X. Xiao

CSE 545: Software Security
Assignment #2

Group No. 12
Lushaank Kancherla - 1224601664
Briana Rajan - 1225414593
Nagireddy Taaraka Siva Sai Teja Reddy - 1226129686

1. Please implement a Caesar Cipher for characters from 'A' to 'Z' (20 points).

Code (Python) -

```
def caesar_encrypt(text,shift):
    result = ""
    for i in range(len(text)):
        char = text[i]
        result += chr((ord(char) + shift-65) % 26 + 65)
    return result


text = input('Please enter Plain text: ')
shift = int(input('Please enter the Key Value: '))

print ("Plain Text is "+ text)
print ("Shift pattern is " + str(shift))
print ("Caesar Cipher is " + caesar_encrypt(text,shift))
```

Input and Output -

```
C:\Users\bibin\OneDrive\Documents\soft sec>python ques_01.py
Please enter Plain text: SOFTWARESECURITY
Please enter the Key Value: 4
Plain Text is SOFTWARESECURITY
Shift pattern is 4
Caesar Cipher is WSJXAEVIWIGYVMXC

C:\Users\bibin\OneDrive\Documents\soft sec>python ques_01.py
Please enter Plain text: TEAMWORK
Please enter the Key Value: 2
Plain Text is TEAMWORK
Shift pattern is 2
Caesar Cipher is VGCOYQTM
```

Steps to Understand -

When the code is run, the user is prompted to enter the Text he wanted to Encrypt and the Shift Key Value as inputs. The code calls the caesar_encrypt function. The function runs a for loop for the length of the Text provided and each character of the Text is then shifted to that many places. The Result is then shown in the Output as Encrypted Text.

2. Please implement a statistical attack for Caesar Cipher by computing Correlation: $\varphi(i)|i = [0,25]$ based on slides L7 pages 16-19. Use your implemented attack to infer the key and the plain text for the following cipher texts (30 points):
    a. XTKYBFWJXJHZWNYD
    b. KCECMKS

Code(Python) -

```
def prob(char, text):
   return(text.count(char)/len(text))

def caesar_decrypt(text,shift):
   result = ""
   for i in range(len(text)):
      char = text[i]
      result += chr((ord(char) - 65 - shift) % 26 + 65)
   return result

def caesar_stat_decrypt(text):
   strcharacters=("".join(set(text)))

frequencies=[0.080,0.015,0.030,0.040,0.130,0.020,0.015,0.060,0.065,0.005,0.005,0.035,0.030,0.070,
0.080,0.020,0.002,0.065,0.060,0.090,0.030,0.010,0.015,0.005,0.020,0.002]
   correlation= [0]*26
   for i in range(26):
      for j in strcharacters:
         c=ord(j) - 65
         p=prob(j,text)
         if(c-i <0):
            k=c-i+26
         else:
            k=c-i
         correlation[i]+= p*(frequencies[k])

          return (sorted(range(len(correlation)), key=lambda i: correlation[i])[-5:])

text = input("Please Enter Cipher Text:").replace(" ","")
top_shifts = caesar_stat_decrypt(text)
for x in top_shifts:
   print(caesar_decrypt(text,x))
```

Input and Output -

```
PS D:\ASU Courses\Sem-1\Software Security\Assignments\Assignment 2> python ques_02_new.py
Please Enter Cipher Text:XTKYBFWJXJHZWNYD
HDUILPGTHTRJGXIN
FBSGJNERFRPHEVGL
EARFIMDQEQOGDUFK
OKBPSWNAOAYQNEPU
SOFTWARESECURITY
PS D:\ASU Courses\Sem-1\Software Security\Assignments\Assignment 2> python ques_02_new.py
Please Enter Cipher Text:KCECMKS
WOQOYWE
TLNLVTB
EWYWGEM
ASUSCAI
MEGEOMU
PS D:\ASU Courses\Sem-1\Software Security\Assignments\Assignment 2>
```

Steps to Understand -
When the code is run, the user is prompted to enter the Cypher Text he wanted to Decrypt without the Shift Key Value. First it calls the stat_decrypt function in which it calculates the correlation for each of the shifts for the encrypted text for which it retains the information for the top five highest correlations and checks the shifts and returns the possible Texts for decrypts.

3. Please implement a variant of Vigenère cipher (20 points):
    a. Key length can be from 1-3
    b. Each character in the key must be upper case letters (i.e., 'A'-'Z')
    c. Each character in the cipher text can be any character from the ASCII table (https://en.wikipedia.org/wiki/ASCII)

Code(Python) -

```
def keyGenerate(string, key):
    key = list(key)
    if len(string) == len(key):
        return(key)
    else:
        for i in range(len(string) - len(key)):
            key.append(key[i % len(key)])
    return("" . join(key))

def cipherText(string, key):
    cipher_text = []
    for i in range(len(string)):
        x = (ord(string[i]) + ord(key[i])) % 94
        x += 33
        cipher_text.append(chr(x))
    return("" . join(cipher_text))

text = input("Please Enter Plain Text:")
keyword = input("Please Enter Keyword:")
key = keyGenerate(text, keyword)
cipher_text = cipherText(text,key)
print("Vigenere Ciphered Text is", cipher_text)
```

Input and Output -

```
PS D:\ASU Courses\Sem-1\Software Security\Assignments\Assignment 2\Assignment_02> python ques_03.py
Please Enter Plain Text:SoftwareSecurity
Please Enter Keyword:HTG
Vigenere Ciphered Text is ^(p!0k}|]pz!}"~&
PS D:\ASU Courses\Sem-1\Software Security\Assignments\Assignment 2\Assignment_02> python ques_03.py
Please Enter Plain Text:VIGnereCIPher
Please Enter Keyword:TGH
Vigenere Ciphered Text is mSR'o}|MTgrp+
PS D:\ASU Courses\Sem-1\Software Security\Assignments\Assignment 2\Assignment_02>
```

Steps to Understand -

In this variant of Vigenere Cipher implemented, the key length is checked initially and made to repeat or is reduced as many times as to match the length of the plain text given. With the key stream, the plain text is looped through by each character and ciphered. Ciphering is done to include all characters from ASCII values of 33 (!) to 126(~).

4. Please implement an exhaustive search algorithm to attack a simpler version of the Vigenère cipher built at question 3 (key length can be 1-3, and cipher text must use upper case letters). Use your implemented attack to infer the key for the following plaintext and ciphertext pairs. You need to provide **screenshots to show the input and the output of your code**, and **an output text file to show the keys and the total number of keys** in your algorithm search. (30 points):

Plain Text	**Cipher Text**
ARIZONASTATEUNIVERSITY	EUCDRHEVNEWYYQCZHLWLNC
COMPUTERSCIENCE	GRGTXNIUMGLYRFY

Code(Python) -

```
keynumber = 0
path = r"C:/Users/bibin/Downloads/"
def keyGenerate(string, key):
    global keynumber
    keynumber+=1
    key = list(key)
    if len(string) == len(key):
        return(key)
    else:
        for i in range(len(string) - len(key)):
            key.append(key[i % len(key)])
    return("" . join(key))

def decipher(string, text):
    length_key=0
    file = open(path+ text + ".txt", 'w')
    file.close()
    for i in range(26):
      key = chr(i+65)
      key_string = keyGenerate(string, key)
      if(decipher_function(string, key, key_string, text) == 1):
              return
```

```
    for i in range(26):
      for j in range(26):
         key = chr(i+65)+ chr(j+65)
         key_string = keyGenerate(string, key)
         if(decipher_function(string, key, key_string, text) == 1):
              return

    for i in range(26):
      for j in range(26):
         for k in range(26):
              key = chr(i+65) + chr(j+65) + chr(k+65)
              key_string = keyGenerate(string, key)
              if(decipher_function(string, key, key_string, text) == 1):
                 return
    return

def decipher_function(string, key, key_string, text):
    original_text = []
    file= open(path+ text + ".txt", 'a')
    for i in range(len(string)):
       x = (ord(string[i]) - ord(key_string[i]) + 26) % 26
       x += ord('A')
       original_text.append(chr(x))
    original_text = "" . join(original_text)
    if text == original_text:
       file.write("Shown below are the details: \n" + "Key Number: " + str(keynumber) + '\n' + ("Key :
" + key +"\n" "Original Text: " + original_text +"\n"))
       file.close()
       return(1)
    file.write (("Key: " + key +", " "Original Text: "+ original_text +"\n"))
    file.close()
    return

ciphered_text=input("Please Enter The Ciphered Text:")
text = input("Please Enter The Plain Text:")
decipher(ciphered_text, text)
```

Input and Output -

```
C:\Users\bibin\Downloads>python ques_04_new.py
Please Enter The Ciphered Text:EUCDRHEVNEWYYQCZHLWLNC
Please Enter The Plain Text:ARIZONASTATEUNIVERSITY

C:\Users\bibin\Downloads>python ques_04_new.py
Please Enter The Ciphered Text:GRGTXNIUMGLYRFY
Please Enter The Plain Text:COMPUTERSCIENCE
```

Steps to Understand -

When the code is run, the user is prompted to enter the Ciphered Text and the Plain Texts as inputs to find the key for the inputs. The texts are then sent to the decrypt function in which it exhaustively searches for each key value of length 1-3. Then, each key value is then sent to the keyGenerate function to match with the length of the text. At each iteration it checks to match with the original text and then prints the Key Value and the iterations it took to arrive at the original text (total number of keys checked to arrive at the solution). It then prints all the things mentioned above in a file.

To run the file -

Kindly specify the “path” in which the output txt file is to be created. The ‘path’ variable mentioned above in the code contains the path to the file and is to be modified in accordance to choice before executing the code.