Name: Jigar Siddhpura **SAPID:** 60004200155

DIV: C/C2 **Branch:** Computer Engineering

IS - Experiment 4 - SINGLE COLUMNAR TRANSPOSITION

Jigar Siddhpura
60004210155
C22

Exp 4 : Columnar Transposition Cipher

Aim: To study & implement Simple columnar transposition cipher.

Theory:

It is a classical encryption technique that involves rearranging characters of plaintext according to specific arrangement. A keyword of non-repeating letters is chosen to determine order of column in transposition grid. Arrange letter of keywords in ascending order to determine sequence of columns. During encryption, write plaintext in rows under column determined by keyword filling in characters row by row, if necessary add filler characters. Read off column in alphabetical order of keyword to obtain ciphertext. For decryption, reorder column of ciphertext according to alphabetical order of keyword. Read off rows to obtain original P.T.

Eg: P.T. :- We are discovered flee at once.

key : zebras

order : Z E B R A S

6 3 2 4 1 5

Encryption: Dividing P.T. into group of 6 characters, as key is of 6 characters.

6	3	2	4	1	5
W	E	A	R	E	D
I	S	C	O	V	E
R	E	D	F	L	E
E	A	T	O	N	C
E	Q	K	J	E	U

So cipher text :

EVLNE ACDTK ESEAQ ROFOT DEECU WIREE

Decryption :

6	3	2	4	1	5
W	E	A	R	E	D
I	S	C	O	V	E
R	E	D	F	L	E
E	A	T	O	N	C
E	Q	K	J	E	U

P.T : WEAREDISCOVEREDFLEEATONCE

Conclusion: Thus, we studied simple columnar transposition with encryption & decryption methods implemented python code for it & tested examples for it.

CODE

```
import math

def find_rank(key):
    rank = 0
    for char in sorted(key):
        key = key.replace(char, str(rank), 1)
        rank += 1
    key = [int(char) for char in key]
    return key

def encrypt(plaintext, key):
    columns = len(key)
    rows = math.ceil(len(plaintext) / columns)
    key_rank = find_rank(key)
    print("Key Rank:", key_rank)
    plaintext += "X" * (rows * columns - len(plaintext))
    matrix = [list(plaintext[i: i + columns]) for i in range(0, len(plaintext), columns)]
    for row in matrix:
        print(row)
    ciphertext = ["*" for _ in range(columns)]
    j = 0
    for i in key_rank:
        ciphertext[i] = [row[j] for row in matrix]
        j += 1
    result = []
    for sublist in ciphertext:
        result.extend(sublist)
    return "".join(result)

def decrypt(ciphertext, key):
    columns = len(key)
    rows = math.ceil(len(ciphertext) / columns)
    key_rank = find_rank(key)
    ciphertext += "X" * (rows * columns - len(ciphertext))
    ciphertext_matrix = [list(ciphertext[i:i + rows]) for i in range(0, len(ciphertext), rows)]
    result = []
    for i in range(rows):
        temp = ["*"] * len(key_rank)
        count = 0
        for rank in key_rank:
            temp[count] = ciphertext_matrix[rank][i]
            count += 1
        result.extend(temp)
    return "".join(result).rstrip("X")

# Example usage
plaintext = input("Enter the plaintext: ").upper()
key = input("Enter the key: ").upper()

print(f"\nPlain text: {plaintext}\nKey: {key}\n")

ciphertext = encrypt(plaintext, key)
```

```
print(f"After encryption, Cipher Text: {ciphertext}\n")

decrypted_text = decrypt(ciphertext, key)
print(f"After decryption, Plain Text: {decrypted_text}")
```

OUTPUT

```
PS D:\SEM-6\IS\EXPERIMENTS> python -u "d:\SEM-6\IS\EXPERIMENTS\single_columnar.py"
Enter the plaintext: GOODMORNINGSIR
Enter the key: KEYWORD

Plain text: GOODMORNINGSIR
Key: KEYWORD

Key Rank: [2, 1, 6, 5, 3, 4, 0]
['G', 'O', 'O', 'D', 'M', 'O', 'R']
['N', 'I', 'N', 'G', 'S', 'I', 'R']
After encryption, Cipher Text: RROIGNMSOIDGON

After decryption, Plain Text: GOODMORNINGSIR
PS D:\SEM-6\IS\EXPERIMENTS>
```