Int. J. Math. And Appl., 8(2)(2020), 99–106
ISSN: 2347-1557
Available Online: http://ijmaa.in/

International Journal of *Mathematics And its Applications*

Applications of Number Theory in RSA Encryption Systems

Shreyan Das[1,*]

1 Bridgewater-Raritan High School, Bridgewater, New Jersey, USA.

Abstract: Number theory is the branch of mathematics that studies the set of integers. Cryptography is the field of study that has as goal the development of algorithms to send secret messages over public channels like the internet. In this article, we describe applications of number theory to cryptography, specifically within the RSA Encryptions system. This usage allows for this type of Encryption to be significantly advantageous.

Keywords: Number Theory, RSA Encryption, Euclidean Algorithm, Euler's Formula, Cryptography.

1. Introduction

Consider the following scenario. Sam and Ray are school classmates. During a recess, Sam tells Ray that he will send him a message during class. The message will be written in a piece of paper. Since they sit across the room, separated by several peers, other students will pass the paper across the room so it reaches Ray. Thus, the paper will be in the hands of several students that will be able to read it. Since Sam does not want anyone but Ray to be able to understand the message, Sam tells Ray: *I am going to change each letter in the message with the next letter in the alphabet, and each letter z with a letter a, so to be able to understand the message, once you receive it, you have to change each letter with the previous letter in the alphabet and replace any letter a with a letter z.*

During class, Sam writes in a piece of paper *Ep zpv xbou up dpnf up nz ipvtf bgufs tdippm?*. He gives the paper to the student sitting next to him and tells him that the paper is for Ray and to pass it. The student does, and after the paper goes through a few students, it reaches Ray. The students that had the paper in their hands read it, but none of them understood its meaning. Ray gets the paper. After following the rule given by Sam on how to change the letters, the message reads *Do you want to come to my house after school?*.

The action that Sam did of changing the original message *Do you want to come to my house after school?* to *Ep zpv xbou up dpnf up nz ipvtf bgufs tdippm?* is called encrypting the message. We say that *Ep zpv xbou up dpnf up nz ipvtf bgufs tdippm?* is the encrypted message. Ray's action of changing the encrypted message *Ep zpv xbou up dpnf up nz ipvtf bgufs tdippm?* to the original message *Do you want to come to my house after school?* is called decrypting or decoding the message.

The field that develops algorithms or rules to encrypt messages, such as the one used by Sam, is called Cryptography. For an introduction to the subject we refer the reader to [1]. The original main use of Cryptography was the sending of messages

* *E-mail: shreyan.das@live.com*

during wars, mainly through basic encryptions, e.g. a classic Caesar shift cipher as demonstrated by Sam and Ray. The purpose of encrypting the messages was that they would be incomprehensible if the enemy intercepted them. Nowadays, cryptography is used in virtually all the transaction done over the internet, and enhanced by using mathematical efficiencies to make the problem of decrypting a message without valuable information as unsolvable as possible.

In this article, we will explain one widely used cryptographic algorithm known as RSA [1]. We will also explain the the mathematics of Number Theory [2] that is necessary to understand why the RSA algorithm works. We will implement our algorithms with the computer language Python.

This article is organized as follows. In Section 2 we introduce the concepts of divisibility and greatest common divisor. In Section 3 we describe the Euclidean Algorithm. In Section 4 we discuss prime numbers and factorization of composites. In Section 5 we introduce the notion congruence. In Section 6 we describe how to solve linear equations with only one unknown but in the context of congruences. In Section 7 we explain the particular case of Euler's formula that we need. In Section 8 we describe the core mathematics behind the RSA algorithm. In Section 9 we describe the RSA algorithm. We end the article with a small discussion in Section 10.

2. Divisibility: The Greatest Common Divisor

Definition 2.1. *Let a and b be two integers. We say that b divides a, or that b is a divisor of a or that b is a factor of a if there is another integer x such that $a = bx$.*

For example, 3 divides 12 because $12 = 3(4)$.

Fact 2.2. *Let a and b be two integers. There exists a unique integer q and a unique non-negative integer r such that $r < |b|$ and $a = qb + r$. We say that q is the quotient of a divided by b and r the remainder.*

In the above fact $|b|$ denotes the absolute value of b.

For example, if we divide 30 by 11, the quotient is 2 and the remainder is 8 because $30 = 2(11) + 8$ and $0 \leq 8 < 11$.

Note that the statement b divides a is equivalent to saying that the remainder of a divided by b is 0.

We will use the computer language Python to implement the algorithms we describe in this article. Computing the integer quotient of a and b and the remainder of a divided by b is very easy in Python. It is simply done with the commands $a//b$ and $a\%b$ respectively. For example, $7//3 = 2$ and $7\%3 = 1$, notating that the integer division of 7 and 3 is 2 and the remainder of 7 dividing 3 is 1. We will use this notation in this article.

Notation 2.3. *Let a and b be integers. The quotient and the remainder of a divided by b are denoted in this article by $a//b$ and $a\%b$ respectively.*

Definition 2.4. *Let a and b be two integers. The greatest common divisor of a and b, that is denoted by $\gcd(a, b)$, is the largest positive number that divides both a and b.*

For example, the greatest common divisor of 30 and 18 is 6. In mathematical notation, $\gcd(30, 18) = 6$.

Fact 2.5. *The following equation is valid for all integers a and b*

$$\gcd(a, b) = \gcd(b, a\%b). \tag{1}$$

For example, if $a = 30$ and $b = 18$, we have that $a\%b = 30\%18 = 12$. Thus, Equation (1) says that $\gcd(30, 18) = \gcd(18, 12)$, and in fact, we do have that $\gcd(30, 18) = 6$ and $\gcd(18, 12) = 6$.

Fact 2.6. *Given two integers a and b, there exists two other integers u and v such that*

$$\gcd(a,b) = ua + vb. \tag{2}$$

For example, $\gcd(30,18) = 6$ and $6 = (-1)30 + (2)18$. In this example, $a = 30$, $b = 18$, $\gcd(a,b) = 6$, $u = -1$ and $v = 2$. Note that $6 = (2)30 + (-3)18$, and thus, in this example, we could have also selected $u = 2$ and $v = -3$. In other words, the pair of numbers u and v is not unique.

3. The Euclidean Algorithm

Let a and b be two non-zero integers. We define the following sequence of integers $r_0, r_1, \ldots, r_n$ as follows:

$$r_0 = a, r_1 = b, \text{ then for } i \geq 1, \text{ if } r_i \neq 0, \text{ then } r_{i+1} = r_{i-1}\%r_i, \tag{3}$$

where n is the integer defined by the fact that $r_{n-1} \neq 0$ and $r_n = 0$.
Note that Equation (3) means that $r_2 = r_0\%r_1$, $r_3 = r_1\%r_2$ and so on.
For example, if $a = 30$ and $b = 18$, then $r_0 = 30$, $r_1 = 18$, $r_2 = 30\%18 = 6$ and $r_3 = 18\%6 = 0$. Note also that $r_{n-1} = r_2 = \gcd(30,18) = \gcd(a,b)$. This is a general fact that results from Fact 2.5. More precisely, we have the following:

Fact 3.1. *Let $r_0, r_1, \ldots, r_n$ be the sequence defined by Equation (3), where n is the positive integer such that $r_{n-1} \neq 0$ and $r_n = 0$. Then $\gcd(a,b) = r_{n-1}$.*

This last fact provides an algorithm to compute the greatest common divisor between two numbers, which becomes more efficient as a and b become large. This was written as an algorithm in Python. The Python codes in this article are in boldface. The Python function that is shown below and we called **gcd**, takes as input two integers a and b and returns their greatest common divisor.

```
def gcd(a,b):
    rp, rc = a, b
    while rc != 0:
        rp, rc = rc, rp%rc
    return rp
```

For example, **gcd**$(30,18)$ returns the number 6.
Let a and b be two non-zero integers. Let $r_0, r_1, \ldots, r_n$ be the sequence defined in Equation (3). We define sequences $u_0, u_1, \ldots, u_n$ and $v_0, v_1, \ldots, v_n$ as follows:

$$u_0 = 1, u_1 = 0, \quad \text{for } 1 \leq i < n, \quad u_{i+1} = u_{i-1} - (r_{i-1}//r_i)u_i \tag{4}$$

$$v_0 = 0, v_1 = 1, \quad \text{for } 1 \leq i < n, \quad v_{i+1} = v_{i-1} - (r_{i-1}//r_i)v_i. \tag{5}$$

Note that $u_0a + v_0b = r_0$ because $u_0a + v_0b = 1r_0 + 0r_1 = r_0$. Similarly, note that $u_1a + v_1b = r_1$ because $u_1a + v_1b = 0r_0 + 1r_1 = r_1$. We also have that $u_2a + v_2b = r_2$ because $u_2a + v_2b = (u_0 - (r_0//r_1)u_1)a + (v_0 - (r_0//r_1)v_1)b = (u_0a + v_0b) - (r_0//r_1)(u_1a + v_1b) = r_0 - (r_0//r_1)r_1 = r_0\%r_1 = r_2$. In fact, we have the following:

Fact 3.2. *For $0 \leq i < n$, we have $u_i a + v_i b = r_i$. In particular, since $r_{n-1} = \gcd(a, b)$, we have*

$$u_{n-1}a + v_{n-1}b = \gcd(a, b). \tag{6}$$

Using the last fact, we define a Python function **gcdr** that takes as input two integers, a and b and returns three integers d, u, v, such that $d = \gcd(a, b)$ and $ua + vb = d$.

```
def gcdr(a,b):
    rp, rc = a, b
    while rc != 0:
        q = rp//rc
        up, uc = uc, up - q*uc
        vp, vc = vc, vp - q*vc
        rp, rc = rc, rp - q*rc
    return rp, up, vp
```

For example, **gcd**$(30, 18)$ returns the three number $6, -1, 2$ and in fact, $6 = \gcd(30, 18)$ and $-1(30) + 2(18) = 6$.

4. Factorization of Primes

Definition 4.1. *A positive integer p is said to be prime if $p > 1$, and the only positive integers that divide p are 1 and p.*

For example, 7 is prime, but 12 is not, because 3 divides 12 and $3 \neq 1$ and $3 \neq 12$.

Fact 4.2. *Let n be a positive integer. There exists a unique finite sequence of primes $p_1 < p_2 < \ldots < p_k$ and a unique finite sequence of non-negative integers $n_1, n_2, \ldots, n_k$ such that $n = p_1^{n_1} p_2^{n_2} \ldots p_k^{n_k}$. This is called the prime factorization of n*

For example $125 = 5^3$, $120 = 2^3 3(5)$, $17 = 17$ are the factorization of a few integers.
This fact can also be demonstrated using Python. A function **primeFactorization** can take any natural number and return the factorization of the number into primes raised to powers, as follows:

```
def primeFactorization(n):
    ncount = 0
    r = []
    while n % 2 == 0:
        nCount, n = nCount+1, n/2
    if nCount > 0:
        r.append([2,nCount])
    nCount = 0
    for i in range(3, int(math.sqrt(n)+1), 2):
        while n % i == 0:
            nCount, n = nCount+1, n/i
        if nCount > 0:
```

```
            r.append([i, nCount])
        nCount = 0
    if n > 2:
        r.append([int(n), 1]
    return r
```

For example, this program **primeFactorization**(2400) would print an output of [[2,5], [3,1], [5,2]], which correctly shows that $2400 = 2^5 3^1 5^2$.

5. Congruences

Definition 5.1. *Let a and b be two integers. Let m be a positive integer. We say that a is congruent b modulus m, and denote this by $a \equiv b \pmod{m}$ if m divides $a - b$*

For example, 12 is congruent 2 modulus 5, i.e. $12 \equiv 2 \pmod{5}$ because 5 divides $12 - 2 = 10$.

Fact 5.2. *Let a be an integer and m a positive integer. Then,*

$$a \equiv a\%m \pmod{m}. \tag{7}$$

For example, $17 \equiv 17\%4 \pmod{4}$. In fact, $17\%4 = 1$ and 4 divides $17 - 1 = 16$.

Fact 5.3. *Let a_1, a_2, b_1 and b_2 be an integers. Let m be a positive integer. Assume that $a_1 \equiv a_2 \pmod{m}$ and $b_1 \equiv b_2 \pmod{m}$. Then,*

$$a_1 + b_1 \equiv a_2 + b_2 \pmod{m} \text{ and } a_1 b_1 \equiv a_2 b_2 \pmod{m}. \tag{8}$$

For example, $7 \equiv 2 \pmod{5}$, $3 \equiv 8 \pmod{5}$ and $10 = 7 + 3 \equiv 10 = 2 + 8 \pmod{5}$ and $21 = 7(3) \equiv 16 = 2(8) \pmod{5}$.

6. Solving the Equation $ax \equiv c \pmod{m}$ for x

Assume that we are given integers a, c and m, with m positive. The goal of this section is to find x such that $ax \equiv c \pmod{m}$ and $1 \leq x < m$.

Fact 6.1. *Let $d = \gcd(a, m)$. Assume d divides c. Let u and v be integers such that $ua + vm = d$. Let $x = (uc/d)\%m$. Then $ax \equiv c \pmod{m}$ and $1 \leq x < m$.*

The validity of this fact results from the following string of congruences. $ax = a((uc/d)\%m) \equiv auc/d \equiv (auc/d + vmc/d) \equiv (c/d)(au + vm) \equiv (c/d)d \equiv c \pmod{m}$.
Using the last theorem, we define a Python function **axcm** that takes as input three integers, a, c and m such that $\gcd(a, m)$ divides c, and returns x the only solution of $ax \equiv c \pmod{m}$ with $1 \leq x < m$.

```
def axcm(a,c,m):
    d,u,v = gcdr(a,m)
    return (u*(c//d))%m
```

For example, **axcm**$(5, 4, 17)$ returns the the integer 11, and in fact $5x \equiv 4 \pmod{17}$ because $5(11) - 4 = 51 = 17(3)$ and thus, 17 divides $5(11) - 4$.

7. Particular Case of Euler's Formula

Fact 7.1. *Let p and q be two different primes. Let $m = pq$. Let a be an integer such that $\gcd(a, m) = 1$. Then, $a^{(p-1)(q-1)} \equiv 1 \pmod{m}$. This is a particular case of a more general result known as Euler's formula.*

To illustrate this fact, we consider $p = 3$ and $q = 5$. On Table 1, we show all the powers of a^k modulus 15 for $1 \leq k \leq 8$, and for all a such that $\gcd(a, 15) = 1$ and $1 \leq a < 15$. We, in fact, see that $a^{(p-1)(q-1)} = a^8 \equiv 1 \pmod{15}$ for all such numbers a.

a	a^2	a^3	a^4	a^5	a^6	a^7	a^8
1	1	1	1	1	1	1	1
2	4	8	1	2	4	8	1
4	1	4	1	4	1	4	1
7	4	13	1	7	4	13	1
8	1	8	1	8	1	8	1
11	1	11	1	11	1	11	1
13	4	7	1	13	4	7	1
14	1	14	1	14	1	14	1

Table 1. **Table of powers of a modulus 15 for all a such that $\gcd(a, 15) = 1$.**

8. Solving the Equation $x^k \equiv b \pmod{m}$ for x

Fact 8.1. *Let b, m and k be three integers that satisfy the following:*

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

Let u and v be integers such that $uk + v(p-1)(q-1) = 1$. Then, the solution of

$$x^k \equiv b \pmod{m} \tag{9}$$

that satisfies $1 \leq x < m$ is

$$x = (b^u)\%m. \tag{10}$$

To understand the validity of this fact, first note that $x^k = ((b^u)\%m)^k \equiv ((b^u))^k \pmod{m} \equiv (b^{uk}) \pmod{m}$, i.e.

$$x^k \equiv b^{uk} \pmod{m}. \tag{11}$$

Since $\gcd(b, m) = 1$, from Fact 7.1 we have that $b^{(p-1)(q-1)} \equiv 1 \pmod{m}$ and thus, $1 \equiv 1^v \equiv (b^{(p-1)(q-1)})^v \equiv b^{v(p-1)(q-1)} \pmod{m}$ i.e.

$$1 \equiv b^{v(p-1)(q-1)} \pmod{m}. \tag{12}$$

The product of the left hand sides of Equations (11) and (12) is congruent to the product of the right hand sides modulus m. Thus, we have $x^k \equiv b^{uk} b^{v(p-1)(q-1)} \equiv b^{uk+v(p-1)(q-1)} \equiv b^1 \equiv b \pmod{m}$, which proves the validity of Fact 8.1.

As an example, consider the case $p = 3$, $q = 5$, $b = 2$ and $k = 3$. In this case, $(p-1)(q-1) = 8$. Note that $3(3) + (-1)8 = 1$ and thus, the u of Fact 8.1 is 3, i.e. $u = 3$. Note also that in this case $m = 15$. Fact 8.1 tells us that the solution of $x^3 \equiv 2 \pmod{15}$ with $1 \leq x < m$ should be $x = (2^3)\%15 = 8$, In fact, $8^3 \equiv 64(8) \equiv 4(8) \equiv 32 \equiv 2 \pmod{15}$.
The Python function below, **xkbm**, takes as input 4 integers, k, b, p, q, that satisfy the three conditions listed in Fact 8.1, and returns as output the integer x that satisfies $x^k \equiv b \pmod{m}$ and $1 \leq x < m$

```
def xkbm(k,b,p,q):
    d,u,v = gcdr(k,(p-1)(q-1))
    return (b**u)%m
```

9. Applications to Cryptography

Sam, the sender, wants to send a message to Ray, the receiver. This message will be a number (it is easy to map text to numbers and vice-versa). Sam wants to keep the message secret and thus, will encrypt it in case it is intercepted. Ray will decrypt it once received. They use the following strategy:

(1). In some way that we will not address, and prior to sending the message, they select together two large prime numbers p and q that only Sam and Ray know. No one else knows about their selection.

(2). Let $m = pq$. Sam selects k such that $\gcd(k, (p-1)(q-1)) = 1$. He publishes both k and m. That means that everyone, not just Sam and Ray, knows the value of k and m.

(3). Sam wants to send the message s, which is a number, to Ray. The only restriction on s is that $1 \leq s < m$. Instead of sending s, he computes $e = (s^k)\%m$ and send e to Ray.

(4). Since Ray knows the values of p, q and k, and we have that $\gcd(k, (p-1)(q-1)) = 1$, Ray can solve for x the the equation $x^k \equiv e \pmod{m}$ with $1 \leq x < m$. But $s^k \equiv e \pmod{m}$ and $1 \leq s < m$. Since the solution for x is unique, we have that $x = s$, and Ray can successfully decode the message.

Since the values of p and q are needed to solve the equation $x^k \equiv e \pmod{m}$ with $1 \leq x < m$ in reasonable time and Ray is the only one (other than Sam) that has these values, Ray is the only one able to decode the message. This method that Sam and Ray are using is widely used and is known as an RSA encryption. Below we display the Python function **enc** that encrypts the message.

```
def enc(s,k,m):
    return (s**k)%m
```

Below is the Python function **enc** that decrypts the message. We do not pass p and q as arguments to stress the fact that both the sender and receiver know these primes before the message is sent.

```
def dec(k,e):
    p,q = .....
```

return xkbm(k,e,p,q)

Notice that if an outside source were to try and decrypt the message knowing only the public information of k, e and m, and m(rooted from the values of p and q) is sufficiently large, the outside source would be tasked with an unreasonable problem in testing every such instance of primes p and q in the decryption algorithm.

10. Discussion

In this article we describe the theory behind the widely used RSA algorithm in Cryptography. Advanced details, such as how to compute powers efficiently are left out. Cryptography is one of the many examples of the power of mathematics. Cryptography has drastically evolved from its early days of simple ciphers, and the usage of number theory concepts make encryptions such as RSA secure even with the usage of computers. Number theory allows for stronger encryptions, and as computers get stronger, encryptions gets stronger. As quantum computers become prevalent, new encryptions grow to keep information safe from unwanted eyes.

11. Acknowledgements

This paper was written with the mentoring of Dr.Guillermo Goldsztein, Department of Mathematics, Georgia Institute of Technology, Atlanta, GA 30332, United States.

References

[1] Joseph H. Silverman, *A Friendly Introduction to Number Theory*, Am. Math. Compet., 10(2006), 12.

[2] Douglas Robert Stinson and Maura Paterson, *Cryptography: theory and practice*, CRC Press, (2018).