NAME: Chaitya Shah

BATCH: C

UID: 2018130046

Experiment 1: Traditional Crypto Methods and Key Exchange

* OBJECTIVE

This experiment will be in two parts:

1. To implement Substitution, ROT 13, Transposition, Double Transposition, and Vernam Cipher in Scilab/C/Python/R. 2) Implement Diffie Hellman key exchange algorithm in Scilab/C/Python/R.
2. INTROUCTION TO CRYTO AND RELEVANT ALGORITHMS

Cryptography:

In cryptography, encryption is the process of transforming information (referred toas plaintext) using an algorithm (called cipher) to make it unreadable to anyone except thosepossessing special knowledge, usually referred to as a key. The result of the processis encrypted information (in cryptography, referred to as cipher text). In many contexts, theword encryption also implicitly refers to the reverse process, decryption (e.g. “software for encryption” can typically also perform decryption), to make the encrypted informationreadable again (i.e. to make it unencrypted). Encryption is used to protect data in transit, for example data being transferred via networks (e.g.the Internet, e-commerce), mobile telephones, wireless microphones, wireless intercom systems, Bluetooth devices and bank automatic teller machines. There have been numerous reports of data in transit being intercepted in recent years/ Encrypting data in transit also helps to secure it as it is often difficult to physically secure all access to networks

Substitution Technique:

In cryptography, a substitution cipher is a method of encryption by which units of plaintext are replaced with ciphertext according to a regular system; the "units" may be single letters (the most common), pairs of letters, triplets of letters, mixtures of the above, and so forth. The receiver deciphers the text by performing an inverse substitution.

There are a number of different types of substitution cipher. If the cipher operates on single letters, it is termed a simple substitution cipher; a cipher that operates on larger groups of letters is termed polygraphic. A monoalphabetic cipher uses fixed substitution over the entire message, whereas a polyalphabetic cipher uses a number of substitutions at different times in the message, where a unit from the plaintext is mapped to one of several possibilities in the ciphertext and vice-versa.

Transposition Technique:

In cryptography, a transposition cipher is a method of encryption by which the positions held by units of plaintext (which are commonly characters or groups of characters) are shifted according to a regular system, so that the ciphertext constitutes a permutation of the plaintext. That is, the order of the units is changed. Mathematically a bijective function is used on the characters' positions to encrypt and aninverse function to decrypt.

In a columnar transposition, the message is written out in rows of a fixed length, and then read out again column by column, and the columns are chosen in some scrambled order. Both the width of the rows and the permutation of the columns are usually defined by a keyword. For example, the word ZEBRAS is of length 6 (so the rows are of length 6), and thepermutation is defined by the alphabetical order of the letters in the keyword. In this case, theorder would be "6 3 2 4 1 5".

In a regular columnar transposition cipher, any spare spaces are filled with nulls; in anirregular columnar transposition cipher, the spaces are left blank. Finally, the message is readoff in columns, in the order specified by the keyword.

Double Transposition:

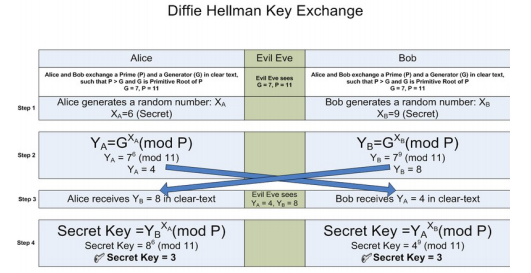
A single columnar transposition could be attacked by guessing possible columnlengths, writing the message out in its columns (but in the wrong order, as the key is not yetknown), and then looking for possible anagrams. Thus to make it stronger, a doubletransposition was often used. This is simply a columnar transposition applied twice. The samekey can be used for both transpositions, or two different keys can be used.

Vernam cipher:

In modern terminology, a Vernam cipher is a symmetrical stream cipher in which theplaintext is XORed with a random or pseudorandomstream of data (the "keystream") of thesame length to generate the ciphertext. If the keystream is truly random and used only once,this is effectively a one-time pad. Substituting pseudorandom data generated bya cryptographically secure pseudo-random number generator is a common and effectiveconstruction for a stream cipher.

Diffie –Hellman Key exchange algorithm:

The Diffie–Hellman key exchange method allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure communications channel. This key can then be used to encrypt subsequent communications using a symmetric key cipher. Although Diffie–Hellman key agreement itself is an anonymous (non-authenticated) key-agreement protocol, it provides the basis for a variety of authenticated protocols, and is used to provide perfect forward secrecy in Transport Layer Security's ephemeral modes (referred to as EDH or DHE depending on the cipher suite).



* LAB TASKS

Write a single program which fits all algorithms. YOU should generate output in following manner:

1. Select the Cryptography Method Provide Choice 1…5 for subjected crypto methods
   1. Substitution
      1. Your choice
      2. Enter Plain text to be encrypted
      3. Enter the no. of Position shift
      4. Encrypted Message
      5. Decrypted Message
   2. ROT 13
      1. Your choice
      2. Enter Plain text to be encrypted
      3. Encrypted Message
      4. Decrypted Message
   3. Transpose
      1. Your choice
      2. Enter Plain text to be encrypted
      3. Encrypted Message
      4. Decrypted Message
   4. Double Transposition
      1. Your choice
      2. Enter Plain text to be encrypted
      3. Encrypted Message
      4. Decrypted Message
   5. Vernam Cipher
      1. Your choice
      2. Enter Plain text to be encrypted
      3. Input Key
      4. Encrypted Message
      5. Decrypted Message

**CODE:**

import math

import numpy as np

def substitution(plainText):

    shift = int(input('Enter the no. of Position shift: '))

    encryptedText = ''

    for char in plainText:

        if(char.isupper()):

            encryptedText += chr((ord(char) + shift-65) % 26 + 65)

        else:

           encryptedText += chr((ord(char) + shift-97) % 26 + 97)

    print('Encrypted Text:',encryptedText)

    decryptedText = ''

    for char in encryptedText:

        if(char.isupper()):

            decryptedText += chr((ord(char) - shift-65) % 26 + 65)

        else:

           decryptedText += chr((ord(char) - shift-97) % 26 + 97)

    print('Decrypted Text:',decryptedText)

def rot13(plainText):

    encryptedText = ''

    for char in plainText:

        if(char.isupper()):

            encryptedText += chr((ord(char) + 13 - 65) % 26 + 65)

        else:

           encryptedText += chr((ord(char) + 13 - 97) % 26 + 97)

    print('Encrypted Text:',encryptedText)

    decryptedText = ''

    for char in encryptedText:

        if(char.isupper()):

            decryptedText += chr((ord(char) + 13 - 65) % 26 + 65)

        else:

           decryptedText += chr((ord(char) + 13 - 97) % 26 + 97)

    print('Decrypted Text:',decryptedText)

def transpose(plainText):

    key = input('Enter the key:')

    key.upper()

    order = sorted(list(key))

    col = len(key)

    ## Encryption

    msg\_len = len(plainText)

    msg\_list = list(plainText)

    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)

    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\_lst = list(encryptedText)

    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

    decryptedText = ''.join(''.join(x for x in y) for y in dec\_matrix)

    decryptedText = decryptedText[:-decryptedText.count('\_')]

    print('Decrypted Text:',decryptedText)

def double\_transposition(plainText):

    key = input('Enter the key:')

    key.upper()

    order = sorted(list(key))

    col = len(key)

    ## Encryption

    msg\_len = len(plainText)

    msg\_list = list(plainText)

    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])

    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('Encrypted Text:',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(x for x in y) for y in dec\_matrix)

    pointer = 0

    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

    decryptedText = decryptedText[:-decryptedText.count('\_')]

    decryptedText = ''.join(''.join(x for x in y) for y in dec\_matrix)

    decryptedText = decryptedText[:-decryptedText.count('\_')]

    print('Decrypted Text:',decryptedText)

def vernam\_cipher(plainText):

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

    while(len(key)!=len(plainText)):

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

    encryptedText = ''

    for i in range(len(plainText)):

        encryptedText += chr(((ord(plainText[i])-65)^(ord(key[i])-65))+65)

    print('Encrypted Text:',encryptedText)

    decryptedText = ''

    for i in range(len(encryptedText)):

        decryptedText += chr(((ord(encryptedText[i]) - 65)^(ord(key[i]) - 65)) + 65)

    print('Decrypted Text:',decryptedText)

def METHOD(i,plainText):

    switcher = {

        1: substitution,

        2: rot13,

        3: transpose,

        4: double\_transposition,

        5: vernam\_cipher

    }

    switcher[i](plainText)

options = """1. Substitution

2. ROT 13

3. Transpose

4. Double Transposition

5. Vernam Cipher"""

print(options)

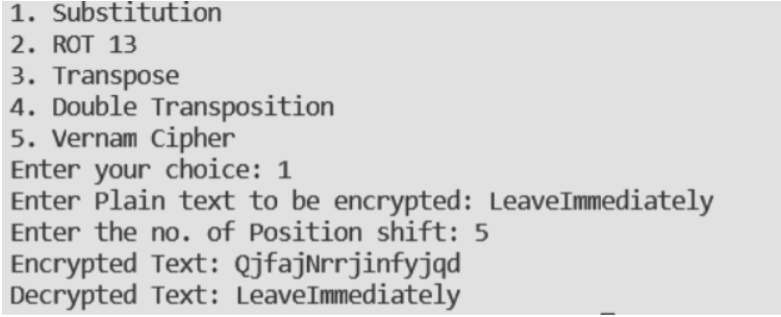
choice = int(input("Enter your choice: "))

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

METHOD(choice,plainText)

**OUTPUT:**

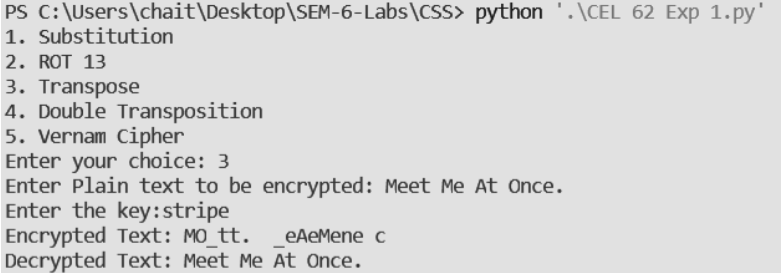
1. **Substitution**



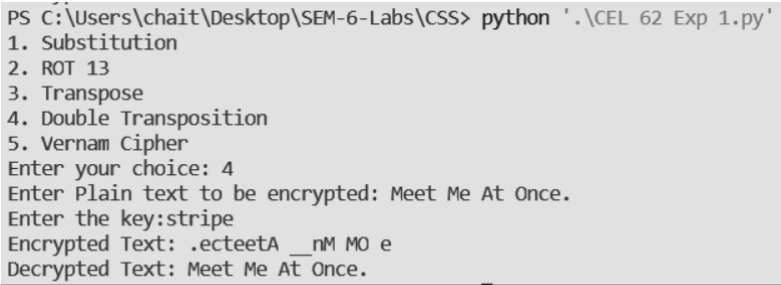
1. **ROT 13**



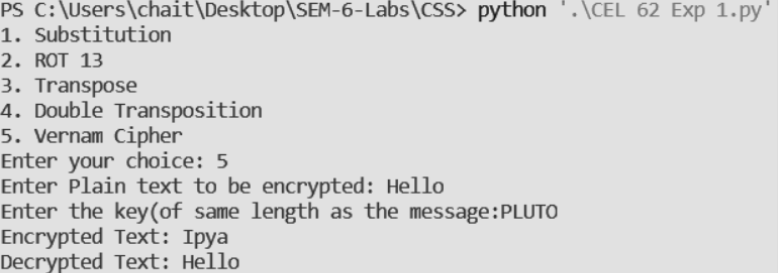
1. **Transpose**



1. **Double transposition**



1. **Vernam Cipher**



**Observations:**

1. Substitution cipher if a form a Ceaser cipher where the shift value can be other than 3.
2. Substitution cipher is very east to crack.
3. ROT-13 is also substitution cipher where shift value is fixed to 13. The code for encryption and decryption is same.
4. The drawback of vernam cipher is the key has to be as long as the message to be encrypted.