**Param Patil**

**2019130049**

**TE Comps**

**Batch C**

**Experiment 1: Traditional Crypto Methods**

**Aim:**

**To implement Substitution, ROT 13, Transposition, Double Transposition and Vernam Cipher in Python.**

**Code:**

import math

print('1 Substitution')

print('2 ROT 13')

print('3 Transpose')

print('4 Double Transposition')

print('5 Vernam Cipher')

a =int(input('Enter the cryptography method:'))

def encryptBySubstitution(text,shift = 1):

cipher = ""

shift = shift%26

for letter in text:

if(ord(letter)>96 and ord(letter)<=122):

new\_value = (ord(letter) + shift)

if(new\_value > 122):

new\_value -= 26

cipher += chr((96 + new\_value)%96 + 96)

elif(ord(letter)>64 and ord(letter)<=90):

new\_value = (ord(letter) + shift)

if(new\_value > 90):

new\_value -= 26

cipher += chr((65 + new\_value)%65 + 65)

else:

cipher +=chr(ord(letter) + shift)

return cipher

def decryptBySubstitution(text,shift = 1):

cipher = ""

shift = shift%26

for letter in text:

if(ord(letter)>96 and ord(letter)<=122):

new\_value = (ord(letter) - shift)

if(new\_value < 97):

new\_value += 26

cipher += chr((96 + new\_value)%96 + 96)

elif(ord(letter)>64 and ord(letter)<=90):

new\_value = (ord(letter) - shift)

if(new\_value < 65):

new\_value +=26

cipher += chr((64 + new\_value)%64 + 64)

else:

cipher +=chr(ord(letter) - shift)

return cipher

def encryptByVernam(text,key):

cipher = ""

for i in range(len(text)):

cipher += chr(((ord(text[i])-65)^(ord(key[i])-65))+65)

return cipher

def decryptByVernam(cipher,key):

text = ""

for i in range(len(cipher)):

text += chr(((ord(cipher[i]) - 65)^(ord(key[i]) - 65)) + 65)

return text

def encryptByTranspose(text,key):

col = len(key)

row = math.ceil(len(text)/col)

# print('The dimensions are: ',row,col,text)

matrix = []

for i in range(0,row):

string = []

for j in range(0,col):

string.append("\_")

matrix.append(string)

# print('The constructed matrix is: ', matrix)

text\_index = 0

for i in range(0,row):

for j in range(0,col):

matrix[i][j] = text[text\_index]

text\_index += 1

if(text\_index >= len(text)):

break

if(text\_index >= len(text)):

break

# print('The column matrix is: ', matrix)

refer\_key = sorted(key)

# print('The sorted list is: ',refer\_key)

cipher = ""

for i in range(0,col):

iterate\_col\_index = key.index(refer\_key[i])

key = key[:iterate\_col\_index] + '-' + key[iterate\_col\_index+1:]

for j in range(0,row):

cipher += matrix[j][iterate\_col\_index]

return cipher

def decryptByTranspose(cipher,key):

col = len(key)

row = math.ceil(len(cipher)/col)

matrix = []

for i in range(0,row):

string = []

for j in range(0,col):

string.append("\_")

matrix.append(string)

refer\_key = sorted(key)

cipher\_index = 0

for i in range(0,col):

iterate\_col\_index = key.index(refer\_key[i])

key = key[:iterate\_col\_index] + '-' + key[iterate\_col\_index+1:]

for j in range(0,row):

matrix[j][iterate\_col\_index] = cipher[cipher\_index]

cipher\_index += 1

if(cipher\_index >= len(cipher)):

break

if(cipher\_index >= len(cipher)):

break

text = ""

for i in range(0,row):

for j in range(0,col):

# if(matrix[i][j] == "\_"):

# continue

text += matrix[i][j]

return text

def filterText(text):

string = ""

for letter in text:

if(letter!="\_"):

string+=letter

return string

if a == 1:

text = input('Enter Plain Text: ')

key = int(input('No of position to be shifted: '))

encrpytedText = encryptBySubstitution(text,key)

print('Encrypted Message',encrpytedText)

print('Decrypted Message',decryptBySubstitution(encrpytedText,key))

elif a == 2:

text = input('Enter Plain Text to be encrypted: ')

encrpytedText = encryptBySubstitution(text, 13)

print('Encrypted Message', encrpytedText)

print('Decrypted Message', decryptBySubstitution(encrpytedText, 13))

elif a == 3:

text = input('Enter Plain Text to be encrypted: ')

key = input('Enter key: ')

encryptedText = encryptByTranspose(text,key)

decryptedText = decryptByTranspose(encryptedText,key)

print("The encrypted cipher is: ",filterText(encryptedText))

print("The decrypted text is: ",filterText(decryptedText))

elif a == 4:

text = input('Enter Plain Text to be encrypted: ')

key1 = input('Enter key 1: ')

key2 = input('Enter key 2: ')

encryptedText1 = (encryptByTranspose(text,key1))

encryptedText2 = (encryptByTranspose(text,key2))

decryptedText1 = decryptByTranspose(encryptedText2,key2)

decryptedText2 = decryptByTranspose(encryptedText1,key1)

print("The encrypted cipher is: ",filterText(encryptedText2))

print("The decrypted text is: ",filterText(decryptedText2))

elif a == 5:

text = input('Enter Plain Text to be encrypted: ')

key = input('Enter key of the same length as the Plain Text:')

while(len(text)!=len(key)):

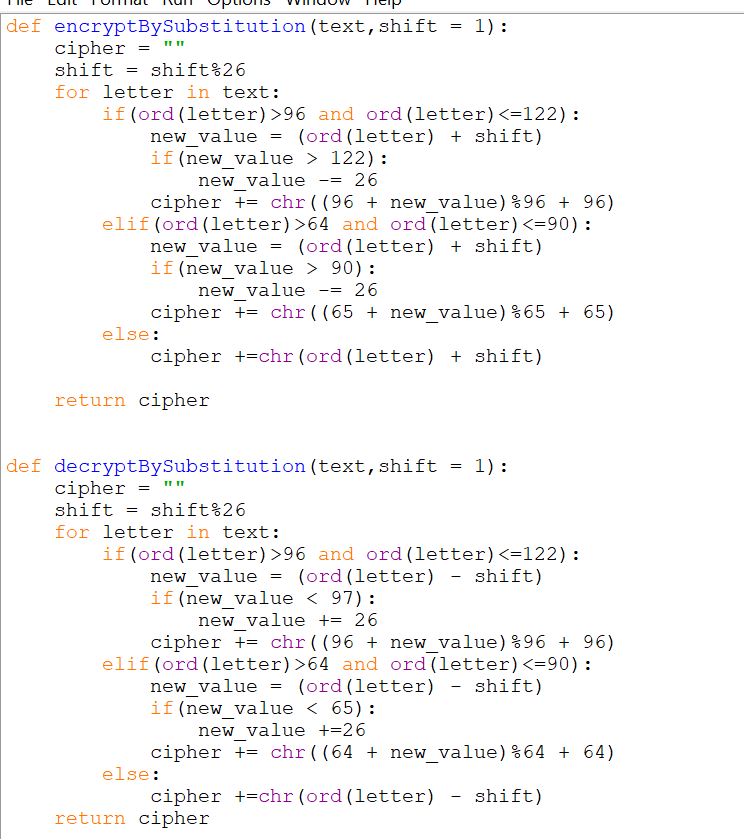
key = input('Enter key of the same length as the Plain Text: ')

encryptedText = (encryptByVernam(text,key))

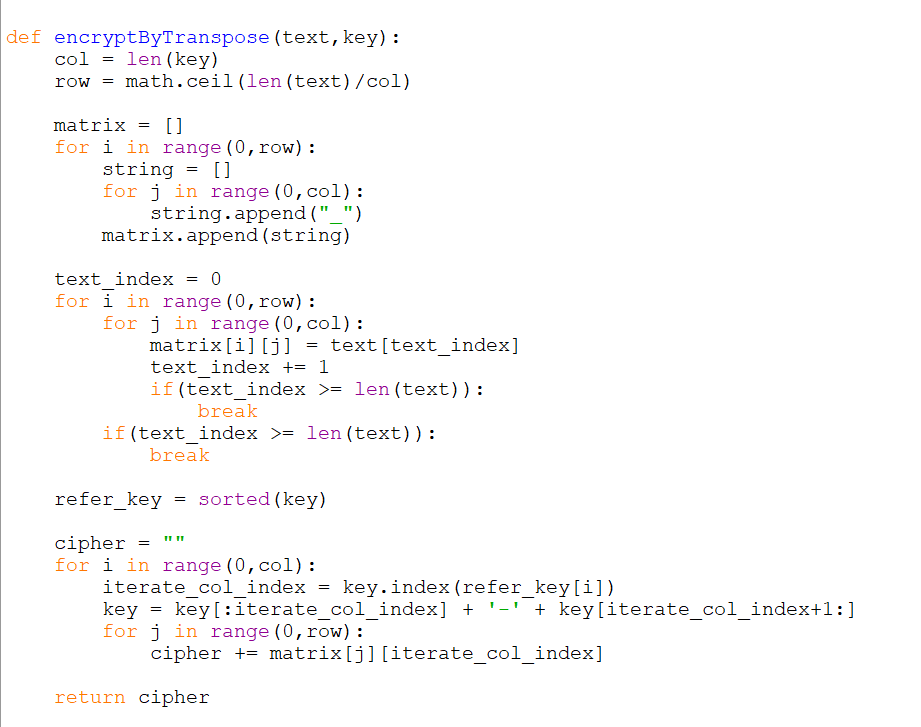
decryptedText = decryptByVernam(encryptedText,key)

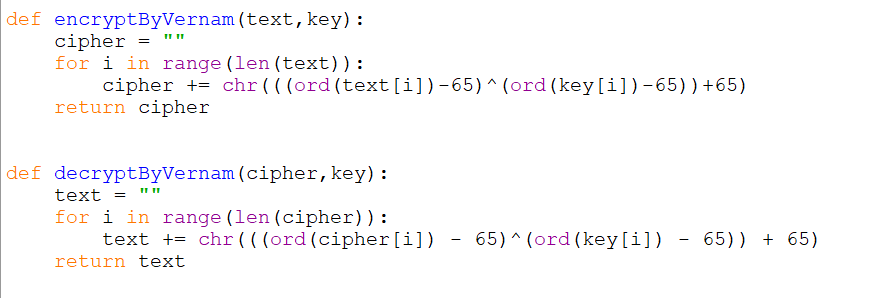
print("The encrypted cipher is: ",(encryptedText))

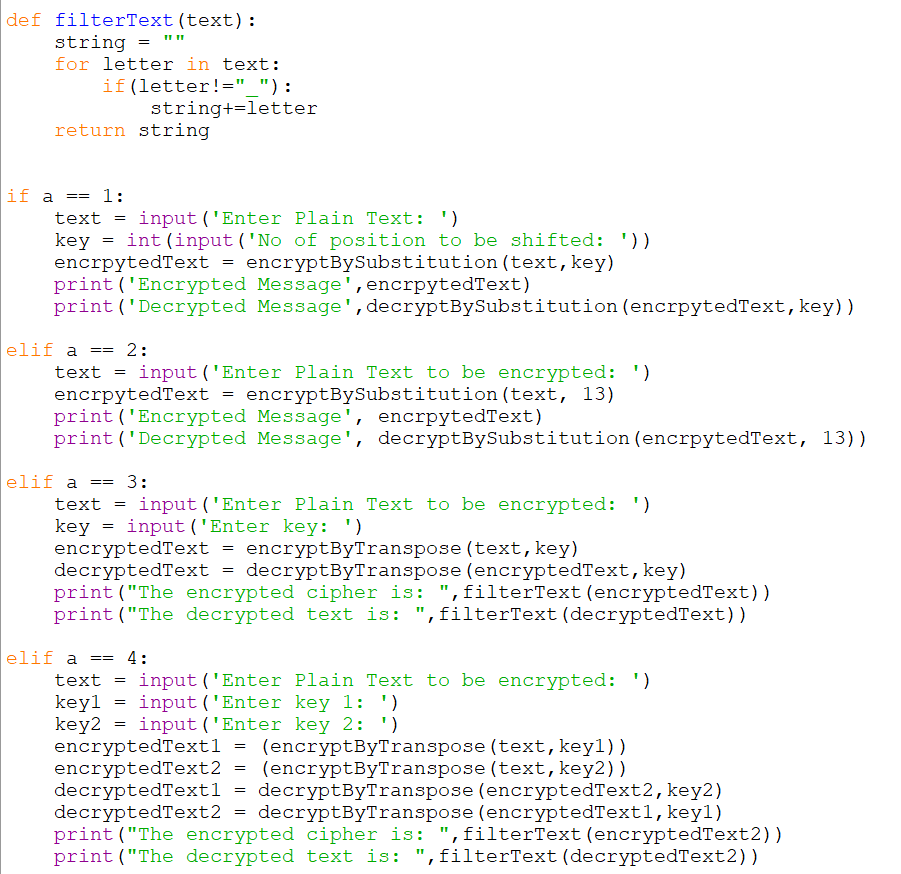
print("The decrypted text is: ",(decryptedText))

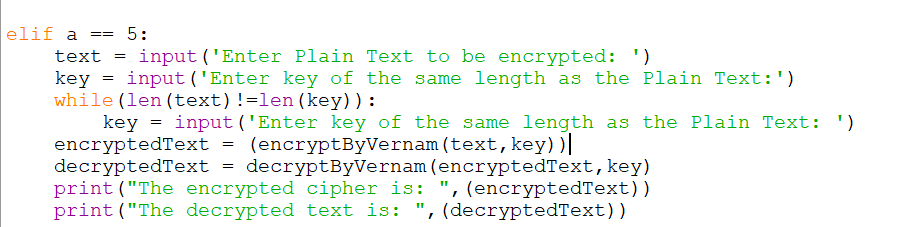




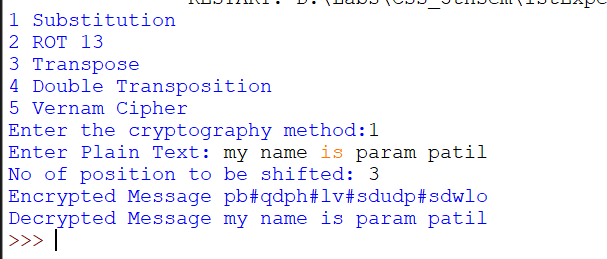




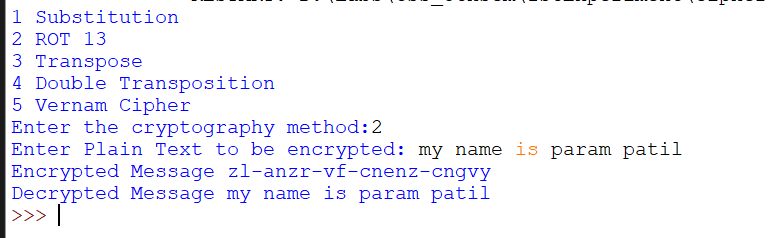




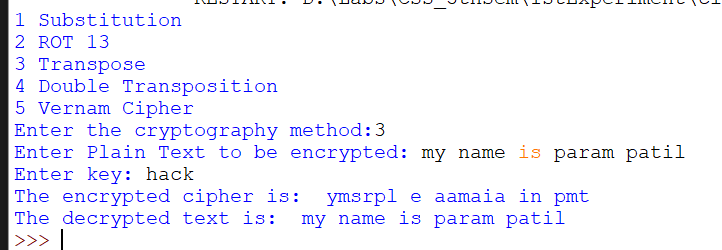
1. **Substitution Cipher**



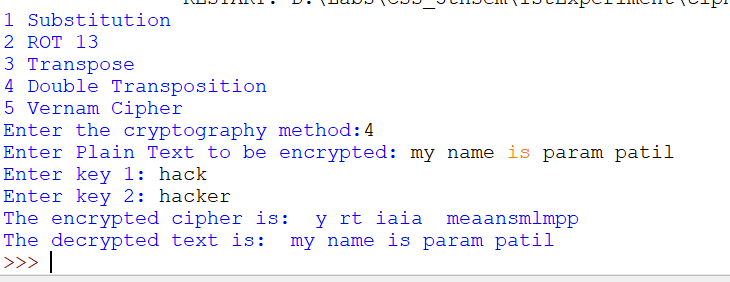
1. **ROT 13**



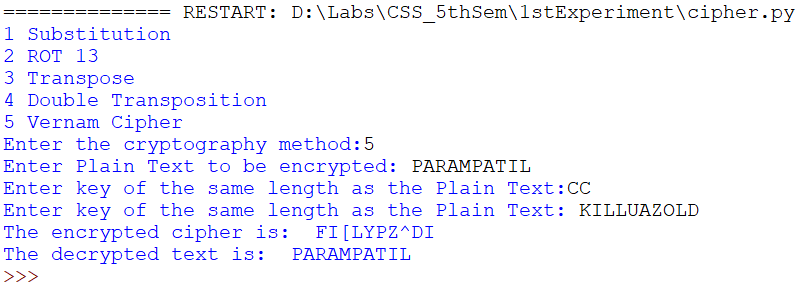
1. **Transpose Cipher**



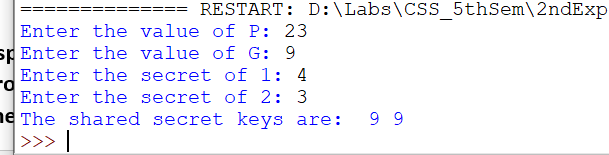
1. **Double Transposition**



1. **Vernam Cipher**



1. **Diffie Hellman**



**Conclusion:**

**In this experiment I have implemented 5 different traditional methods of encryption, where the implementation of the Substitution method was easiest and as a result it is also simple to decode this method. ROT13 is a subset of substitution method where the shift is fixed and it could also be easily deciphered.**

**Transpose and Double Transpose methods seem to be efficient and are more robust as a result difficult to crack. Double Transpose seem to be a better method as there is an additional layer of encryption.**

**In the second experiment, I implemented Diffie Hellman algorithm, which is a method to share a common secret key in public communication.**