

## Q1 Commands

10 Points

List the commands used in the game to reach the ciphertext.

The list of commands used to clear this level are:

go

back

read

## Q2 Cryptosystem

10 Points

What cryptosystem was used in this level?

The Cryptosystem used in this level is Vigenere Cipher with the Key kcgcdfccb.

Vigenere Cipher is a polyalphabetic substitution cipher.

## Q3 Analysis

20 Points

What tools and observations were used to figure out the cryptosystem?

NOTE: Failing to provide proper analysis would result in zero marks for this assignment.

1. The first thing that came to our mind when we saw the ciphertext was that it could be either Caesar cipher or Substitution Cipher. Then we tried all possible shifts, we can have utmost 26 shifts we tried out all of

them. But unfortunately none of the shifts gave us any sensible text. We then tried techniques Index of Coincidence (IOC) as per it the IOC value came out to be 0.042361 so our probable length of key is 9. Similarly Kasiski examination gave us the key size to be 9 (Code is attached in answer 6).

2. While solving the assignment after passing the first round by typing the command we came across a figure which was made up of several lines and dashes (-/). In that it was written that we have to "bow down and then slowly look up and count the lines in the horizontal direction.

3. So then we counted the lines in horizontal direction and the sequence which we got is (1,2,2,5,3,2,6,2,10). But since it was written that we have to bow down and count in the upwards therefore the sequence becomes (10,2,6,2,3,5,2,2,1).

4. Since we got a key we thought this is a hint for using polyalphabetic substitution cipher therefore then we moved to Polyalphabetic substitution cipher. At first we thought that it was the Beaufort cipher then we applied the Beaufort cipher decryption. The Beaufort cipher is a simple polyalphabetic cipher. It uses a table called tabula recta. In Beaufort cipher we use the same steps as in encryption as well as in decryption.

5. The steps are as follows: Firstly we find the plaintext letter in the topmost horizontal row of the table. Secondly we travel down the column, until we find the current key letter. Then finally the leftmost letter in the current row is the new ciphertext/plaintext letter.

6. In beaufort cipher we follow the same steps in encryption as well as in decryption. But unfortunately the decrypted text didn't produced as any sensible text.

7. Then we tried to check if the given cipher is a playfair cipher or not. In Playfair cipher what we actually do is that we firstly generate a key square of 5X5 size. Each of the 25 alphabets must be unique and one letter of the alphabet (usually J) is omitted from the table. The initial alphabets in the key square are the unique alphabets of the key in the order in which they appear followed by the remaining letters of the alphabet in order.

8. After this the ciphertext is split into pairs of two letters (digraphs) and then we form rectangles thereby getting the the deciphered message. But when we applied playfair fair cipher the resultant message obtained didn't formed sensible therefore then we concluded that the cryptosystem used is not playfair cipher.

9. Now we finally checked for Vigenere Cipher. Vigenere Cipher is a Polyalphabetic substitution technique. In Vigenere Cipher the letters are mapped to numbers (For eg: a maps to 0, b to 1 and so on).

10. Therefore our Key is (10,2,6,2,3,5,2,2,1) which becomes "kcgcdfccb". Our key length is 9. We need to map only alphabets and intentionally skip everything which is not an alphabet (ex: space, fullstop, inverted comas, underscore, etc).

Therefore the mapping becomes:

letter -> letter ( Letter in CipherText is replaced by letter Given by Vigenere Cipher ignore case sensitivity )

" -> " (Quotation Mark is replaced with Quotation mark)

, -> , (Comma is replaced with comma)

space->space (Space is replaced with space)

. -> . (Full Stop is replaced with Full Stop)

11. Then we with our python code and the Vigenere cipher and then checked the result.

In Vigenere cipher what actually we do is we take a letter from the cipher text subtract the letter from the key and then do modulo 26. But then the result which we then obtain is negative so then we add 26 to the result to obtain or decrypt text. We actually break the cipher text in 9-9 character blocks since our key is of size 9 and then do the replacement.

The logic which we have used in form of mathematical equations are:

Encryption:-

$$E_i = (P_i + K_i) \% 26$$

Decryption:-

$$D_i = (E_i - K_i + 26) \% 26$$

P=>Plaintext

E=>Encrypted Text

K=> Key

When we applied Vigenere cipher technique decryption then we got meaningful text therefore we concluded that the cryptosystem which is used is Vigenere cryptosystem and the key which we obtained is plain text which we obtained after the decryption is:

Be wary of the next chamber, there is very little joy there. Speak out the password "the\_cave\_man\_be\_pleased" to go through. May you have the strength for the next chamber. To find the exit you first will need to utter magic words there.

And the password which we get is

"the\_cave\_man\_be\_pleased" without the quotes.

## Q4 Decryption Algorithm

15 Points

Briefly describe the decryption algorithm used. Also mention the plaintext you deciphered. (Use less than 350 words)

The Vigenere cipher is a technique for encrypting alphabetic text. It relies on a keyword's letter structure. Vigenere cipher is a special case of polyalphabetic substitution.

Using a Vigenere table or Vigenere square, the text is encrypted and decrypted. The Vigenere table is also known as the tabula recta.

EncryptedText: ' Lg ccud qh urg tgay ejbw dkt, wmg tf su bgud nkudnk lrd vjfbg.

Yrhfm qvd vng sfuuxytj "vkj\_ecwo\_ogp\_ej\_rnfkukf" wt iq urtuwjm.

Ocz iqa jdag vio uzthsivi pqx vkj pgyd encpggt. Uy hopg yjg fhkz

arz hkscv ckoq pgfn vu wwylgt nkioe zttft djkh.'

Algorithm used for the decryption process:

1. Key length identification: Firstly, we'll need to determine the length of the key used to encrypt the message. We tried Index of Coincidence (IOC) as per it the IOC value came out to be 0.042361 so our probable length of key is 9 (Code in Answer 6). Similarly Kasiski examination gave us the key size to be 9. But, fortunately in our case, the key (10,2,6,2,3,5,2,2,1) is given as the reverse count of the horizontal lines, and hence we know the key length, which is 9 characters long. Our Key is "kcgcdfccb"

2. We need to repeat the key until we match the length of our ciphertext. We replace only letter and keep

spaces, quotation marks, full stops and other punctuation marks same as they are in ciphertext. The key needs to be just sufficient to cover all alphabets. The size of the ciphertext, containing only characters is "185" and the key is "9" characters long. So, we need to repeat our key 21 times (ceiling( $185/9$ ))

3. We then create the Vigenere Square, which is a matrix of 26x26 characters, where each row represents a different shift of the alphabet.

4. Find the matching character in the plaintext: Find the character in the Vigenere Square that corresponds to each character in the ciphertext. To accomplish this, locate the row that the key character relates to, and then locate the column that the ciphertext character corresponds to. The equivalent plaintext character is found at the intersection of that row and column. We actually break the cipher text in 9-9 character blocks and then do the replacement.

5. We then repeat the process for every character in the ciphertext, to determining the corresponding plaintext character.

6. We have decrypted the plain text as: Be wary of the next chamber, there is very little joy there. Speak out the password "the\_cave\_man\_be\_pleased" to go through. May you have the strength for the next chamber. To find the exit you first will need to utter magic words there.

7. The Mathematical Logic used is given in answer 4.

Example:

Lg ccud qh urg tday: 11 12 2 2 20 3 16 7 20 17 6  
kc gcdf cc bkc gcdf: 10 2 6 2 3 5 2 2 1 10 2

-----

## APPLYING DECRYPTION

Be wary of the next : 1 4 22 0 17 24 14 5 19 7 4

## Q5 Password

10 Points

What was the final command used to clear this level?

the\_cave\_man\_be\_pleased

## Q6 Codes

0 Points

Upload any code that you have used to solve this level



Download

SecureSavants\_\_ModernCrypto\_Assignment2.ipynb

In [1]:

```
#Index Of Coincidence
alph =
"abcdefghijklmnopqrstuvwxyz"

def isLetter(char):
    return (char in alph)

def countLetters(text):
    count = 0
    for i in text:
        if(isLetter(i)):
            count +=
1
    return count

def getIOC(text):
    letterCounts = []

    # Loop through each
```

```
letter in the alphabet - count
number of times it appears
    for i in
range(len(alph)):
    count = 0
    for j in text:
        if j ==
alph[i]:

count += 1

letterCounts.append(count)

    # Loop through all
letter counts, applying the
calculation (the sigma part)
    total = 0
    for i in
range(len(letterCounts)):
        ni =
letterCounts[i]
        total += ni *
(ni - 1)

    N = countLetters(text)
    c = 26.0 # Number of
letters in the alphabet
    total = float(total) /
((N * (N - 1)))
    return total

t="Lg ccud qh urg tgay ejbwdkt,
wmgtf su bgud nkudnk lrd vjfbg.
Yrhfm qvd vng sfuuxytj
\"vkj_ecwo_ogp_ej_rnfkukf\" wt
iq urtuwjm.   Ocz iqa jdag vio
uzthsivi pqx vkj pygd encpggt.
Uy hopg yjg fhkz  arz hkscv ckoq
pgfn vu wwygt nkioe zttft
djkhth."
text = t.lower()
total = getIOC(text)
print("IOC: " + str(total))
print("Normalised IOC: " +
str(total * 26.0))
```



IOC: 0.04236192714453584  
Normalised IOC: 1.101410105757932

In [2]:

```
#Substitution Cipher

def
substitution_cipher(text,subs_val):
    decrypted_text = ""
    for char in text:
        if char.isalpha():
            # Check if the
character is lowercase
            if char.islower():
                decrypted_text +=
chr((ord(char) - ord('a') +
subs_val) % 26 + ord('a'))
            # Check if the
character is uppercase
            elif char.isupper():
                decrypted_text +=
chr((ord(char) - ord('A') +
subs_val) % 26 + ord('A'))
            else:
                decrypted_text += char
    return decrypted_text

for i in range(1,26):
    print("Substitution Value: ",i)
    Ciphertext = "Hello, World! This
is a test message."
    print("CipherText: ", Ciphertext)
    decrypted_text =
substitution_cipher(Ciphertext,i)
    print("Decrypted:
",decrypted_text)
```

Substitution Value: 1  
CipherText: Hello, World! This is a  
Decrypted: Ifmmp, Xpsme! Uijt jt b  
Substitution Value: 2

CipherText: Hello, World! This is a  
Decrypted: Jgnnq, Yqtnf! Vjku ku c  
Substitution Value: 3

CipherText: Hello, World! This is a  
Decrypted: Kloor, Zruog! Wklv lv d  
Substitution Value: 4

CipherText: Hello, World! This is a  
Decrypted: Lipps, Asvph! Xlmw mw e  
Substitution Value: 5

CipherText: Hello, World! This is a  
Decrypted: Mjqqt, Btwqi! Ymnx nx f  
Substitution Value: 6

CipherText: Hello, World! This is a  
Decrypted: Nkrru, Cuxrj! Znoy oy g  
Substitution Value: 7

CipherText: Hello, World! This is a  
Decrypted: Olssv, Dvysk! Aopz pz h  
Substitution Value: 8

CipherText: Hello, World! This is a  
Decrypted: Pmttw, Ewztl! Bpqa qa i  
Substitution Value: 9

CipherText: Hello, World! This is a  
Decrypted: Qnuux, Fxaum! Cqrb rb j  
Substitution Value: 10

CipherText: Hello, World! This is a  
Decrypted: Rovvy, Gybvn! Drsc sc k  
Substitution Value: 11

CipherText: Hello, World! This is a  
Decrypted: Spwwz, Hzcwo! Estd td l  
Substitution Value: 12

CipherText: Hello, World! This is a  
Decrypted: Tqxxa, Iadxp! Ftue ue m  
Substitution Value: 13

CipherText: Hello, World! This is a  
Decrypted: Uryyb, Jbeyq! Guvf vf n  
Substitution Value: 14

CipherText: Hello, World! This is a  
Decrypted: Vszzc, Kcfzr! Hvwg wg o  
Substitution Value: 15

CipherText: Hello, World! This is a  
Decrypted: Wtaad, Ldgas! Iwxh xh p  
Substitution Value: 16

CipherText: Hello, World! This is a  
Decrypted: Xubbe, Mehbt! Jxyi yi q  
Substitution Value: 17

```

CipherText:  Hello, World! This is a
Decrypted:    Yvccf, Nficu! Kyzj zj r
Substitution Value:  18
CipherText:  Hello, World! This is a
Decrypted:    Zwddg, Ogjdv! Lzak ak s
Substitution Value:  19
CipherText:  Hello, World! This is a
Decrypted:    Axeeh, Phkew! Mab1 bl t
Substitution Value:  20
CipherText:  Hello, World! This is a
Decrypted:    Byffi, Qilfx! Nbcm cm u
Substitution Value:  21
CipherText:  Hello, World! This is a
Decrypted:    Czggj, Rjmgj! Ocdn dn v
Substitution Value:  22
CipherText:  Hello, World! This is a
Decrypted:    Dahhk, Sknhz! Pdeo eo w
Substitution Value:  23
CipherText:  Hello, World! This is a
Decrypted:    Ebiil, Tloia! Qefp fp x
Substitution Value:  24
CipherText:  Hello, World! This is a
Decrypted:    Fcjjm, Umpjb! Rfgq gq y
Substitution Value:  25
CipherText:  Hello, World! This is a
Decrypted:    Gdkkn, Vnqkc! Sghr hr z

```

In [3]:

```

#Beaufort_Cipher
def beaufort_cipher(ciphertext,
key):
    """Decrypts ciphertext using
    the Beaufort Cipher with the
    given key."""
    decryptedtext = ""
    for i in
range(len(ciphertext)):
        # Convert the characters
to uppercase and subtract 'A' to
get a value between 0 and 25
        char_value =
(ord(ciphertext[i].upper()) -
ord('A')) % 26
        key_value = (ord(key[i %
len(key)].upper())-ord('A')) % 26

```

```

        # Apply the Beaufort
        Cipher algorithm to obtain the
        decrypted character
        decrypted_char =
        chr(((key_value - char_value) %
        26) + ord('A'))
        decryptedtext +=
        decrypted_char

        return decryptedtext

def main():
    ciphertext = "Lg ccud qh urg
    tgay ejbwkt, wmgf su bgud
    nkudnk lrd vjfbg. Yrhfm qvd vng
    sfuuxytj
    \"vkj_ecwo_ogp_ej_rnfkukf\" wt iq
    urtuwjm. Ocz iqa jdag vio
    uzthsivi pqx vkj pyd encpggt. Uy
    hpg yjg fhkz arz hkscv ckoq pgfn
    vu wwygt nkioe zttft djktth."
    key = "kcgcdfccb"
    decryptedtext =
    beaufort_cipher(ciphertext, key)
    print("Key:", key)

    print("\nCipherText:", ciphertext)
    print("\nDecrypted message:",
    decryptedtext)

if __name__ == '__main__':
    main()

```

Key: kcgcdfccb

CipherText: Lg ccud qh urg tgay ejbwkt

Decrypted message: ZWNABLZJLDJMLXMJWE

In [4]:

```

#PlayFair Cipher

import string
key="kcgcdfccb"

```

```
print("Key: ",key)
key=key.replace(" ", "")
key=key.upper()
def matrix(x,y,initial):
    return [[initial for i in
range(x)] for j in range(y)]

result=list()
for c in key:
    if c not in result:
        if c=='J':
            result.append('I')
        else:
            result.append(c)
flag=0
for i in range(65,91):
    if chr(i) not in result:
        if i==73 and chr(74) not
in result:
            result.append("I")
            flag=1
        elif flag==0 and i==73
or i==74:
            pass
        else:

result.append(chr(i))
k=0
my_matrix=matrix(5,5,0)
for i in range(0,5):
    for j in range(0,5):
        my_matrix[i]
[j]=result[k]
        k+=1

def locindex(c):
    loc=list()
    if c == 'J':
        c = 'I'
    for i, row in
enumerate(my_matrix):
        for j, char in
enumerate(row):
            if c == char:
                loc.append(i)
```

```

        loc.append(j)
        return loc
    return None

def decrypt():
    msg = "Lg ccud qh urg tgay
ejbwdkt, wmgtf su bgud nkudnk
lrd vjfbg. Yrhfm qvd vng
sfuuxytj
\"vkj_ecwo_ogp_ej_rnfkukf\" wt
iq urtuwjm. Ocz iqa jdag vio
uzthsivi pqx vkj pgyd encpggt.
Uy hopg yjg fhkz arz hkscv ckoq
pgfn vu wwygt nkioe zttft
djkh."

    print("\nCIPHER TEXT: ",msg)
    msg = msg.upper()
    msg = msg.replace(" ", "")
    print("\nPLAIN TEXT:", end='
')

    i = 0
    while i < len(msg):
        loc = locindex(msg[i])
        if i + 1 < len(msg):
            loc1 =
locindex(msg[i + 1])
        else:
            loc1 = None
        if loc is not None and
loc1 is not None:
            if loc[1] ==
loc1[1]:
                print("{}
{}".format(my_matrix[(loc[0] -
1) % 5][loc[1]],
my_matrix[(loc1[0] - 1) % 5]
[loc1[1]]), end=' ')
                elif loc[0] ==
loc1[0]:
                    print("{}
{}".format(my_matrix[loc[0]]
[(loc[1] - 1) % 5],
my_matrix[loc1[0]][(loc1[1] - 1)

```

PLAIN TEXT: NK WW TF TB TQ DS CE XH H

In [5]:

```

#Vignere Cipher

cipher_text="Lg ccud qh urg tgay ejbw
wmgft su bgud nkudnk lrd vjfbg. Yrhfn
sfuuxytj \"vkj_ecwo_ogp_ej_rnfkukf\"
urtuwjm. Ocz iqa jdag vio uzthsivi pc
pgyd encpggt. Uy hopg yjg fhkz arz hk
pgfn vu wwygt nkioe zttft djcth."
key="kcgcdfccb"

j=0
cipher_text_len=len(cipher_text)
key_len=len(key)
plain_text=""
for i in range(cipher_text_len):
    if(ord(cipher_text[i])>=ord('a')):
ord(cipher_text[i])<=ord('z')):
        ord_cipher=ord(cipher_text[i])
        ord_key=ord(key[j])
        plain_text_char_temp=(ord_cipher-ord_key+26)%26

plain_text_char=chr(ord('a')+plain_text_char_temp)

        j=(j+1)%key_len
    elif(ord(cipher_text[i])>=ord('A')):
ord(cipher_text[i])<=ord('Z')):
        ord_cipher=ord(cipher_text[i])
        ord_key=ord(key[j].upper())
        plain_text_char_temp=(ord_cipher-ord_key+26)%26

plain_text_char=chr(ord('A')+plain_text_char_temp)
        j=(j+1)%key_len
    else:
        plain_text_char=cipher_text[i]
        plain_text+=plain_text_char
print("Key: \n"+key)
print("\nCiphertext: \n"+cipher_text)
print("\nKey Length:
"+str(key_len)+"\nCiphertext Length:
"+str(cipher_text_len))

print("\nPlainText: \n"+plain_text)

```



Key:

kcgcdfccb

CipherText:

Lg ccud qh urg tgay ejbwdkt, wmgtf su

Key Length: 9

CipherText Length: 237

PlainText:

Be wary of the next chamber, there is


## Q7 Team Name

0 Points

SecureSavants

## Assignment 2

● Graded


 Select each question to review feedback and grading details.

### Group

Souvik Mukherjee

VISHAL KUMAR

Vikrant Chauhan

 [View or edit group](#)

**Total Points****65 / 65 pts****Question 1**[Commands](#)**10 / 10 pts****Question 2**[Cryptosystem](#)**10 / 10 pts****Question 3**[Analysis](#)**20 / 20 pts****Question 4**[Decryption Algorithm](#)**15 / 15 pts****Question 5**[Password](#)**10 / 10 pts****Question 6**[Codes](#)**0 / 0 pts****Question 7**[Team Name](#)**0 / 0 pts**