Transposition Cipher

Project 2: Encryption & decryption using transposition of the kind discussed in class. Then develop the software to launch a brute-force attack to discover the key. Here assume that the key length is known to be 8 or less.

Solution:

Encryption

```
def encryptMessage(plaintext, key):
    matrix = []
    len_pt = len(plaintext)
    len_key = len(key)
    C = len_key
    R = int(len_pt / len_key)
    rem = len_pt % len_key
    if(rem > 0 ):
        R = R + 1
    i = C - rem
    while(i!=0):
        plaintext = plaintext + '-'
        i = i-1
    temp = 0
    for i in range(R):
        a =[]
        for j in range(C):
            a.append(plaintext[temp])
            temp = temp + 1
        matrix.append(a)
    cipher = ""
    key2 = []
    k = 1
    for i in range(len(key)):
        for j in range(len(key)):
            if key[j]==str(k):
                key2.append(j)
                k=k+1
    for j in key2:
        for i in range(R):
            cipher = cipher + matrix[i][j]
    return cipher
```

To encrypt it, the recipient has to work out the column lengths by dividing the message length by the key length. If the remainder is not zero then number of padding bits are added and if it is zero then covert plaintext into matrix and write it column by column in increasing order of key.

Any spare spaces are filled with nulls or left blank or placed by a character (Example: -).

Finally, the message is read off in columns, in the order specified by the keyword alphabetically or in increasing order.

Decryption

```
def decryptMessage(cipher,key):

    len_cipher = len(cipher)
    len_key = len(key)
    C = len_key
    R = int(len_cipher / len_key)
    rem = len_cipher % len_key
    if(rem > 0 ):
        R = R + 1
    key2 = []
    k = 1
    for i in range(len(key)):
        for j in range(len(key)):
            if key[j]==str(k):
                key2.append(j)
                k=k+1
    mat = []
    for i in range(R):
        new = []
        for j in range(C):
            new.append('0')
        mat.append(new)
    temp = 0
    for j in key2:
        for i in range(R):
            if temp<len_cipher:
                mat[i][j] = cipher[temp]
            temp = temp + 1
     temp = ""   #will contain plaintext
    for i in range(R):
        for j in range(C):
            if(mat[i][j]!='-'):
                temp = temp + mat[i][j]
     return temp
```

To decipher it, the recipient has to work out the column lengths by dividing the message length by the key length. If the remainder is zero then create empty matrix and fill it using ciphertext alphabetically column by column.

Then, write the message row wise, that will be plaintext.

Brute Force Attack

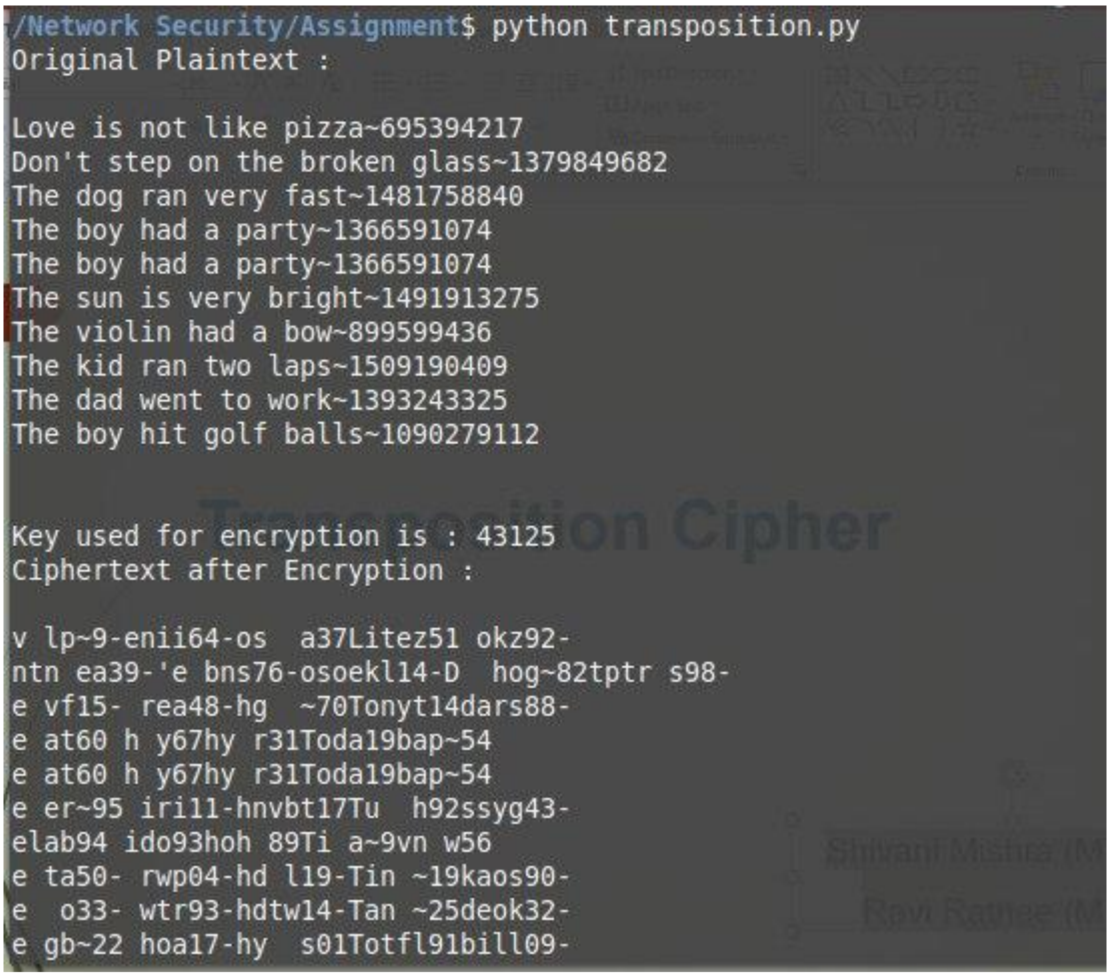
In the Brute Force attack the attacker tries every possible key on a piece of ciphertext until an intelligible translation into plaintext is obtained.

A simple transposition or permutation cipher works by breaking a message into fixed size blocks, and then permuting the characters within each block according to a fixed permutation. The key to the transposition cipher is simply the permutation. The no of the permutation is n! (where n is length of key).

Implementation

Input: Plaintexts with CRC.

Output: Ciphertext after Encryption Algorithm
Plaintext after Decryption Algorithm
Ciphertext with key after brute force attack

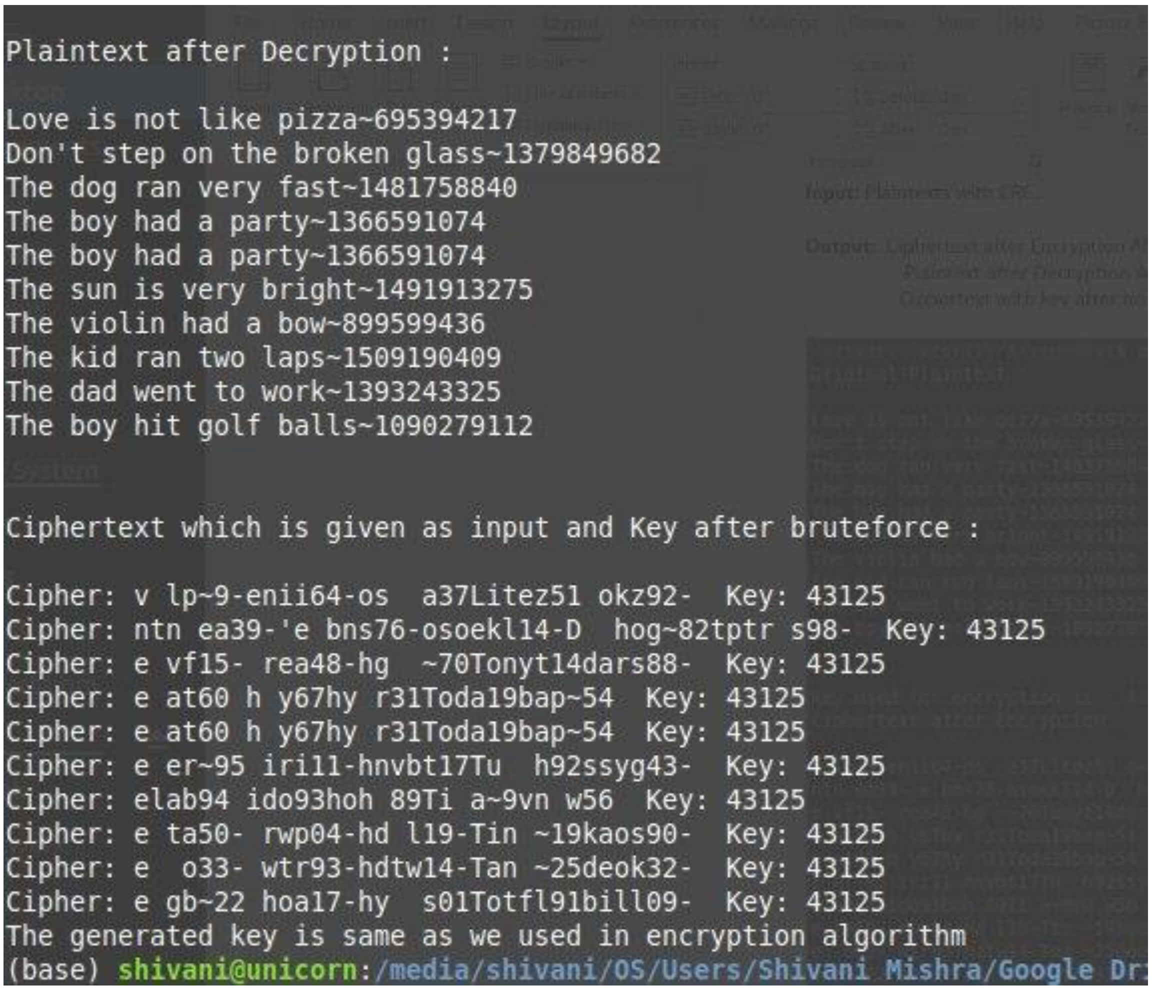


```
/Network Security/Assignment$ python transposition.py
Original Plaintext :

Love is not like pizza~695394217
Don't step on the broken glass~1379849682
The dog ran very fast~1481758840
The boy had a party~1366591074
The boy had a party~1366591074
The sun is very bright~1491913275
The violin had a bow~899599436
The kid ran two laps~1509190409
The dad went to work~1393243325
The boy hit golf balls~1090279112


Key used for encryption is : 43125
Ciphertext after Encryption :

v lp~9-enii64-os  a37Litez51 okz92-
ntn ea39-'e bns76-osoekl14-D  hog~82tptr s98-
e vf15- rea48-hg  ~70Tonyt14dars88-
e at60 h y67hy r31Toda19bap~54
e at60 h y67hy r31Toda19bap~54
e er~95 iri11-hnvbt17Tu  h92ssyg43-
elab94 ido93hoh 89Ti a~9vn w56
e ta50- rwp04-hd l19-Tin ~19kaos90-
e  o33- wtr93-hdtw14-Tan ~25deok32-
e gb~22 hoa17-hy  s01Totfl91bill09-
```



```
Plaintext after Decryption :

Love is not like pizza~695394217
Don't step on the broken glass~1379849682
The dog ran very fast~1481758840
The boy had a party~1366591074
The boy had a party~1366591074
The sun is very bright~1491913275
The violin had a bow~899599436
The kid ran two laps~1509190409
The dad went to work~1393243325
The boy hit golf balls~1090279112


Ciphertext which is given as input and Key after bruteforce :

Cipher: v lp~9-enii64-os  a37Litez51 okz92-  Key: 43125
Cipher: ntn ea39-'e bns76-osoekl14-D  hog~82tptr s98-  Key: 43125
Cipher: e vf15- rea48-hg  ~70Tonyt14dars88-  Key: 43125
Cipher: e at60 h y67hy r31Toda19bap~54  Key: 43125
Cipher: e at60 h y67hy r31Toda19bap~54  Key: 43125
Cipher: e er~95 iri11-hnvbt17Tu  h92ssyg43-  Key: 43125
Cipher: elab94 ido93hoh 89Ti a~9vn w56  Key: 43125
Cipher: e ta50- rwp04-hd l19-Tin ~19kaos90-  Key: 43125
Cipher: e  o33- wtr93-hdtw14-Tan ~25deok32-  Key: 43125
Cipher: e gb~22 hoa17-hy  s01Totfl91bill09-  Key: 43125
The generated key is same as we used in encryption algorithm
(base) shivani@unicorn:/media/shivani/OS/Users/Shivani Mishra/Google Dr
```

Made by: Shivani Mishra (MT20062)

Ravi Rathee (MT20066)