An Introductory Look into Cryptography

By
Authors

George F
Vladimir S
Wesley W
Jason L

Introduction: What is Cryptography?

Cryptography is a field of math that is used to maintain a secure transfer of information from one party to another even in the presence of a third party trying to get it.

Historically, cryptography has been used to plan and win wars. For example, in World War II, the Enigma machine was used by the Axis to send battle plans from one commander to the other and until the Allies were able to crack the code, the Axis was able to launch devastating attacks without the Allies ever knowing about it. But when the Allies did learn the code, the tide of war turned.. Although the Enigma machine was very complex, an encryption using cryptography does not need to be. For example, Julius Caesar used a substitution cipher to secretly send messages to his generals during the Gallic Wars and it was one of the reasons why the Romans were so easily able to defeat the army of Gaul.

Cryptography has been used for thousands of years, but that does not mean that there are no modern applications of this field. In fact, most people use cryptography every day. When you log into your email account or your Facebook account or do anything that requires any amount of security, you are using cryptography. Indeed, most websites that are popular to use such as Google and Yahoo use the "https://" internet protocol. This means that those websites are encrypted with to some level in order to make sure that no one but you and the server you are sending the data to see anything that is even moderately intelligible. These are just a few of the many ways that cryptography is used in our lives.

In Section 1, we will begin to describe some of the basics of cryptography, including some definitions and examples of simple ciphers such as the substitution cipher and transposition cipher. Ultimately, it will establish the paper's main goal: to create an encryption method in which the sender encodes the message that only the receiver can decipher. This is secure encryption. This section will also introduce the idea of the public key cryptosystem, the kind of cryptosystem which we choose to study most in depth. In Section 2, we will begin to introduce the mathematics behind cryptography by discussing modular arithmetic and some of its operations, which is an important branch of mathematics in the field of cryptography. Discussion of modular arithmetic will then be applied in Section 3, which discusses some of many actual methods of encryption, including the Caesar cipher, RSA Cryptosystem, and Diffie-Hellman-Merkle key exchange, and reasons behind their securities. Though these are very different methods, they all establish the same goal of sending a secure message. Each strives to make that message as secure as possible.

After reading this paper, you will develop a further understanding into how we can use mathematics to create data that can be securely sent from one party to another, which is the essence of cryptography.

Section 1: The Basics

Before we can even start to analyze how we can create the secure systems used for encryption, we need to first learn how each part of that secure system is defined. Before we can send a message, we must first create one in everyday normal language. For example, we could write down the phrase "The quick brown fox jumps over the lazy dog" and this would be called our *cleartext.*

Definition 1.1- A message that has yet to be encoded and can be interpreted by anyone who has access to it is called *cleartext* or *plaintext.*

Then, after deciding upon the cleartext, the sender encrypts it in some way, i.e. changes the text in a way so that hopefully only the receiver can interpret it.

Definition 1.2- A message that has been encoded, or encrypted, in some way that is meant only for the receiver to interpret is called *ciphertext.*

Example 1.1- "The quick brown fox jumps over the lazy dog" is an example of cleartext that we may want to transform into ciphertext. Following an encoding where we encode each letter to the next one in the alphabet we get, we may arrive at a ciphertext that looks like the one below:
"Uif rvjdl cspxo gpy kvnqt pwfs uif mbaz eph"

To an onlooker with no understanding of what has been done, it may seem that gibberish was written. As we can see, had we been unaware of the what cleartext was, we likely would not have been able to make the connection that between the cleartext and the ciphertext had we tried to intercept and decode the message. Now, in order for us to get from cleartext to ciphertext, we would need to encrypt it. In essence, encryption is any method that changes the original text into a new message.

Definition 1.3- *Encryption* is any systematic method for transforming cleartext to ciphertext.

Example 1.2- Let us take the previous cleartext, "The quick brown fox jumps over the lazy dog" and encrypt it by shifting each letter to its next letter in the alphabet. So, 'a' now becomes 'b', the 'b' now becomes 'c', and so on until we get to 'z', which then becomes 'a'. Additionally, all spaces in between words will be removed. Below shows the cleartext and the ciphertext of this example.

Cleartext: "The quick brown fox jumps over the lazy dog"
Ciphertext: "Uifrvjdlcspxogpykvnqtpwfsuifmbazeph"

Now assume that we have the ciphertext above, and we wished to discover the meaning of the message. To do this, we would need to decode this message by a

method called *decryption*.

Definition 1.4- The method of transforming ciphertext back to the original cleartext is called *decryption.*

Example 1.3- We can decrypt “Uifrvjdlcspxogpykvnqtpwfsuifmbazeph” by reversing the steps we took in Example 3 and we will get *cleartext* “The quick brown fox jumps over the lazy dog.”

This prompts us to introduce some basic ciphers that directly use the idea of encryption and decryption.

Definition 1.5- In a *substitution cipher,* letters or blocks of letters are replaced by other letters or groups of letters.

Example 1.4- Suppose we translated the normal alphabet to the cipher alphabet below:

normal alphabet: a b c d e f g h i j k l m n o p q r s t u v w x y z
cipher alphabet: d e a c b z y x w v u t s r q p o n m l k j i h g f

> Then, the cleartext “hello” would be encrypted to form the ciphertext “xbttq”. To decrypt the message, the receiver would have to know the cipher alphabet, or an adversary (an enemy who attempts to intercept a message) would have to try to figure it out by trial and error.

Another very basic cipher is known as the *transposition cipher.*

Definition 1.6- In a *transposition cipher,* the letters themselves remained unchanged but are rearranged in a predetermined scheme.

Example 1.5- One example of a transposition is to simply reverse the order of the letters. Thus, the cleartext “Greetings my name is Julio” translates to ciphertext “oiluJ si eman ym sgniteerG”. Of course, transposition ciphers may be more complex, and we invite the readers to imagine more complex forms for themselves.

These two examples should provide a basic understanding of what encryption attempts to accomplish. Naturally, then, a question should now arise: how can we make sure that only those who we want to see the message will be able to decrypt it? Here is where the concept of *public and private key encryption* becomes vital, and more definitions are in store.

Definition 1.7- A *key* is a string of data that is used by an encryption or decryption algorithm to encrypt or decrypt data, respectively. This is comparable to a key on a map, which tells you how to read points on the map that you would not be able to read otherwise.

In public key cryptography, a form of cryptography invented in 1976, a key can either be labeled a *public key* and a *private key*. A public key is a key visible to anyone who wants to use it to encrypt information. A private key is known only to certain people, the decrypters, and this is the only key that can be used to decrypt the ciphertext.

Definition 1.8- A *public key* is a key that can be used by anyone to encrypt cleartext.

Definition 1.9- A *private key* is a key known by a select few to decrypt ciphertext that was created by a certain public key.

With public key encryption, secrecy becomes especially important, and it is therefore important to make sure that the private key is as difficult to compute possible. Below we define a *public key cryptosystem,* i.e. a cryptosystem that uses a public key and a private key, in which the private key is the inverse of the public key.

Definition 1.10- A *public key cryptosystem* is a cryptosystem with cleartext M between encrypter A and decrypter B that follows these steps:

1. A uses B's public key P_B to create the ciphertext $C = P_B(M)$.
2. A sends C to B.
3. B applies his secret key S_B to C to obtain M, so that $S_B(C) = M$.

We see, then, that $S_B(P_B(M)) = M$, indicating that S_B and P_B are inverses of each other. An adversary will likely know B's public key and may well intercept the ciphertext C, but the adversary will not obtain the message M unless he knows B's secret key. His way to find the secret key, then, is to find the inverse of the public key. Therefore, the challenge for A and B in a public key cryptosystem is to design a public key that is as difficult to invert as possible, though feasible to compute if given a method of inversion.

Therefore, public key cryptosystems are based on this idea of creating functions that are easy to compute on inputs but hard to invert. These functions are referred to as *one-way functions,* with a *trap-door function* being the secret inverse to a one-way function. We do not provide technical definitions of these as they are complex and stray from our purpose, and are mentioned only for later reference. We will see that the challenge of creating these functions applies in our later discussion of the *RSA Cryptosystem,* which is one of the most famous public key cryptosystem.

In order for us to be able to prevent adversaries from being able to read our message, we need to make sure that our method of encoding and decoding, or in other words our cryptosystem, as secure as possible. Though different mathematical models have defined what a secure cryptosystem is, the actual definition of security is still somewhat a matter of opinion. Ultimately, though, the goals of security are:

1. To create extreme difficulty in being able to determine the cleartext from an intercepted ciphertext
2. To create a system of encryption that is difficult to detect, even with

multiple samples of ciphertext.

The branch of mathematics called *modular arithmetic* is especially important in mathematically encrypting messages, so we take a digression in Section 2 to establish a foundation for modular arithmetic and some of its operations necessary for the RSA and Diffie-Hellman Cryptosystems. On top of this, we will describe other mathematical operations that are key (no pun intended) to constructing the cryptosystems.

Section 2: Modular Arithmetic and Other Related Mathematics

Now that we have the essential definitions of cryptography under our belt, we will begin to discuss some of the encryption methods used. One of the most classic codes used in encryption is known as the *Caesar Cipher,* in which the letters of the alphabet are all shifted by a predetermined amount in the encryption. For example, in a Caesar Cipher, the encoder could shift all letters over 5 letters further down the alphabet, so that 'A' becomes 'F', 'B' becomes 'G', etc. It is relatively simple to describe such encryption with *modular arithmetic.* Modular arithmetic is useful in this example of encryption but also has great importance in other methods of cryptography, and so we take a digression into modular arithmetic in itself before applying it to methods of cryptography.

Definition 2.1- The congruence relation $x \equiv m \bmod n$ means that $x - m = nk$, for some integer k, and is read "x is congruent to m modulo n." Since k can be any integer, x can be an infinite number of values for constant integers m and n. This statement x and m each have the same remainder when divided by n.

Example 2.1- $1 \equiv 13 \equiv 25 \bmod 12$, because 1, 13, and 25 all have the same remainder when divided by 12.

Notice we used the congruence sign, "≡", instead of the equal sign, "=", when evaluating 25 mod 12. This is to distinguish between modular arithmetic and basic arithmetic, so that we read "25 is congruent to 1 modulo 12." For the purposes of cryptography, it is most convenient to deal with the class of numbers $\{0, 1, ..., m\text{-}1\}$ when using modular operations, and so we write the following definition below.

Definition 2.2- The equation $r = m \bmod n$ gives the remainder r when dividing the integer m by the integer n. Specifically, m mod n is the smallest nonnegative integer r such that

$$m = nq + r,$$

for an integer q such that $nq \leq m$ and $n(q+1) > m$.

Example 2.2- 25 mod 12 = 1.

Another helpful way to describe modular arithmetic is "wrap-around arithmetic." One common example of this mode of thought is in adding time, which uses mod 12 arithmetic. For example, if it is 11:00 and we want to know the time in 3 hours, it will be (11+3) mod 12 = 2. Therefore, the time becomes 2:00, and we reached this value by "wrapping around" 12 o'clock back to 1:00, rather than going on to 13:00.

In the Caesar cipher specifically (not a public key cryptosystem, but interesting to note nonetheless), a computer could implement mod 26 arithmetic. If each letter in the alphabet were labeled 0 to 25, and we used a shift of two letters in our encryption, the computer would use the code $x = (n + 2) \bmod 26$, where x represents the ciphered letter, and n is the original letter. That way, for letters Y and Z (labeled 24 and 25), the encryption would return 0 and 1 for x, which translates to the letters A and B; the shift would "wrap around" the alphabet, and return back to the beginning. In general terms, a shift of k letters in a Caesar cipher could be described mathematically by the function $x = (n + k) \bmod 26$.

Of course, this kind of encoding would be incredibly easy for an adversary to crack. All that he would have to do to obtain the cleartext is simply test each of the possible 26 shifts that are possible by translating each shift into English. Surely, one of these tests would reveal the desired message, and therefore this method of encryption is a relatively unsafe one. Still, we will see that modular arithmetic is a necessity for many important methods of cryptography.

In our earlier example of the Caesar cipher, the inversion was plainly obvious: to simply shift the alphabet backwards k letters if the original action was to shift it forward k letters. Clearly, a more complex combination of operations in modular arithmetic is needed to create a secure encryption, and so we now discuss operations in modular arithmetic.

Arithmetic mod *n*

We will now begin to introduce some of the basic operations of arithmetic in mod n. We have already seen that addition is relatively easy operation to invert. Here, we will explore multiplication, exponentiation, and the multiplicative inverse, as these specific operations become very important in our later discussion of cryptosystems.

Before discussing all of these operations, we must establish the following lemma, which proves an idea stated earlier.

Lemma 2.1: $a \bmod n = (a + kn) \bmod n$, for any positive integers *a, n, and k.*

Proof: By Definition 2.2, we have

$$a = qn + r, \text{ where } r = a \bmod n, \text{ for some } q.$$

Adding kn to both sides gives

$$a + kn = (q+k)n + r.$$

By Definition 2.2, we have $r = (a + kn) \bmod n$, and thus the lemma is proved.

Multiplication

Particularly, we want to prove the following lemma, as this result will be used in exponentiation.

Lemma 2.2: $(a \cdot b) \bmod n = [a \cdot (b \bmod n)] \bmod n$

Proof: We can write

$a = (a \bmod n) + cn$ and $b = (b \bmod n) + dn$, for some integers c and d.

Then,

$$ab = ((a \bmod n) + cn)((b \bmod n) + dn)$$

$$ab = (a \bmod n)(b \bmod n) + dn(a \bmod n) + cn(b \bmod n) + cdn^2.$$

After taking the "mod" of each side, we can simplify this to

$$(a \cdot b) \bmod n = [(a \bmod n)(b \bmod n)] \bmod n, \text{ by Lemma 2.1.}$$

Using $a \bmod n = a - cn$, we have

$$(a \cdot b) \bmod n = [(a - cn)(b \bmod n)] \bmod n$$

$$(a \cdot b) \bmod n = (a(b \bmod n) - c(b \bmod n) \cdot n) \bmod n$$

$$(a \cdot b) \bmod n = [a \cdot (b \bmod n)] \bmod n, \text{ again by Lemma 2.1.}$$

Thus, the lemma is proved. This identity of multiplication becomes very important in our now imminent discussion of exponentiation, which is likewise very important in the RSA Cryptosystem.

Exponentiation

Suppose we wanted to evaluate the following modular exponent:

$$a = 16^{23} \bmod 782.$$

Based on our current knowledge of modular arithmetic, we could calculate 16^{23} directly, and then find its remainder when divided by 782. Though a modern, high-powered computer could make these calculations, larger values would not be within the computer's memory, or at the very least make calculations take an unreasonable amount of time to perform. Therefore, we need a systematic way to deal with modular exponentiation.

To find the value of $r = m^k \bmod n$, we can use the following cycle of steps.

1. Begin with $r = 1$, and $k' = 0$, where k' is an arbitrary integer (obviously related to k.)
2. Increase k' by 1.
3. Reset $r = (r \cdot m) \bmod n$.
4. Stop if $k' = k$, at which point $r = m^k \bmod n$. Otherwise, go back through the cycle starting at step 2.

Proof: We claim that at each pass through step 3, $r = m^{k'} \bmod n$. This also proves that the end of the method will supply us with our desired value.

At our base case of $r = 1$ and $k' = 1$, step 3 gives us a new $r = (1 \cdot m) \bmod n = m^1 \bmod n$. Thus, the base case proves true.

Now, assume true for k'. Then, our next path through step three will give new value r,

$$r = ((m^{k'} \bmod n) \cdot m) \bmod n.$$
$$= m^{k'+1} \bmod n, \text{ by Lemma 2.2}$$

Therefore, by induction, we know that once $k' = k$, we will end up with the desired result for r, that being $m^k \bmod n$.

This method of calculating modular exponents is far more memory efficient than the direct method, as it will almost never involve calculations of numbers of too long a bit length for the computer to calculate. In fact, The largest numbers that could be produced are in the product rm, yet we know that $r < n$ and that m is just the base of the exponent. Thus, as long as m and n are of reasonable bit length (in any typical case, they are), this method of performing exponentiation is possible for a computer to compute (or even a person who wanted to have some fun!).

Now, we discuss the *modular multiplicative inverse,* an intermediate operation of the RSA Cryptosystem.

Definition 2.3: The *modular multiplicative inverse* of an integer m modulo n is an integer x such that

$$mx \equiv 1 \bmod n.$$

For the modular multiplicative inverse to exist, we must have $\gcd(m, n) = 1$.

Example 2.3- The modular multiplicative inverse of 7 modulo 4 could be any numbers of the class {...,-1, 3, 7, 11, …}. For example, if we choose $x = 3$, then the statement $7x \equiv 1 \bmod 4$ is true. In general, the modular multiplicative inverse will refer to the smallest positive integer x that satisfies this congruence.

By our definition of modular congruence, we know, using the variables from the the definition above,

$$mx - 1 = nq, \text{ for some integer } q.$$

Equivalently,

$$mx - nq = 1.$$

The equation above can be solved for x and q by a method known as the *extended Euclidean Algorithm,* which is an algorithm, given the equation

$$ax + by = \gcd(a, b),$$

and given integers a and b, solves for integers x, y, and $\gcd(a, b)$.

Our definition should inspire one question then: why must gcd(m, n) = 1? If we go back to our equation

$$mx - nq = 1,$$

note that gcd(m, n) must divide each term on the left side, and thus it must divide 1 as well. Therefore, it must be true that gcd(m, n) = 1, and therefore the modular multiplicative inverse only exists if this equality is confirmed. Then, with gcd(m, n) = 1, the extended Euclidean Algorithm can be applied directly to solve for x, the inverse.

This relates closely to *Euler's Totient function*, which is also vital to the RSA Cryptosystem, and is defined below.

Definition 2.4- *Euler's Totient function,* $\varphi(n)$, gives the number of positive integers k less than or equal to a positive integer n such that gcd(n, k) = 1.

This leads directly into the following lemma:

Lemma 2.3- If an integer n is a positive, prime integer, then $\varphi(n) = n - 1$.

Proof: If n is a prime number, then its only divisors are 1 and n, and thus every positive integer k less than n satisfies gcd(n, k) = 1, and there are (n - 1) such integers.

It follows from this lemma that, if p and q are distinct prime integers, then $\varphi(n) = (p-1)(q-1)$, where $n = pq$.

The fact above is used both in the process of encryption in the RSA Cryptosystem and in the proof of its correctness, which is performed using *Euler's Theorem.* This theorem is written below.

Euler's Theorem - If an integer a and n are coprime positive integers, then

$$a^k \equiv a \mod n,$$

where $k = \varphi(n)$.

We leave out a proof of this theorem, as it is very complicated and digresses from the purpose of establishing tools for constructing cryptosystems, which we begin to do now. In the following section, we will use the tools established in this section to construct relatively secure cryptosystems like the RSA Cryptosystem and the Diffie-Helman Exchange.

Section 3: Functions and Public Key Cryptosystems

In this section, we will begin to discuss some more complicated cryptosystems, particularly the RSA Cryptosystem and the Diffie-Hellman-Merkle Exchange.

Now, we will proceed to describe the process of constructing an RSA cryptosystem, and prove its correctness afterwards.

RSA Cryptosystem

There are three stops involved in creating the cryptosystem, those being determining the keys, encrypting, and decrypting.

Determining a Public and Private Key:

1. Find the product n of two large, prime numbers p and q.
2. Find the totient $a = \varphi(n) = (p - 1)(q - 1)$.
3. Pick a number e that is relatively prime to a.
4. Find d, where $d^{-1} = e \bmod a$, i.e. where d is the multiplicative inverse of e modulo a. Note that this calculation can be carried out since e is relatively prime to a.
5. The public key will be (n,e) while the private key will be (n,d).

Encryption:

1. The sender will change his desired message into numbers, labeled m, through a pre-arranged method.
2. These numbers will be changed into ciphertext c through the equation $c = m^e \bmod n$, which can be calculated by our memory-efficient method of modular exponentiation discussed in Section 2.

Decryption:

1. The receiver will decrypt through the equation $m = c^d \bmod n$.

Our one-way function here is $c = m^e \bmod n$ and the trapdoor is $m = c^d \bmod n$.

Proof of Correctness: We are going to prove that $m = c^d \bmod n$, where $c = m^e \bmod n$, assuming the variables given in the steps above, which is equivalent to proving that $m \equiv (m^e)^d \bmod n$. Given the equality $a = \varphi(n)$, we know that, since we made d the multiplicative inverse of e modulo a, we have

$$ed \equiv 1 \bmod a,$$

and therefore,

$$ed - 1 = ka, \text{ for some integer } k.$$

Thus, we can substitute into $(m^e)^d \bmod n$ so that now we have

$$m^{ka + 1} \bmod n$$
$$= m(m^a)^k \bmod n.$$

Using our knowledge of modular multiplication, we know this to be equivalent to

$$[m \cdot ((m^a)^k \bmod n)] \bmod n.$$

Now, we wish to evaluate $(m^a)^k \bmod n$, and substitute it into the equation above. By Euler's Theorem, since $a = \varphi(n)$, we know that $m^a \equiv 1 \bmod n$, which therefore means that $(m^a)^k \equiv 1 \bmod n$, as well. Thus, we are left with the congruence

$$[m \cdot ((m^a)^k \bmod n)] \equiv m \bmod n.$$

This confirms that $m \equiv (m^e)^d \bmod n$, and therefore that the methods of encryption and decryption successfully transfer the message m between the sender and receiver.

Now that we have confirmed that the RSA Cryptosystem can successfully transfer a message, we will now provide an example of the cryptosystem in practice. Note that the example below chooses much lower numbers for *p, q, e,* and *d* than would be ordinary. The numbers below were used for the sake of simplicity and understanding.

Example 3.1- Suppose Jack wanted to send the message, "13," to Jill. Beforehand, Jack and Jill agreed that $p = 5$ and $q = 7$ would be the two primes they utilized.

Finding the keys:

1. The product $n = pq = 35$.
2. The totient is $\varphi(n) = (5-1)(7-1) = 24$. .
3. Jack's favorite number was 7, so this was decided to be e.
4. Here, we want to find d where $7d = 1 \bmod 24$. In general, d can be easily solved by a computer with the previously mentioned algorithm *Euclidean's Algorithm.* For this case, the easiest method is trial-and-error by using the formula $7d - 1 = 24k$, where k is the smallest positive integer satisfying this equation.

So, $d = \frac{24k+1}{7} = \frac{24}{7}k + \frac{1}{7}$ and we're looking for the first number where d is an integer. This gives us $k = 2$ and $d = 7$.

Encrypting:

The message is already in number form, so encrypting it is the only thing left for Jack to do:

$c = 13^7 \bmod 35 = 27$

Decrypting:

Jill has received the message safely and deciphers it:

$m = 27^7 \bmod 35 = 13$, getting the desired message $m = 13$.

This is an extremely secure cryptosystem because it requires the factoring of two extremely large prime numbers (prime numbers with more than 100,000 digits), and with the current state of technology and mathematics, this is quite difficult and takes much more time than what is computationally reasonable for someone who wants to access the information.

Diffie-Hellman-Merkle Key Exchange

The Diffie-Hellman-Merkle key exchange, also known as the exponential key exchange is the particular way of sharing cryptographic keys. This method allows two parties to create a default code through a channel without the need of understanding the each others' codes through the following method.

Assume two parties, A and B are sending each other a message over an open channel. A and B have decided to use a common prime p, and a r, where r is a primitive root mod p. Additionally A has chosen secret integer a and B has chosen secret integer b. Then we have that:

A sends to B the following message: $C = r^a \bmod p$
B sends to A the following message: $D = r^b \bmod p$

Now, in order to have a common secret they will do a similar operation, but with C and D. Thus:

A calculates: $E = C^a \bmod p$
B calculates: $D^b \bmod p$, which nicely also equals E

The reason both A and B reach the common secret E is because they both calculate $E = r^{ab} \bmod p$, and thus through the previous key exchange, both parties A and B now share a common secret E. This is remarkable, yet simple. With A and B keeping a and b secret, respectively, they are each sent enough information to calculate the secret $E = r^{ab} \bmod p$, yet an adversary, who only has knowledge of r, p, $C = r^a \bmod p$, and $D = r^b \bmod p$, is unable to be able to find E.

Summary

In this brief introduction to cryptography, the background of cryptography, modular arithmetic, and methods to exchange and encrypt messages were discussed.

Cryptography has been used for centuries was a vital part of many wars and feuds. The Nazi Enigma machine was one such example. The main parts of cryptography consist of establishing a known encryption method, encoding the message, sending it, and deciphering the message. The encryption should be secure, meaning the ciphertext is hard to decipher. These means many of the most basic encryptions are insecure, such as the Caesar cipher. One-way functions are examples of secure encryptions.

One-way functions are easy to compute but hard to invert. Examples include factoring a product of two primes. A special type of one-way function is the trapdoor function, where the receiver has a trapdoor which can decipher encrypted messages. The RSA cryptosystem is a trapdoor function. It involves using special encryption and decryption "keys" to modify the problem of factoring the product of prime numbers.

The Diffie-Helman-Merkle key exchange is a method of exchanging messages and sharing encryption keys. It involves sending random secrets between the two encoders involved. Then, the two will determine the key through a set way.

Bibliography

For discussion of operations in modular arithmetic.
http://www.math.dartmouth.edu/archive/m19w03/public_html/Section2-1

Euler's Theorem, Totient Function, RSA Cryptosystem.
http://www.unc.edu/math/Faculty/petersen/Coding/cr2.pdf

Modular Multiplicative Inverse
http://www.math.cornell.edu/~morris/135/mod.pdf

Diffie-Hellman Exchange
http://math.ucsd.edu/~wgarner/research/pdf/diffie-hellman_key_exchange.pdf

Basics of Cryptography
http://fisher.osu.edu/~muhanna.1/pdf/crypto.pdf