HW05 – ECE 404

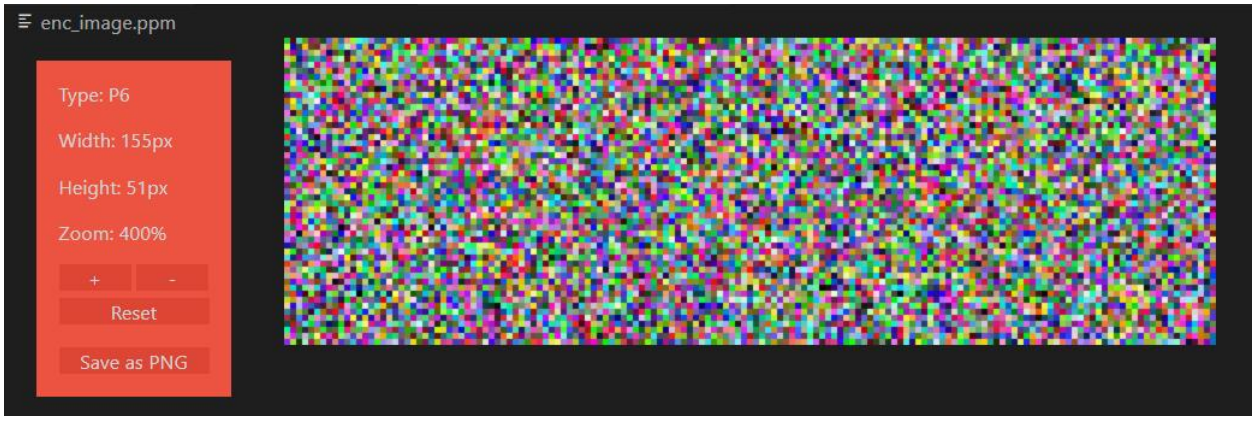
Part 1 : X9.31 Pseudo-Random Number Generator

a. Brief Explanation of Code: The pseudo – random number generator function uses the arguments with the x9.31 algorithm to generate 'totalNum' random numbers as BitVector objects. It takes in 4 arguments: v0 – 128-bit BitVector object containing the seed values, dt – 128 bit BitVector object symbolizing the date and time (which was given to us in this case), totalNum – the total number of random numbers to generate, and key_file – filename for text file containing the ASCII encryption key for AES. It returns a list of BitVector objects with each BitVector object representing a random number generated from X9.31. Primarily using the X9.31 algorithm we run a for loop from 0 to totalNum, and encrypt the date and time with the key_file given. This is stored as a BitVector in dt_bv, which is added to v0 (note that add here means XOR). Then this added result is encrypted using the same key and stored in another BitVector which is now our random number as a BitVector object. Next, we encrypt the xored result of the random number and the dr_bv with the key_file and store it in v0. In this case, v0 denotes the seed for each round. This process carries on until all the random numbers we want are generated. The encryption carried out here is not using triple DES, but it is done using AES.

Part 2 : AES Encryption in Counter Mode

a. Brief Explanation of Code: This code implements the CTR mode AES. It takes in 4 arguments: iv – initialization vector, image_file – input .ppm image file name, out_file – encrypted .ppm image file name, key_file – filename containing encryption key (in ASCII). This function encrypted image_file using CTR mode AES and writes the encryption to out_file. We first open the image file and read the first three lines (header) and output them to the output file. Then, the image file pointer is read as a BitVector object and blocks of 128-bit vectors are encrypted each time. First, I run a for loop to compute the required initialization vectors for the encryption. Once I have created a list with all the initialization vectors, I run another for loop that encrypts the initialization vector using the AES encryption algorithm function, along with the key_file provided. This is then added (note that add here means XOR) with the plaintext BitVector (which is the image file represented as a BitVector object). The result is stored in a block which is a BitVector object and then written to the output file.

b. Encrypted Image :



c. Difference between AES_image() and DES_image() :

In AES image encryption we used CTR mode, also known as the Counter Mode. This retains the pure block structure relationship between the plaintext and ciphertext. For each 128-bit input plaintext block, the scheme produces a 128-bit ciphertext block. Furthermore, the block cipher encryption algorithm carries out a 128-bits to 128-bits transformation. No part of the plaintext is directly exposed to the block encryption algorithm in the CTR mode. The encryption algorithm encrypts only a b-bit integer produced by the counter. What is transmitted is the XOR of the encryption of the integer and the 128 bits of the plaintext. However, the DES image encryption directly exposes the plaintext (image file) with the encryption algorithm. In fact, the Feistel implementation of the algorithm in DES makes the two encryption algorithms wildly different. The DES encryption algorithm does not fully encrypt the image making it very obvious what the image is, which defeats the purpose of encryption. However, the AES image encryption renders the image in such a way where it is impossible to visually decipher what it is. The DES encrypted image shows the outline of the helicopter due to the clear visual of the background. However, the AES encrypted image does not show the difference between the background and the helicopter. Additionally, in terms of algorithms the AES encryption runs faster and performs the XOR operation of the on the plaintext blocks with pre-computed 128-bit blocks.