Deacon Steiner COSC 3020

Lab 3 Explanation

I. METHOD:

For this lab, the method was pretty clearly laid out. There weren't many choices to make. The only thing I considered was reusing the Euclid function (labeled gcd in this lab) from Lab02 to save some time writting code.

APPROACH:

I implemented the functions pretty much as described in the lab document. I added the helper functions which are used by the rsa functions later. Keygen does the calculations needed to obtain the keys and returns (public, private) as a tuple so it can be destructered in main. Encrypt and decrypt both use pythons build in pow() function to do efficient modular exponentiation.

REASONING:

This implementation is sufficiently fast for the scale of this assignment. There are a few improvements to be made in a real world context with large values of p and q. But with 61 and 53, or similar small primes, this is a fine trade-off. Especially due to the necessary overhead required for real-world rsa implementations.

REFLECTION:

The test case I used for the whole program was [92, 7, 100], then, I was able to encrypt and decrypt to compare the recovered grades post decryption to the known grades. This allowed for verification of the algorithm overall.

Questions:

1. Why can anyone encrypt a grade using the public key, but only the instructor can decrypt it?
   Since the public key provides knowlege of n and e, anyone can encrypt the message, however, to decrypt, you must use d and n, and d can only be computed by factoring n into prime factors p and q. This is extremely difficult for modern computers.
2. If an attacker can see e, n, and the ciphertext on the server, what hard problem are they stuck with?
   They must factor n into p and q, which as previously mentioned is very complex with modern computers.
3. Why is using small primes (like 61 and 53) insecure in real life?
   Because, even though prime factorization is a complex problem, the amount of checks needed to find the prime factors is small enough that it can still be done in reasonable time.