short (e.g. 26 letters)
playfair cipher	25-Letter key square (effectively ~28 bits)
DES (Data Encryption Standard)	56 bits
AES (Advanced Encryption Standard)	128, 192 or 256 bits

Scanned with CamScanner

∴ we can see, traditional ciphers have ~~exactly~~ extremly short key lengths making brute force attacks trivial. and modern symmetric ciphers use long key lengths (AES-256 has 2^{256} possible keys).

② Encryption / Decryption speed

Ciphers	Speed
Caesar Cipar	Very fast
Vigenere cipher	Fast
Playfair cipher	Slower than Vigenere
DES	Faster than AES but insecure.
AES	Optimized for hardware/software, efficient with parallelism.

So, we can see traditional ciphers ~~have~~ Extremely fast where as AES is optimized for speed, specially with hardware acceleration.

Scanned with CamScanner

③ Security against Cryptanalysis

Cipher	Resistance to Cryptanalysis
Caesar Cipher	Easily Broken using frequency analysis.
Vigenere cipher	Susceptible to Kasiski examination and frequency analysis.
Playfair Cipher	More secure than Caesar but still vulnerable to frequency analysis.
DES	Vulnerable to brute force due to short key length (56-bit keys)
AES	Resistant to all known cryptanalytic attacks with sufficient key length.

So, we see that traditional ciphers are vulnerable to classical frequency analysis and simple attacks. whereas modern symmetric ciphers are designed to resist differential and linear cryptanalysis, chosen-plaintext attacks and brute force attacks.

Scanned with CamScanner

Answer to the question no 04:

Step 1: Define the action of S_4 on 2-element subsets

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

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

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

Step 2: Show the action ~~of~~ $\cdot$ is well defined

For the action to be well-defined, we must show that:

① $\sigma.\{a,b\}$ is still a 2-element subset of X.

② the action respects composition i.e $(\tau\sigma).\{a,b\}$ $= \tau.(\sigma.\{a,b\})$.

Verification of property 1: Since σ is a permutation, it maps distinct elements of X to distinct elements of X. Thus if $a \neq b$, then $\sigma(a) \neq \sigma(b)$

P.T.O.

Scanned with CamScanner

then $\sigma(a) \neq \sigma(b)$, ensuring that $\sigma \cdot \{a,b\} = \{\sigma(a), \sigma(b)\}$ is still a valid 2-element subset.

Verification of property 2 (Associativity):

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

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

Since both expressions are equal, the action satisfies associativity. Thus, the action is well-defined.

Step 3: Compute the orbit of $\{1,2\}$:

The orbit of $\{1,2\}$ under S_4 consists of all 2 element subsets that can be obtained by applying some permutation $\sigma \in S_4$. The set of all 2 element subsets of X is:

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

Since S_4 acts transitively on these subsets (i.e any pair $\{a,b\}$, can be mapped to any other pair via some permutation), the orbit of $\{1,2\}$ consists of all 6 subsets. Thus, the orbit size is

PTO.

Scanned with CamScanner

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

Compute the stablizer of $\{1,2\}$: The stablizer of $\{1,2\}$ denoted $Stab\{1,2\}$, consists of all permutations $\sigma \in S_4$ that leave $\{1,2\}$ unchanged,

That is, σ must satisfy:

$$\sigma.\{1,2\} = \{1,2\}$$

This means σ can:

→ swap 1 and 2 or leave them unchanged.

→ Permute the remaining elements (3 and 4) freely.

The permutations that satisfy these conditions are:

→ The identify permutation e.

→ The transposition (12), which swaps 1 and 2.

→ Any permutation of 3 and 4 i.e (34), $(12)(34)$

and the identity on $\{3,4\}$.

Thus, the stablizer consists of 4-elements:

$$\{e, (12), (34) (12)(34)\}$$

Therefore the size of the stablizer is:

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

Scanned with CamScanner

Verify / Proof of ~~ot~~ orbit - Stabilizer theorem:

The orbit - stabilizer theorem states that:

$$|S_4| = |orb(\{1,2\})| \cdot |stab(\{1,2\})|$$

Since $S_4 = 24$, $|orb(\{1,2\})| = 6$ $|stab(\{1,2\})| = 4$

we get,

$$24 = 6 \times 4$$

$\therefore$ size of orbit is 4 and size of stabilizer is 6 (proved).

Answer to the question no -5:

To analyze the structure of the finite field $GF(2^2)$, we construct it using the irreducible polynomial x^2+x+1 over $GF(2)$.

Construction of $GF(2^2)$:

Since we are working over $GF(2)$, the elements of $GF(2^2)$ can be presented as:

$$GF(2^2) = \{0, 1, \alpha, \alpha+1\}$$

where α is a root of the irreducible polynomial $x^2+x+1=0$, meaning that it satisfies:

$$\alpha^2+\alpha+1=0 \Rightarrow \alpha^2=\alpha+1$$

P.T.O.

Scanned with CamScanner

IT23618

① Show that $GF(2^2)$ forms a group under m

to form a group under multiplication, the nonzero elements $\{1, \alpha, \alpha+1\}$ must satisfy:

① Closure: Multiplication of any two nonzero elements results in another element within the set.

② Associavity: Inherited from field multiplication.

③ Identify element: The multiplication identity is 1

④ Inverse: Each element must have a multiplicative inverse

we compute the multiplication table using the relation $\alpha^2 = \alpha + 1$:

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

Scanned with CamScanner

observing the table:

→ The identity element is 1.

→ The inverse of α is $\alpha+1$ since $\alpha(\alpha+1)=1$

(N.B $\alpha(\alpha+1)=1$
$\therefore \alpha^{-1}=\alpha+1$)

→ the inverse of $\alpha+1$ is α

Since all the conditions of a group are satisfied, the set $\{1, \alpha, \alpha+1\}$ forms a group under multiplication

(ii) Verify whether the set of all nonzero elements of $GF(2^2)$ is cyclic: A cyclic group is one that can be generated by a single element.

we check whether one of the elements can generate the group;

① Compute power of α:

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

Since $\alpha^3=1$, in order of α is 3.
so it generates the group

② Compute power of $\alpha+1$:

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

Since $(\alpha+1)^3=1$, $\alpha+1$ also generates the group

P.T.O.

Scanned with CamScanner

Thus $\{1, \alpha, \alpha+1\}$ forms a cyclic group of order, meaning that the set of non zero elements in $GF(2^2)$ is cyclic

Answer to the question no - 06:

Define $GL(2,R)$ and the set of scalar matrices:

The general linear group $GL(2,R)$ consists of all 2×2 invertible matrices with real entries:

$$GL(2,R) = \{A \in M_{2\times2}(R) \mid \det(A) \neq 0\}$$

The set of scalar matrices in $GL(2,R)$ is:

$$Z = \{\lambda I \mid \lambda \in R^{\times}\} = \left\{\lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \mid \lambda \neq 0\right\}$$

when I = Identity matrix

Show that Z is a Normal Subgroup of $GL(2,R)$:

To show that Z is a subgroup, we check the subgroup criteria:

① Closure: If $\lambda I, \mu I \in Z$; then their product is: $(\lambda I)(\mu I) = (\lambda\mu I) \in Z$

② Identity: The identity matrix $I = 1I$ is in Z

③ Inverses: If $\lambda I \in Z$, its inverse is: $(\lambda I)^{-1} = \frac{1}{\lambda} I \in Z$, since $\lambda \neq 0$

P.T.O.

Scanned with CamScanner

Thus, Z is a subgroup of $GL(2,R)$

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

The quotient group $GL(2,R)/Z$ consists of cosets of Z in $GL(2,R)$. Each coset can be represented by an equivalence class:

$$[A] = \{\lambda A \mid \lambda \neq 0\}$$

This means that two matrices $A, B \in GL(2,R)$ are in the same coset if and only if they differ by a scalar multiple.

The factor group $GL(2,R)/Z$ essentially consists of equivalence classes of matrices modulo scalar matrices. This is precisely the projective general linear group:

$$PGL(2,R) = GL(2,R)/R^{\times}$$

Interpretation of the structure: The group $PGL(2,R)$ describes linear transformations up to scalar multiplication, which are exactly the transformations that act on the real projective line RP^1. In other words, $PGL(2,R)$ captures the essential symmetries of projective geometry over R, where multiplying a matrix by a scalar

Scanned with CamScanner

doesn't changed the induced transformati on the projective space.

Thus the factor group $GL(2,\mathbb{R})/Z$ is fundamentally related to the projective linear transformations of $\mathbb{RP}^1$, which act by fractional linear transformations:

$$z \longrightarrow \frac{az+b}{cz+d} \quad \text{for} \begin{bmatrix} a & b \\ c & d \end{bmatrix} \in GL(2,\mathbb{R}$$

Answer to the question no 07:

The Diffie-Hellman (DH) key exchange is a cryptographic protocol that allows two parties to securely share a common secret key over an insecure communication channel. It was ~~Hiro~~ introduced by Whitfield Diffie and Martin Hellman in 1976 and serves as the foundation of for many modern cryptograp protocols.

Applications of Diffie-Hellman:

The DH key exchange is widely used in various security protocols:

Scanned with CamScanner

① Secure Web communication (TLS/SSL):

Used in HTTPS (via Elliptic Curve Diffie-Hellman (ECDH)) to establish encrypted connections between web browsers and servers.

② Virtual Private Networks (VPNs):

IPSec VPNs use DH key exchange for securely establishing encryption keys between client and server.

③ Messaging Apps:

Signal, Whatsapp and other at end to end encrypted ~~connection~~ messaging platforms implement DH-based protocols to securely establish session keys.

④ SSH (Secure Shell):

Secure key exchange in SSH uses DH to establish an encrypted connection between client and server.

⑤ Cryptographic Protocols:

Used in PGP (Pretty Good Privacy), Tor and Zero Knowledge Proofs.

P.T.O.

Scanned with CamScanner

IT 23618

Security of the Diffie Hellman Protocol:

The security of the Diffie Hellman key exc. is based on the Discrete Logarithm Probl. (DLP) ;. Below we discuss its security against

→ ~~Given $g^x \bmod p$, computing x~~ is common attacks such as man in the middle and the role of the discrete logarithm problem in ensuring its security.

- Man in the middle attack; Since DH does not provide authentication, an attacker can intercept and modify exchanged public keys values. The attacker (Mallory) could establish two separate key exchanges: one with Alice and one with Bob, allowing Mallory to decrypt, modify and re encrypt messages.

Mitigation; Use authenticated key exchange protocols such as Diffie Hellman with Digital signatures on public keys infrastructure (PKI)

Scanned with CamScanner

Eavesdropping Attack:

An attacker passively intercepting the communication can't derive the shared secret without solving the discrete logarithm problem.

Mitigation: Ensure a sufficiently large prime modulus to prevent brute force - attacks.

② Role of the Discrete Logarithm Problem (DLP)

The security of the Diffie Hellman protocol fundamentally relies on the hardness of the Discrete Logarithm Problem (DLP):

ⓐ Discrete Logarithm Problem (DLP):

→ Given a large prime p and generator g, computing x from $g^x \bmod p$ is computationally infeasible.

Mitigation: An attacker could efficiently solve DLP by computing the secret key g^{ab} by first deriving a or b.

Scanned with CamScanner

IT 23618

The impact on the protocol's security if prime & modulus used is not sufficiently large:

→ Brute - Force Attacks Become Feasible:

Attacks can precompute logarithm tables for small primes and break the key exchange.

Quantum Computing threat: With Shor's algorithm a quantum computer can efficiently solve discrete logarithms, breaking DH security.

Mitigation: Use large p safe primes (e.g. 2048-bit or higher) and consider Elliptic curve Diffie-Hellman (ECDH), which provides equivalent security with smaller key sizes.

Answer to the question no 08:

Proof: The intersection of two subgroups is a subgroup

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

P.T.O

Scanned with CamScanner

Step1: Non emptiness

Since H and K are subgroups of G, they both contain the identity element e of G. Hence, $e \in H \cap K$ so, $H \cap K$ is non empty.

Step2: Closure under the group operation

Let, $a, b \in H \cap K$. By definition, this means that a, b are elements of both H and K. Since H and K are subgroups, they are closed under the group operation, so $ab \in H$ and $ab \in K$. Therefore $ab \in H \cap K$, showing $H \cap K$ is closed under the group operation.

Step3: Closure under Inverses

Let $a \in H \cap K$. Then a is in both H and K. Since subgroups are closed under taking inverses, we have $a^{-1} \in H$ and $a^{-1} \in K$. Thus $a^{-1} \in H \cap K$, proving that $H \cap K$ is closed under taking inverses.

So, we can say $H \cap K$ is also a subgroup of G

Scanned with CamScanner

IT 23618

Example: Intersection of two Subgroups in:

Consider the group Z (integers under addi

and the subgroups:

→ $H = 2Z = \{\ldots -4, -2, 0, 2, 4, \ldots\}$, the set of even integers

→ $K = 3Z = \{\ldots -6, -3, 0, 3, 6, \ldots\}$, the set of multiple of 3

Their intersection is:

$H \cap K = 6Z = \{\ldots -6, 0, 6, 12, \ldots\}$, which consists of the multiples of 6. Since 6Z is closed under addition and inverses, it is a subgroup of Z, confirming our result.

This demonstrates that the intersection of subgroups remains a subgroup (Proved).

Answer to the question no09:

Proof that Z_n is Commutative: The ring Z_n consists of the set of integers modulo n, denoted as:

$$Z_n = \{0, 1, 2, \ldots, n-1\}$$

with addition and multiplication defined modulo n.

Scanned with CamScanner

Commutativity of addition:

For any $a, b, \in Z_n$, we have

$$a+b \equiv b+a \pmod{n}$$

which follows from the commutativity of addition in Z. Thus addition Z_n is commutative.

Commutativity of multiplication: Simillarly, for multiplication:

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

which follows from the commutativity of multiplication in Z. Thus, multiplication in Z_n is also commutative

Since both operations are commutative, Z_n is a commutative ring.

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

$$a.b \equiv 0 \pmod{n}$$

for some $a, b \neq 0$ in Z_n

In Z_n, an element a has a zero divisor if and only if there exists a nonzero b such that:

$$ab \equiv 0 \pmod{n}$$

this occurs precisely when a and n share a nontrivial common factor, i.e. $\gcd(a,n) \neq 1$.

Thus, Z_n has zero divisors whenever n is

P.T.O.

composite (i.e. not prime), since in this case there exists nonzero elements a that are relatively prime to n.

For example:

→ In Z_6, the element 2 is a zero divisor because $2 \times 3 \equiv 0 \mod 6$

→ In Z_7, there is no zero divisor because 7 is prime.

Conditions for Z_n to be a field:

A field is a commutative ring where every non-zero element has a multiplicative inverse.

In Z_n, an element a has a multiplicative inverse if and only if there exists b such that:

$$ab \equiv 1 \pmod{n}$$

This equation has a solution if and only if $\gcd(a,n)=1$

for Z_n to be a field, every nonzero element must have an inverse. This happens if and only if every non zero element is coprime to n, which is true only when n is prime.

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

Scanned with CamScanner

For example:

→ Z_7 is a field because 7 is prime and every non zero element 1, 2, ... 6, has a multiplicati -ve inverse.

→ Z_6 is not a field because gcd (2,6) = 2 and gcd (3,6) = 3, meaning 2 and 3 do not have inverses

Answer to the question no 10:

Vulnerabilities of DES:

① Short key length: The most significant vulnerability of DES lies in its key length. DES uses a 56 bit key, which offers a total of 2^{56} ~~key~~ possible keys.

② Brute force attacks: Brute force attacks refer to the attempt to try all possible keys until the correct one is found. Given that DES has a 56-bit key, an attacker can try 2^{56} keys. Although this number seems large, it's manageable with modern hardware.

Scanned with CamScanner

IT 23618

In 1998, the EFF DES cracker was able break DES in 56 hours using a dedicated machine designed for brute force attacks.

③ Cryptanalytic attacks: In addition to brute force attacks, there are cryptanalytic methods that can exploit weaknesses in DES's design.

④ Feistel Structure Limitations: The Feistel structure used in DES is now considered weak for modern cryptanalytic techniques.

DES is considered insecure for modern use:

Due to its short key length, susceptibility to brute force and cryptanalytic attacks, and the availability of more secure algorithms, DES is considered insecure for modern modern encryption needs.

Scanned with CamScanner

Role of Brute Force Attacks in breaking DES;

For a brute force attack to break DES, the attacker must try every possible key until the correct one is identified. The attacker intercepts an encrypted message (ciphertext) and systematically decrypts it using all possible keys, checking for a meaningful plaintext.

For DES:

→ Key space size: Since DES uses a 56 bit key, the key space is 2^{56} possible keys

Brute force Process: The Brute Force attacker will attempt to decrypt the ciphertext using each of the 2^{56} possible keys and verify if the resulting plaintext makes sense (i.e. if it's readable or conforms to some known format)

Computation time: With enough computational resources, an attacker can eventually find the correct,

P.T.O.

Scanned with CamScanner

key by exhaustively trying all ~~possibles~~ po

Impact of key length on DES Security:

The strength of a symmetric-key algorithm is directly tied to the size of its key, as the number of possible keys determines how difficult it is to perform a brute force attack.

→ 56-bit key in DES provides 2^{56} possible keys, which, although difficult to break using conventional methods in the 1970s and 1980s, is far too small by modern standards.

→ As computational power grew, it became increasingly clear that DES's 56 bit key was insufficient to withstand brute-force attacks.

→ Key length is a major factor in the security of a cipher. For instance, a 128-bit key provides 2^{128} possible keys, which is orders of magnitude more difficult to break brute force compared to a 56 bit key.

Scanned with CamScanner

The development of the AES was specifically aimed at addressing the shortcomings of DES particularly regarding key size, ~~security against~~ and resistance to cryptanalytic attacks.

Here's how AES improved upon DES:

① Larger key sizes:

AES supports key sizes of 128 bits, 192 bits and 256 bits. These key sizes provide a vastly larger key space compared to DES's 56 bit key:

For AES-128: 2^{128} possible keys

For AES-192: 2^{192} " "

For AES-256: 2^{256} " "

The significant larger key sizes make AES resistant to brute force attacks, as the computional effort required to break AES would take million of years with current technology.

② More robust against Cryptanalysis: AES was designed to resist both differential cryptanalysis, which were some of the most powerful methods used against DES

Scanned with CamScanner

→ AES's structure known as the Substitution-Permutation Network (SPN), to which makes AES ~~fare~~ more secure than DES.

③ Block size: While DES operates on 64-bit blocks, AES operates on 128 bit blocks. Larger block sizes provide greater resistance to certain attacks such as birthday paradox attacks.

④ Efficiency: AES is optimized for modern hardware and can be efficiently implemented on a wide range of platforms, including hardware accelerators and software libraries. It is designed to be faster than DES.

Answer to the question no 11;

ⓘ Handling Differential Cryptanalysis in the Feistel structure:

Differential Cryptanalysis works by analyzing how how differences in input pairs propagate through the cipher and how those differences affect the output. In particular, it tries to trace

P.T.O.

how differences at the input (plaintext) stage ~~after the output~~. affect the differences in the output (ciphertext) after several rounds of encryption.

→ In a Feistel structure, each round consists of applying a function to one half of the block and then XORing it with the other half. However, because of the way the round function works (specially with the use of S-boxes and X-OR operation), differential cryptanalysis becomes difficult to apply because:

① Key dependency: Each round's function depends on a different subkey, so an attacker trying to find a differential path through the cipher must account for different keys used in each round, making the analysis more complex.

② X-ORing ~~the~~ of Halves: The Feistel structure combines the results of the round function with the other half of the block, and the nonlinear S-boxes further complicate how differences

propagate. The structure provides partial diffusion across the rounds but not full di in one round.

(3) (iii) Limited round count: DES has 16 rounds. while differential cryptanalysis can be effective in some cases, it is harder ~~than~~ to ~~find~~ find exploitable differential paths in DES when considering the complete set of 16 rounds and the key schedule

That being said, differential cryptanalysis can still be effective against DES because it is relatively susceptible to this attack, particularly if the key size is small.

(ii) AES is much more efficient resistant to differential cryptanalysis due to its design, which incorporates several layers of non-linearity and confusion in its operations, making it more difficult for an attacker to track how difference propagate through the cipher.

Scanned with CamScanner

SubBytes: This step uses a non-linear S-box which is designed to significantly disrupt the propagation of differences. The S-box is carefully chosen to have desirable cryptographic properties, like being invertible and providing resistance to differential cryptanalysis.

Shift Rows: This steps involves shifting the rows of the state matrix, which provides diffusion and spreads out the bits across the state. This action makes it harder for attackers to predict how a difference in one part of the data will influence the final ciphertext.

Mixcolumns: This operation mixes the data in each column, spreading the influence of each bit across the entire state. This is an essential feature but because it ensures that small differences in the input are not easily exploited through differential analysis.

Scanned with CamScanner

IT 23619

Add Round key: In this operation, the ro
is X-ORed with the state, further mixing
data. Since each round uses a different
round key, it adds additional layers of
complexity to the cipher.

The combination of these operations in AES
increases the complexity and randomness
of the cipher, making it much harder to
analyze and exploit any predictable patterns
in how plaintext differences propagate through
the cipher. The more rounds (10, 12, or 14
rounds depending on the key size) and the higher
complexity of operations in AES compared to
DES.

Scanned with CamScanner

Answer to the question no 12:

Finding the modular Inverse using the Extended Euclidean Algorithm:

In modular inverse of an integer a modulo n exists if and only if a and n are co-prime. (i.e. $\gcd(a,n)=1$). The modular inverse is an integer x such that:

$$ax \equiv 1 \pmod{n}$$

To find the inverse, we use the Extended Euclidean Algorithm, which finds integer coefficients x and y satisfying Bézout's identity:

$$ax + ny = \gcd(a,n)$$

Since $\gcd(a,n)=1$, this simplifies to:

$$ax + ny = 1$$

Reducing modulo n, we get:

$$ax \equiv 1 \pmod{n}$$

Thus, x (modulo n) is the modular inverse of a.

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

① Apply the Euclidean Algorithm to $\gcd(3,26)$:

P.T.O.

Scanned with CamScanner

$26 = 3(8) + 2$

$3 = 2(1) + 1$

$2 = 1(2) + 0$

Since the last nonzero remainder is 1, gcd(3,

21 so the modular inverse exists

② Use Back substitution in the Extended Euclidean algorithm: From $3 = 2(1) + 1$, rewrite 1 in terms of 3 and 2:

$$1 = 3 - 1(2)$$

from $26 = 3(8) + 2$, solve for 2:

$$2 = 26 - 3(8)$$

Substitute into $1 = 3 - 1(2)$:

$$1 = 3 - 1(26 - 3(8))$$

$$1 = 3(9) - 1(26)$$

comparing with $1 = 3x + 26y$, we identify $x = 9$,

So: $3^{-1} \equiv 9 \pmod{26}$

③ Verify the solution:

$$3 \times 9 = 27 \equiv 1 \pmod{26}$$

Thus, 9 is indeed the modular inverse.

Scanned with CamScanner

Utilization in RSA Key Generation: The RSA crypto-system relies on modular arithmetic and the Extended Euclidean Algorithm in the key generation process:

① Choose two large Prime Numbers p and q

② Compute $n = p \times q$

③ $\phi(n) = (p-1)(q-1)$

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

⑤ compute the private key d which is the modular inverse of e modulo $\phi(n)$. i.e find d such that $ed \equiv 1 \mod (\phi(n))$

Importance of efficiency in Cryptographic Systems:

① Handling Large numbers: In RSA, the values of p and q are typically 1024-bit to 4096-bit primes, making n extremely large. The extended ~~Eute~~ Euclidean algorithm is efficient, running in $O(\log n)$ time, making it practical even

P.T.O.

Scanned with CamScanner

even for every large values.

② Key Generation Speed; Since RSA relies on finding modular inverses during key generation, the speed of this process is crucial. A slow algorithm would make key generation impractical for real world application.

③ Security and performance; Cryptographic systems need fast modular inverses to handle encryption and decryption efficiently. Delays in modular inversion ~~can~~ could slow down secure communications and digital signatures.

Answer to the question no 13;

① ECB Mode Insecurity for Highly Redundant Data;

The Electronic Codebook (ECB) mode encrypts each plaintext block independently using the same key;

$$c_i = E_K(P_i)$$

Scanned with CamScanner

where,

E_K is the encryption function under key K, P_i is the plaintext block, C_i is the corresponding ciphertext block.

ECB mode has a critical security flaw: identical plaintext blocks produce identical ciphertext blocks because encryption is deterministic.

<u>Proof:</u> ECB mode is insecure for encrypting highly redundant data:

Let, $P_1, P_2 \cdots P_m$ be plaintext blocks, and assume two blocks are identical:

$$P_i = P_j \text{ for some } i \neq j$$

Then their ciphertexts will also be identical:

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

This implies that ECB mode fails to provide semantic security, meaning an attacker observing the ciphertext can detect patterns in plaintext redundancy. This is particularly problematic for images or structured

P.T.O.

Scanned with CamScanner

data, where repeating patterns appear repeated ciphertext blocks.

Example: In an encrypted image using ECB, regions with the same color ~~(pixel~~ will map to the same ciphertext blocks, preserving the structure of the original image and making it visually discernible. Thus, ECB mode is insecure for encrypting redundant data because it fails to hide patterns in plaintext, leading to potential leakage of information.

(11) CBC mode recurrence Relations and Error propagation Analysis :

CBC Mode Encryption Recurrence Relation:

Cipher Block Chaining (CBC) mode introduces feedback by X-ORing each plaintext block with the previous ciphertext block before encryption: $C_i = E_K(P_i \oplus C_{i-1})$

Scanned with CamScanner

for $i \geq 1$ with the initialization vector (IV) as

$$C_0 = IV$$

where,

IV = randomly chosen initialization vector

C_i = ciphertext block at index i

P_i = the plaintext block at index i

E_K = encryption function under key K.

Expanding iteratively:

$$C_1 = E_K(P_1 \oplus IV)$$
$$C_2 = E_K(P_2 \oplus C_1)$$
$$C_3 = E_K(P_3 \oplus C_2)$$
$$\vdots$$

Thus CBC ensures that identical plaintext blocks encrypted differently due to chaining, mitigating the weaknesses of ECB

Decryption Recurrence Relation:

Decryption follows the inverse process:

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

where, D_K is the decryption function under key K.

Expanding iteratively:

$$P_1 = D_K(C_1) \oplus IV$$
$$P_2 = D_K(C_2) \oplus C_1$$
$$P_3 = D_K(C_3) \oplus C_2$$

Scanned with CamScanner

IT 23618

Each plaintext block is recovered by d. decrypting its corresponding ciphertext and X-ORing it with the previous ciphertex

Error propagation in CBC Mode: Suppose a single bit error occurs in C_i during transmission. Let's analyze its effect on decryption;

① Effect on P_i: Since decryption is;

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

an error in C_i directly corrupts $D_K(C_i)$, leading to a completely incorrect P_i due to the non-linearity of D_K.

② Effect on P_{i+1}: The next plaintext block depends on;

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

Since only C_i is corrupted, the X-OR operation introduces only a single bit error in P_{i+1}

③ Effect on ~~for~~ Subsequent Blocks: Blocks P_{i+2}, P_{i+3}, ... remain unaffected because they depend only on unaltered ciphertext blocks.

Scanned with CamScanner

Answer to the question no 14:

The inherent linearity of LFSRs makes them vulnerable to known-plaintext attacks because an attacker can reconstruct the LFSR's internal state using algebraic methods. This allows full recovery of past and future keystream bits, rendering simple LFSR-based ciphers insecure. To enhance security, nonlinearity must be introduced into LFSR-based cryptographic designs.

Mathematical Mitigation: Nonlinearity Introduction

To mitigate this vulnerability, we must break the linearity of the key strem generation. One common approach is to introduce nonlinear transformations. Here's a mathematical approach:

Nonlinear Combination of Multiple LFSRs:

Instead of using a single LFSR, we can use multiple LFSRs and combine their outputs through

Scanned with CamScanner

a nonlinear Boolean function.

① Let, $L_1, L_2, \ldots L_m$ be m LFSRs, each with different primitive polynomials and lengths.

② Define a nonlinear Boolean function f: $\{0,1\}^m \rightarrow \{0,1\}$ such that:

$$z_t = f(x_t^{(1)}, x_t^{(2)} \ldots x_t^{(m)})$$

where $x_t^{(i)}$ is the output bit of the ith LFSR at time t, and z_t is the resulting keystream bit.

③ The function f should be designed to have high nonlinearity and low correlation with linear approximations to resist correlation attacks.

Ex

Answer to the question no 15 :

(i) Shannon's Definition of perfect Secrecy:

Shannon defined a cryptographic system as having perfect secrecy if the ciphertext doesn't reveal any information about the plaintext. Mathematically, this is expressed as:

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

for all plaintexts $m \in M$, all ciphertexts $c \in C$ and all keys $k \in K$ with nonzero probability.

Equivalently, in terms of conditional probabilities

$$\cancel{P(C=c \mid M=m}$$

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

for all $m \in M$ and $c \in C$. This means that knowing the ciphertext c doesn't change the probability distribution of the plaintext m, ensuring complete confidentiality

(ii) Proof that the one time pad Achieves Perfect Secrecy:

To prove that the one-time Pad (OTP) satisfies Shannon's definition to perfect secrecy, consider;

→ plaintext space: M with messages $m \in M$

→ key space: K, where each key k is chosen uniformly at random.

P.T.O.

→ Ciphertext space: $\mathcal{C}$, where each cipher

c is computed as:

$$c = E_k(m) = m \oplus k$$

where $\oplus$ denotes bitwise XOR

Step 1: Compute the probability of a given ciphertext given a plaintext.

$$P(C = c \mid M = m) = P(K = c \oplus m)$$

Since the key k is chosen uniformly at random from K, and assuming $|K| \geq |M|$, every key is equally likely. Therefore,

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

for all $k \in K$. This implies:

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

Step 2: Compute the probability of a given ciphertext.

Since every plaintext m maps uniquely to a ciphertext c under some key k, and since the key is chosen uniformly, the probability of obtaining a specific ciphertext c is:

$$P(C = c) = \sum_{m \in M} P(C = c \mid M = m)\, P(M = m)$$

Using the assumption that plaintexts are chosen

P.T.O.

Scanned with CamScanner

uniformly, $P(M=m) = \frac{1}{|M|}$, so:

$$P(C=c) = \sum_{m \in M} \frac{1}{|K|} \cdot \frac{1}{|M|}$$

Summing over all m, and noting that each ciphertext c is equally likely due to the uniform key distribution:

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

Step 3: Verify Shannon's Condition

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

Substituting values,

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

which is exactly $P(M=m)$. This proves that knowing $C=c$ doesn't change the probability distribution of M, satisfying perfect secrecy.

(iii) Perfect secrecy is Impractical for Large Scale communication:

Although the One time Pad achieves perfect secrecy, it is impractical for large scale communication due to several reasons:

Scanned with CamScanner

IT 23618

① Key Distribution Problem: The sender and must share a key as ~~as~~ long as the message. This requires a secure channel for key distribution, which often negates the advantage of encryption.

② Key Storage overhead: Since the key must be at least as long as the message and never reused, storing and managing keys for large scale communication (~~e.g.~~ is infeasible.

③ No reuse of keys: If a key is reused, security is compriised since an attacker can compute relationships between different ciphertexts. This makes it impractical for systems requiring frequent communications.

④ Computational Constraints; Modern cryptographic systems often rely on computional hardness assumptions (e.g. RSA, AES) rather than perfect secrecy. These allow shorter keys while maintaining practical security.

Scanned with CamScanner

⑤ Scalability issues: Secure key exchange and management become increasingly difficult as the number of users grows, making OTP unsuitable for large networks.

Answer to the question no 16:

Lets use specific values for the parameters of the linear Congruential Generator (LCG):

multiplier, $a = 5$

increment, $c = 3$

modulus, $m = 16$

Seed, $x_0 = 7$

The recurrence relation is

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

Now, Let's compute the first 5 numbers:

① $x_1 = (5 \times 7 + 3) \bmod 16 = (35+3) \bmod 16 = 38 \bmod 16 = 6$

② $x_2 = (5 \times 6 + 3) \bmod 16 = (30+3) \bmod 16 = 33 \bmod 16 = 1$

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

④ $x_4 = (5 \times 8 + 3) \bmod 16 = (40+3) \bmod 16 = 43 \bmod 16 = 11$

⑤ $x_5 = (5 \times 11 + 3) \bmod 16 = (55+3) \bmod 16 = 58 \bmod 16 = 10$

∴ Final LCG sequence: 7, 6, 1, 8, 11, 10

Scanned with CamScanner

IT 23618

Answer to the question no 17:

Ring: A ring $(R, +, \cdot)$ is an algebraic struc consisting of a set R equipped with two bina operations, typically called addition $+$ and multiplication $\cdot$.

Key properties of a ring:

1) $(R, +)$ is an Abelian Group:

Closure: for all $a, b \in R$, $a+b \in R$

Associavity: $(a+b)+c = a+(b+c)$

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

Additive inverse: For every $a \in R$, there exists an element $-a \in R$ such that $a+(-a) = 0$ for all ~~$a \in R$~~

Commutativity: $a+b = b+a$

Scanned with CamScanner

② Multiplication is associative:

for all $a, b, c \in R$, $(a.b).c = a.(b.c)$

③ Left and right distributive laws:

→ $a.(b+c) = a.b + a.c$

→ $(a+b).c = a.c + b.c$

④ Additional properties (optional but common in Rings):

→ A ring is commutative if $a.b = b.a$ for all $a, b \in R$.

→ A ring has a multiplicative identity (or unity) if there exists an element $1 \neq 0$ in R such that $a.1 = a$ for all $a \in R$

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

→ A ring is a field if it is a commutative division ring.

Commutative ring example: Z/nZ (Integers modulo n)

The set of integers modulo n, denoted Z/nZ, is a commutative ring with addition and multiplication modulo n. For instance, $Z/6Z = \{0, 1, 2, 3, 4, 5\}$ with addition and multiplication

P.T.O.

Scanned with CamScanner

modulo 6 satisfies all ring properties and commutative.

Non - Commutative Ring Example: $M_2(R)$ (Matrix Ring)

The set of all 2×2 matrices with real entries, denoted $M_2(R)$, is a non commutative ring under matrix addition and multiplication.

That is,

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, B = \begin{bmatrix} 2 & 0 \\ 1 & 3 \end{bmatrix}$$

do not satisfy $A \cdot B = B \cdot A$, hence multiplication is not commutative

Relation between Rings and Finite Fields:

A finite field (or Galois Field $GF(q)$) is a special type of ring where every nonzero element has a multiplicative inverse, making it a field. Finite fields are constructed from rings as follows:

① Starting from a ring: Consider, the ring $\mathbb{Z}/p\mathbb{Z}$ where p is a prime number.

Scanned with CamScanner

② Checking multiplicative inverse: If p is prime, then every nonzero element in $\mathbb{Z}/p\mathbb{Z}$ has an inverse, making it a field.

③ Constructing Larger fields: More general finite fields $GF(q)$ (where $q = p^n$ for some prime p) can be constructed using polynomial rings $\mathbb{Z}/p\mathbb{Z}[x]$ modulo an irreducible polynomial.

Finite fields play a crucial role in cryptographic applications, such as the AES encryption algorithm and elliptic curve cryptography.

Role of Rings in ~~Cryptogra~~ Cryptographic Algorithms (Example: RSA)

The RSA encryption scheme relies on properties of rings and modular arithmetic:

① Ring structure: RSA works in the ring $\mathbb{Z}/n\mathbb{Z}$ where $n = p \cdot q$ is the product of two large prime numbers.

P.T.O.

Scanned with CamScanner

② Key Generation :

→ select large prime numbers p, q

→ compute $n = p.q$ and $\phi(n) = (p-1)(q-1)$

→ choose an encryption exponent e such that $\gcd(e, \phi(n)) = 1$.

→ Compute decryption exponent d as the modular inverse of e modulo $\phi(n)$.

③ Encryption; Given a plaintext message m, the ciphertext is computed as;

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

④ Decryption; The original message is retrieved using; $m = c^d \bmod n$

Answer to the question no 18;

Given, $p = 5$, $q = 11$

$\therefore n = p.q = 5 \times 11 = 55$

$\phi(n) = (p-1)(q-1) = (5-1)(11-1) = 4 \times 10 = 40$

Choose public key exponent e such that $1 \le e \le \phi(n)$ 40

common choices are $e = 3, 5, 7, 17$

Scanned with CamScanner

IT 23618

Lets choose $e = 3$ (which satisfies $\gcd(3, 40) = 1$)

Private key: The private key d is the modular multiplicative inverse of mod $\phi(n)$

$\therefore e \cdot d \equiv 1 \mod \phi(n)$

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

$\therefore d \equiv 3^{-1} \mod 40$

~~40~~ 41, 81, 121, ...

$\therefore 3 \times 27 \equiv 1 \mod 40$

$\therefore d = 27$

Encryption: Given, the message, $M = 2$

$\therefore$ using encryption formula

ciphertext, $C = M^e \mod n$

$= 2^3 \mod 55$

$= 8$

N.B:

$8 \overline{)55}\ (6$
$\quad 48$

$55 \overline{)8}\ (0$
$\quad 0$
$\quad 8$ → remainder

$8 \mod 55 = 8$

Decryption:

$M = c^d \mod n$

$= 8^{27} \mod 55$

$64 \mod 55$

$55 \overline{)64}\ (1$
$\quad 55$
$\quad 9$

$= 9$

$\therefore 9 \mod 55$

Scanned with CamScanner

IT 23618

$8^2 = 64 \equiv 9 \mod 55$

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

$8^8 = (8^4)^2 = (26)^2 \equiv 676 \mod 55$

$8^{16} = (8^8)^2 \equiv (16)^2 \equiv 36 \mod 55$

$\therefore 8^{27} = 8^{16} \times 8^8 \times 8^2 \times 8 \equiv 36 \times 16 \times 9 \times 8 \mod 55$

Stepwise computation:

$36 \times 16 = 576 \equiv 26 \mod 55$

$26 \times 9 = 234 \equiv 14 \mod 55$

$14 \times 8 = 2 \mod 55$

$\therefore 2 \mod 55 = 2$

$\therefore m = 2.$

$8 \mod 55 = 8$

$64 \mod 55$

$= 9 \mod 55$

$81 \mod 55$

$55)\,81\,(1$
$\quad 55$
$\quad 26$

$676 \mod 55$

$55)\,676\,(12$
$\quad 55$
$\quad 126$
$\quad 110$
$\quad 16$

$(16)^2 = 256 \mod 55$

$55)\,256\,(4$
$\quad 220$
$\quad 36$

$576 \mod 55$

$55)\,576\,(10$
$\quad 550$
$\quad 26$

$234 \mod 55$

$55)\,234\,(4$
$\quad 220$
$\quad 14$

$112 \mod 55$

$55)\,112\,(2$
$\quad 110$
$\quad 2$

$2 \mod 55$

$55)\,2\,(0$
$\quad 0$
$\quad 2$

Scanned with CamScanner

Signing and verifying a message:

Given, $p = 7, q = 3$

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

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

choose, $e = 5$ (which satisfies $\gcd(5, 12) = 1$)

$d = e^{-1} \bmod \phi(n)$

$= 5^{-1} \bmod 12$

$13, 25, 37, \ldots$

$\therefore 5 \times 5 = 25 \equiv 1 \bmod 12$

$\therefore d = 5$

$12 \times 1 + 1 = 13$
$12 \times 2 + 1 = 25$
$12 \times 3 + 1 = 37$

$12 \overline{)25} (2$
$\underline{24}$
1

Given, $H(m) = 3$

The signature S is computed as;

$S = H(m)^d \bmod n$

$= 3^5 \bmod 21$

$= 243 \bmod 21$

$= 12$

$21 \overline{)243} (11$
$\underline{21}$
33
$\underline{21}$
12

So, the signature, $S = 12$

Scanned with CamScanner

IT 23618

Verify the signature;

$H(m)' = S^e \bmod n$

$= 12^5 \bmod 21$

$= (12^4 \times 12) \bmod 21$

$= 3 \bmod 21$

$= 3$

$\therefore H(m)' = H(m) = 3$, ϕ

$\therefore$ the signature is valid.

Here,

$12^5 = 12^4 \times 12$

$= 12^2 = 144 \equiv 18 \bmod 21$

$21 \overline{)144} (6$
126
18

$12^4 = (12^2)^2 = (18)^2 \bmod$
$= 324$

$21 \overline{)324} (15$
21
114
105
9

$\therefore 12^4 = 9 \bmod 21$

$(12)^5 = (12)^4 \times 12$
$= 9 \times 12 = 108$

$21 \overline{)108} (5$
105
3

$\therefore 12^5 = 108 \equiv 3 \bmod 21$

Digital signature Ensures Integrity and Authenticity.

① Integrity: The Hash H(m) ensures that any change to the message will result in a different hash.

→ Since the signature is computed using the hash, altering the message invalidates the signature.

Scanned with CamScanner

② Authenticity;

→ The private key is known only to the signer, ensuring that the legitimate sender could have generated the signature.

→ The public key allows anyone to verify the signature.

③ Non Repudiation;

→ The signer can't deny creating the signature because it is uniquely tied to their private key.

Answer to the question no - 19:

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

ⓐ ∴ the curve equation is

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

∵ Point $P(3, 10)$

∴ puts $x = 3, y = 10$ into the curve equation:

$$10^2 \equiv 3^3 + 3 + 1 \mod 23$$

$$\Rightarrow 100 \equiv 27 + 3 + 1 \mod 23$$

$$\Rightarrow 100 \equiv 31 \mod 23 \Rightarrow 100 \not\equiv 8$$

[31 mod 23 = 8

P.T.O.

~~since 31 ≡ 8 mod~~

N.B:

$100 \equiv 31 \bmod 23$

$23)\ 100\ (4$
$\quad 92$
$\quad 8$

$23)\ 100\ (4$
$\quad 92$
$\quad 8$

$\therefore\ 100 \not\equiv 31 \bmod 23$

Therefore we can say that, $P(3,10)$ does not satisfy the curve equation

Thus, the given ~~curve~~ point $P(3,10)$ does not lie on the curve.

(ii) Compute $2P$ (point doubling):

For point doubling, we use the formula:

$$\lambda = \frac{3x^2+a}{2y} \bmod p \quad \text{--- (1)}$$

puts $x=3$, $y=10$ and $a=1$ in equation (1)

$$\lambda = \frac{3(3)^2+1}{2.10} \bmod 23$$

$$= \frac{3.9+1}{20} \bmod 23$$

$$= \frac{27+1}{20} \bmod 23$$

Scanned with CamScanner

IT 23618

Now, we have to compute $20^{-1} \bmod 23$, we find the modular inverse of 20 modulo 23.

∴ According to Euclidean algorithm:

$$20^{-1} \equiv 14 \bmod 23$$

~~Thus, $\lambda =$~~

$$28 \times 14 \equiv 1 \bmod 23$$

$$\therefore \lambda = 1 \bmod 23$$

Now, we compute x_3 and y_3:

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

$$= 1^2 - 2(3) \bmod 23$$

$$x_3 = 1 - 6 \bmod 23$$

$$x_3 = -5 \bmod 23$$

$$\therefore x_3 = -5 \bmod 23$$

$$\therefore x_3 = 18$$

for y_3: $y_3 = \lambda(x_1 - x_3) - y_1 \bmod p$

$$y_3 = 1(3-18) - 10 \bmod 23$$

$$= (3-18-10) \bmod 23$$

N.B

$18)\ 23\ (1$
$\quad 18$
$\quad 5$

$23)\ 18\ (1$
$\quad 23$
$\quad -5$

$23)\ -5\ (1$
$\quad 23$
$\quad 18$

P.T.O.

Scanned with CamScanner

$y_3 = \cancel{-25}\ (3-28) \bmod 23$

$= -25 \bmod 23$

$= 21$

We have to ... positive ...

$23)\ -25\ (2$
$\quad 46$
$\quad -2$

$23)\ -25\ (2$
$\quad 46$
$\quad 21$

$\therefore$ The result of doubling is:

$2P = (18, 21)$

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

Soln: $\because P \neq Q$

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

$= \dfrac{7-10}{9-3} \bmod 23$

$= \dfrac{-3}{6} \bmod 23$

$= (-3) \times 6^{-1} \bmod 23$

$= (-3) \times 4 \bmod 23$

$= -12 \bmod 23$

$= 11$

$6^{-1} \bmod 23$

$(23 \times 1) + 1 = 24$

$23)\ 24\ (1$
$\quad 23$
$\quad 1$

$\therefore 6 \times 4 = 24$
$\therefore 6 \times 4 \equiv 1 \bmod 23$
$\therefore 6^{-1} \bmod 23 = 4$

$23)\ -12\ (1$
$\quad 23$
$\quad 11$

Scanned with CamScanner

Now,

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

$= (11)^2 - 3 - 9 \bmod 23$

$= (121 - 12) \bmod 23$

$= 109 \bmod 23$

$= 12$

$23)\,109\,(4$
$\quad 92$
$\quad 17$

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

$= 11\,(3 - 17) - 10 \bmod 23$

$= (11 \times -14) - 10 \bmod 23$

$= -154 - 10 \bmod 23$

$= -164 \bmod 23$

$= 20$

$23)\,164\,(8$
$\quad 184$
$\quad 20$

$\therefore$ The result of $P + Q = (17, 20)$

Scanned with CamScanner

Answer to the question no 21:

① Essential characteristics of a secure hash function:

A secure hash function must possess key properties to be considered cryptographically secure:

① Pre-image resistance:

→ This means that given a hash value (output), it should be ~~considered~~ computationally infeasible to find any input (pre-image) that hases to that value.

→ For a hash function H, given $h = H(x)$, it should be hard to find x such that $H(x) = h$.

→ Pre image resistance prevents attackers from reversing the hash function to obtain the original input.

② Second Pre-image Resistance:

→ This means that given an Input x_1, it should be infeasible to find another distinct input x_2 such that $H(x_1) = H(x_2)$

Scanned with CamScanner

→ This property ensures that an attacker can't create two different inputs that produce the same hash value.

③ Collision Resistance:

→ This means that it should be infeasible to find any two distinct inputs x_1 and x_2 such that $H(x_1) = H(x_2)$

→ Collision resistance is critical because finding collisions could undermine the security of systems that rely on the uniqueness of hash values. (e.g. Digital signature)

④ Deterministic:

The same ~~output~~ input always produces the same output.

⑤ Fast computation

⑥ Avalanche effect:

A small change in the input (even a single bit) should result in a completely different hash output. This property ensures that simillar inputs

P.T.O.

Scanned with CamScanner

don't produce simillar hash values, which important for security.

② Fixed output length :

Regardless of the size of the input, the output hash should always be of a fixed length (e.g. 256 bits for SHA-256).

(ii) Impact of Output length on Security:

The length of the output hash significantly affects the security of a hash function in terms of its resistance to collision and brute force attack;

① Collision resistance:

→ The longer the output hash, the more resistant it is to collisions. This is due to birthday paradox, which states that for a output length of n bits, it takes approximately $2^{n/2}$ operations to find a collision. Therefore, longer hashes have exponentially greater collision resistance.

→ For instance, with a 256-bit output (as in SHA-256)

P.T.O.

Scanned with CamScanner

, it would take around 2^{128} operations to find a ~~collisi~~ collision. In constant, a shorter hash output (e.g. 128 bits) would require only 2^{64} operations to find a collision, which is easier to achieve

(2) Brute Force Attack Resistance:

The length of the hash also affects the difficulty of a brute-force attack. For a given input, an attacker must try 2^n different values (on average) to find a matching hash if the hash is n-bits long.

→ A 256-bit output (like SHA-256) provides a very large search space, making brute force attacks impractical.

Thus, the output hash length is crutial for ensuring that the hash function remains secure as computational power increases over time.

Scanned with CamScanner

(iii) Real world applications of SHA :

SHA functions, particularly SHA-256, are widely used in a variety of real-world applications, such as:

① Digital signatures:

Digital signatures use hash functions to create a unique fingerprint (hash) of a message. The hash is then signed with a private key, allowing the recéplent to verify the integrity and authenticity of the message by checking the signature with the public key.

→ SHA-256 is commonly used in this context (e.g. in the RSA and ECDSA algorithms).

② Blockchain Systems:

→ In blockchain, hash functions are used extensively for integrity verification and linking blocks. Each block contains a hash of the previous block, forming a chain of blocks that is tamperevident.

Scanned with CamScanner

→ SHA - 256 is used in Bitcoin and other crypto currencies to hash block headers, ensuring that once a block is added the chain, it is nearly impossible to change it without re-mining the block.

→ Hashing also plays a crucial role in proof of work systems.

③ Password storage:

→ Hash functions like SHA-256 are used for securely storing passwords. Instead of saving the plaintext password, systems store the hash of the password. During login, the password entered by the user is hashed, and the result is compared to the stored hash.

→ This process ensures that even if the password database is compromised, attackers can't recover the actual passwords.

④ Data integrity: Hash functions are used to verify the integrity of data during transmission or storage.

Scanned with CamScanner

Answer to the question no 22?

Galois Field / Finite fields: Galois fields (also known as finite fields) are mathematically structures that are fundamental to many areas of cryptography. These fields consist of a finite set of elements and provide a well defined set of operations (addition, subtraction, multiplication and division) that follow certain algebraic rules. Galois fields play a critical role in cryptographic primitives such as Elliptic curve cryptography (ECC) and the AES encryption algorithm.

Use of Galois fields in Cryptography:

① Elliptic Curve Cryptography (ECC): Elliptic curve Cryptography (ECC) is a public key cryptosystem that uses the algebraic structure of elliptic curves over finite fields, ~~typically~~ typically $GF(p)$ or $GF(2^n)$

Scanned with CamScanner

Let's explore the concepts of Galois fields, specially GF(p) and GF(2^n):

GF(p): Galois fields contain a finite number of elements, denoted as GF(q), where q

GF(p): The field ~~con~~ GF(p) consists of the integers modulo a prime number p. This means the elements of GF(p) are the set $\{0, 1, 2, \ldots p-1\}$ and all arithmetic operation are performed modulo p.

Addition: $a+b \bmod p$

Multiplication: $a \times b \bmod p$

Example: In GF(5), the elements are $\{0,1,2,3,4\}$ and arithmatic is done modulo 5.

∴ Addition; $3+4 \bmod 5 = 7 \bmod 5 = 2$

Multiplication: $2 \times 3 \bmod 5 = 6 \bmod 5 = 1$

GF(p) is commonly used in number theoretic cryptography (e.g. RSA, Diffie-Hellman key exchange). because it provides a simple structure with prime number.

Scanned with CamScanner

GF(2^n): The field GF(2^n) consists of 2^n elements and arithmatic is performed modulo an irreducible polynomial of degree n over GF(2), the binary field GF(2) = {0, 1}. The elements of GF(2^n) are polynomials of degree n-1 with binary coefficients, and the operation of addition and multiplication are defined as follows:.

Addition: Polynomial addition in GF(2^n) is done by performing a bitwise XOR Operation.

Multiplication: Done by modulo an irreducible polynomial of degree n.

Example: In GF(2^3), the elements are $\{0, 1, \alpha, \alpha^2, \alpha+1, \alpha^2+1, \alpha^2+\alpha, \alpha^2+\alpha+1\}$, where α is a root of the irreducible polynomial x^3+x+1. Addition and multiplication are based on this polynomial

... communications, digital signature

P.T.O.

Scanned with CamScanner

Use of Galois Fields in Cryptography:

① Elliptic curve Cryptography (ECC) : Effect

Elliptic Curve Cryptography (ECC) is a public key cryptosystem that uses the algebraic structure of elliptic curves over finite fields (typically $GF(p)$ or $GF(2^n)$)

Elliptic curve over $GF(p)$: An Elliptic Curve is defined by an equation of the form;

$$y^2 = x^3 + ax + b$$

over a finite field $GF(p)$, where p is a prime number. The field $GF(p)$ is used to define the coordinates of the points on the curve. The point addition and point doubling operations are performed using field arithmetic and the difficulty of solving the elliptic curve discrete logarithm problem (ECDLP) is based on the properties of the finite field.

Scanned with CamScanner

Elliptic Curves over $GF(2^n)$: In some implementations of ECC, particularly in binary fields like $GF(2^n)$, elliptic curves are defined with different equations, such as:

$$y^2 + xy = ax^3 + ax^2 + b$$

The field arithmetic in these binary fields involves XOR and polynomial reduction, which is suitable for hardware efficient implementations (e.g. in smart cards and IoT devices).

ECC is known for providing strong security with relatively small key sizes, making it efficient for encryption, digital signatures and key exchange.

② AES Encryption Algorithm: The Advanced Encryption Standard (AES) is a symmetric key encryption algorithm that operates on data blocks of 128 bits and uses $GF(2^8)$ for field

[illegible] cryptic communications, digital signature

arithmetic during key expansion and the encryption process.

AES key expansion: The key expansion in AES involves generating a series of round keys from the original encryption key. During this process, field arithmetic $GF(2^8)$ is used for operations like S-box substitution, which is a non linear transformation based on the inverse of elements in $GF(2^8)$. ~~and of~~

AES Encryption: In the main rounds in AES, operations like SubBytes, ShiftRows, MixColumns and AddRoundKey are used. In MixColumn, a matrix multiplication is performed using arithmetic in $GF(2^8)$.

Scanned with CamScanner

Importance of field Arithmetic in Cryptography:

Security: Field arithmetic provides a structured and well understood mathematical framework for cryptographic operations. The hardness of problem such as the discrete logarithm problem (used in ECC) and the computational infeasibility of reversing transformations in AES are based on the properties of finite fields.

Efficiency: Cryptographic operations often require fast arithmetic to be performed over large sets of numbers. Finite fields ~~allow~~ allow for efficient algorithms that can be implemented in hardware and software. For example, the operations of AES are optimized using binary fields arithmetic for fast, parallel computation on modern processors.

Error detection and correction: Finite fields are also used in error detection and correction schemes, which are important in communication and

P.T.O.

Scanned with CamScanner

storage systems.

Answer to the question no 23

① Shortest Vector Problem (SVP): Shortest vector Problem (SVP) is a computational problem in lattice theory where the goal is to ~~be~~ find the shortest non zero vector within a given lattice, typically measured using the Euclidean norm.

Role of SVP in the security of lattice based cryptographic schemes:

① The security of many lattice based cryptographic schemes relies on the hardness of SVP and related problems like the Closest Vector Problem (CVP) and the Learning With Errors (LWE) problem.

② Public key Encryption, digital signatures and homomorphic encryption schemes based on lattices derive their security from the assumption

Scanned with CamScanner

that solving SVP is ~~feasi~~ infeasible for both classical and quantum computers.

(3) Unlike factoring-based problems (RSA) or discrete logarithm problems (ECC), no efficient quantum algorithm is known to solve SVP in polynomial time.

(ii) Lattice based cryptography vs traditional cryptographic schemes ~~like~~ (RSA and ECC) in the context of Shor's algorithm is given below:

Cryptographic Scheme	Security Basis (Based on)	Broken by Shor's Algorithm?
RSA	Integer factorization problem (IFP)	Efficient quantum attack (means it will be completely broken)
ECC	Elliptic Curve Discrete Logarithm problem (ECDLP)	Efficient quantum attack. (will be broken)
Lattice Based	SVP, CVP, LWE	No efficient quantum attack known. (means it remains secure)

Scanned with CamScanner

(iii) Quantum cryptography vs Lattice based cryptography in terms of their goals and underlying principles is given below;

(1) Different goals:

Lattice Based Cryptography: Aims to ~~goal~~ provide post quantum security - i.e cryptographic systems that remain secure even in the presence of quantum computers.

Quantum Cryptography: Uses principles of quantum ~~mechanisms~~ mechanics to achieve cryptographic security, often in a way that is impossible with classical systems.

Scanned with CamScanner

② Underlying principles:

Feature	Lattice Based Cryptography	Quantum Cryptography
Mathematical Basis	Hard Lattice problems (SVP, LWE)	Quantum mechanics superposition, entanglement)
Computational security	Hardness of Lattice problems (assumed secure even for quantum computers).	Uses quantum physics principles rather than computational assumption.
Resistance to Quantum attacks	Believed to be resistant to quantum computers.	Immune to classical computing attacks due to physical Laws.

Answer to the question no 24:

A Linear Feedback Shift Register (LFSR) of length m over GF(2) generates a binary sequence $K = (K_1, K_2, K_3, \ldots)$ using the recurrence relation:

$$K_t = c_1 K_{t-1} \oplus c_2 K_{t-2} \oplus \cdots \oplus c_m K_{t-m}$$

The characteristics polynomial of the LFSR is

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

Scanned with CamScanner

Since the LFSR is of length m, its state at any time t is determined by the previous m bits, forming an m-bit state register.

There are 2^m possible states but on of them is the all-zero state (000, ...0) which is a fixed point trap because once reached, it remains zero forever.

Thus an LFSR can have at most $2^m - 1$ nonzero states, meaning that if the sequence reaches its maximum period, it must cycle through all $2^m - 1$ non zero states before repeating.

Now, if the characteristics polynomial $p(x)$ is primitive over GF(2), it has the following properties:

① It's roots are primitive elements of $GF(2^m)$. That is the roots generate all non zero elements of the field cyclically.

② The recurrence relation generates an m sequence (maximum length sequence). An m-sequence has a period equal to $2^m - 1$

P.T.O.

Scanned with CamScanner

Thus, if $P(x)$ is primitive, then the LFSR will generate a sequence of maximum possible period $2^m - 1$.

Answer to the question no 25;

① The process of signing a message using an LWE (Learning with Errors) based signature scheme (i.e. key generation, signing and verification steps) is given below:

Step 1; Key generation:

① Select parameters; choose a security parameter n and modulus q, along with error bounds for LWE noise.

② Generate a public matrix A:
A random matrix $A \in \mathbb{Z}_q^{n \times n}$ is selected which defines the LWE problem.

③ Generate the secret key: A small norm vector s_k (the secret key) is randomly chosen from a

P.T.O.

Scanned with CamScanner

bounded distribution.

① compute publickey : $P_K = A \cdot S_K + e$, where

e = small error vector.

∴ public key : $P_K = (A, t)$, where $t = A \cdot S_K + e$

secret key : S_K

Step2; Signing a message M;

~~Using a hash function H(M), commitment~~ $u = A \cdot r$

~~where, r = nonce~~

① Hash the message: ~~H(M)~~ $c = H(M, u)$

② Generate a commitment :

$u = A \cdot r$, where r = random nonce

③ Compute response : $z = r + c \cdot S_K$ where

c is derived from $H(M, u)$

④ Check bounds; If z is too large (i.e does not satisfy certain norm constraints) restart with a new r to ensure security.

Scanned with CamScanner

(5) Signature output: The signature ~~$\sigma(z,u)$~~ $\sigma = (z,u)$

Step 3: Verification;

(i) Recompute the challenge: Compute $c' = H(m,u)$ from the received message and commitment.

(2) Verify the response: Check whether

$$A \cdot z = u + c' \cdot t$$

If the equation holds and z is within the correct bounds, the signature is valid.

(ii) Signing a message using an LWE Based Scheme: Let's outline a practical example of signing a message m using an LWE based signature scheme.

Let, $q = 257$, $n = 4$, and choose a random matrix

$$A \in \mathbb{Z}_q^{4\times 4}$$

→ Choose a secret key sk

→ Compute the public key, $pk = A \cdot sk + e$

Scanned with CamScanner

Signing process for a message M:

① Compute the Hash of the message: $c = H(M, u)$

② Generate a commitment: $u = A.r$

③ Compute the response: $z = r + c.sk$

④ Output the signature: $\sigma = (z, u)$

Verification:

① Compute, $c' = H(M, u)$

② Check whether $A.z = u + c'.t$

If the equation holds and z is within valid bounds, the signature is valid.

Role of LWE in Ensuring Security:

① Hardness of LWE problem: Given only (A, t) an attacker can't recover sk, since solving $A.sk + e = t$ is as hard as solving the LWE problem.

② Unforgeability: A forger can't generate a valid signature (z, u) without knowing sk, as doing so ~~RETURN~~ requires solving an LWE instance

Scanned with CamScanner

③ Quantum Resistance: Unlike RSA, ECC which are broken by Shor's algorithm, LWE remains secure against quantum attacks.

Scanned with CamScanner