International Journal of Mathematics and Statistics Invention (IJMSI)
E-ISSN: 2321 – 4767 P-ISSN: 2321 - 4759
www.ijmsi.org Volume 10 Issue 6 || Nov.-Dec., 2022 || PP-07-13

The Role of Number Theory to Cryptography

Xiaomeng (Christine) Zhu
Hefei No. 8 High School International Department

***ABSTRACT**:Cryptography refers to the field of computer science that studies and develops techniques that make the exchange of secret messages secure. The goal of cryptography is to disguise messages, so they can not be understood by any unintended party that may capture the messages. The algorithms used in cryptography are based on Number Theory. In this article, we review the basics of this connection and how it leads to the RSA algorithm.*

Date of Submission: 10-12-2022 Date of Acceptance: 25-12-2022

I. INTRODUCTION

Number theory is a classical area of mathematics [1]. Number theory is cryptography's [2] fundamental building block. Number theory motivated the development of our robust encryption techniques and systems. The encode and decode constitute the critical scheme of number theory. Encode and decode are a pair. Encode means to transfer a piece of the sequence of characters in a particular way from one communication system to another. The meaning of decode is to convert the received sequence of characters into a form that is comprehensible to the recipient through an information system. The concept of encryption and decryption is produced by encoding-decoding but with more meaningful and valuable real-life applications. Encrypt means translating data or information that needs to be passed into code to prevent unauthorized access. Decryption means to make the data or information received clear and understandable.

Consider the following scenario. When you are a student, maybe you want to talk to your friends in class, but you don't want to make a noise. Then you may have done things like John and Peter. John and Peter are best friends and they are in the same classroom in school. Alice sits on a desk between them. John wants to invite Peter to go to his house after school. They are not allowed to talk during class, but John is impatient, and does not want to wait for recess. He writes a note and tells Alice to give the note to Peter. Even though she is not supposed to, Alice reads the note. The note reads **Dpnf ipnf bgus tdippm.** Alice can not make sense of this message and gives it to Peter. But Peter knows that John is trying to hide the content of his message. John and Peter have agreed that they should change each letter in the message by the letter that comes before in the alphabet when they send each other written messages. Thus, Peter changes **D** by **C, p** by **o, n** by **m, f** by **e,** etc. After going through this exchange of letters, the original message sent by John, which was **Dpnf ipnf bgus tdippm,** now reads **Come home after school.** Although Alice helps

John and Peter pass the note, she does not know the original message because she does not know how they encrypt the sequence of irregular characters. This message transform process is a simple example of a cryptographic algorithm.

But the above example is too simple, and it is very easy to break. We want ways to encode or encrypt that are difficult to break. These algorithms are based on the mathematical field known as number theory. Number theory is the study of properties of integers. Factoring very large numbers is very hard. By very hard we mean that it would take the most powerful computers a very large number of years to factorizing very large integers, even if they use the most advanced known algorithms. The basic cryptographic algorithm works because of this fact.

To unlock the intercepted messages, a decoder is necessary. It was not until World War Ⅱ that cryptography became widely used. Enigma is a famous machine that German Nazis heavily used in WW II [3]. The machine used three or more rotors which rotated at different rates as the user typed on the keyboard and outputs the matched letter of cipher text with random and even unbreakable patterns. The unbreakable Enigma machine was eventually broken by Turin, who designed the famous Turing machine to become one of our original computer models. By deciphering the enemy's code, the enemy's real strategy was known in advance, thus accelerating the process of the Second World War. Now modern cryptography is used everywhere, such as credit card transactions on the internet, computer users access passwords, e-commerce, etc. RSA [4] is a very popular cryptographic algorithm developed in the 1980s by MIT researchers. In this article we will explain the RSA algorithm and number theory behind it.

RSA has been widely used in access control, personnel permission information and other fields. One of the most recent applications that has been widely used in bank systems is digital signature which is based on

RSA. Take the bank as an example, first determine the role: assume Thea is the person who goes to the bank to withdraw money. She has a bank card in her hand, and only she knows the password of this bank card, so she acts as the encryption agent. Flora is an executive in the bank, she is a decryptor. Flora first randomly determines a KEY, which we call a secret key, but this key is always stored in the machine without being published. Then, from this key, another key is computed, which we call the public key. The property of this public key is that it is almost impossible to compute its own private key. Next, the public key is transmitted to Iris through the network. When Thea receives the public key, the public key is used to encrypt the password set, and the encrypted password is sent to Flora through the network. Finally, Flora uses the known private key to decode the information sent over, so as to know the original password set by Thea. To determine whether Thea can withdraw money from the bank. So that's how RSA works in a bank.

In this article we explain the Number Theory behind the RSA algorithm.

II. ELEMENTS OF NUMBER THEORY

As explained in the introduction, the cryptographic algorithm we will describe in this article, RSA, is built from Number Theory. In this section, we will explain the elements of number theory that are necessary to understand how the RSA algorithm works.

Integer Division

We start with the following definition, that reminds the reader what it means for an integer to divide another integer.

Definition: Let a and b be two integers. We say that b divides a if $a = bc$ for some integer c. In this case, we also say that b is a divisor of a, and a is a multiple of b. The notation that indicates that b divides a is $b|a$. To indicate that b does not divide a we write $b \nmid a$.

For example, 6 divides 18 because18 = 6(3). The next theorem states what most of us implicitly know about quotients and remainders when we carry out integer divisions.

Theorem 1: Let a and b be two integers. Assume b> 0. Then, there exists two unique integers, q and r such that $0 \leqslant r < b$ and $a = qb + r$, denote as $a \equiv r \bmod b$.

The number q in the above theorem is known as the quotient of a divided by b and r as the remainder. We will denote q by $a//b$ and r by $a\%b$. These are the commands in the programming language Python.

For example, the quotient of 20 divided by 6 is 3 and the remainder is 2, because 20 = 6(3) + 2 and $0 \leqslant 2 < 6$. Using the notation introduced in the previous paragraph, 20//6 = 3 and 20%6 = 2. Note that b divides a if the remainder of a divided by b id a divided by 0.

Greatest Common Divisor

Given two integers a and b, the largest integer that divides both a and b is known as the greatest common divisor between a and b, and it is denoted by $gcd(a, b)$. For example, the greatest common divisor between 18 and 12 is 6. Using the notation we just introduced, $gcd(18, 12) = 6$.

We will describe an algorithm to compute the greatest common divisor between two given integers a and b. This algorithm is based on two facts. One of these facts is that $gcd(a, 0) = a$ for all positive numbers a. The second fact is the following Theorem.

Theorem 2: Let a and b be two integers with $b \neq 0$. Let r be the remainder of a divided by b. Then $gcd(b, r) = gcd(a, b)$. Using the notation introduced in the above paragraph, $gcd(b, a\%b) = gcd(a, b)$

For example, the remainder of 20 divided by 8 is 4, i.e. 4 = 20%8. Thus, the above theorem says that $gcd(8, 4) = gcd(20, 8)$. In fact, $gcd(8, 4) = gcd(20, 8) = 4$.

Euclidean Algorithm

Our next goal is to explain an algorithm to compute the greatest common divisor of two integers. This algorithm is known as the Euclidean algorithm and is based on Theorem 2.

This algorithm takes as input two integers a and b. If $b = 0$, the algorithm returns a as the greatest common divisor between a and b. As we previously discussed, we know this to be correct. On the other hand, if $b \neq 0$, the algorithm returns $gcd(b,a\%b)$ as the greatest common divisor the greatest common divisor between a and b. The correctness of the algorithm is implied by Theorem 2. It is also easy to check that the algorithm eventually terminates.

Note that the algorithm calls itself. More precisely, to compute $gcd(a, b)$ we need to compute $gcd(b, a\%b)$ if $b \neq 0$. Algorithms of this type are called recurrent algorithms. The pseudo code of this algorithm is

```
def greatest_commom_divisor($a$, $b$):
if $b$ = 0:
return a
return greatest_commom_divisor($b$, $a$%$b$)
```

As an example, if a = 20 and b = 8, i.e we want to compute gcd(20, 8), the algorithm proceeds as follows

1. Since 8 $\neq$ 0, the algorithm calls itself to compute gcd(8, 4), because 4 = 20%8
2. To compute gcd(8, 4), and since 4≠0, the algorithm calls itself to compute gcd(4, 0), because 0 = 8%4
3. The algorithm returns 4 as gcd(4, 0)
4. The algorithm returns 4 as gcd(8, 4)
5. The algorithm returns 4 as gcd(20, 8)

We will need more than simply computing the greatest common divisor of two numbers. More precisely, we will need to compute the number u that is described in the following theorem.

Theorem 3: Let a and b be two integers. There exists two integers u and v such that gcd(a, b) = ua + vb.

For example, if a = 20 and b = 8, we know that gcd(20, 8) = 4, and possible values u and v are u = 1 and v =− 2, because 4 = (1)(20) + (− 2)(8). Note that the pair u =− 1 and v = 3 also works, because 4 = (− 1)(20) + (3)(8).

Our next goal is to extend the Euclidean algorithm, that was described above, so that, instead of only computing gcd(a, b) when a and b are given as input, it also computes a pair u and v as in Theorem 3, i.e, such that gcd(a, b) = ua + vb

Note that the smallest positive number of the form au + bv, where u and v are integers is equal to gcd(a, b). For our applications to cryptography, we will need u. Below, we will describe an algorithm to find u and v. Before we describe this algorithm, consider the following numerical example.

Assume a=60 and b=22. Our goal is to find u and v such that 22u + 60v = gcd(22, 60). We first perform the Euclidean algorithm to compute the greatest common divisor. We find:

$$60 = 2 \times 22 + 16$$
$$22 = 1 \times 16 + 6$$
$$16 = 2 \times 6 + 4$$
$$6 = 1 \times 4 + 2$$
$$4 = 2 \times 2 + 0$$

This shows that gcd(22,60) = 2

In order to get the Euclidean format, we need to reverse the process of above steps. So we get:

$$2 = 6 - 4$$
$$2 = (22 - 16) - (16 - 2 \times 6)$$
$$2 = (22 - 16) - (60 - 2 \times 22 - 2 \times (22 - 16))$$
$$2 = 22 - 60 + 2 \times 22 - (60 - 2 \times 22 - 2 \times 22 + 2 \times 60 - 4 \times 22)$$
$$2 = 22 - 60 + 2 \times 22 - 60 + 2 \times 22 + 2 \times 22 - 2 \times 60 + 4 \times 22$$
$$2 = (-4)60 + 11(22)$$

In this case, u is 11 and v is -4 as indicated.

We now describe the algorithm that takes as input a and b and gives as output gcd(a, b), u and v, where u and v satisfy gcd(a, b) = ua + vb. We assume that b is a positive integer. The algorithm is recursive on b. We will denote by a//b the quotient of a divided by b. For example, 10//4 = 2. This algorithm results from the following two observations:

1. If b = 0, we have that gcd(a,b) = a, and the values$u$ = 1and$v$ = 0satisfy the requirement

$$gcd(a,b) = ua + vb. \quad (1)$$

This case, where b = 0, is called the base case.

2. Ifb>0.Let$q$=a//b,and$r$=a%b.Notethat0 ⩽ r<b.Let$u$'and$v$'be integers such that $gcd(b,r) = u'b + v'r$. Note that r = a − qb. Thus, theequation

$$gcd(b,r) = u'b + v'r(2)$$

becomes

$$gcd(b,r) = u'b + v'(a - qb) = v'a + (u' - qv') \quad (3)$$

Note also thatgcd(a,b) = gcd(b,r). Thus, u = v' and v = u' − qv' satisfy

$$gcd(a,b) = ua + vb \quad (4)$$

The above observations lead to the recursive algorithm below. The input are the two integers a and b, and the output is a triplet gcd(a, b), u, v, where as explained above, gcd(a, b) = ua + vb.

```
def extended_euclidean_algorithm($a$, $b$):
if $b$ == 0:
return $a$, 1, 0
else:
$d$, $u'$, $v'$ = extended_euclidean_algorithm($b$, $a\%b$)
$v$ = $u' - (a//b)v'$
$u$ = $v'$
return $d$, $u$, $v$
```

Let a and b be two positive integers. Let $d = gcd(a, b)$. Let u and v be two integers such that $d = ua + vb$. The above algorithm allows us to find such u and v. However, in our applications to cryptography, we will want u to be non-negative. Let $k$beaninteger. $u' = u + \frac{kb}{d}$ and $v' = v - \frac{ka}{d}$.Notethatboth$u'$and$v'$are integers. Note also that $u'a + v'b = ua + vb = d$. Thus,If$u<0$,wereplace$u$and v by $u + \frac{b}{d}$ and $v - \frac{a}{d}$,respectively,togetnewsolutionsof$d = ua + vb$,with$u$ larger than the original value found. We repeat this step till we obtain u and v integers such that $d = ua + vb$ with u positive. The resulting pseudocode is

```
def extended_euclidean($a$, $b$):
if $b$ == 0:
return $a$, 1, 0
else:
$d$, $u'$, $v'$ = extended_euclidean($b$, $a\%b$)
$v$ = $u' - (a//b)v'$ $u$ = $v'$
while $u$< 0:
$u$ = $u + b/d$
$v$ = $v - a/d$
return $d$, $u$, $v$
```

Modular Arithmetic

The most basic feature that the RSA algorithm used is modular arithmetic. It utilized the features of the field that a number (after some operations) divides a number and the remainder will create a cycle. In these regularities, the cycle is the same for every certain number of operations. Such transformation is called congruence.

Definition: We say that a is congruent to b modulo m(a and b are two integers, and m be another integer), and we write: $a = b\ (modm)$, if m divides $a - b$.

Example: $7 \equiv 2\ (mod\ 5)$
Observation: $a \equiv (a\ \%\ m)\ (modm)$
Example: $72 \equiv 2\ (mod\ 7)$

If a divided by m leaves a remainder of r, then a is congruent to r modulo m. Note that the remainder satisfies $0 \leqslant r < m$, so every integer is congruent, modulo m, to a number between 0 and $m - 1$. Also, it is not always possible to divide congruences. In other words, if $ac \equiv bc\ (modm)$. It need not be true that $a \equiv b(modm)$.

Example: $15 \times 2\ (mod\ 10) \equiv 20 \times 2\ (mod\ 10)$, but $15 \not\equiv 20\ (mod\ 10)$.

Note that, if $x \equiv y(modm)$ and $a \equiv b(mod\ m)$, we have that

$$a + x \equiv b + y(modm) \tag{5}$$

and

$$ax \equiv by\ (modm) \tag{6}$$

Example: $7 \equiv 2(mod\ 5)\ and\ 9 \equiv 4\ (mod\ 5)$. So we can get $63 \equiv 8(mod5)$

In addition to the congruence, the prime number has an essential feature called breakability, which is measured by calculating the Euler ϕ (phi) function. This function is defined in the next subsection. While the value of the ϕ function is greater, it indicates that the number has larger breakability and is more likely to be chosen as an encryption quotient.

Solving the Equation $ax \equiv c(mod m)$ for x

Assume that we are given integers a, c and m, with m positive. The goal of thissection is to find a non-negative integer x such that $ax \equiv c(modm)$ and $x<m$.

Observation: Let $d = gcd(a, m)$. Assume d divides c. Let u and v be integers such that $ua + vm = d$. Let $x = \left(\frac{uc}{d}\right)\%m$. Then, x is a non-negative integer, $ax \equiv c(modm)$ and $x<m$.

To understand the validity of the above observation, first note that, since d divides c, the number c/d is an integer. Thus, uc/d is also an integer. Next, note that $x = \left(\frac{uc}{d}\right)\%m \equiv \frac{uc}{d}(mod\ m)$, from where we have that $ax = a\left(\frac{uc}{d}\right)\%m \equiv \frac{auc}{d}(modm)$. Since $m \equiv 0(modm)$ and d divides c, we have that $\frac{mc}{d} \equiv 0(modm)$ and thus, we also have that $\frac{vmc}{d} \equiv 0(modm)$. This last equation and the previous equation $ax \equiv \frac{auc}{d}(modm)$ imply that $ax \equiv \frac{auc}{d} + \frac{vmc}{d}(modm)$. Thus, pulling $\frac{c}{d}$ as a common factor in the right hand side of the last equation, we get that $ax \equiv \frac{(au + vm)c}{d}(modm)$. But recall that $ua + vm = d$. Thus, we get that $ax \equiv c(modm)$, which was what we wanted to prove.

The last observation leads to the following pseudocode that takes as input three integers, a, c and m, such that m is positive, $gcd(a, m)$ divides c, and gives as output a non-negative integer x such that $ax \equiv c(modm)$ and $x<m$.

```
def ax_eq_c_mod_m($a$, $c$, $m$):
$d$, $u$, $v$ = extended_euclidean_algorithm($a$, $m$)
return $(uc/d)\%m$
```

Example: Let $a = 5$, $c = 4$ and $m = 17$. We have that $gcd(a, m) = gcd(5, 17) = 1$. Thus, the condition that $gcd(a, m)$ divides c is satisfied. We first apply what we called the extended Euclidean algorithm to find u and v such that $u5 + v17 = 1$. We find $u = 7$ and $v =- 2$. The above observation says that a non-negative solution x to $5x \equiv 4(mod\ 17)$ that is smaller than 17 should be $x = (7)(4)\%17 = 11$. In fact, $5(11) = 55 \equiv 4(mod\ 17)$.

Euler Phi Function

Definition: We say that a and b are coprime if $gcd(a, b) \equiv 1$. Let m be a positive integer. We define $\phi(m)$ to be the number of positive integers smaller than m that are coprime with m.

Example: Let $m \equiv 7$, note that all the positive integers smaller than 7 are coprime with 7. This implies that $\phi(7) \equiv 6$.

Observation: If m is a prime number, then all the positive numbers smaller than m are coprime with m. Thus, $\phi(m) \equiv m - 1$.

Example: As mentioned in the last example, $\phi(7) \equiv 6$. This illustrates the formula $\phi(m) \equiv m - 1$ for m prime.

On the other hand, let' s consider a case where m is not prime. For example, let $m = 6$. The positive integers smaller 6 that are coprime with 6 and 1 and 5. Thus, $\phi(6) \equiv 2$.

Observation: Let p and q be two different primes, then $\phi(pq) \equiv (p - 1)(q - 1)$.

The above observation results from the fact that the positive numbers that are smaller than pq and that are not coprime with pq are p, $2p$, ..., $(q - 1)p$ and q, $2q$, ..., $(p - 1)q$. These two lists do not have any numbers in common. The first list has $q - 1$ numbers. The second list has $p - 1$. Thus, the number of positive integers smaller than pq that are coprime with pq is$pq - 1 - (q - 1) - (p - 1) = (p - 1)(q - 1)$, which shows that $\phi(pq) \equiv (p - 1)(q - 1)$.

While the above formula can be generalized to compute $\phi(m)$, where m is not the product of two different integers, we will only need to consider a case more general than the one in the above observation.

Example: $\phi(15) \equiv (3 - 1)(5 - 1) = 8$, which can be verified directly since the positive integers smaller than 15 that are comprime with 15 are 1, 2, 4, 7, 8, 11, 13, 14.

Fermat’s Euler Theorem

The RSA algorithm is based on the following theorem as Fermat’s little theorem

Theorem: Let a and m be coprime integers. Then, $a^{\phi(m)} \equiv 1\ (modm)$.

Example: Let a = 7 and m = 15. Note that gcd(7,15) = 1. In other words, a = 7and m = 15 satisfy the hypothesis of Fermat ’ s little theorem. Note also that ϕ(15) = 8. Thus, this theorem tell us that $7^8 \equiv 1\ (mod\ 15)$. Let’ s verify that this is the case with direct computations. $7^8 = (7^2)^4 = 49^4$, and $49 \equiv 4\ (mod\ 15)$. Thus, $7^8 \equiv 4^4\ (mod15)$. But $4^4 = (4^2)^2 = 16^2$. Since $16 \equiv 1(mod15)$, we have that $4^4 \equiv 1^2 = 1\ (mod\ 15)$

Solvingthe Equation $xk \equiv b(modm)$ for x

Assume that we are given positive integers k, b and m, with m positive. The goal ofthis section is to find a non-negative integer x such that $xk \equiv b(modm)$ and x<m.

Observation: Let k, b and m be three positive integers that satisfy the following:

1. $m = pq$ with p and q prime numbers.
2. $gcd(b, m) = 1$
3. $gcd(k, (p-1)(q-1)) = 1$

Let u and v be integers such that $uk + v(p-1)(q-1) = 1$ and u positive. Let . $x = (b^u)\ \%\ m$. Then, x is a non-negative integer smaller than m and x satisfies

$$x^k \equiv b(modm) \tag{7}$$

Example: Let k = 3, b = 2 and m = 15. Note that the condition gcd(b,m) = 1 becomes gcd(2, 15) = 1, which is correct. Note that $m = pq$ with p = 3 and q. Note that we have $(p-1)(q-1)$ = 8. We use Euclidean algorithm to find u and v, with u positive, such that $uk + v(p-1)(q-1) = 1$. We obtain u = 3 and v =– 1. We compute now $x = (b^u)\ \%m = 2^3\%15 = 8$. The observation says that $x = 8$ is asolution of $x^3 \equiv 2(mod\ 15)$. In fact, $8^3 = 8^2 8 = (64)8 \equiv (4)8(mod\ 15) = 32(mod\ 15) \equiv 2(mod\ 15)$.

To prove the validity of this observation, note the following:

a. $1 = uk + v(p-1)(q-1)$

b. $\phi(m) = (p-1)(q-1)$

c. $b^{\phi(m)} \equiv 1(modm)$

Thus,wehavethat

$$b = b^1 = b^{ul+v(p-1)(q-1)} = (b^u)^k \left(b^{(p-1)(q-1)}\right)^v = (b^u)^k \left(b^{\phi(m)}\right)^v \tag{8}$$

This last equation and the above facts imply that $b \equiv b\ (modm)$. Thus, $x = (b^u)\ \%m$ satisfies $x^k \equiv b(modm)$ and x<m.

Factoring Large Integers Is Hard

Let m be a positive integer. Suppose we want to factorize m. That means, we want to find the sequence $k1$, $k2$,..., kn, where each ki is a prime number and $ki \leqslant ki$+1 for all i.A straightforward algorithm to find this sequence of primes is to initially by set i = 1 and k = 2. At each step, we compute $m\%k$. It is easy to check that, having $m\%k$ = 0, implies that k is prime. Thus, if $m\%k$ = 0, we set $ki = k$, keep the value of k the same, replace m by m/k, and we increase i by 1. On the other hand, if $m\%k \neq 0$, we increase k by one, and keep both i and m the same. We stop once m = 1. The table below illustrates this algorithm. In this table, we factorize 45, to find that 45 = 3 * 3 * 5.

m	k	i	$k1, k2, ., kn$	$m\%k$
45	2	1		1
45	3	1		0
15	3	2	3	0
5	3	3	3,3	2
5	4	3	3, 3	1
5	5	3	3, 3	0
1	5	4	3,3,5	

Table 1. Illustration of the standard algorithm to factorize integers. Factorization of45.

The algorithm we have just described is simple and it is the standard way we all factorize integers. The important fact to note is that this algorithm is not fast. This means that computational time execute the algorithm increases fast with m. This means that factorizing very large becomes prohibitively computationally expensive. In other words, we cannot factorize very large integers with this algorithm. In fact, we cannot factorize very large integers with any algorithm. Nobody has discovered an efficient algorithm to factorize very large integers. This is the key fact that makes the RSA algorithm that we will describe in the next section secure.

III. RSA ALGORITHM

Assume we have two parties, the sender and the receiver. The sender wants to send a message to the receiver. This message may be text, but any text can be converted into a positive integer, so assume the message the sender wants to send the receiver is a positive integer x. We think of s as a large integer, but that is irrelevant.

Assume this message x is a secret. In other words, neither the sender nor the receiver wants anyone else to know the value of the number x. The challenge is that the medium through which the message travels from the sender to the receiver is not secured. Thus, before sending the number x, the sender changes this number to a different number that we call b. We call this number b the encoded message. The sender sends this number b to the receiver, who receives b. Note that other parties may also get a hold of this number b. Nevertheless, b is a useless number. The valuable information or number is b. As we will explain in this section, the RSA algorithm allows only the receiver to recover x from b. The other parties that intercepted the number $b$can not recover the original number x. This makes the RSA algorithm a secure means of sending messages.

The RSA algorithm works as follows:

1. Both the sender and the receiver agree on two large prime numbers, p and q.Nobody else knowsthese numbers.

2. Let x be the message (number) the sender wants to send the receiver. Onlythe sender knows the number x.

3. Let k be a positive integer such that $k = gcd(k, (p-1)(q-1)) = 1$.Everyone knows k. This number k is known as the public key.

4. While only the sender and the receiver know p and q, everyone knows thenumber $m = pq$.

5. The sender computes $b = x^k (modm)$ and send b to the receiver. Everyonecan see b.

6. The receiver recovers the original message x by solving $b = x^k (modm)$ for x with the algorithm described in the previous section. The receiver can easily solve for x because the receiver knows p and q. No one lease can solve for x because no one else knows p and q and factorizing large numbers, such as m, is computationally prohibitively expensive.

IV. CONCLUSION

Number Theory is a classical and old field of mathematics that has attracted the efforts of famous mathematicians. This field is the building block of Cryptography. In this article, we have explained the mathematics behind the RSA algorithm.

The field of cryptography is percolating to everyone's lives. Most recently, cryptocurrencies are becoming more and more popular and are threatening to disrupt the currency system as we know it. Cryptography is a fascinating subject at the intersection of mathematics and computer science. At this point, it is difficult to tell the impact that cryptography, and thus, indirectly, mathematics, will end up having on society, but it certainly promises to be a fascinating and interesting tale.

REFERENCES

[1]. Silverman, J.H., 2014. *A friendly introduction to number theory*. Pearson.

[2]. Hoffstein, J., Pipher, J., Silverman, J.H. and Silverman, J.H., 2008. *An introduction to mathematical cryptography* (Vol. 1). New York:springer.

[3]. Davies, D., 1997. A brief history of cryptography. *Information Security Technical Report*, *2*(2), pp.14-17.

[4]. Mahajan, P. and Sachdeva, A., 2013. A study of encryption algorithms AES, DES and RSA for security. *Global Journal of Computer Science and Technology*.