

NAME: SAKSHI PATIL
BATCH: C
UID: 2018130039

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.

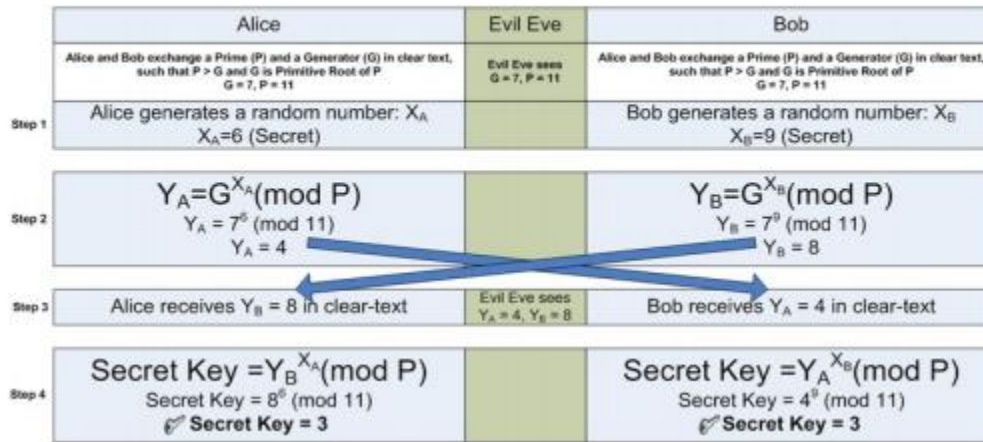
Vernam cipher:

In modern terminology, a Vernam cipher is a symmetrical stream cipher in which the plaintext is XORed with a random or pseudorandom 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 pseudorandom 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



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

Traditional Crypto Methods and Key exchange/PV

v. Decrypted Message

CODE:

```
import string,math

print("Choose the algorithm to encrypt and decrypt your text:")
print("1.Substitution\n2.ROT 13\n3.Transpose\n4.Double transposition\n5.Vernam Cipher")
choice = int(input())
def enc_CipherFunction(text,key):
    result=[]
    for char in text:
        if (char.isupper()):
            result.append(chr((ord(char)+key-65)%26+65))
        elif (char.islower()):
            result.append(chr((ord(char)+key-97)%26+97))
    result="".join(result)
    return result

def dec_CipherFunction(text,key):
    result=[]
    for char in text:
        if (char.isupper()):
            result.append(chr((ord(char)-key-65)%26+65))
        elif (char.islower()):
            result.append(chr((ord(char)-key-97)%26+97))
    result="".join(result)
    return result

def VernamCipherFunction(text, key):
    result = ""
    ptr = 0
    for char in text:
        result = result + chr(((ord(char)-65) ^ (ord(key[ptr])-65))+65)
        ptr = ptr + 1
        if ptr == len(key):
            ptr = 0
    return result

def encryptMessage(msg,key):
    cipher = ""
    k_indx = 0
    msg_len = float(len(msg))
    msg_lst = list(msg)
    key_lst = sorted(list(key))
    col = len(key)
    row = int(math.ceil(msg_len / col))
    fill_null = int((row * col) - msg_len)
```

Traditional Crypto Methods and Key exchange/PV

```

msg_lst.extend('_ ' * fill_null)
matrix = [msg_lst[i: i + col]
           for i in range(0, len(msg_lst), col)]
for _ in range(col):
    curr_idx = key.index(key_lst[k_indx])
    cipher += ".join([row[curr_idx]
                     for row in matrix])
    k_indx += 1

return cipher

def decryptMessage(cipher,key):
    msg = ""
    k_indx = 0
    msg_indx = 0
    msg_len = float(len(cipher))
    msg_lst = list(cipher)
    col = len(key)
    row = int(math.ceil(msg_len / col))
    key_lst = sorted(list(key))
    dec_cipher = []
    for _ in range(row):
        dec_cipher += [[None] * col]
    for _ in range(col):
        curr_idx = key.index(key_lst[k_indx])

        for j in range(row):
            dec_cipher[j][curr_idx] = msg_lst[msg_indx]
            msg_indx += 1
        k_indx += 1
    try:
        msg = ".join(sum(dec_cipher, []))
    except TypeError:
        raise TypeError("This program cannot",
                        "handle repeating words.")

    return msg

if choice==1:
    plain_text=input("Enter the plain text to be encrypted:\n")
    key=int(input("Enter the no. of Position shift:\n"))
    cipher_text=enc_CipherFunction(plain_text,key)
    print("Encrypted message is :",cipher_text)
    decipher_text=dec_CipherFunction(cipher_text,key)
    print("Decrypted message is :",decipher_text)

elif choice==2:
    plain_text=input("Enter the plain text to be encrypted:\n")

```

Traditional Crypto Methods and Key exchange/PV

```

key=13
cipher_text=enc_CipherFunction(plain_text,key)
print("Encrypted message is :",cipher_text)
decipher_text=dec_CipherFunction(cipher_text,key)
print("Decrypted message is :",decipher_text)

elif choice==3:
    plain_text=input("Enter the plain text to be encrypted:\n")
    input_key=input("Enter the input key :\n")
    cipher_text = encryptMessage(plain_text,input_key)
    print("Encrypted Message is:",cipher_text)
    decipher_text = decryptMessage(cipher_text,input_key)
    null_count = decipher_text.count('_')
    if null_count > 0:
        print("Decrypted Message is:",decipher_text[:-null_count])

elif choice==4:
    plain_text=input("Enter the plain text to be encrypted:\n")
    input_key=input("Enter the input key :\n")
    cipher_text = encryptMessage(plain_text,input_key)
    double_cipher_text = encryptMessage(cipher_text,input_key)
    print("Encrypted Message after double transposition is:",double_cipher_text)
    decipher_text=decryptMessage(double_cipher_text,input_key)
    double_decipher_text=decryptMessage(decipher_text,input_key)
    null_count = double_decipher_text.count('_')
    if null_count > 0:
        print("Decrypted Message after double transposition is:",double_decipher_text[:-
null_count])

elif choice==5:
    plain_text=input("Enter the plain text to be encrypted:\n")
    input_key=input("Enter the input key :\n")
    if len(plain_text)!= len(input_key):
        print("length of input key and plain text should be same")
    plain_text=plain_text.upper()
    input_key=input_key.upper()
    cipher_text=VernamCipherFunction(plain_text,input_key)
    print("Encrypted message is :",cipher_text)
    decipher_text=VernamCipherFunction(cipher_text,input_key)
    print("Decrypted message is :",decipher_text)

else:
    print("Select valid number")

```

OUTPUT:

a. Substitution

Traditional Crypto Methods and Key exchange/PV

Choose the algorithm to encrypt and decrypt your text:

- 1.Substitution
- 2.ROT 13
- 3.Transpose
- 4.Double transposition
- 5.Vernam Cipher

1

Enter the plain text to be encrypted:

Hello World

Enter the no. of Position shift:

5

Encrypted message is : MjqqtBtwqi

Decrypted message is : HelloWorld

b. ROT 13

Choose the algorithm to encrypt and decrypt your text:

- 1.Substitution
- 2.ROT 13
- 3.Transpose
- 4.Double transposition
- 5.Vernam Cipher

2

Enter the plain text to be encrypted:

Hello World

Encrypted message is : UryybJbeyq

Decrypted message is : HelloWorld

c. Transpose

Choose the algorithm to encrypt and decrypt your text:

- 1.Substitution
- 2.ROT 13
- 3.Transpose
- 4.Double transposition
- 5.Vernam Cipher

3

Enter the plain text to be encrypted:

Hello World

Enter the input key :

HACK

Encrypted Message is: e llWdHorlo_

Decrypted Message is: Hello World

d. Double transposition

```

Choose the algorithm to encrypt and decrypt your text:
1.Substitution
2.ROT 13
3.Transpose
4.Double transposition
5.Vernam Cipher
4
Enter the plain text to be encrypted:
Hello World
Enter the input key :
HACK
Encrypted Message after double transposition is:  dllHoeWrlo_
Decrypted Message after double transposition is: Hello World_

```

e. Vernam Cipher

```

Choose the algorithm to encrypt and decrypt your text:
1.Substitution
2.ROT 13
3.Transpose
4.Double transposition
5.Vernam Cipher
5
Enter the plain text to be encrypted:
Hello World
Enter the input key :
Olleh dlrow
Encrypted message is : JPAPJAVFAFV
Decrypted message is : HELLO WORLD

```

OBSERVATIONS:

- Caesar cipher is the simplest substitution cipher by Julius Caesar. In this substitution technique, to encrypt the plain text, each alphabet of the plain text is shifted by the 'key' places. And to decrypt the cipher text each alphabet of cipher text is shifted by the alphabet 'key' places before it.
- ROT13 cipher(read as – “rotate by 13 places”) is a special case of the Ceaser cipher in which the shift is always 13. So every letter is shifted 13 places to encrypt or to decrypt the message.
- 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.
- Vernam Cipher is a method of encrypting alphabetic text. It is one of the Transposition techniques for converting a plain text into a cipher text. In this mechanism we assign a number to each character of the Plain-Text, like (a = 0, b = 1, c = 2, ... z = 25). In Vernam cipher algorithm, we take a key to encrypt the plain text which length should be equal to the length of the plain text.

CONCLUSION:

- Thus I understood implementation of various encrypting and decrypting techniques of a message in python.