RSA Public Key Generation Project: Preliminary Design Document

Sam Demorest and Jason Pearson

CS 6310

Dr. Dedoncker

**Research description**

For our project we will be implementing the generation of the public and private keys that were invented by Rivest, Shamir and Adelman known as the RSA algorithm. The idea behind the algorithm is that there is no efficient way to factor prime numbers. Because there is no way to do this, to crack the key it would take lots of computational power making a secure encryption. The public key is available to anyone and this allows one-way encryption of the data. If the public key is large enough there is no reasonable way to decrypt the key. The only way to decrypt the data is to have the secret private key that is also a very large prime number.

To generate an RSA public/private key pair there are a few steps. First we must create two different prime numbers that are not too close in value, these two numbers are referred to as p and q. These two primes should be chosen at random and primality can be tested using the Agrawal-Kayal-Saxena (AKS) primality test. Next we need to calculate a modulus for the public and private keys referred to as n. The length of n is normally expressed in bits and is commonly known as the key length. Now that we have n, p, and q we need to compute  φ(*n*) = φ(*p*)φ(*q*) = (*p* − 1)(*q* − 1) = *n* - (*p* + *q* -1). Where φ is the Euler's totient function. Now that φ(*n*) has been calculated we need to find an integer e such that 1 < e <φ(*n*) and the greatest common divisor of e and φ(*n*) is one and lastly that φ(*n*) and e are coprime. The value of e is what is used for the public key portion of the encryption and works best if it is a smaller value in most applications. Now that we have all of our calculations we finally have d = e-1(mod(φ(*n*)). From here we can construct our actual private and public keys. The public key consists of e and the modulus n and the private key consists of d and the modulus n as well.

**Possible approaches**

Our goal is to be able to create a key pair that will be able successfully encrypt and decrypt data. We are hoping to use Rust, which is a newer programming language that is still in its early development stages. If Rust does not have all the correct functionality that we need we will convert to a more standard language such as C. It will be essential to have unit tests for our project. For example if the numbers we generate turn out not to be prime the whole thing will be incredibly easy to crack. In the end, our stretch goal is to be able to use our generated keys in a program such as GPG.

**Preliminary design**

To bring this project to life we have divided up the generation portion into five steps.

1. Create the large prime numbers p and q
2. Compute N
3. Compute φ(*n*)
4. Find a good value for e
5. Compute d
6. Generate public and private key files with computed values

**AKS References:**

http://www.cse.iitk.ac.in/users/manindra/algebra/primality\_v6.pdf

http://mathworld.wolfram.com/AKSPrimalityTest.html

**Rust References:**

https://doc.rust-lang.org/

http://rustbyexample.com/

https://github.com/rust-lang/rust

**RSA Cryptosystem References:**

http://mathworld.wolfram.com/RSAEncryption.html

http://stackoverflow.com/questions/12749858/rsa-public-key-format

https://engineering.purdue.edu/kak/compsec/NewLectures/Lecture12.pdf