Jasque Saydyk and Emalina Bidari

Professor Elwakil, Melton, and McCarty

CS 136L Section 3801

14 February 2017

Lab 04 - RSA Encryption

1. Problem Statement

In this lab, we were tasked with making a RSA encryption system that could generate public and private keys, able to encrypt a number using the public key, which could then be decrypted using the private key.

Requirements

1. Generate public and private keys using examples in Wiki page on RSA encryption
2. Generate public and private keys using random BigInteger values
3. Encrypting a number using the public key
4. Decrypting that number using the private key

2. Planning

Most of the plan is laid out in the requirements, with requirements one and two indicating that you should have a method that generates keys and two constructors, a default one that randomizes the initial values and another that allows you to customize what those values are. Requirements three and four are to be both separate methods.

The difficult part of this is going to be understanding the math and being able to translate that math into code, especially during the generating keys portion of the program.

3. Implementation and Testing

Implementing this was straight forward, as the most difficult part of the project was understanding the steps involved with making the calculations, then translating that into code. As for testing, we followed the wiki's test values of p = 61 and q = 53 and got the correct answers, and the encryption and decryption of test values work for both the test and randomly generated values, as seen below. Although I am confident the code is implemented correctly, due to the lack of an expansive testing system, I cannot guarantee it.

Result of Tests

```
Prime P: 61
Prime Q: 53
modulusN: 3233
totient: 780
Public Key Exponent: 17
Private Key Exponent: 413

Encrypted value: 2790  Expected Encrypted Value: 2790

Decrypted value is: 65  Expected Decrypted Value: 65

Prime P: 878202974489725038283453967069464900187421823646675285531293743006113788620 31
Prime Q: 845513779412596462733104657923869046826842176675153225735619056504498023007 03
modulusN: 7425327160521914546261143342190266523954738166437611291338232364881010639924370422470685789922509 90
totient: 8634101349444086681699003886267751772040393216787920106207246935908151906888602384645692509037684006
Public Key Exponent: 17
Private Key Exponent: 6602548090751360403652179442440045472736771283426056551805541774517998517032460647082 00

Encrypted value: 659974359083659205093383789 0625

Decrypted value is: 65  Expected Decrypted Value: 65
```

Less Cluttered Result of Tests

```
Encrypted value: 2790  Expected Encrypted Value: 2790

Decrypted value is: 65  Expected Decrypted Value: 65


Encrypted value: 87507831740087890625

Decrypted value is: 65  Expected Decrypted Value: 65
```

4. Reflection

In retrospect, math is hard. This project required that one agonized over every word in the example in the wiki and understand what they mean. Ultimately, this lab tests one's ability to read complex requirements and implement them into code.