RSA Cryptosystem Program

The program implements the RSA cryptosystem, requiring a single file containing a message without NULL characters and providing the public and private keys, the ciphertext before decryption, and the resulting decrypted message in files. The major components include the key generation, prime number generation, modular exponentiation, Extended Euclid's Algorithm, message encoding and decoding, and encryption and decryption. The key generation relies on the prime number generation for the large prime numbers needed and the Extended Euclid's Algorithm to get the private key for decryption. The prime number generation relies on modular exponentiation to test if a number is prime using Fermat's Primality Test. The encryption and decryption both rely on the modular exponentiation algorithm to compute the cipher text and plaintext respectively, using built in exponentiation would take too long to compute.

The algorithms used were the modular exponentiation algorithm by squaring found in the book, Extended Euclid's Algorithm, an encoding algorithm by taking k0 = m0 + m1*256 + m2*256^2 + . . . + ms−1*256^(s−1), and a decoding algorithm by taking mi = ((encoding mod 256^(i+1)) - (encoding mod 256^i)) / 256^i.

After each module was completed, I tested them to make sure that individual part produced something looked correct. Since the key generation and encryption would be difficult and time consuming to check before decryption was finished, not much time was spent to test for correctness until after the program was finished. I also tested the encoding and decoding without any encryption to ensure that they worked. I first tested small messages like "foo" to make sure that they worked, then tested large message of length > 82 to make sure there was no error in the separation and concatenation of the blocks of the message. After the program was finished, I tested it with a small message first, checking each file to make sure they at least looked reasonable, and then tested with the given test case in the assignment pdf file.

The implementation of the keys and the encryption and decryption algorithms went smoothly, but building the encoding and decoding algorithms took some time, but I overcame it by realizing that the encoded block can be treated like a number in base 256, and using that knowledge I was able to figure out how to decode it.

The program I believe will work for any input that contains only ASCII characters, but does not contain the NULL character. This character is used to determine the end of the decoding algorithm.

The program does not need to be compiled. Python 2.7 or higher is required to run the program.