**HW06: RSA**

Homework Number: 06
Name: Jack Gardner
ECN Login: gardne97
Due Date: 3/1/23

**Part 1 Explanation**: In this implementation of RSA, the p and q keys are predetermined and inputted by the user. They are multiplied together to yield n, and they are used to calculate the totient of n, as in (p-1) * (q-1) = totient of n. In encryption, the message is exponentiated by the public key e, which is predetermined. The result is stored in a 256 bit long BitVector and is written to the output file as a hexstring. During decryption, p and q are also read by the user and are used to calculate the totient of n. The private key, d, is calculated as the multiplicative inverse of e in mod totient(n) arithmetic. Each 256 bit segment of the hexstring is read from the input file. From there, the Chinese Remainder Theorem is used to calculate the massive exponent which results from raising the ciphertext to the power of d. This yields the plaintext because e and d are multiplicative inverses of each other, and thus e ^ d is equal to 1. The resulting message is then written to the output file.

## Part 2 Explanation:

In order to crack RSA for e = 3, first three sets of keys are generated and are encrypted using those keys. The n values are written to a file. This yields the three ciphertext files we need to crack RSA. From there, the encrypted files are read simultaneously 256 bits at a time. We then need to calculate M^3 modulo N, where N is the product of n1 * n2 * n3. CRT allows us to do this in a computationally efficient way. A set of three n values and a set of three ciphertext values are passed into the function. We then calculate N as n1 * n2 * n3. For each set of n_i, m_i where n_i is the key used to encrypt m_i we calculate the p_i value as N divided by the n_i we are using. We multiply p by its multiplicative inverse in module n_i, and then multiply that by m_i. We repeat this for all 3 tuples of n and m and sum the values for each. We can return this value modulo N. Since this result is M^3, we cube root the value we have found to yield the cracked text. This value is then written to file, and the next set of ciphertext is cracked.