Sumeet Haldipur

2019130018

TE Comps

Batch A

Experiment 1: Traditional Crypto Methods

Github Link: https://github.com/sumeethaldipur/CSS_LAB/blob/main/Experiment1.py

Aim:

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

Code:

```
print('Select Crypto Method')

print('1 Substitution')

print('2 ROT 13')

print('3 Transpose')

print('4 Double Transposition')

print('5 Vernam Cipher')


a =(int)(input('Enter the Crypto Method: '))


def encrypt(b, key):

  matrix = createMatrix(len(key), b)

  sequence = getSequence(key)

  ans = ""

  for num in range(len(sequence)):

    pos = sequence.index(num+1)

    for row in range(len(matrix)):

      if len(matrix[row]) > pos:
```

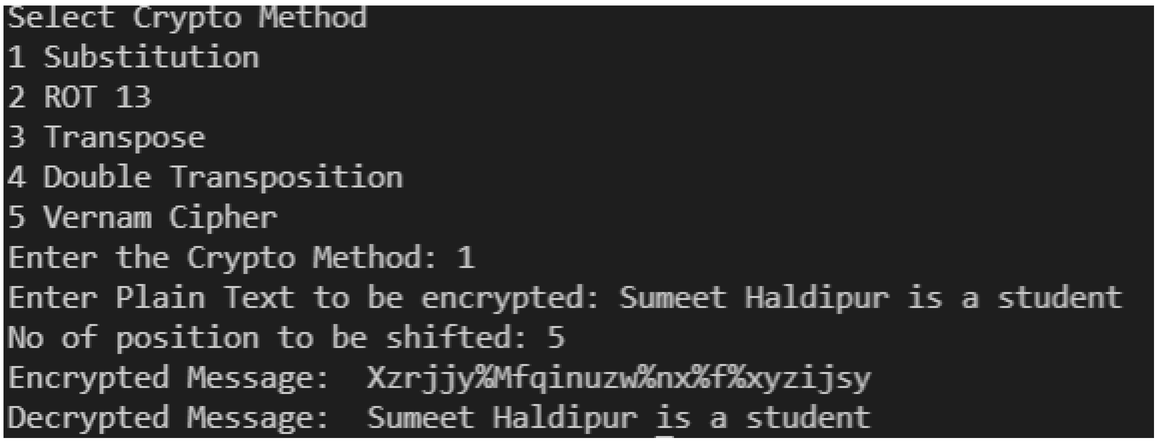
```
      ans += matrix[row][pos]
  return ans


def createMatrix(width, b):
  r = 0
  c = 0
  matrix = [[]]
  for pos, ch in enumerate(b):
    matrix[r].append(ch)
    c += 1
    if c >= width:
      c = 0
      r += 1
      matrix.append([])

  return matrix


def getSequence(key):
  sequence = []
  for pos, ch in enumerate(key):
    previousLetters = key[:pos]
    newNumber = 1
    for previousPos, previousCh in enumerate(previousLetters):
      if previousCh > ch:
        sequence[previousPos] += 1
      else:
        newNumber += 1
    sequence.append(newNumber)
  return sequence
```

```
def decrypt(message, keyword):
  matrix = createDecrMatrix(getSequence(keyword), message)

  plaintext = ""
  for r in range(len(matrix)):
    for c in range (len(matrix[r])):
      plaintext += matrix[r][c]
  return plaintext


def createDecrMatrix(keywordSequence, message):
  width = len(keywordSequence)
  height = len(message) // width
  if height * width < len(message):
    height += 1

  matrix = createEmptyMatrix(width, height, len(message))

  pos = 0
  for num in range(len(keywordSequence)):
    column = keywordSequence.index(num+1)

    r = 0
    while (r < len(matrix)) and (len(matrix[r]) > column):
      matrix[r][column] = message[pos]
      r += 1
      pos += 1
  return matrix



def createEmptyMatrix(width, height, length):
  matrix = []
  totalAdded = 0
  for r in range(height):
    matrix.append([])
    for c in range(width):
      if totalAdded >= length:
        return matrix
      matrix[r].append('')
      totalAdded += 1
  return matrix

# SUBSTITUTION CIPHER
def encryptS(b,k):
    newS = ''
    for i in range(len(b)):
        val = ord(b[i])
        dup = k
        d = val + k
        if val >= 97 and val <= 122:
            if d > 122:
                k -= (122 - val)
                k = k % 26
                newS += chr(96 + k)
            else:
                newS += chr(d)
            k = dup
        elif val >= 65 and val <= 90:
            if d > 90:
                k -= (90 - val)
                k = k % 26
                newS += chr(64 + k)
            else:
                newS += chr(d)
            k = dup
        else:
            newS += chr(d)
    return newS
# SUBSTITUTION DECRYPTION
def decryptS(b,k):
    s = ''
    for i in range(len(b)):
        val = ord(b[i])
        dup = k
        d = val-k
        if val >= 97 and val <= 122:
            if d < 97:
                k += (122-val)
                k = k%26
                s += chr(122 - k)
            else:
                s += chr(d)
            k = dup
        elif val >= 65 and val <= 90:
            if d < 65:
```

```
            k += (90 - val)

            k = k%26

            s += chr(90 - k)

          else:

            s += chr(d)

          k = dup

        else:

          s += chr(d)

    return s


def generateKey(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 encryptVernam(b,final_key):

    b_list = []

    k_list = []

    final_l = []

    for c in b:

        b_list.append(ord(c) - 65)

    for d in final_key:

        k_list.append(ord(d) - 65)

    for i in range(0, len_b):
        final_l.append((b_list[i] + k_list[i]) % 26)
    ans = ''
    for i in range(0, len_b):
        ans = ans + chr(final_l[i] + 65)
    return ans



def decryptVername(encrypted,key):
    orig_text = []
    for i in range(len(encrypted)):
        d = (ord(encrypted[i])-ord(key[i])+26)%26
        d += ord('A')
        orig_text.append(chr(d))
    return ("".join(orig_text))


if a==1:
    b = input('Enter Plain Text to be encrypted: ')
    k = int(input('No of position to be shifted: '))
    s = encryptS(b,k)
    print('Encrypted Message: ',s)
    print('Decrypted Message: ',decryptS(s,k))
elif a==2:
    b = input('Enter Plain Text to be encrypted: ')
    newS = ''
    s = encryptS(b, 13)
    print('Encrypted Message: ', s)
    print('Decrypted Message: ', decryptS(s, 13))
```

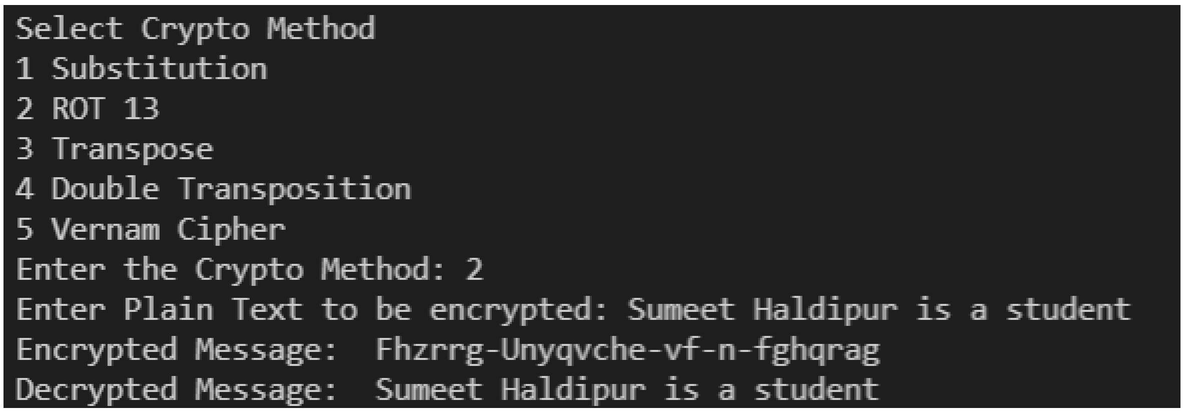
```
elif a==3:
    b = input('Enter Plain Text to be encrypted: ')
    key = input('Enter key: ')
    encypted = (encrypt(b,key))
    decrypted = decrypt(encypted,key)
    print('Encrypted Message: ',encypted)
    print('Decrypted Message: ',decrypted)
elif a==4:
    b = input('Enter Plain Text to be encrypted: ')
    key = input('Enter key:')
    encrypted = encrypt(encrypt(b,key),key)
    decrypted = decrypt(decrypt(encrypted,key),key)
    print(encrypted)
    print(decrypted)
elif a==5:
    b = input('Enter Plain Text to be encrypted: ')
    b.upper()
    len_b = len(b)
    key = input('Enter key text: ')
    key.upper()
    final_key = generateKey(b,key)
    encrypted = (encryptVernam(b,final_key))
    decrypted = decryptVername(encrypted,final_key)
    print(encrypted)
    print(decrypted)
```

1) **Substitution Cipher**



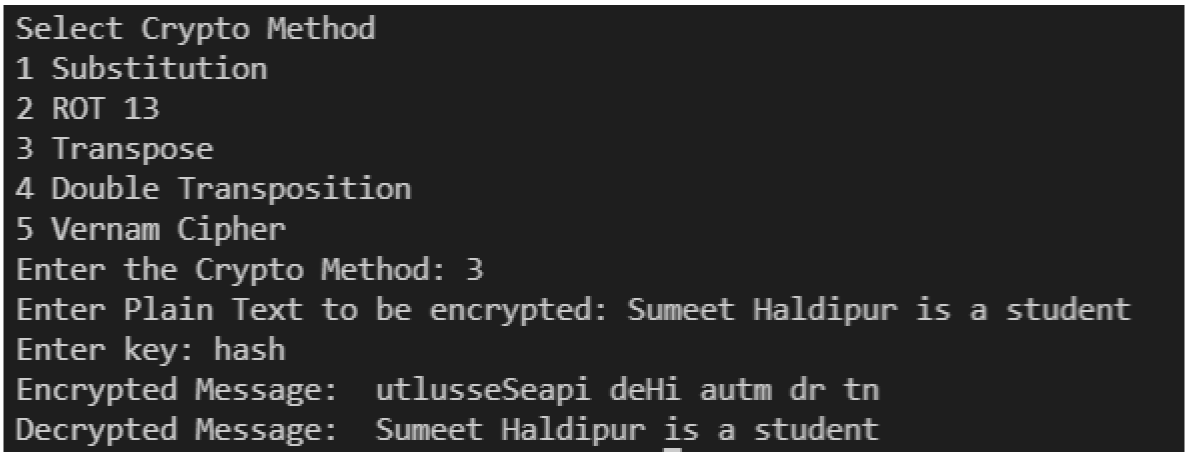
```
Select Crypto Method
1 Substitution
2 ROT 13
3 Transpose
4 Double Transposition
5 Vernam Cipher
Enter the Crypto Method: 1
Enter Plain Text to be encrypted: Sumeet Haldipur is a student
No of position to be shifted: 5
Encrypted Message:  Xzrjjy%Mfqinuzw%nx%f%xyzijsy
Decrypted Message:  Sumeet Haldipur is a student
```

2) **ROT 13**



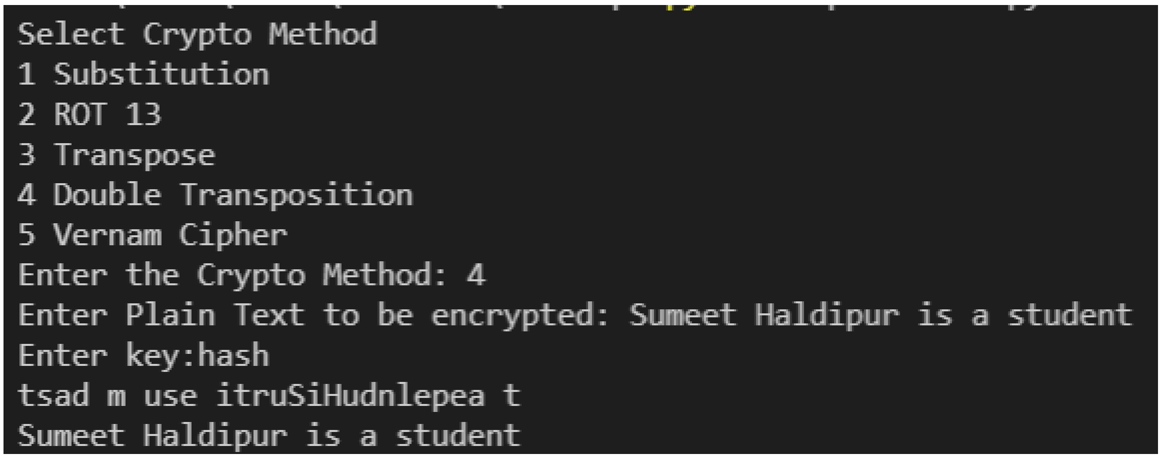
```
Select Crypto Method
1 Substitution
2 ROT 13
3 Transpose
4 Double Transposition
5 Vernam Cipher
Enter the Crypto Method: 2
Enter Plain Text to be encrypted: Sumeet Haldipur is a student
Encrypted Message:  Fhzrrg-Unyqvche-vf-n-fghqrag
Decrypted Message:  Sumeet Haldipur is a student
```

3) **Transpose Cipher**



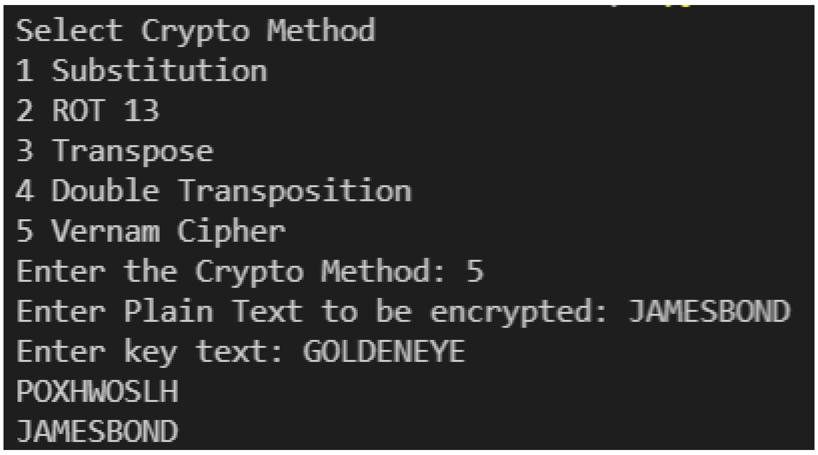
```
Select Crypto Method
1 Substitution
2 ROT 13
3 Transpose
4 Double Transposition
5 Vernam Cipher
Enter the Crypto Method: 3
Enter Plain Text to be encrypted: Sumeet Haldipur is a student
Enter key: hash
Encrypted Message:  utlusseSeapi deHi autm dr tn
Decrypted Message:  Sumeet Haldipur is a student
```

4) **Double Transposition**



```
Select Crypto Method
1 Substitution
2 ROT 13
3 Transpose
4 Double Transposition
5 Vernam Cipher
Enter the Crypto Method: 4
Enter Plain Text to be encrypted: Sumeet Haldipur is a student
Enter key:hash
tsad m use itruSiHudnlepea t
Sumeet Haldipur is a student
```

5) **Vernam Cipher**



```
Select Crypto Method
1 Substitution
2 ROT 13
3 Transpose
4 Double Transposition
5 Vernam Cipher
Enter the Crypto Method: 5
Enter Plain Text to be encrypted: JAMESBOND
Enter key text: GOLDENEYE
POXHWOSLH
JAMESBOND
```

Conclusion:

In this experiment I have implemented 5 different traditional methods of encryption:

Substitution:

In this algorithm, the characters in the inserted text are shifted in a particular format for encryption. The ASCII value of the characters in the inserted text is shifted by the number inserted (i.e. the number of positions to be shifted.) The new ASCII values correspond to new characters that take place of the original characters. Here, the shift of the value can be infinite but to limit to alphabetic representation after encryption, the shift/key that was taken as an input is first modded by 26, as a result the maximum shift an alphabet can take is 26. This generally results in the use of finite ASCII values for this algorithm. Hence, this algorithm is not directly used in modern cryptography, but it lays the foundation for creation of complex cryptography algorithms.

Rot 13:

This algorithm can be defined as a specific case of substitution cipher. In this, the value by which the character is to be shifted is restricted to exactly 13 places. In simple terms, it rotates the ASCII value of each character by 13 and hence it is called "ROT 13". In this, there is rotation of characters for eg. If I were to pick a character as Y then it would be substituted by N. Implementation of this algorithm is similar to substitution algorithm whose observations are mentioned above. The value of the shift is restricted to 13 which makes this algorithm relatively easier to crack and hence limits its applications in modern times.

Transpose:

In this cipher, the plaintext is arranged in columns according to the length of the key meaning that the number of characters in the key decide the number of columns. The plaintext is put in the columns going from left to right. The characters of the key are numbered according to the order in which they appear as the letters of the alphabet. Then, the encrypted text is derived by writing down the characters in the columns in the ascending order of their numbering. The difficulty to crack the encrypted text would depend upon the length of the key and plain text i.e long plain text and short key would be difficult to crack then short text and short key.

Double Transpose:

Transposition can be made stronger by adding another layer of encryption using Double Transposition. This follows the same process as that of transposition but now the input plaintext is taken as the encrypted text that we get through transposition. This is mainly used to provide another level of security to the transposition algorithm.

Vernam:

This cipher was initially used in early 1920's to encode messages in teletype system. Instead of a single key, each plaintext character is encrypted using its own key. This key — or key stream — is randomly generated or is taken from a one-time pad, e.g. a page of a book. The key must be equal in length to the plaintext message. The fact that each character of the message is encrypted using a different key prevents any useful information being revealed through a frequency analysis of the ciphertext. To encrypt the message, each character of the plaintext and the key will need to be converted to their ASCII values. In this the length of the key should be same as that of the plaintext. In this, every character in the plaintext is encrypted using its corresponding character in the key (sequentially). Larger the text, harder the cipher is to crack since the only way to crack this algorithm is by trying out various combinations which would require heavy computation. Vernam cipher is still widely spread in the modern cryptography.