

Number Theory Application to Attack An Unsecure RSA

Frankie Huang - 13521092
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13521092@std.stei.itb.ac.id

Abstract—Data transmission is one of the most important aspects of the digital age. This requires the transmission to be secure enough to prevent an unwanted party accessing the valuable data by applying an encryption. One of the most widely used encryption methods is the RSA algorithm. However, if the key chosen are not secure enough, a malicious party will be able to decrypt the encrypted data.

Keywords—Cryptography, Decryption, RSA, Security

I. INTRODUCTION

In today's digital world, communication has become as important as food, shelter, and clothes for living day-to-day life. As a result, the challenge is to provide means for secure communication to allow people to communicate securely over insecure mediums. Information security plays an important role in protecting digital information against security threats by keeping the information a secret to prevent access by an unauthorized party. Information must be hidden from the unauthorized users (confidentiality), prevents any form of modifications (integrity) and must be readily available to an authorized party (availability). Also known as the CIA triad, the value of an information comprises confidentiality, integrity, and availability, which are considered as the three main goals of security. There are several ways to implement these security goals, among which cryptography is the most general and widely used technique.

In today's standard, RSA is the most used algorithm technique used to encrypt data. The RSA Cryptosystem, also known as the RSA algorithm, was developed in 1977 by three people: Ronald Rivest, Adi Shamir, and Len Adleman; which is based upon the difficulty of factorization of two large primes. The cryptosystem is most commonly used for providing privacy and ensuring authenticity of digital data. These days, RSA is deployed in many commercial systems. It is used by web servers and browsers to secure web traffic, to secure login sessions, and it is at the heart of electronic credit card payment systems. The RSA cryptosystem has been analyzed for vulnerability by many experts. Although no method has been found to completely decrypt an RSA encryption easily, there still exists dangers lurking due to an improper method of generating an RSA. In this paper, I would like to explore and explain the cause of some of these insecure RSA and the possible threat looming because of it. Throughout

this paper, I will use Alice and Bob to denote two generic parties wishing to communicate secretly with each other.

II. THEORETICAL FRAMEWORK

A. Divisibility, Factors and Prime Numbers

If a and b are integers with $a \neq 0$, we say that a divides b if there is an integer c such that $b = ac$, or equivalently, if $\frac{b}{a}$ is an integer. When a divides b , we say that a is a factor or divisor of b , and that b is a multiple of a . The notation $a \mid b$ denotes that a divides b and the notation $a \nmid b$ denotes that a does not divide b [1].

If a does not divide b , then when b is divided by a , there is a quotient q and a remainder r , as shown below.

$$b = qa + r \quad (1)$$

The factors of a number n are defined as a set of numbers $f_1, f_2, f_3, \dots, f_k$, where $1 \leq f_i \leq n$ and $f_i \mid n$; or formally as

$$F = \{f_i : 1 \leq f_i \leq n, f_i \mid n\}$$

An integer p is called a *prime* if the only factors of p are 1 and p . Typically, the sieve of eratosthenes is used to find all primes not exceeding a specified integer. A special type of prime numbers, called *Mersenne primes* are prime numbers that takes the form of $m_p = 2^k - 1$. To test whether m_p is a *Mersenne prime*, we can use the Lucas-Lehmer test [2], which is defined as the recurrence equation below.

$$s_n = s_{n-1}^2 - 2 \pmod{m_p} \\ s_0 = 4$$

The Lucas-Lehmer test defines m_p as a *Mersenne prime*, if and only if $s_{p-2} \equiv 0 \pmod{m_p}$. For example, the sequence obtained for $p = 7$ is given by 4, 14, 67, 42, 111, 0; hence m_7 is prime.

B. Modular Arithmetics

Modular arithmetic, also known as clock arithmetic, is a special type of arithmetic equation where we are only interested in finding the remainder of a division. From (1), we can change the equation into the form of modular arithmetic as follows.

$$b \equiv r \pmod{a}$$

The triple equal sign \equiv denotes *congruence*, and for all a and b , we say a is congruent to $(b + a) \pmod{a}$.

Just like in regular arithmetics, there exist an inverse modulo \bar{a} of a modulo m , such that $a\bar{a} \equiv 1 \pmod{m}$. However, in modular arithmetics not every number has an inverse modulo m . This is due to the fact that there exists an inverse of a modulo m , if and only if $\gcd(a, m) = 1$.

C. Greatest Common Divisor, Euclidean Algorithm, and Linear Congruence

The largest integer that divides both of two integers is called the *greatest common divisor*[1] of these integers. Let a and b be positive integers, there exists such integer d , where $d \mid a$ and $d \mid b$, such that no integer larger than d can divide both a and b . The greatest common divisor of a and b is denoted by $\gcd(a, b)$.

Computing the \gcd of two integers by searching the largest prime factorizations is inefficient, because it is time consuming to find the prime factorizations. Hence, we will use the *Euclidean Algorithm* to efficiently find the \gcd of two numbers. From (1), we can simplify the process of finding a \gcd of two numbers by using the euclidean algorithm defined as follows.

$$\gcd(a,b) = \gcd(b,r) \tag{2}$$

A congruence in the form of

$$ax \equiv b \pmod{m} \tag{3}$$

where m is a positive integer, a and b are integers, and x is a variable, is called a *linear congruence*. There are two ways to solve the linear congruence, the first method is to find the inverse modulo \bar{a} , such that $a\bar{a} \equiv 1 \pmod{m}$. Then, we can multiply (3) with the inverse modulo, such that

$$\begin{aligned} a\bar{a}x &\equiv b\bar{a} \pmod{m} \\ x &\equiv b\bar{a} \pmod{m} \end{aligned}$$

The second method involves brute forcing the value of x by rearranging (3) into

$$\begin{aligned} ax &= km + b \\ x &= \frac{km+b}{a} \end{aligned}$$

D. System of Linear Congruences and Chinese Remainder Theorem

A *system of linear congruences* [1] is a system consisting of more than one linear congruence.

$$\begin{aligned} x &\equiv a_1 \pmod{m_1} \\ x &\equiv a_2 \pmod{m_2} \\ x &\equiv a_3 \pmod{m_3} \\ &\vdots \\ x &\equiv a_n \pmod{m_n} \end{aligned}$$

We can solve the system above using the *Chinese Remainder Theorem* [1]. The *Chinese Remainder Theorem* states that when the moduli of problems involving linear congruences are pairwise relatively prime, there is a unique solution of the system modulo the product of the moduli.

Let $m_1, m_2, m_3, \dots, m_n$ be pairwise relatively prime positive integers greater than one and $a_1, a_2, a_3, \dots, a_n$ arbitrary integers. Then the system

$$\begin{aligned} x &\equiv a_1 \pmod{m_1} \\ x &\equiv a_2 \pmod{m_2} \\ x &\equiv a_3 \pmod{m_3} \\ &\vdots \\ x &\equiv a_n \pmod{m_n} \end{aligned}$$

has a unique solution modulo $m = m_1m_2m_3\dots m_n$. (That is, there is a solution x with $0 \leq x < m$, and all other solutions are congruent modulo m to this solution.)

Generally, the solution of a linear congruence system is defined as

$$x = a_1M_1y_1 + a_2M_2y_2 + \dots + a_nM_ny_n \tag{4}$$

where

$$\begin{aligned} M_i &= \frac{m}{m_i} \\ y_i &\equiv \bar{M}_i \pmod{m_i} \end{aligned}$$

E. Fermat's Little Theorem

Fermat's Little theorem[3] states that if p is prime and a is an integer not divisible by p , then

$$a^{p-1} \equiv 1 \pmod{p} \tag{5}$$

Furthermore, for every a , we have $a^p \equiv a \pmod{p}$

We can use (5) to quickly compute the modular inverse of a .

$$\begin{aligned} a^{p-1} &\equiv 1 \pmod{p} \\ a^{p-1}a^{-1} &\equiv a^{-1} \pmod{p} \\ a^{p-2} &\equiv a^{-1} \pmod{p} \end{aligned}$$

However, there exist composite numbers n , such that the number n satisfies *Fermat's Little Theorem*. Such numbers are called *pseudoprimes*.

F. Continued Fractions and Convergents

Continued Fractions [4] is a way to represent real numbers as a sequence of positive integers. Let's say we have a real number α . We can form a sequence of positive integers from α by

$$\alpha = \alpha_0 + \frac{1}{\alpha_1 + \frac{1}{\alpha_2 + \frac{1}{\dots}}}$$

where

$$\begin{aligned}\alpha_i &= \lfloor \alpha_{i-1} \rfloor \\ \alpha_0 &= \lfloor \alpha \rfloor\end{aligned}$$

We can represent the continued fractions of α as

$$[\alpha_0; \alpha_1, \alpha_2, \alpha_3, \dots, \alpha_n]$$

The k -th convergence of a continued fraction is defined as the approximation of the continued fraction using only the first k terms of the sequence.

$$C_k = \alpha_0 + \frac{1}{\alpha_1 + \frac{1}{\alpha_2 + \frac{1}{\dots + \frac{1}{\alpha_k}}}} \quad (6)$$

G. Euler's Totient Function and Euler's Formula

The *Euler's Totient Function*[3] of an integer n ($\phi(N)$) is the amount of integer from $\{1, 2, 3, \dots, n-1\}$ that are relatively prime to n . Furthermore, it can be proven that if $\gcd(p, q) = 1$, then $\phi(pq) = \phi(p)\phi(q)$.

Let's assume the integer n consists of k amount of distinct prime factors, such that $n = p_1^{e_1} p_2^{e_2} p_3^{e_3} \dots p_k^{e_k}$. Then, we can compute the value of n by

$$\phi(n) = n \prod_{i=1}^k \left(1 - \frac{1}{p_i}\right) \quad (7)$$

Then, it can be proven trivially that if p is prime, then $\phi(p) = p - 1$.

Euler's Formula states that if $\gcd(a, p) = 1$, then

$$a^{\phi(n)} \equiv 1 \pmod{n} \quad (8)$$

H. Fermat Factorization

Fermat Factorization[5] states that every odd integer p and q can be stated as the difference between two integer, such that

$$n = a^2 - b^2 = pq$$

Then, we can refactor the equation above into

$$n = (a + b)(a - b) = pq \quad (9)$$

where

$$\begin{aligned}p &= (a + b) \\ q &= (a - b)\end{aligned}$$

I. RSA Algorithm

The *RSA Algorithm* [6] is an example of asymmetric encryption, also known as public key encryption, where a public key is used to encrypt data and only a secret, private key can be used to decrypt the data.

Common terminologies used in the *RSA Algorithm* includes:

1. The message m ;

2. The encrypted message c ;
3. The modulo n , which is made by multiplying two prime numbers p and q ;
4. The public exponent e ;
5. The private exponent d ;
6. The totient value $\phi(n)$;
7. The public key pair (n, e) ; and
8. The private key pair (n, d) .

To generate the encryption and decryption keys, first compute n as the value of two primes p and q .

$$n = pq$$

These primes should be very large, "random" primes. Although the value n will be public, the factors p and q will be effectively hidden from everyone due to the enormous difficulty of factoring n . This same concept also hides the way d can be derived from e .

Then, we calculate the totient $\phi(n)$ to generate the value d .

$$\phi(n) = (p - 1)(q - 1) \quad (10)$$

Then, we can pick the public exponent e , most commonly 65537. The private exponent d is finally computed by finding the modular inverse of e modulo $\phi(n)$.

$$ed \equiv 1 \pmod{\phi(n)} \quad (11)$$

Or from (2), the value d must be relatively prime to $\phi(n)$. That is, check that d satisfies

$$\gcd(d, \phi(n)) = 1 \quad (12)$$

To encrypt the message M using the public key pair (n, e) , first represent the message into an integer m between 0 and $n-1$. Then, encrypt the message by raising it to the e -th power modulo n . That is, the result (the ciphertext C) is the remainder when m^e is divided by n .

$$C \equiv M^e \pmod{n} \quad (13)$$

To decrypt the ciphertext, we raise it to the d -th power modulo n .

$$M \equiv C^d \pmod{n} \quad (14)$$

III. TYPES OF UNSECURE RSA

For an RSA to be insecure, one must pick a bad number such that the ciphertext can be easily deciphered using a number of methods. We will talk about attacks on RSA caused by a weak public exponent and a poor choice of p and q which results in a value of N that can be factorized or attacked in an easier way.

A. Public Exponent Attacks

Let's assume the sender, Alice, wants to send a secret message to her friend, Bob. She encrypts it using RSA with p and q chosen from a set of strong primes. However, she made a fatal mistake of choosing a weak public exponent e , which opens the possibility of the following attacks.

1. $e = 1$

When $e = 1$, the encryption is virtually useless [7]. From (13), we can derive

$$\begin{aligned} C &\equiv M^e \pmod{n} \\ C &\equiv M \pmod{n} \\ C &= M \end{aligned}$$

Which means that the message is unencrypted and deciphering it is trivial.

2. Small e

In a single-party RSA, when e is small enough compared to the value of N , the encryption method is ineffective at encrypting m [8]. In this case, if $n > m^e$, then n is rendered virtually useless. Deriving from (13), we get

$$\begin{aligned} C &\equiv M^e \pmod{n} \\ C &= M^e \\ M &= \sqrt[e]{C} \end{aligned}$$

However, if $n < M^e$, then the encryption method is a bit more secure, but not impossible to crack. In this case, we can keep adding more multiples of n until the e -th root gives us an integer. Again, deriving from (13), we can get

$$\begin{aligned} C &\equiv M^e \pmod{n} \\ C &= kn + M^e \\ M^e &= C - kn \\ M &= \sqrt[e]{C - kn} \end{aligned}$$

Then, we can take the e -th root of $C - kn$ to get the message m .

3. Hastad's Broadcast Attack

If the value of M^e is far larger than the value of n , we can't use the previous method even if the value of e is small [9]. However, if we manage to get e amount of ciphertext c and moduli n , with the same message m and public exponent e , then the encryption method is no longer secure. Let's say that we've intercepted e amount of c and n , where for all value of c and n

$$C_i \equiv M \pmod{n_i}$$

Using (4), we may compute the value of C , such that

$$C_i \equiv C \pmod{n_i}$$

Consequently, because for all c and n we have the same message m , we can derive from (13) that

$$C \equiv M^e \pmod{n}$$

where

$$n = n_1 n_2 n_3 \dots n_e$$

Furthermore, because the value of e is small enough that for all values of n_i , $m < n_i$, we can say

that $M^e < n$. Thus we can remove the modulo as it is virtually useless.

$$\begin{aligned} C &\equiv M^e \pmod{n} \\ C &= M^e \\ M &= \sqrt[e]{C} \end{aligned}$$

Then, we can take the e -th root of c to get the message m .

4. Wiener's Attack

The *Wiener's Attack* utilizes the convergents of the continued fraction expansion of $\frac{k}{d}$ to attempt to guess the decryption exponent d when e is large, as d becomes smaller as a result [10]. There are 3 requirements for this attack to work [11]:

- $d < \frac{1}{3}n^{\frac{1}{4}}$
- $q < p < 2q$
- $e' < n^{\frac{3}{2}}$; where $e' \equiv e \pmod{\phi(N)}$

Recall that the euler's totient function is equal to the value of $(p-1)(q-1)$, which in itself is almost equal to the value of N . Deriving from (10), we get

$$\begin{aligned} \phi(n) &= pq - (p + q) + 1 \\ \phi(n) &\approx n \end{aligned} \quad (15)$$

Thus, according to (11), we can rearrange the equation, such that

$$\begin{aligned} ed &= k\phi(n) + 1 \\ ed - k\phi(n) &= 1 \\ \frac{e}{\phi(n)} - \frac{k}{d} &= \frac{1}{d\phi(n)} \end{aligned} \quad (16)$$

Since the value of $d\phi(n)$ is likely to be large, we can assume that $\frac{1}{d\phi(n)}$ is virtually zero. Then, from (15) and (16), we get

$$\frac{e}{n} \approx \frac{k}{d}$$

Recall that since e and n are both public information, we can approximate the value of k and d using continued fractions and convergence. Since the convergence of $\frac{k}{d}$ is approximately the value of $\frac{k}{d}$, and the value of $\frac{k}{d}$ is approximately $\frac{e}{n}$, then we can assume that the value of $\frac{k}{d}$ is approximately the convergence of $\frac{e}{n}$.

Backtracking a bit, we can rearrange (10) into

$$\phi(n) = \frac{ed-1}{k}$$

After calculating the convergence of $\frac{k}{d}$, we can check for all k and d whether $\phi(n)$ is an integer.

Continuing from (10), we can rearrange the equation to

$$\begin{aligned} \phi(n) &= pq - (p + q) + 1 \\ p + q &= n - \phi(n) + 1 \end{aligned} \quad (17)$$

Lastly, by constructing a quadratic equation, we can solve the value p and q by changing the value $p + q$ by using (17).

$$\begin{aligned}(x - p)(x - q) &= 0 \\ x^2 - (p + q)x + pq &= 0 \\ x^2 - ((n - \phi(n)) + 1)x + n &= 0\end{aligned}$$

We can verify the result by checking if the value p and q are the correct primes by multiplying the two values and checking if it results in n .

B. Poor Choices of Primes

Now, let's assume that Alice realized her mistake and changed her public key to be more secure. However, she made another mistake of choosing a set of poor prime numbers. This mistake opens up the possibility of the following attacks.

1. N is Prime

When N is prime, then $\phi(n) = n - 1$ due to the definition of Euler's totient function [12]. This makes decryption super simple, since the value e and n are public. Using (11), we can find the value of d by

$$d \equiv e^{-1} \pmod{(n - 1)}$$

Since the value d is known, we can then decrypt the ciphertext c using (14).

2. p or q are Mersenne Primes

If either p or q are Mersenne primes, then calculating the value is easy enough [13]. We just need to loop through all possible Mersenne primes and check if either p or q are that. When the value of either p or q are known, we can easily decrypt the ciphertext c as the value of e , N , and c are known.

3. Same p and q Value

If the value $p = q$, then $n = pq = p^2$ [14]. This makes decryption super simple, since the value n are public. Then, we can compute the value $\phi(n)$ to get the private exponent d . Lastly, we can decrypt the ciphertext c using (14) as the value of d is known.

4. Close p and q Value

When the value p and q are numerically close, we can use Fermat factorization to attempt to guess the prime number p and q [5]. By using (9), we can attempt to guess the value a and b , such that

$$\begin{aligned}b^2 &= a^2 - n \\ b &= \sqrt{a^2 - n}\end{aligned}$$

Then, we can start by assuming $a = \sqrt{n}$ rounded up to the nearest number, then adding it until b is a whole number.

IV. PREVENTIVE METHODS

To prevent a malicious party accessing a private message, we must pick a set of secure numbers for generating the RSA. There exists many tools, mathematicians, and computer

scientists alike trying to find new ways of attacking the RSA algorithm. Although no method of breaking RSA completely has been found, there exists many attacks that can be executed due to issues generated during RSA generation. These are some of the methods available to prevent an attack on the generated RSA.

1. Strong Primes

As a rule of thumb, the chosen RSA should have a set of strong primes p and q to prevent brute forcing the prime values. A good prime is indicated by a large enough prime number but also by checking whether the prime used is vulnerable (i.e. the chosen prime is a Mersenne prime).

2. Big Key Size

As a rule of thumb, RSA keys should be long enough to thwart any attack and make the system attack resistant. Several factors are taken into consideration while deciding the size of RSA key, important ones being: the value of data, expected lifetime of data, threat model, and best possible attack. Generally, most of the current standards require the use of 1024 bits for RSA keys, as 512 bits are deemed not secure enough.

3. Secure Public Exponent

In most RSA implementations, the public exponent e is usually chosen to be either 3 or 65537, which takes 2 and 17 modular multiplications respectively. Considering the existence of possible attacks, it is advised to use 65537 as the public exponent, as it is more secure.

4. Long Private Exponent

From the earlier section, we've known that Wiener's attack can be used if the chosen private exponent is small. For the typical 1024 bit key, a minimum of 300-bit private exponent is advised.

5. Additional Encodings

The message encrypted without any additional encodings besides RSA (commonly known as raw RSA) offers a weak level of security and should never be used. It is always suggested that the message should be encoded beforehand. There are a number of different encoding methods advised, most of them works by adding enough randomness as well as some redundancy. Additional care is needed when using a small public exponent and short padding. PKCS#1 [15] defines two encoding (padding) methods for encryption:

- PKCS#1v1.5 is a simple, ad-hoc design, which is widely deployed (SSL/TLS).
- Optimal Asymmetric Encryption Padding (OAEP) is more complex, but has much better security properties.

V. CONCLUSION

The RSA algorithm is essential in our modern world, used in various different security protocols and sending messages

from the sender to the intended recipient. This makes RSA a widely studied topic, as mathematicians, computer scientists, security experts are trying to find different types of attack possible to undermine the RSA algorithm. Although no definite method of breaking RSA completely has been found, there are many attacks which make a poorly generated RSA vulnerable to a number of different types of attacks, including attacks that are caused by a weak public exponent, poor choices of primes, and many more. To prevent these attacks, the RSA should be generated as securely as possible, but not too secure as it may waste essential computational power.

VI. ACKNOWLEDGMENT

First of all, the author would like to thank God for giving us patience and perseverance for completing the work successfully. The Author would also like to thank the lecturer of class 1 Discrete Mathematics, Mrs. Nur Ulfa Maulidevi of Bandung Institute of Technology. Next, The Author would like to thank the head lecturer of Discrete Mathematics, Mr. Rinaldi Munir, who provides the topics taught in class, of which is used in this paper. The Author will also take this opportunity to express his sincere thanks and deep gratitude to his friends and seniors who may have helped me in one way or another for completing my paper successfully.

REFERENCES


- [1] K. H. Rosen, *Discrete mathematics and its applications*. New York, Ny Mcgraw-Hill Education, 2019.
- [2] E. W. Weisstein, "Lucas-Lehmer Test," *mathworld.wolfram.com*. <https://mathworld.wolfram.com/Lucas-LehmerTest.html> (accessed Dec. 11, 2022).
- [3] "Rings, Fields and Euler's Totient Function - Crypto," *Gitbook.io*, 2022. <https://ir0nstone.gitbook.io/crypto/fundamentals/rings-fields-and-eulers-totient-function> (accessed Dec. 11, 2022).
- [4] "Continued Fractions - Crypto," *Gitbook.io*, 2022. <https://ir0nstone.gitbook.io/crypto/further-maths/continued-fractions> (accessed Dec. 11, 2022).
- [5] "Fermat Factorisation - Crypto," *Gitbook.io*, 2022. <https://ir0nstone.gitbook.io/crypto/rsa/choice-of-primes/fermat-factorisation> (accessed Dec. 11, 2022).
- [6] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 26, no. 1, pp. 96–99, Jan. 1983, doi: 10.1145/357980.358017.
- [7] "e=1 - Crypto," *Gitbook.io*, 2022. <https://ir0nstone.gitbook.io/crypto/rsa/public-exponent-attacks/e-1> (accessed Dec. 11, 2022).
- [8] "Small e - Crypto," *Gitbook.io*, 2022. <https://ir0nstone.gitbook.io/crypto/rsa/public-exponent-attacks/small-e> (accessed Dec. 11, 2022).
- [9] A. Chennagiri, "A Tutorial paper on Hastad Broadcast Attack." [Online]. Available: <http://koclab.cs.ucsb.edu/teaching/cren/project/2017/chennagiri.pdf>
- [10] "Wiener's Attack - Crypto," *Gitbook.io*, 2022. <https://ir0nstone.gitbook.io/crypto/rsa/public-exponent-attacks/wieners-attack> (accessed Dec. 11, 2022).
- [11] ENOENT, "Attacking RSA for fun and CTF points – part 2," *BitsDeep*, May 25, 2018. <https://bitsdeep.com/posts/attacking-rsa-for-fun-and-ctf-points-part-2/>
- [12] "N is prime - Crypto," *Gitbook.io*, 2022. <https://ir0nstone.gitbook.io/crypto/rsa/choice-of-primes/n-is-prime> (accessed Dec. 11, 2022).
- [13] "Mersenne Primes - Crypto," *Gitbook.io*, 2022. <https://ir0nstone.gitbook.io/crypto/rsa/choice-of-primes/mersenne-primes> (accessed Dec. 11, 2022).
- [14] "P=Q - Crypto," *Gitbook.io*, 2022. <https://ir0nstone.gitbook.io/crypto/rsa/choice-of-primes/p-q> (accessed Dec. 11, 2022).

- [15] K. Moriarty, B. Kaliski, J. Jonsson, and A. Rusch, "RFC 8017: PKCS #1: RSA Cryptography Specifications Version 2.2," *IETF Datatracker*, Nov. 13, 2016. <https://datatracker.ietf.org/doc/html/rfc8017> (accessed Dec. 11, 2022).

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, 12 Desember 2022



Frankie Huang
13521092