LABORATORY

CEL62: Cryptography and System Security
Winter 2021

Experiment 1:	**Traditional Crypto Methods and Key Exchange**

Note: Students are advised to read through this lab sheet before doing experiment. On-the-spot evaluation may be carried out during or at the end of the experiment. Your performance, teamwork/Personal effort, and learning attitude will count towards the marks.

Experiment 1: Traditional Crypto Methods and Key Exchange

1 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 to as plaintext) using an algorithm (called cipher) to make it unreadable to anyone except those possessing special knowledge, usually referred to as a key. The result of the process is encrypted information (in cryptography, referred to as cipher text). In many contexts, the word encryption also implicitly refers to the reverse process, decryption (e.g. “software for encryption” can typically also perform decryption), to make the encrypted information readable 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 an inverse 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

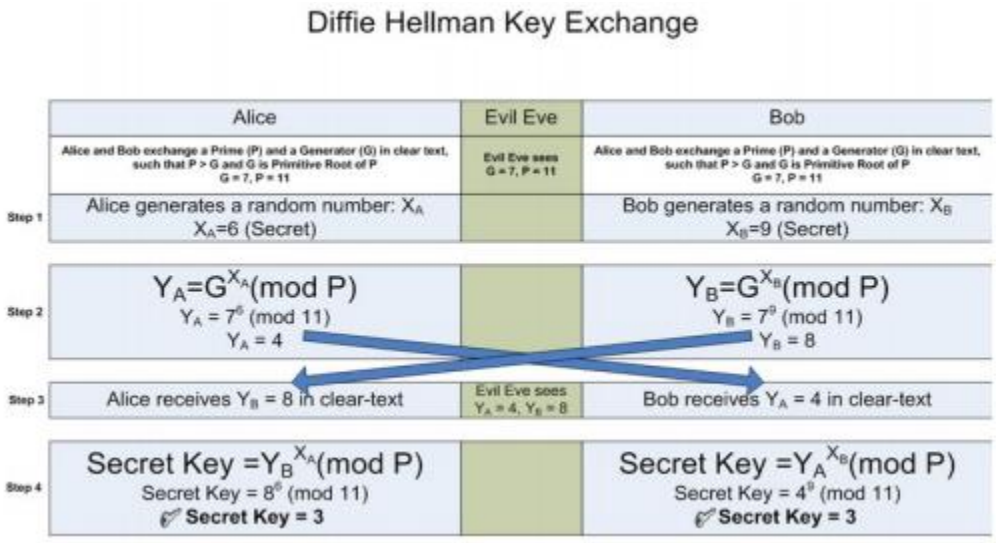
example, the word ZEBRAS is of length 6 (so the rows are of length 6), and the permutation is defined by the alphabetical order of the letters in the keyword. In this case, the order would be "6 3 2 4 1 5".

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

Double Transposition:
A single columnar transposition could be attacked by guessing possible column lengths, writing the message out in its columns (but in the wrong order, as the key is not yet known), and then looking for possible anagrams. Thus to make it stronger, a double transposition was often used. This is simply a columnar transposition applied twice. The same key 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 the plaintext is XORed with a random or pseudo random stream of data (the "keystream") of the same 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 by a cryptographically secure pseudo-random number generator is a common and effective construction 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).



3 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
    a. Substitution
        i. Your choice
        ii. Enter Plain text to be encrypted
        iii. Enter the no. of Position shift
        iv. Encrypted Message
        v. Decrypted Message
    b. ROT 13
        i. Your choice
        ii. Enter Plain text to be encrypted
        iii. Encrypted Message
        iv. Decrypted Message
    c. Transpose
        i. Your choice
        ii. Enter Plain text to be encrypted
        iii. Encrypted Message
        iv. Decrypted Message
    d. Double Transposition
        i. Your choice
        ii. Enter Plain text to be encrypted
        iii. Encrypted Message
        iv. Decrypted Message
    e. Vernam Cipher
        i. Your choice
        ii. Enter Plain text to be encrypted
        iii. Input Key
        iv. Encrypted Message
        v. Decrypted Message
    f. Diffie Hellman
        i. Enter the Prime Number g:
        ii. Enter second Prime Number n:
        iii. Enter the Secret x:
        iv. Enter the Secret y
        v. K_1:
        vi. K_2:

4 SUBMISSION

You need to submit a detailed lab report to describe what you have done and what you have observed as per the suggested output format for all method ; you also need to provide explanation to the observations that are interesting or surprising. In your report, you need to answer all the questions as per the suggested formant listed above.

CODE:

```
import string
import math
import random

low_letters=string.ascii_lowercase
up_letters=string.ascii_uppercase

def substitution_cipher(text,shift):

    punc = '''!()-[]{};:'"\, <>./?@#$%^&*_~'''

    for e in text:
        if e in punc or e==" ":
            text=text.replace(e,"")

    for i,ch in enumerate(text):
        if ch in low_letters:
            new_ch=low_letters[(low_letters.find(ch)+shift)%26]
            text=text[:i]+new_ch+text[i+1:]
        else:
            new_ch=up_letters[(up_letters.find(ch)+shift)%26]
            text=text[:i]+new_ch+text[i+1:]


    return text

def substitution_decipher(text,shift):
    for i,ch in enumerate(text):
        if ch in low_letters:
            new_ch=low_letters[(low_letters.find(ch)-shift)%26]
            text=text[:i]+new_ch+text[i+1:]
        else:
            new_ch=up_letters[(up_letters.find(ch)-shift)%26]
            text=text[:i]+new_ch+text[i+1:]
    return text

def Substitution(text):
    shift=int(input("Enter the position of shift:"))
    encrypted_text=substitution_cipher(text, shift)
    decrypted_text=substitution_decipher(encrypted_text, shift)

    return(encrypted_text,decrypted_text)
```

```
def ROT13(text):
    encrypted_text=substitution_cipher(text, 13)
    decrypted_text=substitution_decipher(encrypted_text, 13)

    return(encrypted_text,decrypted_text)




def tranposition_cipher(text,key): #spare blank spaces have been
replaced by '_'

    sorted_key=sorted(list(key))

    text=list(text)

    cols=len(key)
    rows=int(math.ceil(len(text)/cols))

    text=text+list("_"*(rows*cols - len(text) )) #filling spare
blank spaces

    mat=[list(text[i:i+cols]) for i in range(0,len(text),cols)]
    mat_t= [ [mat[j][i] for j in range(rows)] for i in range(cols)
]

    k=0
    encrypted_text=""

    for _ in range(cols):
        curr_index=key.index(sorted_key[k])
        encrypted_text+= "".join(mat_t[curr_index])
        k+=1

    return(encrypted_text)


def transposition_decipher(encrypted_text,key):

    encrypted_text=list(encrypted_text)
```

```
    sorted_key=list(sorted(key))
    cols=len(key)
    rows=int(math.ceil(len(encrypted_text)/cols))

    d=dict()
    for i,ch in enumerate(sorted_key):
        d[ch]=i


    mat=[     list(encrypted_text[i:i+rows])     for     i     in
range(0,len(encrypted_text),rows)]

    disp_mat = []

    for ch in key:
        disp_mat.append(mat[d[ch]])

    mat_t=  [  [disp_mat[j][i]  for  j  in  range(cols)]  for  i  in
range(rows) ]


    decrypted_text=""
    for i in range(len(mat_t)):
        decrypted_text+="".join(mat_t[i])

    return(decrypted_text)


def Transpose(text):
    key="".join(random.sample(up_letters,5))
    print("Key used for encryption:",key)
    encrypted_text=tranposition_cipher(text,key)
    decrypted_text=transposition_decipher(encrypted_text, key)
    decrypted_text=decrypted_text.replace("_","")

    return(encrypted_text,decrypted_text)




def double_transposition_cipher(text,keys):

    encrypted_text=text
```

```
    for i in range(2):
        encrypted_text=tranposition_cipher(encrypted_text,
keys[i])

    return(encrypted_text)


def double_transposition_decipher(encrypted_text,keys):

    decrypted_text=encrypted_text

    for i in range(1,-1,-1):
        decrypted_text=transposition_decipher(decrypted_text,
keys[i])


    return(decrypted_text.replace("_", ""))


def Double_Transpose(text):

    keys=["".join(random.sample(up_letters,5)) for i in range(2)]
    print("\nKeys          used          for          encrypytion
are:{},{}".format(keys[0],keys[1]))

    encrypted_text=double_transposition_cipher(text, keys)
    decrypted_text=double_transposition_decipher(encrypted_text,
keys)

    return(encrypted_text,decrypted_text)




def verman_cipher(text):
    letters=low_letters+up_letters

    key = ''.join(random.sample(letters,len(text)))
    print("Key used for encryption:",key)

    encrypted_text="".join([chr(ord(c1) ^ ord(c2)) for (c1,c2) in
zip(text,key)])

    decrpyted_text="".join([chr(ord(c1) ^ ord(c2)) for (c1,c2) in
zip(encrypted_text,key)])
```

```
    return(encrypted_text,decrpyted_text)



def diffie_hellman(g,n,x,y):
    generated_key_a= int(pow(g,x,n))
    generated_key_b= int(pow(g,y,n))

    secret_key_a= int(pow(generated_key_b,x,n))
    secret_key_b= int(pow(generated_key_a,y,n))

    print("K1:",secret_key_a)
    print("K2:",secret_key_b)



def main():

    switcher={"A":"Substitution",
              "B":"ROT13",
              "C":"Transpose",
              "D":"Double Tranpose",
              "E":"Verman Cipher",
              "F":"Diffie Hellman"
              }

    print('\n'.join("{}:   {}".format(k,   v)   for   k,   v   in
switcher.items()))

    choice=input("Enter your choice:")

    encrypted_text,decrpyted_text="",""

    if choice=="A":
        plain_text=input("Enter the plain text to be encrypted:")
        encrypted_text,decrypted_text=Substitution(plain_text)
    elif choice=="B":
        plain_text=input("Enter the plain text to be encrypted:")
        encrypted_text,decrypted_text=ROT13(plain_text)
    elif choice=="C":
        plain_text=input("Enter the plain text to be encrypted:")
        encrypted_text,decrypted_text=Transpose(plain_text)
    elif choice=="D":
        plain_text=input("Enter the plain text to be encrypted:")
```

```
encrypted_text,decrypted_text=Double_Transpose(plain_text)
    elif choice=="E":
        plain_text=input("Enter the plain text to be encrypted:")
        encrypted_text,decrypted_text=verman_cipher(plain_text)

    else:
        g=int(input("Enter a prime no. g:"))
        n=int(input("Enter a second prime no. n:"))
        x=int(input("Enter secret key x:"))
        y=int(input("Enter secret key y:"))
        diffie_hellman(g, n, x, y)


    if encrypted_text!="":
        print("Encrypted                    message:{}\nDecrypted
message:{}".format(encrypted_text,decrypted_text))

if __name__ == "__main__":
    main()
```

OUTPUT:

a. Substitution Cipher:
   Output:

```
C:\Users\Rucha Nargunde\Subjects\CSS>python exp1.py
A: Substitution
B: ROT13
C: Transpose
D: Double Tranpose
E: Verman Cipher
F: Diffie Hellman

Enter your choice:A
Enter the plain text to be encrypted:Give me lemon I will give you lemonade
Enter the position of shift:5

Encrypted message:LnajrjqjrtsNbnqqlnajdtzqjrtsfij
Decrypted message:GivemelemonIwillgiveyoulemonade
```

b. ROT 13
Output:

```
C:\Users\Rucha Nargunde\Subjects\CSS>python exp1.py
A: Substitution
B: ROT13
C: Transpose
D: Double Tranpose
E: Verman Cipher
F: Diffie Hellman

Enter your choice:B
Enter the plain text to be encrypted:Give me lemon I will give you lemonade

Encrypted message:TvirzryrzbaVjvyytvirlbhyrzbanqr
Decrypted message:GivemelemonIwillgiveyoulemonade
```

c. Transpose
Output:

```
C:\Users\Rucha Nargunde\Subjects\CSS>python exp1.py

A: Substitution
B: ROT13
C: Transpose
D: Double Tranpose
E: Verman Cipher
F: Diffie Hellman

Enter your choice:C
Enter the plain text to be encrypted:I am Lord Voldemort

Key used for encryption: NSVFI
Encrypted message:mddt  e_ILVm oooarlr
Decrypted message:I am Lord Voldemort
```

d. Double Transpose
Output:

```
C:\Users\Rucha Nargunde\Subjects\CSS>python exp1.py
A: Substitution
B: ROT13
C: Transpose
D: Double Tranpose
E: Verman Cipher
F: Diffie Hellman

Enter your choice:D
Enter the plain text to be encrypted:I am Lord Voldemort

Keys used for encrypytion are:GKNBE,ARJXV
Encrypted message:m Vod_ rdema LortIol
Decrypted message:I am Lord Voldemort
```

e. Verman Cipher
Output:

```
C:\Users\Rucha Nargunde\Subjects\CSS>python exp1.py
A: Substitution
B: ROT13
C: Transpose
D: Double Tranpose
E: Verman Cipher
F: Diffie Hellman

Enter your choice:E
Enter the plain text to be encrypted:Hello how are you

Key used for encryption: bJZBAnGpSzauXPlTF
Encrypted message:*/6..N/$Z =p   ;3
Decrypted message:Hello how are you
```

f. Diffie-Hellman
Output:

```
C:\Users\Rucha Nargunde\Subjects\CSS>python exp1.py

A: Substitution
B: ROT13
C: Transpose
D: Double Tranpose
E: Verman Cipher
F: Diffie Hellman

Enter your choice:F

Enter a prime no. g:13
Enter a second prime no. n:23
Enter secret key x:10
Enter secret key y:34

K1: 16
K2: 16
```

OBSERVATIONS:

1. ROT-13 is a special type of substitution cypher where the value of shift is always 13.
2. Verman cipher, also known has One-pad Cipher, uses the same key for encryption and decryption.
3. It uses XOR operation for encryption. And is considered to be one of the most secure encryption techniques.
4. Due to XOR operation in Verman Cipher, the characters in the encrypted text can fall out of the ASCII range for alphabets and hence the encrypted text may contain special characters.
5. The length of the key used for XOR should equal to the length of the plain-text.

CONCLUSION:

1. Through this experiment, I learned the various encryption techniques used in cryptography such as Substitution,Transposition and Verman Cipher.
2. I also learned about the Diffie-Hellman key exchange algorithm used for securely exchanging cryptographic keys.