Chan Ng Cashin
ECE 404
HW 06

Problem 1:

To implement RSA encryption in rsa.py I first had to generate values for p and q using the given PrimeGenerator class. With these values, I checked if they met certain conditions needed for RSA encryption. In my encryption function I took these values and created a modular to encrypt with. Using this modular, the value that was given for e, and the bitvector of the plaintext I was able to encrypt the message.

For decryption I took the encrypted text file produced by the encryption preformed and calculated values necessary for decryption. I calculated the private exponent d by finding the multiplicative inverse of e and the totient. The totient was calculated by multiplying p-1 and q-1. Next I found the multiplicative inverse of p and q and used those values to calculate Xp and Xq. I then calculated Vp and Vq and used these values, along with the Xp and Xq values and n value to decrypt the encrypted file.

Problem2:

To implement part 1 of problem 2 I essentially created three n, p, and q variables for encryption. Using the encrypt function I made in the previous problem, I created three different encryptions of the plaintext using these different sets of variables. I also created a file that listed all the modular n values used for these encryptions.

To implement part 2 of problem 2 I used the 3 n values to calculate the overall N by multiplying them together. I then calculated the new N values for decryption. From there I found the c values by multiplying these newly calculated N values with their multiplicative inverse. Using these c values and the solve_pRoote class I was able to calculate the decrypted bit_vector and added it to the plaintext.