Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Experiment 3

Date of Performance : **Date of Submission:**

SAP Id: 60004190057 **Name : Junaid Altaf Girkar**

Div: A **Batch : A4**

Aim of Experiment

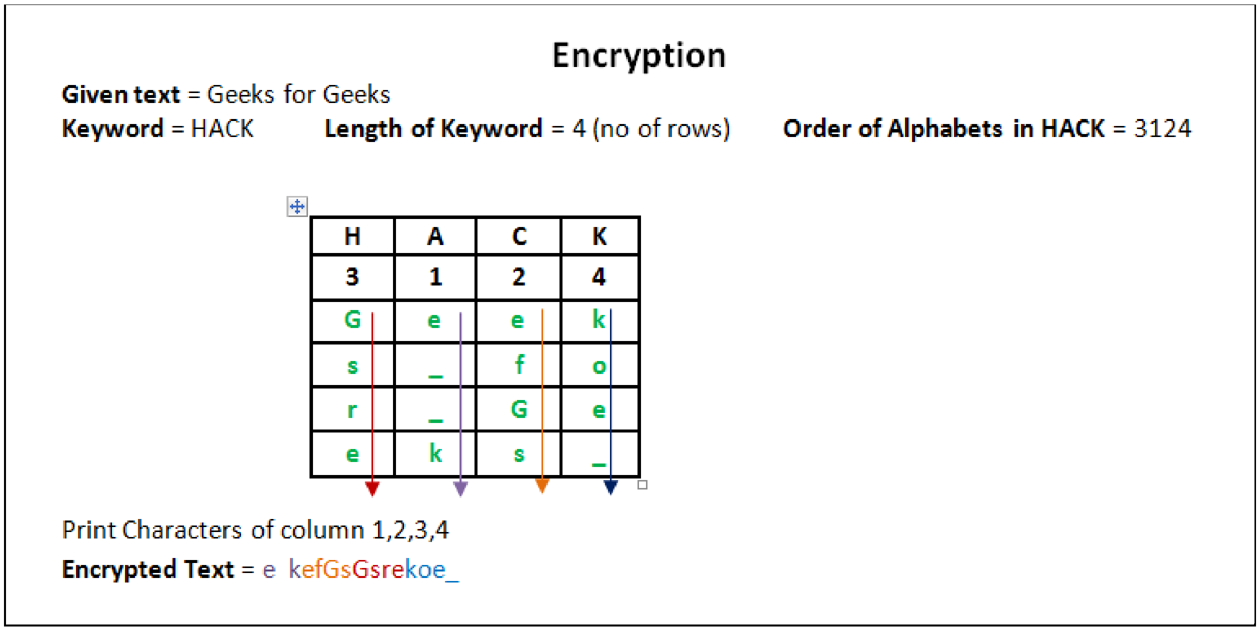
Design and Implement Encryption and Decryption Algorithm for Columnar Transposition Cipher.

Theory / Algorithm / Conceptual Description

The Columnar Transposition Cipher is a form of transposition cipher. Columnar Transposition involves writing the plaintext out in rows, and then reading the ciphertext off in columns one by one.

Encryption

1. In a transposition cipher, the order of the alphabets is rearranged to obtain the cipher-text.
2. 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.
3. Width of the rows and the permutation of the columns are usually defined by a keyword.
4. For example, the word HACK is of length 4 (so the rows are of length 4), and the permutation is defined by the alphabetical order of the letters in the keyword. In this case, the order would be “3 1 2 4”.
5. Any spare spaces are filled with nulls or left blank or placed by a character (Example: _).
6. Finally, the message is read off in columns, in the order specified by the keyword.
7. columnar-transposition-cipher



Encryption

Given text = Geeks for Geeks

Keyword = HACK **Length of Keyword** = 4 (no of rows) **Order of Alphabets in HACK** = 3124

H	A	C	K
3	1	2	4
G	e	e	k
s	_	f	o
r	_	G	e
e	k	s	_

Print Characters of column 1,2,3,4

Encrypted Text = e kefGsGsrekoe_

Decryption

1. To decipher it, the recipient has to work out the column lengths by dividing the message length by the key length.
2. Then, write the message out in columns again, then reorder the columns by reforming the key word.

CODE:

```
import math

key = "DJSCE"

# Encryption
def encryptMessage(msg):
        cipher = ""
        key_index = 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)
        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[key_index])
                cipher += ''.join([row[curr_idx]
                                                for row in matrix])
                key_index += 1

        return cipher

# Decryption
def decryptMessage(cipher):
        decrypted_message = ""

        key_index = 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))

        deciphered_cipher_message = []
        for _ in range(row):
                deciphered_cipher_message += [[None] * col]

        for _ in range(col):
                curr_idx = key.index(key_lst[key_index])
        for j in range(row):
            deciphered_cipher_message[j][curr_idx] = msg_lst[msg_indx]
            msg_indx += 1
        key_index += 1

    try:
        decrypted_message = ''.join(sum(deciphered_cipher_message, []))
    except TypeError:
        raise TypeError("This program cannot",
                        "handle repeating words.")

    null_count = decrypted_message.count('_')

    if null_count > 0:
        return decrypted_message[: -null_count]

    return decrypted_message

msg = "Junaid Girkar"
print("\nPlaintext Message:", msg)

cipher = encryptMessage(msg)
print("\nCiphertext Message: {}".format(cipher))

decrypted_message = decryptMessage(cipher)
print("\nDecryped Message: {}\n".format(decrypted_message))
```

OUTPUT:

```
Plaintext Message: Junaid Girkar

Ciphertext Message: ai_Jdkir_u anGr

Decryped Message: Junaid Girkar
```

CONCLUSION

With the increasing amount of data being generated, it is very important that confidential information does not get leaked and is read by the intended recipient.We learnt about the Columnar Transposition Cipher algorithm and we then wrote a python program to implement it.