Homework Number: 6

Name: Andrew Wu

ECN Login: wu1795

Due Date: 2/27/24

Question 1:

I started out by creating a function that could generate two prime numbers p and q. Inside a while loop that would loop until p and q met certain conditions, I called the findPrime function from PrimeGenerator and generated two prime number, p and q. From there, I anded the first two bits of both p and q with one, and set them equal to variables. For the certain conditions that must be met for the loop to exit, I made sure that the two MSBs of p and q were set, that p and q did not equal each other, and that the gcd of p – 1 and e (65537) and the gcd of q – 1 and e were equal to 1. After finding two prime numbers, p and q, I moved to the encryption of the plaintext where I multiplied p and q to find n. I then casted the input as a bitvector and read in blocks of 128, and padded the block if it was not equal to 128 bits. I then calculated the modulus by setting the block of 128 bits and put it to the power of e, then modded the answer by the modulus n. I then turned the bitvector into 256 bits and wrote the output to hex. For decryption, I first calculated n, the totient, and d. From there, I looped through the entire ciphertext and followed the CRT, calculating Vp, Vq, Xp, and Xq. I then multiplied all those numbers together and modded the answer by n. I then turned the bitvector back into 128 bits and wrote to the output in ASCII.

Question 2:

For encryption, I calculated three separate p and q prime numbers each using the same prime generator function from question 1, and thus calculated three different n numbers. I then wrote the three n numbers to the n_1_2_3.txt file, and pushed the three n values into a shortened version of my encryption function from RSA, producing three encrypted files. For the cracked function. I pulled the three n values and the three encrypted hex numbers from the inputs. I then calculated N, Ni, and the multiplicative inverse of Ni. I then read and sliced the three encrypted texts by 256 bits and calculated m^3 mod N via the CRT. This calculation multiplied the text block with Ni and Ni^-1 and summed the three encrypted texts together for each block. I then took the modulus of the summation by modulo N, used solve_pRoot to calculate the cube root of the sum, and outputted the cracked message. I then wrote the message to the file and repeated the process until all of the blocks of the inputted encrypted texts have been read.