Faculty of Engineering & Technology Electrical & Computer Engineering Department

Applied Cryptography– ENCS4320

HW #1

Prepared by:

Maha Maher Mali 1200746

Instructor: Dr. Mohammed Hussein

Section: 2

Date: 5-12-2023

Table of Contents

Table of Figures

Table of Tables

Acronyms and Abbreviations

E	Encryption Process
D	Decryption Process
M	Message
P	Plaintext (Same Message)
K	Key
V	OR Operation Symbol
$\oplus$	XOR Operation Symbol

Section A
Question 1

Given the ciphertext which is:

" **IURPWKHULYHUWRWKHVHDSDOHVWLQHZLOOEHIUHH**"

Required is to find the plain text (Message).

Step to solve the question:

- First, I placed the alphabet with the number as shown in figure1.

0	1	2	3	4	5	6	7	8	9	10	11	12
A	B	C	D	E	F	G	H	I	J	K	L	M
13	14	15	16	17	18	19	20	21	22	23	24	25
N	O	P	Q	R	S	T	U	V	W	X	Y	Z

Figure 1:Number for alphabet character [1]

- I have the ciphertext and what is required of me is to find the original message, so I will take letter by letter from the ciphertext and calculate the corresponding original text using this equation: where p: plaintext, c: cipher text, k:key

$$p = (c - k)\ mod\ 26$$

Solution:

I will take character by character from cipher text then find the number of characters from figure 1, then subtract the value of key I will tray all key from 0 to 25 until I got a message with clear meaning then mod 26, I will get the number then I will match this number to character in figure 1.

Shift (K=0)➔ IURPWKHULYHUWRWKHVHDSDOHVWLQHZLOOEHIUHH(Same C)

Shift 1 (K=1)➔ HTQOVJGTKXGTVQVJGUGCRCNGUVKPGYKNNDGHTGG(No meaning)

Shift 2 (K=2)➔ GSPNUIFSJWFSUPUIFTFBQBMFTUJOFXJMMCFGSFF(No meaning)

Shift 3 (K=3)➔ FROMTHERIVERTOTHESEAPALESTINEWILLBEFREE(have meaning)

Shift 4 (K=4)➔ EQNLSGDQHUDQSNSGDRDZOZKDRSHMDVHKKADEQDD(No meaning)

Shift 5 (K=5)➔ DPMKRFCPGTCPRMRFCQCYNYJCQRGLCUGJJZCDPCC(No meaning)

➔When I use shift 3 which is key=3 I got the message with cleat meaning, so I will use K=3 to obtain the message step by step as shown in below table.

First column solution	Second column solution
I:(8 - 3) mod 26=5➔F	O: (14 – 3) mod 26=11➔L
U: (20 – 3) mod 26=17➔R	H:(7-3) mod 26=4➔E
R: (17 – 3) mod 26=14➔O	V: (21 – 3) mod 26=18➔S
P: (15 – 3) mod 26= 12➔M	W: (22 – 3) mod 26= 19➔T
W: (22 – 3) mod 26= 19➔T	L: (11 – 3) mod 26=8➔I
K:(10-3) mod 26=7➔H	Q: (16 – 3) mod 26=13➔N
H:(7-3) mod 26=4➔E	H:(7-3) mod 26=4➔E
U: (20 – 3) mod 26=17➔R	Z:(25-3) mod 26=22➔W
L: (11 – 3) mod 26=8➔I	L: (11 – 3) mod 26=8➔I

Y: (24 – 3) mod 26=21➔V	O: (14 – 3) mod 26=11➔L
H:(7-3) mod 26=4➔E	O: (14 – 3) mod 26=11➔L
U: (20 – 3) mod 26=17➔R	E: (4 – 3) mod 26=1➔B
W: (22 – 3) mod 26= 19➔T	H:(7-3) mod 26=4➔E
R: (17 – 3) mod 26=14➔O	I:(8 - 3) mod 26=5➔F
W: (22 – 3) mod 26= 19➔T	U: (20 – 3) mod 26=17➔R
K:(10-3) mod 26=7➔H	H:(7-3) mod 26=4➔E
H:(7-3) mod 26=4➔E	H:(7-3) mod 26=4➔E
V: (21 – 3) mod 26=18➔S	----------
H:(7-3) mod 26=4➔E	----------
D: (3 – 3) mod 26=0➔A	----------
S: (18 – 3) mod 26=15➔P	----------
D: (3 – 3) mod 26=0➔A	----------
Plain text: FROM THE RIVER TO THE SEA PALESTINE WILL BE FREE	

Question 2
Part A

The number of possible keys for 56 -bit key size is: $\mathbf{2^{56}}$

➔Given that the computer performs 4.2 × 10^9 cycles per second per core and can test a key per CPU cycle. since there are 16 cores the total number of attempts per second is $16\ x\ 4.2\ x10^9$

$$Number\ of\ second = \frac{Total\ number\ of\ keys}{\text{Total number of Attempts}} = \frac{\mathbf{2^{56}}}{16\ x\ 4.2\ x10^9} = \mathbf{1072677.643\ sec}$$

To convert to year, we know:

- 60 second in minute.
- 60 minutes in hour.
- 24 hours in day.
- 365 days in year.

➔So, to find the number of years:

$$Number\ of\ year = \frac{\text{Time (in second)}}{60 \text{ x } 60 \text{ x } 24 \text{ x} 365}$$

$$Number\ of\ year = \frac{1072677.643}{60 \text{ x } 60 \text{ x } 24 \text{ x} 365} = \mathbf{0.034\ \ years}$$

Part B

The number of possible keys for 128-bit key size is: $\mathbf{2^{128}}$

➔ Given that the computer performs 4.2×10^9 cycles per second per core and can test a key per CPU cycle. since there are 16 cores the total number of attempts per second is $16\ x\ 4.2\ x10^9$

$$Number\ of\ second\ =\ Total\ number\ of\ keys$$

$$Number\ of\ second = \frac{Total\ number\ of\ keys}{\text{Total number of Attempts per second}} = \frac{\mathbf{2^{128}}}{16\ x\ 4.2\ x10^9} = \mathbf{5.06369\ x\ 10^{27}\ sec}$$

To convert to year, we know:

- 60 second in minute.
- 60 minutes in hour.
- 24 hours in day.
- 365 days in year.

➔So, to find the number of years:

$$Number\ of\ year = \frac{\text{Time (in second)}}{60 \text{ x } 60 \text{ x } 24 \text{ x} 365}$$

$$Number\ of\ year = \frac{5.06369 \text{ x } 10^{27}}{60 \text{ x } 60 \text{ x } 24 \text{ x} 365} = \mathbf{1.60568\ x\ 10^{20}\ years}$$

Question 3

Perfect Secure Definition: regardless of any prior information the attacker has about the plaintext, the ciphertext should not take additional information. And the main condition is the key space should greater than or equal the message space.

$$|k| \geq |M|$$

When Alice removes the zero set from the key space so the key space will less than the message space, and the key is likely to be repeated so the most important condition for one time pad has been violated.so the system **will not be perfect secure**.

Example:

- For 2 bits.
- When message is {00,01,10,11} so the message space is 4
- The key space will be {01,10,11}so the key space is 3 bit.
- So, the key space less than message space, so **it's not perfect secure.**

Using probability: second condition one-time pad is considered perfect secure if it's satisfying this condition:

$$prior\ information \equiv posterior\ information$$

$$pr[M = m] = pr[M = m \mid C = c]$$

For any message M and any ciphertext C, the probability of an attacker guessing the correct message M given the ciphertext C is equal to the probability of guessing any other message.

The solution using probability in details show in figure 3.

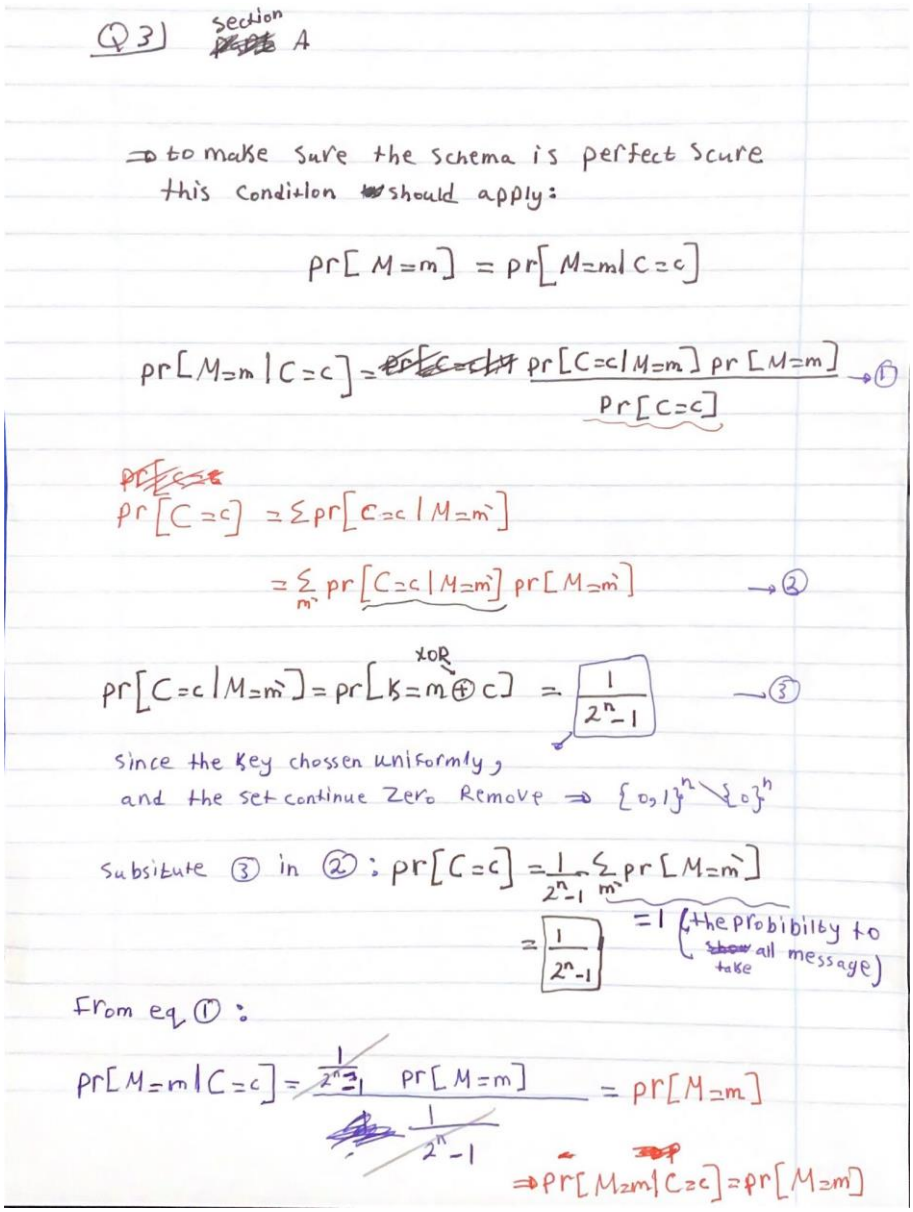

Figure 2:Qustion 3 Solution

Because the first condition which is |k|≥|M| not achieved, so the scheme still not perfect secure.

Question 4

Part A

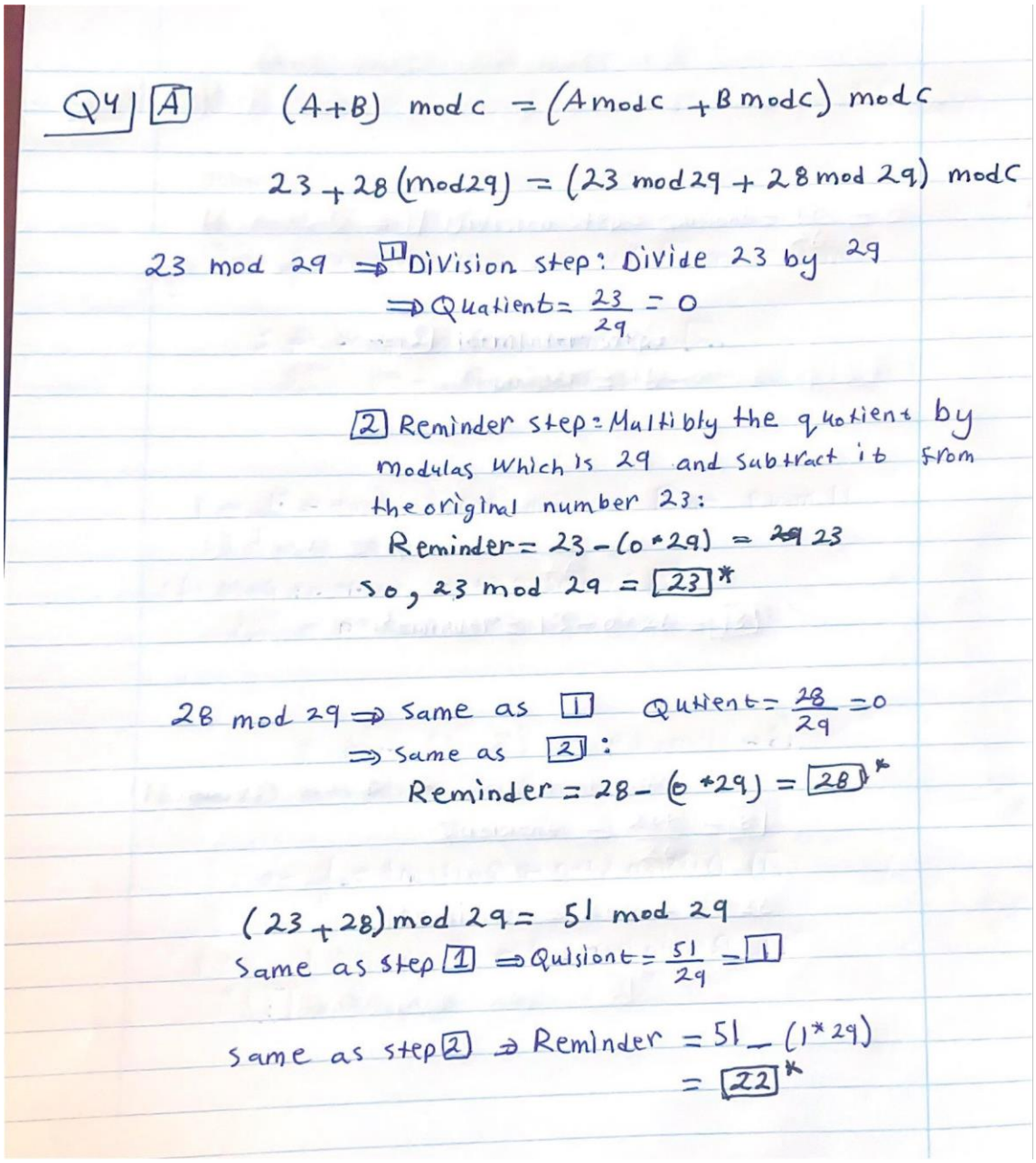

Figure 3: Question 4-part A solution

Part B

Q4 [b] $(A-B) \bmod C = (A \bmod C - B \bmod C) \bmod C$

$3-11 \bmod 9 = (3 \bmod 9 - 11 \bmod 9) \bmod 9$

$3 \bmod 9 \Rightarrow$ [1] Division step: Quitient $= \frac{3}{9} = 0$

[2] Reminder step as Part A:

$\Rightarrow$ Reminder $= 3 - 0 * 9 \Rightarrow$ [3]

$11 \bmod 9 \Rightarrow$ [1] Division step: Quitient $= \frac{11}{9} = 1$

[2] Reminder step same as part A:

$\Rightarrow$ Reminder $= 11 - 1 * 9 =$ [2]

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

$= 1 \bmod 9$

[1] Division step $\Rightarrow$ Quitient $= \frac{1}{9} = 0$

[2] Reminder step $\Rightarrow$ Reminder $= 1 - 0 * 9$

$=$ [1] *

Figure 4 :Question 4-part B solution

Part C

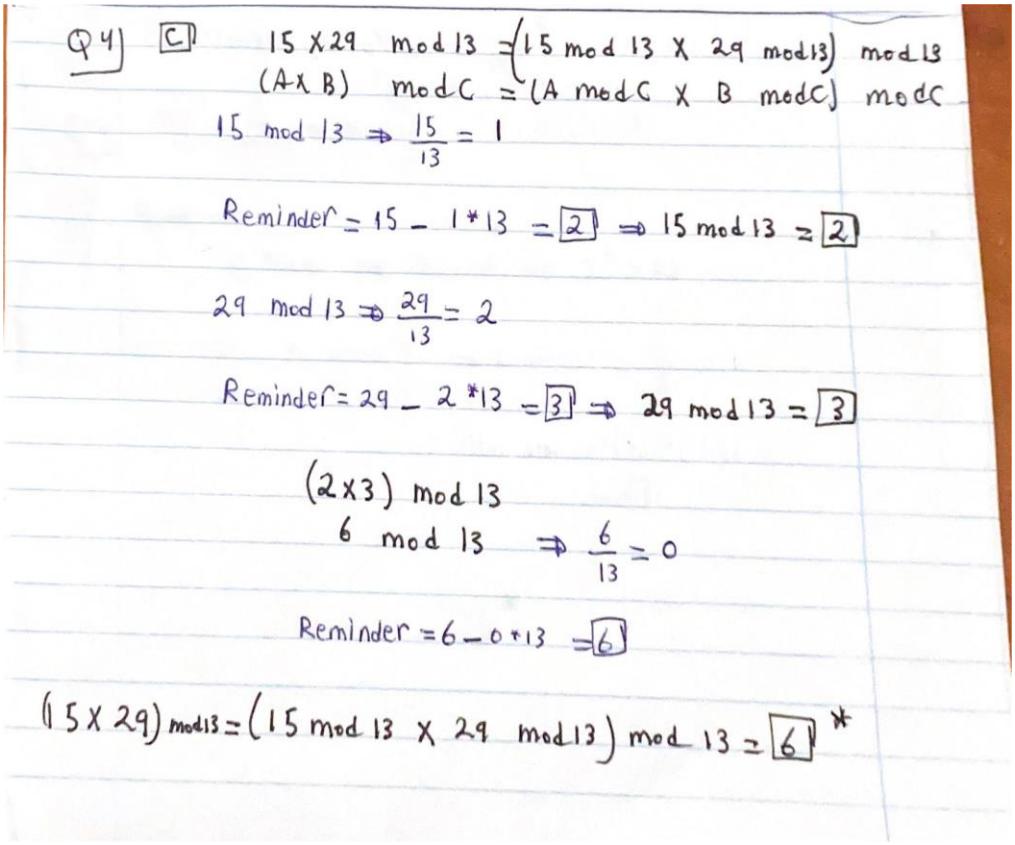

Figure 5: Question 4-part C solution

Part D

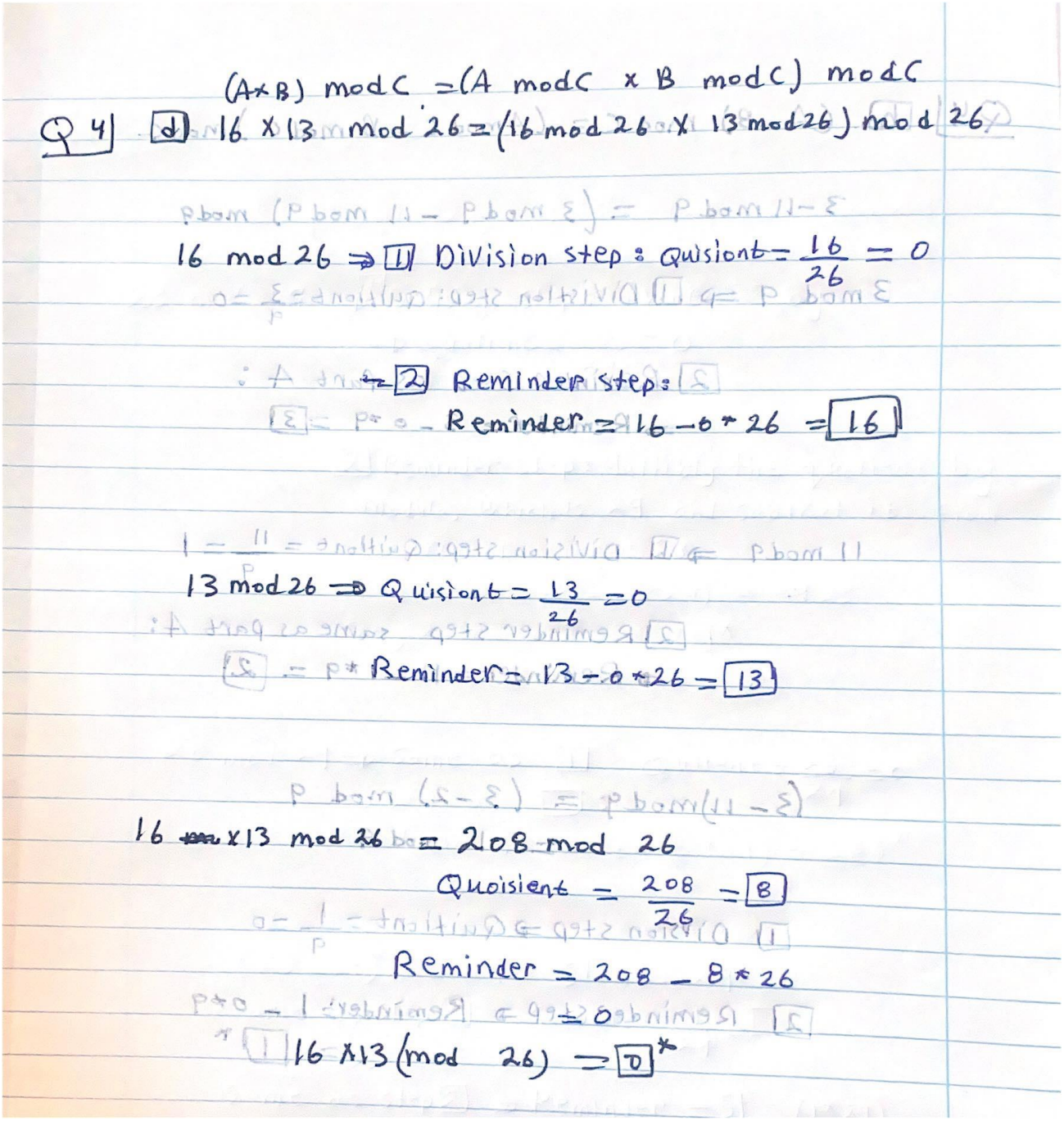

Figure 6: Question 4-part D solution

Part E

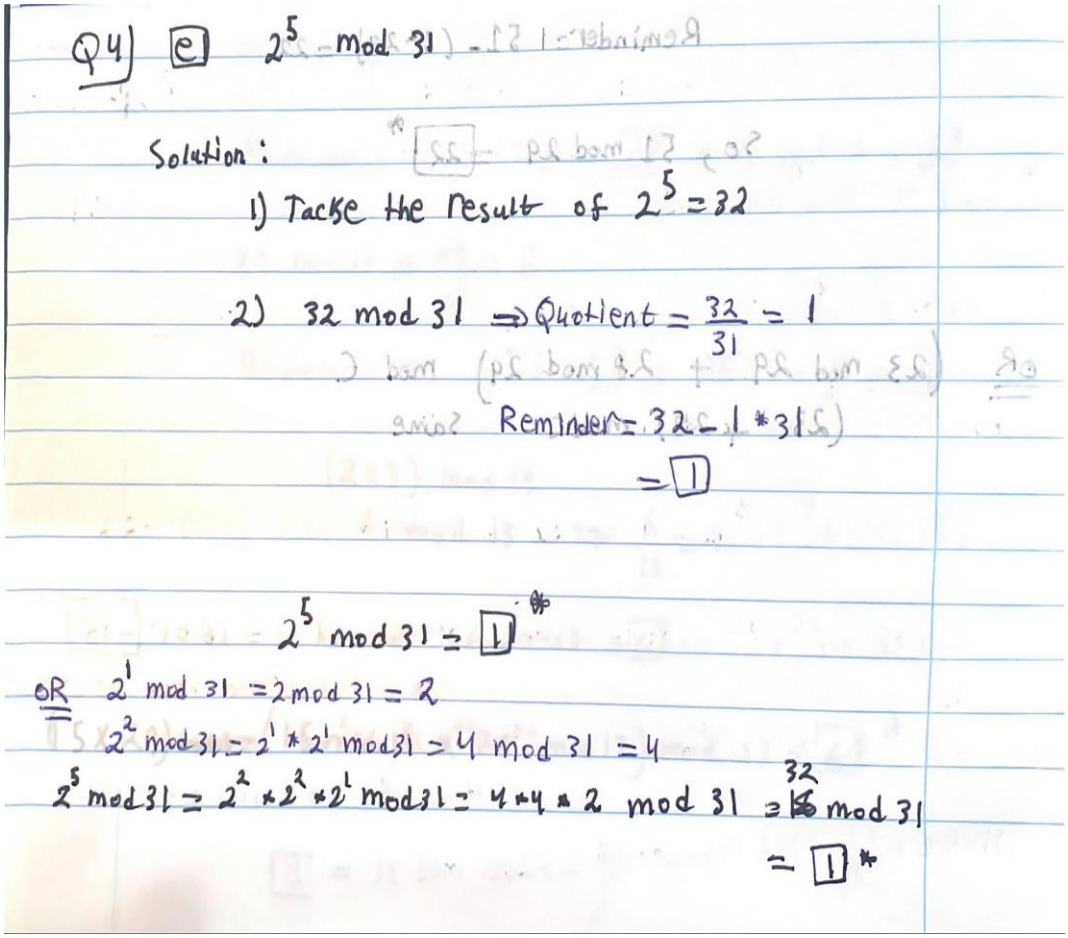

Figure 7: Question 4-part E solution

Part F

Figure 8: Question 4-part F solution

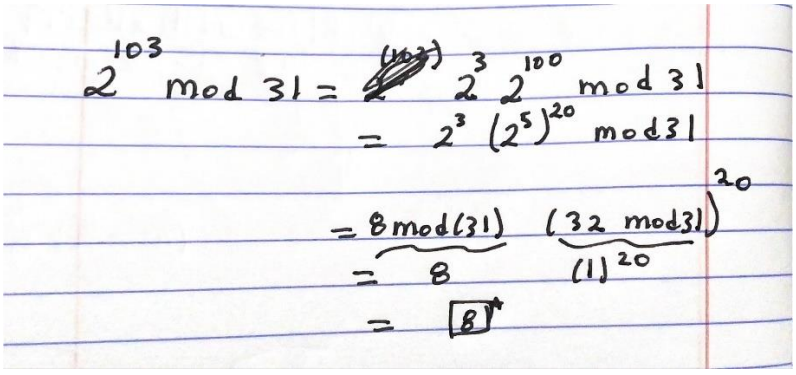

Figure 9: Question 4-part F another solution

Question 5

Table 1: letter encodings

Letter	E	H	I	k	l	r	s	t
Binary	000	001	010	011	100	101	110	111

Given ciphertext which is **"KITLKE"** so I will convert it to binary according table 1:

- Ciphertext(C)= "KITLKE" = 011 010 111 100 011 000, I will use this result to solve part A and B.

Part A

Given plaintext(P) which is **"thirst"** so I will convert it to binary according table 1:

- Plaintext(P) = "thirst" = 111 001 010 101 110 111, this result I will take it and XOR with cipher text C to obtain the key.

- Key(K)= C (XOR) P, the result shown in below figure.

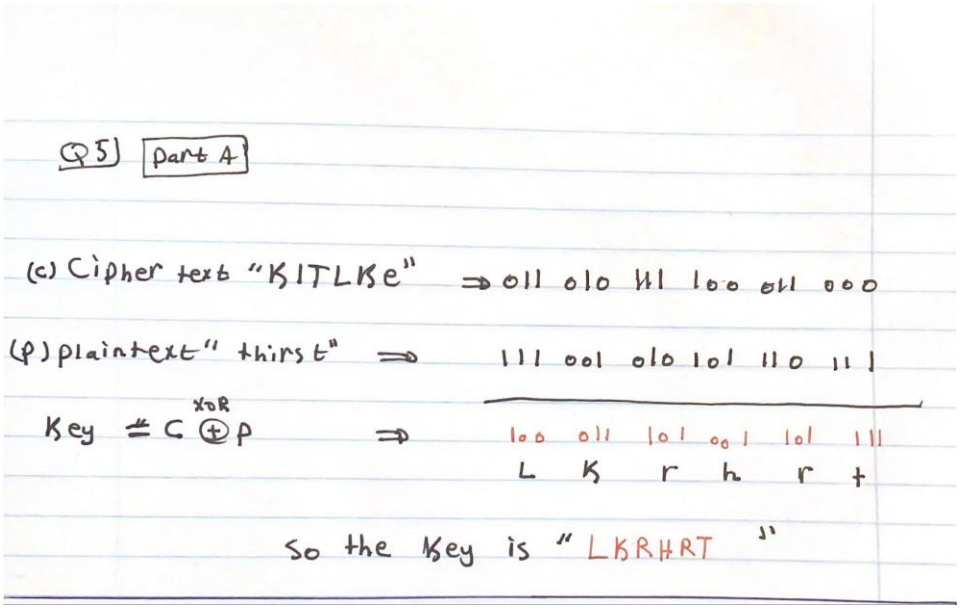

Figure 10: Question 5-part A solution

Part B

Given plaintext(P) which is **"hikers"** so I will convert it to binary according table 1:

- Plaintext(P) = "hikers" = 001 010 011 000 101 110, this result I will take it and XOR with cipher text C to obtain the key.
- Key(K)= C (XOR) P, the result shown in below figure.

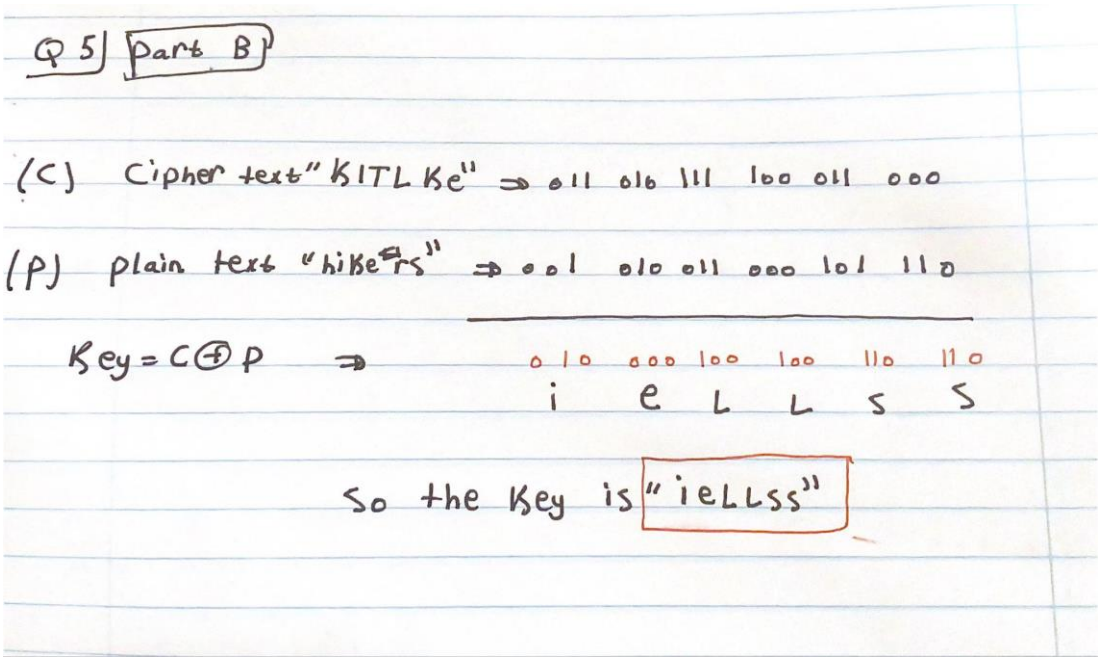

Figure 11: Question 5-part B solution

Question 6

Part A

we used the XOR operation in One time pad scheme, because it provides perfect secrecy when the key stream is truly random, used only once time and is as long as the message.

If we try to replace XOR with OR in the described scheme, the system would not work effectively in encryption. Because XOR operation has a unique property that makes it suitable for stream cipher encryption, particularly in scenario like one time pad.

- XORing a bit with 0 leaves it unchanged :$0 \oplus 0=0$, and $1 \oplus 0=1$.
- XORing a bit with 1 flips its value :$0 \oplus 1=1$, and $1 \oplus 1=1$.

This property is crucial to the security of the one-time pad because it ensure that the result of the XOR operation is unpredictable and the key stream is used only once. If we want to replace XOR by OR:

- ORing a bit with 0 always the result is 1 :$0 \vee 0=0$, and $1 \vee 0=1$.
- ORing a bit with 1 leaves it unchanged:$0 \vee 1=1$, and $1 \vee 1=1$.

This means if we use OR instead of XOR, the encryption would not have the same security properties. Because when we use OR operation the ciphertext would be more predictable, and the attacker might be able to discern patterns in the ciphertext, making it in danger to attack.

If we use AND instead of XOR, the situations are the same. A less secure encryption system might also be produced by ANDing with a stream of random bits since ANDing prefers to zero out bits rather than connect them.

Example: Suppose Alice wants to encrypt the binary message which is M=1010, using stream random bits (Key) K=1101, the long of message is the same with the long of keys.

1. **If we use XOR operation.**
    - C=M⊕K➔C=1010⊕ 1101=0111➔**C=0111 (Ciphertext)**
2. **If we use OR operation.**
    - C=M ∨ K➔ C=1010 ∨ 1101=1111➔**C=1111 (Ciphertext)**
3. **If we use AND operation**
    - C=M & K➔ C=1010 & 1101=1000➔**C=1000 (Ciphertext)**

Now suppose Bob receive the ciphertext and the same stream of random bits K,and he want to make decryption process.

1. **If we use XOR operation.**
    - M=C⊕K➔M=0111⊕ 1101=1010➔**M=1010 (Message)**
2. **If we use OR operation.**
    - M=C+ K➔M=1111+1101=1111➔**M=1111 (Message)**
3. **If we use AND operation**
    - M=C . K➔M=1000 . 1101=1000➔**M=1000 (Message)**

The original message M=1010, and pop receive this message only when we use the XOR operation. If we replace XOR operation by OR, AND operation the decryption process does not correctly recover the original message. **So, the scheme doesn't work with OR, AND.**

Part B

At first, I will find the key space which refer to the total number of possible keys that can be used in cryptographic system:

- Since the key is selected from the set {0,1,2}, and the message M is also chosen from the set {0,1,2}

- Key space =3 x 3=9, this is because for each of the three possible values of K, there are three possible values of M, but the key space **very small so it is possible that attackers are attacking the system.**

The key is designed to be used once in one time pad. The security of this technique is damaged if the same key is used for several message. If the key is used more than once, the XOR process cannot guarantee full secure.

Because we have only 9 key space which is **small**, the attacker could perform an attack the system to try all combinations and recover the key.

So, this given scheme does not have the security guarantees of a one-time pad. this figure below lists the resulting encrypted message using given scheme. We can see that some outcomes exclude certain inputs. For example, given E (K, M) =11 an attacker knows that the sent message M is not 0.

K	M	$E(K,M)$
00	00	00
01	00	01
10	00	10
00	01	01
01	01	00
10	01	11
00	10	10
01	10	11
10	10	00

Figure 12: Question 6-part B

The one-time pad offers more security guarantees than this approach. Perfect secrecy is ensured in the one-time pad as long as the key is truly random, used just once, and as long as the message. This approach does not reach the same level of security and is at attack by attacker because of its limit key space and possibilities for key reusing.

Part C

We want to create a new encryption method with the same security guarantees as the one-time pad, which is E (·, ·). We expect that the attacker will not be able to obtain any information about M given E (K, M). This condition applies to every uniform form E (K, M) in the set {0,1,2}. The encryption algorithm is E (K, M) =(M+K) mod 3.

K	M	E (K, M)
00	00	00
01	00	01
10	00	10
00	01	01
01	01	10
10	01	00
00	10	10
01	10	11
10	10	00

Figure 13 Question 6-part C solution

➔ When K=01 and M=00:

- E (M, K) =(K+M) mod 3= (01+00) mod 3=01

➔ When K=10 and M=01:

- E (M, K) =(K+M) mod 3= (10+01) mod 3=01

You can see that in both situations, the outcomes are similar which is 01. Because of this feature, every possible outcome is guaranteed to be equally likely, making it more difficult for an attacker to find out the original message M.

It is important to remember that this encryption algorithm's security depends on the key K being kept secret, used only once, and randomly generated. The security of the encryption may be damaged if the key is shared or not kept private.

Section B

Question 7

An attacker executing a chosen plaintext attack selects a collection pf plaintext and decrypts the matching cyphertexts. The attacker attempts to determine the encryption key by looking at the links between selected plaintexts and the ciphertexts that match them.

1. **Shift Cipher**: every letter in the plaintext is moved by a certain number of places in the alphabet when we using a shift cipher. An attacker can experiment with various plaintexts in a chosen plaintext attack and see what the ciphertexts result. As there are only 25 potential keys, attempting every combination will allow the attacker to figure out the key with easy. **As a result, the shift cipher can be broken just one selected plaintext**. For the shift cipher: given a single plaintext character P and ciphertext character C, so the key is simply: K=(C-P) mod 26, so, the encryption of only a single plaintext character thus suffices to recover the key.

2. **Substitution Cipher:** every letter in the plaintext is changed with a different letter in a substitution cipher based on a fixed mapping. A chosen plaintext attack allows an attacker to experiment with different inputs and see the related outputs. Much like the shift cipher. The relationship between selected plaintexts and ciphertexts may be analyzed by the attacker to find the key because there are only a limited number of possible mappings. **Again, the s substitution cipher may be broken with just one selected plaintext.**

3. **Vigenère Cipher:** because the Vigenère cipher uses a keyword to determine the shift applied to each letter in the plaintext so the Vigenère cipher is more complex than the shift and substitution cipher. A chosen plaintext attack, however is still possibly successful. An attacker can determine the length of the keyword by choosing a correctly lengthy plaintext and looking at the matching ciphertext. They can do this by analyzing the patterns that appear. Once the length of the keyword has been determined, the encryption can be broken using frequency analysis and other methods. The length of the keyword determines how much selected plaintext is required, however in reality, a little quantity of chosen plaintext could be enough

to find the key. **The Vigenère cipher may require more than one chosen plaintext, depending on the length of the keyword.**

Question 8

The key of the Vigenère cipher is a repeated keyword, and the term period refers to the period of use the keyword. The Vigenère cipher use 26 x 26 table of a to z row and a to z Colum.

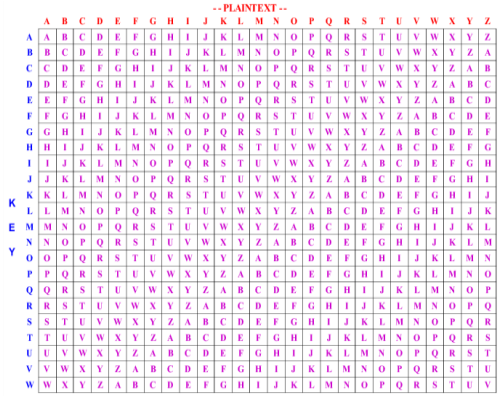

Figure 14: Vigenère table

Part A

a) The Vigenère cipher using period 2

- **For password "abcd"**

  ➔suppose the key is "gh", so I will repeat the key to be the same length of message so the key will be "ghgh". Now from Vigenère cipher table as shown in figure 12 I will shift **a** with key **g** and **b** with key **h**, also **c** with key **g**, and **d** with key **h**.

  a➔key g➔ g

  b➔key h ➔ i

  c➔key g➔ i

  d➔key h ➔ k

  the cipher text is "**giik**".

- **For password "bedg"**

  ➔suppose the key is "gh", so I will repeat the key to be the same length of message so the key will be **"ghgh".** I will shift **b** with key **g** and **e** with key **h**, **d** with key **g**, and **g** with key **h.**

  b→ key g→ h

  e→ key h→ l

  d→ key g→ j

  g→ key h→ n

  the cipher text is "**hljn**"

- **How attacker determine the user password?**

  The attacker now the password is either **"abcd"** or **"bedg",** also the attacker sees the resulting of ciphertext.

  The cipher text "**giik**" the first letter**(g)** and third letter**(i)** have the same amount of shift which is**(g)**, also the second letter **(i)** and the fourth letter**(k)** have the same amount of shift which is **(h).** The same applies to "**hljn**" cipher text, **first letter = third letter in shift**, and **second letter=fourth letter in shift.**

  From given password the difference between letter according to letter number **in figure 1**, in password **"abcd"** ab=cd=1, but in **"bedg"** be ≠ dg.

  ➔so, in the ciphertext **"giik"** find the difference between letter according to figure 1, so **gi= (8-6) =2 and ik= (10-8) =2, so** the attacker knows the password is **"abcd"**

  ➔so, in the ciphertext "**hljn**"find the difference between letter according to figure 1, so **hl= (11-7) =4 and jn=(13-9)=4 ,**so the attacker know the password is **"abcd".**

  If the shift between two first letter does not equal the shift of last two letters, then

the attacker knows the password is "bedg".

Part B

b) The Vigenère cipher using period 3

- **For password "abcd"**

  ➔suppose the key is **"ghk",** so I will repeat the key to be the same length of message so the key will be **"ghkg"**. Now from Vigenère cipher table as shown in figure 12 I will shift **a** with key g and **b** with key **h**, also **c** with key k, and **d** with key **g**.

  a→key g→ g

  b→ key h→ i

  c→key k → m

  d→ key g → j

  the cipher text is "**gimj**"

- **For password "bedg"**

  ➔suppose the key is "ghkg", so I will repeat the key to be the same length of message so the key will be "ghgk". I will shift **b** with key **g** and **e** with key **h**, **d** with key **k**, and **g** with key **g .**

  b→ key g → h

  e→ key h→ l ,

  d→ key k→ n

  g→ key g→ m

  the cipher text is "**hlnm**".

- **How attacker determine the user password?**

  The attacker now the password is either **“abcd”** or **“bedg”,** also the attacker sees the resulting of ciphertext.

  The cipher text **“gimj”** the first letter**(g)** and fourth letter**(j)** have the same amount of shift which is **(g), first letter=fourth letter in shift.** Also, the same for “**hlnm**” ciphertext.

  From given password the difference between letter according to letter number **in figure 1**, in password **“abcd”** ad= (3-0) =3. In password **“bedg”** bg = (6-2) =5.

  ➔ So, in the ciphertext **“gimj”** find the difference between the first and fourth letter according to figure 1, so **gj= (9-6) =3, so** the attacker knows the password is **“abcd”.**

  ➔ In the ciphertext “**hlnm**” find the difference between the first and fourth letter according to figure 1, so **hm= (12-7) =5, so** the attacker knows the password is **“bedg”.**

  So, the attacker knows which password the sender uses by knowing the difference between the first and last letter.

Part C

c) The Vigenère cipher using period 4: in this case the length of key will be 4, suppose the key is **“ghko”.**

So, in this case **we don’t have any relation between the input and ciphertext**, it’s difficult to an attacker to know what is the password. So, to prevent the attacker to know the password the key should equal or greater than password “plaintext”, because this way is more secure the attacker will not manage to know the “plaintext”

Section C
Mono-Alphabetic Substitution Cipher

A mono-alphabetic cipher is a substitution cipher where each letter of the plain text is replaced with another letter of the alphabet. It uses a fixed key which consist of the 26 letters of a "shuffled alphabet". [2]

This type of cipher is a form of symmetric encryption as the same key can be used to both encrypt and decrypt a message. [2]

One approach used to help decrypt a mono-alphabetic substitution cipher is to use a frequency analysis based on counting the number of occurrences of each letter to help identify the most recurrent letters. (e.g., In the English language, letters E, T and A). [2]

Frequency analysis consists of counting the occurrence of each letter in a text. Frequency analysis is based on the fact that, in any given piece of text, certain letters and combinations of letters occur with varying frequencies. For instance, given a section of English language, letters E, T, A and O are the most common, while letters Z, Q and X are not as frequently used.[3]

Algorithm for the code

- Take the cipher text which is given from the question.
- Count the frequency of each letter of the alphabet in the ciphertext.

The frequencies of the English language are:

E	T	A	O	I	N	S	H	R	D	L	C	U	M	W	F	G	Y	P	B	V	K	J	X	Q	Z
12.7	9.1	8.2	7.5	7.0	6.7	6.3	6.1	6.0	4.3	4.0	2.8	2.8	2.4	2.4	2.2	2.0	2.0	1.9	1.5	1.0	0.8	0.15	0.15	0.10	0.07

The frequencies of the intercept are:

F	Q	W	G	L	O	V	H	B	P	I	J	R	Z	E	M	A	C	K	Y	D	S	X	N	T	U
37	26	21	19	17	16	15	14	12	10	9	9	7	7	4	4	3	3	3	3	2	2	1	0	0	0
15.2	10.7	8.6	7.8	7.0	6.6	6.1	5.7	4.9	4.1	3.7	3.7	2.9	2.9	1.6	1.6	1.2	1.2	1.2	1.2	0.8	0.8	0.4	0.0	0.0	0.0

Figure 15: frequency of each letter [4]

- low-frequency letters in ciphertext can be guessed as low-frequency letters in plaintext.

This step would be enough if our message were long enough.

- Use known English letter frequencies as reference.
- Compare the observed frequencies in the ciphertext with the expected frequencies in English language.
- Start creating a temporary key by matching the most common letters in the ciphertext to the most common letters in the English language.

Output of the code

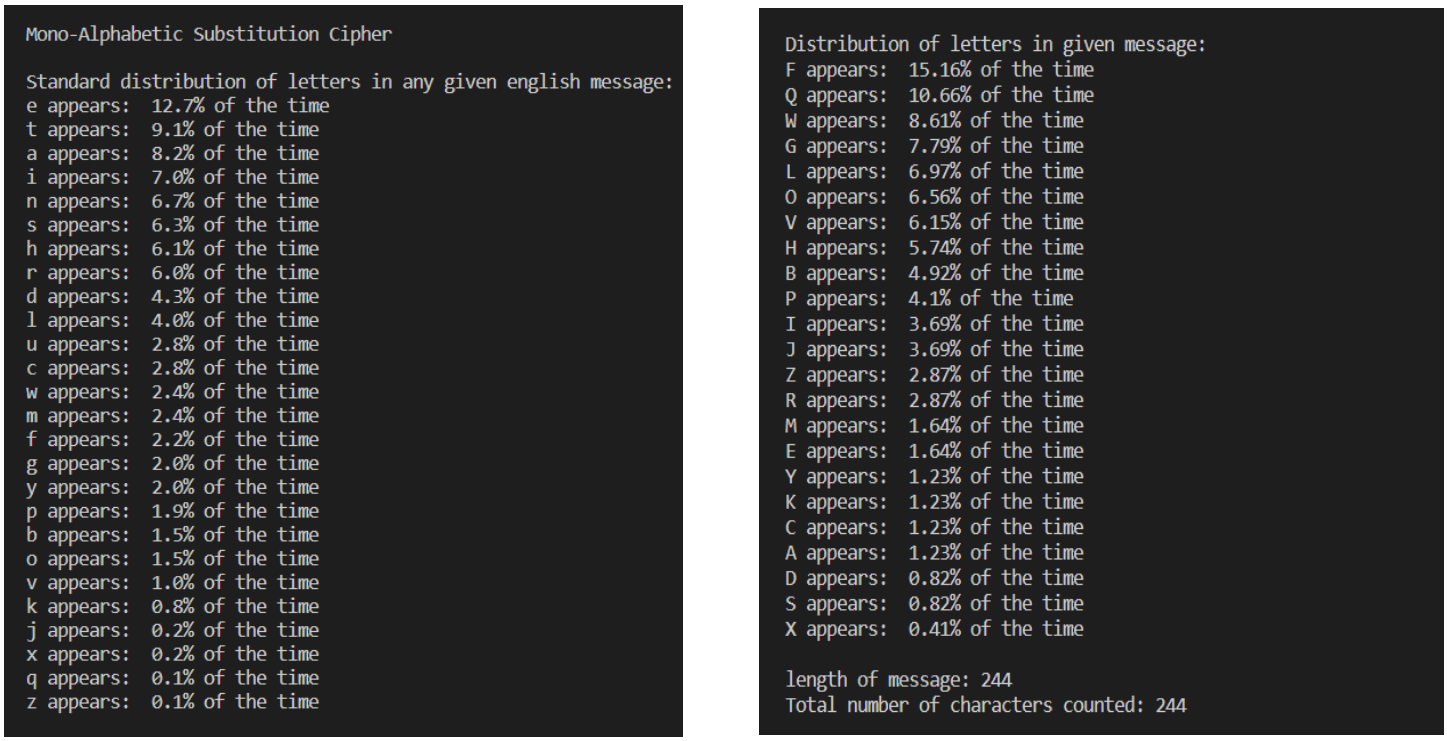

Figure 16:Code output

```
Based on the standard character distribution, it is possible that:

F = e
Q = t
W = a
G = i
L = n
O = s
V = h
H = r
B = d
P = l
I = u
J = c
R = w
Z = m
E = f
M = g
A = y
Y = p
C = b
K = o
D = v
S = k
X = j
```

```
Key Mapping:
F = e
Q = t
W = a
G = i
L = n
O = s
V = h
H = r
B = d
P = l
I = u
J = c
Z = w
R = m
M = f
E = g
C = y
A = p
K = b
Y = o
S = v
D = k
X = j
```

Figure 17:Code output

```
Decrypted Text:
cimftsoirfhdcamatelarieektielewmbdggdcuwttsyudwbnepeitheweaagsiasleierasnlrnmnsnekfeitadnadatsnbeadondnonevencimftdsnacheleathrtaeelts
theltsyelsieaecuiethrnrnmstheiachelesneriththeungsitunrtetiuthhsvepeidathrtauchachelearieuaurwwmtidpdrwtsyierj
PS C:\Users\Lenovo>
```

Figure 18:Code output

Code

```
def main():

    print()
    print("Mono-Alphabetic Substitution Cipher " + "\n")

    # Start by creating a list of known distributions of letters in the english language.
    letterDistribution = {
        "e": 12.7,
        "t": 9.1,
        "a": 8.2,
        "i": 7.0,
        "n": 6.7,
        "s": 6.3,
        "h": 6.1,
        "r": 6.0,
        "d": 4.3,
        "l": 4.0,
        "u": 2.8,
        "c": 2.8,
        "w": 2.4,
        "m": 2.4,
        "f": 2.2,
        "g": 2.0,
        "y": 2.0,
        "p": 1.9,
        "b": 1.5,
        "o": 1.5,
        "v": 1.0,
        "k": 0.8,
        "j": 0.2,
        "x": 0.2,
        "q": 0.1,
        "z": 0.1
        }

    #sort the above dictionary by value in ascending order
    print("Standard distribution of letters in any given english ciphertext: " )
    for w in sorted(letterDistribution, key=letterDistribution.get, reverse=True):
        print(w + " appears: ", str(letterDistribution[w]) + "% of the time")
```

```
# given ciphertext ciphertext to attempt to decrypt.
ciphertext = ('JGRMQOYGHMVBJWRWQFPWHGFFDQGFPFZRKBEEBJIZQQOCIBZKLFAFGQVF'
'ZFWWEOGWOPFGFHWOLPHLRLOLFDMFGQWBLWBWQOLKFWBYLBLYLFSFLJGRMQBOLWJVFPF'
'WQVHQWFFPQOQVFPQOCFPOGFWFJIGFQVHLHLROQVFGWJVFPFOLFHGQVQVFILEOGQILHQ'
'FQGIQVVOSFAFGBWQVHQWIJVWJVFPFWHGFIWIHZZRQGBABHZQOCGFHX')

# create an unordered set out of chars in ciphertext
s = set(ciphertext)

#sum up total number of characters counted
sum = 0;

#try and sort the set
sorted(s)

print("\n")
print("Distribution of letters in given ciphertext: " )

# construct a dictionary made of letters in the ciphertext as keys
# and frequency at which they occur as values
d = {}
for ch in s:
    if ch in ciphertext:
        sum += ciphertext.count(ch)
        d[ch] = round(((ciphertext.count(ch) / len(ciphertext)) * 100), 2)

#sort the dictionary by value instead of key
for w in sorted(d, key=d.get, reverse=True):
    print(w + " appears: ", str(d[w]) + "% of the time")

print()
#print length of ciphertext
print("length of ciphertext: " + str(len(ciphertext)))
#print total characters counted, make sure they match with
#ciphertext length
print("Total number of characters counted: " + str(sum))

print("\n")
print("Based on the standard character distribution, it is possible that: " + "\n")
```

```
    dS = sorted(d, key=d.get, reverse=True)
    stdS = sorted(letterDistribution, key=letterDistribution.get, reverse=True)

    k1 = letterDistribution.keys()

    #print(k1)

    for i in range(len(dS)):
        print(dS[i] + " = " + stdS[i])


    # Print the key mapping
    print("Key Mapping:")
    for i in range(len(dS)):
        print(dS[i] + " = " + stdS[i])

    # Decrypt the ciphertext using the obtained key
    key_mapping = {}
    for i in range(len(dS)):
        key_mapping[dS[i]] = stdS[i]

    # Print the obtained key
    print("\nObtained Key:")
    for char in key_mapping:
        print(f"{char} = {key_mapping[char]}")

    # Decrypt the ciphertext using the obtained key
    decrypted_text = "".join(key_mapping.get(char, char) for char in ciphertext)

    # Print the decrypted text
    print("\nDecrypted Text:")
    print(decrypted_text)

main()
```

References

[1]https://www.youtube.com/watch?v=UURjVI5cw4g&list=PLBlnK6fEyqRgJU3EsOYDTW7m6SUmW6kII .Accessed on 30-11-2023 at 11:50 AM.

[2] https://www.101computing.net/mono-alphabetic-substitution-cipher/ .

Accessed on 4-12-2023 at 11:50 AM.

[3] https://www.101computing.net/frequency-analysis/ .

Accessed on 4-12-2023 at 11:50 AM.

[4] https://crypto.interactive-maths.com/frequency-analysis-breaking-the-code.html .

Accessed on 4-12-2023 at 11:50 AM.