RSA Encrypt/Decrypt Matrix Package

Sandeep Kumar Barada

Guided by: Dr. Panchatcharam Mariappan

Abstract

In the era of data, privacy is paramount. To ensure the security of sensitive information, I have developed a model using the RSA Algorithm for encrypting and decrypting messages. The RSA algorithm utilizes a public key for encryption and a private key for decryption, both of which are generated from a large integer that is the product of two large prime numbers. By applying Euler's phi function on this product, we derive ϕ_n, a critical component for key generation.

1 Introduction

To streamline the implementation, I created a header file for all the logic and computations, complemented by a `project.cpp` file for practical execution. For computational efficiency, I wrote my own GCD function using Euclid's algorithm, which ensures faster computation. Prime number generation is simplified by multiplying all primes less than a certain prime and adding 1, although fixed primes are used for simplicity in this model.

2 Creating a Header File

To streamline the implementation, I created a header file for all the logic and computations, complemented by a `project.cpp` file for practical execution.

3 Creating Own GCD and Swap Function

For computational efficiency, I wrote my own GCD function using Euclid's algorithm, which ensures faster computation. Prime number generation is simplified by multiplying all primes less than a certain prime and adding 1, although fixed primes are used for simplicity in this model.

4 Generating Prime Numbers

We can generate our own prime number by multiplying all the prime numbers less than a particular prime number and adding them by 1. For simplicity purposes, I choose some fixed prime numbers.

5 Creating a Class

To access a limited amount of given user data, I created two kinds of variables: private and public. I used a class named `Avenger` to input all the details of avengers. In the private section, I created a `piv_key` which is of integer data type and a vector named `Piv_vec` which stores integer values. In the public section, I created an integer variable named `pub_key` which is the public key of an avenger and a vector named `pub_vec` which stores a set of possible public keys for a given ϕ_n.

6 Generating Set of Public Keys

Run a for loop starting from 0 up to ϕ_n and increment by 1 each time. Now check whether $\text{GCD}(i, \phi_n) = 1$. If it is one, then put that element in `pub_vec`.

7 Creating a Function to Get Public Key

Ask the user to input a random number and assign that number to `pub_vec`. The element you get from that assign that to `pub_key`.

8 Creating `piv_vec`

Run a for loop from 1 to ϕ_n such that if $\text{GCD}(d \cdot \texttt{pub\_key}, \phi_n) = 1$, then put that into `piv_vec`. Get a `piv_key` from that. We can get a matrix of private keys by running two for loops. The first for loop runs for accessing elements of public elements and the second loop runs to check $\text{GCD}(d \cdot \texttt{pub\_key}, \phi_n) = 1$ and put that into the private key matrix.

9 Conclusion

This RSA Encrypt/Decrypt Matrix Package provides a secure and efficient method for data encryption and decryption. The use of a header file simplifies

the implementation and ensures that the logic is well-organized and easy to maintain.

Acknowledgements

This work was guided by Dr. Panchatcharam Mariappan and developed by Sandeep Kumar Barada.