Name: Namrata Bhorade
TE COMPS
Batch A
UID: 2018130004
Roll No.: 06

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.

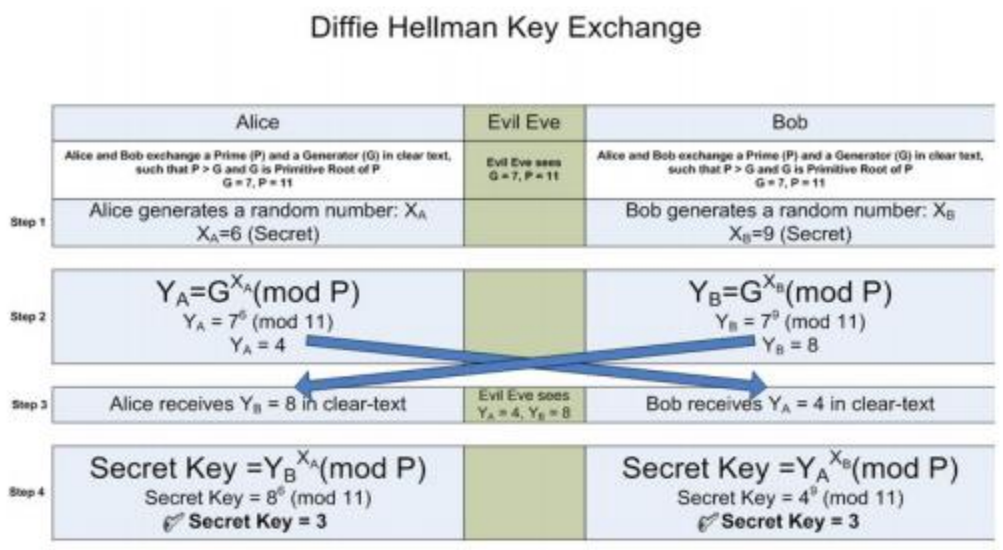
In a columnar transposition, the message is written out in rows of a fixed length, and then read out again column by column, and the columns are chosen in some scrambled order. Both the width of the rows and the permutation of the columns are usually defined by a keyword. For example, the word ZEBRAS 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).


Diffie Hellman Key Exchange
Alice
Evil Eve
Bob
Alice and Bob exchange a Prime (P) and a Generator (G) in clear text, such that P > G and G is Primitive Root of P
G = 7, P = 11
Evil Eve sees G = 7, P = 11
Alice and Bob exchange a Prime (P) and a Generator (G) in clear text, such that P > G and G is Primitive Root of P
G = 7, P = 11
Step 1
Alice generates a random number: X_A
X_A=6 (Secret)
Bob generates a random number: X_B
X_B=9 (Secret)
Step 2
$Y_A=G^{X_A}(\text{mod } P)$
$Y_A = 7^6 (\text{mod } 11)$
$Y_A = 4$
$Y_B=G^{X_B}(\text{mod } P)$
$Y_B = 7^9 (\text{mod } 11)$
$Y_B = 8$
Step 3
Alice receives Y_B = 8 in clear-text
Evil Eve sees $Y_A = 4$, $Y_B = 8$
Bob receives Y_A = 4 in clear-text
Step 4
Secret Key $=Y_B^{X_A}(\text{mod } P)$
Secret Key = 8^6 (mod 11)
Secret Key = 3
Secret Key $=Y_A^{X_B}(\text{mod } P)$
Secret Key = 4^9 (mod 11)
Secret Key = 3

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 numpy as np

def substitution_encrypt(plain_text, position_shift):
    letters = string.ascii_letters
    encrypted_text = ''
    mapping = {}
    for i in range(len(letters)):
        if letters[i].isupper():
            mapping[letters[i]] = letters[26 + (i+position_shift)%26]
        else:
            mapping[letters[i]] = letters[(i+position_shift)%26]

    for c in plain_text:
        if c in letters:
            encrypted_text += mapping[c]
        else:
            encrypted_text += c
    return encrypted_text

def substitution_decrypt(encrypted_text, position_shift):
    letters = string.ascii_letters
    mapping_decrypt = {}
    for i in range(len(letters)):
        if letters[i].isupper():
            mapping_decrypt[letters[i]] = letters[26 + (i-position_shift)%26]
        else:
            mapping_decrypt[letters[i]] = letters[(i-position_shift)%26]

    decrypted_text =''

    for c in encrypted_text:
        if c in letters:
            decrypted_text += mapping_decrypt[c]
        else:
            decrypted_text += c
    return decrypted_text


def substitution():
    print("\n1. Your Choice: Substitution Method")
    letters = string.ascii_letters
    plain_text = input("\n2. Enter plain text to be encrypted: ")
    position_shift = int(input("\n3. Enter the no. of Position shift: "))

    encrypted_text = substitution_encrypt(plain_text, position_shift)
    print(f"\n4. Encrypted Message: {encrypted_text}")

    decrypted_text = substitution_decrypt(encrypted_text, position_shift)
    print(f"\n5. Decrypted Message: {decrypted_text}")

def rot13():
    print("\n1. Your Choice: ROT 13 Method")
    letters = string.ascii_letters
    plain_text = input("\n2. Enter plain text to be encrypted: ")
    position_shift = 13

    encrypted_text = substitution_encrypt(plain_text, position_shift)
    print(f"\n3. Encrypted Message: {encrypted_text}")

    decrypted_text = substitution_decrypt(encrypted_text, position_shift)
    print(f"\n4. Decrypted Message: {decrypted_text}")

def transpose_encrypt(plain_text, key):
    text_length = float(len(plain_text))
    sorted_key = sorted(list(key))
    col = len(key)
    row = int(math.ceil(text_length/col))
    encrypted_text =''
    plain_text += '~'*int(row*col - text_length)
    plain_text_arr = np.array(list(plain_text))
    mat_encrypt = plain_text_arr.reshape(row,col)
    for k in key:
        k_index = sorted_key.index(k)
        encrypted_text += ''.join(row[k_index] for row in mat_encrypt)

    return encrypted_text

def transpose_decrypt(encrypted_text, key):
    message_len = len(encrypted_text)
    sorted_key = sorted(list(key))
    col = len(key)
    row = int(math.ceil(message_len/col))
    mat_decrypt = []
    for _ in range(row):
        mat_decrypt += [[None] * col]
    text_index = 0
    for k in key:
        k_index = sorted_key.index(k)
        for r in range(row):
            mat_decrypt[r][k_index] = encrypted_text[text_index]
            text_index += 1

    decrypted_text = ''.join(sum(mat_decrypt, []))
    return decrypted_text


def transpose():
    print("\n1. Your Choice: Transpose ")
    plain_text = input("\n2. Enter plain text to be encrypted: ")
    key = 'PROBLEMATICS'

    encrypted_text = transpose_encrypt(plain_text, key)
    print(f"\n3. Encrypted Message: {encrypted_text}")

    decrypted_text = transpose_decrypt(encrypted_text, key)[:-
encrypted_text.count('~')]
    print(f"\n4. Decrypted Message: {decrypted_text}")


def doubleTranspose():
    print("\n1. Your Choice: Double Transposition ")
    plain_text = input("\n2. Enter plain text to be encrypted: ")
    key1 = 'CUSTOMIZABLE'
    key2 = 'DEMOGRAPHIC'

    encrypted_text = transpose_encrypt(plain_text, key1)
    encrypted_text = transpose_encrypt(encrypted_text, key1)
    print(f"\n3. Encrypted Message: {encrypted_text}")

    decrypted_text = transpose_decrypt(encrypted_text, key1)
    decrypted_text = transpose_decrypt(decrypted_text, key1)[:-
encrypted_text.count('~')]
    print(f"\n4. Decrypted Message: {decrypted_text}")


def vernamCipher():
    print("\n1. Your Choice: Vernam Cipher ")
    plain_text = input("\n2. Enter plain text to be encrypted: ")
    text_length = len(plain_text)
    key = input('\n3. Input Key: ')
    while(text_length != len(key)):
        key = input(f"The length of your input text is {text_length}.\nHence plea
se enter key of length {text_length}: ")
```

```
    encrypted_text = [chr(((ord(a)-65 ^ ord(b)-
65) % 26) + 65 ) for a,b in zip(plain_text, key)]
    encrypted_text = ''.join(encrypted_text)
    print(f"\n4. Encrypted Message: {encrypted_text}")

    decrypted_text = [chr(((ord(a)-65 ^ ord(b)-
65) % 26) + 65 ) for a,b in zip(encrypted_text, key)]
    decrypted_text = ''.join(decrypted_text)
    print(f"\n5. Decrypted Message: {decrypted_text}")


def diffieHellman():
    g = int(input("\n1. Enter the Prime Number g: "))
    n = int(input("\n2. Enter second Prime Number n: "))
    x = int(input("\n3. Enter the Secret x: "))
    y = int(input("\n4. Enter the Secret y: "))

    x1 = pow(g,x,n)
    #print(f"x1: {x1}")

    y1 = pow(g,y,n)
    #print(f"y1: {y1}")

    kx = pow(y1,x,n)
    print(f"\nK1: {kx}")

    ky = pow(x1,y,n)
    print(f"\nK2: {ky}")


def default():
    print("\nInvalid input")


def function(i):
    switcher = {
        1: substitution,
        2: rot13,
        3: transpose,
        4: doubleTranspose,
        5: vernamCipher,
        6: diffieHellman

    }
    return switcher.get(i, default)()

next = 1
```

```python
while(next):
    choice = int(input("\n1: Substitution Method\n2: ROT 13 Method\n3: Transpose\
n4: Double Transposition\n5: Vernam Cipher\n6: Diffie Hellman \nEnter your choice
: "))
    function(choice)
    next = int(input("\nEnter 1 to continue or 0 to exit: "))
```

Output:

1. Substitution method

```
C:\Users\Namrata Bhorade\Desktop\TE COMPS\CSS\Lab\Exp1>python crypto.py

1: Substitution Method
2: ROT 13 Method
3: Transpose
4: Double Transposition
5: Vernam Cipher
6: Diffie Hellman
Enter your choice: 1

1. Your Choice: Substitution Method

2. Enter plain text to be encrypted: The attack is tomorrow

3. Enter the no. of Position shift: 6

4. Encrypted Message: Znk gzzgiq oy zusuxxuc

5. Decrypted Message: The attack is tomorrow

Enter 1 to continue or 0 to exit: 1
```

2. ROT13

```
1: Substitution Method
2: ROT 13 Method
3: Transpose
4: Double Transposition
5: Vernam Cipher
6: Diffie Hellman
Enter your choice: 2

1. Your Choice: ROT 13 Method

2. Enter plain text to be encrypted: Schedule the transfer

3. Encrypted Message: Fpurqhyr gur genafsre

4. Decrypted Message: Schedule the transfer

Enter 1 to continue or 0 to exit: 1
```

3. Transpose

```
1: Substitution Method
2: ROT 13 Method
3: Transpose
4: Double Transposition
5: Vernam Cipher
6: Diffie Hellman
Enter your choice: 3

1. Your Choice: Transpose

2. Enter plain text to be encrypted: The result is out

3. Encrypted Message: l~t~u~h e~ us~Tsi~rteo ~

4. Decrypted Message: The result is out

Enter 1 to continue or 0 to exit: 1
```

4. Double Transposition

```
1: Substitution Method
2: ROT 13 Method
3: Transpose
4: Double Transposition
5: Vernam Cipher
6: Diffie Hellman
Enter your choice: 4

1. Your Choice: Double Transposition

2. Enter plain text to be encrypted: The attack is tomorrow

3. Encrypted Message:  it atrow khcTroeatmos~~

4. Decrypted Message: The attack is tomorrow

Enter 1 to continue or 0 to exit: 1
```

5. Vernam Cipher

```
1: Substitution Method
2: ROT 13 Method
3: Transpose
4: Double Transposition
5: Vernam Cipher
6: Diffie Hellman
Enter your choice: 5

1. Your Choice: Vernam Cipher

2. Enter plain text to be encrypted: ATTACKER

3. Input Key: QWERTYUI

4. Encrypted Message: QFXRRSQZ

5. Decrypted Message: ATTACKER
```

```
Enter 1 to continue or 0 to exit: 1
```

6. Diffie Hellman

```
1: Substitution Method
2: ROT 13 Method
3: Transpose
4: Double Transposition
5: Vernam Cipher
6: Diffie Hellman
Enter your choice: 6

1. Enter the Prime Number g: 7

2. Enter second Prime Number n: 11

3. Enter the Secret x: 6

4. Enter the Secret y: 9

K1: 3

K2: 3

Enter 1 to continue or 0 to exit: 0
```

Observations:

1. For substitution method, the possible values of position shift are between 0 to 25. The values from 26 onwards will have similar effect as those for 0 to 25.

```
C:\Users\Namrata Bhorade\Desktop\TE COMPS\CSS\Lab\Exp1>python crypto.py

1: Substitution Method
2: ROT 13 Method
3: Transpose
4: Double Transposition
5: Vernam Cipher
6: Diffie Hellman
Enter your choice: 1

1. Your Choice: Substitution Method

2. Enter plain text to be encrypted: The package has arrived

3. Enter the no. of Position shift: 3

4. Encrypted Message: Wkh sdfndjh kdv duulyhg

5. Decrypted Message: The package has arrived

Enter 1 to continue or 0 to exit: 1

1: Substitution Method
2: ROT 13 Method
3: Transpose
4: Double Transposition
5: Vernam Cipher
6: Diffie Hellman
Enter your choice: 1

1. Your Choice: Substitution Method

2. Enter plain text to be encrypted: The package has arrived

3. Enter the no. of Position shift: 29

4. Encrypted Message: Wkh sdfndjh kdv duulyhg

5. Decrypted Message: The package has arrived
```

```
Enter 1 to continue or 0 to exit: 0
```

2. ROT13 cipher is the special case of substitution cipher where position shift value is 13. Performing ROT13 encryption two times gives us the original text back.

```
C:\Users\Namrata Bhorade\Desktop\TE COMPS\CSS\Lab\Exp1>python crypto.py

1: Substitution Method
2: ROT 13 Method
3: Transpose
4: Double Transposition
5: Vernam Cipher
6: Diffie Hellman
Enter your choice: 2

1. Your Choice: ROT 13 Method

2. Enter plain text to be encrypted: Dispatch the items

3. Encrypted Message: Qvfcngpu gur vgrzf

4. Decrypted Message: Dispatch the items

Enter 1 to continue or 0 to exit: 1

1: Substitution Method
2: ROT 13 Method
3: Transpose
4: Double Transposition
5: Vernam Cipher
6: Diffie Hellman
Enter your choice: 2

1. Your Choice: ROT 13 Method

2. Enter plain text to be encrypted: Qvfcngpu gur vgrzf

3. Encrypted Message: Dispatch the items

4. Decrypted Message: Qvfcngpu gur vgrzf

Enter 1 to continue or 0 to exit: 0
```

3. In Transposition cipher the key used should have all of its letters unique so that the number of columns can be equal to the size of key.

4. The keys used for two steps of Double Transposition should be different to get effective encryption. It is difficult to identify the original text from encrypted text because of its complex nature.

5. The procedure for decryption in Vernam Cipher is same as encryption because of the XOR operation used. The randomly generated key will give effective encryption for this method.

Conclusion:

1. In this experiment, I learned about various traditional encryption techniques and implemented them in Python.
2. The encryption provided by Substitution and ROT13 cipher is easy to crack and hence less secure.
3. The transposition and double transposition cipher are complex encryption methods which are difficult to crack and they provide efficient encryption.
4. The encryption provided by Vernam Cipher is the most efficient as it is very difficult to crack because of the key which is of equal length of that of the input text.