Complex Number Analysis in Shor Algorithm for Breaking RSA Encryption

Vincent Rionarlie / 13524031
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13524031@std.stei.itb.ac.id, vinctlie06@gmail.com

***Abstract*—This paper aims to explore the complex number usage in Shor Algorithm, that implements the quantum computing that uses the full concept of quantum mechanics. This paper explores in medium level of detail about quantum mechanics as well as quantum computing methods. The paper then ties the topic to how RSA work, and how Shor algorithm manage to break through the encryption method using quantum computing.**

***Keywords*—Shor, Complex, Quantum, Quantum Computing**

I. Introduction

Cryptography was a product of the need for much more secure information exchange. It has been around for almost 4 centuries, starting with the use of the classic substitution cipher ranging from 1900s B.C. to 500 A.D., as well as its variants that were developed over the period 16th-20th century (e.g., Vigenère cipher), followed by the near-modern invention of symmetrical cryptographic system in the 20th century which works by sharing a common secret key for both encryption and decryption (e.g., Enigma machine and DES)

Over the years, those systems were proven to be not as secure as they were once expected to be. It was due to relatively weak encryption algorithms, as well as the use of the same key for both encryption and decryption purposes. Despite, much later in the future, a very secure symmetric system came into surface, such as Advanced Encryption Standard / AES (2001), it still doesn't solve the main problem of a common key-sharing.

Fortunately, public key cryptography system had been around long before AES was invented. In contrast to symmetrical / secret key cryptography, public key cryptography implements a public key for encryption and a private key for decryption. This means that there's no need for a key distribution mechanism during information exchange. RSA is one of the safest examples of public key cryptography.

RSA works by generating two related keys based on large prime numbers. The information receiver chooses two large prime numbers, multiplies them to form a modulus, resulting in a public and a private key using inverses and modulus. The public key is sent to any sender and can be used publicly to encrypt data, while the private key is used by the receiver themselves to decrypt it.

RSA has been one of the few most used and secure public encryption systems. RSA is effective since it is very easy to multiply two prime numbers, but exponentially harder to factor them back into the original primes. This becomes even more evident as the prime numbers get way larger. Today, most RSA algorithms use a key size of 2048 – 4096, which represents the number of bits for the n value, or the product of the prime numbers. Current normal computers would take about billions, or even trillions of years to successfully factor out RSA-2048, a timescale irrelevant to the human era. It indicates how secure RSA algorithm really is, at least for normal computers.

Then comes quantum computing, a concept that has only recently come into surface of the modern technology industry. It differs from normal computing in the ability to compute lots of parallel processes simultaneously. As a result of the concept, an algorithm, namely Shor Algorithm, was created.

Shor's algorithm is proposed in 1994 by mathematician Peter Shor, and it was the first algorithm to show that a quantum computer could solve a practically important problem exponentially faster than classical computers. As an algorithm under quantum computing, Shor algorithm uses qubits (or equivalently a bit in a quantum computer) as its tiniest metric of operation, which by definition, directly implements complex numbers. Shor algorithm is theorized to be able to break RSA encryption in just a matter of days, even hours, in contrast to classical RSA decryption algorithm.

This study aims to show how Shor algorithm implements imaginary numbers in its calculations, as well as illustrating how Shor algorithm works compared to normal decrypting algorithms.

II. Theoretical Foundation

A. Complex Number

Complex number set ($\mathbb{C}$) is an extension of the real number set ($\mathbb{R}$), where it also contains numbers with imaginary values. An imaginary number, symbolized with i, is derived from the fact that we cannot square root a negative number.

$$i^2 = -1$$
$$i = \sqrt{-1}$$

As the definition proposed, a complex number consists

of two main components,

$$z = a + bi$$

where z, a, and b represent complex number, real term, and the coefficient of the imaginary number, respectively. In addition, there is also a conjugate term for every z, symbolized with $\bar{z}$, such that

$$\bar{z} = a - bi$$

Several axioms related to complex numbers are as follows:

Name	Content
Closure	For all z_1 and z_2, $z_1 + z_2 \in \mathbb{C}$ $z_1 z_2 \in \mathbb{C}$
Identity	For each z, there is an identity element 0 and 1, such that, $z + 0 = 0 + z = z$ $z(1) = (1)z = z$
Inverse	For each z, there is an inverse element -z and $\frac{1}{z}$, such that, $z + (-z) = (-z) + z = 0$ $z\left(\frac{1}{z}\right) = \left(\frac{1}{z}\right)z = 1$
Associativity	For all z_1, z_2, and z_3, $z_1 + (z_2 + z_3) = (z_1 + z_2) + z_3$ $z_1(z_2 z_3) = (z_1 z_2)z_3$
Commutativity	For all z_1 and z_2, $z_1 + z_2 = z_2 + z_1$ $z_1 z_2 = z_2 z_1$
Distributivity	For all z_1, z_2, and z_3, $z_1(z_2 + z_3) = z_1 z_2 + z_1 z_3$ $(z_1 + z_2)z_3 = z_1 z_3 + z_2 z_3$

The addition and subtraction operation of complex numbers treat the real term and the imaginary term of the number like two different variables (e.g., x can only be added with another x, and not any other variables). As for multiplication, it follows the same commutativity, associativity, and distributivity properties for any term's multiplication, but due to the definition of *i* itself, any i^2 term as a result of the associative multiplication must result to -1.

$$z_1 \cdot z_2 = (a_1 + b_1 i)(a_2 + b_2 i)$$
$$z_1 \cdot z_2 = (a_1 a_2 - b_1 b_2) + (a_1 b_2 + a_2 b_1)i$$

On the other hand, division operator requires the whole fraction to be multiplied by the denominator's conjugate over itself (resulting in 1).

$$\frac{z_1}{z_2} = \frac{a_1 + b_1}{a_2 + b_2} \cdot \frac{a_1 - b_2}{a_2 - b_2}$$

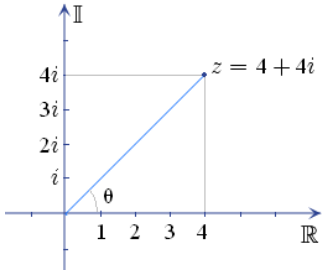
Like its real number counterpart, every complex number has an absolute value, oftentimes called modulus. While a real number's absolute value only indicates its non-negative magnitude, a complex number's modulus represents its overall magnitude (real term and imaginary coefficient).

$$|z| = \sqrt{a^2 + b^2}$$

This definition aligns with the geometry representation of a complex number as a vector in the Real-Imaginary Plane, more commonly known as the Complex Plane.

B. Complex Geometry Representation

A complex plane is a cartesian plane that consists of a real abscissa and an imaginary ordinate. Any complex number z can be visualized as a point of (a, b) in the complex plane.



To rotate a complex point, the Euler Equation can be used to form its image under the rotation.

$$z' = z.e^{i\theta}$$
$$z' = z.(cos\theta + i.sin\theta)$$

One other important representation of a complex number can be done within the polar coordinate system.

$$r = |z|$$
$$z = r.e^{i\theta}$$
$$z = r.(cos\theta + i.sin\theta)$$

This representation of complex number takes two arguments, such as its modulus, and the angle formed between the complex vector and the cartesian coordinate's positive x-axis.

C. Quantum Mechanics

Quantum mechanics describes the behavior of matter and energy at subatomic scales, a scale incomprehensible to us humans, where classical physics no longer applies. At this scale, particles have a wave-particle duality state and exist as probabilities rather than definite positions. That important concept is called a superposition: a principle that a quantum object exists in multiple states at once, weighted with complex probability numbers. One thought experiment that is most fond with this concept is the Schrödinger's Cat Experiment, by Erwin Schrödinger.

Another important concept to take note of is the Wave Function Collapse. It states that any quantum system, while having its own superposition as some functions' values, after being observed, will only 'point' to the basis state whose probability is the highest among all other states. Other states' probability will collapse (to 0%), while the observed final state is then 1 (or 100%).

D. Quantum Computing

Quantum computing system was a result of the need to simulate quantum mechanics, especially suggested by Richard Feynman. David Deutsch, later in 1982, was the first scientist to propose a model of quantum computation. Another question about whether the system would be useful for classical computation problems were, obviously, also raised.

To start off, quantum computing uses a two-state quantum system called a qubit, or quantum bit, as a replacement to the classical bit. A qubit takes a column vector value in the $\mathbb{C}^2$ vector (Hilbert) space. In contrast to

classical bits which represent information by a definitive set of 0s and 1s, where there can only be one state in which the bits take the result of, a qubit can have a lot of the states stored as probability measures at once.

For a qubit to make sense, we have to think like physicists. The quantum state of a qubit is described as a superposition of basis states, represented by $|0\rangle$ and $|1\rangle$, where,

$$|0\rangle = \begin{bmatrix}1\\0\end{bmatrix}, |1\rangle = \begin{bmatrix}0\\1\end{bmatrix}$$

A qubit exists as a linear combination of $\alpha|0\rangle + \beta|1\rangle$, where α and β are complex numbers satisfying the condition $|\alpha|^2 + |\beta|^2 = 1$. The condition ensures that the probability of measuring the qubit in any state sum to 100%.

To rewind, this state is called the superposition principle. Before measurement, the qubit exists in both states simultaneously, with $|\alpha|^2$ being the probability of observing $|0\rangle$ and $|\beta|^2$ the probability of observing $|1\rangle$. When a measurement is done on a qubit, the distribution collapses to only one of the basis states.

Multiple qubits can then be combined to form a quantum register. For n qubits, the system exists in a 2^n dimensional space, meaning that the register can represent 2^n different states in superposition at the same time, (**below is an example with 2 qubits**).

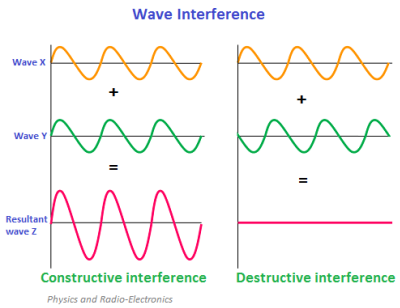
Amplitude	Basis State
α	$\|00\rangle$
β	$\|01\rangle$
γ	$\|10\rangle$
δ	$\|11\rangle$

where

$$|\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1$$

This exponential scale is quantum computing's biggest advantage compared to classical computing. Classical computation of n bits can only result in one of 2^n states at a point in time, but a quantum system of n qubits can process information about all 2^n states simultaneously.

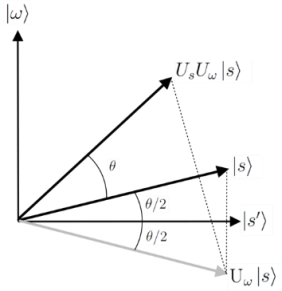
There is also a concept called interference that occurs when amplitudes add up with each other. Because amplitudes are complex numbers, they can add constructively or destructively. This mechanism is the core to achieving quantum computing.



Besides that, quantum gates are also the basic transformation 'methods' used to manipulate qubits, just like logical gates manipulate classical bits. One of them is the Hadamard gate which transforms a qubit to an equal state (same amplitude for every state).

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}$$

Another important gate is the CNOT gate that flips a qubit, inverting the qubit to $|0\rangle$ if and only if it has a value of $|1\rangle$. The combination of the gates makes up a quantum algorithm, completing calculation by 'modifying' the amplitudes of the states. One example is the Grover algorithm that repeatedly flips a qubit vector back and forth, which at the same time also keeps amplifying (increasing the amplitude of) the expected result state (the higher the amplitude, the higher the probability for the result state to come out as the expected).



E. RSA Algorithm

RSA system is achieved from the combination of multiplication and factorization. Like mentioned earlier, multiplying two large prime numbers is easy, but factoring out their original product is a different level of difficult, at least for classical computers.

RSA encryption algorithm can be realized through the steps below.

1) Select two large unique prime numbers p and q (preferably ≥1024 bits for each p and q),
2) Calculate $n = pq$
3) Calculate $\varphi(n) = (p-1)(q-1)$
4) Choose a private key d such that $1 < d < \varphi(n)$ and $\gcd(d, \varphi(n)) = 1$
5) Calculate a public key that satisfies $ed \equiv 1 \bmod \varphi(n)$
6) Encrypt the plaintext M as a ciphertext C, such that $C = M^e \bmod n$

To encrypt a string message *S* from a sender, we need to first convert *S*'s hex representation as an integer *M*. After doing this, we instruct the receiver's computer to send the public and private key to the sender in a form of (*n*, *e*). For the very first step, we find two large prime numbers with the help of Solovay-Strassen primality test to test the primality of a very big number. It, however, will not be discussed in this paper due to context limitations. After getting *n* as the product of both primes, we then follow the algorithm to find the private key *d* and public key *e*. One thing to note is that some RSA algorithms treat the private key *d* as the public key *e* and leave the rest of the algorithm to find the private key *d* instead. This alternative shows the same result in the end but might be a safer one since a

manually chosen component is treated as a public key instead of a private one.

To decrypt a ciphertext C, we will have to use the private key *d* that has been calculated in the receiver's end, so that the recovered plaintext $M = C^d \ mod \ n$. Now, it will be a very difficult task to find *d* only by the information of the public key (*n*, *e*). There are many ways to solve *d*, but in this paper's context, it will only be useful to focus on the method of factoring *n*. One might have realized intuitively that the job gets much, much easier once we factored out *n*. This is obviously because there will be only one possibility of a (*p*, *q*) pair, since they are both unique prime numbers. By doing that, one can immediately find φ(n) to be used in the rest of the algorithm.

Factoring *n* is not a simple task at all, in fact. With the *p* and *q* size of 1024 bits (or more) each, it will take a significant amount of time to compute. In 1978, the authors of RSA presented a table showing the number of operations and time needed to solve for an *n* with some length *k*.

k	Number of Operations	Time
50	$1.4 * 10^{10}$	3.9 days
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

The table is not as relevant anymore, as in 2020, the largest ever *n* that has successfully been factored with a general-purpose algorithm was 250 digits long, which is equivalent to about 830 bits. Nowadays, however, a typical RSA algorithm contains an over 1024 bits long of key, usually 2048 or 4096 bits long. This means, while it is possible that those numbers may be breakable in the future, it will still take a very long time to do the operations.

III. SHOR ALGORITHM

A. Overview

With that said, Shor's algorithm solves the integer factorization problem in polynomial time by reducing it to period finding—a problem quantum computers can solve way faster than classical computers. The algorithm consists of pre-processing and post-processing using a quantum system.

In simple steps, Shor algorithm works through the steps below:

1) If *N* is even, return factor 2
2) Check if $N = a^b$ for integers a, b ≥ 2; if so, return *a*
3) Choose random *a* where $1 < a < N$
4) Calculate $\gcd(a, N)$; if > 1, return this factor
5) Use quantum subroutine to find the period *r* of $f(x) = a^x \ mod \ N$
6) If *r* is odd or $a^{\frac{r}{2}} \equiv -1 \ mod \ N$, return to step 3
7) Compute $\gcd(a^{\frac{r}{2}} \pm 1, N)$ to obtain factors

The period *r* is the smallest positive integer such that $a^r \equiv 1 \ mod \ N$. Once *r* is found, if it is even, we can write:

$$a^r - 1 = \left(a^{\frac{r}{2}} - 1\right)\left(a^{\frac{r}{2}} + 1\right) \equiv 0 \ mod \ N$$

This basically means that *N* divides those two products. If none of the factors are divisible by *N* (which only fails when $a^{\frac{r}{2}} \equiv \pm 1 \ mod \ N$), then $\gcd(a^{\frac{r}{2}} - 1, N)$ and $\gcd(a^{\frac{r}{2}} + 1, N)$ contains non-trivial factors of N.

B. Quantum Period Finding

The quantum component of Shor algorithm determines the period *r* for $f(x) = a^x \ mod \ N$. This requires two quantum registers and proceeds through four stages:

1) Initialization

Prepare two quantum registers, with the first one having $n = [\log_2 N^2]$ qubits for input, and the second one with $[\log_2 N]$ qubits for output, with both initialized to $|0\rangle$:

2) Superposition Creation

Apply Hadamard gates to the first register, creating equal superposition for all input values:

$$|\psi_1\rangle = \left(\frac{1}{\sqrt{2^n}}\right) \sum_{x=0}^{2^n-1} |x\rangle|0\rangle$$

Now, each amplitude is the complex number $\frac{1}{\sqrt{2^n}}$, with all having the same probability values

3) Modular Exponentiation

Apply the unitary operator U_f that computes $f(x) = a^x \bmod N$ in the second register:

$$|\psi_2\rangle = \left(\frac{1}{\sqrt{2^n}}\right) \sum_{x=0}^{2^n-1} |x\rangle|a^x \bmod N\rangle$$

After this step, the quantum computer has simultaneously computed $a^x \bmod N$ for all 2^n values of x (a very useful parallelism). However, in this step, direct measurement would still only yield a random (x, $a^x \bmod N$) pair, collapsing the superposition without revealing the period.

4) Quantum Fourier Transform

Apply the Quantum Fourier Transform (QFT) to the first register. This method transforms the state to concentrate amplitude on values related to the period *r* so we can extract the value through measurement. The QFT is where complex numbers play their most critical role.

C. Period Extraction

The steps of QFT will not be discussed further since it will be very out of context. One thing to be sure is that after applying the QFT, measurement of the first register returns

a value y close to $\frac{mN}{r}$ for some integer m. The probability distribution peaks at these values due to constructive interferences during the QFT process. With this, *r* is already extracted. For comparison, Shor algorithm achieved $O(n^3)$ time complexity compared to the classical

IV. Analysis and Result

A. Role of Complex Number

Based on all the stuff the writer has written above, it can be concluded that Shor algorithm relies very, very heavily on complex numbers. The probability amplitudes α, β in qubit states $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ have to be a complex value to correctly describe any quantum phenomena.

Other than that, complex numbers are also inseparable from quantum gates. Many important gates have complex entries. The phase gate $P(\theta) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{bmatrix}$ for example, adds relative phase between $|0\rangle$ and $|1\rangle$ components—an operation that is impossible with only real values (since we would need imaginary values to add 'rotations').

B. Practical Experiment (Shor)

Consider factoring $N = 15$. Choose $a = 7$. We then try to find the period *r* of $f(x) = 7^x \bmod 15$:

$7^1 \bmod 15 = 7$
$7^2 \bmod 15 = 49 \bmod 15 = 4$
$7^3 \bmod 15 = 28 \bmod 15 = 13$
$7^4 \bmod 15 = 91 \bmod 15 = 1$

Hence, $r = 4$. Because *r* is even and $7^2 = 49 \equiv 4 \not\equiv -1 \pmod{15}$, we proceed with:

$\gcd(7^2 - 1, 15) = \gcd(48, 15) = 3$
$\gcd(7^2 + 1, 15) = \gcd(50, 15) = 5$

We obtain factors 3 and 5. The quantum algorithm finds r = 4 by noticing that the QFT outputs multiples of $\frac{N}{r} = \frac{16}{4} = 4$ (using N = 16 as the smallest power of $2 \geq 15$).

C. Implications for Cryptography

Shor's algorithm poses a threat to RSA and other factorization-based cryptosystems. Current quantum computers (as of 2024) have achieved hundreds of noisy qubits. However, this progress suggests a better quantum computers may start to surface within 10-20 years.

This timeline has pushed NIST to standardize post-quantum cryptographic algorithms, based on problems that are believed to be resistant to quantum attacks, such as lattice-based cryptography, hash-based signatures, and code-based cryptosystems. Organizations handling sensitive data must consider changing to quantum-resistant algorithms now, as encrypted data captured today could be decrypted once quantum computers mature.

V. Conclusion

This paper has demonstrated the fundamental role of complex numbers in the use of Shor's algorithm to break RSA encryption. Through deep analysis of the mathematical structures of quantum computation, we have shown that:

1) Complex numbers are inseparable from quantum mechanics, not just for convenience. Quantum state spaces are complex 'Hilbert' spaces. Any attempt to remake quantum computing with purely real numbers would obviously lower its computational power.
2) RSA's security has been proven weak against quantum computation. The integer factorization problem, almost infeasible for classical computers to solve, is reduced to polynomial-time complexity through quantum period finding and QFT. This fact threatens not just RSA, but also the entire factorization-based cryptographic.

The implications of the paper go beyond cryptography to fundamentals of computation. Complex numbers, that are once considered as just a mathematical abstractions, came in as a very important part for the most powerful known computational tool.

Future research should include developing quantum algorithms for other computationally hard problems, improving quantum error correction to make way for large-scale quantum computation, and designing post-quantum cryptographic systems to get more secure in the future of quantum world.

VI. Acknowledgement

The writer would like to give some acknowledgement towards:

1) Dr. Ir. Rinaldi Munir, M.T., for the chance to explore the world of linear algebra through our own observations. It has truly been a very insightful experience, knowing the writer had absolutely no knowledge about quantum mechanics before.
2) ITB, for the subscriptions to some very useful sources throughout the online library webs.

References

[11] IBM Newsroom (2023) IBM Debuts Next-Generation Quantum Processor & IBM Quantum System Two. Available at: https://newsroom.ibm.com/2023-12-04-IBM-Debuts-Next-Generation-Quantum-Processor-IBM-Quantum-System-Two (Accessed: 24 December 2024).
[10] National Institute of Standards and Technology (2024) NIST Releases First 3 Finalized Post-Quantum Encryption Standards. Available at: https://www.nist.gov/news-events/news/2024/08/nist-releases-first-3-finalized-post-quantum-encryption-standards (Accessed: 24 December 2024).
[9] Zimmermann, P. (2024) Integer Factoring Records. Available at: https://members.loria.fr/PZimmermann/records/factor.html (Accessed: 24 December 2024).
[8] Gyongyosi, L. and Imre, S. (2021) 'A Survey on Quantum Computing Technology', Quantum Information Processing, 20(5), pp. 1-57. Available at: https://link.springer.com/content/pdf/10.1007/s11128-021-03125-w.pdf (Accessed: 24 December 2024).

[7] Centre for Innovation in Mathematics Teaching (n.d.) Complex Numbers, CIMT Further Pure Mathematics. Available at: https://www.cimt.org.uk/projects/mepres/alevel/fpure_ch3.pdf (Accessed: 24 December 2024).
[6] Steane, A.M. (1998) 'Quantum Computing', Reports on Progress in Physics, 61(2), pp. 117-173. Available at: https://iopscience.iop.org/article/10.1088/0034-4885/61/2/002/meta (Accessed: 24 December 2024).

[5] Gelfond, Y. (n.d.) The RSA Algorithm. Available at: http://susanka.org/MathPhysics2/RSA_Algorithm_Yevgeny.pdf (Accessed: 24 December 2024).
[4] Jozsa, R. (1997) 'Quantum Algorithms and the Fourier Transform', Proceedings of the Royal Society A, 454(1969), pp. 323-337. doi: 10.1016/S1363-4127(97)81323-4.
[3] Chandra, S., Paira, S., Alam, S.S. and Sanyal, G. (2014) 'A Study and Analysis on Symmetric Cryptography', International Conference on Science, Engineering and Management Research, pp. 1-8. Available at: https://www.researchgate.net/profile/Sourabh-Chandra/publication/275517121_A_Study_and_Analysis_on_Symmetric_Cryptography/links/5558d99e08aeaaff3bf98ad7/A-Study-and-Analysis-on-Symmetric-Cryptography.pdf (Accessed: 24 December 2024).
[2] Lomonaco, S.J. (2000) 'Shor's Quantum Factoring Algorithm', arXiv preprint quant-ph/0005003. Available at: https://arxiv.org/pdf/quant-ph/0005003 (Accessed: 24 December 2024).
[1] Lavor, C., Manssur, L.R.U. and Portugal, R. (2008) 'Shor's Algorithm for Factoring Large Integers', arXiv preprint arXiv:0809.4416. Available at: https://arxiv.org/pdf/0809.4416 (Accessed: 24 December 2024).

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, 24 Desember 2025

Vincent Rionarlie

.