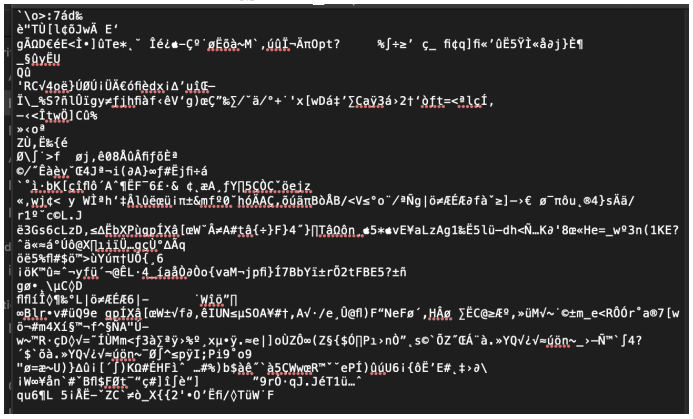
Dhruv Aggarwal
HW2
Aggarw45
1/30

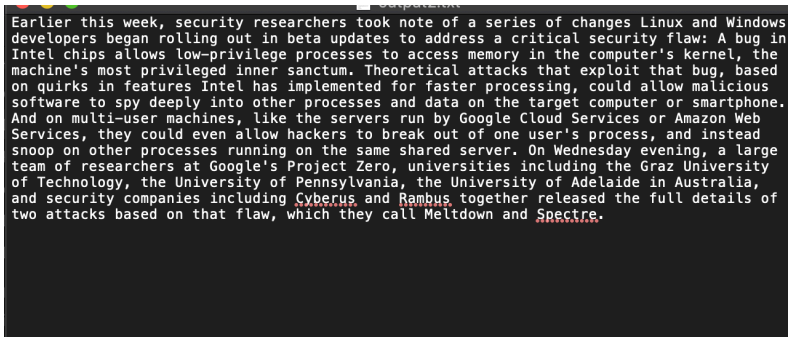
```python

def encrypt(message,key,output):
    file_in = open(key,"r")
    key1 = BitVector(textstring = file_in.read())
    key1 = key1.permute(key_permutation_1)
    round_keys = generate_round_keys(key1)
    bv = BitVector(filename = message)
    output1 = open(output,"wb")
    while (bv.more_to_read):
        bitvec = bv.read_bits_from_file( 64 )
        if bitvec.length() > 0:
            if bitvec.length() < 64:
                bitvec = bitvec.pad_from_right(64-bitvec.length())
            [LE, RE] = bitvec.divide_into_two()
            for round_key in round_keys:
                newRE = RE.permute( expansion_permutation )
                out_xor = newRE^( round_key )
                re_mod = substitute(out_xor)
                re_mod = re_mod.permute(permutation_box)
                temp = RE
                RE = LE ^ re_mod
                LE = temp
            final_string = RE + LE
            final_string.write_to_file(output1)

```

```python
def decrypt(message,key,output):
    file_in = open(key)
    key1 = BitVector(textstring = file_in.read())
    key1 = key1.permute(key_permutation_1)
    round_keys = generate_round_keys(key1)
    round_keys = round_keys[::-1]
    bv = BitVector(filename = message)
    output1 = open(output,"wb")
    while (bv.more_to_read):
        bitvec = bv.read_bits_from_file( 64 )
        if bitvec.length() > 0:
            if bitvec.length() < 64:
                bitvec = bitvec.pad_from_right(64-bitvec.length())
            [LE, RE] = bitvec.divide_into_two()
            for round_key in round_keys:
                newRE = RE.permute( expansion_permutation )
                out_xor = newRE^( round_key )
                re_mod = substitute(out_xor)
                re_mod = re_mod.permute(permutation_box)
                temp = RE
                RE = LE ^ re_mod
                LE = temp
            final_string = RE + LE
            final_string.write_to_file(output1)

if sys.argv[1] == "-e":
    encrypt(sys.argv[2],sys.argv[3],sys.argv[4])

elif sys.argv[1]== "-d":
    decrypt(sys.argv[2],sys.argv[3],sys.argv[4])
```

Encrypted output:



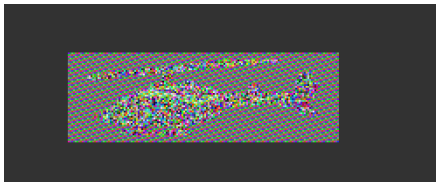
Decrypted Output:



DES_image.py:

```python
def encrypt(message,key,output):
    file_in = open(key,"r")
    key1 = BitVector(textstring = file_in.read())
    key1 = key1.permute(key_permutation_1)
    round_keys = generate_round_keys(key1)
    final_output = open(output,"wb")
    test_file = open("test.ppm","wb")
    with open(message,"rb+") as whole:
        entire=whole.readlines()[0:3]
        final_output.writelines(entire[0:3])
    with open(message,"rb+") as whole:
        non_header = whole.readlines()[3::]
        test_file.writelines(non_header)
    bv = BitVector(filename = "test.ppm")
    while (bv.more_to_read):
        bitvec = bv.read_bits_from_file( 64 )
        if bitvec.length() > 0:
            if bitvec.length() < 64:
                bitvec.pad_from_right(64-bitvec.length())
            [LE, RE] = bitvec.divide_into_two()
            for round_key in round_keys:
                newRE = RE.permute( expansion_permutation )
                out_xor = newRE^( round_key )
                re_mod = substitute(out_xor)
                re_mod = re_mod.permute(permutation_box)
                temp = RE
                RE = LE ^ re_mod
                LE = temp
            final_string = RE + LE
            final_string.write_to_file(final_output)
```

Decrypted Image:



Explanation:

The way I did it was by using the professors script and following his notes. I followed the steps of DES. I basically encrypted in chunks of 64 bits. Split the 64 bits into two halves and then did expansion permutation and substitution and then expanded it again. Added the two haves back. For decryption I used the same code as encryption except I reversed the round keys. For the image I read the header bits and added it to the final image. Then took the rest of it and made a temp.ppm. Then I followed the same steps as encryption of the text except with the temp.ppm and added that to the final image. I know in the professor's pdf it says that you shouldn't have to use the same method and that there is an easier one but I could not figure that out.