Daniel Palumbo
Project Milestone

Encryption of a PNG file using 128 AES CBC Encryption(Java)

After implementing the Advanced Encryption Standard algorithm on plaintext, it is easy to see that the complexity of implementation is not in the algorithm alone. There are many steps that need to be taken if the data that is to be encrypted does not start as plaintext. When implementing AES on anything you will be inputting a 16 byte string of plaintext, therefore, when encrypting anything that translates to more than 16 bytes of hex strings the algorithm must be done as an iterative approach. In this case, an image file was chosen in the format of a PNG(Portable Network Graphic). I have personally never worked with a PNG file so this was luckily a standard that was well defined after a little research. After learning that the PNG file was made up of a header and "chunks" that are expressed as varying sizes of byte, it was planned to just read in the PNG file to then convert it to a byte array. Once the byte array containing the PNG file was created, a string array was made correlating to the hex values of the bytes of the PNG file.

I created a file called PNGHelper that has a constructor that takes a key and a path argument. When a PNGHelper object is created, the key is stored as a field and then the file at the path is read then streamed into a byte array which is then converted to a string array containing hex values. The key and the array of hex strings are then passed to an AES object constructor which does the remainder of the encryption and decryption. After some searching I quickly learned that you can't really effectively encrypt image files using the standard mode of operation ECB since there are many repeated bytes thus CBC(Cipher block chaining) was the chosen mode. With this mode there is a random 16 byte String that is XOR'd with the first block of plaintext and then the previous ciphertext is XOR'd with subsequent plaintext entries. Since there is an operation on each of the entries of plaintext into the AES algorithm, the total data needs to be divisible by 16 and thus must be padded if it is not.

Currently I’m at the point where I have the data of the PNG file ready to encrypt and I have been able to encrypt the first 16 bytes successfully. My next steps are to work on encrypting the rest of the bytes, write the encrypted PNG to another location and then work on decryption. I’m just a little bit behind where I would like to be. I figured I would have the whole file encrypted by the time of the Milestone and then the second timeline would just be working on decryption which I don’t think will be extremely difficult. If time permits, I would like to maybe try some other file types or modes of operations. The biggest struggle for me has been organizing how the different parts of my project interact with one another. I may change this up a bit because I really don’t have a way of returning the encrypted image other than writing it to my computer. I may decide that the PNGHelper class is not needed but for right now it will stay until I get my decryption working.