Journal of Computer Science

Research Article

Enhancing Data Transmission Security through the Modified Hill Cipher and Henon Map Chaos Function

[1]Samsul Arifin, [2]Kevin Tan, [2]Felix Indra Kurniadi, [1]Muhammad Faisal, [3]Amril Mutoi Siregar, [4]Edwin Kristianto Sijabat, [5]Dwi Wijonarko and [6]Puguh Wahyu Prasetyo

[1]Department of Data Science, Faculty of Engineering and Design, Institut Teknologi Sains Bandung, Bekasi, West Java, Indonesia
[2]Department of Computer Science, School of Computer Science, Bina Nusantara University, Jakarta, Indonesia
[3]Department of Informatics, Faculty of Engineering and Design, Institut Teknologi Sains Bandung, Bekasi, West Java, Indonesia
[4]Department of Pulp and Paper Processing Technology, Faculty of Vocational, Institut Teknologi Sains Bandung, Bekasi, West Java, Indonesia
[5]Department of Information Technology, Faculty of Computer Science, University of Jember, Jember, East Java, Indonesia
[6]Department of Mathematics Education, Faculty of Teacher Training and Education, Universitas Ahmad Dahlan, Yogyakarta, Indonesia

Article history
Received: 22-09-2024
Revised: 25-12-2024
Accepted: 03-03-2025

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: The advancement and sophistication of technology in the present era have resulted in millions of people interacting and sharing data through the internet using various platforms such as email, chat applications, and others. This has led to the dissemination of personal information and data worldwide through the internet, which also poses the possibility of data interception by unauthorized parties. Data breaches occur when data is not securely protected. One way to secure data is by using the concept of cryptography. In this paper, a modification will be made to the classical cryptography algorithm, Hill Cipher, by incorporating the chaos function of Henon Map in generating unimodular key matrices. Based on the design and development results, the modified algorithm of Hill Cipher is capable of securing data quite effectively with a sufficiently large key space and a wide encryption set. The modified algorithm of Hill Cipher will be implemented in a data sender application.

Keywords: Cryptography, Unimodular Hill Cipher, Henon Map, Unimodular Matrix, next.js

Introduction

The progress and sophistication of technology in the present era greatly influence our daily lives. Millions upon millions of people interact and share data using email, messaging applications and various other available platforms (Kamalakannan and Tamilselvan, 2015). This has led to an increase in information and privacy data being spread worldwide through the internet network. Especially in terms of data transmission, which has now become an essential tool for people to share information quickly. Moreover, our world has just been shocked by the arrival of the Covid-19 virus pandemic, which has forced many tasks to be carried out online. We all have had to rely on data transmission to facilitate our connectivity needs and work. The extensive use of data transmission for all activities, including work, brings new concerns about the security of our data. This is because users also send data containing sensitive work information or confidential documents, making it essential for them to have a secure pathway to transmit this data (Fadlan *et al.*, 2020). One of the security issues related to data transmission is the possibility of data breaches and cyber-attacks. If these occur, sensitive information such as financial data, business secrets and so on, which are compromised, can pose a danger or harm to both organizations and individuals (Khalaf *et al.*, 2016; Arifin *et al.*, 2022).

The science of cryptography is the most suitable technique to be implemented in a data transmission application to enhance its security. Cryptography is a discipline that operates by transforming a text to be protected into an unreadable form. In cryptography, the text to be protected is called plaintext and the unreadable text is called ciphertext. The transformation of plaintext into ciphertext is a technique commonly known as “encryption”. Conversely, the technique to reverse

on the data to be transmitted, it will make it more difficult for hackers to capture and interpret the information they obtain, even if they manage to acquire the encrypted files (Arifin *et al.*, 2021; Arifin and Muktyas, 2021).

One of the cryptographic algorithms that has been used, is easy to implement and requires minimal computational resources is the Hill Cipher (Ismail *et al.*, 2006). The Hill Cipher, discovered by Lester Hill in 1929, is a classical cryptographic algorithm that utilizes symmetric block cipher techniques (Paragas *et al.*, 2019). The Hill Cipher operates by using an $m \times m>$ matrix as the key matrix in encryption and decryption operations, making this technique primarily reliant on the inverse of the key matrix, which significantly enhances its resilience against brute-force attacks. However, the Hill Cipher algorithm is fundamentally limited to an encryption range modulo 26, covering the alphabet from A to Z with corresponding representation values ranging from 0 to 25. This imposes a limitation on the Hill Cipher, allowing it to only perform encryption and decryption on words or sentences containing the alphabet from A-Z (Rubinstein-Salzedo, 2018). In this study, the provided alternative modifications will expand the encryption and decryption scope of the Hill Cipher, enabling it to accommodate and process all existing characters, including symbols and characters from all languages, with the assistance of Unicode understanding. This algorithm is also vulnerable to known plaintext attacks, where hackers have access to plaintext-ciphertext pairs, allowing them to uncover the key matrix used (Arifin and Muktyas, 2018; Arifin, 2023).

Hill Cipher is a matrix-based encryption algorithm that belongs to symmetric cryptography, where the same key is used for encryption and decryption processes. In Hill Cipher, keys in the form of invertible matrices are used to convert the original message into an encrypted form. This matrix serves as the encryption key and for decryption, the inverse matrix of the key is used to return the original message. As a form of block cipher, Unimodular Hill Cipher processes data by dividing it into blocks that are then encrypted using a key matrix. Due to its simple and efficient nature, Hill Cipher provides good speed in the encryption and decryption process, making it suitable for applications that require high performance. However, like other symmetry methods, Hill Cipher is vulnerable to attacks if the key or matrix pattern is guessable, or if too much data is encrypted without a key change. Therefore, while effective in certain situations, Hill Cipher is often combined with stronger symmetric encryption algorithms to enhance security and protect data from more sophisticated cryptographic threats (Arifin *et al.*, 2022; Paragas, 2020; Jin *et al.*, 2025). This is stated in Figure (1).

A known-plaintext attack can be considered a highly dangerous form of cryptanalysis because the attacker can more easily obtain the encryption key used to encrypt data. With the encryption key in hand, the attacker can perform decryption on other pieces of information encrypted using the same key (Rajput and Nishchal, 2013). This attack technique is usually aided by brute-force methods to attempt many potential key values that could be the actual encryption key. With the advanced and modern computational power, techniques requiring brute force have become easier and faster to accomplish (Abd-Elmonim *et al.*, 2011). By incorporating other algorithms that utilize complex mathematical functions, it is possible to enhance the randomness of the key and also expand the keyspace to reduce the susceptibility to brute-force attempts at guessing the key (Reddy *et al.*, 2012; Arifin *et al.*, 2021).

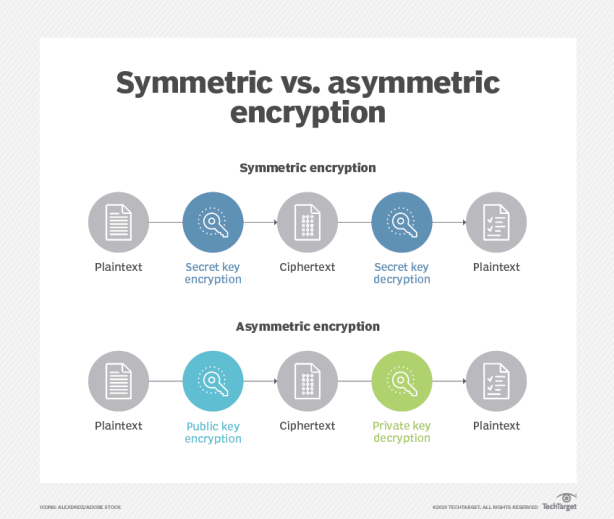

Fig. 1: Symmetric and asymmetric encryption concepts (Awati *et al.*, 2024)

One of the studies conducted to counter known plaintext attacks is the research carried out by Paragas *et al.* (2019). This study modified the key matrix generation phase using layered matrix functions to enhance the randomness of values within the key matrix. The added steps involve creating an initial random matrix, followed by performing inverse and transpose operations, converting to binary form and applying XOR to produce the key matrix. This research was evaluated using the Avalanche Effect, demonstrating that the results from this method are more secure against known plaintext attacks compared to the basic form of the Hill Cipher algorithm (Paragas *et al.*, 2019; Safitri *et al.*, 2023).

Another researcher who also modified the Hill Cipher to mitigate vulnerability against known plaintext attacks is Reddy *et al.* (2012). In their study, a circulant matrix of prime numbers was employed as the key matrix for the Hill Cipher. A circulant matrix is a square matrix where each row is generated by shifting the previous row

to the right by one position and taking the last element from the previous row as the first element of the next row. The research demonstrated that utilizing a circulant matrix of prime numbers can effectively address known plaintext attacks (Reddy *et al.*, 2012; Wen *et al.*, 2024). Another study was also conducted by Ismail *et al.* (2006), adjusting the key matrix by multiplying a vector with each row of the key matrix to generate a new key matrix. The results of this research also demonstrated that making modifications to the key matrix calculation significantly enhances the resilience of the Hill Cipher against known plaintext attacks (Ismail *et al.*, 2006; Mahmoud and Chefranov, 2010; Pribadi *et al.*, 2023).

Based on the research conducted previously, this study will present an alternative modification in the creation of the key matrix, albeit with a different approach. The method to be employed involves incorporating chaos functions into the key matrix computation to leverage the properties of chaotic functions that generate more unpredictable and chaotic random numbers. Since the Hill Cipher utilizes matrices and matrix inverses as its foundation, finding a reversible matrix becomes challenging, especially when the matrix size is larger than 4×4. To expand the key space, the key matrix will be transformed into an unimodular matrix (Arifinet al., 2021). The chaos function to be used is the Henon Map Function, which is a chaotic function that generates chaotic values in the form of a two-dimensional map (Wen, 2014). With the assistance of the chaos function, the generated values are then arranged into an upper triangular matrix. Through the use of elementary row operations, this upper triangular matrix can then be transformed into a key matrix with a determinant value of either one or negative one, signifying an unimodular matrix (Arifin *et al.*, 2021; Rrghout *et al.*, 2024; Ibrahim *et al.*, 2022).

The research conducted by Paragas *et al.* (2019) involves modifying the Hill Cipher algorithm by employing a block cipher consisting of three phases: Key matrix formation, encryption process and decryption process. The researchers divided the plaintext into parts consisting of 128-bit blocks and used the character "space" to fill empty sections. SHA256 was also utilized to create the key matrix, perform the CBC (Ciphertext Block Chaining) process, generate a hexadecimal substitution box, circular shifting, XOR operations and multiple rounds of encryption processing. With multiple layers of operations involved in key matrix formation and encryption processing, the researchers concluded that the algorithm provides a considerably strong level of security. The research conducted involves the fusion of the concepts of Affine Cipher and Hill Cipher, combined with the modern cryptographic concept of Public-Key Cryptography. The researchers first used the Fibonacci Q-Matrix as a public key, where q is a prime number. They then execute the encryption process using a modified Affine Hill Cipher algorithm, with each element within the modulo of the prime number q. As a result, both the sender and receiver of the encrypted output must possess matching keys to achieve the desired encryption outcome. The researchers state that this method is secure due to requiring multiple elements to align for successful decryption. The algorithm also employs a large key space, ensuring a high level of security (Muktyas *et al.*, 2021; Liew and Nguyen, 2020).

The research conducted by Fadlan *et al.* (2020) involves the use of two encryption layers, consisting of Beaufort Cipher and Hill Cipher. The researchers first perform the encryption process using Beaufort Cipher as the initial layer. The first process yields a ciphertext resulting from the encryption using Beaufort Cipher. This ciphertext is then further encrypted using Hill Cipher to produce a new ciphertext. For decryption, the researchers follow the reverse process: Decryption using Hill Cipher to obtain the original plaintext. This is followed by the decryption using Beaufort Cipher, resulting in the final plaintext. The researchers state that the encryption outcome achieves higher security by using two encryption layers compared to just one. The research conducted involves modifying the Hill Cipher by incorporating the Playfair Cipher and utilizing a block cipher concept with a block size of 128 bits and a key size of 256 bits. The researchers employ the Playfair Cipher to determine a rectangular matrix key which, according to research from other papers, is considered safer than a square matrix key as the resulting ciphertext will be longer than the plaintext. The researchers state that the security level of the Hill Cipher is significantly enhanced because the encryption results make it challenging to find linear equations between the ciphertext and the key matrix. The research conducted by Qowi and Hudallah (2021) involves the use of Vigenere Cipher, Caesar Cipher and Hill Cipher for encryption. The researchers aimed to strengthen the key generation process of the Vigenere Cipher, which tends to result in repetitive keys. By adding Hill Cipher and Caesar Cipher, the researchers were able to significantly reduce the level of key repetition in the encryption process. The researchers first created an encryption key using Caesar Cipher. This key was then used in the encryption process of the Modified Vigenere Cipher, which incorporates the concept of Hill Cipher multiplication. The researchers state that the encryption outcome thoroughly eliminates the weaknesses of the Vigenere Cipher, thus providing a very high level of security (Arifin *et al.*, 2021; Ahmed *et al.*, 2020; Abdillah *et al.*, 2021).

Materials and Methods

This methodology section explains two major concepts, namely Key Matrix Generation, which covers the process of generating a key matrix using elements from Unicode or binary value ranges to encrypt text and digital files and Modified Hill Cipher Algorithm, which involves the use of two keys (key1 and key2) and

modification of the traditional Hill Cipher algorithm to produce more secure encryption capable of handling a wider range of characters in Unicode or binary data. Hill Cipher fundamentally employs the method of matrix multiplication, using the plaintext matrix with the key matrix to generate the ciphertext matrix, which is the outcome of the encryption process. Then, the ciphertext matrix can be decrypted by multiplying it with the inverse of the key matrix to regain the plaintext matrix. However, in standard Hill Cipher, the resulting ciphertext tends to be confined within the scope of the alphabet A-Z because the elements of the key matrix lie within $\mathbb{Z}/26\mathbb{Z}$. In this study, two types of encryption modes can be used: The first is text encryption and the second is digital file encryption (Rubinstein-Salzedo, 2018). For text encryption, the range of key matrix elements used is from 0 to 1,114,110, with 1,114,110 being the maximum size for Unicode. By utilizing the Unicode element range, all language characters that are part of Unicode can be encrypted, resulting in encrypted output that is also random characters encompassing various languages. By using Unicode, the elements in the key matrix will be modulo 1,114,111 (Alex Martelli *et al.*, 2023). For digital file encryption, the optimal choice of element range is from 0-255, resulting in the elements being modulo 256. Encryption of digital files is performed by modifying the binary values within the file, causing the encrypted file to produce errors such as unsupported format or others, depending on the operating system and the file extension being encrypted (Paragas *et al.*, 2019a-b; Safitri *et al.*, 2023).

One constraint of using Unicode is that not all points within Unicode represent valid characters to be displayed on a computer (Alex Martelli *et al.*, 2023). Examples of invalid characters are control characters, unprintable characters and Unicode points that remain unassigned. To prevent encryption failures when encountering invalid characters, the process of multiplying the key matrix with the plaintext matrix will be repeated. With each multiplication result, every element will be checked for the presence of invalid characters. If invalid characters are found, the matrix multiplication will be redone by multiplying the key matrix with the previous multiplication result. Once no invalid characters are present, the encryption process concludes and the number of repetitions is recorded. This repetition count will also be used during the decryption process, where the inverse of the key matrix will be multiplied with the ciphertext matrix for the same number of repetitions (Alex Martelli *et al.*, 2023). The key matrix created will always be a unimodular matrix, formed through the aid of elementary row operations. The purpose of using a unimodular matrix is to ensure that the matrix is invertible, allowing for an enlarged key space/size of the key matrix without concerns of non-invertibility. The elements of the key matrix will be generated using the assistance of the Henon Map chaos function to enhance the randomness level of each element (DeBonis, 2022). A 10-character alphanumeric passkey is required and will be modified for use as an initial value in the Henon Map chaos function. The passkey's ASCII codes will be extracted from each character and then processed into decimal form Rrghout *et al.* (2024); Safitri *et al.* (2023); Azanuddin *et al.*, (2024). In Figure (2), we can see a glimpse of the Modified Hill Cipher algorithm.

```
passkey = parameters["passkey"]
sum = 1
new_passkey = ""
for i in range(len(passkey)):
    sum = 1
    for j in passkey[::-1]:
        sum += ord(passkey[i]) * ord(j)
    new_passkey += str(sum)

key2 = float("0." + new_passkey)
```

Fig. 2: Modified Hill Cipher algorithm

This modified Hill Cipher algorithm employs two keys, namely key1 and key2. Key1 functions as a determinant of the key matrix size, which will be derived from one of the divisor factors of the plaintext size. In this study, the divisor factor will be selected by the program based on the following conditions: (i) If the size of the numpy array of plaintext/ciphertext is larger than 200, then select the nearest divisor factor to 100. (ii) If the size of the numpy array of plaintext/ciphertext is larger than 200, but the nearest divisor factor to 100 exceeds 500, then the algorithm will select the nearest divisor factor to 100 that is smaller than 100. (iii) If the size of the numpy array of plaintext/ciphertext is smaller than 200, then the algorithm will select the nearest divisor factor to 200 that is smaller than 200 (Arifin *et al.*, 2022; Muralidharan *et al.*, 2023; Petrenko and Dehtiarova, 2025).

The basis for selecting a factor value around 100 is the observation of encryption and decryption times about the key matrix size. It is observed that if the size of the key matrix used exceeds 100, the encryption and decryption times increase significantly, making it unsuitable for use (Arifin *et al.*, 2023). For key2, it serves as the initial value for the Henon Map chaos function. With the presence of both key1 and key2, the key matrix can be formed. The following is the process of key matrix formation: (a) Obtain the size of the matrix (n) from key1. (b) Calculate the number of elements required to create an upper triangular matrix that will be populated within the size of an $n \times n$ matrix. (c) Generate matrix elements according to the required number of elements using the Henon Map chaos function with the initial value of key2. (d) Create an identity matrix of size $n \times n$. (e) Arrange the generated matrix elements into the identity matrix to form the upper

triangular matrix. (f) Perform elementary row operations on the upper triangular matrix to fill the lower diagonal portion of the matrix. (g) A unimodular key matrix is successfully formed (Sulaiman and Hanapi, 2021; Chauhdary *et al.*, 2022).

For decryption, the passkey is also required and serves the same purpose as in the encryption process. Decryption also utilizes key1 and key2, like the encryption process. The decryption process involves the use of the inverse of the key matrix. The following is the process of finding the inverse of the key matrix:

1. The key matrix is formed using the key matrix formation process from the encryption process.
2. With the matrix size (n) from key1, create a zero matrix of size $n \times 2n$.
3. With the newly created zero matrix, fill the left side of the matrix with the key matrix and the right side of the matrix with the identity matrix (Arifin *et al.*, 2024; Hamissa *et al.*, 2010). Let's assume n = 3, thus the created zero matrix will be 3×6:
$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$
Fill the zero matrix, with k_{ij} representing the key matrix element:
$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$
4. Perform elementary row operations to transform the left side matrix (filled with the key matrix) into an identity matrix.
5. Once the elementary row operations are completed, the left-side matrix will become an identity matrix and the right-side matrix will become the inverse of the key matrix (Fadlan *et al.*, 2020; Keserwani and Govil, 2020; Udayan *et al.*, 2024).

The modified Hill Cipher algorithm will be developed in the Python programming language and will require the NumPy and OS libraries. For a comprehensive explanation, the following outlines the workings of the encryption process of the modified Hill Cipher algorithm: (1) Receive the passkey and the text/digital file to be encrypted. (2) For text, convert each text character into its Unicode code and store it in a numpy array. For files, NumPy can directly convert file data into numbers stored in a Numpy array. (3) Calculate the size of the numpy array and ensure that the size is not odd. (4) Find a divisor factor that can evenly divide the size of the numpy array. (5) Determine a divisor factor that meets the criteria to become key1, which is used as the key matrix size. (6) Process the passkey into decimal form and use it as key2, which serves as the initial value for the Henon Map chaos function. (7) Create the key matrix. (8) Reshape the original 1-dimensional numpy array into a suitable shape for multiplication with the key matrix. (9) Multiply the key matrix with the reshaped numpy array. (10) Reshape the multiplication result back into a 1-dimensional form, which is also the encrypted result. (11) For text, reverse the encrypted numpy array numbers according to the Unicode characters represented by the values of the elements in the encrypted numpy array. For files, the encrypted numpy array will be directly converted back into an encrypted file. Below is the working process of the decryption process of the modified Hill Cipher algorithm: (i) Receive the passkey and the text/digital file to be decrypted. (ii) For text, convert each text character into its Unicode code and store it in a numpy array. For files, numpy can directly convert file data into numbers stored in a numpy array. (iii) Calculate the size of the numpy array and ensure that the size is not odd. (iv) Find a divisor factor that can evenly divide the size of the numpy array. (v) Determine a divisor factor that meets the criteria to become key1, which is used as the key matrix size. (vi) Process the passkey into decimal form and use it as key2, which serves as the initial value for the Henon Map chaos function. (vii) Find the inverse of the key matrix. (viii) Reshape the original 1-dimensional numpy array into a suitable shape for multiplication with the inverse key matrix. (ix) Multiply the inverse key matrix with the reshaped numpy array. (x) Reshape the multiplication result back into a 1-dimensional form, which is also the decrypted result. (xi) For text, reverse the decrypted numpy array numbers according to the Unicode characters represented by the values of the elements in the decrypted numpy array. For files, the decrypted numpy array will be directly converted back into a decrypted file (Arifin, 2023; Indriani *et al.*, 2020; Pine, 2024).

The modified Hill Cipher offers significant advantages in scenarios requiring lighter computational loads, such as Internet of Things (IoT) applications, where resource constraints are a critical consideration. Its design ensures efficient encryption and decryption processes, making it suitable for devices with limited processing power. Unlike RSA, which is optimized for asymmetric encryption and excels in scenarios demanding higher levels of security, the modified Hill Cipher provides a balanced approach by leveraging the simplicity of symmetric encryption while maintaining adequate protection. A key feature of the modified Hill Cipher is its use of the Henon Map chaos function, which generates a larger and more unpredictable keyspace. This enhancement not only increases security against cryptanalysis but also enables the algorithm to achieve comparable levels of protection to advanced cryptographic techniques like AES in specific applications. As a result, the modified Hill Cipher is highly effective for secure communications in

environments requiring both efficiency and reliability (Arifin *et al.*, 2024; Marquez *et al.*, 2024; Landau *et al.*, 2024).

Results and Discussion

This results and discussion section covers five major concepts, namely Output Visualization, which displays the encryption and decryption results in a visual form to make it easier to understand; Estimated Time Comparison, which compares the time required between the modified Hill Cipher algorithm and the standard version; Passkey Strength, which evaluates the strength of the passkey used in the encryption process; Histogram Comparison of File Binary, which compares the distribution of file binaries before and after encryption to measure the changes that occur; and Data Sender Application, which demonstrates the implementation of the modified Hill Cipher algorithm in a data sending application (Arifin *et al.*, 2022-2023; Arifin, 2023; Safitri *et al.*, 2023).

Here is a simple example. Using the modified Hill Cipher, a plaintext such as 'HELLO' is first converted into numerical values based on its Unicode representation. For instance, the characters 'H', 'E', 'L', 'L' and 'O' correspond to the numerical values [72, 69, 76, 76, 79] respectively. These numerical values are then arranged into a matrix, preparing them for encryption. This preprocessing step ensures that the plaintext is in a suitable format for matrix operations. Next, the plaintext matrix is multiplied by the key matrix, which is generated using the Henon Map chaos function to ensure randomness and security. The resulting ciphertext matrix contains transformed numerical values, representing the encrypted text. This ciphertext is then converted back into an encoded textual format, producing an unreadable output that secures the original message. This process highlights the effectiveness of the modified Hill Cipher in encrypting text. In this section, the encryption and decryption processes will be demonstrated from start to finish. A paragraph will be used as the plaintext (Reddy *et al.*, 2012; Toorani and Falahati, 2011; Muttoo *et al.*, 2011). The plaintext used:

> *"In the Pride Lands of Africa, a pride of lion's rule over the kingdom from Pride Rock. King Mufasa and Queen Sarabi's newborn son, Simba, is presented to the gathering animals by Rafiki the mandrill, the kingdom's shaman and advisor. Mufasa's younger brother, Scar, covets the throne. After Simba grows into a cub, Mufasa shows him the Pride Lands and explains the responsibilities of kingship and the 'circle of life,' which connects all living things. One day, Simba and his best friend Nala explore an elephant graveyard, where the two are chased by three spotted hyenas named Shenzi, Banzai and Ed. Mufasa is alerted by his majordomo, the hornbill Zazu and rescues the cubs. Though disappointed with Simba for disobeying him and putting himself and Nala in danger, Mufasa forgives him and explains that the great kings of the past watch over them from the night sky, from which he will one day watch over Simba. Scar, having planned the attack, visited the hyenas and convinced them to help him kill both Mufasa and Simba in exchange for hunting rights in the Pride Lands. Scar sets a trap for Simba and Mufasa, luring Simba into a gorge and having the hyenas drive a large herd of wildebeests into a stampede to trample him. Mufasa saves Simba but winds up hanging perilously from the gorge's edge; he begs for Scar's help, but Scar throws Mufasa back into the stampede to his death. Scar tricks Simba into believing that the event was his fault and tells him to leave the kingdom and never return. Once Simba flees, Scar orders the hyenas to kill Simba, who manages to escape. Unaware of Simba's survival, Scar tells the pride that the stampede killed both Mufasa and Simba and steps forward as the new king, allowing the hyenas into the Pride Lands. After he collapses in a desert, Simba is rescued by two outcasts, a meerkat and warthog named Timon and Pumbaa. Simba grows up with his two new friends in their oasis, living a carefree life under their motto 'hakuna matata' ('no worries' in Swahili). Years later, an adult Simba rescues Timon and Pumbaa from a hungry lioness, who turns out to be Nala. Simba and Nala fall in love and she urges him to return home, telling him that the Pride Lands have become drought-stricken under Scar's reign. Still feeling guilty over Mufasa's death, Simba refuses and storms off. He encounters Rafiki, who tells Simba that Mufasa's spirit lives on in him. Simba is visited by the spirit of Mufasa in the night sky, who tells him that he must take his place as king. After Rafiki advises him to learn from the past instead of running from it, Simba decides to return to the Pride Lands."*
> *(https://en.wikipedia.org/wiki/The_Lion_King)*

Below is the process of encrypting the plaintext:

Plaintext Size: 2632
Plaintext Unicode: $[73, 110, 32, \ldots, 100, 115, 46]$

Divisor factor of the plaintext size (up to the first 3 digits):
$[1, 2, 4, 7, 8, 14, 28, 47, 56, 94, 188]$

Passkey: qu8s9201iw
key2 derived from the processed passkey: 0.936789699446426
key1 from the chosen factor: 188

Random sequence from the Henon Map chaos function:

$$\begin{bmatrix} 12068, 20191, 3485, 21665, 1106458, 7631, 9352, 19679, 3516, \\ 21658, 1106546, 7815, 9188, 19826, 3054, 21462, 1106563, 7866, \\ 9146, 19863, 2935, 21406, 1106582, 7909, 9108, 19895, 2831, \\ 21355, 1106603, 7956, 9066, 19931, 2714, 21296, 1106631, 8020, \\ 9008, 19979, 2556, 21213, 1106679, 8124, 8914, 20057, 2299, \\ 21069, 1106778, 8337, 8717, 20211, \ldots \end{bmatrix}$$

Identity Matrix:

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 1 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix}$$

Upper Triangular Matrix:

$$\begin{bmatrix} 1 & 12068 & 20191 & \cdots & 1112336 & 17481 & 1111558 \\ 0 & 1 & 16506 & \cdots & 1109971 & 14081 & 2038 \\ 0 & 0 & 1 & \cdots & 10393 & 6600 & 21329 \\ 0 & 0 & 0 & \cdots & 1 & 14617 & 1409 \\ 0 & 0 & 0 & \cdots & 0 & 1 & 20399 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix}$$

The key matrix is filled entirely through row elementary operations:

$$\begin{bmatrix} 1 & 12068 & 20191 & \cdots & 1112336 & 17481 & 1111558 \\ 1107967 & 499646 & 743434 & \cdots & 874461 & 679584 & 90116 \\ 10703 & 1041039 & 1080851 & \cdots & 1066566 & 1049206 & 549345 \\ 1108316 & 254933 & 1088921 & \cdots & 259127 & 96323 & 312601 \\ 11274 & 133090 & 354690 & \cdots & 42648 & 997259 & 204763 \\ 5592 & 637596 & 382861 & \cdots & 101199 & 826095 & 207068 \end{bmatrix}$$

Reshape the numpy array plaintext to enable multiplication with the key matrix:

$$\begin{bmatrix} 73 & 110 & 32 & \cdots & 101 & 32 & 76 \\ 97 & 110 & 100 & \cdots & 105 & 99 & 97 \\ 44 & 32 & 97 & \cdots & 102 & 32 & 108 \\ 105 & 109 & 98 & \cdots & 115 & 32 & 116 \\ 111 & 32 & 114 & \cdots & 32 & 116 & 104 \\ 101 & 32 & 80 & \cdots & 100 & 115 & 46 \end{bmatrix}$$

Numpy array ciphertext (result of multiplying the plaintext matrix with the key matrix):

$$\begin{bmatrix} 379932 & 184740 & 433078 & \cdots & 737930 & 672466 & 1076590 \\ 598560 & 28865 & 846924 & \cdots & 1095604 & 236498 & 1084943 \\ 515426 & 809454 & 423317 & \cdots & 95155 & 423248 & 934161 \\ 924447 & 350360 & 918287 & \cdots & 929333 & 1066408 & 269060 \\ 1071224 & 1077438 & 725097 & \cdots & 485576 & 265219 & 78996 \\ 256538 & 249439 & 167318 & \cdots & 548029 & 82481 & 484719 \end{bmatrix}$$

Reshape the numpy array ciphertext into 1 dimension:

$$\begin{bmatrix} 379932, & 184740, & 433078, & \ldots, & 548029, & 82481, & 484719 \end{bmatrix}$$

Below is the process of decryption:

Ciphertext Size: 2632

Ciphertext Unicode:

$$\begin{bmatrix} 379932, & 184740, & 433078, & \ldots, & 548029, & 82481, & 484719 \end{bmatrix}$$

Divisor factor of the plaintext size (up to the first 3 digits):

$$\begin{bmatrix} 1, & 2, & 4, & 7, & 8, & 14, & 28, & 47, & 56, & 94, & 188 \end{bmatrix}$$

Passkey: qu8s9201iw
key2 derived from the processed passkey: 0.936789699446426
key1 from the chosen factor: 188

Random sequence from the Henon Map chaos function:

$$\begin{bmatrix} 12068, 20191, 3485, 21665, 1106458, 7631, 9352, 19679, 3516, \\ 21658, 1106546, 7815, 9188, 19826, 3054, 21462, 1106563, 7866, \\ 9146, 19863, 2935, 21406, 1106582, 7909, 9108, 19895, 2831, \\ 21355, 1106603, 7956, 9066, 19931, 2714, 21296, 1106631, 8020, \\ 9008, 19979, 2556, 21213, 1106679, 8124, 8914, 20057, 2299, \\ 21069, 1106778, 8337, 8717, 20211, \ldots \end{bmatrix}$$

Identity Matrix:

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 1 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix}$$

Upper Triangular Matrix:

$$\begin{bmatrix} 1 & 12068 & 20191 & \cdots & 1112336 & 17481 & 1111558 \\ 0 & 1 & 16506 & \cdots & 1109971 & 14081 & 2038 \\ 0 & 0 & 1 & \cdots & 10393 & 6600 & 21329 \\ 0 & 0 & 0 & \cdots & 1 & 14617 & 1409 \\ 0 & 0 & 0 & \cdots & 0 & 1 & 20399 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix}$$

The key matrix is filled entirely through row elementary operations:

$$\begin{bmatrix} 1 & 12068 & 20191 & \cdots & 1112336 & 17481 & 1111558 \\ 1107967 & 499646 & 743434 & \cdots & 874461 & 679584 & 90116 \\ 10703 & 1041039 & 1080851 & \cdots & 1066566 & 1049206 & 549345 \\ 1108316 & 254933 & 1088921 & \cdots & 259127 & 96323 & 312601 \\ 11274 & 133090 & 354690 & \cdots & 42648 & 997259 & 204763 \\ 5592 & 637596 & 382861 & \cdots & 101199 & 826095 & 207068 \end{bmatrix}$$

Create an empty matrix with size n × 2n:

$$\begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 \end{bmatrix}$$

Fill left-hand side with key matrix and right-hand side with identity matrix:

$$\begin{bmatrix} 1 & 12068 & 20191 & \cdots & 0 & 0 & 0 \\ 1107967 & 499646 & 743434 & \cdots & 0 & 0 & 0 \\ 10703 & 1041039 & 1080851 & \cdots & 0 & 0 & 0 \\ 1108316 & 254933 & 1088921 & \cdots & 1 & 0 & 0 \\ 11274 & 133090 & 354690 & \cdots & 0 & 1 & 0 \\ 5592 & 637596 & 382861 & \cdots & 0 & 0 & 1 \end{bmatrix}$$

Turn lower left triangle into zero (row ops):

$$\begin{bmatrix} 1 & 12068 & 20191 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 16506 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 1 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix}$$

Turn upper right triangle into zero (row ops):

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 455740 & 188693 & 335173 \\ 0 & 1 & 0 & \cdots & 468578 & 1012345 & 147894 \\ 0 & 0 & 1 & \cdots & 1058564 & 498861 & 660879 \\ 0 & 0 & 0 & \cdots & 1 & 1099494 & 703137 \\ 0 & 0 & 0 & \cdots & 0 & 1 & 1093712 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix}$$

The right-hand side matrix is the inverse of the key matrix:

$$\begin{bmatrix} 880754 & 1102043 & 862459 & \cdots & 455740 & 188693 & 335173 \\ 333976 & 1 & 1097605 & \cdots & 468578 & 1012345 & 147894 \\ 619542 & 0 & 1 & \cdots & 1058564 & 498861 & 660879 \\ 779151 & 0 & 0 & \cdots & 1 & 1099494 & 703137 \\ 420612 & 0 & 0 & \cdots & 0 & 1 & 1093712 \\ 1108519 & 0 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix}$$

Reshape ciphertext matrix to enable multiplication with inverse matrix:

$$\begin{bmatrix} 379932 & 184740 & 433078 & \cdots & 737930 & 672466 & 1076590 \\ 598560 & 28865 & 846924 & \cdots & 1095604 & 236498 & 1084943 \\ 515426 & 809454 & 423317 & \cdots & 95155 & 423248 & 934161 \\ 924447 & 350360 & 918287 & \cdots & 929333 & 1066408 & 269060 \\ 1071224 & 1077438 & 725097 & \cdots & 485576 & 265219 & 78996 \\ 256538 & 249439 & 167318 & \cdots & 548029 & 82481 & 484719 \end{bmatrix}$$

Plaintext matrix (after multiplication with inverse matrix):

$$\begin{bmatrix} 73 & 110 & 32 & \cdots & 101 & 32 & 76 \\ 97 & 110 & 100 & \cdots & 105 & 99 & 97 \\ 44 & 32 & 97 & \cdots & 102 & 32 & 108 \\ 105 & 109 & 98 & \cdots & 115 & 32 & 116 \\ 111 & 32 & 114 & \cdots & 32 & 116 & 104 \\ 101 & 32 & 80 & \cdots & 100 & 115 & 46 \end{bmatrix}$$

Reshape plaintext matrix into 1-dimensional numpy array:

$$\begin{bmatrix} 73, & 110, & 32, & \ldots, & 100, & 115, & 46 \end{bmatrix}$$

Figure (3) presents the result of the encryption process, illustrating how the original plaintext data has been successfully transformed into an unreadable ciphertext format using the applied cryptographic algorithm. This transformation ensures that the information is securely protected from unauthorized

access, as the encrypted data cannot be understood without the corresponding decryption key. The figure visually demonstrates the effectiveness of the encryption method in safeguarding data confidentiality, which is a crucial aspect in maintaining information security in digital communication systems (Ibrahim *et al.*, 2022; Tarigan *et al.*, 2021; Arifin *et al.*, 2022).

Fig. 3: Encryption result

In this section, a comparative analysis of encryption and decryption times will be conducted, specifically focusing on digital files, as digital files generally have larger sizes compared to text. The digital files used for comparison will be audio files with sizes ranging from 1-25 megabytes. The 25-megabyte limit is based on the attachment size limit of Gmail. The variations in encryption and decryption times for different file sizes are influenced by the matrix size, which is determined by the algorithm's divisor factor selection criteria. The algorithm selects matrix orders based on file size, ensuring compatibility with the encryption process while maintaining optimal security. However, this approach occasionally results in fluctuations in computational complexity, as seen in the decryption time for a 25 MB file, where a higher matrix order significantly increases processing time. This anomaly reflects the deterministic nature of the key generation process, where security and efficiency are carefully balanced. The use of larger matrix orders enhances the randomness and strength of the encryption, providing greater protection against cryptographic attacks. While this trade-off leads to slightly increased decryption times for certain file sizes, the algorithm remains efficient and suitable for practical applications (Toorani and Falahati, 2011; Hamissa *et al.*, 2011; Praveenkumar *et al.*, 2017). Table (1) presents the comparison of encryption and decryption times for digital files.

Based on Table (1), the encryption and decryption times increase as the size of the digital file increases, but these times are still relatively fast. The encryption and decryption times in the modified Hill Cipher algorithm depend on the size of the key matrix, which is also influenced by the selected divisor factor based on the criteria outlined in section 2.5. It can be observed that for a file size of 25 megabytes, the decryption time is longer compared to a digital file size of 50 megabytes. This is because the smallest selected factor in the hundreds range is 389, meaning that the required matrix size is 389x389, leading to a relatively longer processing time.

Table 1: Estimated time comparison

File Size	Encryption Time (in seconds)	Decryption Time (in seconds)
1 megabyte	0.07481670379638672	0.08078384399414062
5 megabytes	0.29917144775390625	0.5396807193756104
10 megabytes	0.5571625232696533	0.5764575004577637
15 megabytes	0.8887159824371338	1.5647499561309814
20 megabytes	1.2621512413024902	1.4089264869689941
25 megabytes	1.8236854076385498	6.56545615196228
50 megabytes	3.6591148376464844	3.5725536346435547

The passkey used consists of 10 characters comprising a combination of 26 lowercase alphabets, 26 uppercase alphabets and 10 numeric digits from 0 to 9. This implies that the used passkey has a combination of $(26+26+10)^{10}$, which is 62^{10} possible passkey combinations. The strength of the passkey is evaluated using the brute-force method. Brute-force attempts were also made based on the passkey criteria. With the computational power of the system used by the author, a brute-force attempt involving 100,000 passkey combinations takes approximately 754 sec or 12 min and 34 sec. Based on the obtained computation time, an approximation of the time (in years) required to compute all 62^{10} passkey combinations can be calculated:

$$\frac{62^{10}/(100{,}000/754)}{60\times60\times24\times365}$$

The calculation yields a result of approximately 200,669,622.6 years, indicating that the passkey is highly secure against brute-force attacks. The encryption of digital files is carried out by transforming the binary data of the digital file, making the resulting encrypted file unable to be opened. To visualize the changes, histograms will be created for both the original and encrypted files to see if there are any differences between the two histograms. This experiment will be conducted on four digital files with different extensions: .jpg, .mp3, .mp4, .pdf.

Figures (4) through 7 present binary value histograms for various file formats, illustrating the distribution of byte-level data across different types of digital content. Figure (4) displays the histogram for a JPG file, characterized by distinct patterns resulting from image compression and encoding schemes. Figure (5) shows the histogram for an MP3 file, highlighting the unique

binary structure shaped by audio compression techniques. Figure (6) depicts the binary distribution of an MP4 file, where the data layout reflects the complexity of multimedia encapsulation involving both audio and video streams. Lastly, Figure (7) illustrates the binary value histogram for a PDF file, showcasing a distribution influenced by document structure, embedded objects and text encoding. These histograms provide visual insight into the inherent structural differences among file types at the binary level (Porter and Zingaro, 2024; Ansari and Bajaj, 2024). The following are the histograms resulting from the experiment on the four files:

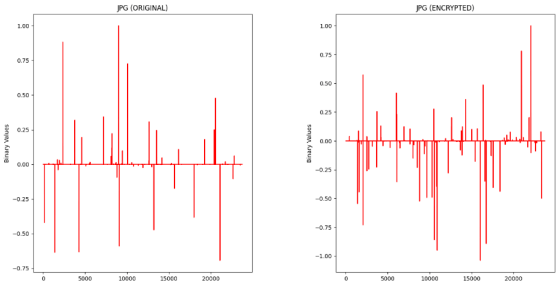

Fig. 4: JPG binary value histogram

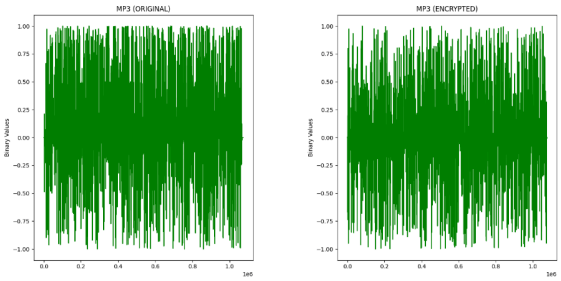

Fig. 5: MP3 binary value histogram

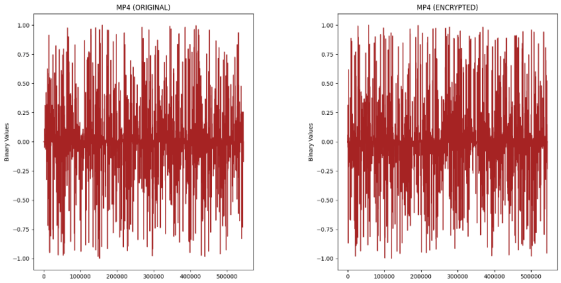

Fig. 6: MP4 binary value histogram

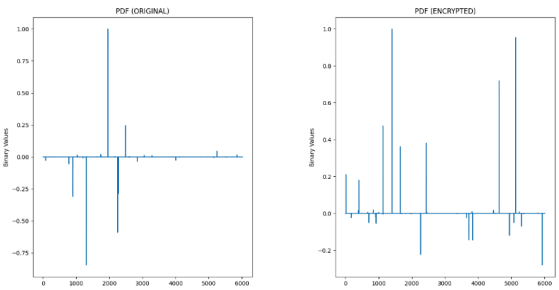

Fig. 7: PDF binary value histogram

The author developed a web-based data sender application using the Next.js framework. The modified Hill Cipher algorithm will be implemented into the data sender application in the form of an API. The application consists of two pages: The first is the encryption page, which also serves as the home page and the second is the decryption page. On the encryption page, users will be prompted to enter a passkey or generate one randomly by clicking the “Generate Passkey” button. Users can then choose between two modes: Write Text and Upload File and input the text/file to be encrypted. Subsequently, users will be asked to provide the recipient's email and phone number. Once all inputs are filled, users can press the “Encrypt” button. The data sender application will call the API with endpoints “/encrypt” or “/encrypt-file” depending on the chosen mode and perform the encryption. The encrypted result will be sent to the recipient's email and the passkey will be sent to the recipient's phone number. On the decryption page, users will also be asked to enter the passkey sent to their phone number. Then users will be prompted to choose a mode and input the encrypted text/file. After that, users can directly press the decryption button. The data sender application will call the API with endpoints “/decrypt” or “/decrypt-file” based on the selected mode. The API endpoint will return the decryption result to the user.

This algorithm is particularly well-suited for Internet of Things (IoT) devices, as its low computational overhead ensures efficient performance on devices with limited processing power. By minimizing resource requirements, it enables secure communication in IoT networks without significantly impacting device functionality or battery life, making it an ideal choice for lightweight encryption needs. In addition to IoT applications, the algorithm is highly adaptable for use in cloud storage systems to ensure secure file sharing. Its versatility also extends to mobile applications, where it can be utilized for encrypted messaging, offering robust security for sensitive communications. This flexibility highlights its potential to function effectively across a wide range of technological ecosystems.

In the program we developed, the graphical user interface (GUI) is designed to be as user-friendly as possible, ensuring that users can easily navigate and operate the available features without requiring advanced technical knowledge. This intention is clearly reflected in the intuitive and responsive design of the Encrypt and Decrypt pages, as shown in Figs. (8-9). These interface layouts not only simplify the encryption and decryption processes but also provide a more comfortable and efficient user experience overall.

Below is the procedure for using the data sender application on the encryption page: (1) Users/Senders are asked to input the passkey, or they can generate one randomly by pressing the Generate Passkey button. (2) Users/Senders are prompted to choose between two modes: Write Text and Upload File. (3) If users select

Write Text, a textbox will appear and if users choose Upload File, an upload button will appear. (4) Then users/senders are asked to provide the recipient's email and phone number. (5) Users/Senders can press the Send button after filling in all the data and the application will initiate the encryption process. The result will be sent to the recipient's email and the passkey will be sent to the recipient's phone number. Below is the procedure for using the data sender application on the decryption page: (i) Users/Recipients are asked to input the passkey sent to their phone number. (ii) Users are prompted to choose between two modes: Write Text and Upload File. (iii) If users/recipients select Write Text, a textbox will appear and if users choose Upload File, an upload button will appear. (iv) Users/recipients can press the decrypt button and the application will initiate the decryption process. If the chosen mode is Write Text, the decryption result will appear in the application. If the chosen mode is Upload File, the decryption result will be downloaded directly to the user/recipient's computer.

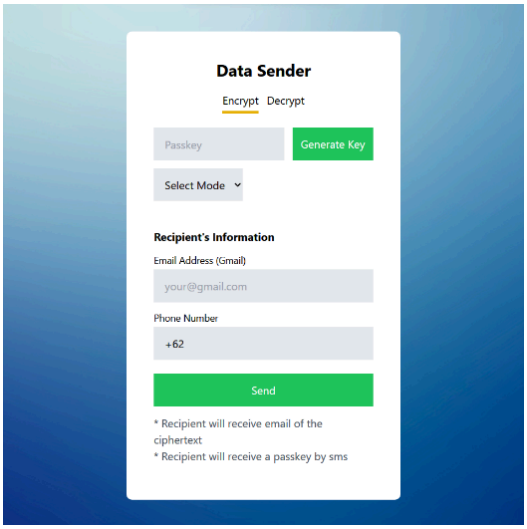

Fig. 8: Encrypt page

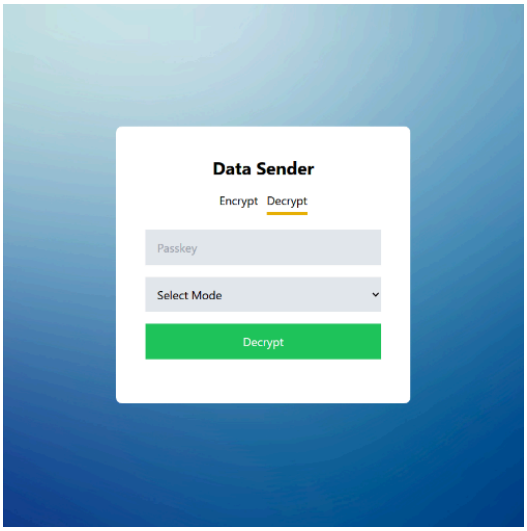

Fig. 9: Decrypt page

The incorporation of chaos functions and unimodular matrices enhances the security of the modified Hill Cipher by significantly increasing the randomness and unpredictability of the key. The chaos function, such as the Henon Map, generates a highly non-linear and complex sequence of numbers, making the key matrix resistant to patterns that could be exploited in cryptanalysis. Additionally, the use of unimodular matrices ensures the invertibility of the key matrix, further strengthening its cryptographic robustness. Compared to the classical Hill Cipher, these modifications effectively mitigate vulnerabilities to known plaintext attacks. By expanding the keyspace and introducing chaotic elements, the algorithm makes it considerably more difficult for attackers to deduce the encryption key, even if they possess plaintext-ciphertext pairs. This enhancement also reduces the feasibility of brute-force methods, as the increased complexity of the key matrix demands exponentially higher computational resources to crack.

The proposed algorithm employs modular arithmetic and chaotic functions, making it highly compatible with modern computational capabilities. This compatibility allows it to take advantage of GPUs and parallel processing, significantly enhancing the efficiency of the encryption process, even for large datasets or real-time applications. Furthermore, the algorithm's adaptability to diverse platforms, such as IoT and cloud-based systems, highlights its robustness in addressing emerging technological advancements. Its lightweight computational requirements and scalability make it particularly well-suited for resource-constrained environments, ensuring secure and efficient data encryption across various modern technologies.

Conclusion

The modified Hill Cipher algorithm has been successfully implemented in the data sender application and can effectively protect information with reliable encryption and accurate decryption. A comparison of encryption time can be observed by comparing it with research conducted where that study used the Strassen algorithm to optimize the computational time required for Hill Cipher. According to that research, matrix multiplication with N order = 32 took 18.7068 seconds, while the modified Hill Cipher algorithm used in this study requires less than 1 second for matrix multiplication with N order >100 (due to the matrix size always being around 100). The addition of the Henon Map chaos function has been successfully incorporated, generating a sequence of random elements for use in key matrix formation. The constructed key matrix is unimodular, achieved through row operations and contains random elements generated by the Henon Map chaos function. The modified Hill Cipher can be further extended by incorporating other techniques, such as combining it with other cryptographic algorithms to add additional layers of security or utilizing different

mathematical functions to enhance the strength of various aspects of cryptographic security. This algorithm can also be implemented in various applications or devices to test its compatibility and security level.

The lightweight nature of the modified Hill Cipher enables seamless integration into IoT devices with limited processing power, ensuring secure communication between connected systems. This feature is particularly advantageous for IoT ecosystems, where efficient encryption is critical to maintaining data integrity without overburdening the hardware. By leveraging its computational efficiency, the algorithm enhances the security of real-time data transmission in resource-constrained environments. In addition to IoT, the modified Hill Cipher demonstrates strong adaptability for encrypting large datasets, making it an excellent choice for mobile and cloud-based applications. Its ability to balance efficiency and security allows it to handle the demands of modern data-intensive platforms, such as secure file sharing and encrypted messaging. This versatility underscores its potential as a robust cryptographic solution across various technological ecosystems.

Acknowledgment

The authors sincerely appreciate the reviewers for their valuable feedback, constructive suggestions and insightful ideas, all of which have significantly enhanced the quality of this manuscript and contributed to its readiness for publication. We are also grateful that this research has finally reached completion, marking an important milestone in our academic journey. Their commitment to the advancement of research in data science and cryptography is truly commendable and greatly acknowledged.

Funding Information

This research was generously supported and funded by the Lembaga Penelitian dan Pengabdian kepada Masyarakat (LPPM) of Institut Teknologi Sains Bandung (ITSB). The support provided by LPPM ITSB played a vital role in facilitating various stages of this study, from initial planning and data collection to implementation and final analysis. Their commitment to fostering academic research and community engagement has been instrumental in enabling the successful completion of this project. The authors are deeply grateful for the trust and resources extended throughout the research process.

Author's Contributions

Samsul Arifin: As the lead author, was responsible for the primary concept of this research, including the design of the modified Hill Cipher algorithm incorporating the Henon Map chaos function. He also led the development of the methodology, result analysis and the overall manuscript writing. His role laid the foundation for the success and completion of this study.

Kevin Tan: Contributed to the technical implementation of the algorithm in a data transmission application, integrating it with frameworks such as Next.js. He also participated in testing and validating the algorithm to ensure its encryption effectiveness.

Felix Indra Kumadi: Focused on the mathematical computation required to generate unimodular matrices used in the algorithm. He also made significant contributions to the literature review on cryptography and chaos theory and was involved in revising the manuscript to maintain its academic quality.

Muhammad Faisal: Was involved in designing the system architecture and the security model for the data sender application. He also helped evaluate the robustness of the encryption through simulation-based analysis and contributed to proofreading the manuscript.

Amril Mutoi Siregar: Contributed to the development and implementation of the backend infrastructure that supports cryptographic operations. Additionally, he assisted in preparing visual illustrations and algorithm flowcharts for the manuscript.

Edwin Kristianto Siaibat: Supported the integration of the modified Hill Cipher algorithm into practical use cases, particularly in secure communication modules. He also contributed to the literature review and identifying recent threats in data transmission security.

Dwi Wijonarko: Played a key role in testing and validating the modified algorithm using various datasets to ensure consistent performance. He also worked on documentation and code optimization for deployment purposes.

Puguh Wahyu Prasetyo: Provided mathematical insight into unimodular matrix generation and supported the theoretical analysis of the Henon Map chaos function. He contributed to ensuring the mathematical accuracy and theoretical soundness presented in the paper.

Ethics

This manuscript presents original research findings that have not been published or disseminated elsewhere. The work contributes new knowledge to the field and has been conducted with strict adherence to ethical standards. The corresponding author affirms that there are no conflicts of interest, financial or otherwise, that could have influenced the outcomes or interpretation of the study. Furthermore, the research complies fully with all relevant ethical guidelines, ensuring that no ethical concerns arise from the study's design, implementation, or reporting. This transparency underscores the integrity and credibility of the research presented.

References

Abd-Elmonim, W. G., Ghali, N. I., Hassanien, A. E., & Abraham, A. (2011). Known-plaintext attack of DES-16 using particle swarm optimization. *IEEE*, 12-16. https://doi.org/10.1109/NaBIC.2011.6089410

Abdillah, A. A., Azwardi, A., Permana, S., Susanto, I., Zainuri, F., & Arifin, S. (2021). Performance evaluation of linear discriminant analysis and support vector machines to classify cesarean section. *Eastern-European Journal of Enterprise Technologies*, *5*(2 (113)), 37-43. https://doi.org/10.15587/1729-4061.2021.242798

Ahmed, M., Abdul-kader, H. S., Kishk, A., & Abdo, A. A. (2020). An Efficient Multi Secret Image Sharing Scheme Using Hill Cipher. In *Proceedings of the International Conference on Artificial Intelligence and Computer Vision (AICV2020)* (Vol. 1153, pp. 604-613). Springer International Publishing. https://doi.org/10.1007/978-3-030-44289-7_57

Ansari, I. A., & Bajaj, V. (Eds.). (2024). *Image Processing with Python: A practical approach*. https://doi.org/10.1088/978-0-7503-5924-5

Arifin, S., Bayu Muktyas, I., & Iswara Sukmawati, K. (2021). Product of two groups integers modulo m,n and their factor groups using python. *Journal of Physics: Conference Series*, *1778*(1), 012026. https://doi.org/10.1088/1742-6596/1778/1/012026

Arifin, S., Kurniadi, F. I., Yudistira, I. G. A., Nariswari, R., Murnaka, N. P., & Muktyas, I. B. (2022). Image Encryption Algorithm Through Hill Cipher, Shift 128 Cipher, and Logistic Map Using Python. *2022 3rd International Conference on Artificial Intelligence and Data Sciences (AiDAS)*, 221-226. https://doi.org/10.1109/aidas56890.2022.9918696

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). Generate a system of linear equation through. *AIP Conference Proceedings*, 020005. https://doi.org/10.1063/5.0041651

Arifin, S., Muktyas, I. B., Al Maki, W. F., & Mohd Aziz, M. K. B. (2022). Graph Coloring Program of Exam Scheduling Modeling Based on Bitwise Coloring Algorithm Using Python. *Journal of Computer Science*, *18*(1), 26-32. https://doi.org/10.3844/jcssp.2022.26.32

Arifin, S., Muktyas, I. B., Prasetyo, P. W., & Abdillah, A. A. (2021). 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., Nicholas, A., Baskoroputro, H., Prabowo, A. S., Ibrahim, M. A., & Rahayu, A. (2023). Algorithm for Digital Image Encryption Using Multiple Hill Ciphers, a Unimodular Matrix and a Logistic Map. *International Journal of Intelligent Systems and Applications in Engineering*, *11*(6s), 311-324 2858.

Arifin, S., Tan, K., Ariani, A. T., Rosdiana, S., & Abdullah, M. N. (2023). The Audio Encryption Approach uses a Unimodular Matrix and a Logistic Function. *International Journal of Emerging Technology and Advanced Engineering*, *13*(4), 71-81. https://doi.org/10.46338/ijetae0423_08

Arifin, S., Wijonarko, D., Suwarno, & Sijabat, E. K. (2024). Application of Unimodular Hill Cipher and RSA Methods to Text Encryption Algorithms Using Python. *Journal of Computer Science*, *20*(5), 548-563. https://doi.org/10.3844/jcssp.2024.548.563

Awati, R., Bernstein, C., & Cobb, M. (2024). Advanced Encryption Standard (AES). *TechTarget*. https://www.techtarget.com/searchsecurity/definition/Advanced-Encryption-Standard

Azanuddin, A., Kartadie, R., Erwis, F., Boy, A. F., & Nasyuha, A. H. (2024). A Combination of Hill Cipher and RC4 Methods for Text Security. *Telkomnika (Telecommunication Computing Electronics and Control)*, *22*(2), 351-361. https://doi.org/10.12928/telkomnika.v22i2.25628

Chauhdary, S. H., Alkatheiri, M. S., Alqarni, M. A., & Saleem, S. (2022). (Retracted) Improved encrypted AI robot for package recognition in IoT logistics environment. *Journal of Electronic Imaging*, *31*(06), 061813. https://doi.org/10.1117/1.jei.31.6.061813

D, M., R, B. G., R, V. S., & K, C. (2023). Performance And Security Enhanced Improved Hill Cipher. *2023 Fifth International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, 1-5. https://doi.org/10.1109/icecct56650.2023.10179696

DeBonis, M. J. (2022). *Introduction to Linear Algebra: Computation, Application, and Theory*. https://doi.org/10.1201/9781003217794

Delfs, H., & Knebl, H. (2015). *Introduction to Cryptography: Principles and Applications*. *53*. https://doi.org/10.1007/978-3-662-47974-2

Fadlan, M., Suprianto, Muhammad, & Amaliah, Y. (2020). Double Layered Text Encryption using Beaufort and Hill Cipher Techniques. *2020 Fifth International Conference on Informatics and Computing (ICIC)*. 2020 Fifth International Conference on Informatics and Computing (ICIC), Gorontalo, Indonesia. https://doi.org/10.1109/icic50835.2020.9288538

Hamissa, G., Sarhan, A., Abdelkader, H., & Fahmy, M. (2011). Securing JPEG architecture based on enhanced chaotic hill cipher algorithm. *The 2011 International Conference on Computer Engineering & Systems*, 260-266. https://doi.org/10.1109/icces.2011.6141053

Hamissa, G., Sarhan, A., Elkader, H. A., & Fahmy, M. (2010). Development of secure encoder-decoder for JPEG images. *The 2010 International Conference on Computer Engineering & Systems*, 189-194. https://doi.org/10.1109/icces.2010.5674851

Ibrahim, M. A., Arifin , S., Yudistira , I. G. A. A., Nariswari , R., Abdillah , A. A., Murnaka , N. P., & Prasetyo, P. W. (2022). An Explainable AI Model for Hate Speech Detection on Indonesian Twitter. *CommIT (Communication and Information Technology) Journal,16*(2), 175-182. https://doi.org/10.21512/commit.v16i2.8343

Indriani, U., Gunawan, H., Yugo Nugroho Harahap, A., & Zaharani, H. (2020). Chat Message Security Enhancement on WLAN Network Using Hill Cipher Method. *2020 8th International Conference on Cyber and IT Service Management (CITSM)*, 1-5. https://doi.org/10.1109/citsm50537.2020.9268838

Ismail, I. A., Amin, M., & Diab, H. (2006). How to repair the Hill cipher. *Journal of Zhejiang University-SCIENCE A*, *7*(12), 2022-2030. https://doi.org/10.1631/jzus.2006.a2022

Jin, J., Wu, M., Ouyang, A., Li, K., & Chen, C. (2025). A novel dynamic hill cipher and its applications on medical IoT. *IEEE Internet of Things Journal*, *12*(10), 14297-14308. https://doi.org/10.1109/JIOT.2025.3525623

Kamalakannan, V., & Tamilselvan, S. (2015). Security Enhancement of Text Message Based on Matrix Approach Using Elliptical Curve Cryptosystem. *Procedia Materials Science*, *10*, 489-496. https://doi.org/10.1016/j.mspro.2015.06.086

Keserwani, P. K., & Govil, M. C. (2020). A Hybrid Symmetric Key Cryptography Method to Provide Secure Data Transmission. *Machine Learning, Image Processing, Network Security and Data Sciences*, 461-474. https://doi.org/10.1007/978-981-15-6318-8_38

Khalaf, A. A. M., El-Karim, M. S. A., & Hamed, H. F. A. (2016). A triple hill cipher algorithm proposed to increase the security of encrypted binary dataand its implementation using FPGA. *International Conference on Advanced Communication Technology, ICACT*, 752-759. https://doi.org/10.1109/ICACT.2016.7423615

Landau, R. H., Páez, M. J., & Bordeianu, Cristian C. (2024). *Computational physics: Problem solving with Python*.

Liew, K. J., & Nguyen, V. T. (2020). Hill Cipher Key Generation Using Skew-symmetric Matrix. *The 7th International Cryptology and Information Security Conference 2020*, 85-93.

Lutz, M. (2013). *Learning python: Powerful object-oriented programming*.

Mahmoud, A. Y., & Chefranov, A. G. (2010). Secure Hill cipher modifications and key exchange protocol. In *IEEE* (pp. 1-6). https://doi.org/10.1109/aqtr.2010.5520828

Marquez, G., Carder, S. L., Lucas, B. L., Morris, H. A., & Hearon, B. F. (2024). Outcomes of anatomic total shoulder arthroplasty revised to reverse shoulder arthroplasty in patients with contained central glenoid bone defects. *Seminars in Arthroplasty: JSES*, *34*(2), 442-450. https://doi.org/10.1053/j.sart.2024.01.010

Martelli, A., Ravenscroft, A. M., Holden, S., & McGuire, P. (2023). *Python in a Nutshell* (4th ed.). O'Reilly Media.

Muktyas, I. B., Sulistiawati, & Arifin, S. (2021). Digital Image Encryption Algorithm Through Unimodular Matrix and Logistic Map using Python. *AIP Conference Proceedings*, 020006. https://doi.org/10.1063/5.0041653

Muttoo, S. K., Aggarwal, D., & Ahuja, B. (2011). A Secure Image Encryption Algorithm Based on Hill Cipher System. *Bulletin of Electrical Engineering and Informatics*, *1*(1), 51-60. https://doi.org/10.11591/eei.v1i1.226

Paragas, J. R. (2020). An Enhanced Cryptographic Algorithm in Securing Healthcare Medical Records. *2020 Third International Conference on Vocational Education and Electrical Engineering (ICVEE)*, 1-6. https://doi.org/10.1109/icvee50212.2020.9243228

Paragas, J. R., Sison, A. M., & Medina, R. P. (2019a). A new variant of hill cipher algorithm using modified S-Box. *International Journal of Scientific & Technology Research*, *8*(10), 615-619,.

Paragas, J. R., Sison, A. M., & Medina, R. P. (2019b). Hill Cipher Modification: A Simplified Approach. *2019 IEEE 11th International Conference on Communication Software and Networks (ICCSN)*, 821-825. https://doi.org/10.1109/iccsn.2019.8905360

Petrenko, S. I., & Dehtiarova, N. V. (2024). *Updating the content and methods of teaching students Python programming*. 197-222. https://doi.org/10.30525/978-9934-26-504-4-10

Pine, D. J. (2024). *Introduction to Python for science and engineering*. https://doi.org/10.1201/9781032673950

Porter, L., & Zingaro, D. (2024). *Learn AI-Assisted Python Programming, Second Edition: With GitHub Copilot and ChatGPT*.

Praveenkumar, P., Amirtharajan, R., Thenmozhi, K., & Rayappan, J. B. B. (2017). Fusion of confusion and diffusion: a novel image encryption approach. *Telecommunication Systems*, *65*(1), 65-78. https://doi.org/10.1007/s11235-016-0212-0

Pribadi, B., Rosdiana, S., & Arifin, S. (2023). Digital forensics on facebook messenger application in an android smartphone based on NIST SP 800-101 R1 to reveal digital crime cases. *Procedia Computer Science*, *216*, 161-167. https://doi.org/10.1016/j.procs.2022.12.123

Rajput, S. K., & Nishchal, N. K. (2013). Known-plaintext attack on encryption domain independent optical asymmetric cryptosystem. *Optics Communications*, *309*, 231-235. https://doi.org/10.1016/j.optcom.2013.06.036

Reddy, K. A., Vishnuvardhan, B., Madhuviswanatham, & Krishna, A. V. N. (2012). A Modified Hill Cipher Based on Circulant Matrices. *Procedia Technology*, *4*, 114-118. https://doi.org/10.1016/j.protcy.2012.05.016

Rrghout, H., Kattass, M., Qobbi, Y., Benazzi, N., JarJar, A., & Benazzi, A. (2024). New image encryption approach using a dynamic-chaotic variant of Hill cipher in Z/4096Z. In *International Journal of Electrical and Computer Engineering (IJECE)* (Vol. 14, Issue 5, pp. 5330-5343). https://doi.org/10.11591/ijece.v14i5.pp5330-5343

Rubinstein-Salzedo, S. (2018). *Cryptography*. https://doi.org/10.1007/978-3-319-94818-8

Safitri, R., Prasetyo, P. W., Wijayanti, D. E., Arifin, S., Setyawan, F., & Repka, J. (2023). Text security by using a combination of the vigenere cipher and the rubik's cube method of size 4×4×4. *Al-Jabar: Jurnal Pendidikan Matematika*, *14*(2), 281-297. https://doi.org/10.24042/ajpm.v14i2.14276

Sulaiman, S., & Hanapi, Z. M. (2021). Extensive Analysis on Images Encryption using Hybrid Elliptic Curve Cryptosystem and Hill Cipher. *Journal of Computer Science*, *17*(3), 221-230. https://doi.org/10.3844/jcssp.2021.221.230

Tarigan, S., Murnaka, N. P., & Arifin, S. (2021). Development of teaching material in mathematics "Sapta Maino Education" on topics of plane geometry. *AIP Conference Proceedings*, 2331. https://doi.org/10.1063/5.0041650

Toorani, M., & Falahati, A. (2011). A secure cryptosystem based on affine transformation. *Security and Communication Networks*, *4*(2), 207-215. https://doi.org/10.1002/sec.137

Udayan, D., Aubrey, L., Chris, M., & Narges, N. (2024). *Introduction to Python Programming*.

Wen, H. (2014). *A review of the Henon map and its physical interpretations*.

Wen, H., Lin, Y., Yang, L., & Chen, Ruiting. (2024). Cryptanalysis of an image encryption scheme using variant Hill cipher and chaos. *Expert Systems with Applications*, *250*, 123748. https://doi.org/10.1016/j.eswa.2024.123748