**Applied Cryptography and Network Security**

**(CSI3002)**

**LAB ASSESSMENT – 2**

**Name : RITHIV.R**

**Reg No : 19MIC0113**

**Slot : L27+L28**

1. **DES ENCRYPTION:**

**Code:**

print('\nDES ENCRYPTION-RITHIV.R(19MIC0113)\n')

PC1 = [

57,49,41,33,25,17,9,

1,58,50,42,34,26,18,

10,2,59,51,43,35,27,

19,11,3,60,52,44,36,

63,55,47,39,31,23,15,

7,62,54,46,38,30,22,

14,6,61,53,45,37,29,

21,13,5,28,20,12,4,

]

PC2 = [

14,17,11,24,1,5,

3,28,15,6,21,10,

23,19,12,4,26,8,

16,7,27,20,13,2,

41,52,31,37,47,55,

30,40,51,45,33,48,

44,49,39,56,34,53,

46,42,50,36,29,32,

]

iterations = [

1,1,2,2,2,2,2,2,1,2,2,2,2,2,2,1]

list\_keys = []

IP = [

58,50,42,34,26,18,10,2,

60,52,44,36,28,20,12,4,

62,54,46,38,30,22,14,6,

64,56,48,40,32,24,16,8,

57,49,41,33,25,17,9,1,

59,51,43,35,27,19,11,3,

61,53,45,37,29,21,13,5,

63,55,47,39,31,23,15,7

]

Expansion = [

32,1,2,3,4,5,

4,5,6,7,8,9,

8,9,10,11,12,13,

12,13,14,15,16,17,

16,17,18,19,20,21,

20,21,22,23,24,25,

24,25,26,27,28,29,

28,29,30,31,32,1,

]

P = [

16,7,20,21,

29,12,28,17,

1,15,23,26,

5,18,31,10,

2,8,24,14,

32,27,3,9,

19,13,30,6,

22,11,4,25

]

sb = [

[

[14,4,13,1,2,15,11,8,3,10,6,12,5,9,0,7],

[0,15,7,4,14,2,13,1,10,6,12,11,9,5,3,8],

[4,1,14,8,13,6,2,11,15,12,9,7,3,10,5,0],

[15,12,8,2,4,9,1,7,5,11,3,14,10,0,6,13]

],

[

[15,1,8,14,6,11,3,4,9,7,2,13,12,0,5,10],

[3,13,4,7,15,2,8,14,12,0,1,10,6,9,11,5],

[0,14,7,11,10,4,13,1,5,8,12,6,9,3,2,15],

[13,8,10,1,3,15,4,2,11,6,7,12,0,5,14,9]

],

[

[10,0,9,14,6,3,15,5,1,13,12,7,11,4,2,8],

[13,7,0,9,3,4,6,10,2,8,5,14,12,11,15,1],

[13,6,4,9,8,15,3,0,11,1,2,12,5,10,14,7],

[1,10,13,0,6,9,8,7,4,15,14,3,11,5,2,12]

],

[

[7,13,14,3,0,6,9,10,1,2,8,5,11,12,4,15],

[13,8,11,5,6,15,0,3,4,7,2,12,1,10,14,9],

[10,6,9,0,12,11,7,13,15,1,3,14,5,2,8,4],

[3,15,0,6,10,1,13,8,9,4,5,11,12,7,2,14]

],

[

[2,12,4,1,7,10,11,6,8,5,3,15,13,0,14,9],

[14,11,2,12,4,7,13,1,5,0,15,10,3,9,8,6],

[4,2,1,11,10,13,7,8,15,9,12,5,6,3,0,14],

[11,8,12,7,1,14,2,13,6,15,0,9,10,4,5,3]

],

[

[12,1,10,15,9,2,6,8,0,13,3,4,14,7,5,11],

[10,15,4,2,7,12,9,5,6,1,13,14,0,11,3,8],

[9,14,15,5,2,8,12,3,7,0,4,10,1,13,11,6],

[4,3,2,12,9,5,15,10,11,14,1,7,6,0,8,13]

],

[

[4,11,2,14,15,0,8,13,3,12,9,7,5,10,6,1],

[13,0,11,7,4,9,1,10,14,3,5,12,2,15,8,6],

[1,4,11,13,12,3,7,14,10,15,6,8,0,5,9,2],

[6,11,13,8,1,4,10,7,9,5,0,15,14,2,3,12]

],

[

[13,2,8,4,6,15,11,1,10,9,3,14,5,0,12,7],

[1,15,13,8,10,3,7,4,12,5,6,11,0,14,9,2],

[7,11,4,1,9,12,14,2,0,6,10,13,15,3,5,8],

[2,1,14,7,4,10,8,13,15,12,9,0,3,5,6,11]

]

]

IP\_inv = [

40,8,48,1,56,24,64,32,

39,7,47,15,55,23,63,31,

38,6,46,14,54,22,62,30,

37,5,45,13,53,21,61,29,

36,4,44,12,52,20,60,28,

35,3,43,11,51,19,59,27,

34,2,42,10,50,18,58,26,

33,1,41,9,49,17,57,25

]

def XOR(i,j):

if(i==j):

return '0'

else:

return '1'

def hextodec(trial):

value = ''

for i in trial:

result = "{0:08b}".format(int(i, 16))[-4:]

value = value+result

return value

def key\_PC\_1(key):

global PC1

value = hextodec(key)

temp = ''

temp1 = ''

for j,i in enumerate(value):

if((j+1)%8==0):

pass

else:

temp = temp + i

for j,i in enumerate(temp):

temp1 = temp1 + value[PC1[j]-1]

return(temp1)

def shift(key):

getter = key\_PC\_1(key)

val1 = getter[:len(getter)//2]

val2 = getter[len(getter)//2:]

myarray = []

for i in iterations:

temp1 = val1[i:]+val1[:i]

temp2 = val2[i:]+val2[:i]

myarray.append(temp1+temp2)

val1 = temp1

val2 = temp2

return myarray

def key\_PC\_2(key):

global list\_keys

myarr = shift(key)

print('\nSub-Key Generation:\n')

for j,i in enumerate(myarr):

rem1 = ''

rem2 = ''

for c2,c1 in enumerate(i):

if((c2+1)%7==0):

pass

else:

rem1 = rem1 + c1

for c2,c1 in enumerate(rem1):

rem2 = rem2 + i[PC2[c2]-1]

list\_keys.append(rem2)

print('K'+str(j+1)+':',rem2)

def Initial\_Perm(message):

temporary = ''

for i in IP:

temporary = temporary + message[i-1]

return temporary

def Exp\_Perm(R):

temp = ''

for i in Expansion:

temp = temp+R[i-1]

return temp

def sbox(Box,message):

b16 = message[0]+message[-1]

b2345 = message[1:-1]

val = Box[int(b16,2)][int(b2345,2)]

res = bin(val)[2:].zfill(4)

return res

def Permutation(msg):

val = ''

for i in P:

val = val + msg[i-1]

return val

def func(R,K,counter):

global sb

val = Exp\_Perm(R)

print('Expansion Result:',val)

modified = ''

for i,j in zip(val,K):

modified=modified+XOR(i,j)

print('XOR('+'R'+str(counter)+',K'+str(counter+1)+'):',modified)

sixthocc = [modified[i:i+6] for i in range(0, len(modified), 6)]

sbval = ''

for Box,message in zip(sb,sixthocc):

sbval = sbval+sbox(Box, message)

print('Sboxes Result:',sbval)

final = Permutation(sbval)

print('Permutation Result:',final)

return final

def swap(a,b):

return b+a

def inverse\_IP(encoded):

temp = ''

for i in IP\_inv:

temp = temp+encoded[i-1]

return temp

def bintohex(val):

number = int(val,2)

hexa\_number = format(number, 'x')

return hexa\_number

def encode(message):

print('\nEncoding Part:')

hexa = hextodec(message)

ipermutted = Initial\_Perm(hexa)

print('\nInitial Permutation Value:',ipermutted)

l = ipermutted[:len(ipermutted)//2]

r = ipermutted[len(ipermutted)//2:]

print('\nL0:',l)

print('R0:',r)

for i in range(16):

print('\nRound',i+1,':\n')

k1 = list\_keys[i]

f = func(r,k1,i)

temp1 = r

value1 = ''

for c1,c2 in zip(l,f):

value1 = value1 + XOR(c1,c2)

r = value1

l = temp1

print('L'+str(i+1),':',l)

print('R'+str(i+1),':',r)

swapped = swap(l,r)

print('\nSwapping Result:')

print('\nL16:',r)

print('R16:',l)

final = inverse\_IP(swapped)

print('\nInverse Pernutation:',final)

fourocc = [final[i:i+4] for i in range(0, len(final), 4)]

result = ''

for i in fourocc:

result = result + bintohex(i)

return result.upper()

message = input('Enter the Message:').lower()

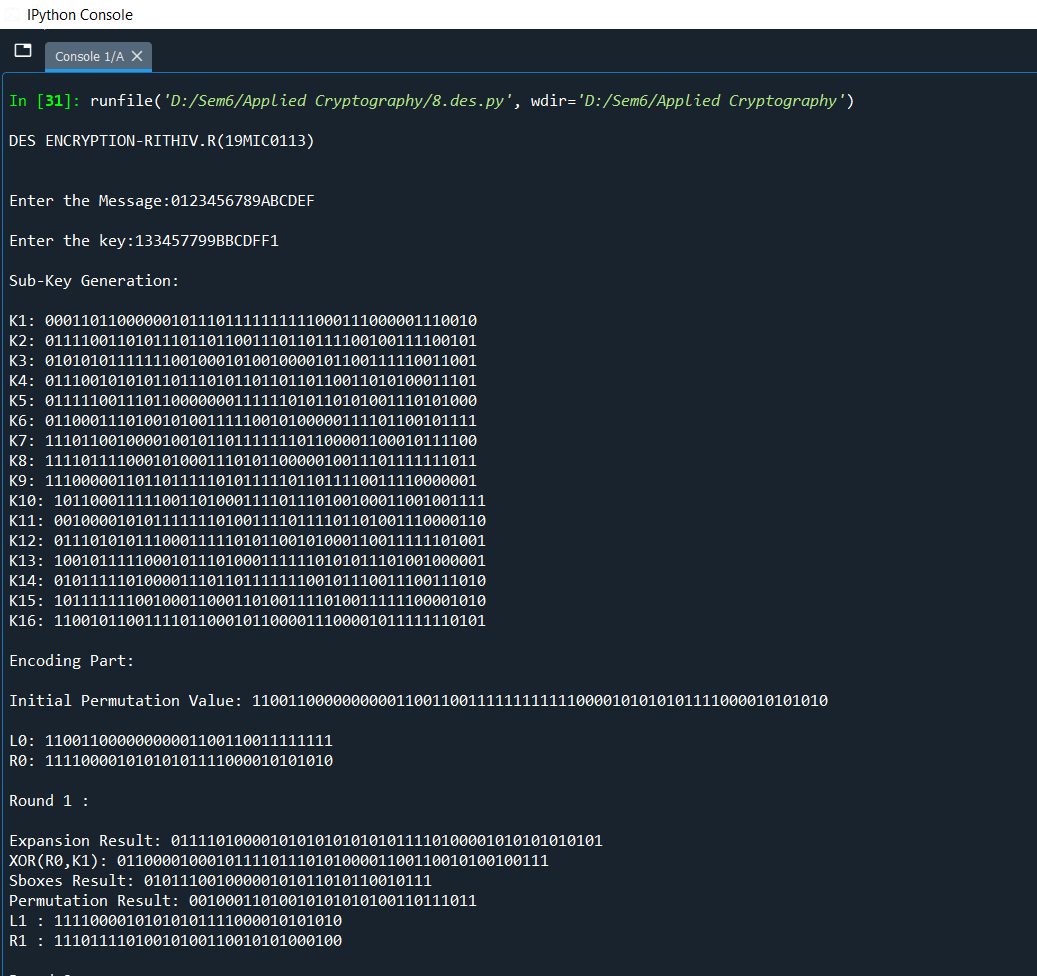
key = input('Enter the key:').lower()

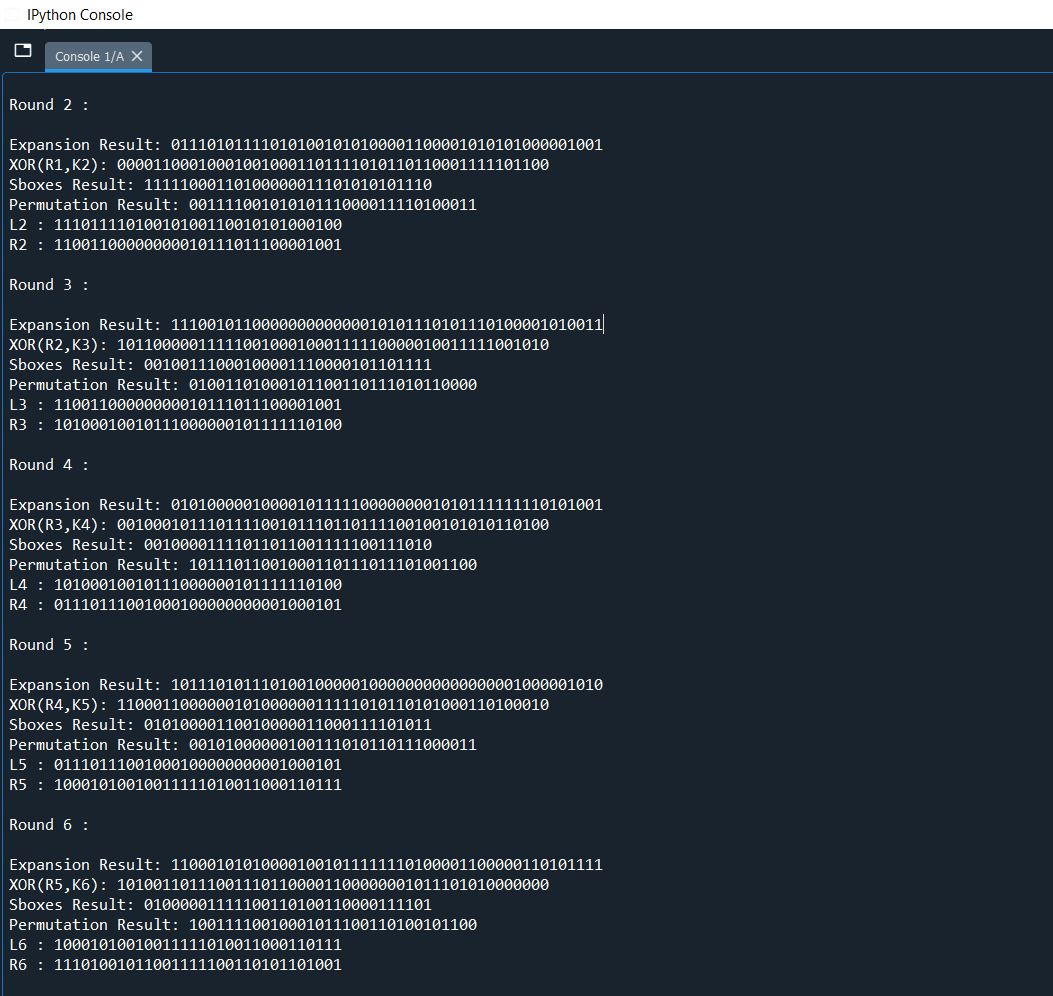
key\_PC\_2(key)

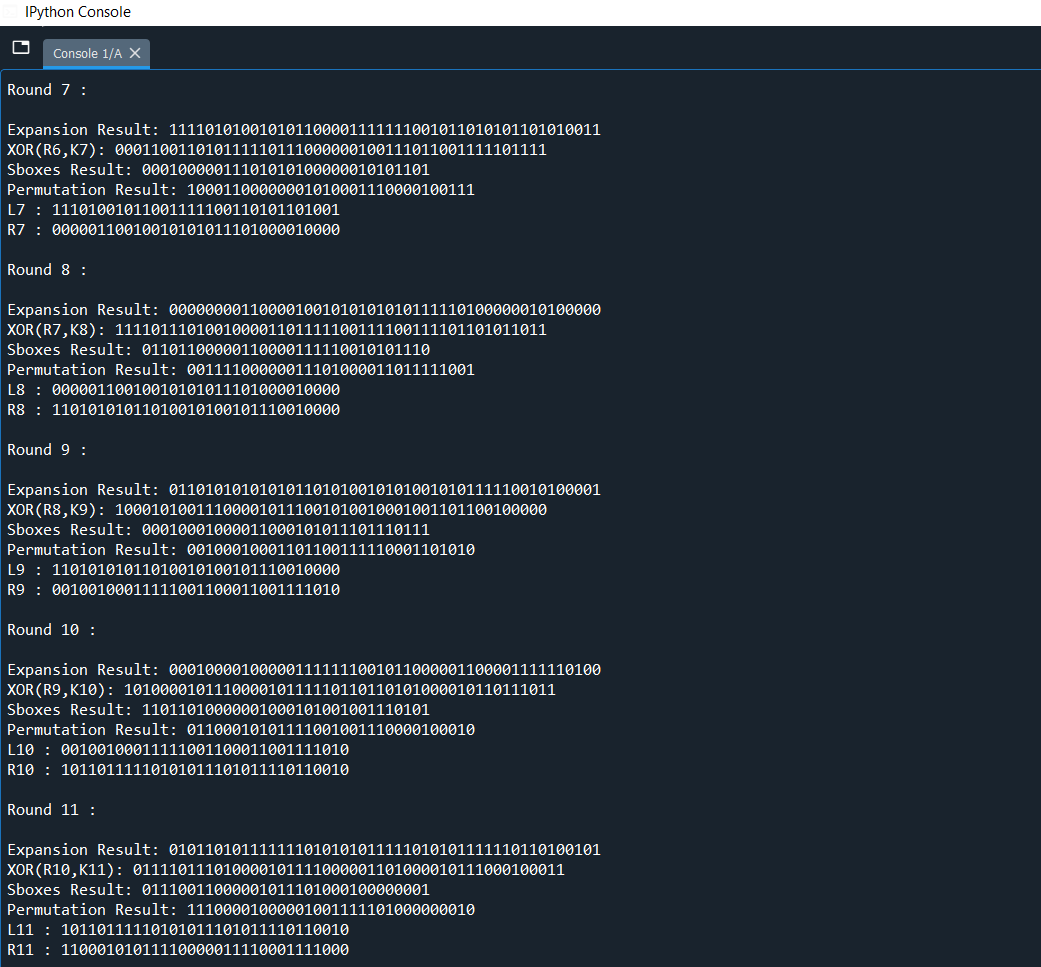
output=encode(message)

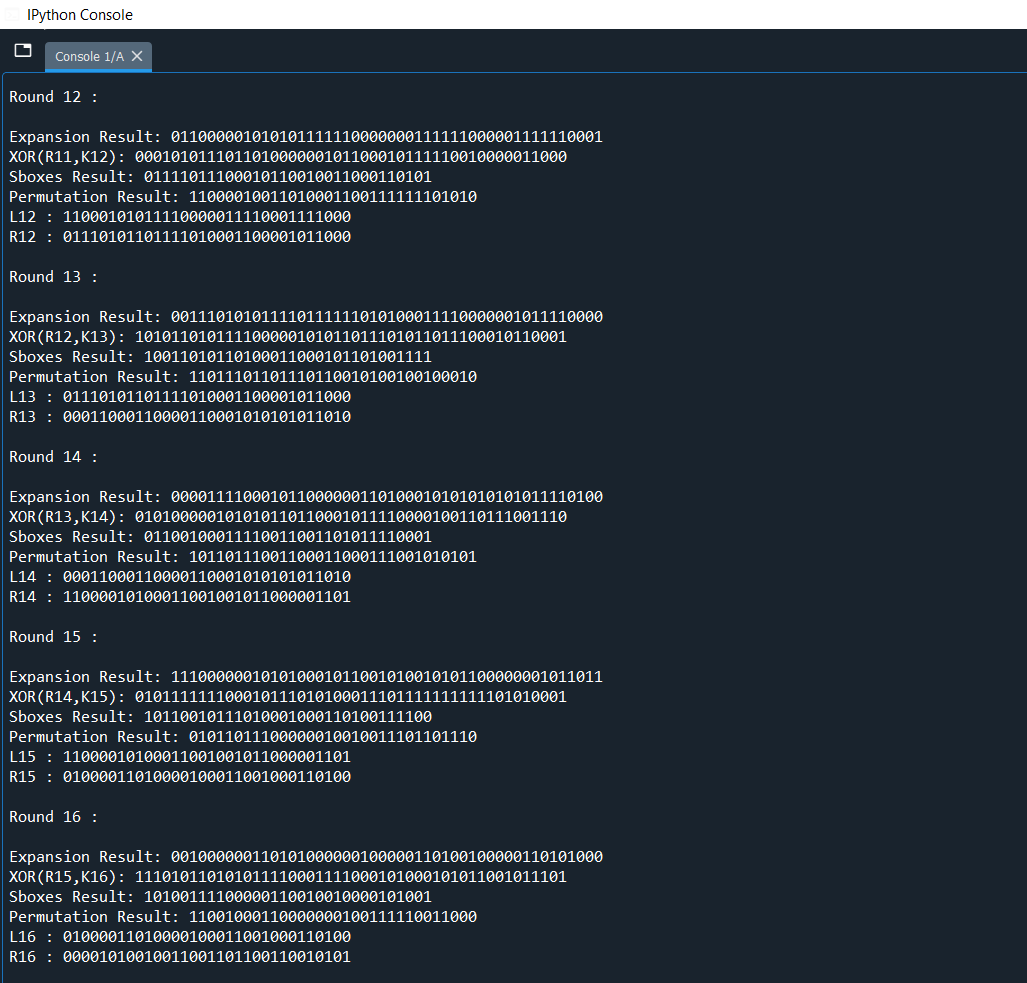
print('\nCipher Text Generated:',output)

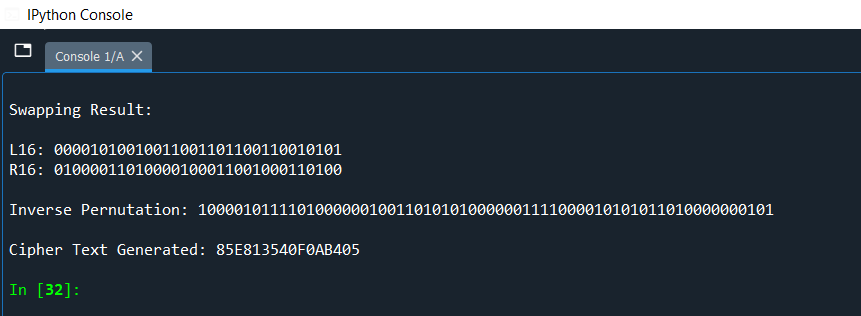
**Output:**











**CODE UPLOADED IN GOOGLE DRIVE LINK:**

[**https://drive.google.com/file/d/1ItzP9\_XlPw6R5mgV1rEwM1V3JDSI\_dwO/view?usp=sharing**](https://drive.google.com/file/d/1ItzP9_XlPw6R5mgV1rEwM1V3JDSI_dwO/view?usp=sharing)