Electronics and Computer Science
Faculty of Physical Sciences and Engineering
University of Southampton

Tan Wei Shawn

30499844

wst1g18@soton.ac.uk

Cryptography coursework

Module Leader: Dr Basel Halak

Contents

1 Cipher 1

1.1 Solution for cipher 1

Formative assessment can be viewed as a mean to enhance the learning process. Based on the results of such assessments, students will be able to assess their knowledge and identify strengths and weaknesses. The teacher will also have indication on how well the students are grasping the fundamental facts and whether he needs to alter their teaching to emphasis some important concepts.

1.2 Cipher 1 Cryptanalysis

Cipher text 1 was first processed using Python code to remove all the punctuations, spaces and also to convert all upper characters to lower characters. Next, frequency analysis was done on the processed cipher text and a few observations were made. The total number of letters were counted to be 321 and the letter 'y' has the most count of 50, followed by 'I', 37 and 'h', 32. The values were included in table 1 and these values were used to plot the normalized frequency analysis bar chart as shown in figure 1 for better data visualization.

Next, the cipher algorithm was first determined using Index of Coincidence (IC). IC was defined by the probability that two randomly selected letters from a text will be identical. In others words, the global distribution of the letters in a cipher text can be evaluated. A typical text message written in English has an IC of 0.0667 and it can be calculated based on equation 1 below.

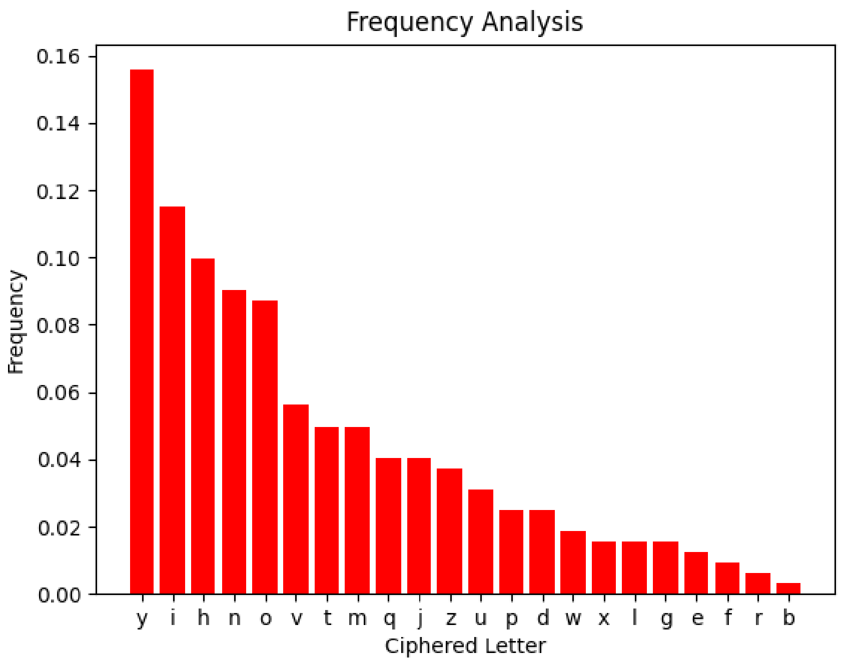


Figure 1: Normalized frequency analysis for the letters in cipher text 1.

Table 1: Frequency of the letter used to generate the bar chart in figure 1.

Ciphered Letters	Count	Ciphered Letters	Count
y	50	D	8
i	37	W	6
h	32	X	5

n	29	L	5
o	28	G	5
v	18	E	4
m	16	F	3
t	16	R	2
j	13	B	1
q	13		
z	12		
u	10		
p	8	Total character = 321	

$$Index\ of\ Coincidence = \frac{\sum_{i=a}^{z} f_i(f_i-1)}{N(N-1)} \qquad (1)$$

f_i denotes the number of occurrences of letter 'i' in the text and N is the total number of letters. The IC was calculated to be 0.07675 for cipher text 1. This suggests that monoalphabetic substitution or transposition cipher were the two possible algorithms for this question. Vigenere cipher and polyalphabetic cipher were both not possible as the IC returned would be close to 0.0385 (IC of a random text) as the letters can be replaced by multiple alphabets.

Hence, the cipher text was analysed again based on the assumption that monoalphabetic substitution was the algorithm used. The steps used to decipher the text are shown below.

- Assume cipher text 'y' is plain text 'e' as it has the highest frequency.
- Single cipher text character 'n' can only be two possible letters given by 'A' and 'I'.
- Repeated words with an extra character at the back suggests the plural form of the word ('niiyiipyoh' and 'niiyiipyohi').
- Pairs of letters are unusual and assuming the 'y' in CT 'nii**y**iip**y**oh' is 'e', the possible letters can only be 'ASSESSpEoh'. This was guessed to be 'assessment'
- The word 'the' is common and will be used at the beginning of each sentence. The CT 'hvy' occurs 3 times and in one of the occurrences it is at the start of the sentence.
- Linking word 'and' can be inferred to be 'noz' in the CT.
- From the info collected thus far, 'ihjyowhvi noz dynroyiiyi' can be substituted to be 'STjENwTHS AND dEArNESSES' which can be inferred to be 'strengths and weaknesses'
- 'hvy xgoznpyohnq xnuhi' -> 'THE xgNzAMENTAq xACTS'. Hence, 'x' should be 'F' and the deciphered text is 'THE FUNDAMENTAL FACTS'.
- 'Xmjpnhtfy' -> 'FmRMATIfE'. Therefore, the deciphered text is 'FORMATIVE'

81% of the letters have been deciphered (18 out of 22) based on the analysis steps above and this is enough to decipher the rest of the ciphered text. The mapping of each letter was given as shown in the table below.

Table 2: Mapping of cipher text character to plain text character.

Cipher Text	Plain Text	Cipher Text	Plain Text
N	A	O	N
E	B	M	O
U	C	L	P
Z	D	-	Q
Y	E	J	R
X	F	I	S

W	G	H	T
V	H	G	U
T	I	F	V
-	J	D	W
R	K	-	X
Q	L	B	Y
P	M	-	Z

2 Cipher 2

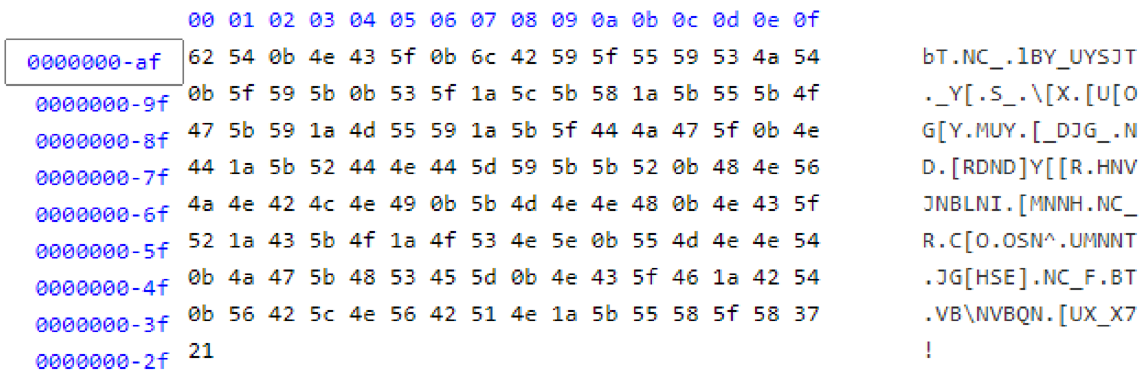
2.1 Solution for cipher 2

Plaintext: In the Victorian era it was popular for people to photograph relatives after they had died often placing them in lifelike poses

Key: ‘0x2B, 0x3A’

2.2 Cipher 2 Cryptanalysis

The hex file was first analysed by using online hex file reader and the results were in the figure below. The CT length was observed to be 129 and the hint given was ‘I’ for cipher text ‘b’ (first letter). Since the cipher was based on hexadecimal, all the bitwise operator such as AND, NAND, OR, NOR, XOR, XNOR, left shift, right shift and bit flip have to be considered. A quick analysis was made using the hint given to obtain the first subkey used to decipher the text. The binary of PT ‘I’ is ‘01001001’ and CT ‘b’ was ‘01100010’. From this, it was obvious that left shit, right shift, bit flip will not be the possible type of decipher algorithm. After computing, AND, NAND, OR and NOR was not possible to decrypt the hex file too. Hence, XOR was the only possible decryption method in this question.



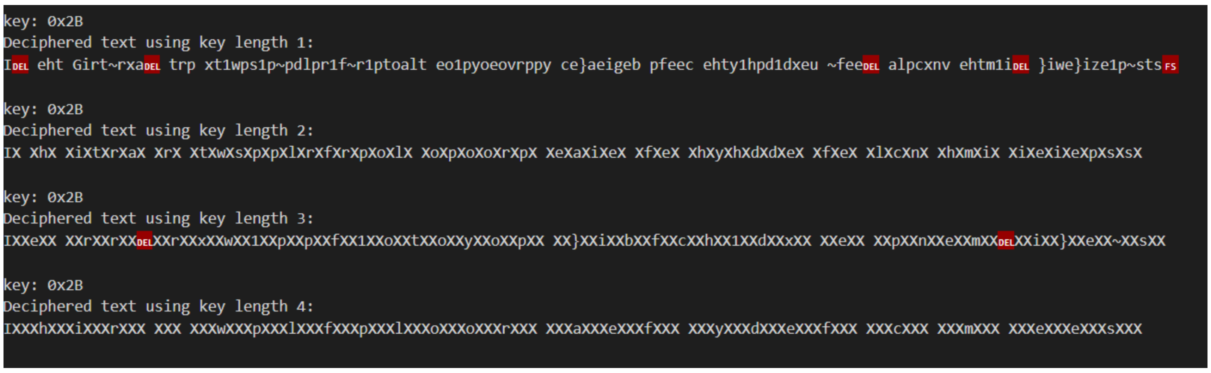
```
              00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
  0000000-af  62 54 0b 4e 43 5f 0b 6c 42 59 5f 55 59 53 4a 54      bT.NC_.lBY_UYSJT
  0000000-9f  0b 5f 59 5b 0b 53 5f 1a 5c 5b 58 1a 5b 55 5b 4f      ._Y[.S_.\[X.[U[O
  0000000-8f  47 5b 59 1a 4d 55 59 1a 5b 5f 44 4a 47 5f 0b 4e      G[Y.MUY.[_DJG_.N
  0000000-7f  44 1a 5b 52 44 4e 44 5d 59 5b 5b 52 0b 48 4e 56      D.[RDND]Y[[R.HNV
  0000000-6f  4a 4e 42 4c 4e 49 0b 5b 4d 4e 4e 48 0b 4e 43 5f      JNBLNI.[MNNH.NC_
  0000000-5f  52 1a 43 5b 4f 1a 4f 53 4e 5e 0b 55 4d 4e 4e 54      R.C[O.OSN^.UMNNT
  0000000-4f  0b 4a 47 5b 48 53 45 5d 0b 4e 43 5f 46 1a 42 54      .JG[HSE].NC_F.BT
  0000000-3f  0b 56 42 5c 4e 56 42 51 4e 1a 5b 55 58 5f 58 37      .VB\NVBQN.[UX_X7
  0000000-2f  21                                                    !
```

Figure 2: Ciphered Hex file text read using online hex file editor [1].

Table 3: Subkey1 answer to cipher with CT ‘b’ to get PT ‘I’

CT(Hexadecimal)	CT (symbol)	Subkey 1 (Hexadecimal)	Subkey 1 (Symbol)	XOR (Hexadecimal)	XOR (Hexadecimal)
0x62	b	0x2B	+	0x49	I

Next, before choosing the cipher algorithm, the key length was deduced to be 3 as it was a divisor of 129. Hence, the cipher text at the position of i+3n where i is the initial position of CT ‘b’, n is the cycle were deciphered using the XOR operator. For positions other than 1+3n (n is the number of cycle), a capital X was used to represent the unknown plaintext due to the unavailability of key at subkey 2 and 3. However, when a key length of 3 was used, symbols such as special character, control character and unconventional punctuations were observed as shown in figure 3. Hence, deduction was made that the key length might not be 3. Besides, it can also be deduced that the key length must be even numbered length as the deciphered answer in key length 2 and 4 makes more sense.

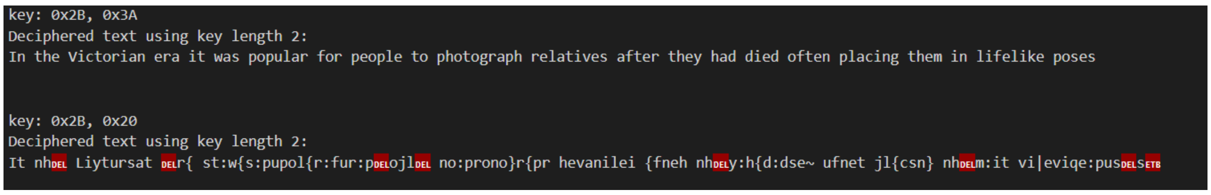


```
key: 0x2B
Deciphered text using key length 1:
IDEL eht Girt~rxaDEL trp xt1wps1p~pdlpr1f~r1ptoalt eo1pyoeovrppy ce}aeigeb pfeec ehty1hpd1dxeu ~feeDEL alpcxnv ehtm1iDEL }iwe}ize1p~stsFS

key: 0x2B
Deciphered text using key length 2:
IX XhX XiXtXrXaX XrX XtXwXsXpXpXlXrXfXrXpXoXlX XoXpXoXoXrXpX XeXaXiXeX XfXeX XhXyXhXdXdXeX XfXeX XlXcXnX XhXmXiX XiXeXiXeXpXsXsX

key: 0x2B
Deciphered text using key length 3:
IXXeXX XXrXXrXXDELXXrXXxXXwXX1XXpXXpXXfXX1XXoXXtXXoXXyXXoXXpXX XX}XXiXXbXXfXXcXXhXX1XXdXXxXX XXeXX XXpXXnXXeXXmXXDELXXiXX}XXeXX~XXsXX

key: 0x2B
Deciphered text using key length 4:
IXXXhXXXiXXXrXXX XXX XXXwXXXpXXXlXXXfXXXpXXXlXXXoXXXoXXXrXXX XXXaXXXeXXXfXXX XXXyXXXdXXXeXXXfXXX XXXcXXX XXXmXXX XXXeXXXeXXXsXXX
```

Figure 3: Deciphered text using key length of 1-4 using XOR algorithm. Capital X was used to replace all the unknown decipher text as only subkey 1 was known.

From figure 3, when key length of 1 and 3 was used, it can be observed that unknown symbols which were not English text were obtained and thus, it can be deduced that only key length of 2 and 4 were possible.

Next, when the key length was 4, the first word is 16 characters long and start off with an 'I'. According to the English dictionary, such combination was not possible and on the other hand, when a key length of 2 was used, the first two letters word given by 'IX' is either 'In' or 'It'. Therefore, this narrows down to XOR with key length of 2 and the possible keys were '0x2B, 0x3A' for 'In' and '0x2B, 0x20' for 'It'.



```
key: 0x2B, 0x3A
Deciphered text using key length 2:
In the Victorian era it was popular for people to photograph relatives after they had died often placing them in lifelike poses


key: 0x2B, 0x20
Deciphered text using key length 2:
It nhDEL Liytursat DELr{ st:w{s:pupol{r:fur:pDELojlDEL no:prono}r{pr hevanilei {fneh nhDELy:h{d:dse~ ufnet jl{csn} nhDELm:it vi|eviqe:pusDELSETB
```

Figure 4: Deciphered text when using XOR and key length of two with different keys.

The correct cipher text was obtained as shown in figure 4 above.

3 Cipher 3

3.1 Solution for cipher 3

Plaintext: Da life spent making mistakes is not only more honourable but more useful than a life spent doin nothing

Key: bfgcdahe

3.2 Cipher 3 Cryptanalysis

The length of cipher text 3 was counted to be 88 and the index of coincidence was calculated to be 0.0588. As the value is close to the normal English text IC, assumptions that the algorithm used is of the type substitution. Polyalphabetic substitution was excluded as the typical IC score would be much lower than this. Hence, transposition cipher is the only sensible algorithm in this case. Besides, to further confirm that substitution cipher was used, frequency analysis was carried out on to the processed cipher text (remove upper letter character) and the bar chart in figure xx below was obtained.

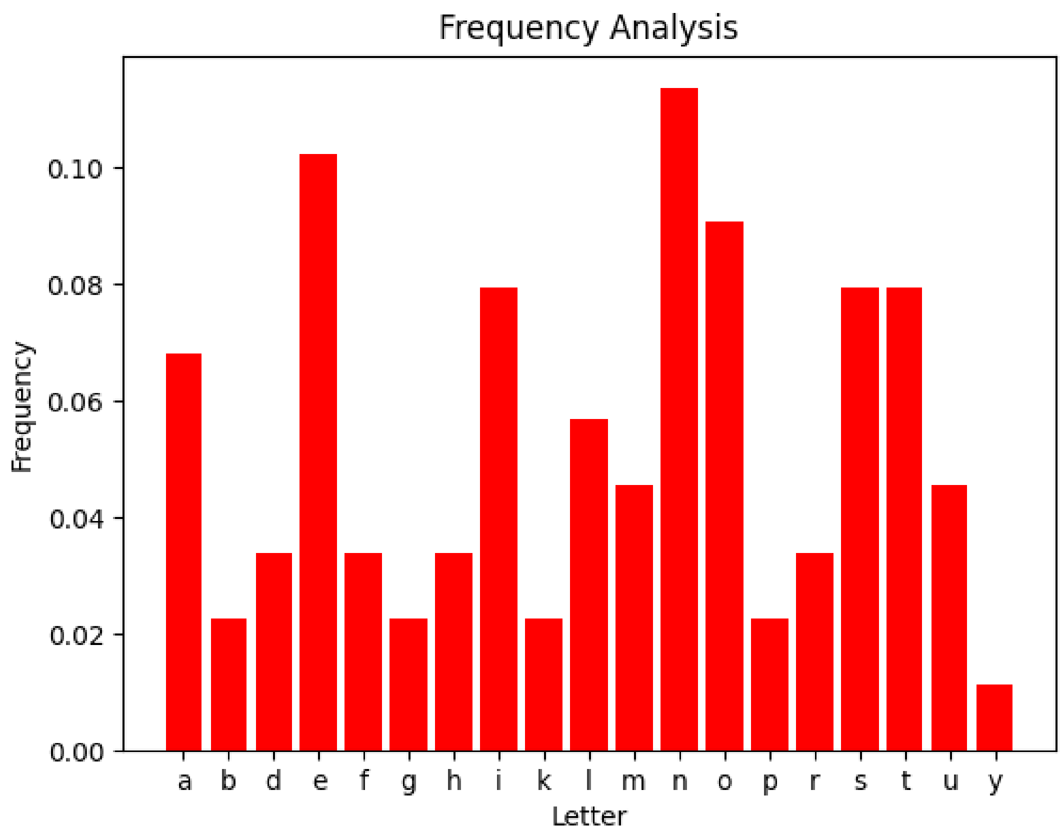


Figure 5: Frequency analysis for the letters in cipher text 3.

Next, the key length was determined by using the divisor of 88 and they were 2, 4, 8, 11, 22 and 44. The divisor of 22 and 44 was disregarded as they were too long to form a key word. Cipher text 3 was then written vertically in to a column of 2, 4, 8 and 11. Anagrams were observed clearly when the key length of 8 were used. For further iteration when a key length of 2, 4, 11 were used, the Excel notes were included in Appendix C. Table 4 shows the unarranged columns with a key length of 8.

Table 4: Key length of 8 without rearranging the columns.

Column Index	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
K	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**h**
plaintext	e	D	i	f	p	a	l	s
	k	e	m	a	n	n	t	i

	a	g	s	t	e	m	i	k
	t	s	n	o	n	i	s	o
	e	l	o	r	o	y	m	h
	b	n	r	a	e	o	u	l
	r	b	m	o	u	u	t	e
	t	s	u	l	a	e	f	h
	e	n	i	f	p	a	l	s
	i	e	d	o	n	n	t	n
	g	o	i	n	d	t	h	s

To rearrange the columns of the cipher text, there will be a possibility of 8P8 which translates to 40320 possibilities. Hence, to reduce this, a few smart guesses and observations have to be made. The letters in the last row might not all be used in the plaintext and hence, deduction will not be made based on this row. From the first row in the plaintext, a capital D was observed and this indicates that it should be at the start of the sentence or some sort of naming was involved. Table 5 gives a visual representation of this.

Table 5: Capital D in column 2 moved to column 1.

Column Index	**2**	**1**	**3**	**4**	**5**	**6**	**7**	**8**
K	**b**	**a**	**c**	**d**	**e**	**f**	**g**	**h**
plaintext	D	e	i	f	p	a	l	s
	e	k	m	a	n	n	t	i
	g	**a**	**s**	**t**	**e**	**m**	**i**	**k**
	s	t	n	o	n	i	s	o
	l	e	o	r	o	y	m	h
	n	b	r	a	e	o	u	l
	b	r	m	o	u	u	t	e
	s	t	u	l	a	e	f	h
	n	e	i	f	p	a	l	s
	e	i	d	o	n	n	t	n
	o	g	i	n	d	t	h	s

Another observation made here was in row 4 where double letter, N, S, O was observed. Assumptions were made that the two 'O' will not be next to each other and stemming from this, it was obvious that the word 'mistake' can be formed in row 3. By this, the cipher text was broken.

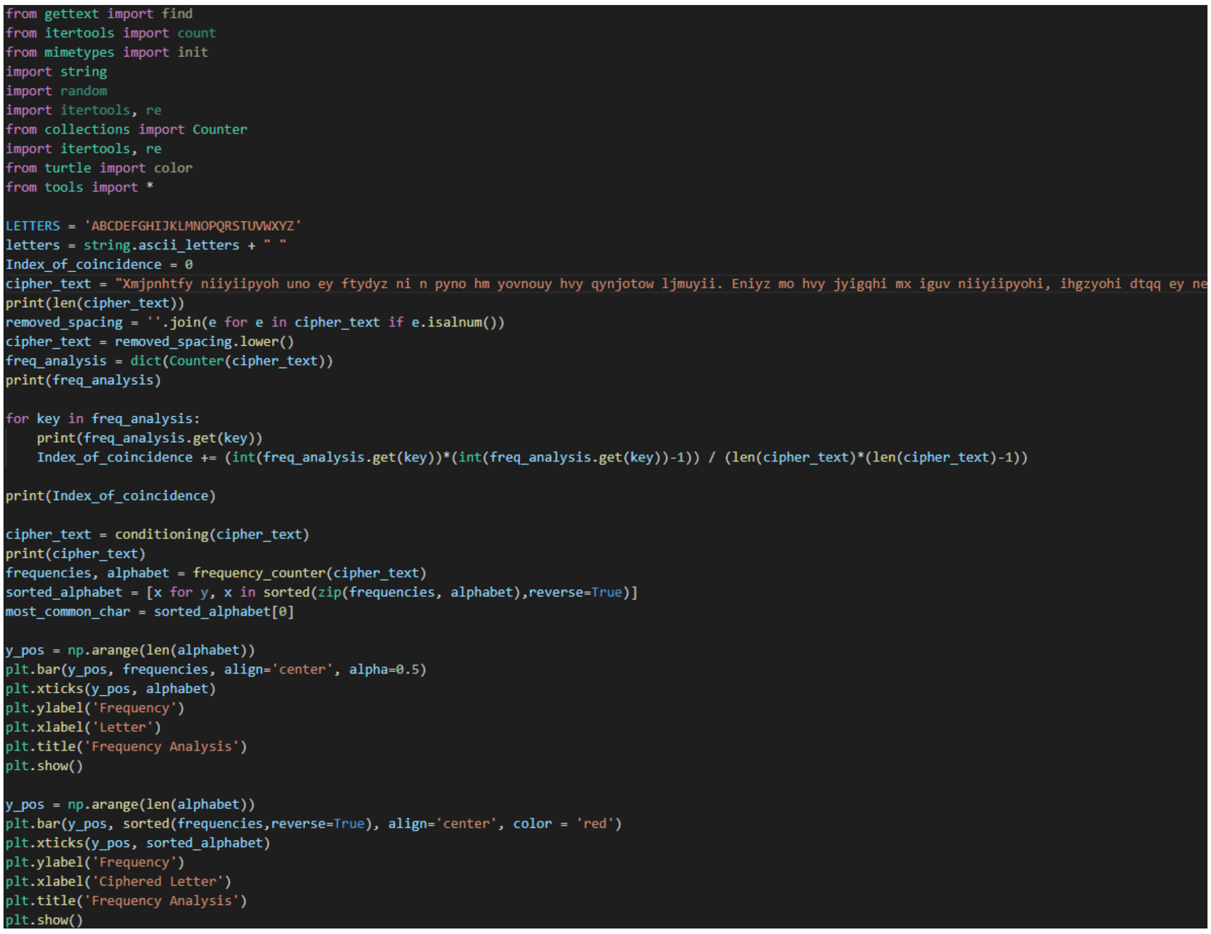
Table 6: Final answer with a key of 'bfgcdahe' and the greyed-out boxes represent filler text.

Column Index	**2**	**6**	**7**	**3**	**4**	**1**	**8**	**5**
K	**b**	**f**	**g**	**c**	**d**	**a**	**h**	**e**
plaintext	D	a	l	i	f	e	s	p
	e	n	t	m	a	k	i	n
	g	m	i	s	t	a	k	e
	s	i	s	n	o	t	o	n
	l	y	m	o	r	e	h	o
	n	o	u	r	a	b	l	e
	b	u	t	m	o	r	e	u
	s	e	f	u	l	t	h	a
	n	a	l	i	f	e	s	p

	e	n	t	d	o	i	n	n
	o	t	h	i	n	g	s	n

An observation made here was in column 5 of the second last row where the word ‘doin’ should be ‘doing’ and a ‘g’ is missing from it. Lastly, the boxes shaded in grey are fillers as the original text was 85 but a key of length 8 was used. This ensures that the key length is a divisor of the plain text length. This also further confirms the hypothesis made earlier where not all letters were used in the last column.

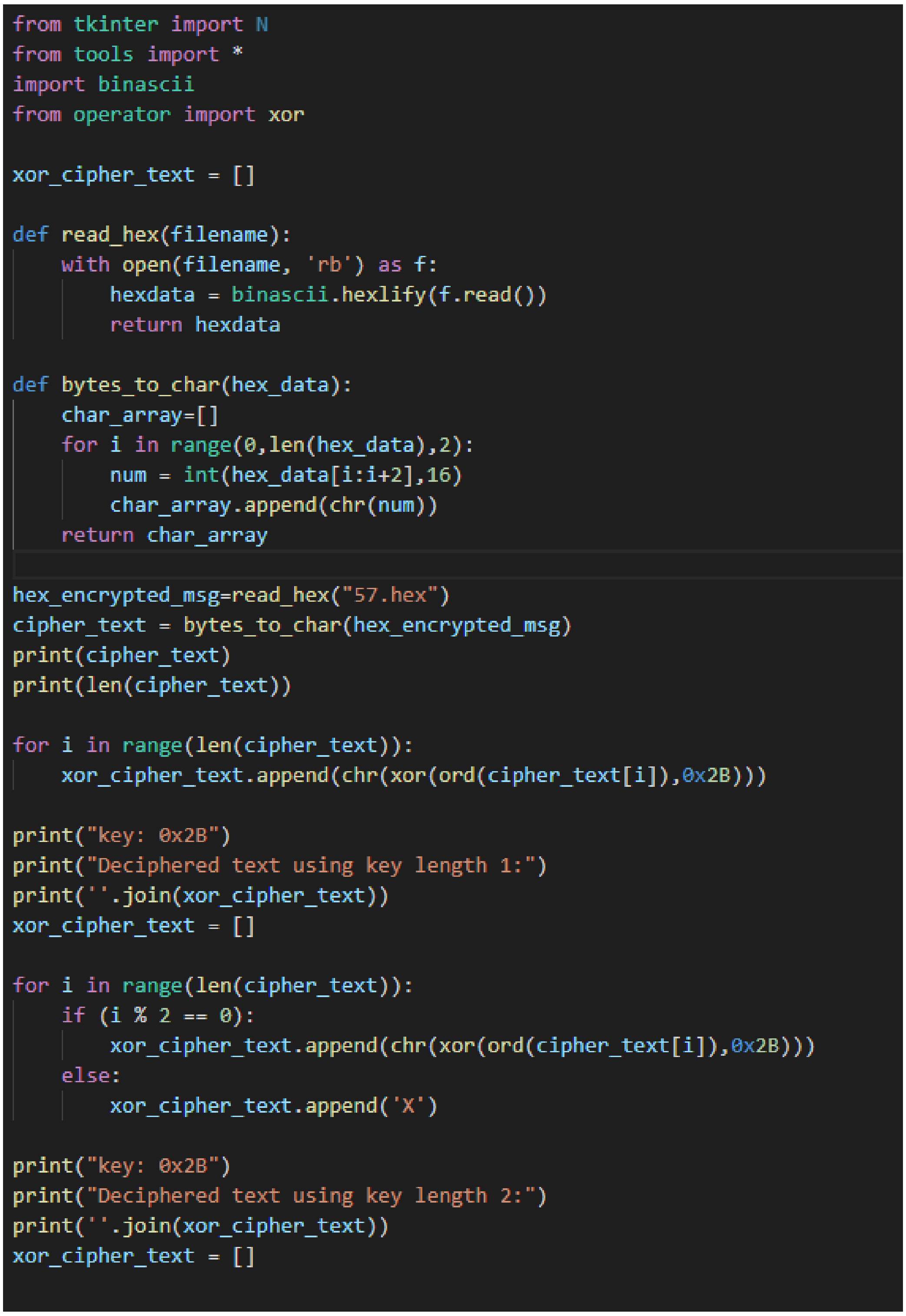
4 Appendix A



```python
from gettext import find
from itertools import count
from mimetypes import init
import string
import random
import itertools, re
from collections import Counter
import itertools, re
from turtle import color
from tools import *

LETTERS = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
letters = string.ascii_letters + " "
Index_of_coincidence = 0
cipher_text = "Xmjpnhtfy niiyiipyoh uno ey ftydyz ni n pyno hm yovnouy hvy qynjotow ljmuyii. Eniyz mo hvy jyigqhi mx iguv niiyiipyohi, ihgzyohi dtqq ey ne
print(len(cipher_text))
removed_spacing = ''.join(e for e in cipher_text if e.isalnum())
cipher_text = removed_spacing.lower()
freq_analysis = dict(Counter(cipher_text))
print(freq_analysis)

for key in freq_analysis:
    print(freq_analysis.get(key))
    Index_of_coincidence += (int(freq_analysis.get(key))*(int(freq_analysis.get(key))-1)) / (len(cipher_text)*(len(cipher_text)-1))

print(Index_of_coincidence)

cipher_text = conditioning(cipher_text)
print(cipher_text)
frequencies, alphabet = frequency_counter(cipher_text)
sorted_alphabet = [x for y, x in sorted(zip(frequencies, alphabet),reverse=True)]
most_common_char = sorted_alphabet[0]

y_pos = np.arange(len(alphabet))
plt.bar(y_pos, frequencies, align='center', alpha=0.5)
plt.xticks(y_pos, alphabet)
plt.ylabel('Frequency')
plt.xlabel('Letter')
plt.title('Frequency Analysis')
plt.show()

y_pos = np.arange(len(alphabet))
plt.bar(y_pos, sorted(frequencies,reverse=True), align='center', color = 'red')
plt.xticks(y_pos, sorted_alphabet)
plt.ylabel('Frequency')
plt.xlabel('Ciphered Letter')
plt.title('Frequency Analysis')
plt.show()
```

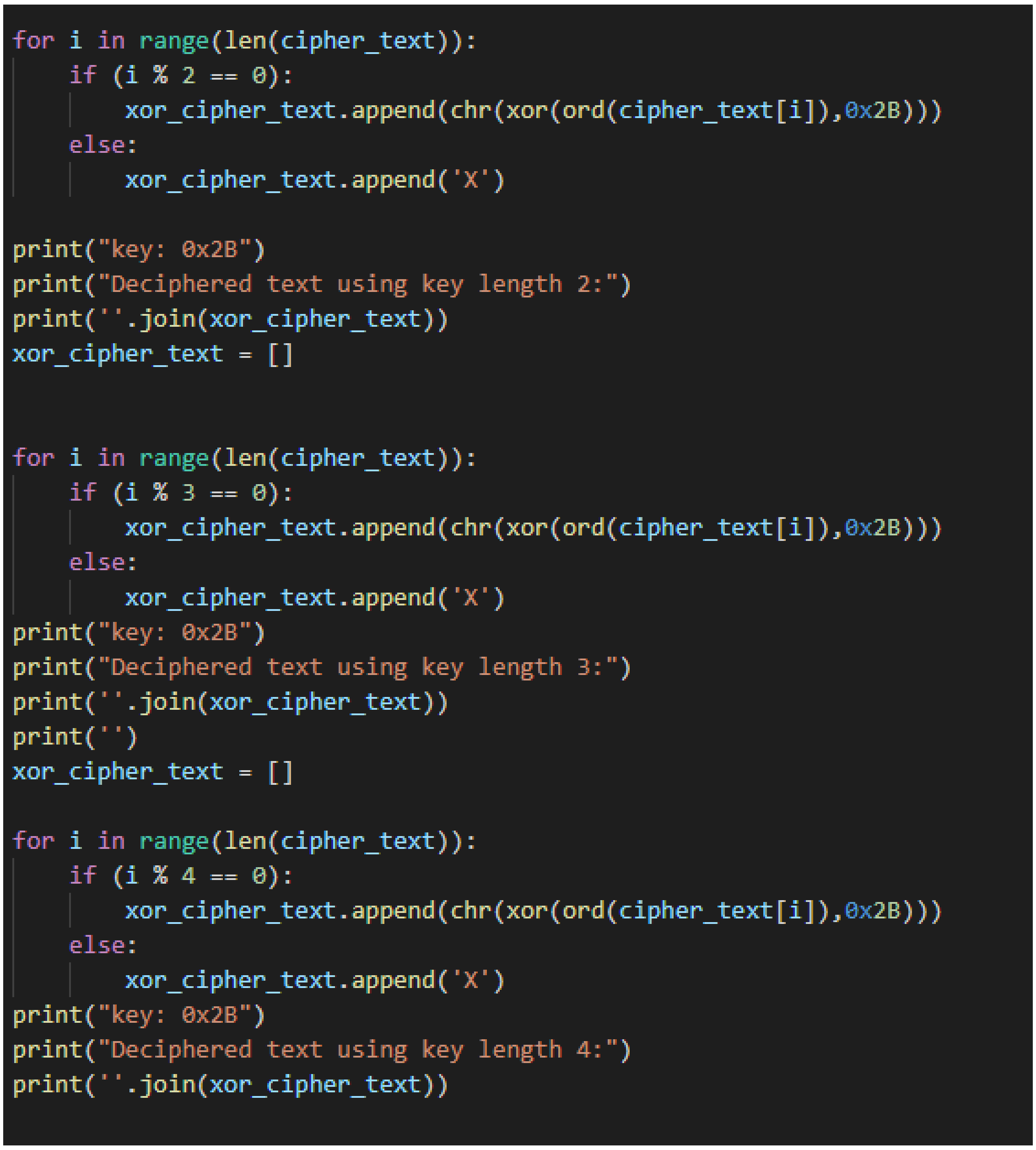
Figure 6: Function to do frequency analysis and plot the bar chart.

5 Appendix B



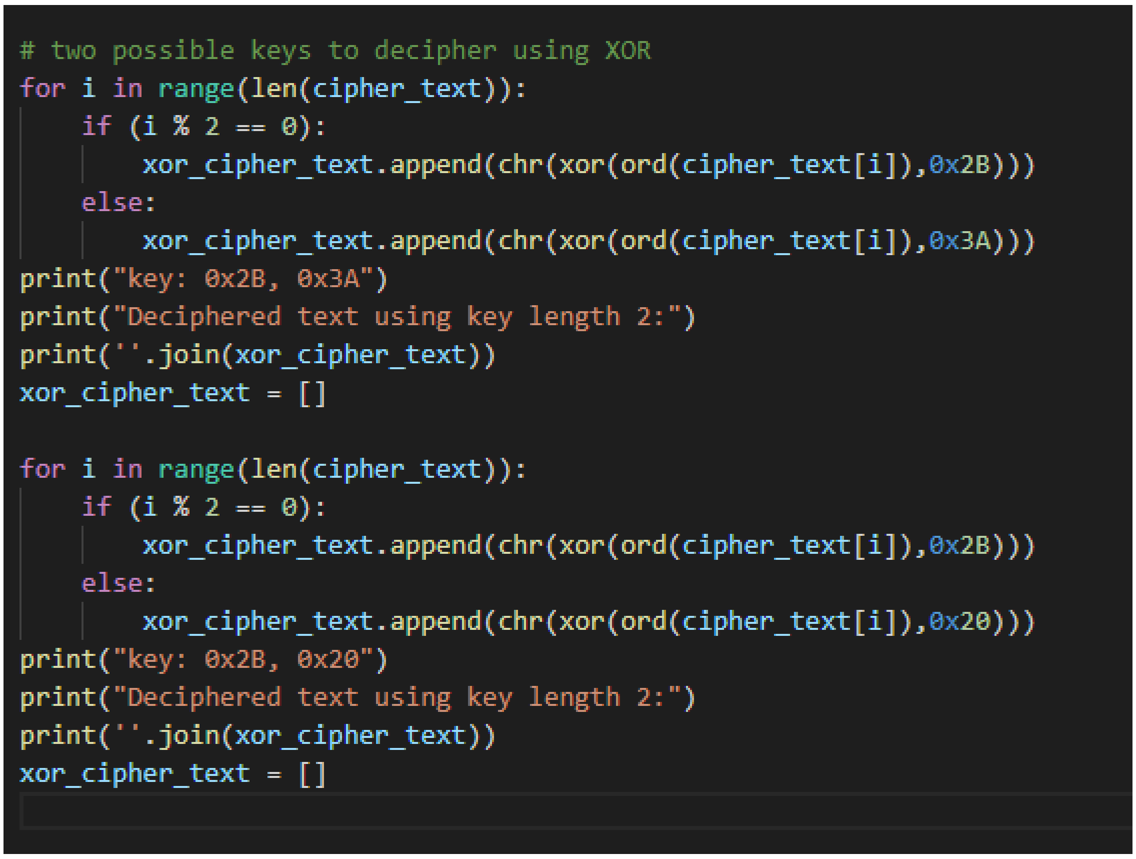
```python
from tkinter import N
from tools import *
import binascii
from operator import xor

xor_cipher_text = []

def read_hex(filename):
    with open(filename, 'rb') as f:
        hexdata = binascii.hexlify(f.read())
        return hexdata

def bytes_to_char(hex_data):
    char_array=[]
    for i in range(0,len(hex_data),2):
        num = int(hex_data[i:i+2],16)
        char_array.append(chr(num))
    return char_array

hex_encrypted_msg=read_hex("57.hex")
cipher_text = bytes_to_char(hex_encrypted_msg)
print(cipher_text)
print(len(cipher_text))

for i in range(len(cipher_text)):
    xor_cipher_text.append(chr(xor(ord(cipher_text[i]),0x2B)))

print("key: 0x2B")
print("Deciphered text using key length 1:")
print(''.join(xor_cipher_text))
xor_cipher_text = []

for i in range(len(cipher_text)):
    if (i % 2 == 0):
        xor_cipher_text.append(chr(xor(ord(cipher_text[i]),0x2B)))
    else:
        xor_cipher_text.append('X')

print("key: 0x2B")
print("Deciphered text using key length 2:")
print(''.join(xor_cipher_text))
xor_cipher_text = []
```

Figure 7a: Function to illustrate deciphered text when key length of 1 and 2 was used with only subkey 1 known.



```python
for i in range(len(cipher_text)):
    if (i % 2 == 0):
        xor_cipher_text.append(chr(xor(ord(cipher_text[i]),0x2B)))
    else:
        xor_cipher_text.append('X')

print("key: 0x2B")
print("Deciphered text using key length 2:")
print(''.join(xor_cipher_text))
xor_cipher_text = []


for i in range(len(cipher_text)):
    if (i % 3 == 0):
        xor_cipher_text.append(chr(xor(ord(cipher_text[i]),0x2B)))
    else:
        xor_cipher_text.append('X')
print("key: 0x2B")
print("Deciphered text using key length 3:")
print(''.join(xor_cipher_text))
print('')
xor_cipher_text = []

for i in range(len(cipher_text)):
    if (i % 4 == 0):
        xor_cipher_text.append(chr(xor(ord(cipher_text[i]),0x2B)))
    else:
        xor_cipher_text.append('X')
print("key: 0x2B")
print("Deciphered text using key length 4:")
print(''.join(xor_cipher_text))
```

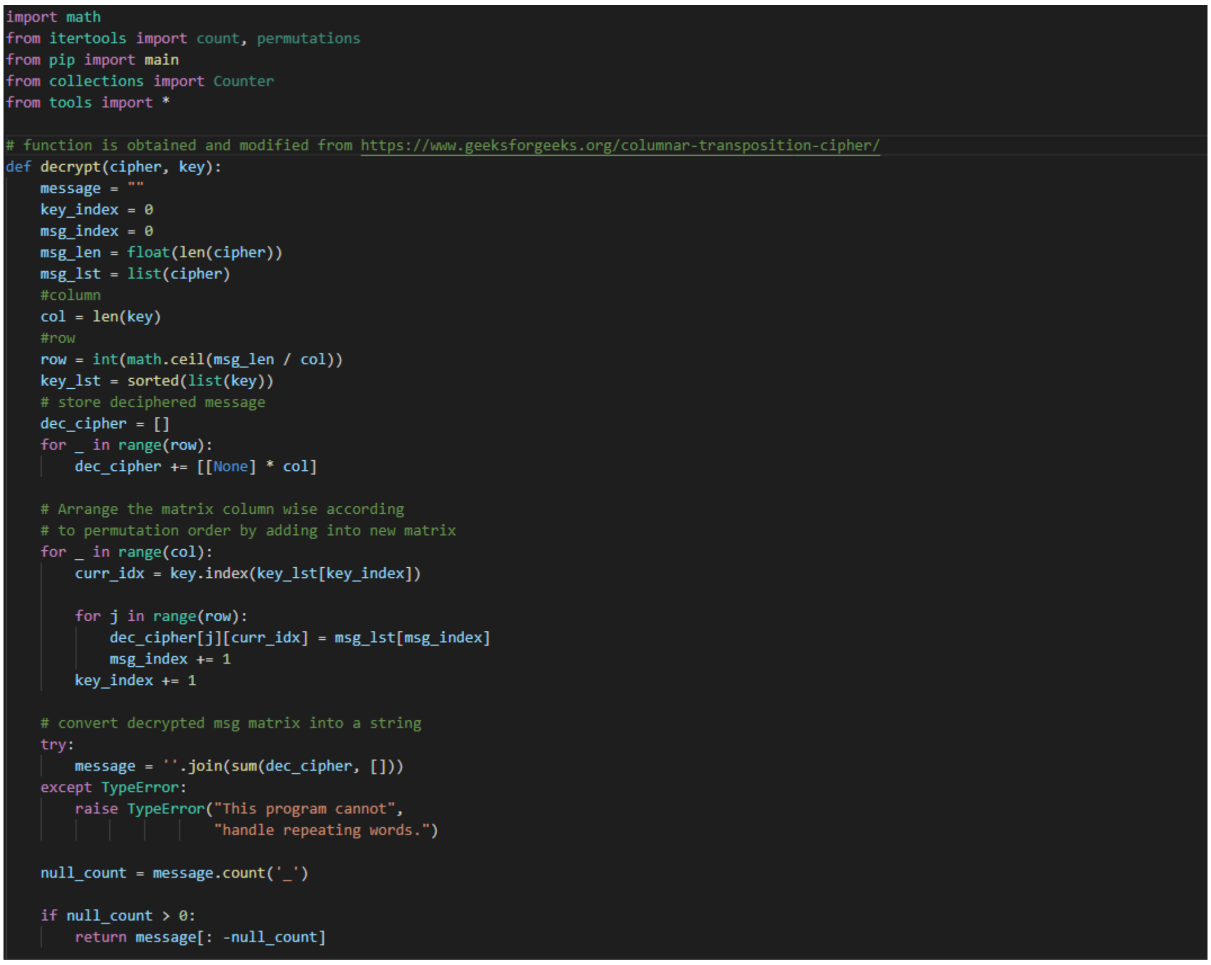
Figure 7b: Function to illustrate deciphered text when key length of 3 and 4 was used with only subkey 1 known.



```python
# two possible keys to decipher using XOR
for i in range(len(cipher_text)):
    if (i % 2 == 0):
        xor_cipher_text.append(chr(xor(ord(cipher_text[i]),0x2B)))
    else:
        xor_cipher_text.append(chr(xor(ord(cipher_text[i]),0x3A)))
print("key: 0x2B, 0x3A")
print("Deciphered text using key length 2:")
print(''.join(xor_cipher_text))
xor_cipher_text = []

for i in range(len(cipher_text)):
    if (i % 2 == 0):
        xor_cipher_text.append(chr(xor(ord(cipher_text[i]),0x2B)))
    else:
        xor_cipher_text.append(chr(xor(ord(cipher_text[i]),0x20)))
print("key: 0x2B, 0x20")
print("Deciphered text using key length 2:")
print(''.join(xor_cipher_text))
xor_cipher_text = []
```

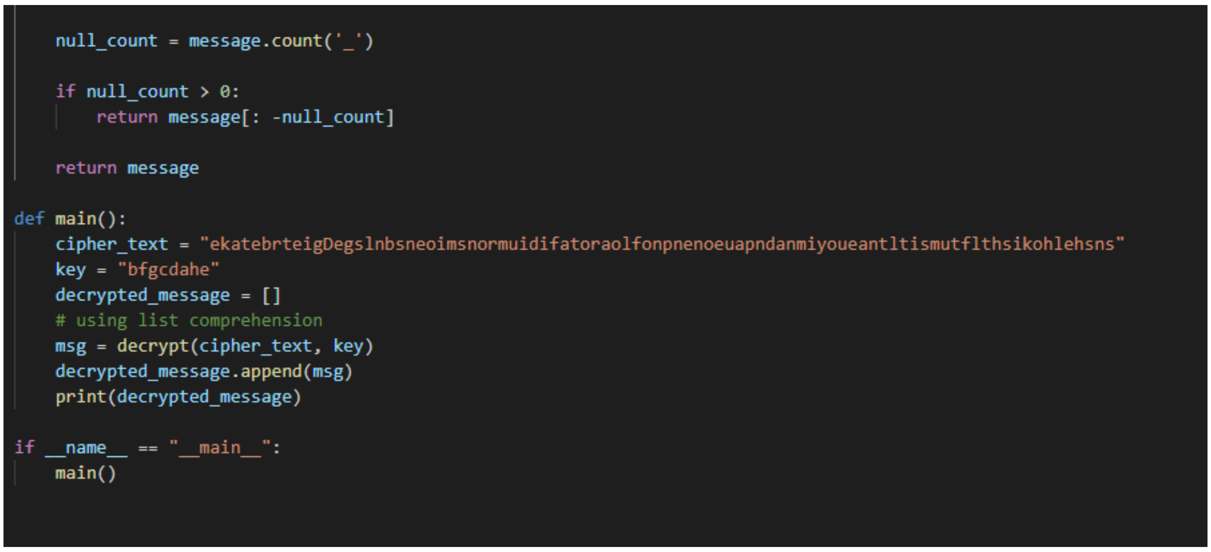
Figure 8: Final two possible key to obtain the correct answer.

6 Appendix C



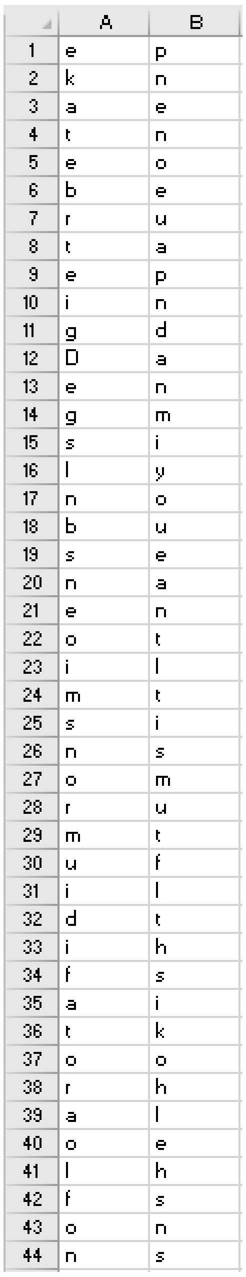
```python
import math
from itertools import count, permutations
from pip import main
from collections import Counter
from tools import *

# function is obtained and modified from https://www.geeksforgeeks.org/columnar-transposition-cipher/
def decrypt(cipher, key):
    message = ""
    key_index = 0
    msg_index = 0
    msg_len = float(len(cipher))
    msg_lst = list(cipher)
    #column
    col = len(key)
    #row
    row = int(math.ceil(msg_len / col))
    key_lst = sorted(list(key))
    # store deciphered message
    dec_cipher = []
    for _ in range(row):
        dec_cipher += [[None] * col]

    # Arrange the matrix column wise according
    # to permutation order by adding into new matrix
    for _ in range(col):
        curr_idx = key.index(key_lst[key_index])

        for j in range(row):
            dec_cipher[j][curr_idx] = msg_lst[msg_index]
            msg_index += 1
        key_index += 1

    # convert decrypted msg matrix into a string
    try:
        message = ''.join(sum(dec_cipher, []))
    except TypeError:
        raise TypeError("This program cannot",
                        "handle repeating words.")

    null_count = message.count('_')

    if null_count > 0:
        return message[: -null_count]
```

Figure 9: decryption algorithm for columnar transposition method.



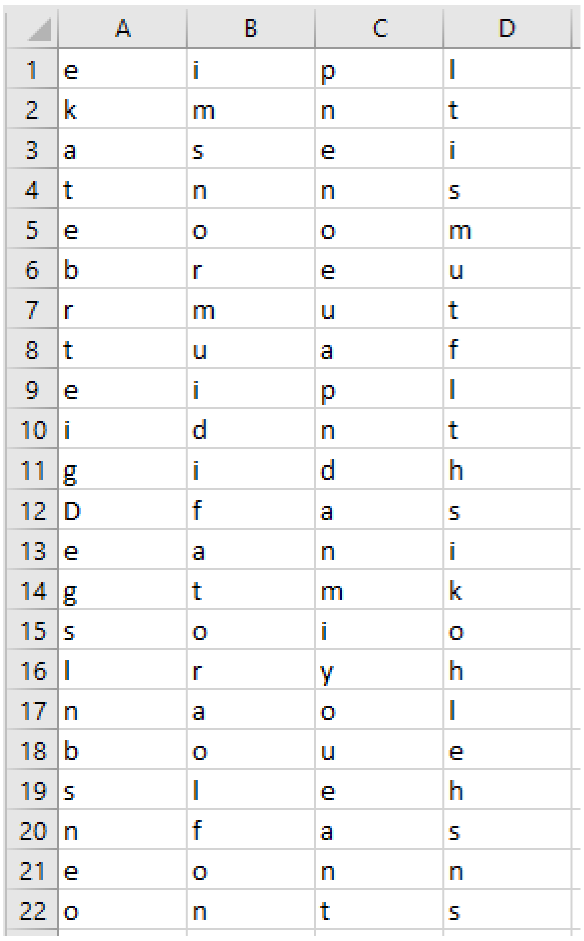
```python
    null_count = message.count('_')

    if null_count > 0:
        return message[: -null_count]

    return message

def main():
    cipher_text = "ekatebrteigDegslnbsneoimsnormuidifatoraolfonpnenoeuapndanmiyoueantltismutflthsikohlehsns"
    key = "bfgcdahe"
    decrypted_message = []
    # using list comprehension
    msg = decrypt(cipher_text, key)
    decrypted_message.append(msg)
    print(decrypted_message)

if __name__ == "__main__":
    main()
```

Figure 10: Main function given key and cipher text.



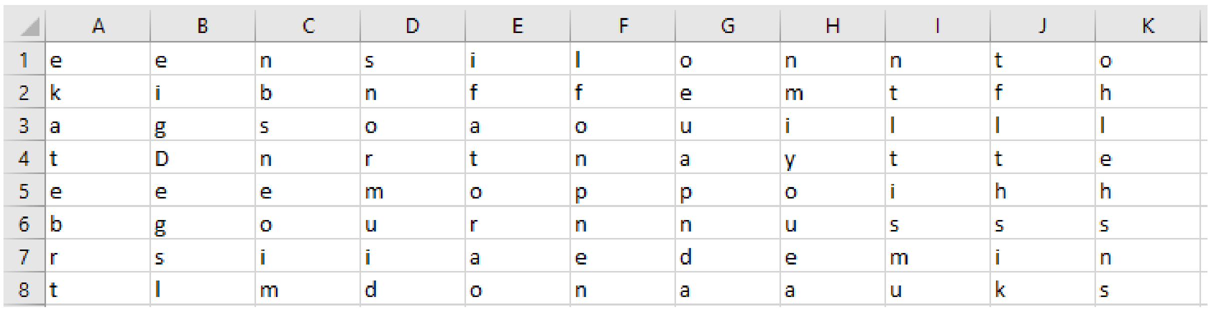
	A	B
1	e	p
2	k	n
3	a	e
4	t	n
5	e	o
6	b	e
7	r	u
8	t	a
9	e	p
10	i	n
11	g	d
12	D	a
13	e	n
14	g	m
15	s	i
16	l	y
17	n	o
18	b	u
19	s	e
20	n	a
21	e	n
22	o	t
23	i	l
24	m	t
25	s	i
26	n	s
27	o	m
28	r	u
29	m	t
30	u	f
31	i	l
32	d	t
33	i	h
34	f	s
35	a	i
36	t	k
37	o	o
38	r	h
39	a	l
40	o	e
41	l	h
42	f	s
43	o	n
44	n	s

Figure 11: Transposition cipher with a key length of 2.



	A	B	C	D
1	e	i	p	l
2	k	m	n	t
3	a	s	e	i
4	t	n	n	s
5	e	o	o	m
6	b	r	e	u
7	r	m	u	t
8	t	u	a	f
9	e	i	p	l
10	i	d	n	t
11	g	i	d	h
12	D	f	a	s
13	e	a	n	i
14	g	t	m	k
15	s	o	i	o
16	l	r	y	h
17	n	a	o	l
18	b	o	u	e
19	s	l	e	h
20	n	f	a	s
21	e	o	n	n
22	o	n	t	s

Figure 12: Transposition cipher with a key length of 4.



	A	B	C	D	E	F	G	H	I	J	K
1	e	e	n	s	i	l	o	n	n	t	o
2	k	i	b	n	f	f	e	m	t	f	h
3	a	g	s	o	a	o	u	i	l	l	l
4	t	D	n	r	t	n	a	y	t	t	e
5	e	e	e	m	o	p	p	o	i	h	h
6	b	g	o	u	r	n	n	u	s	s	s
7	r	s	i	i	a	e	d	e	m	i	n
8	t	l	m	d	o	n	a	a	u	k	s

Figure 8: Transposition cipher with a key length of 11.