**CS6008 Cryptography and Network Security.**

**Assignment No.7**

**Name:** G Vignesh **Reg No:**2018103078

**Date:**6.6.2022

**MODULE V : INTRODUCTION TO CRYPTOGRAPHY**

**External Learning-** Implementing Simple Caesar Ciphers and breaking it using frequency analysis

**Aim:**

To implement Simple Caesar cipher encryption of any text data input with key **K** and to perform Frequency analysis with encrypted data , general frequency distribution of alphabets in English sentences to identify the keys and decrypt the Caesar encrypted data to plain text.

**Tools used:**

* Ubuntu terminal
* Python compiler
* Vscode (Code Editor)

**Description:**

The caesar cipher is named after Julius Caesar who used it when sending secret military messages to his troops. This is a simple substitution cipher where very character in the plain-text is shifted by a certain number known as the "key" or "shift". Key sizes are limited to 1 to 15 , since 0 and 26 does not shift alphabets, and values more than that are again repeat due to module 26.

**Input:** cipher text encrypted by Caesar cipher with key k

**Output**: With frequency analysis identify Key k and Decrypt the cipher to plain text

**Example:**

The following message:

**"Hello, captain"** And our shift is "2". We can then encode the message, one letter at a time. "H" would become "J",

since "J" is two letters away, and so on. If the shift is ever two large, or our letter is at the end of the alphabet, we just start at the beginning ("Z" would shift to "a" then "b" and so on).

Our final Caesar cipher would be **"Jgnnq, ecrvckp"**

**Encryption:**

**Cipher text = (plaintext character + key)mod26**

**Decryption:**

**Plain Text = (Cipher text character - key) mod 26**

**Execution:**

**Step-1:Encrypting the data using Caesar cipher.**

**Caesar-encrytor.py**

import sys

**def** encrypt(plain\_text,key) -> str:

    result = ""

*# traverse text*

    for i in range(len(plain\_text)):

        char = plain\_text[i]

*# Encrypt uppercase characters*

        if (char.isupper()):

            result += chr((ord(char) + key-65) % 26 + 65)

*# Encrypt lowercase characters*

        else:

            result += chr((ord(char) + key - 97) % 26 + 97)

    return result

if \_\_name\_\_=="\_\_main\_\_":

    if(len(sys.argv) <3):

        print("Enter the plaintext and space key to shift")

        exit(0)

    with open(sys.argv[1],"r") as f:

        plain\_text=f.read()

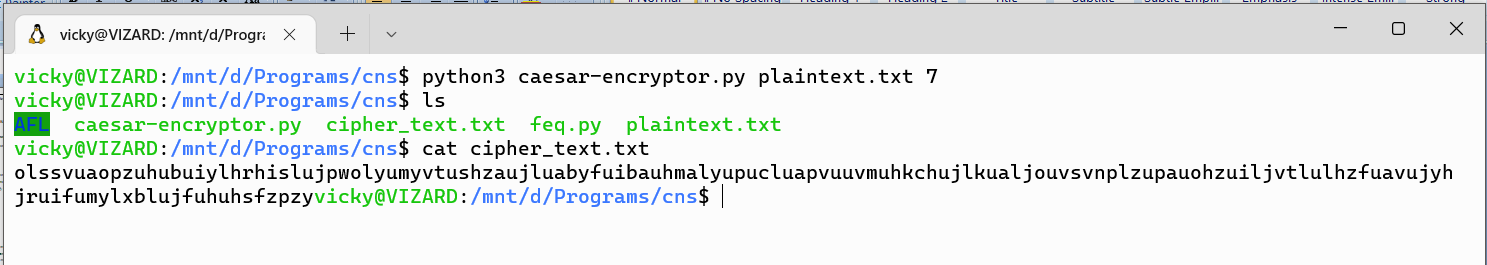
    key=sys.argv[2]

    cipher\_text=encrypt(plain\_text,int(key))

    with open("cipher\_text.txt","w") as f:

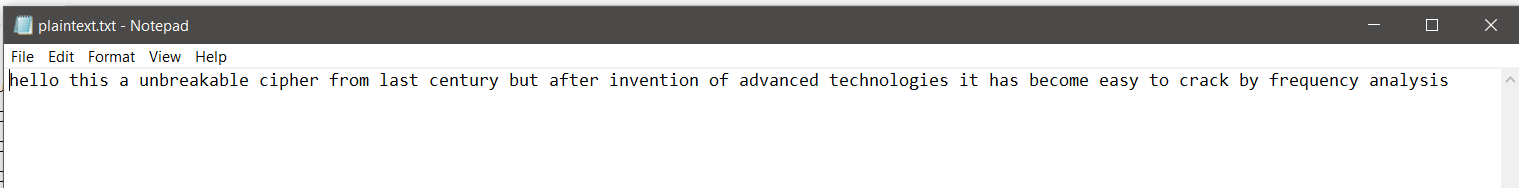
        f.write(cipher\_text)

plain text placed in file called plaintext.txt is passed to encryptor with key value as 7.

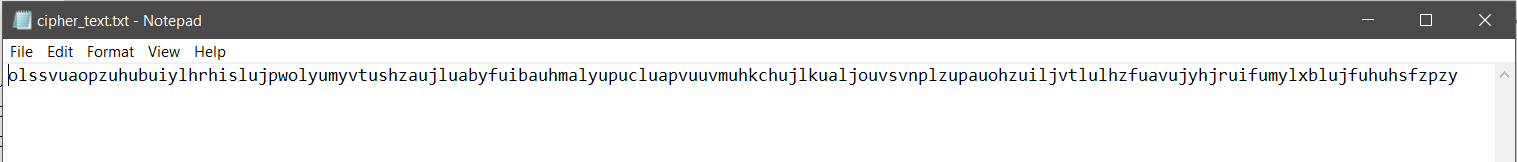


Text file name and key value is given as command line argument to the python program, cipher data created is placed in **cipher\_text.txt**  which is viewed by cat.

**Plain-text used:**

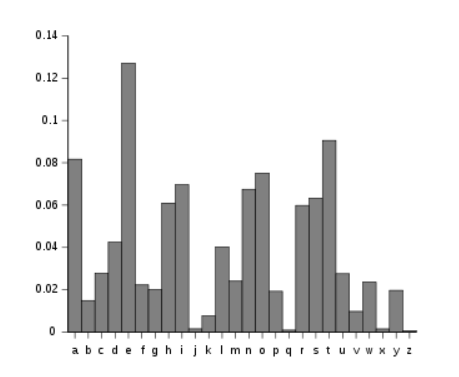
****

**Equivalent Caser-cipher obtained**:



**Step-2: Frequency analysis**

Frequency of alphabets in general English sentences is shown in a graph.

****

From a quick look at this graph we can see that some letters are much more common than others, for example, the letters **e,t,a,o,i** are the most common. We can leverage this information to our advantage in order to attack the Caser Cipher.

**LETTER\_FREQUENCY = {'e': 12.7, 't': 9.06, 'a': 8.17, 'o': 7.51, 'i': 6.97, 'n': 6.75, 's': 6.33, 'h': 6.09, 'r': 5.99, 'd': 4.25, 'l': 4.03, 'c': 2.78, 'u': 2.76, 'm': 2.41, 'w': 2.36, 'f': 2.23, 'g': 2.02, 'y': 1.97, 'p': 1.93, 'b': 1.29, 'v': 0.98, 'k': 0.77, 'j': 0.15, 'x': 0.15, 'q': 0.10, 'z': 0.07}**

Information Leaking in this cryptosystem, because the most used letter in the english alphabet is E.

So we know that the second most used letter in encrypted cipher text corresponds to E,since the first letter is the space.

**Step-3: Identifying the key**

Python code is implemented to count the frequency of each alphabet in the cipher text obtained , and take the second most and subtract from e to obtain the key value.

import sys

LETTERS = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'

*#The methode to do frequency analysis : we just count the occurrences of the given characters*

**def** frequency\_analysis(cipher\_text):

    cipher\_text = cipher\_text.upper()

    letter\_frequencies = {}

*#Initialize the dictionary (of course with 0 frequencies)*

    for letter in LETTERS :

        letter\_frequencies[letter] = 0

    for letter in cipher\_text:

*#We keep incrementing the occurence of the given letter*

        if letter in LETTERS:

            letter\_frequencies[letter]+=1

    return letter\_frequencies

**def** caesar\_crack(cipher\_text):

    frequency= frequency\_analysis(cipher\_text)

    print(frequency)

    frequency = sorted(frequency.items(), key=**lambda** x: x[1], reverse = True )

    print("\nHere is the frequency sorted : ")

    print(frequency)

*#To find the letter with the  second highest frequency*

    print("\nThe possible key value is : %s" % (LETTERS.find(frequency[1][0]) - LETTERS.find('E')))

if \_\_name\_\_ == '\_\_main\_\_':

    if(len(sys.argv) <2):

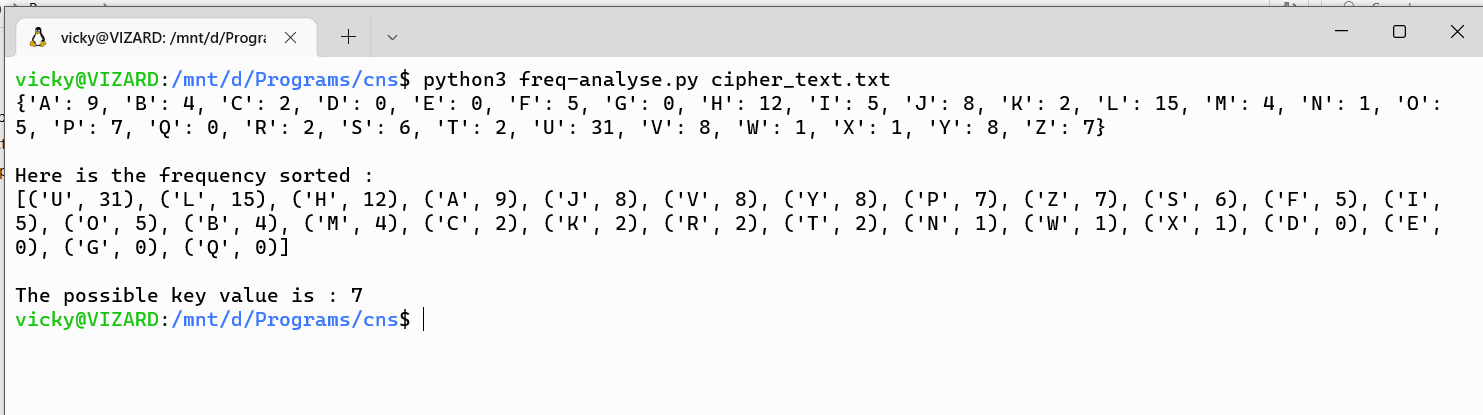
        print("Enter the cipher text file")

        exit(0)

    with open(sys.argv[1],"r") as f:

        cipher\_text=f.read()

    caesar\_crack(cipher\_text)



The cipher text obtained by earlier encryption is passed to the frequency analyser and we can see that **L** is the second most found word in ciphertext , when we relate it to E we can find that the key used may be 7.

**Step-4: Decryption by frequency analaysed key**

We know that the key used for Caesar cipher encrypton is 7 but in order verify our frequency analyser we check it with the Decrytor to confirm.

Caesar-decryptor

import sys

**def** decrypt(plain\_text,key) -> str:

    result = ""

*# traverse text*

    for i in range(len(plain\_text)):

        char = plain\_text[i]

*# Encrypt uppercase characters*

        if (char.isupper()):

            result += chr((ord(char) + key+65) % 26 + 65)

*# Encrypt lowercase characters*

        else:

            result += chr((ord(char) + key + 97) % 26 + 97)

    return result

if \_\_name\_\_=="\_\_main\_\_":

    if(len(sys.argv) <3):

        print("Enter the ciphertext file and space key to shift")

        exit(0)

    with open(sys.argv[1],"r") as f:

        cipher\_text=f.read()

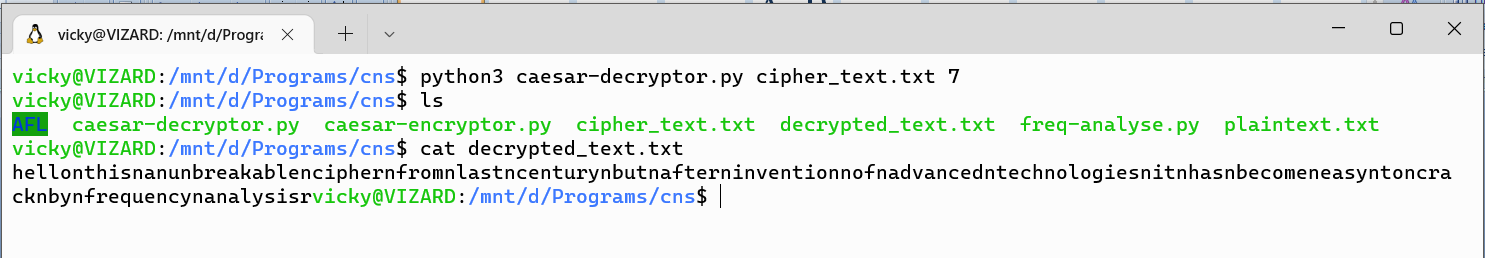
    key=sys.argv[2]

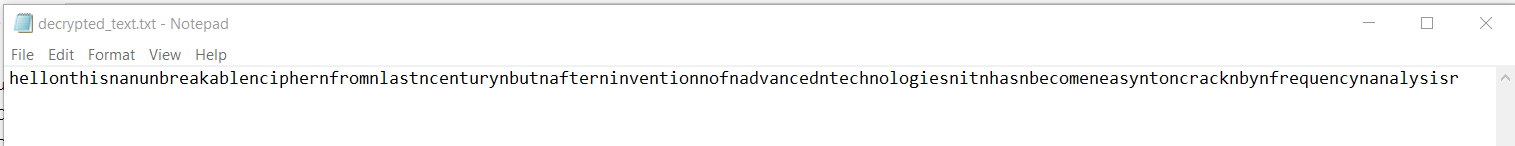
    plain\_text=decrypt(cipher\_text,int(key))

    with open("decrypted\_text.txt","w") as f:

        f.write(plain\_text)

The obtained Caesar cipher text and the key from the frequency analysis is passed the program and we get back the original text data but without space, since we only decipher the alphabets.





**Conclusion:**

Caesar cipher easily crackable by frequency analyse by leverging the factor of alphabets appear most common in English senetences are **e,t,a,o,i.** Another optionof frequency analysis is paired frequency analysis identifying what pair of sentences appear most commonly in cipher text, and match them to either **th or es or is**  which are common pairs in English and finding the keys.

Since its easily crackable many ciphers later arrived like affine cipher, hill cipher and the then the cryptography encryption algorithms like AES,DES to safe guard the data eventhough many new attacks are being practiced.