Understanding Asymmetric Encryption Algorithms

Colin MacCreery
Western Michigan University
College of Engineering
Computer Science Department
Kalamazoo, Michigan
Email: colin.c.maccreery@wmich.edu

Jason Pearson
Western Michigan University
College of Engineering
Computer Science Department
Kalamazoo, Michigan
Email: jason.j.pearson@wmich.edu

***Abstract*—This paper contains information on how various cryptology algorithms work including RSA public key encryption, ElGamal asymmetric cryptology and Elliptic Curve public key encryption.**

I. Introduction

Writing an introduction sentence is hard.

A. Types of Encryption

1) Asymmetric Encryption: Asymmetric encryption is also known as public key cryptography. These public key systems work by having a public key and a private key. The duty of the public key is to encrypt a message. The encrypted message is only able to be decrypted by an appropriate private key. One major down side of public key systems is that creating the key tends to be computationally expensive.

2) Symmetric Encryption: Symmetric encryption uses a single key for encryption and decryption. The keys on both sides are not always identical but both can encrypt and decrypt a message. Typically a symmetric encryption uses one of two methods stream ciphers or block ciphers. Stream ciphers encrypt data as it comes typically a few bytes at a time. For block ciphers a number of bits is taken at a time and is encrypted. If the data is too small to fit into a block it is padded so that it can fit into a block. Some common algorithms that use symmetric encryption are AES and blowfish.

3) Hashing: Hashing isn't exactly a method of encryption but uses encryption ideas to create secure fields. To hash a

4) Hybrid Cryptosystems: These kinds of cryptosystems give us the best of both asymmetric and symmetric encryption methods. TODO talk about hybrid systems

B. Dependant Algorithms

There are many algorithms that different forms of cryptography rely on. This could be like an appendix type area maybe? I want to try and keep the algorithm talk less cluttered by talking about the parts they depend on first. then in the algorithm section we can focus on that we already explained what the underlying parts are here is the algorithm.

1) AKS Primality Test: Do we need to go into detail of algorithms that our algorithm is Dependant on? Maybe we make this section to help fill space up and make us look smart?

2) Euler's Totient Function: used in the rsa algorithm for something

3) Modular Arithmetic: what all of rsa is based on more or less define gcd and modular exponentiation

4) Padding Schemes: I bet a lot of these will need padding schemes and we should probably have a talk about this

II. RSA Asymmetric Cryptography

The RSA asymmetric cryptography algorithm is named after its creators Ron Rivest, Adi Shamir and Leonard Adleman and was first thought of in 1973. At the time it was highly classified and wasn't declassified until 1997. The idea behind this algorithm is to use large prime numbers to create a public key and a private key.

A. Private and Public Key Generation

The key generation process is a five step process.

1) Creating Large Prime Numbers: First we calculate two large prime numbers using the AKS Test. These will be denoted p and q. We want both of these values to be relatively the same length digit wise but it can differ a bit.

2) Compute n: Computing n is just the product of p and q. The value of n will be used as the modulus for our keys and it's length is known as the key length.

$$n = p * q$$

3) Compute Euler's Totient Function: Now we must use Euler's totient function to give us a max value for our public key e. Luckily we are able to use algebra to make computing this value much easier.

$$\phi(n) = \phi(p)\,\phi(q) = (p-1)(q-1) = n - (p+q-1)$$

4) Create Public Key: Now we need to create a value for e which will be our public key exponent for encrypting a message. For this value we want to use a reasonably small value without it being too small.

$$1 < e < \phi(n)$$

5) Create Private Key: Lastly since we have generated a value for e we will create d which will be used for decryption. To generate d we use modular arithmetic.

$$d \equiv e^{-1} \pmod{(n)}$$

B. Example of Usage

With the values of e and d determined here is how a string can be encrypted and decrypted. To send a message we need to convert it's text into a number. This is done by using an agreed upon padding scheme. To compute a ciphertext c we use the following equation.

$$c \equiv m^e \, (mod \, (n))$$

C. Security Concerns

III. ElGamal Asymmetric Cryptography

A. Private and Public Key Generation

B. Example of Usage

C. Security Concerns

IV. Elliptic Curve Asymmetric Cryptography

A. Private and Public Key Generation

B. Example of Usage

C. Security Concerns

V. Comparison

Here would be a good spot to compare and contrast all of them.

VI. Conclusion

The conclusion goes here.

Acknowledgment

I don't plan on having acknowledgments but since it is in the template i will leave it here for now

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.