Roll. No. A016	Name: Varun Khadayate
Class B.Tech CsBs	Batch: 1
Date of Experiment: 30-07-2021	Subject: Cryptology

Aim

To implement Monoalphabetic Cipher.

Theory

With only 25 possible keys, the Caesar cipher is far from secure. A dramatic increase in the key space can be achieved by allowing an arbitrary substitution. Before pro- ceeding, we define the term permutation. A permutation of a finite set of elements S is an ordered sequence of all the elements of S, with each element appearing exactly once. For example, if S = {a, b, c}, there are six permutations of S:

abc, acb, bac, bca, cab, cba

In general, there are ! permutations of a set of elements, because the first element can be chosen in one of n ways, the second in n - 1 ways, the third in n - 2 ways, and so on.

Recall the assignment for the Caesar cipher:

```
plain: a b c d e f g h i j k l m n o p q r s t u v w x y z
cipher: D E F g H I J K L M N O P q R S T U V W x y z a B C
```

If, instead, the “cipher” line can be any permutation of the 26 alphabetic characters, then there are 26! or greater than 4 * 1026 possible keys. This is 10 orders of magnitude greater than the key space for DES and would seem to eliminate brute-force techniques for cryptanalysis. Such an approach is referred to as a monoalphabetic substitution cipher, because a single cipher alphabet (mapping from plain alphabet to cipher alphabet) is used per message.

There is, however, another line of attack. If the cryptanalyst knows the nature of the plaintext (e.g., noncompressed English text), then the analyst can exploit the regularities of the language. To see how such a cryptanalysis might proceed, we give a partial example here that is adapted from one in [SINK09]. The ciphertext to be solved is

```
UzqSOVUOHxMOPVgPOzPEVSgzWSzOPFPESxUDBMETSxaIz
VUEPHzHMDzSHzOWSFPaPPDTSVPqUzWyMxUzUHSx
EPyEPOPDzSzUFPOMBzWPFUPzHMDJUDTMOHMq
```

As a first step, the relative frequency of the letters can be determined and compared to a standard frequency distribution for English, such as is shown in Figure 2.5 (based on [LEWA00]). If the message were long enough, this technique alone might be sufficient, but because this is a relatively short message, we cannot expect an exact match. In any case, the relative frequencies of the letters in the ciphertext (in percentages) are as follows:

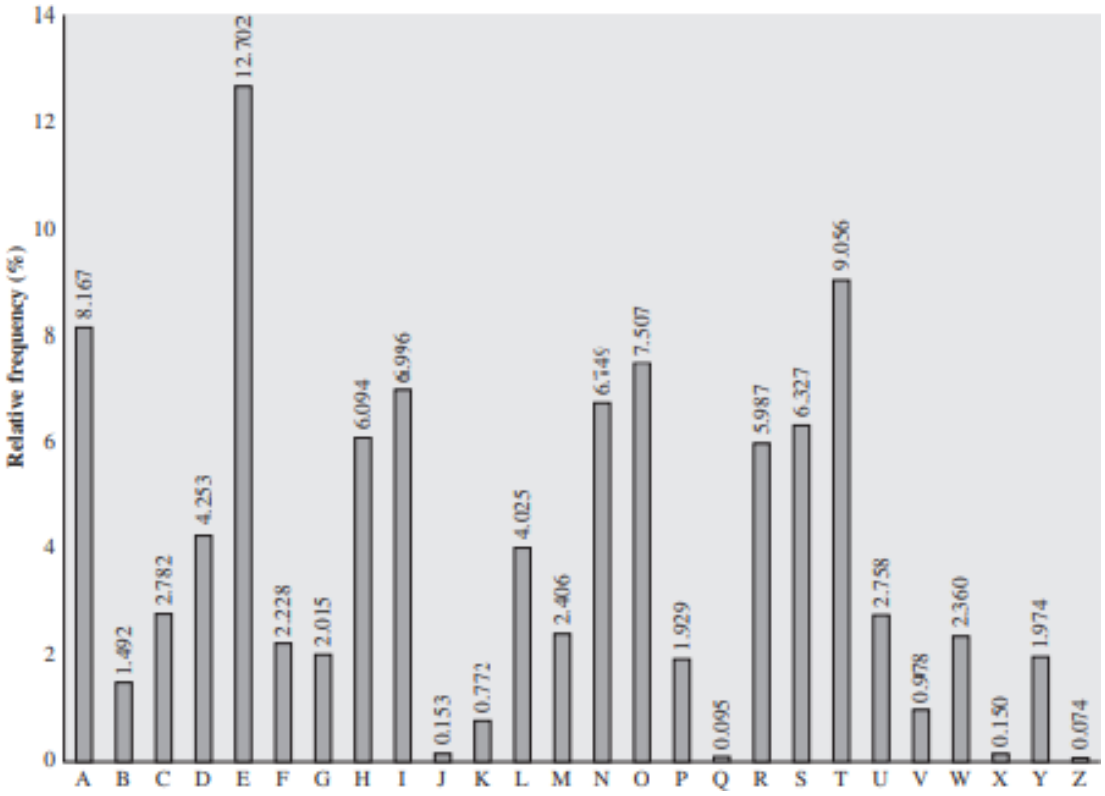
P	13.33	H	5.83	F	3.33	B	1.67	C	0.00
Z	11.67	D	5.00	W	3.33	G	1.67	K	0.00
S	8.33	E	5.00	Q	2.50	Y	1.67	L	0.00
U	8.33	V	4.17	T	2.50	I	0.83	N	0.00
O	7.50	X	4.17	A	1.67	J	0.83	R	0.00
M	6.67								

Comparing this breakdown with Figure 2.5, it seems likely that cipher letters P and Z are the equivalents of plain letters e and t, but it is not certain which is which. The letters S, U, O, M, and H are all of relatively high frequency and probably correspond to plain letters from the set {a, h, i, n, o, r, s}. The letters with the lowest frequencies (namely, A, B, G, Y, I, J) are likely included in the set {b, j, k, q, v, x, z}. There are a number of ways to proceed at this point. We could make some tentative assignments and start to fill in the plaintext to see if it looks like a reasonable "skeleton" of a message. A more systematic approach is to look for other regularities.

For example, certain words may be known to be in the text. Or we could look for repeating sequences of cipher letters and try to deduce their plaintext equivalents.

A powerful tool is to look at the frequency of two-letter combinations, known as **digrams**. A table similar to Figure 2.5 could be drawn up showing the relative frequency of digrams. The most common such digram is th. In our ciphertext, the most common digram is ZW, which appears three times. So, we make the correspondence of Z with t and W with h. Then, by our earlier hypothesis, we can equate P with e. Now notice that the sequence ZWP appears in the ciphertext, and we can translate that sequence as "the." This is the most frequent trigram (three-letter combination) in English, which seems to indicate that we are on the right track.

Next, notice the sequence ZWSZ in the first line. We do not know that these four letters form a complete word, but if they do, it is of the form th_t. If so, S equates with a.



So far, then, we have

```
UzqSOVUOHxMOPVgPOzPEVSgzWSzOPFPESxUDBMETSxaIz
 t a        e  e te  a that e e a       a
VUEPHzHMDzSHzOWSFPaPPDTSVPqUzWyMxUzUSx
  e t   ta t ha e ee  a e  th   t a
EPyEPOPDzSzUFPOMBzWPFUPzHMDJUDTMOHMq
 e  e e tat  e   the   t
```

Only four letters have been identified, but already we have quite a bit of the message. Continued analysis of frequencies plus trial and error should easily yield a solution from this point. The complete plaintext, with spaces added between words, follows:

```
it was disclosed yesterday that several informal
butdirect contacts have been made with political
representatives of the viet cong in moscow
```

Code

```
import random

while True:
    ch = int(input('Welcome to Mono-Alphabetic Cipher Encryption and
Decryption Program Made by Varun Khadayate..\n [*] Press 1 for Encryption \n
[*] Press 2 for Decryption \n [*] Press 0 to exit..\n \nYour Choice:: '))

    if ch == 1:
        print("\n=============================================")
        print("            !!!Encryption!!!                  ")
        text = input("\nText:").lower()

        alpha = [a for a in range(97,123)]

        charactrize=[ chr(al) for al in alpha]
        charactrize.append(' ')

        shuffled = random.sample(alpha,len(alpha))

        key =[chr(sh) for sh in shuffled]
        key.append(' ')

        print("\n=============================================")
        print("         !!!Encrypted Successfully!!!         ")
        print("\nEncypted Text ::")
        for tx in text:
            print(key[charactrize.index(tx)],end="")
        print("\nUsing Key:",key,"\n")
        print("\n=============================================")


    elif ch == 2:
        print("\n=============================================")
        print("               !!!Decryption!!!               ")
        text = input("Encrypted text:").lower()
        key = eval(input("Key:"))

        alpha = [a for a in range(97,123)]

        charactrize=[chr(al) for al in alpha]
        charactrize.append(' ')

        print("\n=============================================")
        print("         !!!Decrypted Successfully!!!         ")
        print("\nDecypted Text ::")
        for tx in text:
            print(charactrize[key.index(tx)],end="")
```

```
        print("\n=============================================")

    elif ch == 0:
        print("\n=============================================")
        print("   Thank You for using the Software ;)     ")
        print("                  Exiting Now.               ")
        print("=============================================")
        exit()
```

Output

```
PS E:\College-Codes\Fourth Year\SEM VII> & C:/Users/varun/AppData/Local/Programs/Python/Python310/python.exe "e:/College-Codes/Fourth Year/SE
M VII/CT/Monoalphabetic_Cipher.py"
Welcome to Mono-Alphabetic Cipher Encryption and Decryption Program Made by Varun Khadayate..
 [*] Press 1 for Encryption
 [*] Press 2 for Decryption
 [*] Press 0 to exit..

Your Choice:: 1

=========================================
          !!!Encryption!!!

Text:varun khadayate

=========================================
        !!!Encrypted Successfully!!!


=========================================
Welcome to Mono-Alphabetic Cipher Encryption and Decryption Program Made by Varun Khadayate..
 [*] Press 1 for Encryption
 [*] Press 2 for Decryption
 [*] Press 0 to exit..

Your Choice:: 2

=========================================
             !!!Decryption!!!
Encrypted text:yevdj pfemekeug
Key:['e', 's', 'l', 'm', 'g', 'i', 't', 'f', 'o', 'b', 'p', 'c', 'n', 'j', 'h', 'w', 'q', 'v', 'x', 'u', 'd', 'y', 'z', 'r', 'k', 'a', ' ']

=========================================
        !!!Decrypted Successfully!!!

Decypted Text ::
varun khadayate
=========================================
```

```
Welcome to Mono-Alphabetic Cipher Encryption and Decryption Program Made by Varun Khadayate..
 [*] Press 1 for Encryption
 [*] Press 2 for Decryption
 [*] Press 0 to exit..

Your Choice:: 0

=========================================
     Thank You for using the Software ;)
                Exiting Now.
=========================================
```