RSA Group Project

Andrew Farmer, Clayton McEntire, Chandler Richmond

University of Central Arkansas

Algorithms CSCI 3330

Dr. Hu

Date: 2/16/2022

# Introduction

For this project the use of several algorithms is required for it to correctly Encrypt and Decrypt user input strings. Starting out with generating two large integers and determining if they are pseudo prime using Fermat’s test. Then a relatively prime is found from the pseudo primes using Euclid's GCD, and d is found from the Extended Euclid algorithm. From these elements we have found, RSA encryption and decryption is now possible.

**Member Contributions / Responsibilities**

|  |  |  |
| --- | --- | --- |
| Andrew Farmer | Clayton McEntire | Chandler Richmond |
| Responsibilities:  Git Hub Creation and  Management | Responsibilities:  Group Communication | Responsibilities:  NONE |
| Contributions:  Key Generation  Euclid’s GCD  RSA Decryption  RSA Encryption  Digital Signature  Project Report | Contributions:  Rand Prime Generation  Extended Euclid  Fermat’s Test  UI Elements  RSA Encryption  Project Report | Contributions:  NONE |

**Solution Design**

The project was divided into 3 main tasks, Key Generation, RSA Encryption/Decryption, and Digital Signature and Digital Signature Authentication. We were able to implement all of the functions except for Digital Signature Authentication.

**Problem Analysis and Algorithm Identification**

Key generation used the Fermat’s test to determine if the p and q generated were pseudo primes. The Fermat’s test was run 200 times to ensure accuracy, If the number was determined to not be a pseudo prime, they were regenerated. Euclid’s gcd and extended gcd were called to generate e and d for encryption. From there we have our Public and Private keys.

RSA Encryption received the keys e, d, and phi = (P\*Q), from there the RSA encryption algorithm can be completed using the pow function and converting the string to ASCII code using

**Implementation**

The project was Implemented using Spyder Python, and the providedcourse materials.

Clayton worked on Key Generation in the project along with encryption of the input text. He implemented the Extended Euclid’s algorithm and determining the pseudoprimes with the Fermat’s test. To better understand proper implementation of the algorithms he went to tutoring with Tarrant at the UCA Library as well as Office hours to understand finding d. He implemented the RSA encryption of the text. Clayton also worked with Andrew in the library to implement core functionality.

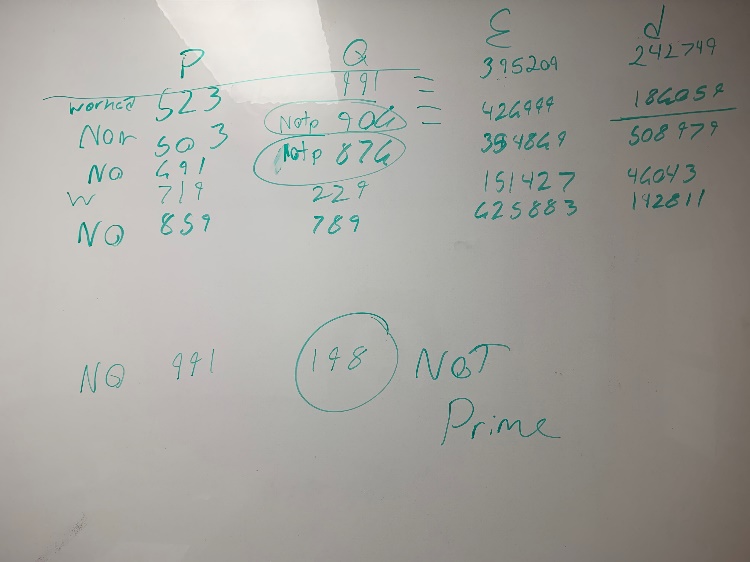
Andrew worked on key generation as well as decryption of the text and message authentication. He helped to implement finding e and d from our generated pseudo primes from the lecture notes. After he had all major bugs fixed, he went onto implementing the digital signature while Clayton worked on the project report.

Chandler contributed nothing to this project. Was in the group text and after a week of nothing from him, he texted saying he had had the flu and requested a new GitHub link be sent. A zoom call option was offered to work together, and he declined. He added his name to the top of the code and added a few unnecessary print, both were later removed.

**Testing**

We went through a lot of testing during the implementation of our code. The hardest algorithms conceptually to understand were the generating e and d, we had no good way of seeing if they worked correctly when implementing so we ultimately ended up implementing RSA encryption and decryption using (7,115), (63,115) as our keys.

Once we had the encryption and decryption working, we went back to figuring out what was wrong with the way we were finding q. After fixing the major problems we found the program would work properly some of the time and not function.

To try and deduce what was wrong we created a table with the P, Q and E, D that were being generated. After checking the P and Q online with an [encryption calculator](https://encryption-calc.herokuapp.com/) the q’s being generated were not prime meaning encryption and decryption were not possible. We then realized we were only checking P in the Fermat’s test and not Q. After running the Fermat’s test on both, we still had minor issues with Q not being prime, we ended up increasing the Fermat’s loop to 200 times and we had no further issues with our main functions.

**Summary**

Overall, we learned how to properly implement several algorithms that worked together to create one fully functional RSA encryption system. It was very interesting to see how the text would be encrypted and decrypted as numbers before being converted back to a string along with how the algorithms worked together to create the encryption. The hardest to get correct was the E and D generation because without it correctly working nothing else in the program would function as it should. While not having one of the functions, due to time constraints and lack of participation by Chandler, we do have a working encryption system.