Cracking the Code: Schmidt-Samoa
Assignment 5 Writeup

Sam Leger

February 26, 2023

Professor Long
CSE13S Winter Quarter

1 Schmidt-Samoa

The Schmidt-Samoa Cryptosystem is a public-key encryption algorithm that was created by Michael Shmidt and Michael Samoa, in 1986. Its strength lies in the difficulty of factoring the product of two large prime numbers, which is considered an NP problem. It is very difficult to calculate, but once shown a solution, it is easily verifiable. In this system, each user has a public key and a private key. The public key is used to encrypt messages that can only be decrypted with the corresponding private key, which is kept secret by the user.

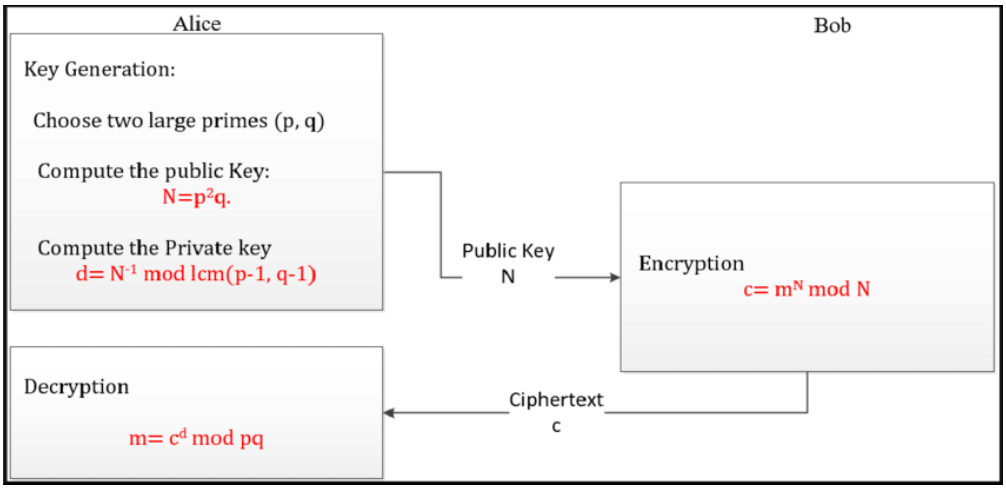

Figure 1: Schmidt-Samoa Cryptosystem

2 Numtheory

The `numtheory` module was the most intensive portion of the whole assignment, and that was due to the nature of the `mpz_t` integers that were necessary. Learning the GMP library wasn't too difficult, but making sure variables were properly freed and there were no memory leaks was very tedious. This was the first assignment so far where I experienced serious memory leak issues that took ages to track down. I employed the use of `Valgrind`, as well as `gdwarf-4` during compilation to make debugging easier. This was also the first assignment for which I used scan-build in the `Makefile`. Another reason `numtheory` proved especially difficult was the rigorous testing. I had to write test harnesses for each individual function in the library, and test them thoroughly since even the smallest error would snowball into something greater down the line. I found the `is_prime()` function to be the most difficult.

3 SS

`SS` was definitely a step down in terms of difficulty, and most of the module flew by. The main problems I ran into were regarding `ss_encrypt_file()` and `ss_decrypt_file()`. I struggled with the calculation and allocation of each block, as well as working with the input/output streams. Actually, near the end of the assignment, when I thought I was finished, my decrypted message had occasional filler characters all over the place, and I had to go back and rewrite my encrypt file function. I found that I was using the `getline()` function when I should've been using `fread()`. After the fix, everything worked properly.

4 Keygen, Encrypt, and Decrypt

The three main program files were the easiest to implement. Since I had experience with parsing command line options in the last three assignments, that part was relatively straightforward. The one hiccup I had was on the implementation of `-i` and `-o` argument options. After I figured out how to properly switch the input/output streams, things were running smoothly. (Until I had to go back to the SS module).

5 Closing Thoughts

Getting to work with cryptography was really interesting to me. When I first attended the lecture on cryptography, I was captivated but also intimidated by the math portion. I liked how relevant it was, being heavily integrated into the world around us, for example, secure communication over the internet (SSL/SSH), digital signatures, and even password encryption. I take advantage of cryptography daily, one example being VPN technology that uses symmetrical key cryptography. The last two weeks have shown me that I am capable of understanding the math that powers basic cryptography, and I really enjoyed implementing my own system.