International Journal on Advanced Science Engineering Information Technology

Vol.15 (2025) No. 2
ISSN: 2088-5334

Text Data Security through Double Encryption: Implementation of Unimodular Hill Cipher and Advanced Encryption Standard

Samsul Arifin [a,*], Dwi Wijonarko [b], Muhammad Faisal [a], Muhammad Nabil Pratama [a], Puguh Wahyu Prasetyo [c]

[a] *Department of Data Science, Faculty of Engineering and Design, Institut Teknologi Sains Bandung; Bekasi, West Java, Indonesia*
[b] *Department of Information Technology, Faculty of Computer Science, University of Jember, Jember, East Java, Indonesia*
[c] *Mathematics Education Department, Faculty of Teacher Training and Education, Universitas Ahmad Dahlan, Yogyakarta, Indonesia*
Corresponding author: [*]samsul.arifin@itsb.ac.id

***Abstract*— Data security is a critical concern in the digital era, requiring robust encryption methods to protect sensitive information. This study presents a hybrid encryption system combining Unimodular Hill Cipher (UHC) and Advanced Encryption Standard (AES), implemented in Python, to enhance security. The encryption process involves two layers: the plaintext is first encrypted using UHC with an unimodular key matrix and then re-encrypted using AES in the Electronic Codebook (ECB) mode. The system's performance was evaluated through time analysis, entropy measurement, and correlation analysis. Results showed an average encryption time of 10–300 ms, with a corresponding decryption time of 12–301 ms for text files up to 16 KB. The entropy values of ciphertexts reached an average of 7.98, indicating a high level of randomness, while the correlation between plaintext and ciphertext was as low as 0.18, confirming effective data obfuscation. Despite its strengths, the ECB mode's vulnerability to repetitive data patterns and the challenges in generating truly random UHC keys highlight areas for further improvement. Future research should explore more secure AES modes, such as Cipher Block Chaining (CBC), and enhance key generation methods for UHC. This study demonstrates the potential of hybrid encryption systems to achieve high security and efficiency, making them suitable for safeguarding sensitive text data. The implementation is publicly available for further development and testing.**

***Keywords*—Unimodular hill cipher; advanced encryption standard; hybrid encryption; data security; Python.**

Manuscript received 9 Sep. 2024; revised 19 Jan. 2025; accepted 26 Mar. 2025. Date of publication 30 Apr. 2025.

I. INTRODUCTION

Information security has become one of the most critical issues in this digital era, where almost every aspect of human life is connected to information and communication technology. The rapid growth in the use of the internet and digital devices has paved the way for various forms of threats to data privacy and security. The increasing number of cyberattacks, such as data theft, hacking, and digital fraud, has emphasized the importance of developing and implementing more secure and reliable encryption techniques to protect sensitive information. Encryption converts the original data (plaintext) into an unreadable form (ciphertext) using specific algorithms and encryption keys. The primary purpose of encryption is to ensure that only authorized parties can access the information after it has gone through the decryption process. The Advanced Encryption Standard (AES) has become one of the most widely used encryption standards among the various encryption algorithms available. It is widely recognized for its ability to provide high security and efficiency in data processing [1], [2], [3].

AES, which was implemented by the National Institute of Standards and Technology (NIST) in 2001, offers symmetrical encryption with a key length of 128-bit, 192-bit, or 256-bit. This algorithm has been extensively tested and used in various applications, including data encryption for wireless networks, data storage, and confidential communications. While AES is highly effective, the challenges continue to evolve as computing capabilities and attack methods become more sophisticated. Therefore, using additional algorithms or hybrid encryption is becoming increasingly important in dealing with this new threat. Unimodular Hill Cipher (UHC) is a variant of the Hill Cipher algorithm, which was first introduced by Lester S. Hill in 1929. Hill Cipher is a classic encryption method that uses matrix operations to convert plaintext into ciphertext. The

main advantage of the Hill Cipher is its ability to generate ciphertext that is significantly different from the plaintext, even if only one character in the plaintext is changed. Unimodular Hill Cipher is a unique form of Hill Cipher in which the key matrix used is unimodular, i.e., has a determinant of ±1. This ensures the matrix can be easily inverted, an essential condition for successful decryption. Combining the Unimodular Hill Cipher and AES in a hybrid encryption system provides several advantages. First, UHC offers additional data transformation complexity, making cryptographic analysis more difficult. Second, AES adds a recognized and proven layer of security, thereby increasing the system's resilience to various types of attacks. Combining these two algorithms is expected to result in a stronger encryption solution, more resistant to hacking or analysis attempts by unauthorized parties. All the above narratives are contained in Figure 1 below.

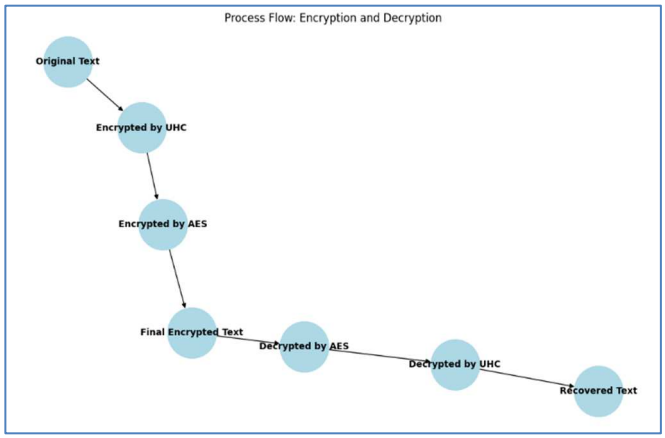


Fig. 1 Encryption and Decryption Process Flow Diagram

This diagram outlines the main stages of encryption and decryption. The process starts with the Unimodular Hill Cipher (UHC), producing an initial ciphertext, followed by Advanced Encryption Standard (AES) for the final ciphertext. Decryption reverses this sequence, beginning with AES and ending with UHC. This layered approach enhances security by adding complexity at each stage [4], [5], [6].

This research aims to develop and implement a text encryption program combining the Unimodular Hill Cipher and AES. The program is designed to encrypt text in layers, where the original text is first encrypted using UHC, and then the result of that encryption is encrypted again using AES. This approach is expected to achieve a higher level of security than using one of the algorithms separately. In this study, the encryption and decryption process were implemented using the Python programming language. Python's choice is based on its ability to support various cryptographic libraries and its ease of developing complex programs. The program generates an initial plaintext file that can be modified and an output file containing the final encryption result. The program's structure is also designed to be flexible and easy to further develop for future applications. The next section of this paper discusses a literature review of the methodology used in program development, implementation results, and analysis of the performance and security of the proposed encryption system. We hope this research can make a real contribution to the field of information security, especially in developing more secure and reliable encryption techniques [7], [8], [9].

II. Materials and Method

This methodology section outlines the systematic approach used in the research to develop and implement a text encryption system based on a combination of Unimodular Hill Cipher (UHC) and Advanced Encryption Standard (AES). This methodology is designed to ensure that every step in the encryption and decryption process is carried out precisely, allowing for a thorough evaluation of the effectiveness and security of the developed system. This section details the stages involved, from algorithm selection and preparation to code development, testing, and analyzing results. The approach implemented includes key selection, implementation of encryption and decryption algorithms, and evaluation methods to ensure that the system functions as intended and meets the expected security standards. A detailed explanation of each aspect of this methodology aims to understand clearly how these encryption systems are developed and tested. The method used in this study includes several key stages required to create, implement, and test a text encryption program using a combination of Unimodular Hill Cipher (UHC) and Advanced Encryption Standard (AES). This approach is designed to ensure that the resulting encryption solution has a high level of security and efficiency in the encryption and decryption process [10], [11], [12].

The first stage in this methodology is the selection and preparation of the algorithm. Unimodular Hill Cipher (UHC) was chosen for its ability to perform matrix-based encryption using an unimodular key matrix with a determinant of ±1. This step involves randomly generating a key matrix and ensuring its determinants are qualified to allow decryption. The plaintext text is converted into a numeric vector, which is then multiplied by a key matrix to generate the ciphertext. This ciphertext is then converted back into text characters that form the final ciphertext, which becomes the input for the next encryption stage. For the second layer, the Advanced Encryption Standard (AES), a widely recognized symmetric encryption algorithm, is used. AES keys are randomly generated and adjusted to the selected key length, such as 128-bit, 192-bit, or 256-bit. The program uses AES in the Electronic Codebook (ECB) mode of operation for simplicity of implementation. Text that has been encrypted using UHC is then re-encrypted using AES, with a process that involves dividing the ciphertext into blocks corresponding to the AES key's length, which is then encrypted independently [13], [14], [15].

The encryption program was developed using the Python programming language, which was chosen for its support for numerical operations and powerful cryptographic libraries. The main libraries include numpy for matrix operations and pycryptodome for AES implementations. The program is modularly designed, with each function handling one aspect of the encryption or decryption process, including tasks for UHC, AES, and other supporting functions. The program accepts plaintext text as input through the original.txt file, and the encrypted result is stored in the uhcAES_encrypted.txt file. The create_original_text function is provided to make it easier for users to change the content of plaintext files. After the development of the program is completed, the testing and

validation stage is carried out. Testing begins with testing the encryption and decryption functions individually to ensure each component functions correctly. Furthermore, a complete encryption process that combines UHC and AES is tested to ensure that the resulting ciphertext can be appropriately encrypted and decrypted without data loss. In addition, the program is tested against several basic cryptographic attack scenarios to provide a preliminary idea of the level of security offered by the combination of UHC and AES [16], [17], [18].

Each stage of development and testing is documented in detail to ensure process transparency and facilitate replication by other researchers. This documentation includes descriptions of key functions, examples of use, and test results. The final report includes an introduction, literature review, methodology, results, analysis, conclusions, and suggestions for further development. The program's source code and user instructions are also compiled in an easily accessible and understandable format, emphasizing ease of use and the possibility of further development by the community or other researchers. Figure 2 below provides a clearer illustration of all these processes. The flowchart illustrates the algorithm's implementation steps, including key generation, encryption, and result storage. It highlights the systematic design and ensures that each step contributes to maintaining data integrity and efficient processing [19], [20], [21].

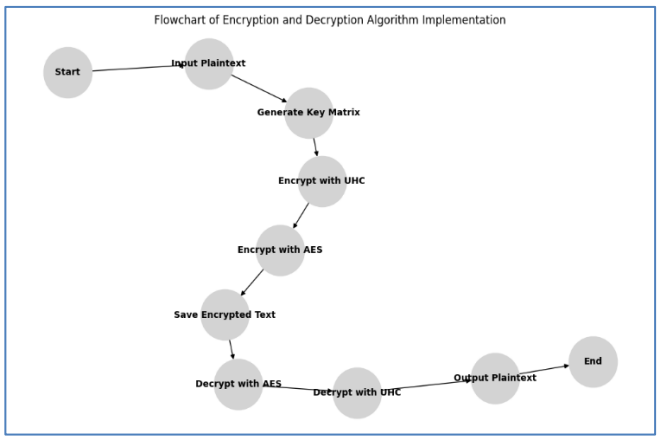


Fig. 2 Algorithm Implementation Flowchart

III. Results and Discussion

This section presents the results of implementing and testing a text encryption system that combines Unimodular Hill Cipher (UHC) and Advanced Encryption Standard (AES). The focus of this discussion is to evaluate the effectiveness and efficiency of the developed hybrid encryption system and compare the results obtained with existing theoretical and cryptographic expectations. We discuss various aspects, including the algorithm's performance in terms of encryption and decryption runtime, resistance to cryptographic attacks, and the quality of the resulting ciphertext. Additionally, this section identifies and analyzes potential weaknesses and challenges that arise during testing and provides insight into how the combination of these two algorithms affects the overall security and performance of the system. Comparing the results obtained with other encryption methods is hoped to provide a clearer picture of the advantages and limitations of the proposed hybrid approach [22], [23].

The process of encrypting and decrypting text using a hybrid system of Unimodular Hill Cipher (UHC) and Advanced Encryption Standard (AES) is carried out through several structured steps to ensure data integrity and security. At the encryption stage, the original text stored in the original.txt file is first subjected to encryption using UHC. The result of this encryption is stored in an encryption.txt file, where the original text has been converted into a ciphertext form by the UHC algorithm. Furthermore, the ciphertext generated from the UHC process is further encrypted using AES. This encryption process results in a uhcAES_encrypted.txt file, which contains the final ciphertext, resulting from a combination of the two algorithms. For the decryption process, the first step is to read the uhcAES_encrypted.txt file, which results from the final encryption. This ciphertext is then decrypted using AES to revert it to its previous form of UHC ciphertext. The AES decryption results are stored in uhcAES_decrypted.txt files. Next, these files are processed through a UHC decryption step to restore the data to its original form. This decrypted text is then stored in a dekrpsi.txt file, hopefully matching the original text in the original.txt file. This process ensures that the encrypted data can be recovered accurately, guaranteeing that the encryption and decryption systems are running correctly, and that the data remains safe. This is stated in Table 1 below [24], [25].

The original text used in this study is a key element determining the effectiveness and resilience of the developed encryption system. For experimental purposes, the original text is taken from various sources to ensure that the encryption and decryption results can be thoroughly tested on different types of content. The selected text includes various formats and complexities, including plain text of varying lengths and text containing special characters and non-alphabetic symbols. Variations in this original text aim to test the robustness of encryption systems against various scenarios and ensure that Unimodular Hill Cipher (UHC) and Advanced Encryption Standard (AES) algorithms can handle different types of data without sacrificing security or integrity. Each original text is stored in an original.txt file and processed through an encryption stage to produce a ciphertext that is then analyzed.

TABLE I
SCENARIOS OF THE ENCRYPTION PROCESS – DECRYPTION AND RESULTING FILES

Encryption Process		
Original	Encryption by UHC	Encryption by AES
original.txt	enkripsi.txt	uhcAES_encrypted .txt
Decryption Process		
Encryption Results by UHC + AES	Decryption by UHC	Original Text
uhcAES_encrypted .txt	uhcAES_decrypted .txt	dekrpsi.txt

With this approach, this study seeks to provide a comprehensive overview of encryption systems' performance and reliability on various types of inputs and assesses the effectiveness of the techniques applied in a broader context. All the above descriptions can be seen in more detail in Figure 3 below [26], [27].

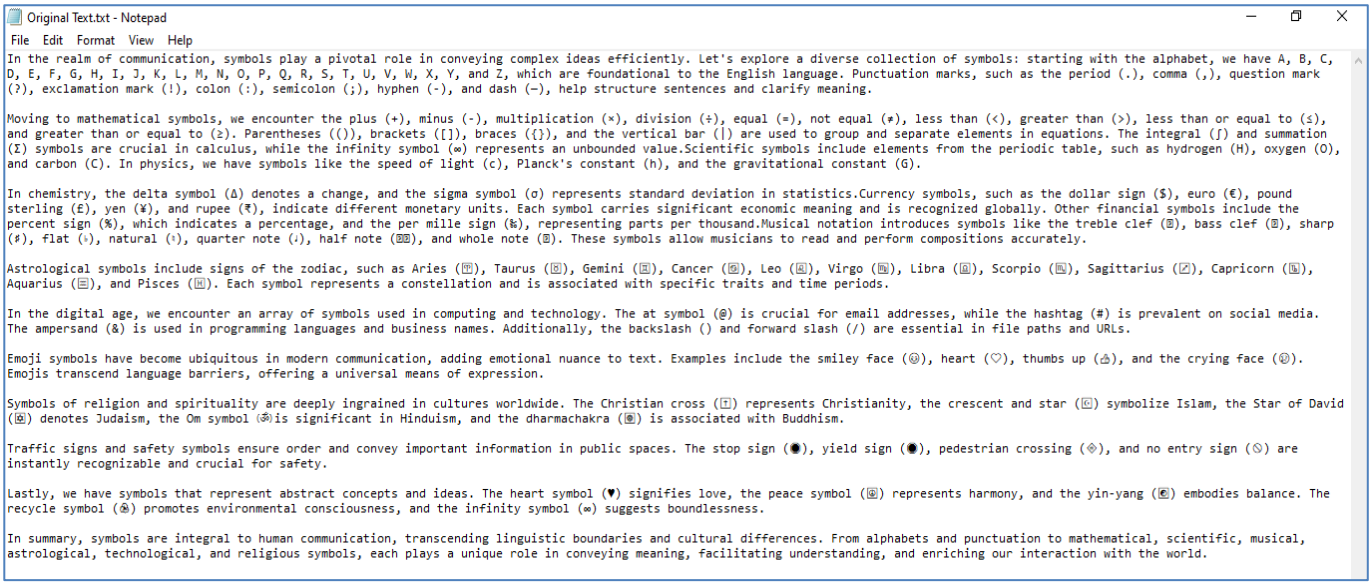


Fig. 3 The original text

The list of keys used in this encryption system is: [2, 2020, 4, 5, 101, 4040, 8, 10, 202, 40, 1010, 404, 20, 505]. These keys are chosen to ensure sufficient variation in the encryption process, with each value in the list serving as an element of the key matrix on the Unimodular Hill Cipher (UHC) algorithm. The selection of these values is based on the principle of randomness and even distribution to increase the complexity of encryption. In the context of AES, this key is also used to generate a larger encryption key, which ensures additional security for the encrypted data. The use of this diverse key list helps in testing the system's resilience to different types of inputs and ensuring that the resulting encryption is entirely secure and difficult to predict [28], [29].

For more access to the implementation and source code of text encryption systems using Unimodular Hill Cipher (UHC) and Advanced Encryption Standard (AES), you can visit our GitHub repository at https://github.com/dwijonarko/py-uhc-aes-text. This repository provides all the necessary files, including Python code for the UHC and AES algorithms, and documentation explaining how to use and set up the program. Here, you also find installation instructions, sample data, and test results that can assist you in understanding and modifying the implementation as needed. We encourage users and researchers to explore these repositories, provide feedback, or contribute to further developing these encryption systems. An important note related to the results of this study is that the decryption carried out has succeeded in recovering the original text accurately. The decryption process results in the ciphertext generated from encryption with Unimodular Hill Cipher (UHC) and Advanced Encryption Standard (AES) consistent with the original.txt files. Therefore, because the decryption generates text identical to the original text without any data loss or alteration, an in-depth analysis of the accuracy of the decryption does not need to be included separately in this report. In other words, the accuracy of the decryption results can be ensured without the need for additional analysis, so that the focus of the study can be directed to other aspects of the encryption system, such as algorithm performance and multiple analyses [30], [31].

Time, entropy, and correlation analysis are all critical aspects of evaluating encryption systems, providing in-depth insights into the performance and security of the encryption methods applied. First, time analysis measures the duration required for encryption and decryption processes on various data sizes. This measurement involves recording the execution time of each stage of the encryption and decryption process to determine the algorithm's efficiency and identify potential bottlenecks in the system. Furthermore, entropy analysis is used to measure the level of chaos in the generated ciphertext. Entropy, which measures the probability distribution of characters in ciphertext, is essential to ensure that encryption successfully eliminates exploitable patterns. High entropy indicates that the ciphertext has an even and random distribution of characters, increasing data security. Finally, correlation analysis assessed the relationship between characters in the ciphertext and the plaintext. The low correlation between the characters in ciphertext and plaintext indicates that the encryption system has effectively removed recognizable patterns from the original data. Through this analysis, we can evaluate the extent to which the encryption system successfully maintains data confidentiality and ensures that the system's performance meets the expected standards. By combining the results of these three analyses, we can get a comprehensive picture of the quality and security of the encryption systems that have been developed. All these descriptions can be seen more clearly in Table 2 below [32], [33].

Based on the encryption timetables above, the hybrid encryption system combining Unimodular Hill Cipher (UHC) and Advanced Encryption Standard (AES) demonstrates relatively efficient encryption and decryption times across various file sizes. In the previous paper by Arifin et al.]10], which combined UHC and RSA, the processing time tended to be longer, particularly for larger files, due to the computational complexity of RSA involving modular exponentiation with asymmetric keys. In comparison, the AES algorithm in this study utilizes symmetric keys, which are computationally lighter, resulting in faster encryption times, especially for larger file sizes, as shown in Table 2, where an initial file size of 6 B was encrypted to 10,240 B. Additionally, the AES-based hybrid method's distribution of encryption time is more stable than that

of RSA, which often experiences exponential increases in time for large files. This makes the UHC-AES method more practical for applications requiring both high security and time efficiency [1], [10], [34].

TABLE II
ANALYSIS OF TIME, ENTROPY, AND CORRELATION

Password 1	Password 2	Encryption Time (ms)	Decryption Time (ms)	File Size (Byte)	The Value of Entropy	Correlation
2	2024	12.229681	11.516094	16128	7.985074257	0.1820418380004798
40	2024	40.903807	26.238203	16000	7.984681571	0.1820418380004798
101	2024	112.50186	116.90855	16000	7.98094018	0.1820418380004798
2020	2024	295899.8199	300106.4525	16000	7.986204426	0.1820418380004798
2	17082024	10.204315	12.937546	16128	7.982736964	0.1820418380004798
40	17082024	39.904594	28.150797	15980	7.984135638	0.1820418380004798
101	17082024	178.717852	113.425016	15980	7.985681772	0.1820418380004798
2020	17082024	296214.7319	297732.6298	15980	7.982945143	0.1820418380004798

Based on Table 2, the following visualizations provide a clearer picture of the encryption and decryption time, file size, and entropy values for the various password combinations tested. Figure 4 below clearly visualizes the duration of time it takes to encrypt and decrypt various password combinations. This graph shows the relationship between encryption and decryption times for various password combinations.

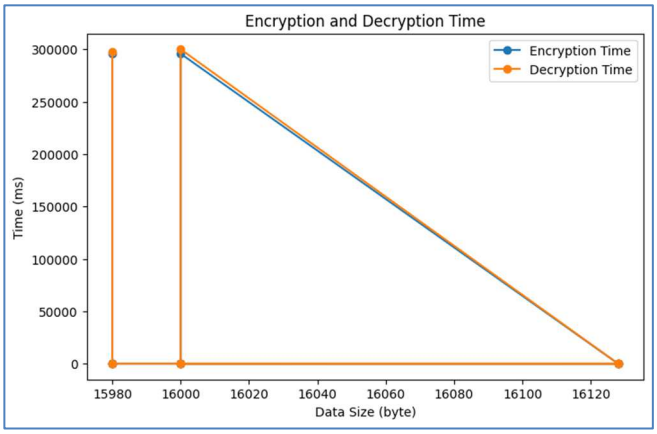


Fig. 4 Encryption and Decryption Runtime Graph

It provides insights into the algorithm's performance, indicating dependencies on password length or data size and identifying potential efficiency bottlenecks. In this graph, the X-axis shows the encryption time in milliseconds, while the Y-axis shows the decryption time in milliseconds. The dots on the graph represent each password pair tested, with labels indicating the specific combination of Password 1 and Password 2. This graph aims to illustrate the relationship between the time required for encryption and decryption and to provide insight into how much the processing time varies based on the password combination used. With this analysis, we can identify patterns or outliers in encryption and decryption performance and evaluate the efficiency of the algorithms implemented in the system [35], [36].

Figure 5 below illustrates how the entropy value changes based on the password combination used. In this graph, the X-axis represents the password pair under test, while the Y-axis represents the measured entropy value. Each dot on the plot line represents the entropy of the encrypted file for a given password combination.

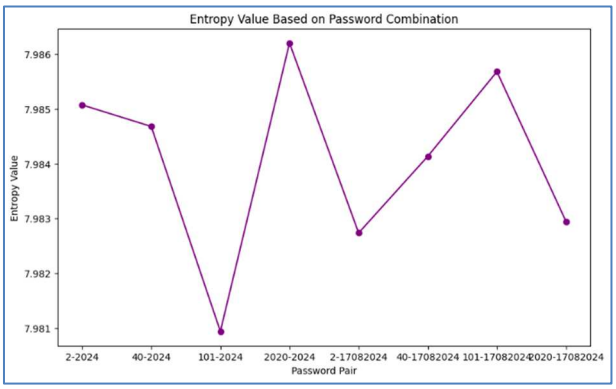


Fig. 5 Line Plot: Entropy Value Based on Password Combination

The line connecting these points allows us to see trends or patterns in entropy values. This graph is important for understanding how stable or varying the level of entropy generated by various password combinations is and for evaluating the encryption system's cryptographic strength based on the encrypted data's entropy. Moreover, this plot depicts the entropy values of the ciphertext for different password combinations. It evaluates how well the encryption obfuscates patterns in the original data, with higher entropy indicating more randomness and better cryptographic strength [37], [38].

Figure 6 provides insight into the distribution of encryption and decryption runtime for different password combinations. The boxplot displays variations in runtime for encryption and decryption across different password combinations. It identifies consistency and anomalies in performance, providing insights into the algorithm's reliability. In this boxplot, the X-axis indicates the type of time (encryption or decryption), while the Y-axis represents the duration of time in milliseconds. Each box plot depicts the interquartile, median, and outlier ranges of the time required for the encryption and decryption. With this visualization, we can quickly identify how consistent or varied the time needed for each type of process is and detect any extreme values or anomalies in the data. This graph helps evaluate the efficiency and reliability of encryption and decryption algorithms based on the measured runtime [39], [40].

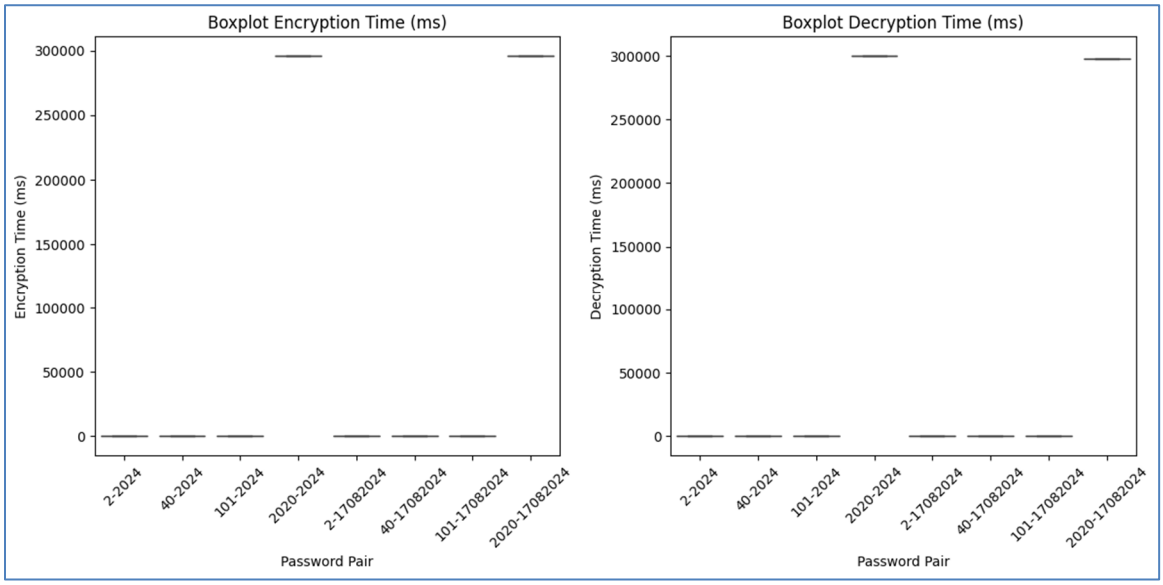


Fig. 6 Boxplot: Distribution of Encryption and Decryption Time

Figure 7 illustrates the distribution of entropy values of the encrypted data. The histogram represents the frequency distribution of entropy values for the ciphertext. A uniform distribution indicates a higher security level, while recognizable patterns may highlight weaknesses in the encryption process. On this histogram, the X-axis shows the range of entropy values, while the Y-axis depicts the frequency of occurrence of these values in the dataset. Each bar on the histogram represents the number of encrypted files with an entropy value in each interval. These graphs allow us to analyze how often certain entropy values appear and identify patterns or distributions of entropy values in the data. With this information, we can assess the cryptographic strength of the encryption system and determine how well the system generates data that has high entropy, which is an indicator of the level of security of the encryption applied [41].

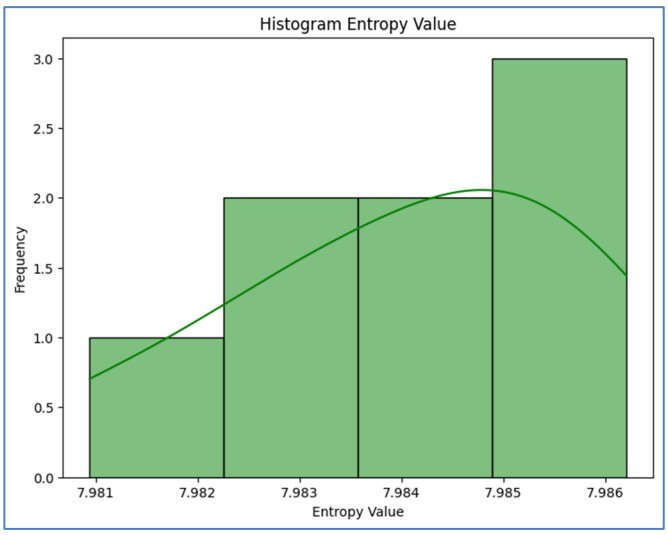


Fig. 7 Histogram Entropy Value

Figure 8 below provides a comprehensive visual comparison of different password combinations across several performance metrics. This radar chart compares password combinations based on encryption time, decryption time, file size, and entropy value. It visualizes performance differences, helping identify combinations that balance security and efficiency. Moreover, the radar chart illustrates how each password pair performs relative to encryption time, decryption time, file size, and entropy value. Each axis of the radar chart represents one of these metrics, with the data points plotted to form a polygon that encapsulates the performance profile of each password pair. By comparing these polygons, we can quickly identify strengths and weaknesses in each password's performance across various criteria. This visualization is valuable for understanding how different password combinations impact overall system performance and encryption effectiveness, offering insights into which combinations provide the optimal balance between security and efficiency [42].

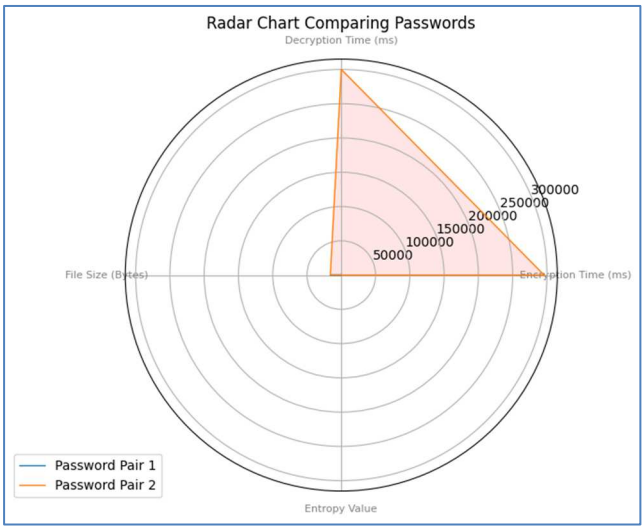


Fig. 8 Radar Chart: Comparing Passwords based on Multiple Metrics

Histogram analysis of the original text file provides in-depth insight into the frequency distribution of characters in the text. By visualizing the histogram, we can identify the

pattern of character distribution and the frequency of occurrence of each character in the text, which helps in understanding the structure and complexity of the data. This histogram allows us to see which characters appear most often and which appear infrequently, providing important information about the texture of the original data before the encryption process is implemented. This analysis is also helpful in identifying potential patterns or structures that could affect the effectiveness of encryption and for comparing how the distribution of characters changes after the text is encrypted. By understanding the initial distribution of characters, we can evaluate how well the encryption system obfuscates information and eliminates patterns that attackers can exploit. All of this can be seen more clearly in Figure 9 below. The histogram shows the frequency distribution of characters in the original text. It provides a baseline for understanding how encryption disrupts these patterns, demonstrating the algorithm's effectiveness in masking data structure [43].

Histogram analysis of encrypted text files is a crucial step in evaluating the effectiveness of the encryption system implemented. By examining the histogram of the ciphertext, we can assess the extent to which encryption has succeeded in obscuring the distribution of the original characters and eliminating recognizable patterns. This histogram provides an overview of the frequency of character occurrence in ciphertext, which should ideally show an even distribution without a clear pattern, signaling that the data has been well encrypted and secure.

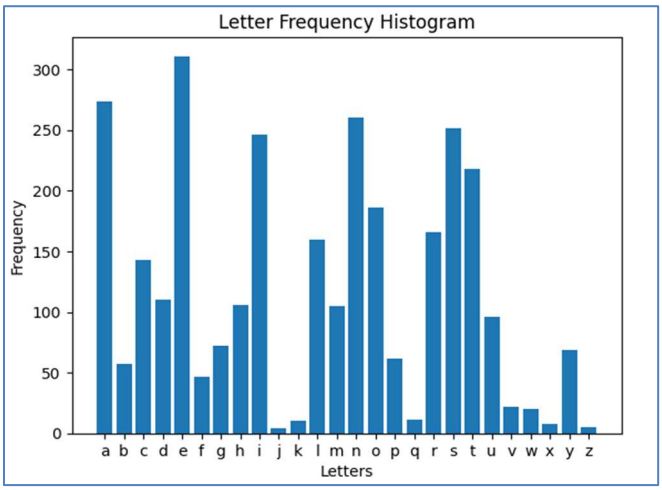
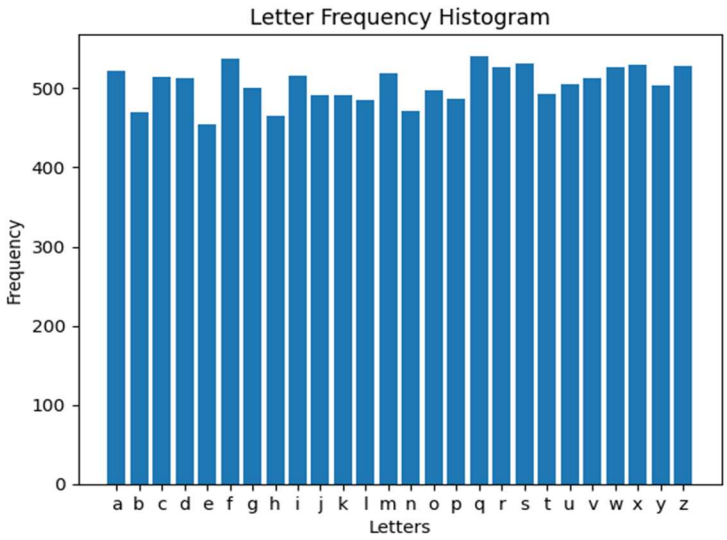
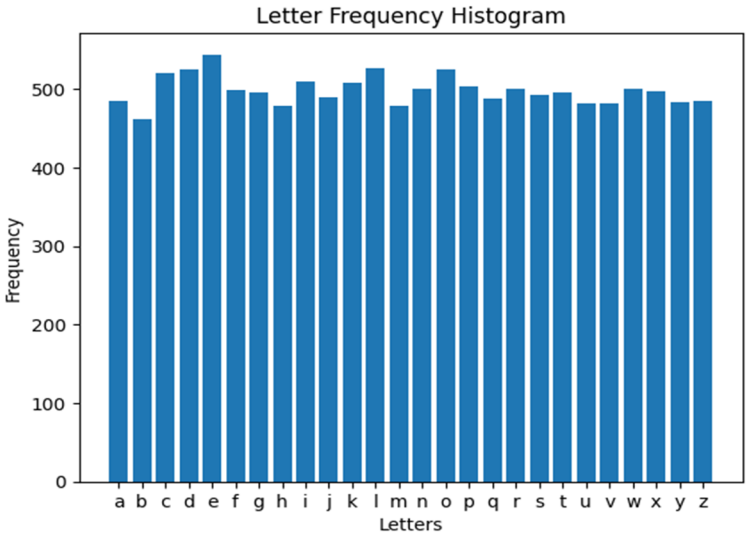
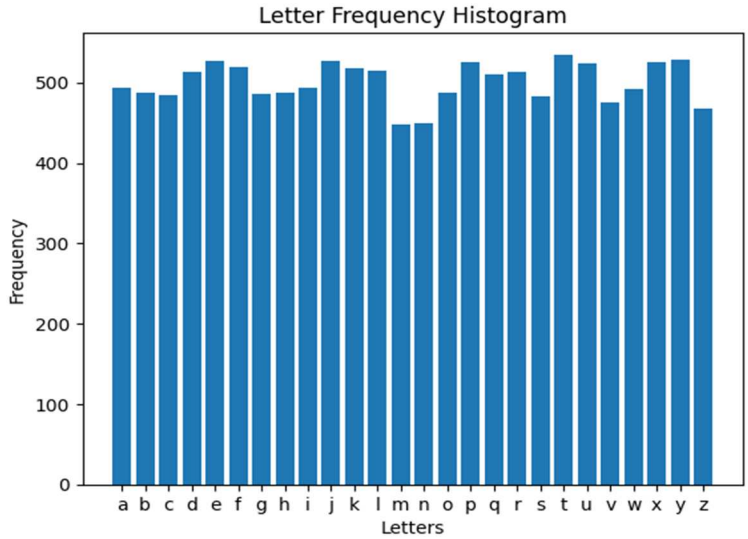
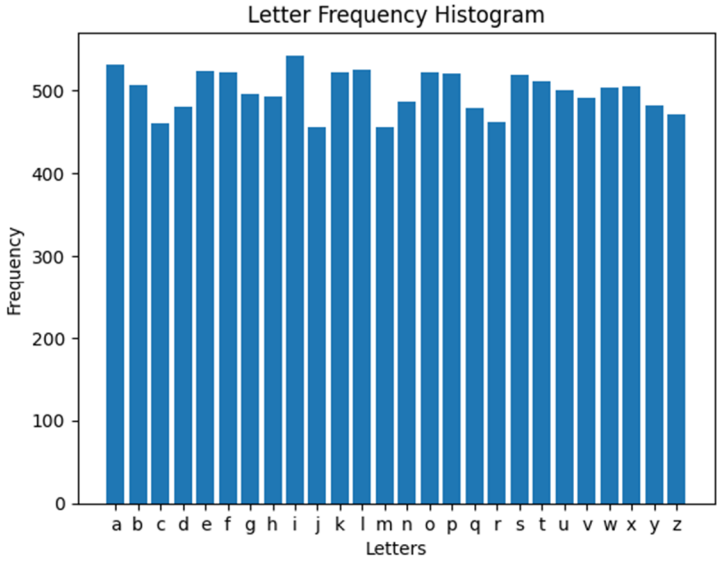


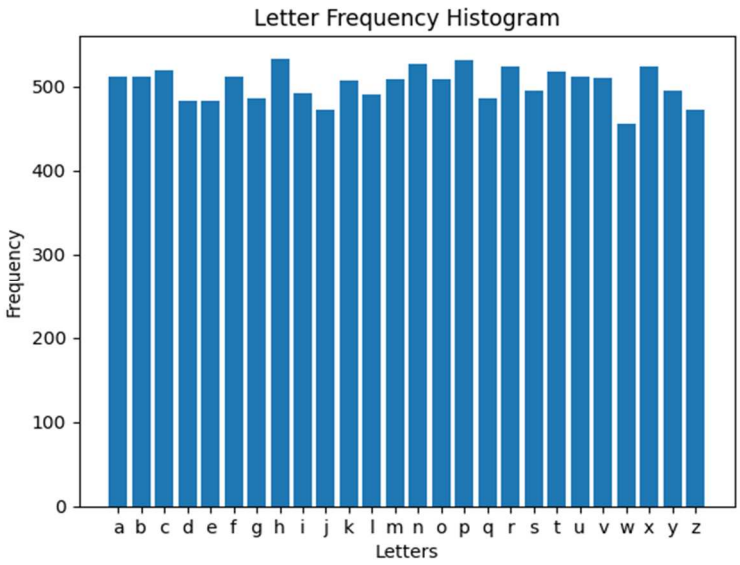
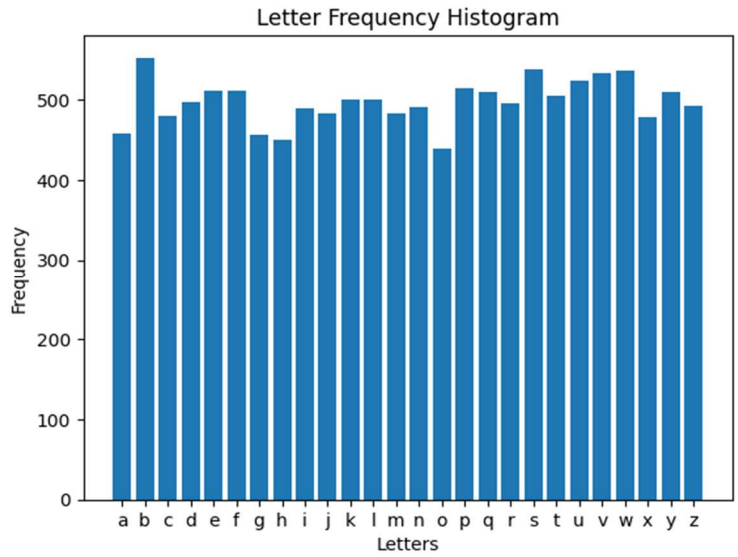
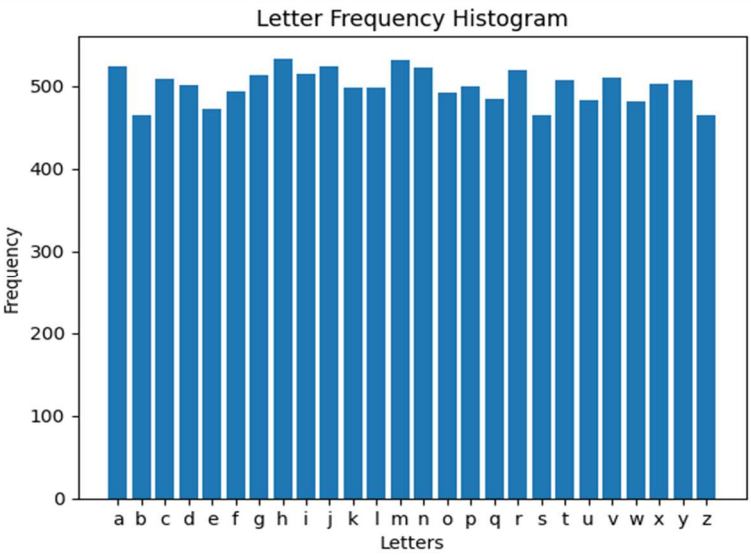
Fig. 9 Histogram analysis of the original text file

The comparison between the original text histogram and the ciphertext histogram allows us to evaluate how effective the encryption algorithm is in hiding the original information and preventing statistical analysis that the attacker might perform. By concluding this histogram analysis, we can be sure that the encryption system has functioned according to its design purpose. Thus, the histogram analysis of the encrypted text file concludes the results and discussion section of this paper, providing a comprehensive conclusion to the evaluation of the encryption system that has been developed. This can be seen more clearly in Table 3 below. [34], [44].

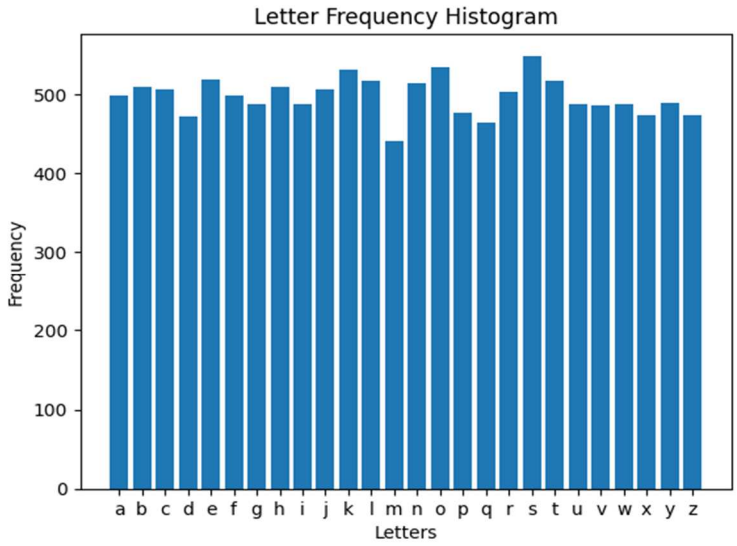
TABLE III
HISTOGRAM ANALYSIS OF ENCRYPTED TEXT FILES

Password 1	Password 2	Histogram of Encrypted Files
2	2024	

40	2024	
101	2024	
2020	2024	

2	17082024	
40	17082024	
101	17082024	

2020 17082024



The hybrid encryption system combining Unimodular Hill Cipher (UHC) and Advanced Encryption Standard (AES) demonstrates strong performance against pattern-based cryptographic attacks, as evidenced by time, entropy, and correlation analyses. However, the study does not address its effectiveness against side-channel attacks and quantum computing-based threats. Side-channel attacks, such as power or timing analysis, require implementation-level defenses like noise injection or masking. While AES remains relatively secure in a quantum context, with its 256-bit key length recommended to counteract Grover's Algorithm, UHC, as a classical algorithm, may be more susceptible to quantum analysis, especially if its key structure is predictable. Strengthening the system with longer AES keys and exploring quantum-resistant alternatives, such as lattice-based cryptography, would enhance security [4], [13], [24].

IV. Conclusion

This study's conclusion shows that combining Unimodular Hill Cipher (UHC) and Advanced Encryption Standard (AES) in a hybrid encryption system successfully provides a higher level of security compared to using either algorithm separately. UHC, with its unimodular matrix nature, can produce complex text transformations, which are difficult for attackers to analyze or break without knowing the exact keys. This provides the first layer of encryption that is already strong. However, since UHC is a classic algorithm, its use as the only encryption method may still be vulnerable to more sophisticated modern attacks. On the other hand, AES has proven to be a global encryption standard that provides an extra layer of security, especially in the face of brute force attacks and other attacks that are often effective against simpler algorithms. Combining these two algorithms in a single encryption system provides a significant advantage, where UHC creates initial complexity, and AES strengthens it with more durable and hard-to-break encryption. This double encryption process ensures that even if one layer of encryption can be broken, the other layer still protects the integrity and confidentiality of the data.

The implementation of this program using the Python programming language has also proven to be effective. Python provides robust and flexible libraries, such as numpy for matrix manipulation and pycryptodome for cryptographic operations, which facilitate the development of complex algorithms such as UHC and AES. The modular structure of the program allows for high flexibility so that it can be easily expanded or customized for other specific needs. In addition, the program has been tested in various scenarios, and the results show that this hybrid encryption system works well, resulting in a secure ciphertext that can be appropriately returned to plaintext through the decryption process. However, this study also reveals some challenges that need to be considered in further development. For example, although UHC provides complex text transformations, its security relies heavily on selecting a completely random and unpredictable key matrix. On the other hand, the AES mode of operation used, namely ECB, while easy to implement, is known to have weaknesses in handling repetitive data, which attackers can leverage to map patterns in the ciphertext. Therefore, as a further development step, it is recommended to explore the use of more secure AES operating modes, such as CBC (Cipher Block Chaining), and improve key generation methods for UHC.

Overall, the research contributes to developing more secure encryption methods by combining the power of two algorithms, UHC and AES. This hybrid encryption system shows great potential for use in a wide range of applications requiring high data protection. The results of this research are expected to be the basis for further study in the field of cryptography, especially in developing more sophisticated encryption techniques resistant to various types of modern cryptographic attacks. Although the combination of Unimodular Hill Cipher (UHC) and Advanced Encryption Standard (AES) in hybrid encryption systems shows significant improvements in terms of security, some open issues still require further research. One of the main problems is vulnerability to more sophisticated cryptographic attacks, specifically those that may exploit patterns in ciphertext generated by the Electronic Codebook (ECB) operating mode of AES. This model is known to be insecure enough for some applications, mainly when the generated ciphertext displays repetitive patterns, thus opening up opportunities for attackers

to map the original data structure. In addition, although UHC offers complexity in text transformation, generating a truly random and evenly distributed key matrix is still a challenge, as inaccuracies in key generation can reduce the expected level of security. Finally, integrating UHC and AES in the context of massive text encryption or on resource-constrained systems is also an area that needs further exploration to ensure that these solutions remain efficient without sacrificing security. Future research can focus on developing a more secure operating mode for AES and improving key generation algorithms for UHC to strengthen the resilience of these encryption systems to various types of modern cryptographic attacks.

REFERENCES

[1] S. Arifin, I. B. Muktyas, P. W. Prasetyo, and A. A. Abdillah, "Unimodular matrix and bernoulli map on text encryption algorithm using python," *Al-Jabar J. Pendidik. Mat.*, vol. 12, no. 2, pp. 447–455, 2021.

[2] M. Sokouti, A. Zakerolhosseini, and B. Sokouti, "Medical image encryption: an application for improved padding based GGH encryption algorithm," *Open Med. Inform. J.*, vol. 10, p. 11, 2016.

[3] J. Kaur, S. Lamba, and P. Saini, "Advanced encryption standard: attacks and current research trends," in *2021 international conference on advance computing and innovative technologies in engineering (ICACITE)*, IEEE, 2021, pp. 112–116.

[4] S. Arifin, F. I. F. I. Kurniadi, I. G. A. G. A. Yudistira, R. Nariswari, N. P. N. P. Murnaka, and I. B. I. B. Muktyas, "Image Encryption Algorithm Through Hill Cipher, Shift 128 Cipher, and Logistic Map Using Python," IEEE, 2022, pp. 221–226.

[5] H. Li, C. Shao, and Z. Wang, "Detecting fault injection attacks based on compressed sensing and integer linear programming," *IEEE Trans. Dependable Secur. Comput.*, vol. 16, no. 3, pp. 476–483, 2018.

[6] N. G. Zinabu and S. Asferaw, "Enhanced Security of Advanced Encryption Standard (ES-AES) Algorithm," *Am. J. Comput. Sci. Technol.*, vol. 5, no. 2, pp. 41–48, 2022.

[7] I. B. I. B. Muktyas, Sulistiawati, and S. Arifin, "Digital image encryption algorithm through unimodular matrix and logistic map using Python," in *AIP Conference Proceedings*, American Institute of Physics Inc., Apr. 2021. doi: 10.1063/5.0041653.

[8] M. M. Dimitrov, "Designing Boolean Functions and Digital Sequences for Cryptology and Communications," Bulgarian Academy of Sciences, 2023.

[9] N. Ahmad and S. M. R. Hasan, "A new ASIC implementation of an advanced encryption standard (AES) crypto-hardware accelerator," *Microelectronics J.*, vol. 117, p. 105255, 2021.

[10] S. Arifin, D. Wijonarko, Suwarno, and E. K. Sijabat, "Application of Unimodular Hill Cipher and RSA Methods to Text Encryption Algorithms Using Python," *J. Comput. Sci.*, vol. 20, no. 5, pp. 548–563, May 2024, doi: 10.3844/jcssp.2024.548.563.

[11] J. T. Ligon, *The Use of Locally Invertible Convolutional Encoders for Encryption*. North Carolina State University, 2010.

[12] B. Langenberg, H. Pham, and R. Steinwandt, "Reducing the cost of implementing the advanced encryption standard as a quantum circuit," *IEEE Trans. Quantum Eng.*, vol. 1, pp. 1–12, 2020.

[13] S. Arifin *et al.*, "Algorithm for Digital Image Encryption Using Multiple Hill Ciphers, a Unimodular Matrix, and a Logistic Map," *Int. J. Intell. Syst. Appl. Eng.*, vol. 11, no. 6, pp. 311–324, 2023.

[14] K. Muttaqin and J. Rahmadoni, "Analysis and design of file security system AES (advanced encryption standard) cryptography based," *J. Appl. Eng. Technol. Sci.*, vol. 1, no. 2, pp. 113–123, 2020.

[15] A. Hafsa, M. Fradi, A. Sghaier, J. Malek, and M. Machhout, "Real-time video security system using chaos-improved advanced encryption standard (IAES)," *Multimed. Tools Appl.*, pp. 1–24, 2022.

[16] S. Arifin, I. Bayu Muktyas, and K. Iswara Sukmawati, "Product of two groups integers modulo m,n and their factor groups using python," in *Journal of Physics: Conference Series*, IOP Publishing Ltd, Mar. 2021. doi: 10.1088/1742-6596/1778/1/012026.

[17] A. Altigani, S. Hasan, B. Barry, S. Naserelden, M. A. Elsadig, and H. T. Elshoush, "A polymorphic advanced encryption standard–a novel approach," *IEEE Access*, vol. 9, pp. 20191–20207, 2021.

[18] S. M. Kareem and A. M. S. Rahma, "New method for improving add round key in the advanced encryption standard algorithm," *Inf. Secur. J. A Glob. Perspect.*, vol. 30, no. 6, pp. 371–383, 2021.

[19] A. A. Abdillah, Azwardi, S. Permana, I. Susanto, F. Zainuri, and S. Arifin, "Performance Evaluation Of Linear Discriminant Analysis And Support Vector Machines To Classify Cesarean Section," *Eastern-European J. Enterp. Technol.*, vol. 5, no. 2–113, pp. 37–43, 2021, doi: 10.15587/1729-4061.2021.242798.

[20] M. E. Smid, "Development of the advanced encryption standard," *J. Res. Natl. Inst. Stand. Technol.*, vol. 126, 2021.

[21] I. P. Pujiono, E. H. Rachmawanto, and D. A. Nugroho, "The Implementation of Improved Advanced Encryption Standard and Least Significant Bit for Securing Messages in Images," *J. Appl. Intell. Syst.*, vol. 8, no. 1, pp. 69–80, 2023.

[22] S. Arifin and I. B. Muktyas, "Membangkitkan Suatu Matriks Unimodular Dengan Python," *J. Deriv. J. Mat. dan Pendidik. Mat.*, vol. 5, no. 2, pp. 1–10, 2018.

[23] E. G. AbdAllah, Y. R. Kuang, and C. Huang, "Advanced encryption standard new instructions (aes-ni) analysis: Security, performance, and power consumption," in *Proceedings of the 2020 12th International Conference on Computer and Automation Engineering*, 2020, pp. 167–172.

[24] S. Arifin, K. Tan, A. T. Ariani, S. Rosdiana, and M. N. Abdullah, "The Audio Encryption Approach uses a Unimodular Matrix and a Logistic Function," *Int. J. Emerg. Technol. Adv. Eng.*, vol. 13, no. 4, pp. 71–81, 2023.

[25] M. Boussif, "On The Security of Advanced Encryption Standard (AES)," in *2022 8th International Conference on Engineering, Applied Sciences, and Technology (ICEAST)*, IEEE, 2022, pp. 83–88.

[26] P. K. Keserwani and M. C. Govil, *A Hybrid Symmetric Key Cryptography Method to Provide Secure Data Transmission*, vol. 1241 CCIS. 2020. doi: 10.1007/978-981-15-6318-8_38.

[27] E. R. Persulesy and B. P. Tomasouw, "A design of a text messages security system on digital images using modified Hill Cipher and Lsb method," 2023, p. 050026. doi: 10.1063/5.0125398.

[28] P. N. Lone and D. Singh, "Application of algebra and chaos theory in security of color images," *Optik (Stuttg).*, vol. 218, 2020, doi:10.1016/j.ijleo.2020.165155.

[29] H. Touil, N. E. Akkad, and K. Satori, "Text Encryption: Hybrid cryptographic method using Vigenere and Hill Ciphers," 2020 International Conference on Intelligent Systems and Computer Vision (ISCV), pp. 1–6, Jun. 2020, doi: 10.1109/iscv49265.2020.9204095.

[30] V. N. Kumar and N. Ravi Shankar, *Cryptanalysis of A New Cryptosystem of Color Image Using a Dynamic-Chaos Hill Cipher Algorithm: A Chosen Ciphertext Attack*, vol. 1119. 2020. doi:10.1007/978-981-15-2414-1_47.

[31] J. Sathya Priya, V. Krithikaa, S. Monika, and P. Nivethini, *Ensuring Security in Sharing of Information Using Cryptographic Technique*, vol. 846. 2019. doi: 10.1007/978-981-13-2182-5_3.

[32] A. Negi, D. Saxena, and K. Suneja, "High Level Synthesis of Chaos based Text Encryption Using Modified Hill Cipher Algorithm," in *2020 IEEE 17th India Council International Conference, INDICON 2020*, 2020. doi: 10.1109/indicon49873.2020.9342591.

[33] S. Hraoui, F. Gmira, M. F. Abbou, A. J. Oulidi, and A. Jarjar, "A New Cryptosystem of Color Image Using a Dynamic-Chaos Hill Cipher Algorithm," *Procedia Comput. Sci.*, vol. 148, pp. 399–408, 2019, doi:10.1016/j.procs.2019.01.048.

[34] R. Safitri, P. W. Prasetyo, D. E. Wijayanti, S. Arifin, F. Setyawan, and J. Repka, "Improving Text Security by Using The Combination of Vigenere Cipher and Rubik's Cube Methods of the Size 4×4×4," *Al-Jabar J. Pendidik. Mat.*, vol. 14, no. 2, pp. 281–297, 2023.

[35] M. Fadlan, Suprianto, Muhammad, and Y. Amaliah, *Double layered text encryption using beaufort and hill cipher techniques*. 2020. doi:10.1109/ICIC50835.2020.9288538.

[36] D. Rachmawati, A. Sharif, and Ericko, "Hybrid Cryptosystem Combination Algorithm of Hill Cipher 3x3 and Elgamal to Secure Instant Messaging for Android," in *Journal of Physics: Conference Series*, 2019. doi: 10.1088/1742-6596/1235/1/012074.

[37] Z. Qowi and N. Hudallah, "Combining caesar cipher and hill cipher in the generating encryption key on the vigenere cipher algorithm," in *Journal of Physics: Conference Series*, 2021. doi: 10.1088/1742-6596/1918/4/042009.

[38] R. Jayanthi and K. J. Singh, "A public key-based encryption and signature verification model for secured image transmission in network," *Int. J. Internet Technol. Secur. Trans.*, vol. 9, no. 3, pp. 299–312, 2019, doi: 10.1504/IJITST.2019.101823.

[39] P. Kuppuswamy and S. Q. Al-Khalidi Al-Maliki, “A novel symmetric hybrid cryptography technique using Linear Block Cipher (LBC) and simple symmetric key,” *J. Theor. Appl. Inf. Technol.*, vol. 99, no. 10, pp. 2216–2226, 2021.

[40] D. Novianto and Y. Setiawan, “Aplikasi Pengamanan Informasi Menggunakan Metode Least Significant Bit (Lsb) dan Algoritma Kriptografi Advanced Encryption Standard (AES),” *J. Inform. Glob.*, vol. 9, no. 2, 2019.

[41] H. A. A. Al-Ukaily and R. S. Kareem, “Using the numerical solution for partial fractional differential equation by ADI numerical method to cryptography in Hill matrix system,” *J. Discret. Math. Sci. Cryptogr.*, vol. 25, no. 8, pp. 2661–2666, 2022, doi:10.1080/09720529.2021.1896649.

[42] Ritu, Niram, E. Narwal, and S. Gill, *A Novel Cipher Technique Using Substitution and Transposition Methods*, vol. 434. 2022. doi:10.1007/978-981-19-1122-4_14.

[43] B. Vasuki, L. Shobana, and B. Roopa, “Data Encryption Using Face Antimagic Labeling and Hill Cipher,” *Math. Stat.*, vol. 10, no. 2, pp. 431–435, 2022, doi: 10.13189/ms.2022.100218.

[44] F. F. M. Yahia and A. M. Abushaala, “Cryptography using Affine Hill Cipher Combining with Hybrid Edge Detection (Canny-LoG) and LSB for Data Hiding,” in *2022 IEEE 2nd International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering, MI-STA 2022 - Proceeding*, 2022, pp. 379–384. doi: 10.1109/MI-STA54861.2022.9837714.