RSA Cryptosystem Generation of Public and Private Keys in Rust

Dr. Elise de Doncker, Jason Pearson, Sam Demorest

Abstract—The goal of our project was to develop two programs. The first program would be for determining large prime numbers and the second program for using the first application to create public and private key pairs and using these pairs to encrypt/decrypt a message.

Keywords—RSA, rust, key generation, large prime numbers

1 Introduction

THIS project under the supervision of Dr. Elise de Doncker is to implement the numerous algorithms needed to create a public and private RSA key pair set.

2 Research

At the start of our project we first researched what an RSA key is and how to generate them and discovered it only takes a few steps. The first step is to create two prime numbers. The larger the prime number the better the encryption. Then with these two prime numbers we would be able to create a public and private key.

We found two algorithms for finding if a number is prime or not. The first is the AKS Primality Test. This primality test is the best deterministic primality test known in terms of time complexity. This being said it is still very slow when compared to non deterministic primality testing algorithms. So we also looked at the Miller Rabin probabilistic primality tests to also test numbers which runs much faster than the AKS primality test.

Once we had found the algorithms needed to do all the actions we had to select a programming language to use and we chose rust. The main reason for choosing rust is because it is low level and has great memory management built into the language, this would allow our program to go faster which is a must because the time complexity of algorithms is quite high.

3 Design

4 Implementation

5 Testing

6 Result Analysis

7 Goals Reached

8 User Guide

9 Conclusion

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

Dr. Elise de Doncker