CSS Experiment 2

Prerak Khandelwal
TE COMP B
Roll no 22

Aim:

To implement and design the product cipher using Substitution and Transposition ciphers.

Theory:

Caesar Cipher:

The Caesar cipher is the simplest and oldest method of cryptography. The Caesar cipher method is based on a mono-alphabetic cipher and is also called a shift cipher or additive cipher. Julius Caesar used the shift cipher (additive cipher) technique to communicate with his officers. For this reason, the shift cipher technique is called the Caesar cipher. The Caesar cipher is a kind of replacement (substitution) cipher, where all letter of plain text is replaced by another letter.

Let's take an example to understand the Caesar cipher, suppose we are shifting with 1, then A will be replaced by B, B will be replaced by C, C will be replaced by D, D will be replaced by C, and this process continues until the entire plain text is finished.

Caesar ciphers is a weak method of cryptography. It can be easily hacked. It means the message encrypted by this method can be easily decrypted.

The formula of encryption is:

En (x) = (x + n) mod 26

The formula of decryption is:

Dn (x) = (xi - n) mod 26

If any case (Dn) value becomes negative (-ve), in this case, we will add 26 in the negative value.
Where,
E denotes the encryption
D denotes the decryption
x denotes the letters value
n denotes the key value (shift value)

Example: Use the Caesar cipher to encrypt and decrypt the message "JAVATPOINT," and the key (shift) value of this message is 3.
Encryption
We apply encryption formulas by character, based on alphabetical order.
The formula of encryption is:
En (x) = (x + n) mod 26

Plaintext: J → 09	En: (09 + 3) mod 26	Ciphertext: 12 → M
Plaintext: A → 00	En: (00 + 3) mod 26	Ciphertext: 3 → D
Plaintext: V → 21	En: (21 + 3) mod 26	Ciphertext: 24 → Y
Plaintext: A → 00	En: (00 + 3) mod 26	Ciphertext: 3 → D
Plaintext: T → 19	En: (19 + 3) mod 26	Ciphertext: 22 → W
Plaintext: P → 15	En: (15 + 3) mod 26	Ciphertext: 18 → S
Plaintext: O → 14	En: (14 + 3) mod 26	Ciphertext: 17 → R
Plaintext: I → 08	En: (08 + 3) mod 26	Ciphertext: 11 → L
Plaintext: N → 13	En: (13 + 3) mod 26	Ciphertext: 16 → Q
Plaintext: T → 19	En: (19 + 3) mod 26	Ciphertext: 22 → W

The encrypted message is "MDYDWSRLQW". Note that the Caesar cipher is monoalphabetic, so the same plaintext letters are encrypted as the same letters. For example, "JAVATPOINT" has "A", encrypted by "D".

Transposition Cipher:

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 cipher text constitutes a permutation of the plaintext. That is, the order of the units is changed (the plaintext is reordered). Mathematically a bi-jective function is used on the characters' positions to encrypt and an inverse function to decrypt.

Keyed Transposition Cipher:

Keyed transposition cipher uses keys to encrypt and decrypt the messages. It shares the same secret key among the senders and the receivers. Key is used as position finder for the cipher text. We can illustrate this by giving a nice example as-

Suppose given plaintext is- "prob hatd euri goth erek". And lets the key is=3201. Then the cipher text will be-(Here I assumed starting from 0)

```
Plain Text:   prob hatd euri goth erek
Key:          3201 3201 3201 3201 3201

Cipher Text:  obrp tdah riue thog ekre
Positions:    0123 0123 0123 0123 0123
Key:          3201 3201 3201 3201 3201

Plain Text:   prob hatd euri goth erek
```

For encryption:

Step1: Take the inputs, plaintext and key.
Step2: Calculate the length of the key.
Step3: Divide the plaintext into multiple numbers of words equal to the key length.
Step4: Use the key to encrypt each word.
a. Until the end of a word put each character in its corresponding key position.
b. Do step 4.a to the end of the plaintext.
Step5: Display the encrypted text.

For decryption:

Step1: take the inputs, cipher text and key.
Step2: Calculate length of the key.
Step3: Divide the plaintext into multiple numbers of words equal to the key length.
Step4: Use the key and positions to decrypt each word.
Step5: Display the decrypted message.

Program for Caesar Cipher:

```
# encryption
def encrypt(text,s):
    result = ""
    for i in range(len(text)):
        c = text[i]

        if c==" ":
            result += " "
            continue
        # if uppercase
        if(c.isupper()):
            result += chr(((ord(c) + s - ord("A")) % 26) + ord("A"))

        # if lowercase
        else:
            result += chr(((ord(c) + s - ord("a")) % 26) + ord("a"))
    return result

# decryption
def decrypt(text,s):
    result = ""
    for i in range(len(text)):
        c = text[i]

        if c==" ":
            result += " "
            continue
```

```
        # if uppercase
        if(c.isupper()):
            result += chr(((ord(c) - s - ord("A") + 26) % 26) + ord("A"))

        # if lowercase
        else:
            result += chr(((ord(c) - s - ord("a") + 26) % 26) + ord("a"))
    return result

text = input("Enter Plain Text: ")
s = int(input("Enter key value: "))
cipher = encrypt(text,s)
print("Cipher Text: ", cipher)
print("Recovered plain Text: ", decrypt(cipher,s))
```

Output for Caesar Cipher:

```
PS D:\TE\SEM_6\CSS\CODES> python -u "d:\TE\SEM_6\CSS\CODES\caesarcipher.py"
Enter Plain Text: PRERAKKHANDELWAL
Enter key value: 4
Cipher Text:  TVIVEOOLERHIPAEP
Recovered plain Text:  PRERAKKHANDELWAL
```

Program for Keyed Transposition Cipher:

```
def keyed(s,b,k):
    inter=[]
    for i in s:
        inter.append(i)
    inter1=[]
    while(len(inter)!=0):
        l=[]
        while(len(l)!=b):
            if(len(inter)==0):
                l.append(" ")
            else:
                l.append(inter.pop(0))
        inter1.append(l)
    inter2=[]

    for j in inter1:
        inter2.append("".join(j))
    ans=[]

    for kl in inter2:
        ans1=[]
        leng=len(kl)
```

```
        tempk=k[:leng+1]
        for h in tempk:
            ans1.append(kl[h-1])
        if " " in ans1:
            ans1.remove(" ")
        ans.append("".join(ans1))
    en="".join(ans)
    print(en)
    inter=[]
    for i in en:
        inter.append(i)
    inter1=[]
    while(len(inter)!=0):
        l=[]
        while(len(l)!=b):
            if(len(inter)==0):
                l.append(" ")
            else:
                l.append(inter.pop(0))
        inter1.append(l)
    inter2=[]

    for j in inter1:
        inter2.append("".join(j))
    ans=[]
    for kl in inter2:
        d={}
        ans1=[]
        for i in range(len(k)):
            d[k[i]]=kl[i]
        for i1 in range(len(k)):
            if(d[i1+1]==" "):
                continue
            else:
                ans1.append(d[i1+1])
        ans.append("".join(ans1))
    dn="".join(ans)
    print ("---DECRYPTED TEXT IS---")
    print(dn)

print("--- KEYED TRANSPOSITION CIPHER   ")
inp=input("ENTER A STRING:")
b=int(input("ENTER BLOCK SIZE:"))
k=input("ENTER KEY ARRAY SEPARATED BY COMMA:").split(",")
kg=list(map(int,k))
print("-- ENCRYPTED TEXT IS ")
keyed(inp,b,kg)
```

Output for Keyed Transposition Cipher:

```
                    > python -u "d:\TE\SEM_6\CSS\CODES\transposition.py"
--- KEYED TRANSPOSITION CIPHER
ENTER A STRING:PRERAKKHANDELWAL
ENTER BLOCK SIZE:5
ENTER KEY ARRAY SEPARATED BY COMMA:3,1,4,5,2
-- ENCRYPTED TEXT IS
EPRARHKANKLDWAEL
---DECRYPTED TEXT IS---
PRERAKKHANDELWAL
```

Conclusion:

Thus, we have seen how Caesar Cipher is used to substitute characters by their equivalents using key and it is one of the easiest to implement. However, there are chances that eve can decrypt the message easily thus making it vulnerable. Transposition cipher on other hand, uses some complicated steps to relocate characters with plain text and making the cipher text hard to be decrypted by an eve who doesn't have the real key.