Assignment (cryptography)

Atiqur Rahman Sajib

IT-21040

ii) Lots of confidentiality: Even past communication store today could be decrypted in the future.

iii) Compromised authentication: Digital signature based on RSA/ECC could be forged, certificate and blockchain systems.

iv) Need to Migrate: A rapid shift to post-quantum cryptography would be required to secure future and existing systems.

(2) Ans:

Role of quantum key distribution (QKD): Establishes symmetric keys with information-theoretic security by encoding bits on quantum states, eavesdropping includes detectable errors.

Differ from classical public-key encryption (PKE):

i) Security from physics, not computational hardness.

ii) Needs a quantum channel + authenticated classical channel; limited distance/rate, specialized hardware.

iii) Doesn't replace bulk encryption or authentication, still use AES/MAC + a classical auto method.

(3) Lattice-based crypto vs number-theoretic (RSA/DH/ECC) in context of quantum resistance.

i) Assumption : Lattice Problems (LWE/RLWE/SIS) vs factoring/dicreate log.

ii) Quantum : No known efficient quantum attacks on lattice problem, belived PQ-Secure, RSA/ECC fall to shor.

iii) Math/ops : Mostly linear algebra mod q, good performance.

iv) Security : Worst-case to average-case reductions for lattice, RSA/ECC rely on average-case hardness only.

④ Python-based PRNG that uses the current system time and a custom seed value –

code:

```
import time
A = 6364136223846793005
B = 1442695040888963407

MOD = 1 << 64

def lcg(seed):
    x = seed % MOD
    while True:
        x = (A * x + C) % MOD
        yield x

def mix_seed(user_seed: int) -> int:
    t = time.time_ns()
    s = (t ^ ((user_seed + 0x9E3779B97F4A7C15) *
        * 0xBF58476D1CE4E5B9)) & (MOD - 1)
    return s

if __name__ == "__main__":
user_seed = 12345
```

```
s = mix_seed (user_seed)
g = leg (s)
Print ("mixed_seed = ", s)
for _ in range (10):
    v = next (g)
    Print (v, V/MOD)
```

Example output:

```
mixed_seed = 133079665867147 82339
17142038108726654746 0.9288
12078600592703506077 :0.6546
.....
.....
.....
```

(5) Sieve of Eratosthenes (Prime<50) -

Algorithm: Mark multiple of each unmarked number starting from 2up to $\sqrt{n}$.

Prime <50 are!

2,3,5,7,11,13,17,19,23,29,31,37,41,43,47

Complexity:

Time → $O(n \log\log n)$ vs trial division $O(n\sqrt{n})$ to list primes up to n.

Explaination of sieve of Eratosthenes:

i) List numbers from 2 to n.
ii) start with the first unmarked number P — initially Prime.
iii) Mark all multiples of P as composite.
iv) Find the next unmarked number
v) Repeat until $P^k > n$.
vi) All unmarked numbers left are Primes.

⑥ A carmichael Number is a composite number n that satisfies Fermat's Little Theorem for all integers a that are coprime to n:

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

They are called absolute Pseudoprimes.

Necessary and Sufficient condition –

i) n is square-free.

ii) n has at least three distinct Prime factors.

iii) For every Prime divisor P of n:

$$(n-1) \bmod (P-1) = 0$$

Varify and cheak the number if it charmichael-

i) 561 → 3, 11, 17, all $(P-1) \mid 560$ → carmicheal.

ii) 1105 → 5, 13, 17, all $(P-1) \mid 1104$ → carmicheal.

iii) 1729 → 7, 13, 19, all $(P-1) \mid 1728$ → carmicheal.

∴ All three are carmicheal numbers.

(7) Valid Algebric structure –

→ Z_{11} with operation $(+, \cdot)$: Yes it's ring (actually a field) because 11 is prime.

→ $(Z_{37}, +)$: Abelian group (cyclic of order 37).

→ $(Z_{35}, \times)$: Not a group (zero divisors, non-units like 5 has no inverse)

The Units $U(35)$ of size $\varphi(35) = 24$ no form an Abelian group.

(8) Remainder of -52 mod 31.

Find the least non-negative residue r with $0 \leq r < 31$ such that $-52 \equiv r \pmod{31}$

Method 1 (add multiples of 31):

→ $-52 + 31 = -21$ (still negative)

→ $-21 + 31 = 10$. (in range of ~~pos~~ 0...30).

$\therefore r = 10$.

Method 2 (Euclidian division):

Find q, r with $-52 = 31q + r$, $0 \le r < 31$

Take $q = -2$;

$-52 = 31(-2) + 10$ as $r = 10$.

Therefor -

$-52 \equiv 10 \pmod{31}$.

$(-52 \bmod 31) = 10$.

⑨ Multiplicative inverse of $7 \bmod 26$.

Extended Euclid:

$26 = 3 \cdot 7 + 5$

$7 = 1 \cdot 5 + 2$.

$5 = 2 \cdot 2 + 1$.

~~$1 = 3 \cdot 26$~~

$1 = 3 \cdot 26 - 11 \cdot 7$

So, $-11 \cdot 7 \equiv 1 \pmod{26}$

inverse of is 15 (since $-11 \equiv 15$).

Ans.

check: $7 \cdot 15 = 105 \equiv 1 \pmod{26}$.

(10) Evaluating ~~$(-8 \times 5 = -40)$~~ $-8 \times 5 \bmod 17$.

i) Multiply → $-8 \times 5 = -40$

ii) Now add multiples of 17 until result is in 0-16:

$$-40 + 3 \times 17$$
$$= -40 + 51$$
$$= 11.$$

∴ Result is 11.

Tips for negative modular multiplication:

i) Replace negative numbers with their positive equivalent mod m.

Example: $-8 \equiv 9 \pmod{17}$

Then $9 \times 5 = 45 \equiv 11 \pmod{17}$

(11)

Bezoutis Theorem: For integers a, b (not both zero), their exist integers x, y such that:

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

Proof idea: Use the extended Eucledian algorithm to express gcd (a, b) as a linear combination of a and b.

Finding inverse of 97 mod 385:

1. $\gcd(97, 385) = 1 \rightarrow$ Inverse exists
2. Extended Euclid:

$$1 = 385(32) - 97(127).$$

3. So, $97(-127) \equiv 1 \pmod{385}$.
4. Positive inverse: $-127 \equiv 258 \pmod{385}$

∴ Inverse of 97 mod 385 is 258.

(Ans).

(12) Given Equation -

$$43x \equiv 1 \pmod{240}$$

We need the modular inverse of ~~43 mod~~ 43 mod 240.

Steps (Extended Euclid):

1. $240 = 43(5) + 25$
2. $43 = 25(1) + 18$
3. $25 = 18(1) + 7$
4. $18 = 7(2) + 4$
5. $7 = 4(1) + 3$
6. $4 = 3(1) + 1$
7. Back-substitute $\rightarrow 1 = 43(67) - 240(12)$

$\therefore x \equiv 67 \pmod{240}$

$\therefore x = 67$. (Ans).

(13) Fermat's little Theorem:

If p is prime and a is an integer not divisible by p, then $a^{p-1} \equiv 1 \bmod p$.

Primality Test: If $a^{p-1} \not\equiv 1 \bmod p$ for some a with $1 < a < p$, then p is composite.

→ Evaluate $5^{123} \bmod 175$.

~~step 1: Find $5^{123} \bmod 25$~~

· step 1: Mod 25 ; $5^2 = 25 \equiv 5^{123} \equiv 0$

step 2: Mod 7 : $\varphi(7) = 6$

$$123 \bmod 6 = 3$$

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

step 3: combine via CRT:

$$x \equiv 0 \pmod{25}.$$

$$x \equiv 6 \pmod 7$$

$$\equiv 125.$$

$\therefore 5^{123} \equiv 125 \pmod{175}$.

(14)

Chinese Remainder Theorem (CRT):

If $n_1, \ldots, n_k$ are pairwise co-prime integers, then for any integers $a_1, \ldots, a_k$ are the system -

$$x \equiv a_i \pmod{n_i} \quad [i = 1, \ldots k]$$

has a unique solution modulo $N = \prod_i n_i$.

Proof idea: construct $N_i = N/n_i$, find y_i with $N_i y_i \equiv 1 \pmod{n_i}$, then -

$$x \equiv \sum_i a_i N_i y_i \pmod{N}.$$

satisfies all congruences.

Now - $x \equiv 2 \pmod{3}$, $x \equiv 3 \pmod{5}$, $x \equiv 2 \pmod{7}$

Method 1: list numbers $\equiv 2 \pmod{3}$: 2, 5, 8, 11, 14, 17, 20, 23, ...

check mod 5 and 7.

$23 \equiv 3 \pmod{5}$ and $23 \equiv 2 \pmod{7}$

So, $x = 23$ works.

Method 2:

i) $N = 3 \cdot 5 \cdot 7$

$= 105$

ii) $N_1 = 35$.

Find y_1 with $35y_1 \equiv 1 \pmod 3$

$35 \equiv 2 \pmod 3$, so,

$2 \times 2 \equiv 1$

$y_1 \to 2$.

contribution: $2 \cdot 35 \cdot 2 = 140$

iii) $N_2 = 21$. $21y_2 \equiv 1 \pmod 5 \to 21 \equiv 1 \pmod 5$

$y_2 \to 1$.

contribution: $3 \times 21 \times 1 = 63$

iv) $N_3 = 15$. $15y_3 \equiv 1 \pmod 7 \to 1 \equiv 1$

$15 \equiv 1 \pmod 7$.

$y_3 = 1$.

contribution: $2 \times 15 \times 1 = 30$.

Sum $140 + 63 + 30 = 233$. Reduce mod 105:

$233 - (2 \cdot 105)$

$= 233 - 210$

$= 23$.

$\therefore x \equiv 23 \pmod{105}$.

(15) CIA triad in information security -

i) Confidentiality : Keep data secret from unauthorized parties.

controls: encryption, access control, strong authentication, network segmentation.

ii) Integrity : Ensure data is correct and unmodified.

controls : cryptographic hashes, digital signature, checksum, input validation.

iii) Availability : Ensure authorized users can access system and data when needed.

controls : Redundancy, backups, load blancing. Pacching and monitoring, disaster recovery plan.

(16)

steganography: Hides the existance of a message in another file (image, audio, video).

Example: least significant bit (LSB) image Pixel modification, audio echo hiding. DCT.. coefficient changes in JEPG.

Goal: Secret message looks like normal media.

Cryptography: scrambles message into unreadable ciphertext using an algorithm + key ... Presence of message is obvious but content are protected.

Key difference:

steganography conceals existance.

Cryptography conceals content.

(17) Phishing vs malware vs DOS.

i) Phishing: social-engineering attacks to trick users into revailing credentials or clicking malicios links.

Impact: credential theft, account compromise, data breaches.

ii) Malware: Malicious software (viruses, trojans, ransomware) installed to steal data, damage system, or create backdoors.

Impact: Data loss, Unauthorized access, lateral movement.

iii) Denial-of-service (DoS): overwhelm resources to make services unavailable.

Impact: Downtime, revenue loss, reputation damage.

(18) How GDPR helps mitigate cyber attacks and protect privacy:

i) Data minimization and Purpose limitations: store less data.

ii) Mandatory breach notification: Faster containment and user awareness.

iii) Security by design and by default: encryption, access control.

iv) Data Processor contracts and accountability: Third parties must meet same security standard.

v) Heavy fines: Incentivize strong cybersecurity Practices.

(19) DES (Data Encryption Standard): 64-bit block cipher, 56-bit effective key.

Process:

i) Initial Permutation (IP): Fixed reordering of 64 input bits.

ii) 16 Feistel Rounds: each round are -

→ Split into L_i, R_i

→ $L_{i+1} = R_i$

→ $R_{i+1} = L_i \oplus f(R_i, K_i)$

→ f: Expand 32 → 48 bits → XOR with Round keys → S-box Substitution → Permutation.

iii) Final Permutation: Inverse of IP.

DES is sucered due to short 56-bit key.

(20) DES Round computation :

Given that -

$R_0 = 0x F0F0F0F0$,

$K_1 = 0x \oplus 0F0F0F0F$

$L_0 = 0xAAAAAAAA$

Assume $f(R_0, K_1) = R_0 \oplus K_1$

Compute $f(R_0, K_1) = 0xF0F0F0F0 \oplus 0x0F0F0F0F$

$= 0xFFFFFFFF$.

$L_1 = R_0 = 0xF0F0F0F0$.

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

$= 0xAAAAAAAA \oplus 0xFFFFFFFF$

$= 0x55555555$.

$\therefore L_1 = 0xF0F0F0F0$

$R_1 = 0x55555555$. (Ans)

(21) SubBytes with the given partial AES-S-box—

input word: [0x23, 0xA7, 0x4C, 0x19].

i) 0x23 → row 0x2, col 0x3 → 0xD4

ii) 0xA7 → row 0xA, col 0x7 → 0x63

iii) 0x4C → row 0x4, col 0xC → 0x2E

iv) 0x19 → row 0x1, col 0x9 → not shown in the Partial table (for reference, the standard AES-S-box maps 0x19 → 0xD4)

∴ output: [0xD4, 0x63, 0x2E, 0xD4].

(22) Given that:

Input key: [0x1A, 0x2B, 0x3C, 0x4D]

Round key word: [0x55, 0x66, 0x77, 0x88]

~~Now Multiply (XOR) to generate output —~~

~~i) 0x1A~~

Now Add RoundKey (XOR input with round key)—

i) $0x1A \oplus 0x55 = 0x4F$

ii) $0x2B \oplus 0x66 = 0x4D$

iii) $0x3C \oplus 0x77 = 0x4B$

iv) $0x4D \oplus 0x88 = 0xC5$

∴ output word : $[0x4F, 0x4D, 0x4B, 0xC5]$. Ans

(23) Given Matrix —

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix}$$

Input column: $[0x01, 0x02, 0x03, 0x04]$

~~Using AES field math~~

~~= (x02 + x~~

Using AES field math

= (x02 = x time, x03 = x time ⊕ identity).

$r_0 = 02 \cdot 01 \oplus 03 \cdot 02 \oplus 01 \cdot 03 \oplus 01 \cdot 04$

$= 0x02 \oplus 0x06 \oplus 0x03 \oplus 0x04$

$= 0x03$

$r_1 = 01 \cdot 01 \oplus 02 \cdot 02 \oplus 03 \cdot 03 \oplus 01 \cdot 04$

$= 0x01 \oplus 0x04 \oplus 0x05 \oplus 0x04$

$= 0x04$

$r_2 = 01 \cdot 01 \oplus 01 \cdot 02 \oplus 02 \cdot 03 \oplus 03 \cdot 04$

$= 0x01 \oplus 0x02 \oplus 0x06 \oplus 0x0C$

$= 0x09$

$r_3 = 03 \cdot 01 \oplus 01 \cdot 02 \oplus 01 \cdot 03 \oplus 02 \cdot 04$

$= 0x03 \oplus 0x02 \oplus 0x03 \oplus 0x08$

$= 0x0A$.

∴ Output column: [0x03, 0x04, 0x09, 0x0A]. Ans.

(24) AES - OFB mode -

i) Treats AES as a keystream gene

ii) Process :

IV → AES (key) → keystream

↓

cipher text ← XOR wit

iii) Decryption : same keystream

with ciphertext →

iv) Synchronization : Both end mus

same key

generate identical keystream.

(v) Error effect: 1-bit error in ciphertext only affect the same bit in plaintext (no propagation).

(25) AES modes and error propagation-

1) CBC (cipher block chaining):

→ 1-bit in ciphertext is → entire current block corrupted + 1-bit flipped in next block.

→ High error spread, so integrity is badly affected.

2) CFB (cipher feedback):

→ 1-bit error → corrupts current block segment and

flips same bit in next segment.

→ Less spread than CBC but, still alters multiple bytes

Impact: Both CBC and CFB cause decrypted data corruption if cipher text changes, making them unsuitable where bit errors are common useless integrity checks (MAC) are used

(26) Best AES Modes for large files with Parallel processing -

i) CTR (counter) Mode: Best choice.

→ Encrypts counters with AES to Produce keystream.

→ Each block is independent, perfect for parallel encryption or decryption.

→ Allows random access to encrypted data.

ii) ECB: Not secured (pattern leakage).

iii) CBC: secure but sequentially. CBC can't fully parallelize encryption.

Among them CTR is Fast, Parallel, no pattern leak, works for large files and streaming.

(27) Given that -

$M = 1$

$e = 5$

$n = 14$

$d = 11$

Encryption:

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

$$= 1^5 \bmod 14$$

$$= 1.$$

Decryption:

$$M = c^d \bmod n$$

$$= 1^{11} \bmod 14$$

$$= 1.$$

∴ ciphertext = 1.

Decrypted message = 1. (same as original).

(28) Given that -

$H(M) = 5$

$d = 3$

$n = 33$

Signature generation :

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

$$= 5^3 \bmod 33$$

$$= 125 \bmod 33$$

$$= 26.$$

∴ Signature 26 means, sender signs the hash using their private key d. Receiver verify with the public key.

(29) Diffie-Hellman key exchange:

Given that -

$p = 17,\ g = 3,\ a = 4;\ b = 5$

Aley's public key:

$$A = g^a \bmod p$$
$$= 3^4 \bmod 17$$
$$= 81 \bmod 17$$
$$= 13.$$

Badol's public key:

$$B = g^b \bmod p$$
$$= 3^5 \bmod 17$$
$$= 5.$$

Shared secret (At Aleya):

$$S = B^a \bmod p$$
$$= 5^4 \bmod 17$$
$$= 4.$$

Shared secret (At Badol):

$$S = A^b \bmod p$$
$$= 13^5 \bmod 17 = 4.$$

$\therefore$ shared secret = 4. (Ans)

(30) Simple hash function example:

Hash Rule: Sum of ASCII values of character mod 100

→ 'AB': 65+66 = 131.

= 131 mod 100

= 31

→ 'BA': 66+65 = 131

= 131 mod 100

= 31

collison: Different inputs → Same hash (31).

Implication:

weak hash → easy to find collisons → Bad for integrity and 'digital' signature.

(31) Given that -

message = 15, security key = 7.

$MAC = (message + Secret\ key) \bmod 17$

Step 1: Compute MAC -

$MAC = (15+7) \bmod 17$

$= 22 \bmod 17$

$= 5.$

original MAC is 5.

step 2: Attacket change forge message to 10:

$MAC = (10+7) \bmod 17$

$= 17 \bmod 17$

$= 0$

∴ Yes, if they Atteker see an exciting (message, MAC) pair, because the schema is linear, Attecker can Adjust MAC by the message difference. If attacker does not see any MAC or key, they cannot compute it.

(32) TSL Handshake and main steps & how symmetric keys are established -

i) client → server : ClientHellow (version, cipher suite)

ii) server to client : ServerHello, certificate, server key exchange, ServerHelloDone.

iii) client varrifies certificate, sends ClientKeyExchange.

iv) Both compute pre-master to master secret to derive symmetric keys via PRF.

v) Exchange change CipherSpec and Finished message to start encrypted traffic.

Assymetric crypto is used for server authentication and for security exchanging the Pre-master secret. Ephemeral DH gives forward security. Symmetric keys are derived from the agreed secret.

(33) SSH layered architecture:

i) Transport layer: confidentiality, integrity, server authentication, key exchange.

ii) User authentication layer: authenticates the client

iii) connection layer: Multiplexes many logical channels.

Transport layer:

i) Provide encryption.
ii) Handle key exchange
iii) Compress data optionally.

User Authentication layer:

i) Password based authentication
ii) Public key authentication
iii) Keyboard-interactive or multi-factor.

Connection Layer:

i) Multiplexes multiple logical channels over the single encrypted SSH connection.
ii) Manage flow control, channel closing, data forwarding.

(34) step in the TLS Handshake Process -

i) clientHello : client sends supported TLS version, cipher suites.

ii) serverHello : sends serverHelboDone to indicate it is ready for client response

iii) clientExchange : client encrypts Pre-master secret with server's public key.

iv) certificate Verify : Client may sends it's certificate and Prove Possession of Private key.

v) Key Derivation : Both sides user Pre-master secret + both nonces to generate the master secret.

vi) ExchangeCipherSpec : Client and server signal they are switching to the newly negotiated symmentic encryption.

(35) The Equation of Elliptic curve -

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

condition: $4a^3 + 27b^2 \not\equiv 0 \pmod{p}$

→ Why used in Kryptography -

i) Security based on Elliptic curve Discrete Logarithm Problem, which is hard to solve.

ii) Acheives same security with much smaller keys compared to RSA/DH.

iii) Smaller keys → faster computation, less memory, lower power and Ideal for constrained device.

(36) Why Ecc gives same security with smaller keys -

i) Ecc security relies on ECDLP, which currently has no sub-exponential time algorithm.

ii) Because ECDLP is harder per bit, smaller key sizes give same security.

iii) Example : 256-bit Ecc ≈ 3072-bit RSA

384-bit Ecc ≈ 7680-bit RSA

iv) Benefits: Faster encryption/decryption, less bandwidth, lower storage and power use.

(37) Given that :-

$y^2 = x^3 + 2x + 3 \mod 97$ where

Point $P = (3, 6)$ lies on the curve.

L.H.S $= y^2$

$= 6^2$

$= 36$

R.H.S $= x^3 + 2x + 3$

$= 3^2 + 2 \times 3 + 3$

$= 36$

Since L.H.S = R.H.S.

P is on the curve.

Proved.

(38) Given :-

$p = 23,\ g = 5,\ h = 8$

$m = 10,\ K = 6$

$c_1 = g^K \bmod p$

$= 5^6 \bmod 23$

$= 8$

$c_2 = m \times h^K \bmod p$

$= 10 \times 8^6 \bmod 23$

$= 10 \times 12 \bmod 23$

$= 120 \bmod 23$

$= 15$

∴ Ciphertext $(8, 15)$.

(39) Lightweight cryptography for IOT is important.

Because :-

i) IoT devices have tight constrains (CPU, memory, battery);

ii) lightweight crypto gives acceptable

security with low resource usage.

Example is PRESENT.

It's a lightweight block cipher.

Alternatives: Simon / Speck family.

Lightweight AEADs.

(40) The Three common IoT-specific attacks —

1. Firmware hijacking: Mitigation: Signed firmware, Secure boot, authenticated update channels, strick code integrity check.

2. physical tempering: Temper evidence hardware, Secure elements / TCM, device attestation, disable debug ports.

3. Botnet: Enforce strong Un

disable default passwords, ra

lockout, network segmentation.

. IDS/IPS