Name: Ameya Jangam
UID:2019130025
Subject:CSS
TE COMPS

Experiment 1

Aim: To implement Substitution, ROT13, Transposition, Double Transposition, Vernam Cipher in Python.

Code:

```
import numpy as np
import math
import copy
def substitution():
   print ("You have selected substitution algorithm..")
   givenString=takeInput("Enter string:")
   shifts=int(input("Enter the no. of shifts "))
   encryptedText=""
   for char in givenString:
       if char==" ":
           encryptedText+=" "
       else:
           encryptedText+=chr(ord('a')+(ord(char)-ord('a')+shifts)%26)

   decryptedText=""
   for char in encryptedText:
       if char==" ":
           decryptedText+=" "
       else:
           decryptedText+=chr(ord('a')+(ord(char)-ord('a')-shifts+26)%26)
   printOutput(encryptedText,decryptedText)
   return

def rot13():
   print ("You have selected ROT13 algorithm..")
   givenString=takeInput("Enter string:")
   encryptedText=""
   shifts=13
   encryptedText=""
   for char in givenString:
```

```
        if char==" ":
            encryptedText+=" "
        else:
            encryptedText+=chr(ord('a')+(ord(char)-ord('a')+shifts)%26)

    decryptedText=""
    for char in encryptedText:
        if char==" ":
            decryptedText+=" "
        else:
            decryptedText+=chr(ord('a')+(ord(char)-ord('a')-shifts+26)%26)
    printOutput(encryptedText,decryptedText)
    return

def transpose():
    print ("You have selected Transposition algorithm..")
    givenString=takeInput("Enter string: ")
    key = input('Enter the key:')
    key.upper()
    order = sorted(list(key))
    col = len(key)

    # Encryption
    msg_len = len(givenString)
    msg_arr = list(givenString)
    row = int(math.ceil(msg_len/col))
    null_values = row*col - msg_len
    msg_arr.extend('_'*null_values)
    matrix = np.array(msg_arr).reshape(row,col)
    encryptedText = ''
    for i in range(col):
        index = key.index(order[i])
        encryptedText += ''.join([row[index] for row in matrix])
    print('Encrypted Text:',encryptedText)

    # Decryption
    encryptedText_arr = list(encryptedText)
    decryptedText = ''
    ptr = 0
    decrypt_matrix = np.array([None]*len(encryptedText)).reshape(row,col)
    for i in range(col):
        index = key.index(order[i])
        for j in range(row):
            decrypt_matrix[j,index] = encryptedText_arr[ptr]
            ptr += 1
```

```
    decryptedText = ''.join(''.join(x for x in y) for y in decrypt_matrix)
    #REMOVE UNDERSCORES FROM STRING
    real_decrypted_text=""
    for char in decryptedText:
        if char=="_":
            break
        else:
            real_decrypted_text+=char

    decryptedText=real_decrypted_text
    print('Decrypted Text:',decryptedText)

def doubleTranspose():
    print ("You have selected Double Transposition algorithm..")
    givenString=takeInput("Enter string:")
    key = input('Enter the key:')
    key.upper()
    order = sorted(list(key))
    col = len(key)

    ## Encryption
    msg_len = len(givenString)
    msg_list = list(givenString)
    row = int(math.ceil(msg_len/col))
    null_values = row*col - msg_len
    msg_list.extend('_'*null_values)
    matrix = np.array(msg_list).reshape(row,col)
    middleText,encryptedText = '',''

    for i in range(col):
        index = key.index(order[i])
        middleText += ''.join([row[index] for row in matrix])
    print("Single Encryption:",middleText)

    middletxt_lst = list(middleText)
    matrix = np.array(middletxt_lst).reshape(row,col)
    for i in range(col):
        index = key.index(order[i])
        encryptedText += ''.join([row[index] for row in matrix])
    print('Double Encryption:',encryptedText)

    ## Decryption
    encryptedText_lst = list(encryptedText)
    middleText,decryptedText = '',''
    pointer = 0
```

```
    dec_matrix = np.array([None]*len(encryptedText)).reshape(row,col)
    for i in range(col):
        index = key.index(order[i])
        for j in range(row):
            dec_matrix[j,index] = encryptedText_lst[pointer]
            pointer += 1

    middleText = ''.join(''.join(col for col in row) for row in dec_matrix)
    pointer = 0
    print('Single Decryption:',middleText)

    middletxt_lst = list(middleText)
    dec_matrix = np.array([None]*len(middleText)).reshape(row,col)
    for i in range(col):
        index = key.index(order[i])
        for j in range(row):
            dec_matrix[j,index] = middletxt_lst[pointer]
            pointer += 1
    real_decrypted_text=""
    for char in decryptedText:
        if char=="_":
            break
        else:
            real_decrypted_text+=char

    decryptedText=real_decrypted_text
    real_decrypted_text=""
    decryptedText = ''.join(''.join(col for col in row) for row in dec_matrix)
    real_decrypted_text=""
    for char in decryptedText:
        if char=="_":
            break
        else:
            real_decrypted_text+=char

    decryptedText=real_decrypted_text
    print('Double Decryption:',decryptedText)

def vernamCipher():
    print ("You have selected Vernam Cipher algorithm..")
    givenString=takeInput("Enter string:")
    givenKey=takeInput("Enter key:")

    if(len(givenString)!=len(givenKey)):
          return "\nError: Length of given string and given key should be equal\n"
```

```
    encryptedText=""
    for item in range(0,len(givenString)):
        if givenString[item]==" " and givenKey[item]==" ":
            encryptedText+=" "
        else:

encryptedText+=chr(ord('a')+(ord(givenString[item])+ord(givenKey[item])-2*ord('a')
)%26)

    decryptedText=""
    for item in range(0,len(givenString)):
        if encryptedText[item]==" " and givenKey[item]==" ":
            decryptedText+=" "
        else:

decryptedText+=chr(ord('a')+(ord(encryptedText[item])-ord(givenKey[item])+26)%26)

    printOutput(encryptedText,decryptedText)
    return


def exitter():
    print("Exiting...")
    exit()


def returnAlgoFunc(func):
    return func()

def takeInput(text):
    givenString=input(text)
    return givenString

def printOutput(encryptedText,decryptedText):
    print(f"Encrypted Text: {encryptedText}")
    print(f"Decrypted Text: {decryptedText}")


print("Select one of the cryptography methods:")
switcher = {
    1: substitution,
    2: rot13,
    3: transpose,
    4: doubleTranspose,
    5: vernamCipher,
```

```
    6:exitter
}

while True:
    print("1. Substitution Algorithm")
    print("2. ROT 13 Algorithm")
    print("3. Transpose Algorithm")
    print("4. Double Transposition Algorithm")
    print("5. Vernam Cipher Algorithm")
    print("6. Exitter")
    print("Enter your option:")
    option = int(input())
    returnAlgoFunc(switcher.get(option))
```

Output:

1. Substitution Algorithm

```
ameya@ameya-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~/Ameya-Ubuntu/SPIT LAB SUBMISSI
ONS/SEM 5$ python3 exp1.py
Select one of the cryptography methods:
1. Substitution Algorithm
2. ROT 13 Algorithm
3. Transpose Algorithm
4. Double Transposition Algorithm
5. Vernam Cipher Algorithm
6. Diffie Hellman Algorithm
7. Exit
Enter your option:
1
You have selected substitution algorithm..
Enter string:ameya jangam
Enter the no. of shifts 2
Encrypted Text: cogac lcpico
Decrypted Text: ameya jangam
```

2. ROT13 Algorithm

```
1. Substitution Algorithm
2. ROT 13 Algorithm
3. Transpose Algorithm
4. Double Transposition Algorithm
5. Vernam Cipher Algorithm
6. Diffie Hellman Algorithm
7. Exit
Enter your option:
2
You have selected ROT13 algorithm..
Enter string:ameya jangam is the attacker
Encrypted Text: nzrln wnatnz vf gur nggnpxre
Decrypted Text: ameya jangam is the attacker
```

3. Transposition Algorithm

```
Select one of the cryptography methods:
1. Substitution Algorithm
2. ROT 13 Algorithm
3. Transpose Algorithm
4. Double Transposition Algorithm
5. Vernam Cipher Algorithm
6. Diffie Hellman Algorithm
7. Exit
Enter your option:
3
You have selected Transposition algorithm..
Enter string: ameya jangam is the attacker
Enter the key:210
Encrypted Text: e nmshaae_maaait tk_ayjg  etcr
Decrypted Text: ameya jangam is the attacker
```

4. Double Transposition Algorithm

```
1. Substitution Algorithm
2. ROT 13 Algorithm
3. Transpose Algorithm
4. Double Transposition Algorithm
5. Vernam Cipher Algorithm
6. Diffie Hellman Algorithm
7. Exit
Enter your option:
4
You have selected Double Transposition algorithm..
Enter string:ameya jangam is the attacker
Enter the key:210
Single Encryption: e nmshaae_maaait tk_ayjg  etcr
Double Encryption: nheaitager sama _j cema_atky t
Single Decryption: e nmshaae_maaait tk_ayjg  etcr
Double Decryption: ameya jangam is the attacker
```

5. **Vernam Cipher Algorithm**

```
1. Substitution Algorithm
2. ROT 13 Algorithm
3. Transpose Algorithm
4. Double Transposition Algorithm
5. Vernam Cipher Algorithm
6. Diffie Hellman Algorithm
7. Exit
Enter your option:
5
You have selected Vernam Cipher algorithm..
Enter string:michael patil
Enter key:bhushan scott
Encrypted Text: npwzhey hchbe
Decrypted Text: michael patil
```

6. **Diffie Hellman Algorithm**

```
1. Substitution Algorithm
2. ROT 13 Algorithm
3. Transpose Algorithm
4. Double Transposition Algorithm
5. Vernam Cipher Algorithm
6. Diffie Hellman Algorithm
7. Exit
Enter your option:
6
Enter Prime Number (g):7
Enter Second Prime Number (p):11
Enter Secret (Xa):6
Enter Secret (Xb):9
Ya:4
Yb:8
The secret key k1 shared with A is: 3
The secret key k2 shared with B is: 3
```

Conclusion:

From this experiment, I got to learn various encryption algorithms and implement them in Python.

1. **Substitution Method & ROT13 Method**
   The substitution and ROT13 algorithms required that when we substitute the ASCII value of a character, the final ordinal number of all characters should not exceed the total number of characters available and hence modularising the ordinal number to prevent overflow which leads to run time error.

2. **Single Transposition Method**

I faced problems while implementing the Transposition Algorithm since the solution that I implemented wasn't very efficient and the code was hard to read and that solution didn't handle null spaces and regular spaces correctly and used to give me the wrong decrypted text. A better implementation that I thought of was to not create multiple matrices for creating encrypted texts and decrypted texts, instead read them in a different order while preserving the original matrix order and reserve the computation time and space required for creating and creating new matrices with different orders. The NumPy array made the operations for reading the text characters much easier when it was earlier implemented via regular arrays.

3. **Double Transposition Method**

Once I learned about the single transposition method, this became much easier as I had to just doubly encrypt and reverse my way out hence doubly decrypting the files. This method proves to be more secure than the single transposition method.

4. **Vernam Cipher Method**

The Vernam cipher involved the concept of encrypting it with a key of same length and modularising the result with 26 (Total number of alphabets). Since the given string consists of spaces too, I took care of it by skipping the spaces and focussing on the important text which had to be encrypted.

5. **Diffie Hellman Method**

From Diffie Hellman Method, I learnt how we can exchange shared secret keys over an insecure channel with the help of simple mathematical operations.

The Diffie Hellman algorithm makes use of primitive roots of any prime numbers that are given by the user. In this case, I have given the two prime numbers (G and N) as inputs in order to save the computation time of checking whether the number is prime or not.

Since the exponentiation and modulo operation are used together in this algorithm, I used the pow(a,b,p) which took care of the exponentiation and then did a modular operation on it i.e replicating the expression $a^b \% p$ which is used in calculating the generating keys i.e Y_a and Y_b.

GitHub Link = https://github.com/ameyajangam22/CSS-LAB-2019130025