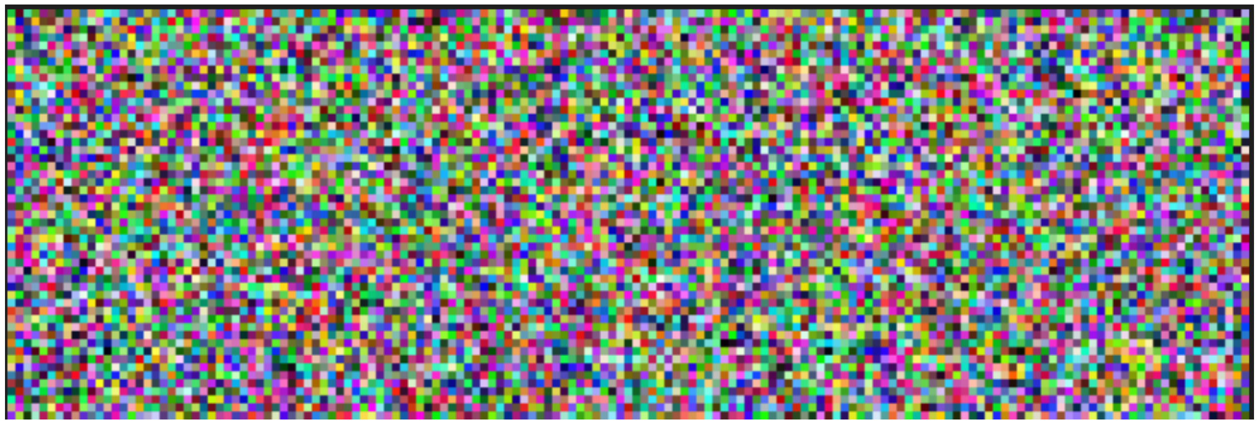
x931(v0, dt, totalNum, key_file)

This function first generates the substitution tables for the aes encryption. Then it iterates for the totalNum variable which is the amount of random numbers it wants. For each iteration, it encrypts the dt input, xors the output of the first encryption with the v0 input, then uses that as the ‘plaintext’ input for the random number encryption. After receiving the random number from the second encryption, it appends the random number to a list. Then xors the random number with the dt_encrypted output to make a new v0. This all follows the diagram shown in Section 10.6 of the lecture notes.

I also created a new aes function which takes in different inputs compared to my solution from hw4. The changes are very minimal in terms of the algorithm and is only different due to the inputs to the function.

cts_aes_image(iv, image_file, out_file, key_file)

This function is very similar to my AES code from hw4. I have functions written in the AES.py submission from hw4 to use as functions for this submission. The first thing we do is initialize the substitution and round key tables. We then grab the header from the image file by checking for 3 ‘\n’ characters. We then run the counter mode AES program. We get an initialization vector as an input from the function. We use this iv and run AES on it using the input key. After the 15 rounds, we then xor it with a 128 bit block from the plaintext, then we have our first output ciphertext block. We then increment the iv by 1 by converting it to an int, incrementing it, then converting it back into a bitvector. We do this for the rest of the file. We then write the output to the output file given in the input.



The main difference is the mode in which we are encrypting the image. In hw 2 we used ecb mode which just took each 128 bit block one at a time until it was finished. This was bad because it was still easy to see patterns in the picture and you could still see the outline of the helicopter. Because we used counter mode for this and xor each 128 bit output of the AES encryption with the plaintext, we can remove those patterns between each 128 bit block which allows for no pattern to be recognized. We also used AES rather than DES for encryption.