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: Do we want to talk about hashing? Hashing isn't exactly a method of encryption but uses encryption ideas to create secure fields. could be used for more filler maybe?

4) Hybrid Cryptosystems: Hybrid cryptosystems use both asymmetric and symmetric encryption schemes. A hybrid system works by encrypting the message with a symmetric encryption key. This key is then encrypted using an asymmetric encryption technique. The symmetric key is then appended to the message. The benefit of using two encryption methods is that less time is needed for encrypting and decrypting the message due to using a symmetric technique plus since the symmetric key is encrypted using an asymmetric method we can still get the security of a public private key pair. A new symmetric key is generated each time to improve security further.

B. Dependant (Dependent?) 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) Cyclic Groups: integer and modular addition –¿ related to eulers totient function what all of rsa is based on more or less define gcd and modular exponentiation
cyclic groups are also used by elgamal i have discovered

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

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

5) Extended Euclidean Algorithm: This is used in rsa for creating a decryption key

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 was not declassified until 1997. The idea behind this algorithm is to use large prime numbers to create a public key and a private key. Security for this algorithm rises from the difficulty of factoring large integers.

A. Private and Public Key Generation

The key generation process is a five step process.

1) Creating Large Prime Numbers: First two large prime numbers are calculated using a prime number testing algorithm such as AKS. 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 compute the modular multiplicative inverse of e (modulo $\phi(n)$). This value is commonly computed using the Extended Euclidean Algorithm. This is represented by the equation below.

$$d \equiv e^{-1}\,(\mathrm{mod}\,(\phi(n)))$$

B. Encrypting and Decrypting a Message

With the values of e and d determined the next step is to use this key pair to encrypt and decrypt a cipher text. To start a message we need to convert it's text into a number. This is done by the use of ASCII values and by using an agreed upon padding scheme. To compute a ciphertext c and restore it to a readable message we use the following equations respectively.

$$c \equiv m^{e}\,(mod\,(n)) \text{ and } m \equiv c^{d}\,(mod\,(n))$$

C. Example of Usage

1) Creating Large Prime Numbers: To start with an example we will need to prime numbers for p and q. This example will be a simple one so smaller prime numbers are used. Let p = 991 and q = 821.

2) Compute N: We then use p and q to determine our value for n. Since $n = p * q$ this makes the value of n = 813611.

3) Compute Euler's Totient: Compute the Euler's totient value next.

$$\phi(813611) = \phi(991)\,\phi(821) = (990)(820) = \\ 813611 - (1811) = 811800$$

4) Create Public Key: With these values created a public and private key can be created. Let e = 7423 which was arbitrarily chosen and now d is computed.

5) Create Private Key: Using the extended Euclidean algorithm to compute the modular multiplicative inverse we get d = 788287. The values of d and e are our private and public keys respectively and can now be used for message encryption/decryption.

Let our plain text message m = "Hi". First we use an ASCII table to determine the value of m as a large integer. A simple way to do this is to concatenate each ASCII value together to form the integer. So m = 72105. Now that we have all the values we need we can encrypt and decrypt a message

$$\begin{aligned} c &\equiv 72105^{7423} mod(813611) \\ c &= 707473 \\ d &\equiv 707473^{788287} mod(813611) \\ m &= 72105 \end{aligned}$$

The ASCII values for our message 'Hi' were 72 and 105 so that means it was successfully decrypted.

D. Security Concerns

E. Discussion

III. ELGAMAL ASYMMETRIC CRYPTOGRAPHY

ElGamal public key cryptography is an alternative to RSA. This encryption algorithm was first described by Taher Elgmal in 1985. The ElGamal algorithm is dependent on the difficulty of computing discrete logs in a large prime modulus.

A. Private and Public Key Generation

The first step is to create a large prime number p. This p will be the exclusive max in our cyclic group. After p is generated an arbitrary number g that is between 1 and p - 1 is created. The variable g is the generator for our cyclic group. Then one last number x is selected where x is between 1 and p - 1, x will be used as the private key. We now compute a y value which will be the last part of our public key which is (p,g,y). To compute y see the next figure.

$$y = g^{x} \mathrm{mod} p$$

B. Encrypting and Decrypting a Message

When using ElGamal for encryption the cipher text will be twice the length of the plain text message. To encrypt first a random value k is chosen where k is between 1 and q - 1. Each time a message is sent the value of k should be changed. The message is then broken up into chunks and a padding scheme is followed so that the string is less than our value of p. Now for each chunk of the message m to be encrypted we create an ordered pair (a,b) by using the following equations.

$$\begin{aligned} a &= g^{k} mod\,(p) \\ b &= y^{k}\,(m)\,mod\,(p) \end{aligned}$$

A series of these order pairs are sent then to the receiver who can then decrypt them and assemble the message. To decrypt each ordered pair we can use the following equations.

$$s = a^x$$
$$m = b * s^{-1} mod\,(p)$$

C. *Example of Usage*

D. *Security Concerns*

E. *Discussion*

IV. Elliptic Curve Asymmetric Cryptography

A. *Private and Public Key Generation*

B. *Encrypting and Decrypting a Message*

C. *Example of Usage*

D. *Security Concerns*

E. *Discussion*

V. Comparison

Here would be a good spot to compare and contrast all of them. this is the area that i think he wants us to focus on but I don't know

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

VII. References

A. *General References*

http://mathworld.wolfram.com/CyclicGroup.html
http://www.cs.princeton.edu/ dsri/modular-inversion-answer.php
https://eprint.iacr.org/2012/195.pdf

B. *RSA References*

http://doctrina.org/Why-RSA-Works-Three-Fundamental-Questions-Answered.html

C. *ElGamal References*

https://asecuritysite.com/encryption/elgamal
http://homepages.math.uic.edu/ leon/mcs425-s08/handouts/el-gamal.pdf
http://www.nku.edu/ christensen/1002mat584ElGamal

D. *Elliptic Curve References*