**PROBLEM 1**

**Script**

*"""  
Homework Number: 2  
Name: Sneha Mahapatra  
ECN Login: mahapat0  
Due Date: 01/30/2020  
"""*# !/usr/bin/env python  
# !/usr/bin/env python -W ignore:tostring:DeprecationWarning  
### DES\_text.py  
import codecs  
import sys  
import BitVector  
# from get\_encryption\_key import \*  
from generate\_round\_keys import \*  
from illustrate\_des\_substitution import \*  
import warnings  
warnings.simplefilter("ignore", DeprecationWarning)  
expansion\_permutation = [31, 0, 1, 2, 3, 4, 3, 4, 5, 6, 7, 8, 7, 8, 9, 10, 11, 12, 11, 12, 13, 14, 15, 16, 15, 16, 17,18, 19, 20, 19, 20, 21, 22, 23, 24, 23, 24, 25, 26, 27, 28, 27, 28, 29, 30, 31, 0]  
p\_box\_permutation = [15, 6, 19, 20, 28, 11, 27, 16, 0, 14, 22, 25, 4, 17, 30, 9, 1, 7, 23, 13, 31, 26, 2, 8, 18, 12, 29, 5, 21, 10, 3, 24]  
SIZE = 64  
  
def encrypt():  
 FILEREAD = open(sys.argv[4], 'r')  
 key = FILEREAD.read()  
 FILEREAD.close()  
 keyBit = get\_encryption\_key(key)  
 round\_key = generate\_round\_keys(keyBit)  
 bv = BitVector(filename=sys.argv[3])  
 text\_file = open(sys.argv[5], "w")  
 while (bv.more\_to\_read):  
 bitvec = bv.read\_bits\_from\_file(SIZE)  
 if (len(str(bitvec)) % SIZE != 0):  
 x = bitvec.length() % SIZE  
 bitvec.pad\_from\_right(SIZE-x)  
 if (len(str(bitvec)) > 0):  
 [LE, RE] = bitvec.divide\_into\_two()  
 for keyR in round\_key:  
 temp = RE  
 newRE = RE.permute(expansion\_permutation)  
 out\_xor = newRE ^ keyR  
 output = substitute(out\_xor)  
 round\_i = output.permute(p\_box\_permutation)  
 LE = LE ^ round\_i  
 RE = LE  
 LE = temp  
 bitX = RE + LE  
 myhexstring = bitX.get\_bitvector\_in\_hex()  
 text\_file.write(myhexstring)  
 text\_file.close()  
 pass  
  
  
def decrypt():  
 FILEREAD = open(sys.argv[4], 'r')  
 key = FILEREAD.read()  
 FILEREAD.close()  
 keyBit = get\_encryption\_key(key)  
 round\_key = generate\_round\_keys(keyBit)  
 round\_key = round\_key[::-1]  
  
 FILEHEX = open(sys.argv[3], 'r')  
 hexString = FILEHEX.read()  
 bv = BitVector(hexstring=hexString)  
  
 bvList = list(bv)  
 FILEHEX.close()  
  
 text\_file = open(sys.argv[5], "w")  
  
 secOfBits = bv.length() / SIZE  
 index = 0  
 index1 = 0  
 totalList = []  
 while (index < secOfBits):  
 bitvec = BitVector(bitlist=bvList[index1:index1 + SIZE])  
 [LE, RE] = bitvec.divide\_into\_two()  
 for keyR in round\_key:  
 temp = RE  
 newRE = RE.permute(expansion\_permutation)  
 out\_xor = newRE ^ keyR  
 output = substitute(out\_xor)  
 round\_i = output.permute(p\_box\_permutation)  
 LE = round\_i ^ LE  
 RE = LE  
 LE = temp  
 bitX = RE + LE  
 if(secOfBits - index == 1):  
 strX =bitX.get\_text\_from\_bitvector()  
 strC = list(strX)  
 for k in strC:  
 if(k.isprintable()):  
 text\_file.write(k)  
 else:  
 text\_file.write(bitX.get\_text\_from\_bitvector())  
 index += 1  
 index1 += SIZE  
 text\_file.close()  
 pass  
  
def main():  
 charX = sys.argv[2]  
 if (charX == '-e'):  
 encrypt()  
 elif (charX == '-d'):  
 decrypt()  
 else:  
 print("Either -e or -d")  
 pass  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()  
# Convert Image to Bit vector and get rid of first three lines

**Explanation**

Des uses the Fiestal Structure to encrypted their data. What’s great about the Fiestal Structure is that every 64 bit section in the encrypt is the same as the reverse of decrypt. That’s how I treated it when writing the code. So encrypt and decrypt are very similar I only had to change the way the file is read as one is a string and the other is hex.

**Encrypted**



**Decrypted**

Earlier this week, security researchers took note of a series of changes Linux and Windows developers began rolling out in beta updates to address a critical security flaw: A bug in Intel chips allows low-privilege processes to access memory in the computer's kernel, the machine's most privileged inner sanctum. Theoretical attacks that exploit that bug, based on quirks in features Intel has implemented for faster processing, could allow malicious software to spy deeply into other processes and data on the target computer or smartphone. And on multi-user machines, like the servers run by Google Cloud Services or Amazon Web Services, they could even allow hackers to break out of one user's process, and instead snoop on other processes running on the same shared server. On Wednesday evening, a large team of researchers at Google's Project Zero, universities including the Graz University of Technology, the University of Pennsylvania, the University of Adelaide in Australia, and security companies including Cyberus and Rambus together released the full details of two attacks based on that flaw, which they call Meltdown and Spectre.

**PROBLEM 2**

**Script**

*"""  
Homework Number: 2  
Name: Sneha Mahapatra  
ECN Login: mahapat0  
Due Date: 01/30/2020  
"""*#!/usr/bin/env python  
#!/usr/bin/env python -W ignore::DeprecationWarning  
### DES\_text.py  
import codecs  
import sys  
import io  
import numpy  
import BitVector  
#from get\_encryption\_key import \*  
from math import \*  
from generate\_round\_keys import \*  
from illustrate\_des\_substitution import \*  
import warnings  
warnings.simplefilter("ignore", DeprecationWarning)  
expansion\_permutation = [31, 0, 1, 2, 3, 4, 3, 4, 5, 6, 7, 8, 7, 8, 9, 10, 11, 12, 11, 12, 13, 14, 15, 16, 15, 16, 17, 18, 19, 20, 19, 20, 21, 22, 23, 24, 23, 24, 25, 26, 27, 28, 27, 28, 29, 30, 31, 0]  
p\_box\_permutation = [15,6,19,20,28,11,27,16,0,14,22,25,4,17,30,9,1,7,23,13,31,26,2,8,18,12,29,5,21,10,3,24]  
SIZE = 64  
def encrypt():  
  
 FILEREAD = open(sys.argv[3], 'r')  
 key = FILEREAD.read()  
 FILEREAD.close()  
  
 image = open(sys.argv[2], 'rb')  
 imageno = image.readline()  
 height\_width = image.readline()  
 max\_pixVal = image.readline()  
 bv = BitVector(filename=sys.argv[2])  
 image.close()  
  
 keyBit = get\_encryption\_key(key)  
 round\_key = generate\_round\_keys(keyBit)  
  
 text\_file = open(sys.argv[4], "wb")  
 text\_file.write(imageno)  
 text\_file.write(height\_width)  
 text\_file.write(max\_pixVal)  
 while (bv.more\_to\_read):  
 bitvec = bv.read\_bits\_from\_file(SIZE)  
 if (len(str(bitvec)) % SIZE != 0):  
 x = bitvec.length() % 64  
 bitvec.pad\_from\_right(SIZE - x)  
 if (len(str(bitvec)) > 0):  
 [LE, RE] = bitvec.divide\_into\_two()  
 for keyR in round\_key:  
 temp = RE  
 newRE = RE.permute(expansion\_permutation)  
 out\_xor = newRE ^ keyR  
 output = substitute(out\_xor)  
 round\_i = output.permute(p\_box\_permutation)  
 LE = LE ^ round\_i  
 RE = LE  
 LE = temp  
 bitX = RE + LE  
 bitX.write\_to\_file(text\_file)  
 text\_file.close()  
 pass  
  
def main():  
 encrypt()  
 pass  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()  
#Convert Image to Bit vector and get rid of first three lines

**Explanation**

I used the same enycrpted function, only this time I treated the file as a binary. The file given was a pbm format and I had to make sure to deal with the three line header and than the body.

**Image**

