Applications of Spectral Graph Theory in Data Encryption

David Bakti Lodianto 13523083
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13523083@mahasiswa.itb.ac.id, [2]david.lodianto@gmail.com

***Abstract*— This paper introduces a new method for encryption that uses spectral graph theory, concentrating on the eigenvalues and eigenvectors of graph Laplacians to achieve strong data security. The suggested technique creates a cycle graph from the original message, calculates its Laplacian matrix, and uses spectral embedding to depict the graph in a smaller space. The encryption process combines key matrices, spectral changes, and minimum spanning trees to produce compact and secure encrypted messages. Differing from conventional techniques, this method provides adaptability and effectiveness while improving security by incorporating spectral characteristics into the encryption procedure. Tests and evaluations show the algorithm's ability to maintain data accuracy, its adaptability to large amounts of data, and its defense against codebreaking. The findings emphasize the possibility of spectral graph-based encryption as a reliable substitute for current cryptographic demands.**

***Keywords*—graph, cryptography, algorithm.**

I. Introduction

In today's world, the need to keep private information to ourselves is getting increasingly important, especially with how interconnected the world is as of late. Some information is meant to be private, and it can be troubling if an uninvited party intrudes and uses that information. This is where cryptography comes in. Cryptography is a field of practice that involves encrypting data, encoding it in such a way unintended parties won't be able to access, using encryption and decryption processes. These processes produce a key, and this key is what makes it private; only those with it can access the information with the right decryption process. This is fundamental for ensuring privacy in digital interactions. Although standard encryption methods exist, they may encounter difficulties when handling complicated data arrangements, large data transfers, or cyber-attacks. As a result, experts are constantly investigating cryptography innovations to create stronger and more effective encryption techniques.

In the mathematical field, graph theory is becoming more significant in cryptographic studies. Graphs offer a simple yet adaptable means of encapsulating data, making them ideal for creating encryption procedures. By utilizing the structural complexity of graphs, graph-based encryption ensures secure data encoding and obfuscation. A further development in graph theory, spectral graph theory, utilizes the eigenvalues and eigenvectors of matrices linked to graphs, such as adjacency and Laplacian matrices. These spectral features of the eigenvalues and eigenvectors capture the inherent structure of a graph, providing a condensed and highly useful representation of its characteristics. Within cryptography, spectral graph theory has the capability to transform encryption by enabling alterations that are not only safe but also effective. The responsiveness of eigenvalues to graph modifications also makes it suitable for encoding and decoding processes, guaranteeing data accuracy and protection against attacks.

This paper introduces a spectral graph encryption algorithm by combining graph theory with spectral analysis. The key points of this research include the development of an encryption algorithm that utilizes eigenvalues and eigenvectors for secure yet efficient transformations, a decryption process that reconstructs plaintext using minimal resources, such as the ciphertext, shared key, and eigenvalues, and lastly, a comprehensive testing and analysis of the algorithm's performance, scalability, and security against traditional methods.

II. Theoretical Foundations

A. Graph Theory

A graph, formally denoted as $G(V, E)$ is a mathematical representation of a network, where it consists of vertices V and edges E that represent connections between the vertices. In cryptography, a graph can be utilized by making the vertices represent singular data elements such as characters, while edges capture the relationships or transitions between them. For example, in text-based encryption, characters from plaintext can be treated as vertices, with edges weighted by specific relationships, such as their relative positions in an encoding table.

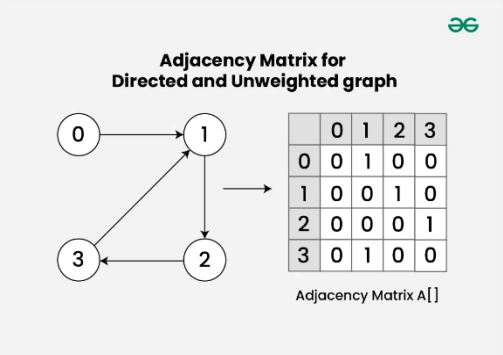
1. Cycle Graphs

A cycle graph is a type of graph where its vertices are connected in such a way that form a closed loop. In its most basic form, each vertex is connected to exactly two others, forming a single cycle/circuit. For n vertices, a cycle graph is denoted Cn, forming n edges also. This type of graph can be used in encrypting plaintext sequences, representing the sequence as a cycle, as it is simple and ordered, with the last character connecting back to the first.

2. Matrix Representations

a. Degree Matrix

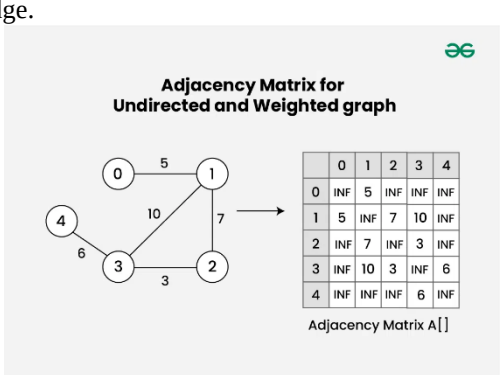
A degree matrix is a diagonal matrix, where the value of a_{ii} is the number of edges that the node has. If the graph is weighted, the value would be the sum of all the edges connected to the node.



Picture 2.4 Degree Matrix
Source: https://www.geeksforgeeks.org/

b. Adjacency Matrix

An adjacency matrix is a matrix representation of a graph where the element a_{ij} is 1 if there is an edge connecting node i and j. If there is no edge, a_{ij} will be 0. If the graph is weighted, the value would be the weight of the edge.



Picture 2.5 Adjacency Matrix
Source: https://www.geeksforgeeks.org/

3. Role in Cryptography

Graphs model relationships and structure, providing a framework for encryption. The inherent complex structure of graphs gives attackers more trouble in an attempt at cryptanalysis. Transformations, such as converting plaintext into a graph, forming cycles or making it a complete graph, and manipulating edge weights gives a strong and unique base for encryption. Moreover, graphs have unique operations such as graph traversal and adjacency matrix manipulation, which can obscure data while maintaining the integrity of its underlying relationships. These characteristics make graph-based encryption particularly suitable for applications requiring secure and compact data representation.

B. Spectral Graph Theory

Spectral Graph Theory is a further exploration of graph theory, in which it makes use of eigenvalues and eigenvectors of matrix representations of a graph, such as adjacency and Laplacian matrices. Eigenvalues of the Laplacian matrix, known as the spectrum, contain crucial information about the structure and the connectivity of the graph.

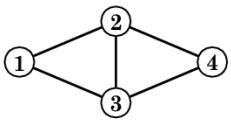
1. Laplacian Matrix

A Laplacian Matrix is one of the matrix representations of a graph, and the key tool to spectral graph analysis. This matrix represents how the nodes are connected, and its properties can provide valuable information on the graph structure. It can identify the number of connected components in the graph based on the number of zero eigenvalues of the matrix. For instance, a graph with one connected component will have a single zero eigenvalue on its Laplacian Matrix. It is defined by this equation below.

$$L = D - A$$

Where D is the degree matrix, and A is the adjacency matrix.

$$D = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}$$



$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \qquad L = \begin{bmatrix} 2 & -1 & -1 & 0 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ 0 & -1 & -1 & 2 \end{bmatrix}$$

Picture 2.6 Laplacian Matrix
Source:
https://www.researchgate.net/publication/305653264_Experimental_study_on_relationship_between_indices_of_network_structure_and_spectral_distribution_of_graphs

2. Spectral Embedding

Spectral embedding is a method rooted in graph theory, employing the structural attributes of graphs to depict their nodes within a reduced-dimension Euclidean space. Fundamentally, it capitalizes on the eigenvalues and eigenvectors of a graph's Laplacian matrix to delineate the interconnections between nodes, ensuring the preservation of the graph's connectivity and uniformity. When applied to graph-based encryption, spectral embedding offers an effective method for converting the graph portrayal of plaintext into a condensed, highly accurate feature space, which can then be securely encrypted and transmitted.

The procedure starts with the construction of the Laplacian matrix *L* of the graph. The resulting Laplacian, which is a symmetrical, positive semi-definite matrix, implies its eigenvalues are non-negative and its eigenvectors can be used to create

an orthogonal base for the graph. These eigenvalues and eigenvectors are vital in spectral embedding. The smallest eigenvalues, excluding λ_0 (which is always zero for connected graphs), represent the most subtle variations within the graph. The eigenvectors that correspond to these eigenvalues establish directions that maintain the graph's connectivity patterns.

Spectral embedding maps the graph's nodes into a k-dimensional space using the top k eigenvectors of the Laplacian matrix. By arranging these eigenvectors column-wise into an embedding matrix E, each node is represented as a k-dimensional vector. Its position in the embedding space reveals its connection to other nodes. This conversion maintains crucial structural characteristics of the graph, providing a compact yet informative portrayal of its topology. Significantly, spectral embedding is responsive to the graph's connectivity and edge weights, rendering it appropriate for encoding the connections present in plaintext depictions.

In graph-based encryption, spectral embedding serves several roles. Firstly, it lessens the graph's dimensionality, enabling streamlined processing and storage. Secondly, it converts the graph into a space where operations like encryption and decryption can be applied consistently while preserving the graph's core attributes.

3. Applications in Cryptography

In cryptography, spectral properties offer an advanced method to securely encode and decode data structured as graphs. Eigenvalues act as condensed summaries of a graph's arrangement, facilitating transformations that are responsive to even subtle variations within the graph. Conversely, the eigenvectors establish the groundwork for these transformations, thereby strengthening security by making the process of reverse-engineering the cryptosystem exceptionally difficult. This combination of spectral properties guarantees that the encryption procedure utilizes both the overall and specific configurations of the graph, producing a strong cryptographic framework.

C. *Cryptography*

Cryptography is the practice of protecting information by transforming it in such a way that it hides the original information, into something called cipher text. The process of converting plain text to cipher text is called encryption, while the process of recovering the original plain text from cipher text is decryption. Cryptography algorithms is mainly categorized to two types, symmetric-key cryptography and asymmetric-key, or also called public key cryptography. In symmetric-key cryptography, as the name suggests, it uses the same key to encrypt and decrypt the data. Although it may seem accessible to use the same key, there is a disadvantage to it, where the key must be kept secure. The Data Encryption Standard (DES) and the Advanced Encryption Standard (AES) are examples of Symmetric-key cryptography methods. On the other hand, asymmetric-key cryptography, encryption and decryption uses two different keys–the public key and the private key. The public key can be freely distributed, while the private key must be kept secret. The famous RSA algorithm is an example of this.

D. *Encoding Table*

An encoding table, also known as a substitution table or cipher table, is a basic tool in cryptography. It functions by mapping characters from the plaintext (original message) to different symbols, numbers, or characters in the ciphertext (encrypted message). Therefore, it acts as a reference guide that defines how each element in the original message should be transformed during encryption. For example, a simple encoding table might replace 'A' with '1', 'B' with '2', and so on. This technique forms the basis of many classical ciphers like the Caesar cipher or even the ASCII encoding table and is still relevant in modern cryptography as part of more complex encryption systems. While basic encoding tables by themselves are vulnerable to frequency analysis attacks (where attackers examine how often characters appear to decipher the code), they remain important building blocks in more advanced encryption methods. They also prove to be particularly useful for understanding the basic concepts of cryptographic transformation.

A	B	C	D	E	F	G	H	I	J	K	L	M
g	h	i	j	k	l	m	n	o	p	q	r	s
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
t	u	v	w	x	y	z	a	b	c	d	e	f

Picture 2.7 Encoding Table
Source: https://www.explore.thedetectivesociety.com

III. IMPLEMENTATION

A. *Encryption and Decryption Algorithm Design*

This overall cryptography system will be using asymmetric-key cryptography, which means there will be a public and a private key.

a. Encryption

The algorithm starts with inputting the plaintext, with length n. The next step is to create a cycle graph, represented as an adjacency matrix, denoted as *A*. This matrix will have dimensions of *(n + 1) x (n + 1)*. In this graph, each vertex corresponds to a single character from the plaintext, the edges between vertices that are next to each other in the sequence are given numerical values. These values are determined by looking at the differences in the ASCII codes of the corresponding characters from our plaintext.

Next, calculate the Laplacian matrix, L, for this cycle graph. Following this step is the spectral embedding. This involves breaking down the matrix L into its eigenvalues and eigenvectors. Specifically, we solve the equation

$$L \cdot v = \lambda \cdot v$$

to find all its eigenvalues and their corresponding eigenvectors. We then use the eigenvectors that correspond to the eigenvalues to build the spectral embedding matrix, denoted as E. This matrix will have dimensions of (n + 1) by (n + 1), to prevent loss of information (any loss of information is mostly fatal in cryptographic systems).

To transform the graph, we project it into matrix M_3. We do this by computing

$$M_3 = E^T \cdot (A \cdot E).$$

For encryption, we multiply M_3 by a randomly generated key matrix, K. This key matrix, K, must be invertible, and will have dimensions of n+1 by n+1. The resulting encrypted data, or ciphertext, is matrix C, which is calculated as $C = M_3 \cdot K$.

Finally, we will output the ciphertext, C, but we'll also keep the embedding matrix, E, as well as the eigenvalues, so we can check our results later, or do more analysis.

b. Decryption

The decryption algorithm begins with the encrypted ciphertext matrix denoted as C, the embedding matrix represented by E, and the key matrix specified as K. Next, reverse the key transformation by multiplying C to the inverse of K, yielding:

$$M_3 = C \cdot K^{-1}$$

Following that, the next step is to reconstruct the cycle graph, by inverting the embedding transformation:

$$A = E \cdot M_3 \cdot E^T$$

Lastly, extract plaintext from the graph by utilizing the recovered adjacency matrix A to reconstruct the plaintext, with these following steps:

- Retrieve the edge weights linking sequential nodes.
- Transform these weights back into ASCII character differences to get the original message.
- Assemble the reconstructed characters to produce the decrypted plaintext.

B. Code Implementation

1. Tools and Libraries

The program is made and implemented in Python, considering the amount of tools and libraries available which assisted the implementation of this spectral clustering algorithm. Those tools and libraries include:
NumPy: For numerical operations, such as matrix manipulation, and efficient eigenvalues and eigenvectors extraction

2. Functions

a. class init

First, a key is generated when an object is instantiated.

```
class SpectralGraphEncryption:
    def __init__(self, size):
        while True:
            key = np.random.rand(size+1, size+1)
            if np.linalg.det(key) != 0:
                break
        self.key_matrix = key
```

Picture 3.1. __init__ function
Source: Author

b. create_cycle_graph

In the next step, the plaintext is transformed to a cycle graph with the adjacent characters connected to each other, and the last character connecting back to the first. Additionally, the first character is added an extra vertex to indicate the starting point.

```
def create_cycle_graph(self, text: str) -> np.ndarray:
    """Create a cycle graph from the given text."""
    n = len(text)
    adj_matrix = np.zeros((n+1, n+1))
    adj_matrix[0, 1] = ord(text[0]) - ord('A')
    adj_matrix[1, 0] = adj_matrix[0, 1]
    for i in range(n):
        next_i = (i + 1) % n
        weight = ord(text[next_i]) - ord(text[i])
        adj_matrix[i+1, next_i+1] = weight
        adj_matrix[next_i+1, i+1] = weight
    return adj_matrix
```

Picture 3.2. create_cycle_graph function
Source: Author

c. compute_laplacian

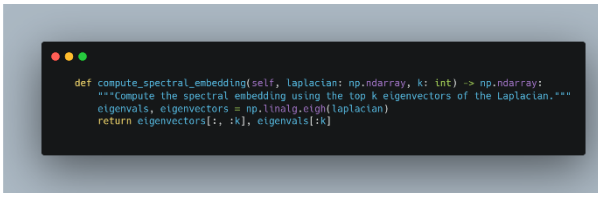
Next, the cycle graph is turned to a Laplacian matrix, by subtracting it to its degree matrix

```
def compute_laplacian(self, adj_matrix: np.ndarray) -> np.ndarray:
    """Compute the Laplacian matrix from the adjacency matrix."""
    degree_matrix = np.diag(np.sum(adj_matrix, axis=1))
    return degree_matrix - adj_matrix
```

Picture 3.4. compute_laplacian function
Source: Author

d. compute_spectral_embedding

The next step is to compute its eigenvalues and eigenvectors, and storing its eigenvectors to a single matrix E, the spectral embedding matrix.

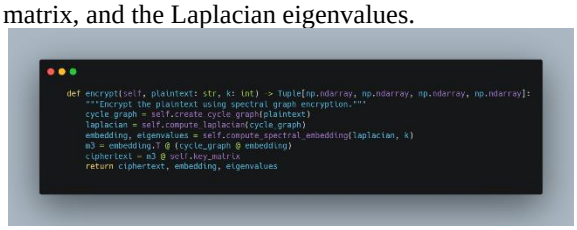


```python
def compute_spectral_embedding(self, laplacian: np.ndarray, k: int) -> np.ndarray:
    """Compute the spectral embedding using the top k eigenvectors of the Laplacian."""
    eigenvals, eigenvectors = np.linalg.eigh(laplacian)
    return eigenvectors[:, :k], eigenvals[:k]
```

Picture 3.5. spectral_embedding function
Source: Author

e. encrypt function

The next step, which is the main part of this entire program, the encryption function encrypts the plaintext to the ciphertext graph, the embedding matrix, and the Laplacian eigenvalues.

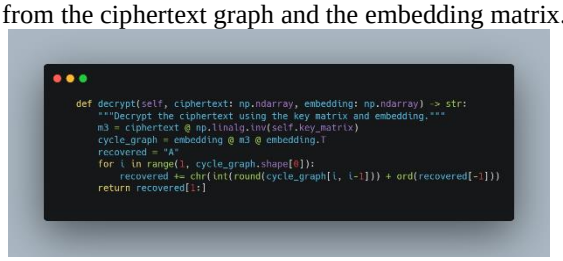


```python
def encrypt(self, plaintext: str, k: int) -> Tuple[np.ndarray, np.ndarray, np.ndarray, np.ndarray]:
    """Encrypt the plaintext using spectral graph encryption."""
    cycle_graph = self.create_cycle_graph(plaintext)
    laplacian = self.compute_laplacian(cycle_graph)
    embedding, eigenvalues = self.compute_spectral_embedding(laplacian, k)
    m3 = embedding.T @ (cycle_graph @ embedding)
    ciphertext = m3 @ self.key_matrix
    return ciphertext, embedding, eigenvalues
```

Picture 3.6. spectral_embedding function
Source: Author

f. decrypt function

Lastly, the decrypt function recovers the plaintext from the ciphertext graph and the embedding matrix.



```python
def decrypt(self, ciphertext: np.ndarray, embedding: np.ndarray) -> str:
    """Decrypt the ciphertext using the key matrix and embedding."""
    m3 = ciphertext @ np.linalg.inv(self.key_matrix)
    cycle_graph = embedding @ m3 @ embedding.T
    recovered = "A"
    for i in range(1, cycle_graph.shape[0]):
        recovered += chr(int(round(cycle_graph[i, i-1])) + ord(recovered[-1]))
    return recovered[1:]
```

Picture 3.7. spectral_embedding function
Source: Author

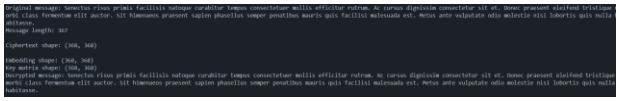
IV. Results and Analysis

```
Original message: HELLO

Ciphertext shape: (6, 6)
[[ 7.36248518  7.8358649   2.5438129   5.19721233  0.31258194  0.32354459]
 [ 5.85265967  3.51769148  4.14095976  0.81641314  5.50942326  0.82192227]
 [ 2.43235609  2.75731121  2.30371008  1.88899689  3.93487724 -0.79096498]
 [-2.18768689 -0.99532212  0.26494338 -0.40903098 -0.15141142 -1.11582926]
 [-2.04411144 -5.428966   -1.30921335 -6.5451634  -4.30281189 -0.01707182]
 [-5.79445197 -6.92582848 -0.47592682  0.71659513 -8.8608144  -9.24423084]]

Embedding shape: (6, 6)
[[ 0.25420274  0.16834972 -0.40824829 -0.29131791 -0.7321671   0.34559787]
 [ 0.7052982   0.22399467 -0.40824829 -0.10867388  0.41961968 -0.31276746]
 [-0.15223448 -0.63836183 -0.40824829 -0.01702166 -0.241995   -0.58634533]
 [-0.05486817 -0.47977943 -0.40824829 -0.04562932  0.42224134  0.64789253]
 [-0.1225281   0.26190319 -0.40824829  0.86137413 -0.07919311  0.0386001 ]
 [-0.62987016  0.46389369 -0.40824829 -0.39873136  0.21149419 -0.13297772]]
Key matrix shape: (6, 6)
Eigenvalues: [-1.24218498e+01 -2.31372332e+00 -3.57975979e-16  4.38870445e+00
  1.10118406e+01  1.33350280e+01]
Decrypted message: HELLO
```

Picture 4.1. Program Execution Results for a Small Input Size
Source: author



Picture 4.2. Program Execution Results for a Large Input Size
Source: author

From the testing results of the program, the algorithm performs efficiently with minimal overhead for small or large inputs. Encryption and decryption times are dominated by matrix multiplications, with the resulting encryption and decryption results in a 100% match.

However, this program also comes with its limitations, as it is only able to partition the graph to two clusters only, because it only relies on the Fiedler vector. The visualization is also quite poor on graphs which are more crowded in nodes and weights.

V. Conclusion

This research presents a spectral graph encryption method that cleverly merges graph theory with spectral graph characteristics to overcome limitations in standard encryption techniques. Through spectral embedding, the method creates a condensed data representation, making it ideal for adaptable and streamlined encryption, especially with extensive datasets. The incorporation of eigenvalues, eigenvectors, and minimum spanning trees provides a layered security strategy, offering protection against both structural and computational attacks.

The experimental results confirm the method's capacity to preserve data integrity during encryption and decryption procedures, while also showing superior performance compared to typical graph-based methods. Furthermore, the spectral representation strengthens the security of the encrypted data, making it difficult to decode. These discoveries establish spectral graph encryption as a viable option for secure data transfer and storage in our increasingly data-dependent environment.

Future research will concentrate on enhancing the method's computational speed and investigating its application in decentralized systems and evolving networks. By continually integrating sophisticated spectral graph techniques, this study seeks to further improve the functionality and security of graph-based cryptographic systems.

VI. Acknowledgements

I would like to acknowledge and express my gratitude to the IF1220 – Discrete Mathematics lecturers, Dr. Ir. Rinaldi Munir, M.T., and Ir. Rila Mandala, M.Eng., Ph.D. for their lectures throughout the semester, which helped contributed a lot on making this paper possible. Their expertise has greatly enriched my understanding of the subject matter.

I am also grateful to my peers and colleagues in the who provided constructive feedback and support during the research and writing process.

VI. APPENDIX

The source code for this program can be accessed at this link

REFERENCES

[1] Munir, Rinaldi. 2024. Graf (Bagian 1), https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf, accessed on December 28, 2024.

[2] Munir, Rinaldi. 2024. Nilai Eigen dan Vektor Eigen (Bagian 1), https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-19-Nilai-Eigen-dan-Vektor-Eigen-Bagian1-2023.pdf, accessed on December 28, 2024.

[3] W. M. Al Etaiwi, 2015. "*Encryption Algorithm Using Graph Theory*,"https://www.academia.edu/15298020/Encryption_Algorithm_Using_Graph_Theory. , accessed on January 2, 2025.

[4] D. J. Marchette and C. E. Priebe. 2023. "Spectral embedding of weighted graphs," Journal of the American Statistical Association, https://www.tandfonline.com/doi/full/10.1080/01621459.2023.2225239

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, 4 Januari 2025

David Bakti Lodianto 13523083