Theoretical Mathematics, The Creation of The RSA Algorithm, and Breaking It Using Algorithms Based on The Same Idea

Muhammad Galih Raihan Ramadhan 13519017
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13519017@itb.ac.id

Over the course of human history, cryptography has been used to mask information from certain bodies or groups of people. It has been a very important part in the past, and such is also the case in the current times. One of such cryptography methods, which are widely used in terms of sending data, is known as the RSA algorithm. This algorithm has been used ever since it was discovered up until know, proving just how difficult of a task it is to break it, and also just how sufficiently strong it is as a cryptography algorithm. It is not invincible however, given enough time and information, any kind of algorithm will be broken by a sufficiently strong machine. While such machine that is able to decrypt the algorithm on a reasonable time has not been made yet, mathematicians has been trying over time to find algorithm that may help with such cases, by expanding the fields of factorization algorithm, which is the base idea of the RSA algorithm.

Keywords Factorization Algorithm, Prime Numbers, RSA Algorithm, Theoretical Mathematics

I. Introduction

Theoretical mathematics and cryptography has a rather significant relationship, while it may not seem so. Theoretical mathematics plays an important role in the development in the fields of cryptography, both in creating the encrypting theorem and also the decrypting theorem. This paper will discuss the use of the theoretical mathematics in cryptography, more specifically the rather infamous RSA algorithm.

While discussing how theoretical mathematics are used in the RSA algorithm itself, this paper will also discuss possible uses for those who seek to break the algorithm, since doing so would also allow researchers to improve this or other already existing algorithm.

It needs to be noted though, not every available theorem that can be used to break the RSA algorithm will be discussed here. The algorithm that has been chosen, all of them has been chosen with significant amount of a gap in the time of their discoveries, with the first one being discussed being the earliest and the last being the latest. Such thing is chosen to show how, over time the fields of cryptography has grown alongside theoretical mathematics, even though the RSA algorithm barely has any changes ever since it was discovered.

II. Theoretical Mathematics And It's Significance In Cryptography

A. Theoretical Mathematics and Cryptography

Theoretical mathematics, as the name may imply is math done for its own sake, rather than to be used to understand the real world. While it may have originated from real world problems, theoretical mathematics are not motivated by the applications in the real world. Regardless, this branch of mathematics is often used again in other branches of science, such as physics after much significance innovation regarding the said theory.

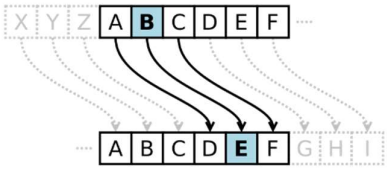
On the other hand, cryptography is said to be "An Art To Mask Information" which many interpret to change an information, it can be anything, letters, numbers, or symbols into another form of it with certain rules. While it may be useful to hide this information from certain people, without a way to turning it back to its original form, it also useless to those who has access and would like to see the original, yet to be cryptographed information. Due to this, a certain set of rules are also needed to change the already changed information into the original information. Each of these rules, including those that are used to change the original information need to be consistent with every possible component that could be part of the information, if such state of rules is able to be created, then it can be used safely, disregarding the component of the information, as long that every component of it is able to be changed with the said rules, and vice versa. As an added note, the action of changing the original information into another form is usually called encoding or encrypting, while changing it back to the original form is called decoding or decrypting.

B. Caesar Cipher

Starting from the beginning, theoretical mathematics has been used in cryptography since the days of the Ancient Romans, whether intentionally or not. The most famous, and coincidentally also the simplest example of such cryptography is known as the Caesar Cipher.

The image below shows a simple illustration for the innerworkings of the Caesar Cipher. As stated above, it is a quite simple form of cryptography, it simply changes whatever letter in the alphabet by the letter a certain space after or before it. The image above shows a Caesar Cipher where each letter is

encrypted to be the letter three spaces after it, therefore A becomes D, B becomes E, C becomes F and so on. As another example, say we have the sentence "Meet me at the restaurant".



Source http://www.science4all.org/article/cryptography-and-number-theory/

If we were to encrypt it using this Caesar Cipher, it would become "Phhw ph dw wkh uhvwdxudqw" which sounds like nothing that makes any sense in the English language, therefore the encrypting works perfectly, as any person without knowing how to decrypt the message would not be able to understand it. As for decrypting it, since the method of encrypting it is to change every letter into the letter three spaces after it, then we simply need to change each letter into the letter 3 spaces before it to decrypt it back to the original message.

Now with that being said, where does theoretical mathematics comes into the picture? Since the Caesar Cipher was created long ago, it is safe to assume that it was never used to encrypt something that would take a lot of time to decrypt, let's say for an example, the Cipher was never used to encrypt a message that has more than one hundred thousand letters, because that would take a person a really long time to decipher. However, since we live in the current age and society, we have the capability to program a computer to both encrypt and decrypt a message using the Caesar Cipher.

We start of by assigning each letter to a number, as an example the letter "a" becomes the number "1", the letter "b" becomes the number "2" and so on. For this example, we would treat capital letters the same way we would treat a lowercase letter. Using the English alphabet, we would get the numbers between the range [1-26]. To encrypt a message using the Cipher that was previously shown, we change the letter into the corresponding letter, add it by two then change it back into the corresponding letter.

The equations below show step by step the encryption of the letter "a" using the Cipher, we will denote this action using *f(a)*.

$$f(a) = 1 \tag{1}$$

$$f(a) = 1 + 2 = 3 \tag{2}$$

$$f(a) = 3 = c \tag{3}$$

So now we know the equation works for the letter "a", but how about the letter at the end portion of the alphabet? We will now find it out by trying the algorithm with the letter "z".

$$f(z) = 26 \tag{4}$$

$$f(z) = 26 + 2 = 28 \tag{5}$$

Since we assign only numbers in the range [1-26] the number twenty-eight cannot be changed back into a letter. So instead of just adding the number by two, we will now be using the *mod* operator which returns the remainder of the division operator between two numbers, as shown below.

$$23 \ (mod \ 4) = 3 \tag{6}$$

If we implement this into the algorithm, using the number of letters in the alphabet as the number we *mod* the letter we want to encrypt with,

$$f(z) = 26 \tag{7}$$

$$f(z) = 26 + 2 = 28 \tag{8}$$

$$f(z) = 28 \ (mod \ 26) = 2 \tag{9}$$

$$f(z) = 2 = b \tag{10}$$

And now we can encrypt the letter "z". Coincidentally, since "z" is the last letter in the alphabet and also assigned to the highest number in the range we are using, we can be sure that every letter in the alphabet can now be encrypted and decrypted.

This method might be sufficient for the times of the Caesars, however as time goes, there have been methods created to cipher such a simple encrypting method. One of the easiest way to deciphering such method is by looking for the letter with the most frequency and assuming it is the in actuality, the most used letter in the language the message is written.

As seen in Fig. 1, the most used letter in the English alphabet is "e". Using this knowledge we can ascertain certain characteristic of the encrypting algorithm. Let's use our encrypted message from earlier, "Phhw ph dw wkh uhvwdxudqw". In this message we can see that the letter "h" has the highest frequency of appearing in the sentence, and it indeed is the letter "e" in the original message, therefore the assumption is correct. After the discovery of that fact, we now know that each letter is changed with the corresponding letter, that is three spaces after it. Even if the first assumption is wrong, we can use the next letter with the highest appearance frequency and do the same method until we get the correct message.

Such an easily decoded method of encrypting proved to not be quite useful at all to encrypt any kind of data in the modern age, therefore researchers, with the help of mathematicians have developed other, much harder encryption method. The method that will be the main focus for the rest of this paper is named the RSA algorithm, founded back in 1978, by Ron Rivest, Adi Shamir, and Leonard Adleman [1].

III. Prime Numbers and Fermat's Little Theory

Prime numbers are defined to be numbers that are only able to be divided completely by that number itself or the number one.

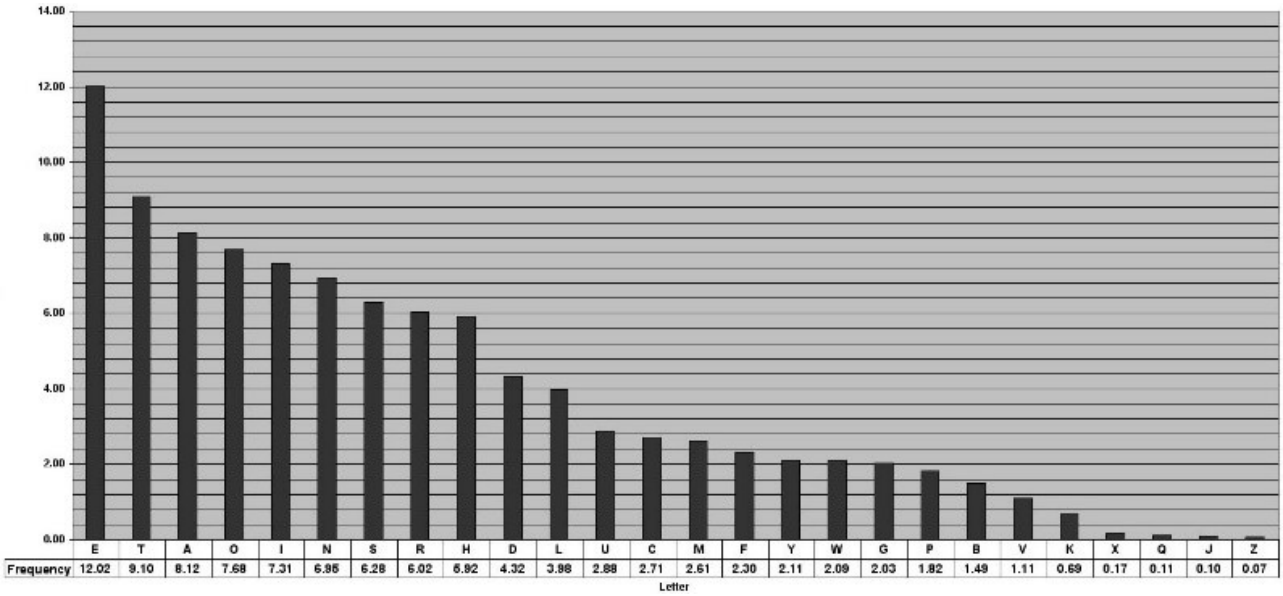


Fig. 1 The Frequency of Letters Used in The English Alphabet

Source http://www.science4all.org/article/cryptography-and-number-theory/

In other words, if one were to have a prime number n and apply the operation *mod m* where m is a positive number which is equal or less than n, it would only result in zero if m is one or n itself. With that being said, Fermat's little theory is a theorem that states, for every natural numbers a. and every prime number p, the equation below stands

$$a^p \equiv a \ (mod\ p) \tag{11}$$

To put in another way, for every a that is a power of p, if one were to divide it by p itself, the remainder would always be a. The intricacies of this theorem will not be discussed on this paper, however please do understand that it is a very important theorem when we are talking about the application of theoretical mathematics in cryptography.

IV. The RSA Algorithm

A. Sending Data

The RSA algorithm is, as stated before, an encryption algorithm that is quite safe and hard to decrypt and used to send data without the fear that other groups of people other than the intended receiver will be able to understand. To do this, the algorithm is programmed to send not only the encrypted data, but also a set of other information, being the *public encryption key* and a *signature*.

The *public encryption key*, along with the *decryption key* is a special set of data that is different for each sender and receiver that sends data using the RSA algorithm. The decryption key however, is not public, as in the only the user knows the actual value of the *decryption key*, unlike the *public encryption key*, as the name may suggest, which is a public information.

These data are not only used to encrypt and decrypt data, it is also used to produce the signature to be sent from the sender to the receiver. Signatures are data that are used to confirm that the information that was received by the receiver is sent by the correct sender and not from any other sources. In RSA, the signature is produced by decrypting using the sender's decryption key, then we encrypt the decrypted data using the receiver's *public encryption key*. This way, the receiver will be able to deduce that the sent data is indeed sent from the correct sender, as the receiver could easily decrypt the signature with their decryption key, then continue to encrypt it with the sender's encryption key, which will return the original message.

C. Encryption and Decryption Keys

The method of determining the encryption and decryption keys are, arguably quite simple, especially if we compare it with the multitude of theorem that has been tried and used to break it in a swift manner.

First, we determine two reasonably large prime numbers, p and q. The sizes of these numbers in digit will signify how long it will take for someone trying to decrypt a message encrypted using the method. The numbers that are usually used to encrypt data for public use, span for as long one hundred digits.

Now, we will produce the encryption and decryption keys, respectively we will refer as e and d. Since d acts as the decryption keys, the data that will not be shared to the public, it is preferable for it to have a much higher value compared to e. Then d is defined by the equation,

$$gcd\big(d, (p-1)(q-1)\big) = 1 \tag{12}$$

where *gcd* is defined to be the greatest common divider between 2 numbers, in this case being d and $(p-1)$ multiplied by $(q-1)$. And e will be defined by any number that fulfills the

equation

$$ed = 1\ (mod\ \varphi(n))$$ (13)

In this equation the symbol φ is known as the Euler totient function, which produces the number of positive numbers that is less than *n*, where *n* is *pq*. For primes *p*, it is clear that

$$\varphi(p) = p - 1$$ (14)

which for *n* equals to *pq* we can expand

$$\varphi(n) = \varphi(p)\ .\ \varphi(q)$$

$$\varphi(n) = (p-1)(q-1)$$

$$\varphi(n) = n - (p+q) + 1$$ (15)

and if we combine it with (13), we will be able to get the equation

$$ed = k\ .\varphi(n) + 1$$ (16)

for any integers *k.*

Now if we want to encrypt some data, we will need to convert it first into an integer. Say for example, we want to encrypt a text message and we convert it into an integer using the same method we used while discussing the Caesar Cipher. Now that we have converted the data into the integer *M*, we will encrypt it into the integer *C*, using the equation

$$C \equiv M^e (mod\ n)$$ (17)

and if we want to decrypt it, we simply use the equation

$$M \equiv C^d (mod\ n)$$ (18)

The consistency of this theorem can be proven by examining Fermat's Little Theorem, (11) and (13). Which allows us to yield the equation

$$m^{ed} \equiv m (mod\ n)$$ (19)

D. Security

As stated before, the level of security given by this theorem lies in the size of the integer *p* and *q*. The larger they are, the longer it will take for anyone to break through it. The reason for it is because, any theorem that are used to factorize a certain number will always take a significant amount of time, significance enough that, given that both *p* and *q* are at least 100 digits long, it will take such a long time, that it won't be practical to do.

Table I

Comparison for The Size of n *and The Time Taken to Break The RSA Algorithm Using The Schroeppel Factoring Algorithm*

Digits	Number of operations	Time
50	1.4×10^{10}	3.9 hours
75	9.0×10^{12}	104 days
100	2.3×10^{15}	74 years
200	1.2×10^{23}	3.8×10^{9} years
300	1.5×10^{29}	4.9×10^{15} years
500	1.3×10^{39}	4.2×10^{25} years

Source Milanov, Evgeny. The RSA Algorithm, *page 9*

V. BREAKING THE RSA

Breaking the RSA, in reality only requires us to factor the integer *n* into it's two components, *p* and *q*. Therefore, if we can discover an algorithm that is able to factorize such a big number over a short period of time, the RSA algorithm will no longer prove to be useful as a encrypting algorithm. Luckily, mathematicians have been rigorous and tireless in their efforts to find these theorems. Here we will discuss some of the algorithms.

A. Fermat Factorization Method

This method, as the name states, comes from the very same person that discovered *Fermat's Little Theorem.* This method states, that given a positive odd integer *n*, it can be represented as [3]

$$n = x^2 - y^2$$ (20)

which also gives us

$$n = (x - y)(x + y)$$ (21)

and if we were to equate $(x - y)$ and $(x + y)$ both as a factor of *n*, both *a* and *b*, respectively. In this method we will start by using the value

$$x_i = \lceil \sqrt{n} \rceil$$ (22)

and substitute it into (21). Need to be noted that $\lceil n \rceil$ represents the ceiling function of *n*. And then we check if Δx_i is a square number in the following equation

$$\Delta x_i = {x_i}^2 - n$$ (23)

if turns out to be true, then we can generate these two facts regarding *n* and its factors

$$x = x_i \tag{24}$$

and

$$y = \sqrt{\Delta x_i} \tag{25}$$

If it isn't a square however, we repeat the step again this time using x_2 equals to one over x_1 and so on, until there exists a value x_i where Δx_i is a square number.

As one may guess, this method is categorized to be rather slow, since on average, one will need to reiterate starting with a number that has half the digits of *n* until it gets to the middle point between it and the integer *n*. As an illustration, if our *n* is the number 10^{100} then on average, we will need to reiterate about over the range of $[10^{50}, 10^{75}]$. Needless to say that this is a huge range to be reiterating each integer one by one, therefore most would say that it is not an advisable way to decrypt the RSA algorithm.

B. Quadratic Sieve

The Quadratic Sieve is an algorithm created in 1981 by Carl Pomerance, extending ideas from Kraitchik and Dixon. It was known to be the fastest factoring algorithm before the discovery of the Number Field Sieve in 1993 [4]. This method works by first finding *x* and *y* with respect to *n* that fulfills the equations

$$x \not\equiv \pm y \ (mod\ n) \tag{26}$$

and

$$x^2 \equiv y^2 \ (mod\ n) \tag{27}$$

which combined, implies

$$(x - y)(x + y) \equiv 0 \tag{28}$$

We will then compute if the integers $(x - y,\ n)$ using the Euclidean Algorithm to see if it is indeed a nontrivial divisor. There is at least a 50% chance for the factor to be nontrivial. The steps used to find *x* and *y* starts by defining *r* as

$$r = \lfloor \sqrt{n} \rfloor + k \tag{29}$$

and compute it for *k* equals to 1,2,3,... and $\lfloor n \rfloor$ being the floor function of *n*. Next, we look for factors *p* such that

$$n \equiv r^2 \ (mod\ p) \tag{30}$$

where *n* and *p* fulfill the Legendre Symbol

$$\left(\frac{n}{p}\right) = 1 \tag{31}$$

Then, compute the equation

$$x^2 \equiv n \ (mod\ p) \tag{32}$$

which must be solved for every integer *p*. Finally, a *sieve* is applied to find the values

$$f(r) = r^2 - n \tag{33}$$

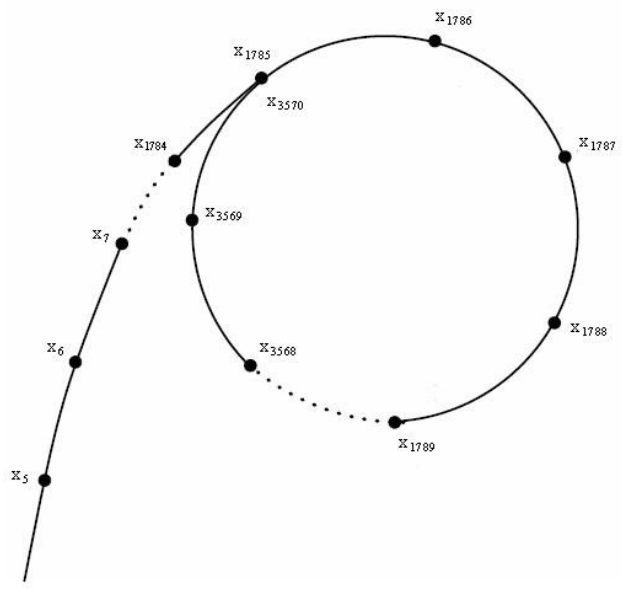
Gaussian elimination then can be used to find the product of *f(r)*, which yields a perfect square. This method takes about *exp(sqrt(ln n ln ln n))* steps, which is faster than the then previously fastest factoring algorithm being the *Continued Fraction Factorization Algorithm*.

C. Pollard's Rho Algorithm

Pollard's Rho Algorithm is a prime factorization algorithm known also as the *Pollard Monte Carlo Factorization Method* [6]. It starts by first defining *n* as a multiplication product of *p* and *q*. and now we iterate the equation

$$x_{n+1} = ({x_n}^2 + 1) \ (mod\ n) \tag{34}$$

starting on any integer. Reiterating the equation over and over will eventually produce a cycle, which is represented by the image below



Source
https://en.wikipedia.org/wiki/File:Pollard_rho_cycle.jpg

While doing this, for every iteration of the equation, we see if

it falls into a cycle using *Floyd's Cycle-Finding Algorithm*, where it keeps two nodes, i and j [7]. After each step, we evaluate

$$gcd(x_i - x_j, n) \neq 1 \tag{35}$$

If the evaluation is correct, then it implies there is a repetition in the sequence, and therefore whatever the actual value of the *gcd* operation should be a nontrivial divisor of *n.* However, it may also return *n* as well, in which case the method will need to be repeated with different parameters.

VI. Conclusion

As a conclusion, I would like refer to the beginning part of this paper, where the significance of theoretical mathematics and cryptography are being discussed. As stated, it has been in this state for ages, and it will stay in a quite similar state for the foreseeable future. Although there may be a or some breakthrough in the fields of science that may provide us with the ability to discover new applications for theoretical physics in this field of computer science, the tried and true algorithm will always be available for us to refer back to it and perhaps search for ways to improve it, instead of looking for a new one from scratch.

VII. Acknowledgment

I would like to thank Mr. Rinaldi Munir for assigning with such a task, thus allowing to be able to train ourselves, his students, to be comfortable with the process to making a research paper.

References

[1] Milanov, Evgeny. "*The RSA Algorithm*." From https://sites.math.washington.edu/~morrow/336_09/papers/Yevgeny.pdf, accessed on December 2nd 2020, at 13.50.

[2] http://www.science4all.org/article/cryptography-and-number-theory/ accessed on December 2nd 2020, at 1.50.

[3] Weisstein, Eric W. *"Fermat's Factorization Method."* From *MathWorld*—A Wolfram Web Resource. https://mathworld.wolfram.com/FermatsFactorizationMethod.html accessed on December 7 2020 at 12.41

[4] Landquist, Eric. *"The Quadratic Sieve Factoring Algorithm."* From www.cs.virginia.edu/crab/QFS_Simple.pdf accessed on December 9th 2020, at 9.41.

[5] Weisstein, Eric W. *"Quadratic Sieve."* From *MathWorld*—A Wolfram Web Resource. https://mathworld.wolfram.com/QuadraticSieve.html accessed on December 9th at 7.40.

[6] Weisstein, Eric W. *"Pollard rho Factorization Method."* From *MathWorld*—A Wolfram Web Resource. https://mathworld.wolfram.com/PollardRhoFactorizationMethod.html accessed on December 9th at 16.21.

[7] https://www.uio.no/studier/emner/matnat/ifi/INF3380/v12/undervisningsmateriale/inf3380-floyd.pdf accessed on December 9th 2020 at 16.26.

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 11 Desember 2020

M Galih R R 13519017