RSA encryption

Dongjing Wang

December 27, 2022

1 Files

- `.gitingore`: Since generally only files related to the program are uploaded, I added .gitignore to let Git ignore the configuration file of VSCode and the binary files generated after compilation.
- `CHANGELOG`: In order to better debug and plan the coding process, I will record every change I make in the CHANGELOG file.
- `functions.cc`: This file contains all the functions needed for RSA encryption and decryption. This file defines them.
- `functions.hh`: This file contains all the functions needed for RSA encryption and decryption. However, this file only declares all the functions used.
- `LICENSE`: Since I want to legally protect my work, I added LICENSE to my repository.
- `main.cc`: This file includes the scheduling of functions and UI construction for all `functions.hh` files.
- `Makefile`: I have formulated the necessary ”rules” in the Makefile to use the ”make” file according to the needs of my compiled project
- `NOTE.md`: Many times I will forget some previous ideas, so I created a `NOTE.md` to write down what should be paid attention to when making changes next time, and what things must be modified next time.
- `Rabin-Miller Algo.jpg`: When I was using regular data types (int, long, long long) I needed to implement many operations by myself. Verifying whether a number is prime is one of them. Since I still can’t understand the Rabin-Miller algorithm, I just use a flowchart to implement it step by step. However, since I used the GMP library, this is not as useful as before, but I still keep it.
- `README.md`: This file contains a brief introduction to my project. For example, the introduction of usage and functions.

- `WRITEUP.tex`: This file contains the source code that generated this file.
- `WRITEUP.pdf`: This is the document you are reading.

2 Design

2.1 RSA encryption

RSA encryption must first generate two super large prime numbers (usually 1024 bits, 2048 bits, or 4096 bits). My program defaults to 4096 bits. I wrote a helper function to achieve this. After this, we need to find ϕ, which is $(p-1) \times (q-1)$. And n, which is $p \times q$. Since it would be very difficult to do with regular data types (int, long, or long long), after encountering this problem I decided to rewrite the program and use the GMP library for the second pass. After this, we need to find e, which is a number relatively prime to ϕ. In order to realize the hierarchy as much as possible, I also modularized this part of the code into a function. The last step is to find d, that is, $d = e^{-1} \bmod \phi$. In this way, the variables required by the public key pair and private key pair are all calculated. Of course, if the user indicates that the public key pair required for encryption already exists, the previous steps can be directly skipped. The last step to do is encryption. And before that we also need to convert the string to a 256-base number. Similarly, I used a helper function to perform bit operations. After that we can encrypt this string of numbers. Encryption (the function of) is based on $c = m^e \bmod n$.

2.2 RSA decryption

The formula followed by decryption is $m = c^d \bmod n$. The first step is to ask the user to provide private key pair and encrypted information. However, there are two major differences from encryption. The first point is that this program only accepts file input because it is afraid that the user will make a mistake in entering the private key pair and encrypted information in the standard input. The second point is that since the decrypted result can be read directly, the result is printed directly to standard output instead of text.

3 Discussion

My first exposure to cryptography was in high school. Since I had no exposure to programming before college, I was just trying to understand some cryptography. And the second contact is in a class at the university. At that time, I first came into contact with RSA encryption and decryption. However, it was still too difficult for me at that time. Although I managed to finish the outline with the help of TA and tutors, I still didn't know the specific details at all. Now, a year later, I'm trying to write this project from scratch again, and it brings me a completely different experience. The last time I studied mathematics systematically was when I took a compulsory course in mathematics in college last year. But at that time, I also learned some multivariable calculus. And now the mathematics required for cryptography is a bit baffling to me. There seem to be many strange calculation methods in cryptography, such as *Modular Multiplicative Inverse*, *Modular Exponentiation*. Although these calculation methods are not incomprehensibly difficult, they still have a certain interference effect on clarifying the project ideas.

I lost the PDF document that my professor provided in my previous class. There is some methodology about this RSA encryption above. This led me to learn about RSA encryption and decryption from scratch. To be honest it was quite painful for me. It was the same when I once tried to teach myself machine learning. When it comes to theory, many mathematical formulas that I can't understand are mentioned on the website or in lectures. But later, I try to sort out what I have to do slowly, and turn what needs to be done into a list. And use the hierarchy to sort out all the necessary things from the broadest to the details. While I didn't use DESIGN.pdf in the format I was once taught, I did, and did appreciate the importance of designing before I started.

After doing this, I found that I was no longer doing projects just for the sake of doing projects. I really enjoyed the learning process.