Somaya Jannat
IT-21034

1.

Ans:

Shor's algorithm is a quantum algorithm that efficiently solves the integer factorization and discrete logarithm problems in polynomial time. The security of RSA depends on the difficulty of factoring large integers, while ECC relies on the hardness of the elliptic curve discrete logarithm problem.

On classical computers, these problems are computationally infeasible for large key sizes. These means a sufficiently powerful quantum computer running Shor's algorithm can break RSA and ECC by finding private keys from public keys.

The consequences for current digital infrastructure are serious. Many protocols like HTTPS, SSL/TLS, email encryption and block chain technologies rely on RSA or ECC for security.

2.

Ans: Quantum Key Distribution (QKD) is a technique for secure key exchange that uses the principles of quantum mechanics to establish a shared secret key between two parties. It's security is based on the no-cloning theorem and the fact that any measurement of a quantum state disturbs it, allowing the communicating parties to detect the presence of an eavesdropper. Unlike classical methods, QKD provides information-theoretic security, independent of computational assumptions.

In future cryptographic system, QKD is expected to play a central role in protecting sensitive communications against the threat of quantum computers. By enabling the secure generation and distribution of symmetric encryption keys, QKD ensures that even adversaries with unlimited computational power cannot compromise the confidentiality of the exchanged key without detection.

Difference between QKD and classic Public-key Encryption:

Feature	Quantum key Distribution (QKD)	Classical Public-key Encryption (ECC/RSA)
Security foundation	Laws of quantum mechaniques mics	Computational hardness of math problems
Quantum Resistance	Inherently resistance to quantum attack	Vulnarable to Shor's algorithm.
Evasdropping detection	Yes, via disturbance/error rates.	No, inharent detection
Channel Requirements	Quantum + classical channel.	Classical channel only
Hardware Requirements	Specialized quantum devices	Standard computing hardware.

IT-21034

3.
Ans:
Lattice-based cryptography is a classic of cryptographic schemes whose security is based on the hardness of mathematical problems in high-dimensional lattices, such as, the Shortest Vector Problem (SVP) or the Learning With Errors (LWE) problem. These problems are believed to remain hard even for quantum computers.

In contrast, traditional number-theoretic approches like RSA and ECC rely on the difficulty of problems such as integer factorization and discrete logarithm. While these are secured against classical adversaries, they can efficiently solved on a sufficiently powerful quantum computer using Shor's algorithm, rending RSA and ECC insecure in a post-quantum world.

Main Differences based on quantum resistance:

Feature	Lattice-Based Cryp to	RSA/ECC
Security foundation	Hard lattice problems (SVP, LWE, etc.)	Integer factorization. Discrete logarithm
Quantum Resistance	Believe secure against quantum attacks	Broken by Shor's Algorithm
Maturity & usage	Newer, still under standardization (NIST)	Wil Widely deployed in current infrastructure
Efficiency	Larger key size but efficient operations	Small keys; efficient in classical settings
Application	Encryption, Signature homomorphic crypto	Encryption, signatur

Lattice-based cryptography offers a promising alternative to traditional number-theoretic schemes, espicially in the era of quantum computing. It's assumed quantum resistance and versatility make it a strong candidate for future cryptographic standards, though practical deployment requires optimization for key size and performance.

4.

Ans: Python PRNG using system time + custom seed:

Program:

```
import time
def custom_prng(seed):
    current_time = time.time.ns()
    state = (seed ^ current_time) & 0XFFFFFFFFFFFFFFFF
```

a = 6364136223846793005

c = 1

state = (a * state + c) & 0xFFFFFFFFFFFF
FFFFF.
return state.
seed_value = 12345
random_number = custom_prng(seed_value
print ("Custom PRNG output:", random_
number)

output: custom PRNG output = 483992
293458715215721.

5. Ans:

Sieve of Eratosthenes Algorithm:

The Sieve of Eratosthenes is a ancien
and efficient method to find all the prime
numbers up to a given limit n.

Step:

1. Create a boolean list is_prime
[0 --- n] initialized to true.

2. Set is_prime[0] and is_prime[1] to false
3. Starting from p=2, if is_prime[p] is true, mark all multiples of p from p^2 to n as false.
4. Increment p and repeat until $p^2 > n$.
5. All numbers marked true are prime.

Program to find all Prime number Less than 50.

```
def sieve_of_eratosthenes(limit):
    is_prime = [True] * (limit + 1)
    is_prime[0], is_prime[1] = False, False
    p = 2
    while p * p <= limit:
        if is_prime[p]:
            for i in range(p*p, limit+1, p):
                is_prime[i] = False
        p += 1
```

IT-21034

primes = [num for num, prime in enumerate(is_prime) if prime]
return primes

print("Primes less than 50:", sieve_of_eratosthenes(50))

Output:

prime number less than 50 = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]

Time complexity Comparison:

Method	Time complexity	Remark
Seive of Eratosthenes	$O(n \log \log n)$	
Trial Division	$O(n\sqrt{n})$	

6.

Ans:

Definition:

A carmichal number is a composit integer n such that

$$a^{n-1} \equiv 1 \pmod{n}$$

for every integer a with gcd $(a, n) = 1$.

In other words, n is a Fermat pseudoprime to every base coprime with n.

Necessary and sufficient condition (Korselt criterion):

A positive composite integer n is a Carmichael number if all three of the following hold:

1. n is composite
2. n is square free
3. for every prime divisor it, holds that $p-1 \mid n-1$

Sketch of proof:

→ If n is Carmichale then for every prime $p \mid n$ and every a with gcd $(a, n) = 1$ one uses the property $a^{n-1} \equiv 1 \pmod{n}$ and reduce modulo p to obtain $a^{n-1} \equiv 1 \pmod{p}$. By farmat's little theorem the multiplicative order of any $a \bmod p$ devid $p-1$, so $p-1$ divides $n-1$. Also, one shows n must be squarefree because if $p^k \mid n$ one can pick an a with certain reduction modulo p^k to contradict the universal congruence.

We test each n by factoring it, checking squarefree and checking $p-1 \mid n-1$ for every prime ~~ber~~ divisor p.

(a) $n = 561$

→ factorization: $561 = 3 \times 11 \times 17$.

→ check divisibility of $n-1 = 560$:

→ for $p = 3$: $p-1 = 2$, $(560 \div 2 = 280)$

→ for $p = 11$: $p-1 = 10$; $(560 \div 10 = 56)$

→ for $p = 17$: $p-1 = 16$; $(560 \div 16 = 35)$

As, all condition satisfied so, 569 is a Carmicheal number.

(b) $n = 1105$

→ factorization: $1105 = 5 \times 13 \times 17$

→ $n-1 = 1104$

→ for $p = 5$: $p-1 = 4$ | $(1104 \div 4 = 276)$

→ for $p = 13$: $p-1 = 12$ | $(1104 \div 12 = 92)$

→ for $p = 17$: $p-1 = 16$ | $(1104 \div 16 = 69)$

All condition satisfied. 1105 is a carmicheal.

(a) $n = 1729$

→ factorization: $7 \times 13 \times 19$

→ $n-1 = 1728$

→ for $p = 7$: $p-1 = 6$ | $(1728 \div 6 = 288)$

→ for $p = 13$: $p-1 = 12$ | $(1728 \div 12 = 144)$

→ for $p = 19$; $p-1 = 18$ | $(1728 \div 18 = 96)$

All condition satisfied so, 1129 is a Carmichael ~~m~~ number.

7. Ans:

(a) Is $\mathbb{Z}_{11}$ with $(+, \cdot)$ ring?

⇒ Yes. In fact $\mathbb{Z}_{11}$ is a commutative ring with unity and because 11 is prime, it is a field.

Justification:

→ $(\mathbb{Z}_{11}, +)$ is an abelian group; addition mod 11 is closed, associative has identify 0, every element a has an additive inverse $-a \equiv 11-a$

and addition is commutative.

→ Multiplicative mod 11 is closed and associative and distributes over addition. There is a multiplicative identity 1.

→ Therefor all ring axioms hold, so, $\mathbb{Z}_{11}$ is a ring.

(b) Are $(\mathbb{Z}_{37}, +)$ and $(\mathbb{Z}_{35}, \times)$ Abelian group

=> (i) $(\mathbb{Z}_{37}, +)$ is an Abelian group.

~~Reseaon:~~

Reason: Addition modulo 37 satisfied all group of axioms: closure, associativity identity 0, inverses (for a the inverse is 37 − a) and commutativity. Thus $(\mathbb{Z}_{37}, +)$ is a cyclic abelian group of order 37.

(iii) $(\mathbb{Z}_{35}, \times)$ — No:

Reason: Multiplication modulo 35 is associative and has identity 1, and is commutative but many elements do not have multiplicative inverses. For example, 5 in $\mathbb{Z}_{35}$ has $\gcd(5, 35) = 5 > 1$, and there is no x with $5 \cdot x \equiv 1 \pmod{35}$ because $5x$ is divided by 5.

8. Ans:

We want the least nonnegative residue of -52 modulo 31

$-52 \equiv -52 + 2 \cdot 31 = -52 + 62 = 10 \pmod{31}$

the remainder 10.

9. Ans:

Multiplicative inverse of 7 modulo 26
(Use Extended Euclidean algorithm)

$7x \equiv 1 \pmod{26}$

$7x + 26y = 1$

Euclidean algorithm:

$$26 = 3 \cdot 7 + 5$$
$$7 = 1 \cdot 5 + 2$$
$$5 = 2 \cdot 2 + 1$$
$$2 = 2 \cdot 1 + 0$$

Back Substitution:

$$1 = 5 - 2 \cdot 2$$
$$2 = 7 - 1 \cdot 5 \Rightarrow 1 = 5 - 1(7 - 5)$$
$$= 3 \cdot 5 - 2 \cdot 7$$
$$5 = 26 - 3 \cdot 7 \Rightarrow 1 = 3(26 - 3 \cdot 7) - 2 \cdot 7$$
$$= 3 \cdot 26 - 11 \cdot 7$$

thus, $-11 \cdot 7 + 3 \cdot 26 = 1$, so, -11 is an inverse modulo 26. Reduce to $0 \leq x \leq 26$:

$-11 \equiv 15 \pmod{26}$.

10. Ans:

$-8 \times 5 = -40$, $-40 \equiv +40 + 3 \cdot 17 = 11 \pmod{17}$.

Alternatively reduce factors first

$-8 \equiv 9 \pmod{17}$ then $9 \cdot 5 = 45 \equiv 11$

11. Ans:

Bezout's theorem (identity)

For integers a, b not both zero there exist integers x, y such that

$$ax + by = \gcd(a, b).$$

Proof: Apply the Euclidean algorithm to a, b, the back-substitution step expresses the gcd as an integer linear combination of a and b.

To find the inverse of 97 modulo 385 we need integers x, y with $97x + 385y = 1$.

Extend Euclidean:

$$385 = 3 \cdot 97 + 94$$
$$97 = 1 \cdot 94 + 3$$
$$94 = 31 \cdot 3 + 1$$
$$3 = 3 \cdot 1 + 0$$

Back-substitution:

$$1 = 94 - 31 \cdot 3$$
$$3 = 97 - 1 \cdot 94 \Rightarrow 1 = 94 - 31(97 - 94)$$
$$= 32 \cdot 94 - 31 \cdot 97$$
$$94 = 385 - 3 \cdot 97 \Rightarrow 1 = 32(385 - 3 \cdot 97)$$

$-31 \cdot 97 = 32 \cdot 385 + 127 \cdot 97$

Thus, $-127 \cdot 97 + 32 \cdot 385 = 1$; so, -127 is an inverse of $97 \bmod 385$.

$$-127 \equiv 385 - 127 = 258 \pmod{358}$$

12.

Bezout's identity:

For integers a, b not both zero there exist integers x, y with $ax + by = \gcd(a, b)$. The Euclidean algorithm produces ~~there~~ these x, y by back-substitution, so solutions exists and are computable.

To find x with $43x \equiv 1 \pmod{240}$

Euclidean algorithm:

$$240 = 5 \cdot 43 + 25$$
$$43 = 1 \cdot 25 + 18$$
$$25 = 1 \cdot 18 + 7$$

$18 = 2 \cdot 7 + 4$

$7 = 1 \cdot 4 + 3$

$4 = 1 \cdot 3 + 1$

$3 = 1 \cdot 3 + 0$

Back-substitute to express 1 as $43x + 240y$:

$1 = 4 - 1 \cdot 3$

$3 = 7 - 1 \cdot 4 \Rightarrow 1 = 2 \cdot 4 - 7$

$4 = 18 - 2 \cdot 7 \Rightarrow 1 = 2(18 - 2 \cdot 7) - 7$

$= 2 \cdot 18 - 5 \cdot 7$

$7 = 25 - 1 \cdot 18 \Rightarrow 1 = 2 \cdot 18 - 5(25 - 18)$

$= 7 \cdot 18 - 5 \cdot 25$

$18 = 43 - 1 \cdot 25 \Rightarrow 1 = 7(43 - 25) - 5 \cdot 25$

$= 7 \cdot 43 - 12 \cdot 25$

$25 = 240 - 5 \cdot 43 \Rightarrow 1 = 7 \cdot 43 - 12(240 - 5 \cdot 43)$

$= 67 \cdot 43 - 12 \cdot 240$

thus $67 \cdot 43 - 12 \cdot 240 = 1$ so, $x = 67$ is a inverse.

13.

Ans:

Farmat's Little theorem statement and proof:

Let p be prime number and integer a with $p \nmid a$. Then.

$$a^{p-1} \equiv 1 \pmod{p}$$

Proof: The set $G = \{1, 2, \ldots, p-1\}$ forms a multiplicative group $(\mathbb{Z}/p\mathbb{Z})^{\times}$ of size $p-1$. Multiplying each element by a permutes G. Hence,

$$\prod_{x \in G} x \equiv \prod_{x \in G} (ax) \equiv a^{p-1} \prod_{x \in G} x \pmod{p}$$

Cancel $\prod_{x \in G} x \neq 0$ to get $a^{p-1} \equiv 1 \pmod{p}$.

Using FLT as a (probable) Primiarly test

for prime, p any a with gcd(a,p)=1 must satisfy $a^{p-1} \equiv 1 \pmod{p}$. If some a fails, p is composite. If all tested as pass, p is "Fermat probable prime," but not guaranteed prime (Carmicheal numbers are counterexamples)

Is 561, prime by FLT?

$561 = 3 \cdot 11 \cdot 17$ (composite). Yet 561 is a Carmicheal number, so for all a coprime to 561, $a^{560} \equiv 1 \pmod{561}$. Thus fermat's test would incorrectly lebel 561 as "probable prime."

Conclusion: 561 is not prime even though it passes FLT for many bases.

Compute $5^{123} \bmod 175$

Since, $175 = 25 \cdot 7$ and 5 is not coprime to 25:

→ Mod 25: $5^2 \equiv 25 \equiv 0 \Rightarrow 5^{123} \equiv 0 \pmod{25}$

→ Mod 7: gcd(5,7) = 1 and by FLT, $5^6 \equiv 1$

Now, $123 \equiv 3 \pmod 6$ so,

$5^{123} \equiv 5^3 = 125 \equiv 6 \pmod 7$.

Solve by CRT:

$x \equiv 0 \pmod{25}, x \equiv 6 \pmod 7$

Let, $x = 25k$ then $25k \equiv 6 \pmod 7$

$\Rightarrow 4k \equiv 6 \pmod 7 \Rightarrow k \equiv 5 \pmod 7$

smallest $k = 5 \Rightarrow x = 25 \cdot 5 = 125$

14.

Ans:

Statement: (Pairwise coprime moduli)

→ If $m_1, \ldots, m_n$ are pairwise coprime and $a_i \in \mathbb{Z}$, then the system $x \equiv a_i \pmod{m_i}$ has a unique solution modulo $M = m_1, \ldots m_n$.

Proof (Constructive):

Let, $M_i = M/m_i$ and choose y_i with $M_i y_i \equiv 1 \pmod{m_i}$. Then,

$$x \equiv \sum_{i=1}^{n} a_i M_i y_i \pmod{M}$$

solves the systems;

$$x \equiv 2(3),\ x \equiv 3(5),\ x \equiv 2(7)$$

Here, $M = 3 \cdot 5 \cdot 7 = 105$

$M_1 = 35$ $\quad 35 \equiv 2(3),\ y_1 = 2 \Rightarrow e_1 = 35 \cdot 2 = 70;$

$M_2 = 21$ $\quad 21 \equiv 1(5),\ y_2 = 1 \Rightarrow e_2 = 21;$

$M_3 = 15$ $\quad 15 \equiv 1(7),\ y_3 = 1 \Rightarrow e_3 = 15.$

Then,

$x \equiv 2e_1 + 3e_2 + 2e_3 = 2 \cdot 70 + 3 \cdot 21 + 2 \cdot 15 = 233$

$\equiv 23 \pmod{105}$

15. CIA triad (Information Security):

→ Confidentiality: Prevent unauthorized disclosure of data. Mechanisms: encryption, access control, least privilege data classification.

→ Integrity: ensure data is accurate unaltered. Mechanisms: hashes / MACs. digital signatures, checksums, versioning input validation,

→ Availability: ensure systems / data are accessible when needed. Mechanism: redundancy, failover, backups, DOS protection, capacity planning; ~~non~~ monitoring.

16. Ans:

Steganography vs Cryptography:

→ Cryptography hides the content but not the existance of a massege.

→ Stegenography hides the existance of a massege by embadding it in innocaous carriers

Common Stego technique:

→ Spatial - domain LSB substitution in images

→ Transform-domain embedding: DCT (JPEG) DWT (wavelets).

→ Spread - spectrum and echo hiding.

→ Masking / brightness / alpha tweaks, palette manipulation.

→ Text Stego: Whitespace, homoglyphs, synonym substitution, markup / metal field

→ Network Stego: Converts channels in headers.

17.
Ans:

Phishing vs Malware vs DoS:

→ Phishing: Social-engineering to steal credentials/PII via deceptive emails/sites/SMS.

Impact: account takeover, fraud, initial foothold for further attacks.

Countermeasures: user training, phishing-resistant MFA (FIDO2), DMARC/SPF/DKIM, URL filtering.

→ Malware: Malicious software. Delivered by phishing, drive-by downloads, exploits USB.

Impact: data theft, encryption, botnet enlistment.

Count measure: EDR/AV, patching, least privillege, application.

→ DoS / DDoS — Overwelm resources / bandwidth to deny service. Methods: volumetric floods, protocol attacks, application layer.

Impact: downtime, SLA penalties,

Countmeasures: scrubbing / CDN, rate limiting.

18.

Ans: The way GDPR helps mitigate attacks and protect privacy is given below-

~~Pra~~

Principles: Lawfulness, fairness, transparency, purpose limitation, data minimization, accuracy storage, limitation, integrity & ~~confe~~ confidentiality, accountability.

Security obligations: Appropriates techinal and organizational measures,

Privacy by design / default: integenate controls eady.

Data subject right: access/ rectification erasure / portability/ restriction.

Breach notification: controllers must notify authority within 72h and when high risk effected individuals.

DPIA & DPO: Risk assessment and governance.

Enforcement: Heavy fines incentivize compliance.

19. Ans:

DES basics:

→ Block/key: Input is 64-bit block.

→ Initial Permutation: Bitwise permutation of the 64 bit input.

→ feistel structure - 16 rounds: split input after IP into L_0, R_0. For round $i = 1$ in 16.

$$L_i = R_{i-1}; \quad R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

→ Expansion E: expand R_{i-1} from 32 to 48 bits.

→ Key mixing: XOR with 48-bit subkey K_i.

→ S box: eight $6 \rightarrow 4$ nonlinear substitution

→ Permutation: permute the 32 bit S-box output.

→ Swap and final permutation (IP^{-1}): after round 16, swap halves and apply IP^{-1} to get the 64 bit ciphertext. IP^{-1} also has no cryptographic strength.

20.

Ans: DES, First round

Given: $R_0 = 0XF0F0F0F0$, $K_1 = 0X0F0F0F0F0F$ and $L_0 = 0XAAAAAAAA$.

Under the XOR-only assumption take the 32-bit part of the key and compute $f(R_0, K_1) = R_0 \oplus K_1 = 0XF0F0F0F0 \oplus 0X0F0F0F$

0f0f = 0x ffffffff

Then the feistel update is

$L_1 = R_0 = 0x f0f0f0f0$,

$R_1 = L_0 \oplus f(R_0, K_1) = 0xAAAAAAAA$

$\oplus$ 0x ffffffff = 0x55555555

21. AES Subbytes using the partial S-box provided:

→ Input word: [0x23, 0xAZ, 0x4c, 0x1n

Index S-box by hex row = high nibble
col = low nibble.

→ 0x23 ⇒ row 2, col 3 → D4

→ 0xAZ ⇒ row A, col Z → 63

→ 0x4c ⇒ row 4, col C → 2E

→ 0x19 ⇒ row 1, col 9 → c6

output word : 0XD4, 0x63, 0X 2E ; 0XC6.

22. AES Add Round key

→ Input: [0x1A, 0X 2B, 0X3C, 0X4D]

Round key: [0X 55, 0X 66, 0X77, 0X88]

compute XOR per byte:

→ 0X1A ⊕ 0X55 = 0X4F

→ 0X 2B ⊕ 0X66 = 0X4D

→ 0X 3C ⊕ 0X 77 = 0X 4B

→ 0X 4D ⊕ 0X88 = 0XC5.

23. AES Mix Columns:

→ Use Rijndael's fixed matrix over GF(2^8)

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}, \quad a_0 = 01, a_1 = 02, a_2 = 03, a_3 = 04$$

Rules in GF(2^8) (poly $x^8+x^4+x^3+x+1$)

2. x = x time (x); $3 \cdot x = (2 \cdot x) \oplus x$.

pre compute:

→ $2.01 = 02$, $2.02 = 04$, $2.06 = 06$, $2.04 = 08$

→ $3.01 = 03$, $3.02 = 06$, $3.03 = 05$, $3.04 = 0C$

Now, the outputs:

→ $r_0 = 2a_0 \oplus 3a_1 + 1a_2 \oplus 1a_3 = 02 \oplus 06 \oplus 03 + 04$
$= 03$

→ $r_1 = 1a_0 \oplus 2a_1 \oplus 3a_2 \oplus 1a_3 = 01 \oplus 04 \oplus 05$
$\oplus 04 = 04$

→ $r_2 = 1a_0 \oplus 1a_1 \oplus 2a_2 \oplus 3a_3 = 01 \oplus 02 \oplus 06$
$\oplus 0C = 09$

→ $r_3 = 3a_0 \oplus 1a_1 \oplus 1a_2 \oplus 2a_3 = 03 \oplus 02 \oplus 03$
$\oplus 08 = 0A$.

output column: $[0x03, 0x04, 0x09, 0x0A]^T$
in binary: 00000011, 00000100, 00001001, 00001010.

24. AES-OFB mode:

→ How it works:

$O_0 = E_K(IV)$; for $i \geq 1$; $O_i = E_K(O_{i-1})$

Encrypt: $C_i = P_i \oplus O_i$; Decrypt: $P_i = C_i \oplus$

→ Synchronization: Both sides must share the same IV and process the same number of block- the keystream is independent of the plaintext. Losing or inserting bits breaks sync.

→ Properties: No padding, bit errors in C_i flip only the corresponding bit in P_i (no). Never reuse an IV with the same key — that would reuse the keystream.

25. Ans;

Error propagation (CBC vs CFB):

→ CBC: $C_i = F_K(P_i \oplus C_{i-1})$

A 1-bit error in C_i =>

→ P_i decrypts to a random-looking block.

→ The same bit in P_{i+1} flips.

Later blocks OK.

CFB:

A 1-bit error in C_i =>

→ that bit flips in P_i

→ The next segment P_{i+1} is garble then recovery.

Impact: CBC causes one full-block corruption + 1 flipped bit next block

CFB causes one flipped bit then one garbled next segment.

26: Mode for large, parallel processing: CTR

→ ECB: parallelizable but leaks patterns

→ CBC: hides patterns, but encryption not parallelizable, decryption can be

parallelized.

→ CRT: $C_i = P_i \oplus E_K(\text{Nonce} \| \text{ctr}_i)$.
Each keystream block is independent ⇒ full parallelism, random access, precomputation, no padding
Requirements: Never reuse the same counter under a key.
Recomendation: Use CTR.

27: Ans:

Tiny RSA:
$M = 1$; $e = 5$; $n = 14$
Decrypt with $d = 11$
Encrypt: $C = M^e \mod n = 1^5 \mod 14 = 1$
Decrypt: $M' = C^d \mod n = 1^{11} \mod 14 = 1$

28. Ans:

RSA signature

given hash $H(M)=5$

private key $d=3$, $n=33$

signature $s \equiv H(M)^d \bmod n = 5^3 \bmod 33$

$= 125 \bmod 33 = 125-99=26$

varification check : $s^e \equiv H(M) \pmod n$ with the public key.

29 : Ans

Diffie - Hellman ($P=17, g=3$)

privat : $a=4$ (Aleya), $b=5$ (Badol)

public keys:

→ Aleya: $A = g^a \bmod p = 3^4 \bmod 17$

$= 81 \bmod 17 = 13$

→ Balol: $B = g^b \bmod p = 3^5 \bmod 17$

$= 243 \bmod 17 = 5.$

Shared secret $= B^a \equiv A^b \equiv 13 \pmod{17}$.

30. Ans:

simple hash $H(x) = (\Sigma \text{ASCII chars of } x) \bmod 100$.

→ ASCII : A = 65, B = 66

→ For "AB" : $H(AB) = (65 + 66) \bmod 100$

$= 131 \bmod 100 = 31$

→ For "BA" : $H(BA) = (66 + 65) \bmod 100$

$= 31$

So, AB and BA produce the same hash

Implication: This shows the function is not collision resistant. A cryptographic hash must make finding distinct input with the same output infeasible this simple sum modulo 100 is ~~strd~~ trival to collide. so it is un-

suitable for cryptographic hash.

31. Ans:

MAC = (Message + Secret key) mod 17.

Given message = 15, key = 7.

Compute MAC;

MAC = (15+07) mod 17 = 22 mod 17 = 5

If attacker changes message to 10 but doesnot know the key.

→ Without any MAC oracle or knowledge of the original MAC the attacker can not compute

the correct MAC.

→ However, if the attacker observes the original valid MAC, they can compute the new MAC:

$$MAC_{10} = MAC_{15} + (10 + 15) \bmod 17$$

$$= 5 + (-5) = 0$$

so, this MAC scheme is insecure because it if linear.

32. Ans:

TLS handshake – steps & how symmetric keys are established typical simplified handshake:

1. Client Hello: Client sends supported versions, cipher suits, random nonce

2. ServerHello: Server picks version sends Server Random.

3. Server certificate: Server sends it certificate for authentication.

4. Sever key Exchange: for Ephemeral Diffie-Hellman.

5. ServerHello Done.

6. Client key exchange:

→ RSA key exchange: Client generates a pre-master secret and encrypts it with server's key.

→ (EC) DHE: Client sends it Diffie-Hellman public value enabling shared secret derivation.

7. Both sides computes master secret from pre-master + Client Random + Server Random, then derive symmetric session key.

8. ChangeCipherSpec; Client and server signal switching to the negotiated cipher and veryfy the handshake.

How symmetric key Establish Secretly:

→ Either via asymmetric encryption of a pre-master secret (RSA) or via DHE / ECDHE. DHE/ECDHE provides forward secrecy; The session keys are derived from ephemeral DH secrets that are not recoverable from the server's long-term private key along.

33. Ans:

SSH layered architecture – roles;

SSH is orginized into three main protocol layers (running over TCP)

1. Transport layer Protocal;

→ Provides server authentication

→ After this layer, a secure channel is established for higher layers.

2. User ~~Athen~~ authentication Protocol

→ Authenticates the client to the server.

→ Uses the secure channel from the transport layer ~~protocal~~ protect credentials.

3. Connection Protocol:

→ Multiplexes multiple logical channels over the single SSH connection

→ Provides channel-level request and flow control.

34. Ans:

TLS Steps:

1. Client HELLO.
2. ServerHello, Server ~~cert~~ certificate, server hello done.
3. Client key Exchange.
4. Both compute master secret → derive symmetric keys.

5. Client Change Cipher Spec, Client Finished, Server Change Cipher Spec, Server Finished.

6. Secure application data channel established.

35. Ans:

Elliptic curve over a finite field.

$\Rightarrow$

General form over finite field F_p

$$y^2 \equiv x^3 + ax + b \pmod{p}$$

with the non-singularity condition $4a^3 + 27b^2 \not\equiv 0 \pmod{p}$.

why use cryptography:

→ The points on the curve form an group. the Elliptic Curve Discrete Logarithm Problem.

→ security per bit is high.

→ Efficient implementations, small key/certificate size and suitability for constrained devices make ECC popular for modern protocols.

36. Ans:

ECC achieves the same security as RSA with a smaller key size because the Elliptic Curve Discrete Logarithm Problem is mathematically harder than the integer factorization problem used in RSA.

→ In RSA, security depends on the difficulty of factoring a large number $n = pq$.

→ In ECC, security depends on finding K given P and KP (ECDLP), which has no known sub-exponential time algorithm for general curves.

ECC key can be much shorter while ~~attacking~~ offering the same computational difficulty to an attacker.

shorter keys - faster computation, less bandwidth, smaller certificates.

37. Ans:

given, $P = (3,6)$

curve: $y^2 \equiv x^3 + 2x + 3 \pmod{97}$

point: $P = (3,6)$

Step 1: Calculate $y^2 \bmod 97$

$$y^2 = 6^2 = 36$$

Step 2: Calculate $x^3 + 2x + 3 \bmod 97$

$$x^3 = 3^3 = 27$$

$$x^3 + 2x + 3 = 27 + (2 \cdot 3) + 3 = 27 + 6 + 3$$

$$= 36$$

Step 3: Compare both sides:

$$36 \bmod 97 = 36$$

Both sides are equal => Yes, P lies on the curve.

38: Ans:

ElGamal Encryption:

Given, $p = 23, g = 5, h = 8, m = 10, k = 6$

Formula:

→ $c_1 = g^k \pmod p$

→ $c_2 = m \cdot h^k \pmod p$

Step 1: $c_1 = 5^6 \bmod 23$

$5^2 = 25 = 2 \pmod{23}$

$5^4 = 2^2 = 4$

$5^6 = 5^4 \cdot 5^2 \equiv 4 \cdot 2 = 8 \pmod{23}$

Sol Step 2:

$$h^K = 8^6 \bmod 23$$

$$8^2 \equiv 64 \equiv 18$$

$$8^4 \equiv 18^2 = 324 \equiv 2$$

$$8^6 \equiv 8^4 \cdot 8^2 \equiv 2 \cdot 18 = 36 \equiv 13$$

Step 3: $c_2 = m \cdot h^K \equiv 10 \cdot 13 = 130 \equiv 15 \pmod{23}$

ans: $(c_1, c_2) = (8, 15)$.

39. Ans:

Lightweight cryptography in IoT:

→ Lightweight cryptography in ~~IoT is~~ important fo IoT because IoT device often have limited CPU power, memory

and battery life. These algorithms are designed to use minimal computational resources, low power and small memory footprint, while still ensuring security.

Example Algorithm:

→ PRESENT cipher → a lightweight block cipher widely used in IoT for low-power devices.

40. Ans:

Three IoT specific attack and mitigation:

1. Firmware Hijacking: Attacker modifies devices firmware to insert malicious code.

mitigation: Secure boot, firmware signing and regular updates.

2. Physical Tampering: Attacker physically accesses the device to extract keys or modify hardware.

mitigation: Tamper-resistant casing intrusion detection sensors.

3. Botnets – IoT devices infected and controlled remotely to perform DDoS attcks.

Mitigation: Strong authentication, disable, unused services, keep software update

[IT-21034]