**Practical 5**

Write a program to perform encryption and decryption using Hill Cipher Technique.

* **CODE :-**

import math

def generate\_matrix(key):

alphabet = "ABCDEFGHIKLMNOPQRSTUVWXYZ"

matrix = []

for i in key.upper():

if i not in matrix:

matrix.append(i)

for j in alphabet:

if j not in matrix:

matrix.append(j)

matrix\_group = []

for e in range(5):

matrix\_group.append('')

matrix\_group[0] = matrix[0:5]

matrix\_group[1] = matrix[5:10]

matrix\_group[2] = matrix[10:15]

matrix\_group[3] = matrix[15:20]

matrix\_group[4] = matrix[20:25]

return matrix\_group

def msg\_to\_digraphs(msg):

message = []

for k in msg:

message.append(k)

for j in range(len(message)):

if " " in message:

message.remove(" ")

i = 0

for e in range(len(message) // 2):

if message[i] == message[i + 1]:

message.insert(i + 1, 'X')

i = i + 2

if len(message) % 2 == 1:

message.append("X")

i = 0

new = []

for x in range(1, len(message) // 2 + 1):

new.append(message[i:i + 2])

i = i + 2

return new

def find\_position(key, letter):

x = y = 0

for i in range(5):

for j in range(5):

if key[i][j] == letter:

x = i

y = j

return x, y

def encryption(message):

message = msg\_to\_digraphs(message)

key\_matrix = generate\_matrix(key)

cipher = []

for e in message:

p1, q1 = find\_position(key\_matrix, e[0])

p2, q2 = find\_position(key\_matrix, e[1])

if p1 == p2:

if q1 == 4:

q1 = -1

if q2 == 4:

q2 = -1

cipher.append(key\_matrix[p1][q1 + 1])

cipher.append(key\_matrix[p1][q2 + 1])

elif q1 == q2:

if p1 == 4:

p1 = -1;

if p2 == 4:

p2 = -1;

cipher.append(key\_matrix[p1 + 1][q1])

cipher.append(key\_matrix[p2 + 1][q2])

else:

cipher.append(key\_matrix[p1][q2])

cipher.append(key\_matrix[p2][q1])

return ''.join(cipher)

def cipher\_to\_digraphs(cipher):

i = 0

new = []

for x in range(len(cipher) // 2):

new.append(cipher[i:i + 2])

i = i + 2

return new

def decryption(cipher):

cipher = cipher\_to\_digraphs(cipher)

key\_matrix = generate\_matrix(key)

plaintext = []

for e in cipher:

p1, q1 = find\_position(key\_matrix, e[0])

p2, q2 = find\_position(key\_matrix, e[1])

if p1 == p2:

if q1 == 4:

q1 = -1

if q2 == 4:

q2 = -1

plaintext.append(key\_matrix[p1][q1 - 1])

plaintext.append(key\_matrix[p1][q2 - 1])

elif q1 == q2:

if p1 == 4:

p1 = -1;

if p2 == 4:

p2 = -1;

plaintext.append(key\_matrix[p1 - 1][q1])

plaintext.append(key\_matrix[p2 - 1][q2])

else:

plaintext.append(key\_matrix[p1][q2])

plaintext.append(key\_matrix[p2][q1])

for i in range(len(plaintext)):

if "X" in plaintext:

plaintext.remove("X")

output = ""

for j in plaintext:

output += j

return output.lower()

key = input("Enter key : ")

message = input("Enter original message : ")

print(msg\_to\_digraphs(message))

print(generate\_matrix(key))

print("Encryption")

msg\_final=message.upper()

print(encryption(msg\_final))

choice=int(input("Continue decryption process, press 1:"))

if(choice==1):

key = input("Enter key : ")

cipher = input("Enter cipher text: ")

print("Decryption")

cipher\_final=cipher.upper()

print(decryption(cipher\_final))

else:

print("Invalid Choice.")

exit();

* **OUTPUT :-**

