94

Q1:

Ans: Plaintext: Happy Holidays Key (Shift): 3

Q2:

(a) Ans:

To encrypt using a one-time pad, performing a bitwise XOR (exclusive OR) operation between the binary representations of the key and the plaintext

Step 1: Convert the plaintext and key to binary.

According to the given Table 1, the binary representation of Plaintext "dcliz":

d = 000 c = 001 l = 011 i = 100 z = 101

The binary representation for "clzyu":

c = 001 l = 011 z = 101 y = 111 u = 110

Step 2: XOR the binary values of the plaintext and key.

Plaintext	"dcliz"	000	001	011	100	101
Key	"clzyu"	001	011	101	111	110
XOR		001	010	110	011	011

Step 3: Convert the XOR result back to letters.

001 = c 010 = b 110 = u 011 = l 011 = l

Thus, the ciphertext is: "cbull"

(b) Ans: Using XOR the ciphertext with the plaintext to get the key:

Step 1: Convert the plaintext "yibc" and ciphertext "zuld" to binary.

Plaintext "yibc":	y = 111	i = 100	b = 010	c = 001
Ciphertext "zuld":	z = 101	u = 110	l = 011	d = 000

Step 2: XOR the ciphertext and plaintext to find the key.

Ciphertext	"yibc"	101	110	011	000
Plaintext	"zuld"	111	100	010	001
XOR		010	010	001	001

Step 3: Convert the XOR result back to letters.

010 = b 010 = b 001 = c 001 = c

Thus, the secret key is: "bbcc"

Q3:

(a) Ans: Alice's public key: (N, e) = (323, 5)

Alice's private key: d = 29

Message (M): 121

Step 1: Encrypt the message using Alice's public key (N, e).

The RSA encryption formula: $C = M^e \bmod N$

Substituting the values: $C = 121^5 \bmod 323$

Step 2: Compute 121^5 mod 323 (modular exponentiation).

We will break this into smaller steps using modular exponentiation:

- $121^2 = 14641$, and $14641 \bmod 323 = 77$
- $121^4 = (121^2)^2 = 77^2 = 5929$, and $5929 \bmod 323 = 108$
- $121^5 = 121 \times 121^4 = 121 \times 108 = 13068$, and $13068 \bmod 323 = 164$

Thus, the ciphertext C = 164.

Step 3: Decrypt the ciphertext using Alice's private key (N, d).

The RSA decryption formula: $M = C^d \bmod N$

Substituting the values: $M = 164^{29} \bmod 323$

Again, use **modular exponentiation** to compute $164^{29} \bmod 323$:

- $164^2 = 26896$, and $26896 \bmod 323 = 89$
- $164^4 = 89^2 = 7921$, and $7921 \bmod 323 = 154$
- $164^8 = 154^2 = 23716$, and $23716 \bmod 323 = 181$
- $164^{16} = 181^2 = 32761$, and $32761 \bmod 323 = 59$
- $164^{29} = 164 \times 89 \times 154 \times 181 \times 59 = 421434576$, and $421434576 \bmod 323 = 12$

  Thus, decryption complete and original message M = 121 is recovered.

(b) Ans: Message (M): 2

Alice's private key (d): 29

Public key (N, e): (323, 5)

Step 1: Alice signs the message.

The RSA signature is generated by: $S = M^d \bmod N$

Substituting the values: $S = 2^{29} \bmod 323$

Using modular exponentiation:

- $2^2 = 4$
- $2^4 = 16$
- $2^8 = 256$
- $2^{16} = 256^2 = 65536$, and $65536 \bmod 323 = 59$
- $2^{29} = 2 \times 4 \times 16 \times 256 \times 59 = 120832$, and $120832 \bmod 323 = 227$

  Thus, the signature S = 227.

Step 2: Bob verifies the signature.

To verify the signature, check if: $M = S^e \bmod N$

Substituting the values: $M = 227^5 \bmod 323$

Again, we use modular exponentiation:

- $227^2 = 51529$, and $51529 \bmod 323 = 76$
- $227^4 = 76^2 = 5776$, and $5776 \bmod 323 = 289$
- $227^5 = 227 \times 289 = 65503$, and $65503 \bmod 323 = 2$

  Since 2 = M, Bob can successfully verify the signature.

Q4:

(a) Ans: According to the A5/1 cipher, for X, Y, and Z:

- X: 1110101000101000101
- Y: 0100001001110110011101
- Z: 11010100101101001101000

The first keystream bit = $1 \oplus 1 \oplus 0 = 0$

- X: 1110101000101000101
- Y: 1010000100111011001110
- Z: 01101010010110100110100

  The second keystream bit = $1 \oplus 0 \oplus 0 = 1$

(b) Ans: After generating two keystream bits, the registers:

- X: 0111010100010100010
- Y: 1010000100111011001110
- Z: 10110101001011010011010

Q5:

(a) Ans: The Y register steps when its clocking bit matches the majority bit among x_8, y_{10}, and z_{10}. Each clocking bit has an equal chance of being 0 or 1, and the majority rule applies. The probability that the Y register steps can be broken down as:

For three-bit representation can be seen as 000 001 010 011 100 101 110 111.

For every register, the probability that only one register will be in the majority will happen in 6 out of 8 of the cases, as shown above. Where 6/8 = 3/4 = 75%. Thus, the Y register steps about **75% of the time**.

(b) Ans: All three registers step together when all their clocking bits are the same. This occurs when either 000 or 111.

P (all step) = P ($x_8 = y_{10} = z_{10} = 0$) + P ($x_8 = y_{10} = z_{10} = 1$) = 1/8 + 1/8 = 2/8 = 25%.

Thus, **all three registers step happens 25% of the time**.

(c) Ans: At least two registers step happens in cases where:

- All three registers step (which we already calculated as 25% in (b)).
- Exactly two out of three registers step.
  For two out of three registers step: 100, 010, 001, 011, 101, 011, that is 6/8 = 75%.
  Thus, the probability of two or more registers stepping (i.e., at least two) is 25% + 75% = **100%** of the time. This includes the case where all three step.

(d) Ans: No register steps when all the clocking bits are different from the majority bit, which is impossible in this system. The majority rule ensures that each step at least two registers are clocked. Thus, the probability that no register steps is **0%**.

Q6:

(a) F (R_i, K_i) = D (D is some constant)

- $L_1 = R_0$
- $R_1 = L_0 \oplus D$
- $L_2 = R_1 = L_0 \oplus D$
- $R_2 = L_1 \oplus D = R_0 \oplus D$
- $L_3 = R_2 = R_0 \oplus D$
- $R_3 = L_2 \oplus D = L_0 \oplus D \oplus D = L_0$
  Thus, $C = (L_3, R_3) = (R_0 \oplus D, L_0)$

(b) F (R_i, K_i) = K_i

- $L_1 = R_0$
- $R_1 = L_0 \oplus K_0$
- $L_2 = R_1 = L_0 \oplus K_0$
- $R_2 = L_1 \oplus K_1 = R_0 \oplus K_1$
- $L_3 = R_2 = R_0 \oplus K_1$
- $R_3 = L_2 \oplus K_2 = (L_0 \oplus K_0) \oplus K_2$
  Thus, the ciphertext: $C = (R_0 \oplus K_1, (L_0 \oplus K_0) \oplus K_2)$

(c) F (R_i, K_i) = $R_i \oplus K_i$

- $L_1 = R_0$
- $R_1 = L_0 \oplus R_0 \oplus K_0$
- $L_2 = R_1 = L_0 \oplus R_0 \oplus K_0$
- $R_2 = L_1 \oplus F(R_1, K_1) = L_1 \oplus R_1 \oplus K_1 = R_0 \oplus L_0 \oplus R_0 \oplus K_0 \oplus K_1 = L_0 \oplus K_0 \oplus K_1$
- $L_3 = R_2 = L_0 \oplus K_0 \oplus K_1$
- $R_3 = L_2 \oplus R_2 \oplus K_2 = L_0 \oplus R_0 \oplus K_0 \oplus R_2 \oplus K_2 = L_0 \oplus R_0 \oplus K_0 \oplus L_0 \oplus K_0 \oplus K_1 \oplus K_2 = R_0 \oplus K_1 \oplus K_2$

  Thus, the ciphertext: $C = (L_0 \oplus K_0 \oplus K_1, R_0 \oplus K_1 \oplus K_2)$

(d) Ans: No, it is not reasonable because the Feistel structure relies on the function F acting on R_i. Using $L_i \oplus K_i$ would violate the Feistel structure, as it would involve both halves of the input, making it harder to invert and defeating the purpose of a Feistel cipher.

Q7:

(a) Ans: **No**, Bob cannot detect the attack. In CBC mode, each ciphertext block depends on the previous ciphertext block. If C_2 is altered, it will cause a mismatch when Bob re-encrypts the plaintext and compares it to C_3. Specifically, the decryption of C_3 will be affected by the modified C_2, making the final comparison fail.

(b) Ans: If $P_0' = P_0$ and the IV remains the same, then the first block $C_0 = C_0'$. However, since the other plaintext blocks differ (P_1', P_2', P_3'), none of the subsequent ciphertext blocks C_1, C_2, C_3 will be the same as C_1', C_2', C_3'. Therefore, **only the first block C_0 will be equal**.

Q8:

(a) Ans: **No**, the multiple encryptions do **not increase security**. In RSA, the security is based on the difficulty of factoring the modulus N, which is the product of two large primes.

(b) Ans: The reason this does not improve security is that RSA decryption can still be achieved with one modular exponentiation using the combined decryption exponents. Multiple encryptions **using different exponents do not improve security because the final encryption**, which can still be decrypted in one step by computing the product of the decryption exponents: $C_2^{d2.d1.d0} \bmod N = M$.

Thus, Bob can compute the message directly without needing multiple decryption steps. The process is essentially equivalent to using a single exponentiation with the product of the exponents. The multiple encryption steps do not add complexity or strength to the encryption.

Q9:

Ans: **Yes**, the **man-in-the-middle attack can succeed** in this scenario if Trudy intercepts the Diffie-Hellman key exchange between Alice and Bob.

i. Trudy intercepts the communication between Alice and Bob.

ii. Trudy establishes a separate key with Alice and another key with Bob. For example, Trudy uses her private key *t* to compute $g^{at} \bmod p$ with Alice and $g^{bt} \bmod p$ with Bob.

iii. Alice and Bob believe they are communicating with each other securely, but they are each sharing a key with Trudy.

-6

Since Alice and Bob are unaware of Trudy's presence, they may believe they have established a secure shared key, but Trudy can intercept and decrypt their messages using her own keys. Therefore, **the attack succeeds** if no additional authentication measures are in place.

Q10:

(a) Ans: According to the given table:

- For $x_0 = 0$:
  i. $x_1x_2 = 00 \rightarrow y_0y_1 = 01 \rightarrow y_1 = 1 \neq x_0$
  ii. $x_1x_2 = 01 \rightarrow y_0y_1 = 00 \rightarrow y_1 = 0 = x_0$
  iii. $x_1x_2 = 10 \rightarrow y_0y_1 = 11 \rightarrow y_1 = 1 \neq x_0$
  iv. $x_1x_2 = 11 \rightarrow y_0y_1 = 10 \rightarrow y_1 = 0 = x_0$
- For $x_0 = 1$:
  i. $x_1x_2 = 00 \rightarrow y_0y_1 = 11 \rightarrow y_1 = 1 = x_0$
  ii. $x_1x_2 = 01 \rightarrow y_0y_1 = 10 \rightarrow y_1 = 0 \neq x_0$
  iii. $x_1x_2 = 10 \rightarrow y_0y_1 = 00 \rightarrow y_1 = 0 \neq x_0$
  iv. $x_1x_2 = 11 \rightarrow y_0y_1 = 01 \rightarrow y_1 = 1 = x_0$

Out of the 8 possible input combinations, $y_1 = x_0$ happens in 4 cases:

$$(x_0, x_1x_2) = (0, 01) \rightarrow y_1 = 0$$
$$(x_0, x_1x_2) = (0, 11) \rightarrow y_1 = 0$$
$$(x_0, x_1x_2) = (1, 00) \rightarrow y_1 = 1$$
$$(x_0, x_1x_2) = (1, 11) \rightarrow y_1 = 1$$

Thus, the probability is: $P(y_1 = x_0) = 4/8 = 1/2$.

(b) Ans: According to the given table:

- For $x_0 = 0$:
  i. $x_1x_2 = 00 \rightarrow y_0y_1 = 01 \rightarrow y_1 = 1 \neq x_1$
  ii. $x_1x_2 = 01 \rightarrow y_0y_1 = 00 \rightarrow y_1 = 0 = x_1$
  iii. $x_1x_2 = 10 \rightarrow y_0y_1 = 11 \rightarrow y_1 = 1 = x_1$
  iv. $x_1x_2 = 11 \rightarrow y_0y_1 = 10 \rightarrow y_1 = 0 \neq x_1$
- For $x_0 = 1$:
  i. $x_1x_2 = 00 \rightarrow y_0y_1 = 11 \rightarrow y_1 = 1 \neq x_1$
  ii. $x_1x_2 = 01 \rightarrow y_0y_1 = 10 \rightarrow y_1 = 0 = x_1$
  iii. $x_1x_2 = 10 \rightarrow y_0y_1 = 00 \rightarrow y_1 = 0 \neq x_1$
  iv. $x_1x_2 = 11 \rightarrow y_0y_1 = 01 \rightarrow y_1 = 1 = x_1$

Out of 8 cases, $y_1 = x_1$ happens in 4 cases:

$$(x_0, x_1x_2) = (0, 01) \rightarrow y_1 = 0$$
$$(x_0, x_1x_2) = (0, 11) \rightarrow y_1 = 1$$
$$(x_0, x_1x_2) = (1, 10) \rightarrow y_1 = 0$$
$$(x_0, x_1x_2) = (1, 01) \rightarrow y_1 = 1$$

Thus, $P(y_1 = x_1) = 4/8 = 1/2$.

(c) For each combination of $x_1$and x_2, calculate $x_1 \oplus x_2$ and compare it to y_0:

- For $x_0 = 0$:

i. $x_1x_2 = 00 \rightarrow x_1 \oplus x_2 = 0$ & $y_0y_1 = 01 \rightarrow y_0 = 0 \rightarrow y_0 = x_1 \oplus x_2$
ii. $x_1x_2 = 01 \rightarrow x_1 \oplus x_2 = 1$ & $y_0y_1 = 00 \rightarrow y_0 = 0 \rightarrow y_0 \neq x_1 \oplus x_2$
iii. $x_1x_2 = 10 \rightarrow x_1 \oplus x_2 = 1$ & $y_0y_1 = 11 \rightarrow y_0 = 1 \rightarrow y_0 = x_1 \oplus x_2$
iv. $x_1x_2 = 11 \rightarrow x_1 \oplus x_2 = 0$ & $y_0y_1 = 10 \rightarrow y_0 = 1 \rightarrow y_0 \neq x_1 \oplus x_2$

- For $x_0 = 1$:

i. $x_1x_2 = 00 \rightarrow x_1 \oplus x_2 = 0$ & $y_0y_1 = 11 \rightarrow y_0 = 1 \rightarrow y_0 \neq x_1 \oplus x_2$
ii. $x_1x_2 = 01 \rightarrow x_1 \oplus x_2 = 1$ & $y_0y_1 = 10 \rightarrow y_0 = 1 \rightarrow y_0 = x_1 \oplus x_2$
iii. $x_1x_2 = 10 \rightarrow x_1 \oplus x_2 = 1$ & $y_0y_1 = 00 \rightarrow y_0 = 0 \rightarrow y_0 \neq x_1 \oplus x_2$
iv. $x_1x_2 = 11 \rightarrow x_1 \oplus x_2 = 0$ & $y_0y_1 = 01 \rightarrow y_0 = 0 \rightarrow y_0 = x_1 \oplus x_2$

Out of the 8 possible input combinations, $y_1 = x_0$ happens in 4 cases:

$$(x_0, x_1x_2) = (0, 00) \rightarrow y_0 = 0$$
$$(x_0, x_1x_2) = (0, 10) \rightarrow y_0 = 1$$
$$(x_0, x_1x_2) = (1, 01) \rightarrow y_0 = 1$$
$$(x_0, x_1x_2) = (1, 11) \rightarrow y_0 = 0$$

Thus, the probability is: $P(y_0 = x_1 \oplus x_2) = 4/8 = 1/2$.

(c) Ans:

No, **this formula is not secure** because the same keystream *E (IV, K)* is used for each block. This means that the same keystream is XORed with different plaintext blocks P_i, which allows an attacker to perform the following attack:

- If an attacker knows or guesses part of one plaintext block P_i , they can recover the keystream *E (IV, K)* by XORing it with the corresponding ciphertext block C_i .
- Once the keystream is known, the attacker can decrypt other ciphertext blocks by XORing them with the same keystream, revealing the plaintext of other blocks.
- To make counter mode encryption secure, the keystream must be different for each block. The approach is to increment the IV for each block: $C_i = P_i \oplus E(IV + i, K)$, where $IV + i$ ensures that a different keystream is used for each block, preventing the reuse of the same keystream across different blocks.

Q11:

(a) Ans:

The **cube root attack** occurs when RSA encryption is performed with a low public exponent, such as $e = 3$, is used in RSA encryption and the message M is small enough. In RSA, the ciphertext C is calculated as: $C = M^e \bmod N$.

When M is small, M^e (i.e., M^3) may be less than the modulus N, meaning that no modular reduction takes place. In such cases, the ciphertext is simply $C = M^3$, and an attacker can compute the cube root of C to recover M directly.

The attack exploits the fact that if the message is small and no reduction by NNN occurs, the RSA encryption does not fully protect the message. The attacker can compute: $M = \sqrt[3]{C}$.

Thus, the attack works when $M^3 < N$, and this cube root can be computed easily without factoring N. For instance, if the message $M = 2$, and $e = 3$, the ciphertext will be $C = 2^3 = 8$. If $N > 8$, an attacker can compute the cube root of C (which is 2) to recover the plaintext directly.

(b) Ans:

- Compute the ciphertext C for M = 3:

$$C = M^3 \bmod N = 3^3 \bmod 33 = 27 \bmod 33 = 27$$

To check if the cube root attack works, calculate the cube root of C = 27:

$$\sqrt[3]{27} = 3$$

Since M = 3 is small and $3^3 = 27$ is less than N = 33, the cube root attack works, and an attacker can recover M = 3 by simply taking the cube root of 27.

- Compute the ciphertext C for M = 4:

$$C = M^3 \bmod N = 4^3 \bmod 33 = 64 \bmod 33 = 31$$

To check if the cube root attack works, we calculate the cube root of C = 31:

$$\sqrt[3]{31} \approx 3.14$$

The cube root of 31 is not an integer, so the cube root attack fails. The ciphertext C = 31 was reduced modulo N, which means the attacker cannot directly compute the cube root to recover M = 4.

(d) Ans:

To prevent the cube root attack, **message padding** is used.

Padding increases the size of the message before encryption, making sure that M^3 (or the cube of the padded message M′) exceeds the modulus N, thus preventing the attack.

When the padded message M′ is large enough, the cube of M′ will exceed N, ensuring that modular reduction occurs during encryption. This makes it computationally infeasible for an attacker to directly compute the cube root and recover the plaintext.

Common Padding Scheme: PKCS#1 v1.5: This scheme adds specific padding bytes to the original message M before encryption, which increases the length of M and ensures that it is large enough to prevent direct cube root attacks.

Padding Structure:

The padded message M′ is structured as follows:

M′ = 0x00 || 0x02 || random non-zero bytes || 0x00 || M

Where:

- First 0x00: A leading 0x00 byte is added to indicate the start of padding.
- 0x02: This byte indicates that padding is being used (0x02 is for public key encryption).
- Random non-zero bytes: A sequence of random non-zero bytes is added to fill up the required length. These random bytes ensure that the padded message is large enough.
- Second 0x00: A byte indicating the end of the padding.
- M: The original message.

The length of the padded message M′ is chosen to ensure that $(M')^3 > N$, making the cube root attack impossible.

Assume:

- The modulus N is 128 bytes.
- The message M is only 20 bytes long.

To prevent the cube root attack, we need to pad M so that M′ is much larger than M.

The random non-zero bytes fill up the space so that the total length of M′ is around 128 bytes. And after padding, the message is large enough that $(M')^3$ exceeds N, making the cube root attack ineffective.