Cryptographic Analysis of JSON Web Token

Rava Maulana - 13521149[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13521149@std.stei.itb.ac.id

***Abstract*—JSON Web Token (JWT) is a popular method for token-based authentication. This method of authentication uses a token to transfer informations in a secure manner. A number of cryptographic techniques is used to encrypt information. Cryptography is used in JSON Web Token as a signing process to ensure the validity of the data. The common algorithms used in JWT signing process are SHA256, RSA, and ECDSA. The effectiveness and security measures are analysed further in this paper. Furhtermore, this paper also discuss the general principle of token-based authentication, particularly JSON Web Token, and its usage in the modern age.**

***Keywords*—Authentication, Cryptography, JSON Web Token, Number Theory**

I. Introduction

JSON Web Token (JWT) is a standard for producing and exchanging authentication and authorization tokens. A JWT is made up of claims, which are statements about an entity (such a user) and their activities or rights, and is JSON-based. A JWT's content is signed using a secret key or a public/private key combination, enabling the recipient to confirm the token's legitimacy.

JWTs can be used for a variety of purposes, such as enabling single sign-on (SSO) and displaying access and identity tokens. They are frequently utilized in contemporary web and mobile applications, and numerous third-party services and libraries support them.

JWT is used in the context of SSO to authenticate a user with a single set of credentials across numerous services or applications. By doing this, the user can log in once and access several services without having to do so for each one separately. In addition to improving user experience and lowering security threats like password fatigue, this can also make things more fluid and user-friendly. Access and identity tokens, which are used to authorize access to restricted resources and identify the user making the request, are also frequently represented using JWT. In this instance, the JWT comprises claims that describe the identity of the user and any rights or privileges they may have. When a user tries to access a protected resource, they must first present a JWT to the resource server. The server can then check the token's authenticity and grant or refuse access based on the token's claims.

JWT is simple to integrate into online and mobile applications because it is supported by numerous third-party services and libraries. It is also extensively utilized in industry, and popular authentication and authorization protocols like OAuth and OpenID Connect support it.

A JWT's signature ensures that the contents of the token cannot be changed without nullifying the signature, effectively acting as a type of encryption. A JWT can be signed using a variety of techniques, including symmetric algorithms (like HMAC) and asymmetric algorithms (such as RSA and ECDSA).

II. Theoretical Basis

A. Cryptography

Cryptography is the process of securing information using codes and ciphers. Different cryptography method are used throughout the history, predating back to ancient Egypt. In the past, cryptography has been used to protect important information particularly connected to wars. Cryptography plays an important role in World War II where it is used to secure communication between the alllies and axis. In the modern time, cryptography is used in various fields for securing information such as digital transaction and blockchain technology.

Modern cryptography is not complete without ciphers, which are used for a variety of purposes, including secure communication, data storage, and electronic commerce. The choice of a cipher depends on the particular requirements and security demands of the application. There are many distinct types of ciphers, each with unique strengths and drawbacks.

The process of converting plaintext to ciphertext is called encryption. Encryption is usually done by a mathematical function to map each letter in the alphabet into a different letter. Most encryption algorithm requires a key to secure information. A key defines the transformation that is done to each letter in plaintext. Some common encryption algorithm used today are RSA public-key algorithm and SHA256 encryption algorithm.

The process of converting ciphertext into plaintext is called decryption. The decryption process is the reverse of encryption. An encryption-decryption algorithm must make sure that the plaintext is the same before and after the encryption-decryption process. In general, there are two types of encryption algorithm, symmetric key algorithm and asymmetric key algorithm. In symmetric key encryption algorithm, the same key is used on the encryption and decryption process. An example of this algorithm is SHA256

encryption algorithm. The key must be kept in a secure location because everyone with the key will be able to decrypt the ciphertext. In asymmetric key encryption algorithm, different keys are used in the encryption and decryption process. The most common type of this algorithm is public key cryptography. The public key is made available to the public to encrypt the message, while the private key is stored in secure location to decrypt the message. An example of this algorithm is RSA encryption algorithm.

Number theory is used in cryptography to create algorithms that are based on mathematical properties of numbers. For instance, RSA and ECDSA, two popular public-key encryption algorithms, are built on the challenge of factoring big composite numbers into their prime factors. These methods rely on the fact that factorizing huge numbers is computationally impractical, hence the algorithm's security is founded on the fundamental ideas of number theory.

Other cryptographic techniques also make use of number theory, for example, to generate random numbers that are utilized to create cryptographic keys. In this scenario, techniques based on mathematical properties—such as the distribution of prime numbers or the patterns of digits in huge numbers—are used to generate the random numbers.

The simplest form of encryption algorithm is the caesar cipher. This cipher predates to the roman empire under the ruling of Julius Caesar. This algorithm shifts each letter in the alphabet by three letter. The plaintext is encrypted by the function

$$E(p) = (p+3) \bmod 26$$

The ciphertext is then decrypted by the function

$$D(c) = (c-3) \bmod 26$$

This cryptography method can be extended by varying the shift amount by a certain amount *k*. An example of algorithm using the similar principle as caesar cipher is the ROT13 algorithm in the linux operating system, shown in Fig. 1.

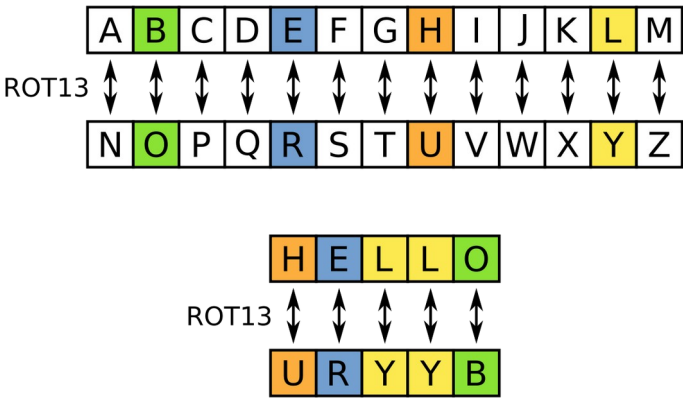

Figure 1: ROT13 encryption algorithm commony used in linux operating system

(Source: https://en.wikipedia.org/wiki/ROT13)

One of the widely used cryptography method is the RSA public-key cryptography. The keys used in this algorithm are a public key and a private key. The steps for generating the pair of keys are,

1. Choose two prime numbers *p* and *q*.
2. Calculate $n = pq$.
3. Calculate $m = (p-1)(m-1)$.
4. Choose a number *e* that is relatively prime to *m*.
5. Calculate *d* from the congruence $ed \equiv 1 \pmod m$

The number *e* is the public key and *d* is the private decryption key. The plaintext is encrypted by the function

$$c = p^e \bmod n$$

while the ciphertext is decrypted by the function

$$p = c^d \bmod n$$

This algorithm is commonly used in secure communication protocols, such as Transport Layer Security (TLS) and Secure Socket Layer (SSL) protocol. This algorithm is also commonly used in electronic transaction for securing card information.

B. Token Based Authentication

Token-based authentication is a method of safeguarding access to a computer or network by utilizing a unique, randomly generated token. The user often receives this token as a code or a link, which they must enter or click on in order to access the system. They are normally given access to the system for a predetermined amount of time after authenticating themselves using the token before it expires and they must request a new one. Since it can add an extra layer of protection and make it harder for unwanted users to access a system, token-based authentication is frequently employed as an alternative to conventional username- and password-based authentication.

The use of a token to give access to a system is a feature shared by both token-based authentication and session-based authentication. But there are some significant distinctions between the two approaches.

One significant distinction is that, in token-based authentication, the token is frequently held on the client side as opposed to the server in session-based authentication. This means that in order to provide a user access to the system using session-based authentication, the server must keep track of the user's session and keep a record of their token. With token-based authentication, on the other hand, the client can retain the token locally and use it to authenticate themselves without the server having to keep track of the session.

Another significant distinction is that whereas with token-based authentication, the token may have a set lifespan and may need to be refreshed or renewed after a specific amount of time, with session-based authentication, the token is normally valid for the duration of the user's session. This means that with token-based authentication, the user may need to authenticate themselves frequently in order to continue using the system, whereas with session-based authentication, the user may not need to authenticate themselves again until they log out or their session expires.

Overall, token-based authentication has a few advantages over session-based authentication, but the biggest one is that it can offer a higher level of security because the token can be more readily withdrawn or reissued if necessary. It also makes it simpler to establish single sign-on (SSO) solutions and gives more flexibility in terms of how long a user is allowed access to the system. An overview of session-based authentication and token-based authentication is shown in Fig. 2 and Fig. 3.

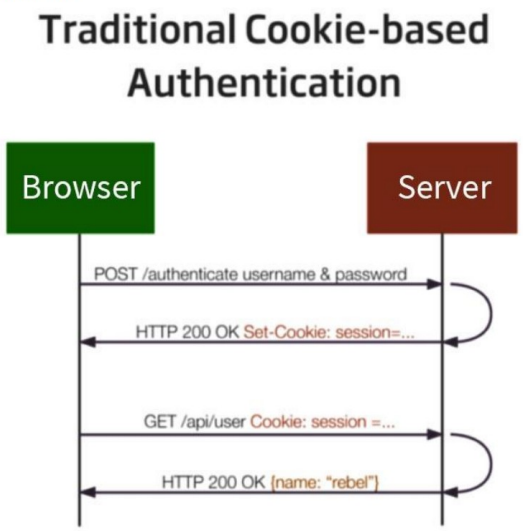

Figure 2: Session-based authentication

(Source: https://iopscience.iop.org/article /10.1088/1757-899X/550/1/012023/pdf)

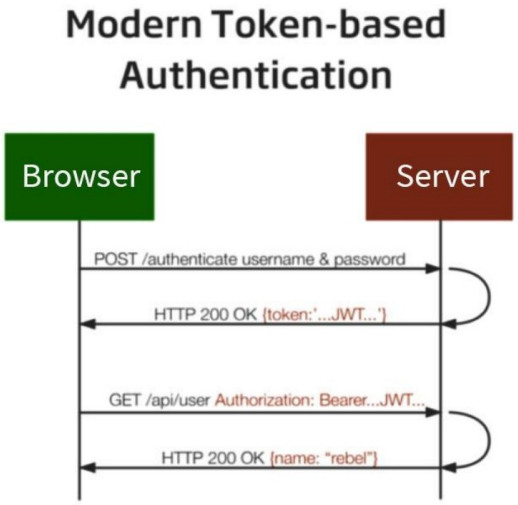

Figure 3: Token-based authentication

(Source: https://iopscience.iop.org/article /10.1088/1757-899X/550/1/012023/pdf)

Token-based authentication can offer a high level of security because it grants access to a system using a special, randomly generated token. A code or link containing this token is frequently delivered to the user, who must then input it or click on it to obtain access to the system. Unauthorized users would now have to obtain the token in order to authenticate themselves, which makes it more difficult for them to access the system.

Additionally, the usage of multi-factor authentication, in which a user must present various forms of identification to prove their identity, is frequently made possible by token-based authentication. The user might be needed to submit other information in addition to the token, such as a username and password, a fingerprint, or other biometric information. Unauthorized users may find it considerably harder as a result to access the system.

The token's ease of renewal or revocation is another benefit of token-based authentication. For instance, the token can be promptly invalidated if the user's device is lost or stolen, preventing anyone else from using it to access the system. By doing so, you may be able to add another layer of security and prevent illegal access.

Overall, token-based authentication can be regarded as having high levels of security because it offers numerous layers of security and permits the inclusion of more security measures as needed.

III. JSON Web Token

JSON web token uses Javascript Object Notation (JSON) data format to encrypt sensitive data. JSON web token consists of three parts, header, payload, and signature. Each part are concatenated into a string, which is the token itself. The header contains information such as the type of the token and the algorithm that is used to encrypt the signature. The payload contains sensitive information that may be used in an authentication method such as username and password. The signature is used to verify the validity of the token. Different encryption algorithm produces different signature for JSON web token.

The sensitive data contained in the payload are called claims. JSON web token specifies three different types of claims: registered, public, and private claims. The registered claim are reserved claims that specifies useful information about the token, such as expiration time (exp claim), issuer (iss claim), and audience (aud claim). The public claim is publicly available to use, but it must be registered in the IANA JSON Web Token Registry to avoid naming collision between claims. The private claim is a custom claim made by the application to store application-specific data in the token.

The steps for generating a JSON web token are,

1. Specify the signing algorithm and the type of the token in the header.
2. Base64url encode the header.
3. Specify the claims in the payload.
4. Base64url encode the payload.
5. Hash the encoded header and payload using the signing algorithm to produce the signature.
6. Concatenate the encoded header, payload, and signature into a single string separated by dot '.' character.

Fig. 4 and 5 shows an example of JSON web token header and payload. The payload is used to store username and password for user authentication. The resulting JSON web token signed by HS256 signing algorithm is showed in Fig. 6.

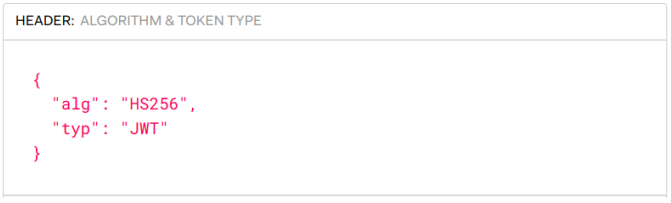

Figure 4: Example of a JWT header

(Source: Personal document)

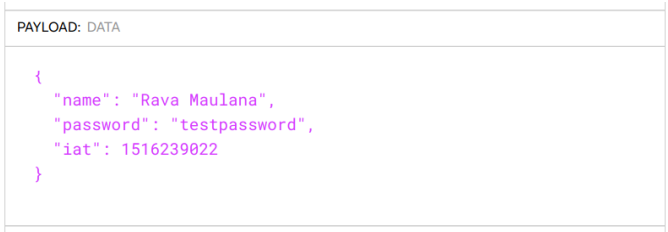

Figure 5: Example of a JWT payload

(Source: Personal document)

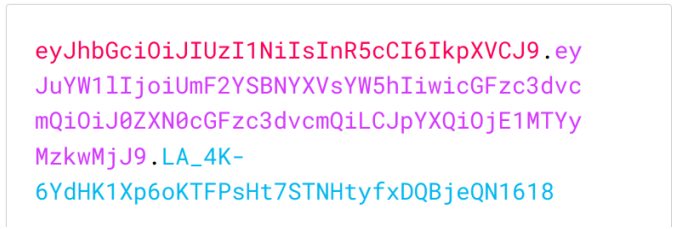

Figure 6: Example of an encrypted JWT

(Source: Personal document)

JSON web token is commonly used to authenticate user on an application. The transfer of the token from the server to the client is done by http request protocol. The server will issue an access token as a response for certain request such as login or signup request. The token is then stored on the client side storage such as localstorage for web-based application or local cache for native application. The access token can be used to replicate a user session by using exp claim to sets the expiry of the token. The authenticity of the token is signed by the server issuing the token.

Whenever the user requests for authorized resource on the server, the server will require the access token in the request. To check the integrity in the token, the server decodes the claims in the token, including the signature. The signature is then verified by using the secret key stored in the server and the signing algorithm used by the token. If the signature is valid, the server can trusts the claims in the JSON web token and authorize the request.

The security of JSON web token depends on how well the server stores the secret key. The secret key is commonly stored on the server's environment for maximum security. The information in the payload itself doesn't have a moderate security to allows the payload to contain sensitive information. Data in the payload can be obtained by decrypting the payload using base64url encryption. If a token is intercepted by third-party user, the user can view the contents of the payload without knowing the secret key stored in the server. This is the reason JSON web token isn't used to transfer sensitive information, rather it is used to transfer general information, such as email, in a secure manner.

The advantage JSON web token offers is the ability to securely transfers information between the server and the client. This transfer of information is device agnostic, in which the server can reuse the same codebase to send data to web application and mobile application. This way of information transfer is possible because the token relies only on signing of the token, which is done on the server, compared to session based authentication which have different protocol of storing information for each devices.

IV. Encryption Analysis

A. SHA256 Signature

A digital file or other input is frequently used to create a distinct, fixed-length string of text using the cryptographic hash function SHA256. It is one of several various hashing algorithms used for this purpose and is regarded as being very efficient and safe. The SHA256 algorithm creates a fixed-length text string known as the hash by applying a sequence of mathematical operations to an input (such as a digital file or message). The hash is specific to the input and will alter if even one character is changed. As a result, the hash may be used to check the accuracy of the input because it will create a new hash if the input is changed.

One of SHA256's main benefits is how challenging it is to undo the hashing operation and recover the original input from the hash. As a result, you may use the hash as a type of digital fingerprint to confirm the legitimacy of a file or message without knowing what's within.

The cryptographic hash function SHA256 is regarded as being exceptionally safe. It is practically impossible to reconstruct the original input from the hash because it is made to be hard to reverse. As a result, it is highly challenging for attackers to alter or fake data that has been hashed with SHA256. Furthermore, SHA256 is made to be collision-resistant, which makes it extremely improbable that two separate inputs will result in the same hash. This makes it challenging for attackers to produce a fake input with the same hash as a real input, making it harder for them to get around authentication mechanisms and tamper with data.

Although no cryptographic method is totally secure, SHA256 is generally thought of as being quite secure and is utilized in a variety of security-critical applications. It has so far shown to be resistant to attacks despite being frequently exposed to intensive security testing.

When a JWT is signed with SHA256, the algorithm creates a distinct hash from the contents of the JWT (such as the user's identity and any other pertinent information). The public key of the organization that signed the JWT is then included in the JWT along with this hash. By computing the hash and comparing it to the one present in the JWT, this enables the recipient of the JWT to confirm its legitimacy.

There are various advantages to using SHA256 in JWTs. First off, it contributes to maintaining the JWT's integrity by generating a different hash whenever the JWT's contents are altered. As a result, recipients of the JWT may be sure that it wasn't modified with while in transit. Second, the usage of SHA256 makes it challenging for attackers to counterfeit JWTs since they must be able to compute the proper hash for the JWT in order to make it seem genuine. This lessens the chance of illegal access to resources that are protected. All things considered, the usage of SHA256 in JWTs contributes to the security and integrity of the JWT and helps to guard against unauthorized access to resources that are under protection.

B. RSA Signature

Public-key encryption techniques like RSA are frequently utilized. It has the names of its three creators, Ron Rivest, Adi Shamir, and Leonard Adleman, who developed it in the 1970s while employed at MIT. In RSA, each message's sender and recipient has a public key and a private key. While the private key needs to be kept a secret, the public key can be shared with anybody. The sender uses the recipient's public key to encode the message before encrypting it. The message can then be decoded by the recipient using their private key. Because it is highly difficult for someone who does not have the private key to decipher the encrypted communication, this sort of encryption is regarded as being exceptionally secure.

RSA is typically regarded as being exceedingly secure. Banks, governments, and other organizations frequently utilize it to safeguard sensitive data. The difficulty of factorizing huge numbers is what gives RSA its security. The security of the RSA technique depends on the fact that it is computationally impossible to factorize a big composite number (such as the product of two prime numbers) into its prime factors. In RSA, the public and private keys are derived from two huge prime numbers.

The security of RSA (and any other encryption scheme) can, however, be compromised if the private key is lost, leaked, or otherwise made public. To lessen the chance of a security breach, it is crucial to maintain private keys safe and secure and to update and replace them on a regular basis.

One of the algorithms that can be applied to sign a JWT is RSA. In this instance, the issuer creates a public/private key combination and signs the JWT using the private key. Those that require the public key to confirm the JWT's legitimacy can use it. The recipient of a JWT uses the issuer's public key to decode the signature and certify that it agrees with the JWT's contents. Because it is thought to be exceptionally secure, RSA is a preferred option for JWT signatures. Additionally, many JWT libraries and other tools support it.

C. ECDSA Signature

One kind of digital signature method is called ECDSA, which stands for "Elliptic Curve Digital Signature Algorithm." A digital communication or document can be signed with it, and its legitimacy can be confirmed. ECDSA is based on the idea of elliptic curves, a category of curves that may be described by a mathematical equation. To create a public/private key pair in ECDSA, a set of mathematical processes and an elliptic curve are used. The public key is used to validate the signature while the private key is used to sign messages.

The sender must first compute a cryptographic hash of the message using a hash function, such as SHA-256, before signing it with ECDSA. The sender then applies a series of mathematical operations to the hash value and the private key to create a digital signature for the message using the private key. The message is then sent with the signature attached.

The recipient applies the identical set of mathematical operations to both the received signature and the message's hash using the public key to validate the signature. The recipient can be sure that the message hasn't been tampered with and that it comes from the expected sender if the outcome of the operations matches the expected value, in which case the signature is thought to be legitimate.

The mathematical features of elliptic curves, which make it challenging to calculate the private key from the public key, provide the foundation for ECDSA's security. The security of the ECDSA algorithm depends on the fact that it is computationally impossible to deduce the private key from the public key. The private key in this approach is derived from a randomly generated point on an elliptic curve.

Due to its resistance to attacks including brute-force attacks and mathematical attacks, ECDSA is regarded as a secure signing method. It is frequently employed in protocols and software programs that call for digital signatures, such as SSL/TLS certificates and monetary instruments.

The security of ECDSA (and any other encryption technique) can, however, be compromised if the private key is lost, leaked, or otherwise made public. To lessen the chance of a security breach, it is crucial to maintain private keys safe and secure and to update and replace them on a regular basis.

One of the techniques that can be applied to sign a JWT is ECDSA. The issuer in this instance creates a public/private key pair using an elliptic curve, and then signs the JWT using the private key. Those that require the public key to confirm the JWT's legitimacy can use it. The recipient does a series of mathematical operations to the received signature and the hash of the received JWT in order to validate the signature of a JWT using the issuer's public key. The signature is deemed to be valid if the outcome of the operations matches the predicted value.

V. Conclusion

JSON web token (JWT) is a popular authentication method for a modern application. JWT uses the more modern concept of token-based authentication, as opposed to the traditional session-based authentication. Information can be transferred securely as JWT’s claims. The efficiency of JSON web token came from the fact that the server only needs to verify the signature of the token without storing any information in its database. JWT’s advantages made it a popular choice for single sign-on (SSO), transport layer security (TLS), and online transaction, such as e-commerce and blockchain.

The security of JSON web token came from the robustness of its signing algorithm. The most common algorithm for signing a JWT are SHA256, RSA, and ECDSA algorithms. These algorithms use a key or set of keys as a part of its encryption process. A symmetric encryption algorithm uses only one key, whereas an asymmetric enryption algorithm uses a pair of public and private keys to encrypt its content. The signing algorithm used in JWT uses mathematical concepts such as number theory, prime number theory, and elliptic curves. These applied usage of pure mathematics allows for a secure way to transfer data across computers in a network.

VI. Acknowledgment

I would like to show my appreciation to Ms. Fariska

Zakhralativa Ruzkanda, S.T. M.T. for teaching me discrete mathematics for this semester, which exposes me to a whole lot of new concepts and research topics in the field of computer science. This assignment in particular has encouraged me to research an important topic in modern cryptography, which extends my knowledge in number theory and cryptography. I am grateful to be able to finish this paper as a way to improve my writing and research skills.

REFERENCES

[1] M. Jones, J. Bradley, N. Sakimura, "JSON Web Token (JWT)", in Internet Engineering Task Force (IETF), 2015.
[2] N. Rasyada, "SHA-512 Algorithm on JSON Web Token for Restful Web Service-Based Authentication", in Journal of Applied Data Sciences Vol. 3, No. 1, 2022, pp. 33-43.
[3] A. Rahmatulloh, R. Gunawan, F. M. S. Nursuwars, "Performance Comparison of Signed Algorithms on JSON Web Token", in IOP Conference Series: Materials Science and Engineering, 2019.
[4] M. F. Mushtaq, S. Jameel, A. H. Disina, Z. A. Pindar, N. S. A, Shakir, M. M. Deris, "A Survey on The Cryptographic Encryption Algorithms", in International Journal of Advanced Computer Science and Applications, Vol. 8, No. 11, 2017.
[5] R. Munir, "Teori Bilangan bagian 1", in Informatika STEI ITB, https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2022-2023/matdis22-23.htm, 2022.
[6] R. Munir, "Teori Bilangan bagian 2", in Informatika STEI ITB, https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2022-2023/matdis22-23.htm, 2022.
[7] R. Munir, "Teori Bilangan bagian 3", in Informatika STEI ITB, https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2022-2023/matdis22-23.htm, 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 2020

Rava Maulana - 13521149