Journal of Computer Science

Original Research Paper

Application of Unimodular Hill Cipher and RSA Methods to Text Encryption Algorithms Using Python

[1]Samsul Arifin, [2]Dwi Wijonarko, [3]Suwarno and [4]Edwin Kristianto Sijabat

[1]Department of Data Science, Faculty of Engineering and Design, Institut Teknologi Sains Bandung; Bekasi, West Java, Indonesia
[2]Department of Information Technology, Faculty of Computer Science, University of Jember, Jember, East Java, Indonesia
[3]Department of Primary Teacher Education, Faculty of Humanities, Bina Nusantara University, Jakarta, Indonesia
[4]Department of Pulp and Paper Processing Technology, Faculty of Vocational, Institut Teknologi Sains Bandung; Bekasi, West Java, Indonesia

Article history
Received: 14-06-2023
Revised: 10-10-2023
Accepted: 13-01-2024

Corresponding Author:
Samsul Arifin
Department of Data Science, Faculty of Engineering and Design, Institut Teknologi Sains Bandung; Bekasi, West Java, Indonesia
Email: samsul.arifin212@gmail.com

Abstract: Text encryption is one of the techniques used to maintain the confidentiality of information in digital communications. In this study, we propose to apply a combination of the Unimodular Hill Cipher and RSA methods to a text encryption algorithm using the Python programming language. The Unimodular Hill Cipher method uses an unimodular matrix to replace text characters with encrypted characters, while RSA (Rivest-Shamir-Adleman) is a public key encryption algorithm that relies on modulo arithmetic properties. The purpose of this research is to combine the strengths of the two methods and produce a more secure text encryption system. Unimodular Hill Cipher provides the advantage of randomizing text characters by using matrix modulo operations, while RSA provides a high level of security through the use of public and private key pairs. In this study, we explain in detail the basic theory and algorithms of the Unimodular Hill Cipher and RSA. We also describe the implementation steps of both methods in the Python programming language. The text data used in this study went through a preprocessing stage before being encrypted. We also analyze the results of the encryption using several statistical methods to measure how close the relationship between the original text and the result of the encryption is. In a comparative analysis with the previous paper, in this study, the use of the Unimodular Hill Cipher and RSA methods in the context of Python provides additional insight into the performance and level of security of both. The experimental results show that the combination of the Unimodular Hill Cipher and RSA methods can produce a higher level of security in text encryption. It is hoped that this research can contribute to the development of a more effective and secure text encryption algorithm.

Keywords: Unimodular Hill Cipher, RSA, Text Encryption, Python

Introduction

In today's digital era, data and information security is very important. To protect the confidentiality and integrity of data, encryption methods play a crucial role. In this study, we will discuss the application of the Unimodular Hill Cipher and RSA methods to text encryption algorithms using the Python programming language. Unimodular Hill Cipher and RSA are two encryption methods that have proven effective in protecting sensitive data. Unimodular Hill Cipher uses a matrix as an encryption key, with the condition that the matrix must be unimodular to produce secure encryption. This method has the advantage of producing strong encryption and has a high level of decryption difficulty. Meanwhile, RSA is a public key encryption method that uses mathematical operations on large prime numbers. RSA has the advantage of providing high security, especially in the exchange of encryption keys. The main objective of this study is to apply the Unimodular Hill Cipher and RSA methods to text encryption algorithms using Python. We will experiment with using short text and long text as sample data for the encryption process. In addition, we will also perform statistical analysis and execution time measurements to test the encryption strength of these two methods. It is hoped that this research

can provide a better understanding of the application of Unimodular Hill Cipher and RSA in text encryption using Python (Zhang, 2020; Arifin *et al*., 2021a).

In this study, we have several main objectives and research contributions that we would like to share. First, our goal is to provide a complete practical guide on how to implement two powerful encryption algorithms, namely Unimodular Hill Cipher and RSA, using the Python programming language. We want to provide a useful resource for information security practitioners and software developers interested in securing text data. Second, we contribute by providing an in-depth understanding of the performance comparison between Unimodular Hill Cipher and RSA in the context of text encryption. It includes analysis of encryption speed with various types of text data, which can help algorithm selectors in selecting the most appropriate one for their needs. Thus, our main contribution is to provide practical guidance and a deep understanding of the application of two powerful encryption algorithms in security software development. We hope that this study will be a useful resource for the information security community and software developers focused on text encryption. In developing applications that use encryption methods, speed and efficiency are important. Therefore, we will optimize the performance of the algorithm using the Python programming language. Python was chosen because of its popularity and also because it has advantages in terms of speed and ease of use. We will use existing libraries, such as NumPy and Cryptography, to ensure an efficient and accurate implementation. It is hoped that the results of this research can contribute to the development of a stronger and more efficient text encryption method. In addition, implementation in the Python programming language is also expected to make it easier for users to implement these two methods in the applications they develop. Through this research, it is hoped that data and information security can be guaranteed, as well as provide significant benefits in maintaining data confidentiality and integrity in this increasingly complex digital era (Santoso, 2021; Basavaiah *et al*., 2021).

Hill Cipher is one of the classic cryptographic methods that use matrices to encrypt and decrypt text. Previous research has implemented Hill Cipher in text security and provided a theoretical basis for this algorithm. RSA (Rivest-Shamir-Adleman) is one of the most popular and powerful asymmetric encryption methods. Previous research has demonstrated the effectiveness of RSA in protecting text security using public and private keys. Several studies have proposed combining the Hill Cipher and RSA methods to increase the level of security in text encryption. This approach takes advantage of the advantages of each method to produce a stronger encryption scheme (Jayanthi and Singh, 2019; Agustini and Kurniawan, 2019). An illustration of the concept of asymmetric encryption can be seen in the following Fig. 1.

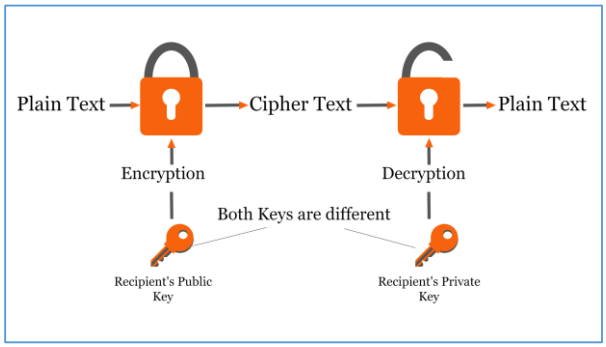

Fig. 1: An illustration of the RSA (asymmetric encryption) (Gibson, 2020)

Python is a popular programming language and is often used in implementing cryptographic algorithms. Several studies have used Python to implement text encryption methods, including Hill Cipher and RSA. Unimodular Hill Cipher is a variation of Hill Cipher that uses an unimodular matrix in the encryption and decryption process. Previous studies have shown that the Unimodular Hill Cipher has advantages in terms of key complexity and security. RSA has advantages in high security and scalability. Previous research has proven the effectiveness of RSA in protecting sensitive text and digital data. Although this article focuses on text encryption, several studies have also adopted these methods for video security. Implementation of the video involves processing frames and audio, as well as encryption techniques suitable for video data. Several related studies have been conducted in the context of the application of text encryption methods using Python. This study provides a deeper understanding of the advantages and disadvantages of each method, as well as provides insights into the development of more secure and efficient encryption solutions (Oliphant, 2007; Fadlan and Amaliah, 2020).

After conducting a literature review, several research gaps can be identified. Although many studies have focused on the implementation of encryption methods such as Unimodular Hill Cipher and RSA, there has been no research that specifically compares the performance and level of security of the two in the context of using Python to encrypt text algorithms. In addition, most existing research tends to focus on data security aspects, without considering the speed factor in encryption. Therefore, there is a need for comprehensive research and comparison between Unimodular Hill Cipher and RSA in terms of security and encryption speed, especially in implementation using the Python programming language. This kind of research can provide deeper insight into selecting the right encryption method based on the needs of a particular application. We would like to emphasize several significant novelties in this study. First, we present

a practical implementation of these two encryption algorithms using the Python programming language, which allows direct application in security software development. The practical implementation of these two encryption algorithms uses the Python programming language which allows direct application in security software development. In the context of this study, this effort is more implementative and practical rather than presenting an innovative contribution to new developments in the field. Nevertheless, this implementation has important value as an alternative method, learning tool, and in-depth understanding of the performance of related encryption algorithms. Second, we in detail compare the performance of Unimodular Hill Cipher and RSA in terms of encryption time with diverse text data, providing valuable insights into the practical applications of both. Third, we try to fill the research gap by providing a more in-depth comparative analysis between these algorithms, which can help researchers and practitioners in selecting appropriate algorithms for their data security needs. We identified several research gaps on which to base this investigation, namely the lack of research that provides an in-depth comparison between the Unimodular Hill Cipher Method and RSA in the context of text encryption algorithms using the Python programming language, then the lack of a comprehensive understanding of the relative performance of these two methods in this scenario different uses. Through this comparative analysis, we seek to answer these questions and fill the research gaps we identified, providing better guidance for those involved in selecting data security algorithms. These novelties constitute our important contribution to the development of the field of text encryption and we hope that our work will become a useful reference resource for the information security community (Arifin and Muktyas, 2021; Xiao and Watson, 2019).

In a security context, the Unimodular Hill Cipher and RSA methods have their respective advantages. The Unimodular Hill Cipher method is known to have high security because it uses a key matrix that can be difficult to crack by attacks such as frequency analysis or brute force. On the other hand, RSA is known as a public key cryptography method that has a high level of security, especially in terms of key security. However, in terms of encryption speed, the Unimodular Hill Cipher tends to be faster due to its simple operation. A detailed comparison of the encryption speed and security of these two methods will be explained further in this study by referring to relevant literature. In the context of this research, we are aware that many previous studies have worked on similar systems related to text encryption. Therefore, we feel it is important to perform a careful comparative analysis of the performance of the algorithms we discuss in our work compared to similar previous work. The results of our analysis show that Unimodular Hill Cipher and RSA have their respective advantages and disadvantages. In terms of encryption speed with small text data, Unimodular Hill Cipher tends to be faster and more efficient. However, when dealing with larger text data, RSA shows superiority due to its ability to handle larger data volumes securely. This analysis provides readers with a deeper understanding of when and where each algorithm may be a better choice, depending on the context of their application. By conducting this comparative analysis, we hope that the contribution of our research will be to provide more precise guidance for users and developers who wish to choose an encryption algorithm that suits their needs, based on a solid understanding of the advantages and disadvantages of each algorithm in various situations (Arifin *et al*., 2016; Benssalah *et al*., 2021).

Materials and Methods

The methodology in this study aims to implement the Unimodular Hill Cipher and RSA methods in text encryption algorithms using the Python programming language. We use simple short text and long text stored in the "lorem.txt" file as sample data for the encryption process. The Unimodular Hill Cipher method will be used to encrypt text, with a randomly generated encryption matrix. Furthermore, the RSA method will be used as a public key encryption method to strengthen the security of encrypted text (Negi *et al*., 2020; Arifin *et al*., 2021b). The Unimodular Hill Cipher method is an encryption method that uses the matrix concept to convert the original text into encrypted text. In the Unimodular Hill Cipher, the encryption matrix used must meet the unimodular requirements, namely having a determinant that is an integer and an inverse modulus. The encryption process in this method involves multiplying the original text matrix by the encryption matrix, while the decryption process involves multiplying the encrypted text matrix by the decryption matrix which is the inverse of the encryption matrix. The advantage of the Unimodular Hill Cipher method is its ability to encrypt text with a high level of security. The combination of matrix operations in this method makes the encrypted text difficult to decrypt without proper encryption matrix knowledge. In addition, the Unimodular Hill Cipher method can also be applied to various types of text data, including long text. However, the Unimodular Hill Cipher method also has some drawbacks. One of them is the sensitivity to changes in the encryption matrix. If there is a small change in the encryption matrix, for example, one of the matrix elements changes its value, then the decryption result may become invalid. In addition, the encryption and decryption processes in this method can be more complex and take longer than other encryption methods (Siahaan, 2018; Mervat *et al*., 2017).

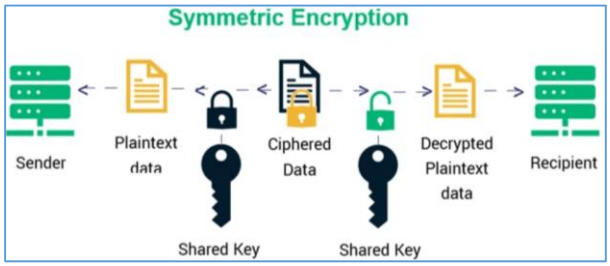

Fig. 2: An illustration of the symmetric encryption (Sertifikat-SSL/TLS, 2021)

Unimodular Hill Cipher is a text encryption method that uses matrix operations to convert clear text into encrypted text. This method is based on the theory of matrix algebra and uses an unimodular key matrix as the encryption key. The unimodular key matrix has a determinant which is a positive or negative integer 1. The steps for encryption using the Unimodular Hill Cipher are as follows: (a) Key generation: Random generation of an unimodular key matrix with the appropriate size. (b) Text-to-matrix conversion: The text matrix is constructed by combining letters in clear text. (c) Encryption: The text matrix is multiplied by the unimodular key matrix using the matrix modulo operation to produce an encrypted matrix. (d) Matrix to encrypted text conversion: The encrypted matrix is converted back to encrypted text by converting the matrix into the appropriate sequence of letters (Chillali, 2017; Ismail and Misro, 2022). An illustration of the concept of symmetric encryption can be seen in the following Fig. 2.

RSA is one of the most popular public key encryption methods. This method is based on the mathematical concept of difficult integer factorization. In RSA, there are two keys, namely the public key used for encryption and the private key used for decryption. The encryption process is carried out by taking the modulo of the exponential of the natural number with the public key, while the decryption process is carried out by taking the modulo of the exponential of the encrypted number with the private key. The main advantage of RSA is its very high security. RSA uses integer factorization which is very difficult to crack, making it difficult for someone who does not have the private key to decrypt encrypted text. Additionally, RSA supports secure key exchange and can be used in a variety of applications, including large data encryption. However, RSA also has drawbacks. One of them is its relatively slow performance compared to other encryption methods. This is due to the complexity of the mathematical operations involved in the RSA encryption and decryption process. Also, in some cases, using RSA may require complex key management, especially in terms of private key store security (Vasuki *et al*., 2022; Arifin and Garminia, 2018). Rivest-Shamir-Adleman (RSA) is an encryption method that uses the concepts of public keys and private keys. RSA is based on the difficulty of factoring large numbers. This method uses a pair of keys, namely the public key used for encryption and the private key used for decryption. The public key can be given to anyone, while the private key must be kept secret. The steps in the encryption and decryption process using RSA are as follows: (a) Key generation: Generate a pair of RSA keys consisting of a public key (e, N) and a private key (d, N). N is the product of two large prime numbers, while e and d are integers that meet certain conditions. (b) Encryption: Clear text messages are converted to integers using a specific encoding scheme. After that, the message is encrypted using the public key (e, N) and modulo exponential operation. (c) Decryption: The encrypted message is decrypted using the private key (d, N) and modulo exponential operation. (d) Conversion of integers to text: Decryption results in the form of integers converted back into readable text (Kinganga *et al*., 2021; Chauhdary *et al*., 2022).

To clarify, both Unimodular Hill Cipher and RSA methods have weaknesses and are potentially vulnerable to certain attacks. The Unimodular Hill Cipher method can be vulnerable to attacks such as frequency analysis if the length of the encrypted text is not long enough to obscure the frequency distribution of letters. However, by setting the size of the key matrix and open text large enough, this drawback can be overcome. On the other hand, RSA is known to be robust against mathematical attacks such as modulus factorization, especially if the key is large enough. The choice of method depends on the context of use and security priorities. Unimodular Hill Cipher may be preferable in situations where encryption speed is a more important factor than the highest level of security, while RSA is better suited to scenarios where absolute security is the top priority. The main difference between the Unimodular Hill Cipher method and RSA lies in the complexity of implementation and the mathematical complexity involved. Unimodular Hill Cipher is relatively simpler in terms of implementation because it involves basic matrix operations such as matrix multiplication and modulus. The selection of matrix key parameters can be adjusted to the desired security measure. However, the level of security is highly dependent on the size of the key matrix and the length of the exposed text. On the other hand, RSA involves more complex number theory mathematics, especially in generating keys and calculating modulus functions. This makes RSA more difficult to implement efficiently, especially in resource-limited applications. The security level of RSA depends on factors such as key length and complexity of the modulus factorization algorithm. In terms of the difficulty of finding or guessing parameters, both methods have a high level of security when strong key parameters are used and attempts to solve these parameters by chance are very difficult. Unimodular Hill

Cipher and RSA methods used in text encryption have a high level of security when strong key parameters are used. However, no encryption method is completely free of vulnerabilities. Essentially, the Unimodular Hill Cipher method relies on the mathematical nature of matrix operations, while RSA relies on the difficult problem of modulus factorization. For Unimodular Hill Cipher, security is highly dependent on the size of the key matrix and the length of the open text. In the case of RSA, security depends on the key length and other factors. Despite the high level of security, both Unimodular Hill Cipher and RSA are vulnerable to attacks if the parameters are chosen carelessly or if special attacks such as brute force attacks are carried out. Therefore, it is important to always choose strong key parameters and follow security best practices in the implementation of these two methods (Paragas *et al*., 2019; Feng *et al*., 2020).

Python is a high-level programming language known for its simple and easy-to-understand syntax. This language is very popular among software developers because of its ability to tackle a wide variety of programming tasks, from web development to data analysis. Python takes a structured approach and supports the object-oriented programming paradigm, thereby enabling developers to create code that is easier to organize and maintain. One of Python's main strengths is its ease of use. Its clean and easy-to-understand syntax makes it a very beginner-friendly language for beginners just entering the world of programming. Python also provides many ready-to-use libraries and frameworks, such as NumPy, Pandas, and Django, which allow developers to speed up the application development process by using existing functions and features (Ibarrondo and Viand, 2021; Arifin and Muktyas, 2018). Apart from that, Python also has a strong and active ecosystem. The large Python developer community is spread all over the world and they contribute to developing new libraries and frameworks and provide regular support and updates. This makes Python an ever-evolving language that is constantly getting feature updates and security improvements. However, like any other programming language, Python also has its drawbacks. One drawback is the relatively slow execution speed compared to other programming languages that use direct compilation. Although Python has optimized performance by using interpreters and other techniques, in some cases that require large data processing or complex calculations, Python may not be as fast as other programming languages that are closer to machine language (LaMalva *et al*., 2023; Saraswat and Raj, 2021). Also, because Python is a dynamic language, meaning that variable types do not need to be declared explicitly, this can cause some problems during debugging if a variable is miswritten or an unwanted data type change occurs. However, with available testing and rolling methods, this problem can be well addressed. Overall, Python is a powerful and versatile programming language. Its advantages including ease of use, active ecosystem, and simple syntax make it a popular choice for many types of software development. However, the disadvantages are related to the speed of execution and flexibility in setting variable types. With these considerations, the developer must choose a programming language that suits the needs and characteristics of the project being worked on (Pham *et al*., 2019; Arifin and Garminia, 2019). In this study, we try to combine symmetric and asymmetric encryption methods, which can be illustrated in the following Fig. 3.

Implementation of the Unimodular Hill Cipher and RSA methods using the Python programming language is relatively easy and makes it possible to encrypt data of various sizes. Python provides libraries and modules that support the matrix operations and large calculations required by Unimodular Hill Cipher as well as the complex mathematical calculations required by RSA. Both methods can be used to encrypt text data of varying sizes, ranging from small text strings to larger blocks of text. However, it should be noted that in the case of RSA, the larger the size of the data to be encrypted, the greater the key length required to maintain a high level of security. Therefore, selecting a key length appropriate to the size of the data to be encrypted is an important consideration in implementation. Unimodular Hill Cipher and Rivest-Shamir-Adleman (RSA) are two encryption methods that have significant differences in terms of characteristics and advantages. First, in terms of mathematical basis, Unimodular Hill Cipher uses an unimodular matrix with determinant 1 to perform encryption and decryption, relying on matrix mathematical operations. Meanwhile, RSA is based on the factorization problem of large integers, which involves modular mathematical operations with exponentiation. Furthermore, in the context of encryption type, Unimodular Hill Cipher is symmetric, where the encryption and decryption keys are identical. In contrast, RSA is an asymmetric encryption method, requiring users to have a public key for encryption and a separate private key for decryption. The advantages and disadvantages of both also need to be considered. Unimodular Hill Cipher has good encryption speed, especially for small data and its implementation is relatively simple. However, this method is vulnerable to cryptanalysis attacks, especially hill-climbing attacks if the text sample size is large enough and is not suitable for large data encryption. On the other hand, RSA offers a high level of security as it relies on hard factorization of large integers, making it suitable for large data encryption. However, RSA is slower in terms of encryption speed and has higher implementation complexity. The choice between Unimodular Hill Cipher and RSA should be based on specific needs and the level of security required, as well as the size and type of data to be encrypted (Siregar *et al*., 2019; Obaida *et al*., 2022).

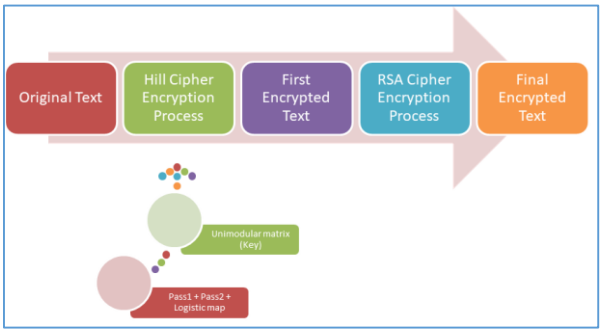

Fig. 3: An illustration of the combined method of Unimodular Hill Cipher and RSA

By combining the Unimodular Hill Cipher and RSA methods, it is expected to create a stronger and more secure text encryption system. Unimodular Hill Cipher provides uniqueness by using matrix operations, while RSA provides security through the concepts of public and private keys. The implementation of these two methods in the text encryption algorithm using the Python programming language will provide insight and a better understanding of the strength and effectiveness of the encryption method in protecting the confidentiality and integrity of text data. In this study, we used two types of text as test samples, namely short simple texts, and long texts. The simple short text consists of a few short sentences specially designed for testing purposes, while the long text is taken from the more commonly used text file, namely "lorem.txt". The use of simple short text provides the advantage in testing and understanding more quickly implementations of the combined Unimodular Hill Cipher and RSA methods. This short text allows us to perform encryption and decryption quickly and get clear results for analysis (Sulaiman and Hanapi, 2021; Lestari and Yudhanegara, 2019). Meanwhile, the use of long text from the "lorem.txt" file provides a more realistic context for testing the reliability and performance of the implemented encryption algorithm. These long texts have a much larger and more varied character count, allowing us to test the encryption system's ability to handle more complex text data. The encryption and decryption processes are performed on both types of text using the Python programming language. We used available libraries and modules such as pycrypto for the RSA implementation, as well as NumPy for matrix operations on the Unimodular Hill Cipher (Team, 2019; Haryanto *et al*., 2019).

The steps to encrypt text using the Unimodular Hill Cipher algorithm and RSA in the Python programming language are as follows. In the Unimodular Hill Cipher method: (a) Key matrix preparation: Determine the unimodular key matrix which is a square matrix with determinant 1. (b) Convert text to number matrix: Convert the text of the message to be encrypted into a number matrix, usually using ASCII values or other characters. (c) Matrix replacement: The message number matrix is multiplied by the key matrix. (d) Modulus: The result of matrix multiplication is modulus by the number of characters used. (e) Back-to-text conversion: The resulting modulus matrix is converted back to encrypted text. More on Rivest-Shamir-Adleman (RSA): (a) Key preparation: Generate a pair of RSA keys, namely a public key (published) and a private key (secret). This involves selecting two large prime numbers and calculating the associated key. (b) Encryption: The text of the message to be encrypted is converted into an integer and then encrypted using the RSA public key. (c) Decryption: Encrypted messages can be decrypted by the recipient using the appropriate RSA private key. To ensure data security when using the Unimodular Hill Cipher or RSA algorithm, some important security measures to consider are (a) Strong key selection: The key used in encryption (either the matrix key in Unimodular Hill Cipher or the RSA key) must be strong and safe. This involves the use of large prime numbers in the case of RSA and strong unimodular matrices in the case of Unimodular Hill Cipher. (b) Key security: The private key (RSA) or key matrix (Unimodular Hill Cipher) must be stored securely and only accessed by authorized parties. (c) Key monitoring and renewal: The keys used must be monitored periodically and updated as deemed necessary to maintain security. (d) Protection of encrypted data: Encrypted data must be protected from unauthorized access during storage and transmission. This can be achieved by the use of security protocols such as HTTPS for data transmission. (e) Error handling: Protection against attacks and error handling in the encryption and decryption process is important to avoid information leakage. (f) Cooperation with security experts: If sensitive or important data is encrypted, it is important to consult with computer security experts to ensure that the steps taken are adequate to protect the data. It is important to remember that the security of an encryption system depends not only on the algorithm used but also on its implementation and management. Strong keys and good security practices are key components in keeping data safe when using encryption algorithms such as Unimodular Hill Cipher or RSA (Feng *et al*., 2021; Akinboboye *et al*., 2022).

During testing, we performed an analysis of the time needed to encrypt and decrypt both types of text. We also measure the level of security of the encryption results using statistical methods such as correlation tests. This study uses a correlation test using the Jaccard Similarity and Levenshtein distance methods to analyze the level of similarity between the original text file and the encrypted file. Jaccard Similarity is used to measure the degree to which two data sets have the same elements, while Levenshtein distance is used to

measure the changes that need to be made to make the two strings identical. By combining these two methods, this research can provide a more comprehensive understanding of how similar or different the original text and the encrypted text are, so that it can provide valuable insights into evaluating the quality of the encryption method used. By using simple short text and long text from the "lorem.txt" file, we can present comprehensive and relevant results on the effectiveness and security of the combination of the Unimodular Hill Cipher and RSA methods in the text encryption process (LaMalva *et al*., 2023; Saraswat and Raj, 2021; Patel, 2022; Jatmoko *et al*., 2018).

Results and Discussion

The encryption system proposed in this research has been thoroughly evaluated by focusing attention on encryption time as one of the key metrics. This evaluation was carried out to measure the relative performance of Unimodular Hill Cipher and RSA in encrypting text based on various sizes and types of data. The results of this evaluation help us understand how these two methods behave in the context of text use and provide important insights into selecting an encryption method that suits specific needs and conditions. The following is a simple example of implementing a Python program that has been made. The text used is a simple short text: "Dwi Wijonarko and Samsul Arifin". The full program code is at https://github.com/dwijonarko/py-hybrid-encryption. The following Table 1 is about the time analysis of the encryption results from the proposed program.

The following is the character frequency of the original text. Letter Frequency {'d': 2, 'w': 2, 'i': 4, 'j': 1, 'o': 2, 'n': 3, 'a': 4, 'r': 2, 'k': 1, 's': 2, 'm': 1, 'u': 1, 'l': 1, 'f': 1}. Word Frequency {'dwi': 1, 'wijonarko': 1, 'and': 1, 'samsul': 1, 'arifin': 1}. Bigram Frequency {'dwi wijonarko': 1, 'wijonarko dan': 1, 'dan samsul': 1, 'samsul arifin': 1}. Trigram Frequency {'dwi wijonarko and': 1, 'wijonarko and samsul': 1, 'and samsul arifin': 1}. Special Character Frequency {}. Next is the character frequency of the encryption results with password 1 which is 8 and password 2 which is 2023. The following is the result of the resulting encryption text.

v6CowLazN6pKIZVTQzo5g+gPs1obgOCb+2denR
8b5Npo3OqLh6TKv4A6EGjWLWoyPgwcS5KKJWQ
DtII0D6YAnodfqLFtd6YMqwURvOmWKhyjUUfrKsa
j+i5GAf1C/mTCb2zd/tyetz8/BLeuAblVi4OqsohZ+iD
VU967Vf1U6MXzb721yW27P0HnwmfFvcWFD6ebgp
0dejjCWrDkLpHHLVwFQ2itaPtESU6L/TdzeQ0IK+ej
p1OFKJAI3d0MheoEyqO9RMTeg7O/+KEA5V/ZvPy
ZFvXSRiPxg/CUAd3syAb6EKZjjwGowJzzVwNFAL
AHvQR7eH9b4kBApcKtKg== with the character frequency of the encrypted text as follows. Letter frequency {'v': 14, 'c': 9, 'o' 16, 'w': 15, 'l': 11, 'a': 14, 'z': 13, 'n': 6, 'p': 12, 'k': 14, 'i' 10, 't': 12, 'q': 11, 'g': 12, 's': 7, 'b': 11,'d' 13, 'e': 15, 'r': 7, 'h': 9, 'j': 11, 'y': 9, 'f': 12, 'm': 8, 'u':8, 'x':3}. Word Frequency {}. Bigram Frequency {}. Trigram Frequency {}. Special Character Frequency {'+': 6, '/': 7, '=': 2}. Character histograms of the original text and encrypted text can be seen in the following Fig. 4.

Unimodular Hill Cipher is an encryption method that uses an unimodular matrix to encrypt text. In this process, the text message is broken down into blocks of text that fit the size of the matrix, and then multiplied by the key matrix to produce the encrypted text. The strategy to strengthen the security of encrypted text via Unimodular Hill Cipher is (a) Selection of a strong key matrix: Selecting a strong key matrix with determinants that cannot be divided into small factors to avoid hill-climbing attacks. (b) Modulus usage: Uses a modulus (e.g., 26 for the English alphabet) to ensure the encryption result is within a valid character range. (c) Using Larger text blocks: Use a key matrix of a larger size to enlarge the key space and make attacks more difficult. (d) Key randomization: Periodically randomizes the key matrix to avoid attacks based on statistical analysis. RSA is an asymmetric encryption method that uses a pair of keys, namely a public key (published) and a private key (secret). Text messages are encrypted with a public key and can only be decrypted with the corresponding private key. Strategies to strengthen the security of encrypted text via RSA are (a) Selection of large prime numbers: Using very large prime numbers in the key generation process to avoid factorization attacks. (b) Private key security: Protect the private key strictly and keep it confidential. (c) Key monitoring: Monitor keys periodically and replace them if necessary to avoid attacks. (d) Secure padding Selection: Uses a secure padding scheme such as PKCS#1 OAEP to protect messages from padding-related attacks. (e) Key activity monitoring: Monitors key activity to detect signs of suspicious activity. The security strategy implemented in these two algorithms is very important to ensure that the encrypted text remains safe from cryptanalysis and other security attacks. Additionally, understanding the basic principles of security algorithms and keys is key to the effective use of Unimodular Hill Cipher or RSA (Nassar *et al*., 2023; Guleria and Mishra, 2021).

Next session, we will examine the results of the analysis and implementation of the proposed algorithm. The following Table 2 is an analysis of the implementation of the program with long and dynamic text, with the following Fig. 5 text sources.

Table 1: Time analysis of the encryption results from the proposed program

Password 1	Password 2	Encryption time (seconds)	Description time (seconds)	Initial file size (B)	Encrypted file size (B)
1	2023	0.006876468658447266	0.005435943603515625	31 B	172 B
2	2023	0.007310390472412109	0.005073070526123047	31 B	344 B
8	2023	0.014883995056152344	0.007751941680908203	31 B	344 B
1	03062023	0.006204843521118164	0.005136966705322266	31 B	172 B
2	03062023	0.008365869522094727	0.005556821823120117	31 B	344 B
8	03062023	0.009653568267822266	0.008013725280761719	31 B	344 B

Table 2: Time analysis of the encryption results from the proposed program

Password 1	Password 2	Encryption Time (seconds)	Description Time (seconds)	Initial File size (B)	Encrypted file size (B)
2	2023	0.019130229949951172	0.0038444995880126953	6 B	10240 B
120	2023	0.2046496868133545	0.15006422996520996	6 B	10072 B
420	2023	3.383981466293335	3.3521833419799805	6 B	10072 B
2	03062023	0.0053691864013671875	0.004362583160400391	6 B	10240 B
120	03062023	0.18319964408874512	0.1541285514831543	6 B	10072 B
420	03062023	3.368849515914917	3.4521877765655518	6 B	10072 B

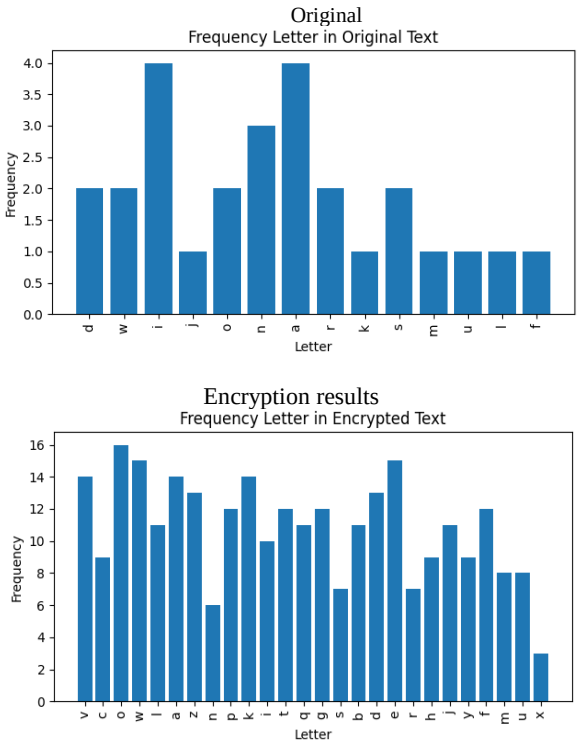

Fig. 4: Character histogram of the original text and encrypted text

Next, we will examine the character frequency of the original text. The resulting letter frequency is as follows. {'l': 93, 'o': 60, 'r': 71, 'e': 167, 'm': 64, 'i': 135, 'p': 34, 's': 129,' u': 124, 'd': 41, 't': 104, 'a': 114, 'c': 55, 'n': 85, 'g': 18, 'h': 11, 'f': 10, 'j': 4, 'v': 18, 'q': 25, 'b': 16, 'x': 5}. Word Frequency {'lorem': 4, 'ipsum': 3, 'dolor': 2, 'sit': 2, 'amet': 2, 'consectetur': 3, 'adipiscing': 1, 'elite': 2 , 'phasellus': 4, 'feugiat': 4, 'magna': 2, 'justo': 4, 'sed': 8, 'volutpat': 1, 'tortor': 1, 'interdum': 1, ' aliquam': 2, 'in': 7, 'sollicitudin': 2, 'velit': 3, 'quisque': 4, 'aliquet': 2, 'a': 4, 'sodales': 1, 'posuere': 1, 'arcu': 1, 'risus': 1, 'placerat': 2, 'quam': 3, 'vehicula': 1, 'urna': 2, 'nibh': 2, 'integer': 1 , 'ut': 4, 'congue': 2, 'mi': 1, 'ullamcorper': 3, 'pellentesque': 4, 'malesuada': 3, 'eget': 4, 'curabitur': 1, ' ultricies': 3, 'eu': 3, 'blandit': 1, 'nisl': 2, 'suspendisse': 3, 'tempus': 1, 'sapien': 1, 'dui': 2, 'viverra' : 2, 'mattis': 2, 'maecenas': 1, 'suscipit': 1, 'nec': 4, 'eros': 4, 'pharetra': 2, 'dictum': 1, 'non': 3 , 'tristique': 1, 'maximus': 4, 'lobortis': 2, 'aenean': 2, 'condimentum': 2, 'sagittis': 1, 'praesent': 1, 'faucibus': 3, ' nulla': 2, 'egestas': 2, 'commodo': 1, 'donec': 1, 'luctus': 1, 'libero': 3, 'id': 2, 'lacinia': 1, 'lacus' : 1, 'ac': 3, 'nullam': 2, 'mollis': 3, 'bibendum': 1, 'nunc': 1, 'quis': 3, 'tincidunt': 3, 'odio': 1 , 'tight': 1, 'neque': 2, 'felis': 1, 'at': 4, 'elementum': 1, 'mass': 2, 'accumsan': 1, 'vitae': 3, ' tempor': 1, 'ante': 1, 'fusce': 1, 'orci': 2, 'vestibulum': 1, 'tellus': 3, 'semper': 2, 'facilysis': 1, 'lectus' : 2, 'mauris': 1, 'duis': 1, 'varius': 1, 'sem': 1, 'vulputate': 1, 'venenatis': 1, 'nisi': 4, 'hendrerit': 2 , 'ligula': 1, 'ex': 1, 'laoreet': 1, 'ornare': 2, 'cursus': 1, 'morbi': 1, 'silence': 1, 'imperdiet': 1, ' vivamus': 1, 'proin': 1, 'et': 1}.

Furthermore, the resulting bigram frequency is as follows. {'lorem ipsum': 1, 'ipsum dolor': 1, 'dolor sit': 1, 'sit amet': 2, 'amet consectetur': 1, 'consectetur adipiscing': 1, 'adipiscing elite': 1, 'phasellus elite': 1, 'phasellus feugiat': 1, 'feugiat magna': 1, 'magna justo': 2, 'justo sed': 1, 'sed volutpat': 1, 'volutpat tortor': 1,' tortor interdum': 1, 'interdum sed': 1, 'sed aliquam': 1, 'aliquam in': 1, 'in sollicitudin': 2, 'sollicitudin velit': 1, 'velit quisque': 1, 'quisque aliquet': 1, 'aliquet ipsum': 1, 'ipsum a': 1, 'a sodales': 1, 'sodales posuere': 1, 'posuere arcu': 1, 'arcu risus': 1, 'risus placerat ': 1, 'placerat quam': 1, 'quam sit': 1, 'amet vehicula': 1, 'vehicula urna': 1, 'urna nibh': 1, 'nibh a': 1, 'a quam': 1, 'quam integer': 1, 'integer ut': 1, 'ut congue': 1, 'congue mi': 1, 'mi quisque': 1, 'quisque ut': 1, 'ut ullamcorper': 2, 'ullamcorper justo': 1, 'justo pellentesque': 1, 'pellentesque malesuada': 1, 'malesuada eget': 1, 'eget lorem': 1, 'lorem ut': 1, 'ullamcorper

curabitur': 1 , 'curabitur sed': 1, 'sed ultricies': 1, 'ultricies urna': 1, 'urna eu': 1, 'eu blandit': 1, 'blandit nisl': 1, 'nisl suspendisse': 1, 'suspendisse tempus': 1, 'tempus sapien': 1, 'sapien eget': 1, 'eget dui': 1, 'dui viverra': 1, 'viverra mattis': 1, 'mattis maecenas': 1,' maecenas magna': 1, 'justo suscipit': 1, 'suscipit nec': 1, 'nec eros': 1, 'eros pharetra': 1, 'pharetra dictum': 1, 'dictum aliquet': 1, 'aliquet eros': 1, 'eros quisque': 1, 'quisque mattis': 1, 'mattis placerat': 1, 'placerat lorem': 1, 'lorem non': 1, 'non tristique': 1, 'tristique ut': 1, 'ut consectetur': 1, 'consectetur maximus': 1, 'maximus lobortis': 1, 'aenean lobortis': 1, 'aenean condimentum': 1, 'condimentum ultricies': 1, 'ultricies sagittis': 1, 'sagittis praesent': 1, 'praesent faucibus': 1, 'faucibus nulla': 1, 'nulla sed': 1, 'sed egestas': 1, 'egestas commodo': 1, 'commodo donec': 1, 'donec condimentum': 1, 'condimentum luctus': 1, 'luctus libero': 1, 'libero id': 1, 'id lacinia': 1, 'lacinia lacus': 1, 'lacus congue': 1 , 'congue ac': 1, 'ac nullam': 1, 'nullam mollis': 1, 'mollis bibendum': 1, 'bibendum nunc': 1, 'nunc quis': 1, 'quis faucibus': 1, 'faucibus nullam': 1, 'nullam tincidunt': 1, 'tincidunt eros': 1, 'eros sed': 1, 'sed malesuada': 1, 'malesuada pellentesque': 1, 'pellentesque odio': 1,' odio repot': 1, 'erat feugiat': 1, 'feugiat neque': 1, 'neque in': 1, 'sollicitudin feis': 1, 'felis elite': 1, 'elite at': 1, 'at dolor': 1, 'dolor sed': 1, 'sed nec': 1, 'nec elementum': 1, 'elementum massa': 1, 'massa quisque': 1, 'quisque dui': 1, 'dui ipsum': 1, 'ipsum accumsan': 1, 'accumsan vitae': 1, 'vitae nibh': 1, 'nibh in': 1, 'in tempor': 1, 'tempor lobortis': 1, 'lobortis ante': 1, 'ante fusce': 1, 'fusce in': 1, 'in malesuada': 1, 'malesuada orci': 1, 'orci vestibulum': 1, 'vestibulum at': 1, 'at tellus': 1, 'tellus nec': 1, 'nec libero': 1, 'libero semper': 1, 'semper facilisis': 1, 'facilisis non': 1, 'non ac': 1, 'ac lectus': 1 , 'lectus suspendisse': 1, 'suspendisse id': 1, 'id mauris': 1, 'mauris in': 1, 'in velit': 1, 'velit ultricies': 1, 'ultricies pellentesque': 1, 'pellentesque at': 1, 'at nec': 1, 'nec justo': 1, 'justo duis': 1, 'duis eu': 1, 'eu quam': 1, 'quam viverra': 1, ' viverra velit': 1, 'velit maximus': 1, 'maximus mollis': 1, 'mollis phasellus': 1, 'phasellus maximus': 1, 'maximus varius': 1, 'varius consectetur': 1, 'consectetur phasellus': 1, 'phasellus egestas': 1, 'egestas ac': 1, 'ac sem': 1, 'sem a': 1, 'a vulputate': 1, 'vulputate suspendisse': 1, 'suspendisse quis': 1, 'quis eros': 1, 'eros pellentesque': 1, 'pellentesque venenatis': 1, 'venenatis nisi': 1, 'nisi a': 1, 'a maximus': 1, 'maximus nisi': 1, 'nisi phasellus': 1, 'phasellus vitae': 1, 'vitae tincidunt': 1, 'tincidunt nisl': 1, 'nisl in': 1, 'in faucibus': 1, 'faucibus lectus': 1, 'lectus sed': 1, 'sed feugiat': 1, 'feugiat hendrerit': 1, 'hendrerit ligula': 1, 'ligula vitae': 1, 'vitae feugiat': 1, 'feugiat ex': 1 , 'ex laoreet': 1, 'laoreet in': 1, 'in nulla': 1, 'nulla ornare': 1, 'ornare nisi': 1, 'nisi aliquam': 1, 'aliquam cursus': 1, 'cursus libero': 1, 'libero eu': 1, 'eu ornare': 1, 'ornare neque': 1, 'neque morbi': 1, 'morbi sed': 1, 'sed mollis': 1, ' silent mollis': 1, 'aenean silent': 1, 'aenean non': 1, 'non mass': 1, 'quis mass': 1, 'quis tellus': 1, 'tellus imperdiet': 1, 'imperdiet pharetra': 1, 'pharetra vivamus': 1, 'vivamus eget': 1, 'eget orci': 1, 'orci eget': 1, 'eget nisi': 1, 'nisi hendrerit': 1, 'hendrerit ullamcorper ': 1, 'ullamcorper proin': 1, 'proin at': 1, 'at semper': 1, 'semper lorem': 1, 'lorem et': 1, 'et tincidunt': 1, 'tincidunt tellus': 1}.

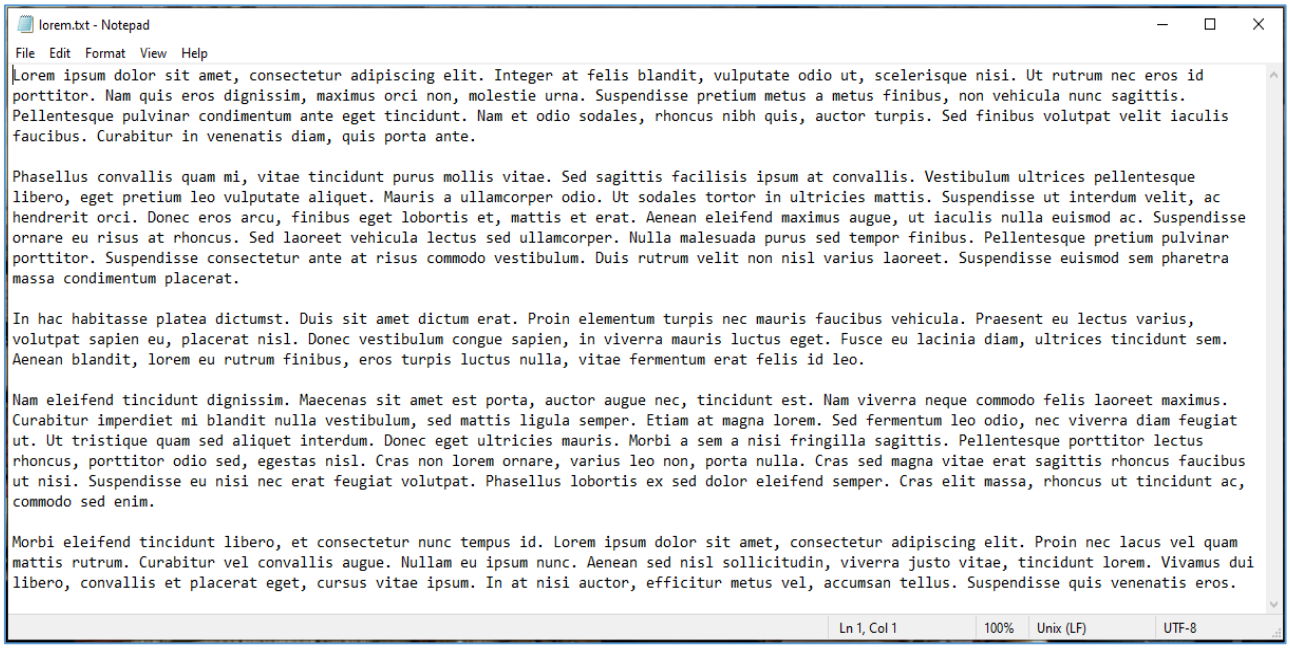

Fig. 5: The original text

Fig. 6: Encrypted text

Furthermore, the resulting trigram frequency is as follows. {'lorem ipsum dolor': 1, 'ipsum dolor sit': 1, 'dolor sit amet': 1, 'sit amet consectetur': 1, 'amet consectetur adipiscing': 1, 'consectetur adipiscing elite': 1,' adipiscing elite phasellus': 1, 'phasellus feugiat elite': 1, 'phasellus feugiat magna': 1, 'feugiat magna justo': 1, 'magna justo sed': 1, 'justo sed volutpat': 1, 'sed volutpat tortor': 1, 'volutpat tortor interdum': 1, 'tortor interdum sed': 1, 'interdum sed aliquam': 1, 'sed aliquam in': 1, 'aliquam in sollicitudin': 1, 'in sollicitudin velit' : 1, 'sollicitudin velit quisque': 1, 'velit quisque aliquet ipsum': 1, 'quisque aliquet ipsum': 1, 'aliquet ipsum a': 1, 'ipsum a sodales': 1, 'a sodales posuere': 1 , 'sodales posuere arcu': 1, 'posuere arcu risus': 1, 'arcu risus placerat': 1, 'risus placerat quam': 1, 'placerat quam sit': 1, 'quam sit amet': 1,' sit amet vehicula': 1, 'amet vehicula urna': 1, 'vehicula urna nibh': 1, 'urna nibh a': 1, 'nibh a quam': 1, 'a quam integer': 1, 'quam integer ut': 1, 'integer ut congue': 1, 'ut congue mi': 1, 'congue mi quisque': 1, 'mi quisque ut': 1, 'quisque ut ullamcorper': 1, 'ut ullamcorper justo': 1, 'ullamcorper justo pellentesque': 1, 'justo pellentesque malesuada': 1, 'pellentesque malesuada eget': 1, 'malesuada eget lorem': 1, 'eget lorem ut': 1, 'lorem ut ullamcorper': 1 , 'ullamcorper curabitur': 1, 'ullamcorper curabitur sed': 1, 'curabitur sed ultricies': 1, 'urna ultricies urna': 1, 'urna eu ultricies': 1, 'urna eu blandit': 1, ' eu blandit nisl': 1, 'blandit nisl suspendisse': 1, 'nisl suspendisse tempus': 1, 'suspendisse tempus sapien': 1, 'tempus sapien eget': 1, 'sapien eget dui': 1, 'eget dui viverra': 1, 'dui viverra mattis': 1, 'viverra mattis maecenas': 1, 'mattis maecenas magna': 1, 'maecenas magna justo': 1, 'magna justo suscipit': 1, 'justo suscipit nec' : 1, 'suscipit nec eros': 1, 'nec eros pharetra': 1, 'eros pharetra dictum': 1, 'pharetra dictum aliquet': 1, 'dictum aliquet eros': 1, 'aliquet eros quisque': 1 , 'eros quisque mattis': 1, 'quisque mattis placerat': 1, 'mattis placerat lorem': 1, 'placerat lorem non': 1, 'lorem non tristique': 1, 'non tristique ut': 1, ' tristique ut consectetur': 1, 'ut consectetur maximus': 1, 'consectetur maximus lobortis': 1, 'maximus lobortis aenean': 1, 'lobortis aenean condimentum': 1, 'aenean condimentum ultricies': 1, 'ultricies sagittis': 1, 'sagittis praesent': 1, 'praesent faucibus': 1, 'praesent faucibus nulla': 1, 'faucibus nulla sed': 1, 'nulla sed egestas': 1, 'sed egestas commodo': 1, 'egestas commodo donec': 1, 'commodo donec condimentum': 1, 'donec condimentum luctus': 1, 'condimentum luctus libero': 1, 'luctus libero id': 1, 'libero id lacinia': 1, 'id lacinia lacus': 1, 'lacinia lacus congue': 1, 'lacinia congue ac': 1, 'congue ac nullam': 1, 'ac nullam mollis': 1, 'nullam mollis bibendum': 1, ' mollis bibendum nunc': 1, 'bibendum nunc quis': 1, 'nunc quis faucibus': 1, 'quis faucibus nullam': 1, 'faucibus nullam tincidunt': 1, 'nullam tincidunt eros': 1, 'tincidunt eros sed': 1, 'eros sed malesuada': 1, 'sed malesuada pellentesque': 1, 'malesuada pellentesque odio': 1, 'pellentesque odio heer': 1, 'odio heat feugiat': 1, 'erat feugiat neque': 1, 'feugiat neque in': 1, 'neque in sollicitudin': 1, 'in sollicitudin felis': 1, 'sollicitudin felis elit': 1, 'felis elite at': 1, 'elite at dolor': 1 , 'at dolor sed': 1, 'dolor sed nec': 1, 'sed nec elementum': 1, 'elementum sed

nec': 1, 'elementum sed quisque': 1, 'massa quisque dui': 1, ' quisque dui ipsum': 1, 'dui ipsum accumsan': 1, 'ipsum accumsan vitae': 1, 'accumsan vitae nibh': 1, 'vitae nibh in': 1, 'nibh in tempor': 1, 'in temporary lobortis': 1, 'tempor lobortis ante': 1, 'lobortis ante fusce': 1, 'ante fusce in': 1, 'fusce in malesuada': 1, 'in malesuada orci': 1, 'malesuada orci vestibulum': 1, 'orci vestibulum at': 1, 'vestibulum at tellus': 1, 'at tellus nec': 1, 'tellus nec libero': 1, 'nec libero semper': 1, 'libero semper facilisis': 1 , 'semper facilisis non': 1, 'facilisis non ac': 1, 'non ac lectus': 1, 'lectus ac lectus suspendisse': 1, 'lectus suspendisse id': 1, 'suspendisse id mauris': 1, ' id mauris in': 1, 'mauris in velit': 1, 'in velit ultricies': 1, 'velit ultricies pellentesque': 1, 'ultricies pellentesque at': 1, 'pellentesque at nec': 1, 'at nec justo': 1, 'nec justo duis': 1, 'justo duis eu': 1, 'duis eu quam': 1, 'eu quam viverra': 1, 'quam viverra velit': 1, 'viverra velit maximus': 1, 'velit maximus mollis': 1, 'maximus mollis phasellus': 1, 'phasellus maximus mollis': 1, 'phasellus maximus varius': 1, 'maximus varius consectetur': 1, 'phasellus varius consectetur': 1 , 'consectetur phasellus egestas': 1, 'phasellus egestas ac': 1, 'egestas ac sem': 1, 'ac sem a': 1, 'sem a vulputate': 1, 'a vulputate suspendisse': 1,'vulputate suspendisse quis': 1, 'suspendisse quis eros': 1, 'quis eros pellentesque': 1, 'eros pellentesque venenatis': 1, 'pellentesque venenatis nisi': 1, 'venenatis nisi a': 1, 'nisi a maximus': 1, 'a maximus nisi': 1, 'maximus nisi phasellus': 1, 'phasellus vitae nisi': 1, 'phasellus vitae tincidunt': 1, 'vitae tincidunt nisl': 1, 'tincidunt nisl in': 1, 'nisl in faucibus': 1, 'in faucibus lectus': 1, 'faucibus lectus sed': 1, 'lectus sed feugiat': 1, 'sed feugiat hendrerit': 1, 'feugiat hendrerit ligula': 1 , 'hendrerit ligula vitae': 1, 'ligula vitae feugiat': 1, 'vitae feugiat ex': 1, 'feugiat ex laoreet': 1, 'ex laoreet in': 1, 'laoreet in nulla': 1,' in nulla ornare': 1, 'nulla ornare nisi': 1, 'ornare nisi aliquam': 1, 'nisi aliquam cursus': 1, 'aliquam cursus libero': 1, 'cursus libero eu': 1, 'libero eu ornare': 1, 'eu ornare neque': 1, 'ornare neque morbi': 1, 'neque morbi sed': 1, 'morbi sed mollis': 1, 'sed mollis silence': 1, 'mollis silence aenean' : 1, 'aenean non silence': 1, 'aenean non massa': 1, 'quis non massa': 1, 'quis quis tellus masses': 1, 'quis tellus imperdiet': 1, 'tellus imperdiet pharetra': 1 , 'imperdiet pharetra vivamus': 1, 'pharetra vivamus eget': 1, 'vivamus eget orci': 1, 'eget orci eget': 1, 'orci eget nisi': 1, 'eget nisi hendrerit': 1,'nisi hendrerit ullamcorper': 1, 'hendrerit ullamcorper proin': 1, 'ullamcorper proin at': 1, 'proin at semper': 1, 'at semper lorem': 1, 'semper lorem et': 1, 'lorem et tincidunt': 1, 'et tincidunt tellus': 1} . Finally, the resulting special character frequency is as follows: {',': 21, '.': 33}.

The following is a character frequency analysis of the encryption results with password 1 being 120 and password 2 being 2023. The resulting Encryption Text is as follows, which can be seen in the following Fig. 6.

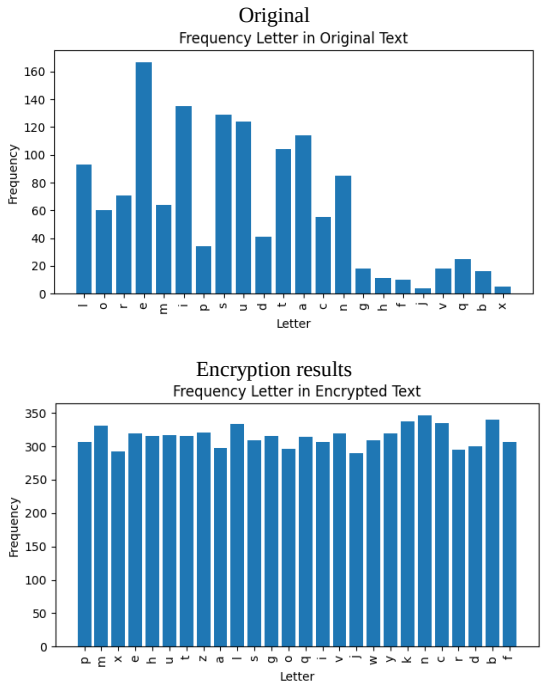

Fig. 7: Character histogram of the text before and after the encryption process

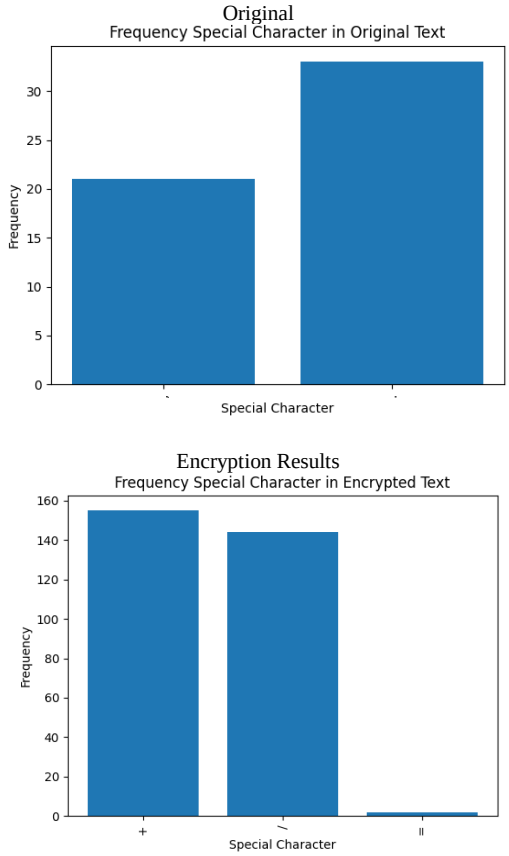

Fig. 8: Histogram of the special character frequencies of the text before and after the encryption process

Fig. 9: Correlation test of 2 texts

With letter frequency {'p': 307, 'm': 331, 'x': 292, 'e': 319, 'h': 316, 'u': 317, 't': 316, 'z': 321, 'a': 297, 'l': 334, 's': 309, 'g': 315, 'o' 296, 'q': 314, 'i' 306, 'v': 320, 'j': 290, 'w': 309, 'y': 320, 'k': 337, 'n': 347, 'c': 335, 'r': 295, 'd': 300, 'b': 340, 'f': 306}, word frequency {}, bigram frequency {}, trigram frequency {} and special character frequency {'+': 155, '/': 144, '=': 2}. Character histogram of the text before and after the encryption process and the histogram of the special character frequencies of the text before and after the encryption process can be seen in the following Figs. 7-8 respectively.

In our evaluation of this study, we performed a series of rigorous tests on the use of Unimodular Hill Cipher and RSA in encrypting text using Python. First, we measure the encryption time for various text sizes and the results show that Unimodular Hill Cipher has better speed, especially for small text sizes, while RSA is more efficient for large data sizes. Security analysis shows that RSA is more resistant to brute force attacks than Unimodular Hill Cipher, although both have vulnerabilities to cryptanalysis attacks such as frequency analysis. Furthermore, testing on various types of data revealed that they work well for both natural language text and binary data, with little error. However, decryption results for complex codes show that Unimodular Hill Cipher is more susceptible to decryption errors. Test results on different hardware configurations also show that Unimodular Hill Cipher can be implemented well on simpler hardware, while RSA requires more resources. In conclusion, this evaluation helps understand the strengths and weaknesses of each encryption method in the context of using texts with various characteristics. The choice of method depends on specific needs, including speed, security, and available hardware resources. In the context of this research, we are aware that many previous studies have worked on similar systems related to text encryption. Therefore, we feel it is important to perform a careful comparative analysis of the performance of the algorithms we discuss in our work compared to similar previous work. The results of our analysis show that Unimodular Hill Cipher and RSA have their respective advantages and disadvantages. In terms of encryption speed with small text data, Unimodular Hill Cipher tends to be faster and more efficient. However, when dealing with larger text data, RSA shows superiority due to its ability to handle larger data volumes securely. This analysis provides readers with a deeper understanding of when and where each algorithm may be a better choice, depending on the context of their application. By conducting this comparative analysis, we hope that the contribution of our research will be to provide more precise guidance for users and developers who wish to choose an encryption algorithm that suits their needs, based on a solid understanding of the advantages and disadvantages of each algorithm in various situations (Kumar and Dua, 2023; Rodríguez-Sánchez *et al*., 2020). Finally, it will be discussed about the correlation test between the original text and the encrypted text. The Jaccard Similarity results with a value of 0.0 indicate that there are no word similarities between the original text file (lorem.txt) and the encrypted file (encrypti.txt). Jaccard Similarity measures the extent to which two data sets have the same elements relative to the total elements present. A value of 0.0 indicates that no elements are in common between the two text files. Meanwhile, the result of Levenshtein Distance which has a value of 9203 shows the distance

(change) that needs to be done to convert one string into another. The Levenshtein Distance measures the degree to which two strings differ in terms of the number of delete, replace, or add operations needed to make them identical. A value of 9203 indicates that there is a large difference between the original text file and the encrypted file, with many operations that need to be performed to equate the two. In both cases, a result close to zero (0.0) on Jaccard Similarity and a high score (9203) on Levenshtein Distance indicates that the original text file and the encrypted file are very different from each other in terms of the words used or their overall structure. This indicates that the encryption process has resulted in significant changes to the original text. The correlation test of the original text and final encrypted text can be seen in the following Fig. 9.

Conclusion

In this research, we have successfully applied a combination of the Unimodular Hill Cipher and RSA methods to a text encryption algorithm using the Python programming language. The research results show that the combination of these two methods can produce a higher level of security in the text encryption process. Through analysis and testing, we found that the Unimodular Hill Cipher can scramble text characters well using the modulo matrix operation. In addition, the RSA method provides a high level of security through the use of public keys and private keys. The combination of these two methods allows us to combine the advantages of each method and produce a more robust text encryption system. Apart from that, in this study, we also analyzed the encryption results using statistical methods such as the correlation test. The results of the analysis show that the relationship between the original text and the results of the encryption is minimal, indicating a good level of security from the encryption system implemented. In both cases, a result close to zero (0.0) on Jaccard Similarity and a high score (9203) on Levenshtein Distance indicates that the original text file and the encrypted file are very different from each other in terms of the words used or their overall structure. This indicates that the encryption process has resulted in significant changes to the original text. However, we realize that several aspects still need attention for further development. For example, further research is needed to improve the speed and efficiency of the implemented algorithm. In addition, a more in-depth security analysis can also be carried out to test the security of the encryption system against attacks. Overall, this research contributes to the development of an effective and secure text encryption algorithm. The combination of the Unimodular Hill Cipher and RSA methods shows great potential in securing text data and can be applied in various applications that require information confidentiality. It is hoped that this research can become the basis for further development in the field of information security.

In this research, we also recommend using Unimodular Hill Cipher and RSA for text encryption with Python, especially suitable for types of text data that are quite small in size or data that do not require a very high level of security. Examples are short text messages or internal communications that require light encryption. However, it should be noted that although the Unimodular Hill Cipher method has good encryption speed for small data, it also has vulnerabilities to cryptanalysis attacks, especially if the text sample size is large enough. Therefore, for data that requires high security or is large in size, using RSA is recommended. The use of RSA provides a higher level of security because it relies on difficult large integer factorization problems. However, keep in mind that RSA implementations tend to be more complex and require extra attention to security and management keys. Before encrypting data, it is important to consider the level of security required and the type of data to be encrypted in order to choose wisely between Unimodular Hill Cipher and RSA.

Acknowledgment

The authors would like to thank the reviewers for their informative comments suggestions ideas, which have helped Mould this manuscript into something that is worthy of publication.

Funding Information

This study is supported and funded by Lembaga Penelitian dan Pengabdian kepada Masyarakat (LPPM), Institut Teknologi Sains Bandung.

Author's Contributions

Samsul Arifin: Coding the program, written and finalized the manuscript.

Dwi Wijonarko: Collecting the data, tidying up the theoretical basis, and the methods we use.

Suwarno: Simulating the data, tidying up the theoretical basis and the methods we use.

Edwin Kristianto Sijabat: Written and finalized the manuscript.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that there is no conflict of interest in this study and no ethical issues involved.

References

Agustini, S., & Kurniawan, M. (2019). Peningkatan Keamanan Teks Menggunakan Kriptografi Dan Steganografi. *Scan: Jurnal Teknologi Informasi dan Komunikasi*, *14*(3), 33-38. https://doi.org/10.33005/scan.v14i3.1685

Akinboboye, A. J., Oluwole, A. S., Akinsanmi, O., & Amoran, A. E. (2022). Cryptographic Algorithms for IoT Privacy: A Technical Review. *Int. J. Eng. Trends Technol*, *70*(8), 185-193. https://doi.org/10.14445/22315381/IJETT-V70I8P219

Arifin, S., & Garminia, H. (2019). Uniserial dimension of module zm × zn over Z using Python. *Int. J. Sci. Technol. Res*, *8*, 194-9. https://www.researchgate.net/publication/334769797_Uniserial_Dimension_Of_Module_ZmxZn_Over_Z_Using_Python

Arifin, S., & Muktyas, I. B. (2018). Membangkitkan suatu matriks unimodular dengan python. *Jurnal Derivat: Jurnal Matematika dan Pendidikan Matematika*, *5*(2), 1-9. https://doi.org/10.31316/j.derivat.v5i2.361

Arifin, S., & Muktyas, I. B. (2021, April). Generate a system of linear equation through unimodular matrix using Python and Latex. In *AIP Conference Proceedings* (Vol. 2331, No. 1). AIP Publishing. https://doi.org/10.1063/5.0041651

Arifin, S., Garminia, H., & Astuti, P. (2016). Dimensi Valuasi Dari Daerah Ideal Utama. In *Prosiding Seminar Nasional*. https://core.ac.uk/download/pdf/80763448.pdf#page=18

Arifin, S., & Garminia, H. (2018). Valuation Dimension of Ring Zn Using Python. *Int. J. Eng. Technol*, *7*, 6351-6. https://doi.org/ 10.14419/ijet.v7i4.16094

Arifin, S., Muktyas, I. B., Prasetyo, P. W., & Abdillah, A. A. (2021a). Unimodular matrix and bernoulli map on text encryption algorithm using python. *Al-Jabar: Jurnal Pendidikan Matematika*, *12*(2), 447-455. https://doi.org/10.24042/ajpm.v12i2.10469

Arifin, S., Muktyas, I. B., & Sukmawati, K. I. (2021b, February). Product of two groups integers modulo m, n and their factor groups using python. In *Journal of Physics: Conference Series* (Vol. *1778*, No. 1, p. 012026). IOP Publishing. https://doi.org/10.1088/1742-6596/1778/1/012026

Basavaiah, J., Anthony, A. A., & Patil, C. M. (2021). Visual Cryptography Using Hill Cipher and Advanced Hill Cipher Techniques. In *Advances in VLSI, Signal Processing, Power Electronics, IoT, Communication and Embedded Systems: Select Proceedings of VSPICE 2020* (pp. 429-443). Springer Singapore. https://doi.org/10.1007/978-981-16-0443-0_34

Benssalah, M., Rhaskali, Y., & Drouiche, K. (2021). An efficient image encryption scheme for TMIS based on elliptic curve integrated encryption and linear cryptography. *Multimedia Tools and Applications*, *80*(2), 2081-2107. https://doi.org/10.1007/s11042-020-09775-9

Chauhdary, S. H., Alkatheiri, M. S., Alqarni, M. A., & Saleem, S. (2022). Improved encrypted AI robot for package recognition in IoT logistics environment. *Journal of Electronic Imaging*, *31*(6), 061813-061813. https://doi.org/10.1117/1.JEI.31.6.061813

Chillali, A. (2017). Matrix encryption scheme. *Adv. Sci. Technol. Eng. Syst.*, vol. *2*, no. 4, pp. 56-58, https://doi.org/10.25046/aj020408

Fadlan, M., & Amaliah, Y. (2020, November). Double layered text encryption using beaufort and hill cipher techniques. In *2020 5th International Conference on Informatics and Computing (ICIC)* (pp. 1-6). IEEE. https://doi.org/10.1109/ICIC50835.2020.9288538

Feng, M., Chen, J., Xiang, X., Deng, Y., Zhou, Y., Zhang, Z., ... & Bu, H. (2020). An advanced automated image analysis model for scoring of ER, PR, HER-2 and Ki-67 in breast carcinoma. *IEEE Access*, *9*, 108441-108451. https://doi.org/10.1109/ACCESS.2020.3011294

Feng, W., Qin, Z., Zhang, J., & Ahmad, M. (2021). Cryptanalysis and improvement of the image encryption scheme based on Feistel network and dynamic DNA encoding. *IEEE Access*, *9*, 145459-145470. https://doi.org/10.1109/ACCESS.2021.3123571

Gibson, D. (2020). Rivest-Shamir-Adleman (RSA). https://cybersecurityglossary.com/rivest-shamir-adleman-rsa/, last accessed Jan, 2024.

Guleria, V., & Mishra, D. C. (2021). Multiple RGB image encryption algorithm with multilayers by Affine Hill Cipher with FrDCT and Arnold Transform. *Fractals*, *29*(06), 2150151. https://doi.org/10.1142/S0218348X21501516.

Haryanto, E. V., Nasution, E. D. P., Akbar, M. B., & Riza, B. S. (2019, November). Application of Hill Cipher and LSB+ 1 Methods for Messaging in Messages Inpicture. In *Journal of Physics: Conference Series* (Vol. *1361*, No. 1, p. 012009). IOP Publishing. https://doi.org/10.1088/1742-6596/1361/1/012009

Ibarrondo, A., & Viand, A. (2021, November). Pyfhel: Python for homomorphic encryption libraries. In *Proceedings of the 9th on Workshop on Encrypted Computing and Applied Homomorphic Cryptography*, (pp. 11-16). https://doi.org/10.1145/3474366.3486923

Ismail, N. H. M., & Misro, M. Y. (2022). An improved image encryption algorithm based on Bézier coefficients matrix. *Journal of King Saud University-Computer and Information Sciences*, *34*(10), 10056-10067. https://doi.org/10.1016/j.jksuci.2022.10.005

Jatmoko, C., Handoko, L. B., & Sari, C. A. (2018). Uji Performa Penyisipan Pesan Dengan Metode LSB dan MSB Pada Citra Digital Untuk Keamanan Komunikasi. *Dinamika Rekayasa*, *14*(1), 47-56.

Jayanthi, R., & Singh, K. J. (2019). A public key-based encryption and signature verification model for secured image transmission in network. *International Journal of Internet Technology and Secured Transactions*, *9*(3), 299-312. https://doi.org/10.1504/IJITST.2019.101823

Kinganga, J. M., Kasoro, N. M., Mabela, R. M. M., Kyamakya, K., & Kazadi, E. K. (2021, December). HRS-3K: A Hybrid Encryption System Based on Matrix Computation and RSA with Disordered alphabet in $\mathbb{Z}/44\mathbb{Z}$. In *2021 International Conference on Cyber Security and Internet of Things (ICSIoT)* (pp. 15-21). IEEE. https://doi.org/ 10.1109/ICSIoT55070.2021.00012

Kumar, A., & Dua, M. (2023). Audio encryption using two chaotic map based dynamic diffusion and double DNA encoding. *Applied Acoustics*, *203*, 109196. https://doi.org/10.1016/j.apacoust.2022.109196

LaMalva, G., Schmeelk, S., & Dinesh, D. (2023, March). Python Cryptographic Secure Scripting Concerns: A Study of Three Vulnerabilities. In *Future of Information and Communication Conference* (pp. 602-613). Cham: Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-28073-3_42

Lestari, K. E., & Yudhanegara, M. R. (2019). Penelitian pendidikan matematika.

Mervat, A. A., Younes, O. S., & Abdul-Kader, H. S. (2017, October). Multi secret Sharing Based on Hill Cipher and Blakley Secret Sharing. In *2017 27th International Conference on Computer Theory and Applications (ICCTA)* (pp. 76-81). IEEE. https://doi.org/10.1109/ICCTA43079.2017.9497162

Nassar, M., Ali, A. M., El-Shafai, W., Saleeb, A., Abd El-Samie, F. E., Soliman, N. F., ... & Ahmed, H. E. H. (2023). Hybrid of Distributed Cumulative Histograms and Classification Model for Attack Detection. *Computer Systems Science and Engineering*, *45*(2). https://doi.org/10.20884/1.dr.2018.14.1.200

Negi, A., Saxena, D., & Suneja, K. (2020, December). High level synthesis of chaos based text encryption using modified hill cipher algorithm. In *2020 IEEE 17th India Council International Conference (INDICON)* (pp. 1-5). IEEE. https://doi.org/10.1109/INDICON49873.2020.9342591.

Obaida, T. H., Jamil, A. S., & Hassan, N. F. (2022). A Review: Video Encryption Techniques, Advantages and Disadvantages. *Webology (ISSN: 1735-188X)*, *19*(1). https://www.webology.org/abstract.php?id=1934

Oliphant, T. E. (2007). Python for scientific computing. *Computing in Science and Engineering*, *9*(3), 10-20. https://doi.org/10.1109/MCSE.2007.58

Paragas, J. R., Sison, A. M., & Medina, R. P. (2019). A new variant of Hill cipher algorithm using modified S-box. *Int. J. Sci. Technol. Res*, *8*(10), 615-619.

Patel, V. (2022). Lorem Ipsum is simply dummy text of the printing and typesetting industry. *Journal for Transdisciplinary Research in Arts and Sciences*, *1*(1), 1-1.

Pham, V., Kim, N., Seo, E., Ha, J. S., & Chung, T. M. (2019). A Method to Enhance the Security Capability of Python IDE. In *Future Data and Security Engineering: 6th International Conference, FDSE 2019, Nha Trang City, Vietnam, November 27-29, 2019, Proceedings 6* (pp. 399-410). Springer International Publishing. https://doi.org/10.1007/978-3-030-35653-8_27

Rodríguez-Sánchez, F., Carrillo-de-Albornoz, J., & Plaza, L. (2020). Automatic classification of sexism in social networks: An empirical study on twitter data. *IEEE Access*, *8*, 219563-219576. https://doi.org/10.1109/ACCESS.2020.3042604

Santoso, Y. S. (2021). Message Security Using a Combination of Hill Cipher and RSA Algorithms. *Jurnal Matematika Dan Ilmu Pengetahuan Alam LLDikti Wilayah 1 (JUMPA)*, *1*(1), 20-28. https://doi.org/10.54076/jumpa.v1i1.38

Saraswat, P., & Raj, S. (2021). Encryption and decryption techniques in cloud computing. *International Journal of Innovative Research in Computer Science and Technology*, *9*(6), 225-228. https://acspublisher.com/journals/index.php/ijircst/article/view/11131

Sertifikat-SSL/TLS. (2021). Cara Kerja Public Key & Private Key Dalam Enkripsi Asimetris Sertifikat Ssl/Tls Berkualitas. *PusatSSL*. https://pusatssl.com/public-key-private-key-enkripsi-asimetris-ssl-tls/

Siahaan, A. P. U. (2018). Application of Hill Cipher algorithm in securing text messages. https://doi.org/10.31227/osf.io/n2kdb

Siregar, B., Gunawan, H., & Budiman, M. A. (2019, August). Message Security Implementation by Using a Combination of Hill Cipher Method and Pixel Value Differencing Method in Mozilla Thunderbird Email Client. In *Journal of Physics: Conference Series* (Vol. *1255*, No. 1, p. 012034). IOP Publishing. https://doi.org/10.1088/1742-6596/1255/1/012034

Sulaiman, S., & Hanapi, Z. M. (2021). Extensive analysis on images encryption using hybrid elliptic curve cryptosystem and hill cipher. *Journal of Computer Science*, *17*. https://doi.org/10.3844/JCSSP.2021.221.230

Team, O. (2019). Why People use Lorem Ipsum to represent dummy text? The Research of Loerem Ipsum. *Journal of Education*, *1*(1), 11-16. https://orcid.org/0000-0002-8094-4890

Vasuki, B., Shobana, L., & Roopa, B. (2022). Data encryption using face antimagic labeling and hill cipher. *Math. Stat.*, *10*(2), 431-435. https://doi.org/10.13189/ms.2022.100218

Xiao, Y., & Watson, M. (2019). Guidance on conducting a systematic literature review. *Journal of Planning Education and Research*, *39*(1), 93-112. https://doi.org/10.1177/0739456X17723971

Zhang, Z. (2022, October). Research on an Encryption Method Combining RSA and Hill Cipher. In *Proceedings of the 2022 6th International Conference on Electronic Information Technology and Computer Engineering* (pp. 1113-1118). https://doi.org/10.1145/3573428.3573628