Assignment 6 Design

Brandon Chuang

11/8/2021

An RSA Key generator, Encryptor, and Decryptor, made in C using the gmp library.

1 Introduction

RSA encryption is an encryption method that involves two keys: a public key (that others have) and a private key (that only you have). It hinges on the idea that multiplying two prime numbers is much easier than finding the two prime numbers that, multiplied together, equal a number you're given. Since there are an infinite number of prime numbers, and these number can get very large very quickly, messages encrypted with RSA can remain unbroken for longer than it is worth it to decrypt, or longer than is even possible. The messages will be encrypted with the public key, and have to be decrypted by the private key.

2 RSA Math/Number Theory

RSA encryption works because, to our knowledge, multiplying two primes is much easier than factoring two primes out of their product to the degree that it would take too much energy to even process once the amount of bits becomes too large. Therefore, we need to have a variety of ways to create, manipulate, multiply, and mod prime numbers for RSA encryption to work.

2.1 Creating Primes (Miller-Rabin Primality Test)

The method used to find primes is to randomly guess numbers and check each one to see if it is prime. To see if it is prime, we will be using the Miller-Rabin Primality Test. It is important to note that this test is not definitive: it can only determine if it is definitely not prime or if it probably is prime. The likelihood of it being wrong goes down based on the amount of iterations of it ran, so it becomes very improbable for it to be incorrect.

2.2 Creating the Necessary Keys/Values

We will create two primes, both of which will determine specific different numbers:

- public modulus: This number will simply be the two primes multiplied together.
- public exponent: This number will be a random number around the same size as the modulus in bits that is coprime with the totient ((p-1)(q-1) where p and q are the aforementioned primes). Being coprime means that the greatest common divisor between the two numbers is 1.
- private key: The private key is the modular inverse of the totient.
- signature: The signature is just the username (written in base 62), raised to the private key's power and modded by the public modulus.

3 GMP Library

This library allows for the use of arbitrarily large numbers. In C, you are capped at only using 64 bits to write numbers. Wile this is big, we need larger numbers for encryption. The GMP library includes a type known as mpz_t, which allows us to go beyond 64 bit numbers.

4 Key Generators (Keygen)

The key generator will create two files: the public key and the private key. Keygen will create 2 primes for the public file, a public modulus, and a public exponent for the public key file. It will also create a prime number for the private key file, and also write in the public modulus. The public key file will include the public modulus, the public exponent, the signature and the username. The private key file will include the public modulus and the private key.

5 Encryptor

The encryptor takes the public key and encrypts the message. It will first read through the public key and verify that it is a valid key. Then, it will encrypt the contents of the input file given and write it to the output file.

6 Decryptor

The decryptor takes the private key and decrypts the file given as input. It essentially does encryption but backwards.

7 Input

The encryptor and decryptor will have the same exact commands:

- -h: prints out help.
- -n: specifies the public key for encrypt and the private key for decrypt (defaults: rsa.pub and rsa.priv respectively).
- -i: specifies the input file (default: stdin).
- -o: specifies the output file(default: stdout).
- -v: enables verbose printing (prints the public modulus and the private key).

The key generator will have more unique commands:

- h: prints out help.
- b: specifies the minimum amount of bits the required for the public modulus.
- i: specifies the amount of iterations done by the Miller-Robin primality test.
- n: specifies the output file of the public key (default: rsa.pub).
- d: specifies the output file of the private key (default: rsa.priv).
- s: sets the seed for random (default: output of time(NULL))
- v: enables verbose printing (prints the username, the signature, the first large prime, the second large prime, the public modulus, the public exponent, and the private key)