CS566 – Project Proposal

10/17/2018

**Understanding the RSA Algorithm**

The purpose of this project will be an investigation of the RSA algorithm. This algorithm is named after Ron Rivest, Adi Shamir, and Leonard Adleman (Rosenthal, Rosenthal, & Rosenthal, 2014). RSA is an algorithm that is used in public key cryptography, which is used widely in computing. Public key cryptography is most commonly used to safely transfer symmetric encryption keys.

The basic premise of public key cryptography is that the recipient of the data shares a public key with anyone that wishes to securely send data to them. A public key is comprised of a pair (*N*, *e*) where N is the product of two large prime numbers, *p* and *q*. *e* is an encryption exponent based on the equation below (Smart, 2016).

gcd(e,(p − 1) · (q − 1)) = 1

The recipient holds a private key that is used to decrypt incoming data. A private key is comprised of the elements (*d*, *p*, *q*) where *p* and *q* were the large prime numbers used to create the public key, and *d* which is the decryption exponent. A decryption exponent is calculated by the following method:

e · d = 1 (mod (p − 1)(q − 1))

The encrypted data, or the cipher text, is calculated using the following equation:

c ← me (mod N).

while decryption of the data is performed with the following equation:

m ← cd (mod N).

If *N* can be factored into *p* and *q*, then the decryption exponent, *d*, can be calculated. With this information, the cipher text could be decrypted. RSA-768, which is a 232-digit number, took over two years and hundreds of computer systems to solve. It is estimated that a 1024-bit RSA modulus would be nearly 1000 times more difficult to solve (Aoki et al., 2010). Larger key sizes can be used such as 2048-bit, 3072-bit, or even larger as needed.

My interest in this project is better understanding the implementation of this algorithm due to its wide-spread use. The paper would start with an introduction to the algorithm, where and how it used, and how it is the implemented. An example of the algorithm would then be presented of both encryption and decryption of a trivial message. The project would end with an attempt to implement the RSA algorithm using Python, unless it proves more practical to choose a different language. If time and the length of the paper will allow for it, then I may also decide to briefly investigate an interesting algorithm or two for factoring *N* in the RSA algorithm.

**Resources Used for Initial Investigation**

Rosenthal D., Rosenthal D., Rosenthal P. (2014) Sending and Receiving Secret Messages. In: A Readable Introduction to Real Mathematics. Undergraduate Texts in Mathematics. Springer, Cham

Smart N.P. (2016) The “Naive” RSA Algorithm. In: Cryptography Made Simple. Information Security and Cryptography. Springer, Cham

Aoki et al. (2010) Factorization of a 768-bit RSA modulus. <https://eprint.iacr.org/2010/006.pdf>