LABORATORY

CEL62: Cryptography and System Security
Winter 2021

Nikita Chitre
Batch A
TE Comps

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

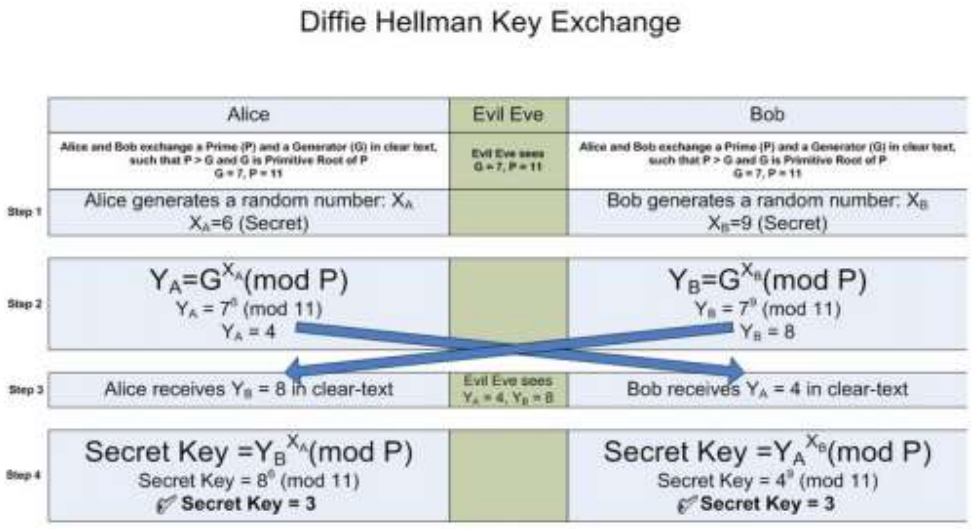
width of the rows and the permutation of the columns are usually defined by a keyword. For example, the word ZEBRAS is of length 6 (so the rows are of length 6), and the permutation is defined by the alphabetical order of the letters in the keyword. In this case, the order would be "6 3 2 4 1 5".

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:

CODE

```
import math

def substitution(text,key):
    #encrypt
    cipherText = ''
    for ch in text:
        if ch.isupper():
            cipherText += chr((ord(ch) - 65 + key) % 26 + 65)
        else:
            cipherText += chr((ord(ch) - 97 + key) % 26 + 97)

    #decrypt
    plainText = ''
    for ch in cipherText:
        if ch.isupper():
            plainText += chr((ord(ch) - 65 - key) % 26 + 65)
        else:
            plainText += chr((ord(ch) - 97 - key) % 26 + 97)
    print('\nYour choice: 1')
    print('Entered Plain text: '+text)
    print('Entered Shift Position: '+str(key))
    print('Encrypted message: '+cipherText)
    print('Decrypted message: '+plainText)

def rot13(text):
    #encrypt
    cipherText = ''
    for ch in text:
        if ch.isupper():
            cipherText += chr((ord(ch) - 65 + 13) % 26 + 65)
        else:
            cipherText += chr((ord(ch) - 97 + 13) % 26 + 97)
    #decrypt
    plainText = ''
    for ch in cipherText:
        if ch.isupper():
            plainText += chr((ord(ch) - 65 - 13) % 26 + 65)
        else:
            plainText += chr((ord(ch) - 97 - 13) % 26 + 97)
    print('\nYour Choice: 2')
    print('Entered Plain Text: '+text)
    print('Encrypted message: '+cipherText)
```

```
    print('Decrypted message: '+plainText)


def encrypt_transpose(text,key):
    key = [int(x) for x in str(key)]
    cipher = ''
    msg_len = len(text)
    text_list = list(text)
    cols = len(key)
    rows = math.ceil(msg_len/cols)
    pad = cols*rows - msg_len
    text_list.extend(' '*pad)

    #Writing text in matrix
    matrix = [text_list[idx:idx+cols] for idx in range(0,msg_len,cols)]

    #print cipher text
    for c in range(0,cols):
        cipher += ''.join([row[key[c]-1] for row in matrix])
    return cipher

#decrypt
def decrypt_transpose(text,key):
    key = [int(x) for x in str(key)]
    plain_text = ''
    msg_len = len(text)
    text_list = list(text)
    cols = len(key)
    rows = math.ceil(msg_len/cols)

    matrix = [[None]*cols for r in range(0,rows)]

    #Finding decryption key
    dec_key = [0] * (cols+1)
    for idx in range(0,cols):
        dec_key[key[idx]] = idx+1
    dec_key = dec_key[1:]

    #Writing the ciphertext in terms of matrix
    idx = [i for i in range(0,msg_len,rows)]
    for c in range(0,cols):
        i = idx[dec_key[c]-1]
        for r in range(0,rows):
            matrix[r][c] = text_list[i]
```

```
            i +=1
    #Reading the matrix
    plain_text = ''.join(sum(matrix,[]))
    return plain_text



def encrypt_double_transpose(text,key1,key2):
    cipher1 = encrypt_transpose(text,key1)
    cipher2 = encrypt_transpose(cipher1,key2)
    return cipher2

def decrypt_double_transpose(text,key1,key2):
    plain_text1 = decrypt_transpose(text,key2)
    plain_text2 = decrypt_transpose(plain_text1,key1)
    return plain_text2

def vernam(text,key):
    result = ''
    idx = 0
    for ch in text:
        result = result + chr(((ord(ch) -65 ^ ord(key[idx])-65) % 26) + 65 )
        idx +=1
    print(result)
    return result

choice = int(input('Enter your choice of encryption:\n Select 1 for Substitution.\n 2 for ROT13.\n
3 for Transposition.\n 4 for Double Transposition.\n 5 for Vernam.\n'))



if choice == 1:
    text = input('Enter plain text to be encrypted\n')
    key = int(input('Enter the no. of position shift\n'))
    substitution(text,key)

elif choice == 2:
    text = input('Enter plain text to be encrypted\n')
    rot13(text)

elif choice == 3:
    text = input('Enter plain text to be encrypted\n')
    key = int(input("Enter the key\n"))
    cipher = encrypt_transpose(text,key)
    plainText = decrypt_transpose(cipher,key)
```

```
    print('\nYour Choice: 3')
    print('Entered Plain Text: '+text)
    print('Encrypted Text: '+cipher)
    print('Decrypted Text: '+plainText)

elif choice == 4:
    text = input('Enter plain text to be encrypted\n')
    key1 = int(input("Enter the key1\n"))
    key2 = int(input("Enter the key2\n"))
    cipher = encrypt_double_transpose(text,key1,key2)
    plainText = decrypt_double_transpose(cipher,key1,key2)
    print('\nYour Choice: 4')
    print('Entered Plain Text: '+text)
    print('Encrypted Text: '+cipher)
    print('Decrypted Text: '+plainText)

elif choice ==5:
    text = input('Enter plain text to be encrypted\n')
    key = input("Enter the key\n")
    cipher = vernam(text,key)
    decipher = vernam(cipher,key)
    print('\nYour Choice: 5')
    print('Entered Plain Text: '+text)
    print('Entered key: '+key)
    print('Encrypted Text: '+cipher)
    print('Decrypted Text: '+decipher)
```

Output:-

Substitution Method:

```
Your choice: Substitution
Entered Plain text: EnemyAttacks
Entered Shift Position: 5
Encrypted message: JsjrdsFyyfhpx
Decrypted message: EnemyAttacks
```

Observations:-

- Every symbol in plain text is shifted by some fixed number.
- The plain text can easily be found from the ciphertext by using brute force approach.
- The encrypted values lies between 0 to 25 since mod operation is used.

ROT13 Method:

```
Your Choice: ROT13
Entered Plain Text: universeisamazing
Encrypted message: havirefrvfnznmvat
Decrypted message: universeisamazing
```

Observation:

- ROT13 cipher (read as – "rotate by 13 places") is a special case of the Shift cipher in which the shift is always 13.
- ROT13 cipher can be broken by just shifting the letters 13 places. Therefore it has no practical use.

Transposition Method:

```
Your Choice: Transposition
Entered Plain Text: enemyattackstonightz
Entered Key: 32154
Encrypted Text: etthntsgeakiycnzmaot
Decrypted Text: enemyattackstonightz
```

Observations :

- A transposition cipher doesn't change the characters in the plain-text when it generates the cipher-text, it just re-arranges them.
- The key should have unique values.

Double Transposition Method:

```
Your Choice: Double Transposition
Entered Plain Text: enemyattackstonightz
Entered Key1: 34125
Entered Key2: 23154
Encrypted Text: toiyttnceakgmaszhetn
Decrypted Text: enemyattackstonightz
```

Observations:

- Double Transposition applies same columnar transposition twice.
- Stronger than Transposition Method as we use two different keys and apply columnar transposition twice.

Vernam Method:

```
Your Choice: Vernam
Entered Plain Text: ATTACKER
Entered key: QWERTYUI
Encrypted Text: QFXRRSQZ
Decrypted Text: ATTACKER
```

Observations:

- The same XOR function is used for both encryption and decryption.
- For two different letters we get the same encrypted letter in this example.
- The Key length should be the same as plain text length.

Conclusion:-

1. From this experiment I learnt about various encryption methods.
2. Substitution cipher can be easily cracked using brute force method.
3. The strength of the algorithm increases from Substitution to Vernam.
4. ROT13 is special case of Substitution cipher.
5. Double Transposition is columnar transposition applied twice hence it is stronger than simple Transposition.