1e814a805d4782629cf47383dc12b46d6

Schmidt-Samoa Cryptography

Private Key to Decrypt Title

Arnav Nepal

February 27, 2023

1 Introduction

In this assignment we were tasked with implementing the Schmidt-Samoa algorithm for public key cryptography in C. Schmidt-Samoa is a theoretically more consistently secure (albeit less efficient) cryptography method than RSA. SS generate pair of keys - public and private keys - to perform encryption and decryption. The public key, composed of key n (a product of primes) and a username, is used to encrypt messages. After the encryption, data can be transmitted across an insecure digital medium to the recipient, who can then decrypt the message using the corresponding private key. Keys are generated by the user, and the public key is distributed freely while the private key if kept by the user.

2 Lessons Learned

This assignment served as an introduction to using external packages and tools to help in creating programs. We had to use the GNU multi precision (GMP) library to work with very large integers, which is necessary when it comes to efficient cryptography. We also learned the basic of cryptography and modular arithmetic, which are closely related.

2.1 Modular Arithmetic and GMP: numtheory.c

In numtheory.c, we implemented a small library of mathematic functions to perform the mathematical operations necessary for SS cryptography. We had to implement functions to obtain the modular power and modular inverse of the primes we generated in order to generate components of the public and private keys. Implementing this file taught me the importance of first designing my code in a way I can understand before using any external packages. I followed the recommendation by course staff to first make all the functions within numtheory.c in pure C code, which turned out to be very helpful. In addition to making the GMP translation process very simple due to having the code to translate right there, it also made it much easier

to find errors in my code and debug since I was able to "follow" the GMP code by looking at the commented out C code it was trying to emulate.

In retrospect, I think not writing out the logic of this file in pure C last quarter is what caused me to fail this assignment, as I had errors come up in the last minute in numtheory.c. I found it difficult to follow along the GMP code when I had a last minute error come up (of course, I also did things last minute last quarter. I've been able to change this quarter, though!).

2.2 Public Key Cryptography: ss.c

In ss.c, we implemented the bulk of the logic of the program. By comparison, keygen.c, encrypt.c, and decrypt.c have little to no unique logic, and mostly consist of calls to ss.c functions. This makes sense, as ss.c contains the functions to actually encrypt and decrypt functions. Implementing this library taught me the importance of breaking down problems in an understandable and efficient way. This is most pertinent to the file encryption and decryption methods, which asked us to encrypt and decrypt us in blocks of text. While I found them a little bit challenging to implement due to needing to use fread() and mpz_import() (which were somewhat difficult to understand), breaking down text into blocks made it easier to understand what was going on. It also helped me debug because I was able to print out ever block, which helped me find bugs within the loop (since they repeat themselves), which might not have been as easy if we were required to just generate a large enough key to encrypt everything in one block.

3 Schmidt-Samoa, RSA, and Cryptography in Our World

This assignment was very informative in regards to teaching me about cryptography. While I had some basic knowledge of cryptography prior to this assignment, going in-depth with the mathematics and actually implementing it was a very insightful experience. It;s an important topic to understand because it is extremely important in the real world. Most of the internet now uses the the HTTPS protocol instead of the insecure HTTP, which encrypts all user data during transmission. Of course, ISP's and internet browsers might still be able to track our activities, but having a layer of protection at the very least protects users against malicious middlemen. Furthermore, cryptography provides protection and privacy from the government and state actors. Many governments, even in the "free" world have tried (and in many cases succeeded) in outlawing or limiting secure communication for civilians. However, it is important to note that cryptography is not the be-all and end-all form of privacy; authoritarian governments (e.g. China) in some cases don't care about encryption since they have already back-doored the devices that contain the data. Nevertheless, Cryptography is an extremely important layer of security in protecting our information and privacy.

4 conclusion

This was the most interesting assignment thus far. I really enjoyed learning about and implementing the Schmidt-Samoa algorithm because it felt like I was gaining an insight into how my own data has been protected all these years.

This assignment also made me further understand the importance of designing my code before writing it down. My initial design was not the best, and did not include a few functions from ss.h (including big functions like ss_encrypt_file). However, I forced myself to go back and update the design before I began writing down code for those functions. I only did it because professor long had told us that it was much easier to write code after writing down the design for it - which I didn't fully believe. For previous assignment, we were given extensive psuedocode, and making functions was almost always a simple translation from psuedocode, so I think I found the design not too helpful. However, this assignment made me realize how important the design is in guiding my thought process while writing down code, especially in ss.c where we were given relatively less specific instructions in designing the functions. Even when my design got things wrong, it was able to guide my thought process into the next step, whereas I might have just blankly stared at the code before.