

**Experiment 2**

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

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

**Div:** **A** **Batch : A4**

**Aim of Experiment**

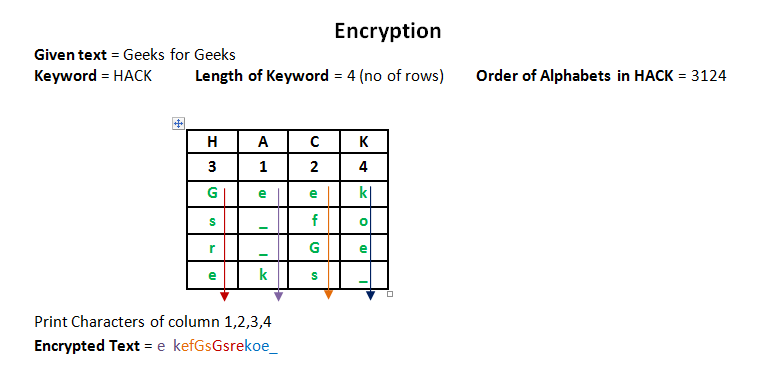
Design and Implement Encryption and Decryption Algorithm for Columnar Transposition Cipher.

**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



**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)) |
| --- |

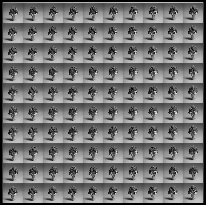
OUTPUT:

| Plaintext Message: Junaid Girkar  Ciphertext Message: ai\_Jdkir\_u anGr  Decryped Message: Junaid Girkar |
| --- |

CODE:

| import cv2 from google.colab.patches import cv2\_imshow import numpy as np  plaintext = list('harrydidyouputyournameinthegobletoffire') key = 'junaidabcd'  def encryption(message, key):  ciphertext = []  # message\_list = list(message)  message\_length = len(message)  sorted\_key = sorted(list(key))  column = len(key)  row = int(ceil(message\_length/column))  dummy\_characters = (row\*column) - message\_length  message.extend('\*'\*dummy\_characters)  matrix = list()  counter = 0  for i in range(row):  temp = list()  for j in range(column):  temp.append(message[counter])  counter += 1  matrix.append(temp)  counter = 0  for j in range(column):  index = key.index(sorted\_key[counter])  # ciphertext += ''.join([row[index] for row in matrix])  for row in matrix:  ciphertext.append(row[index])  counter += 1  return ciphertext ciphertext = encryption(plaintext,key) print('For given plaintext : ' + ''.join(plaintext) + ' the corresponding ciphertext is :' + ''.join(ciphertext))  def decryption(ciphertext, key):  plaintext = []  message\_length = len(ciphertext)  # message = list(ciphertext)  sorted\_key = sorted(list(key))  column = len(key)  row = int(message\_length/column)  counter = 0  matrix = list()  for j in range(row):  matrix.append(['']\*column)  for j in range(column):  index = key.index(sorted\_key[j])  for i in range(row):  matrix[i][index] = ciphertext[counter]  counter+=1  for i in range(row):  for j in range(column):  if matrix[i][j] != '\*':  plaintext.append(matrix[i][j])  return plaintext plaintext = decryption(ciphertext,key) print('For given ciphertext : ' + ''.join(ciphertext) + ' the corresponding plaintext is :' + ''.join(plaintext))  key = "authorized" image = cv2.imread("Brocolli.jpg",0) image = np.array(image,dtype=np.int64) cv2\_imshow(image)  shape = image.shape flat\_image = list(image.flatten())  encrypted = encryption(flat\_image,key) encrypted = np.array(encrypted,dtype=np.int64) encrypted = encrypted.reshape(shape) cv2\_imshow(encrypted)  encrypted\_flat\_image = list(encrypted.flatten())  decrypted = decryption(encrypted\_flat\_image,key) decrypted = np.array(decrypted,dtype=np.int64) decrypted = decrypted.reshape(shape) cv2\_imshow(decrypted) |
| --- |

OUTPUT:

**Original Image Encrypted Image Decrypted Image**

**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.