```
#                                CSCI 7000 - Cryptography and Cryptanalysis (Fall 2015)
#                                                Homework - 2
#                                                                                BHARADWAJ THIRUMAL




from collections import deque
import binascii
import copy
from itertools import repeat


#Hex to binary
conversion-------------------------------------------------------------------------

def binarify(hexStr, padLength):
    return(bin(int(hexStr,16))[2:].zfill(padLength))

#Split binary
input-----------------------------------------------------------------------------------

def splitBin(binStr, binStrlen, splitLength):
    expInpList = []

    for i in range(0,binStrlen,splitLength):
            expInpList.append(int(binStr[i:i+splitLength],2))
    return expInpList

def splitBin1(binStr, binStrlen, splitLength):
    expInpList = []

    for i in range(0,binStrlen,splitLength):
            expInpList.append(binStr[i:i+splitLength])
    return expInpList

#Expansion
E-box-----------------------------------------------------------------------------------

def expandBin(binStr):
    exp =[  0,
         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 ]

    expStr = ''
    for i in range(1,49):
        expStr = expStr + binStr[exp[i]]
    return expStr

#Reverse the bit mixing
permutation-----------------------------------------------------------------
```

```
def unPermute(permutedStr):
    p = [ 0,
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 ]

    unPermuteStr = ['']*33
    for i in range(1,33):
        unPermuteStr[p[i]] += permutedStr[i]
    return unPermuteStr

#S-Box lookup
function-------------------------------------------------------------------------------

def findOutputXor(bList,j):

    s = [
[
  [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] ]
```

```
]

    tempList = [],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[]
    for i in range(0,64):
        b = bin(i)[2:].zfill(6)
        bPair = bin(bList[i])[2:].zfill(6)

        bRow = int((b[0]+b[5]),2)     #Getting row from Bj
        bCol = int(b[1:5],2)     #Getting column from Bj

        bPairRow = int((bPair[0]+bPair[5]),2)     #Getting row from Bj*
        bPairCol = int(bPair[1:5],2) #Getting column from Bj*

        xorValue = s[j][bRow][bCol] ^ s[j][bPairRow][bPairCol]
        if i not in tempList[xorValue] and bList[i] not in tempList[xorValue]:
            tempList[xorValue].append(i)
            tempList[xorValue].append(bList[i])
    return tempList


#Differential attack - parent function-----------------------------------------------------------

def undes(L3, L3Pair, L0, L0Pair, R3, R3Pair):

    str1 = L3 ^ L3Pair
    str1 = format(str1,'02x')

    temp = binarify(str1, 32)   #Convert hex to binary

    expInpList = []
    expInpList1 = []

    expInpList = splitBin(temp,len(temp),4)     #Split into 8 4-bit binaries

    temp = expandBin('0'+temp)       #Expansion E-Box
    expInpList1 = splitBin(temp,len(temp),6)        #Split into 8 6-bit binaries (E1' to E8')


    test = L0 ^ L0Pair ^ R3 ^ R3Pair
    permutedOutXor = format(test,'02x')

    permutedOutXorBin = binarify(permutedOutXor,32)

    outputXorList = []

    tempL = []

    tempL = unPermute('0'+permutedOutXorBin)        #Unpermute the string
    tempx = ''
    for i in range(1,33):
        tempx += tempL[i]

    outputXorList = splitBin(tempx,len(tempx),4)        #Split into 8 4-bit binaries (C1' to
    C8')

    B = [[]]*9
    tempList = []
```

```
    i = 0

    for BjXor in expInpList1:
        for Bj in range(0,64):
            tempList.append(BjXor ^ Bj)
        B[i] = copy.deepcopy(tempList)
        tempList = []
        i += 1

    list1 = [[]]*9

    for j in range(0,8):    #Map Bj and Bj* for each Bj' to Cj
        list1[j] = findOutputXor(B[j],j)

    possiblePairs = [[]]*8
    for i in range(0,8):
        possiblePairs[i] = list1[i][outputXorList[i]]

    keyCandidateList = [], [], [], [], [], [], [], [], []

    str2 = format(L3,'02x')
    str2 = binarify(str2, 32)

    str2 = expandBin('0'+str2)

    eList = []
    eList = splitBin(str2,len(str2),6)

    for i in range(0,8):
        for j in range(0, len(possiblePairs[i])):
            keyCandidateList[i].append(eList[i] ^ possiblePairs[i][j])

    return keyCandidateList

#PC2Inverse

def pc2Inverse(rKey,flagx):
  pc2 = [ 0,
       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 ]
  unPermuteStr = ['x']*57
  tempStr = ''
  for i in range(1,49):
    unPermuteStr[pc2[i]] = rKey[i]
  if(flagx == 1):
    for strx in unPermuteStr:
      tempStr += strx
    #print(len(tempStr))
    return tempStr[1:]
  return unPermuteStr[1:]

def pc1Inverse(rKey,flagx):
  pc1 = [ 0,
       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 ]
  print(len(pc1))
  unPermuteStr = ['x']*65
  tempStr = ''
  for i in range(1,57):
    unPermuteStr[pc1[i]] = rKey[i]
  if(flagx == 1):
    for strx in unPermuteStr:
      tempStr += strx
    #print(len(tempStr))
    return tempStr[1:]
  return unPermuteStr[1:]


def concatList(l1):
  tempStr = ''
  for strx in l1:
    tempStr += strx
  return tempStr


pairs = [
        [
            [ [0x748502cd, 0x38451097], [0x2e48787d, 0xfb8509e6] ],
            [ [0x38747564, 0x38451097], [0xfc19cb45, 0xb6d9f494] ]
        ],
        [
            [ [0x48691102, 0x6acdff31], [0xac777016, 0x3ddc98e1] ],
            [ [0x375bd31f, 0x6acdff31], [0x7d708f6d, 0x4bc7ef16] ]
        ],
        [
            [ [0x357418da, 0x013fec86], [0x5a799643, 0x9823cf12] ],
            [ [0x12549847, 0x013fec86], [0xae46e276, 0x16c26b04] ]
        ]
]

candidateKeysList0 = [], [], [], [], [], [], [], []
candidateKeysList1 = [], [], [], [], [], [], [], []
candidateKeysList2 = [], [], [], [], [], [], [], []

candidateKeysList0 = undes(pairs[0][0][1][0], pairs[0][1][1][0], pairs[0][0][0][0],
pairs[0][1][0][0], pairs[0][0][1][1], pairs[0][1][1][1] )
candidateKeysList1 = undes(pairs[1][0][1][0], pairs[1][1][1][0], pairs[1][0][0][0],
pairs[1][1][0][0], pairs[1][0][1][1], pairs[1][1][1][1] )
candidateKeysList2 = undes(pairs[2][0][1][0], pairs[2][1][1][0], pairs[2][0][0][0],
pairs[2][1][0][0], pairs[2][0][1][1], pairs[2][1][1][1] )

masterList = []

for i in range(0,8):
  for j in range(0,len(candidateKeysList0[i])):
    findThis = candidateKeysList0[i][j]
    if findThis in candidateKeysList1[i] and findThis in candidateKeysList2[i]:
      masterList.append(findThis)

print(masterList)
```

```
dummyList = copy.deepcopy(masterList)
dupList = []

i = 0
for num1 in dummyList:
  if dummyList.count(num1) > 1:
    i += 1
    dupList.append(num1)
  dummyList = copy.deepcopy(dummyList[i:])

possibleKey1 = ''
possibleKey2 = ''
possibleKey3 = ''
possibleKey4 = ''

roundKey = []
roundKeyList = [[]]

j = 0

for num2 in dupList:
  for i in range(0,9):
    dummyList = copy.deepcopy(masterList)
    if dummyList[i] == num2:
      dummyList.pop(i)
      for num in dummyList:
        num = format(num,'02x')
        roundKey.append(binarify(num,6))
      j += 1

for i in range(0,33,8):
  roundKeyList.append(roundKey[i:i+8])

possibleKey1 = concatList(roundKeyList[1])
possibleKey2 = concatList(roundKeyList[2])
possibleKey3 = concatList(roundKeyList[3])
possibleKey4 = concatList(roundKeyList[4])



afterPC2List1 = []
afterPC2List2 = []
afterPC2List3 = []
afterPC2List4 = []

afterPC2List1 = (pc2Inverse('0' + possibleKey1,0))
afterPC2List2 = (pc2Inverse('0' + possibleKey2,0))
afterPC2List3 = (pc2Inverse('0' + possibleKey3,0))
afterPC2List4 = (pc2Inverse('0' + possibleKey4,0))


print(afterPC2List3)

afterPC2List1 =  deque(afterPC2List1)
afterPC2List1.rotate(4)
afterPC2List2 =  deque(afterPC2List2)
```

```
afterPC2List2.rotate(4)
afterPC2List3 =  deque(afterPC2List3)
afterPC2List3.rotate(4)
afterPC2List4 =  deque(afterPC2List4)
afterPC2List4.rotate(4)

print(afterPC2List3)

possibleKey1 = concatList(afterPC2List1)
possibleKey2 = concatList(afterPC2List2)
possibleKey3 = concatList(afterPC2List3)
possibleKey4 = concatList(afterPC2List4)



afterPC1List1 = (pc1Inverse('0' + possibleKey1,1))
afterPC1List2 = (pc1Inverse('0' + possibleKey2,1))
afterPC1List3 = (pc1Inverse('0' + possibleKey3,1))
afterPC1List4 = (pc1Inverse('0' + possibleKey4,1))

print(afterPC1List1,len(afterPC1List1))
print(afterPC1List2,len(afterPC1List2))
print(afterPC1List3,len(afterPC1List3))
print(afterPC1List4,len(afterPC1List4))
```