Integra: Journal of Integrated Mathematics and Computer Science

Journal Homepage: https://integrajimcs.com/index.php/integra/index

integra
Journal of Integrated Mathematics and Computer Science

Research Paper

Key Matrix Generation Using Random Functions in Hill Cipher Modulo 95 Cryptography

Nurdin Bahtiar[1], Aris Puji Widodo[1], Nikken Prima Puspita[2*]

[1] *Department of Informatics, Faculty of Science and Mathematics, Universitas Diponegoro, Central Java, 50275, Indonesia*
[2] *Department of Mathematics, Faculty of Science and Mathematics, Universitas Diponegoro, Central Java, 50275, Indonesia*
[*] **Corresponding author**: nikkenprima@lecturer.undip.ac.id

Keywords
Key Matrix, Randomization Algorithm, Hill Cipher, Encryption, Decryption

Abstract
In symmetric cryptography, the confidentiality of the chosen key and the security of its delivery mechanism are paramount to minimize the risk of unauthorized disclosure. Typically, in such systems, the sender and recipient focus primarily on the message (plaintext and ciphertext) rather than the complexities associated with key management. This approach aims to alleviate the burden of selecting a suitable and robust key for communicating parties. This study introduces a Hill Cipher modulo 95 cryptography method employing a matrix-based key, where key generation is achieved through a quantifiable randomization algorithm. The developed 2x2 key matrix facilitates a substantial number of possible keys, specifically 95^4 (exceeding 81 million). The key matrix generation process incorporates several functions, including those for ASCII character conversion, prime number verification, relative primality checks, modulo arithmetic, inverse modulo computation, determinant calculation, and inverse matrix determination. To simulate the encryption and decryption process, a desktop application was developed using the Lazarus Development IDE version 3.6. The application demonstrates effective generation of the required key matrix.

Received: 13 December 2024, Accepted: 23 February 2025
https://doi.org/10.26554/integrajimcs.20252111

1. INTRODUCTION

Data security is paramount in contemporary application development, forming the cornerstone for maintaining user trust, ensuring regulatory compliance, and protecting sensitive information from unauthorized access. As applications are increasingly integrated into various facets of life, spanning personal communications, financial transactions, critical infrastructure management, and healthcare systems, the imperative to safeguard the data they process and store intensifies [1]. The escalating sophistication of cyber threats, coupled with the growing complexity of application architectures and the proliferation of attack vectors, necessitates a comprehensive and proactive approach to data security [2]. This approach should encompass robust security measures, secure coding practices, and continuous monitoring to mitigate potential vulnerabilities and ensure data confidentiality, integrity, and availability.

Data encryption is a critical measure for protecting sensitive information against security threats such as data theft and unauthorized access. By transforming data into an unintelligible format without the appropriate decryption key, encryption ensures that only authorized entities can access the information. In an era characterized by frequent cyber-attacks and data breaches, encryption serves as a fundamental tool for preserving individual privacy and ensuring business security. Furthermore, encryption facilitates adherence to stringent data security regulations across diverse industries, thereby reinforcing trust among customers and business partners [3].

Symmetric cryptography, which employs a single secret key for both encryption and decryption, establishes a confidential channel between communicating parties who share the key. This

method represents a foundational approach to ensuring confidentiality, as unauthorized entities are unable to decipher encrypted information without possessing the key. Symmetric encryption is valued for its efficiency, rendering it suitable for encrypting large volumes of data, as commonly seen in file encryption, messaging applications, and data transfer protocols [4].

Hill Cipher cryptography, developed by Lester S. Hill in 1929, is an encryption technique rooted in linear algebra that utilizes matrix operations to secure messages. In this method, the plaintext is converted into a numerical representation and subsequently encrypted through multiplication by a predetermined key matrix. The key matrix must be invertible in modulo arithmetic to enable decryption. The strength of the Hill Cipher lies in its capacity to transform multiple characters simultaneously, thereby increasing resistance to frequency analysis. However, the Hill Cipher has limitations, including vulnerability to specific attacks if the key is not appropriately selected. Nevertheless, this method represents an important precursor in the evolution of modern cryptography [5].

The uniqueness of the encryption key is a crucial determinant of data security against unauthorized threats. A unique key provides an additional layer of protection by increasing the difficulty for attackers to guess or replicate it. The utilization of a unique key ensures that each encryption instance is exclusive to a particular user or system, thus mitigating the risk of key compromise. Moreover, employing a unique key contributes to data integrity and confidentiality, as only the corresponding key can decrypt the information. Across both business and personal contexts, key uniqueness is a fundamental prerequisite for maintaining trust and safeguarding sensitive information from evolving security threats. Consequently, the selection or generation of a unique key is an indispensable step in any encryption system.

Determining an appropriate encryption key presents a significant challenge. The diversity of encryption algorithms necessitates keys with specific formats and lengths, complicating the key selection process. Furthermore, key strength is contingent upon its uniqueness, and generating a truly unique key demands a meticulous approach and robust software. Therefore, application users face the challenge of obtaining a sufficiently robust key, a task that is not universally straightforward.

This research introduces a method for generating keys in the context of classical symmetric cryptography, specifically the Hill Cipher. Generating the required invertible matrix key presents a non-trivial challenge for users, particularly when performed manually.

2. LITERATURE REVIEW

Cryptography, historically limited to securing confidential communications, has undergone significant evolution. Now, computer networks increasingly rely on cryptography as a fundamental mechanism to guarantee the privacy of digital information. Early forms, like the first Caesar cipher or subsequent mono-alphabetic substitution ciphers, were notably vulnerable to statistical attacks [6].

The Hill Cipher is a symmetric cryptography method that has been utilized in various applications. However, some studies indicate that this method is susceptible to known-plaintext attacks [7]. Research has explored variations in the value and dimensions of the key matrix as a potential strategy to address this vulnerability. An alternative perspective, another article asserts that lattice-based cryptography is a highly promising and practical solution for securing the Internet of Things (IoT) against the looming threat of quantum computing [8].

2.1 Hill Cipher Cryptography Algorithm

The Hill Cipher cryptography method employs a transformation matrix as a core component of its operation. Introduced by Lester S. Hill in 1929, this technique was detailed in his publication, "Cryptography in an Algebraic Alphabet" [9]. The Hill Cipher is based on the principle that matrices can be used for encryption due to their invertible property, which allows for decryption using the inverse of the key matrix. Consequently, decryption of the ciphertext necessitates that the message recipient first calculate the inverse of the key matrix. Thus, the inverse matrix can only be determined if the key matrix is known [10].

To encrypt plaintext $\mathbf{P} = (p_1, p_2, \dots\ p_n)$ using the key matrix $\mathbf{K} = [k_{ij}]$ which produces ciphertext $\mathbf{C} = (c_1, c_2, \dots\ c_n)$ can be expressed as follows [10]:

$$\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} k_{11} & k_{12} & \cdots & k_{1n} \\ k_{21} & k_{22} & \cdots & k_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ k_{n1} & k_{n2} & \cdots & k_{nn} \end{bmatrix} \cdot \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{bmatrix} \mod 26$$

or in the form of a system of linear equations in modulus 26 expressed as:

$$\begin{aligned} c_1 &= (k_{11}p_1 + k_{12}p_2 + \cdots + k_{1n}p_n) \mod 26 \\ c_2 &= (k_{21}p_1 + k_{22}p_2 + \cdots + k_{2n}p_n) \mod 26 \\ &\cdots \\ c_n &= (k_{n1}p_1 + k_{n2}p_2 + \cdots + k_{nn}p_n) \mod 26 \end{aligned}$$

or in notation is stated as:

$$\mathbf{C} = \mathbf{KP} \mod 26$$

Meanwhile, to decrypt the ciphertext $\mathbf{C} = (c_1, c_2, \dots, c_n)$ using the key matrix $\mathbf{K}^{-1} = [k_{ij}]^{-1}$ which produces the plaintext $\mathbf{P} = (p_1, p_2, \dots, p_n)$ can be expressed in the notation:

$$\mathbf{P} = \mathbf{K}^{-1}\mathbf{C} \mod 26$$

2.2 ASCII Codes

The standard ASCII (American Standard Code for Information Interchange) code uses 7 bits to represent each character, so there are 27 = 128 unique characters. But 33 of these characters are non-printable control characters. For example, the backspace, delete, escape, enter, and other characters. The rest, amounting to 95 characters (including the space character) are printable characters that can be seen or not seen visually on the computer

screen including upper and lower case letters, numbers, punctuation marks, and symbols [11, 12, 13, 14].
The 95 characters in ASCII contained in the discussion of this research include:

1. Capital letters (A to Z) totaling 26 characters
2. Lowercase letters (a to z) totaling 26 characters
3. Numbers (0 to 9) totaling 10 characters
4. Special symbol characters totaling 32 characters, *namely* ! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [\] _ ` { } ~
5. A space character

In this research, 95 characters were used because that is the number of characters used by humans in communicating in everyday life. This application is a development of research that was previously conducted by researchers [15].

A coded message can be transformed into its ASCII equivalent, then converted into its binary representation. Once in binary, it's ready to be embedded within a cover file using an appropriate cryptographic algorithm [16]. Several studies present new cryptographic algorithms. For example, the proposed algorithm can be used to encrypt and decrypt text messages based on ASCII character codes. The main idea of the proposed algorithm is to associate each character in the plain text with the previous character during encryption and decryption [17].

2.3 Random Numbers

In cryptography, generating random numbers is absolutely crucial. These numbers must be able to withstand attacks from adversaries. Therefore, it's essential to validate their functionality and robustness against various attacks, including fault injection attacks [18]. We've determined that while all of them are cryptographically secure against traditional attacks, they are vulnerable to quantum computers. This is because they can be broken using Shor's and Grover's algorithms [19].

Generating random numbers is a core challenge across numerous information processing fields. This includes not only classical and quantum cryptography, but also areas like mathematical modeling, Monte Carlo methods, gambling, and many more. For most of these applications, both the quality of the randomness and the efficiency of the generation process are critically important [20]. Random number generation has been the subject of numerous studies in computer science and information theory. For instance, some research has yielded explicit findings on the performance of the interval algorithm for generating random numbers, specifically when using real number expressions [21].

A data-driven approach relies on stochastic processes simulated with random numbers. True random numbers are inherently unpredictable, lacking any bias or correlation, making it virtually impossible to create a system that produces 'genuinely' random values. Consequently, research has focused on developing pseudo-random number generators. While not truly random, these pseudo-random numbers prove practically useful [22].

3. METHODS

In this research, several function modules were used as follows:

1. **ApakahPrima** function. This function is used to check whether an integer number i is a prime number or not. Here is the pseudocode:

```
1. prima ← true
2. for k ← 2 to (i div 2) do
   if (i mod k = 0)
      prima ← false
      break
3. apakahPrima ← prima
```

2. **ApakahRelatifPrima** function. This function is used to check whether two integers a and m are relatively prime or not. Here is the pseudocode:

```
1. RelatifPrima ← false
2. if a <= m then
      batas ← a
      else
      batas ← m
3. for I ← 1 to batas do
       if apakahPrima(i)
          if (a mod i = 0) and (m mod i = 0)
             RelatifPrima ← true
             break
4. apakahRelatifPrima ← RelatifPrima
```

3. **Modulo** function. This function is used to calculate the modulo value y of an integer number x that is entered. Here is the pseudocode:

```
1. Modulo ← x - (x div y) * y;
```

4. **Invers_Modulo** Function. This function is used to calculate the inverse modulo m value of an integer a that is entered. Here is the pseudocode:

```
1. aI ← 0
2. if apakahRelatifPrima(a,m)
      for I ← 1 to m do
          if (a*i) mod m = 1
             aI ← i
             break
3. Invers_Modulo ← aI
```

4. RESULTS AND DISCUSSION

In this research, a desktop-based software has been developed to implement the developed method [23, 24]. The development was carried out using Lazarus Development IDE Version 3.6 [25].

The explanation using the application is shown as follows. Suppose given an example of a plaintext:

"Pasukan datang jam 03 petang"

Table 1. Experiment to Generate 2x2 Key Matrix

Experiment	2x2 Matrix	Invers	Determinant	Determinant Invers in Z^{95}
1	$\begin{pmatrix} 6 & 7 \\ 10 & 84 \end{pmatrix}$	$\begin{pmatrix} 86 & 72 \\ 35 & 74 \end{pmatrix}$	434	44
2	$\begin{pmatrix} 86 & 44 \\ 84 & 63 \end{pmatrix}$	$\begin{pmatrix} 29 & 28 \\ 88 & 23 \end{pmatrix}$	1722	8
3	$\begin{pmatrix} 39 & 36 \\ 35 & 13 \end{pmatrix}$	$\begin{pmatrix} 29 & 22 \\ 91 & 87 \end{pmatrix}$	-753	-27
4	$\begin{pmatrix} 8 & 70 \\ 19 & 47 \end{pmatrix}$	$\begin{pmatrix} 12 & 65 \\ 76 & 93 \end{pmatrix}$	-954	-24
5	$\begin{pmatrix} 90 & 19 \\ 43 & 93 \end{pmatrix}$	$\begin{pmatrix} 91 & 57 \\ 9 & 85 \end{pmatrix}$	7553	2
6	$\begin{pmatrix} 92 & 65 \\ 19 & 11 \end{pmatrix}$	$\begin{pmatrix} 63 & 25 \\ 38 & 26 \end{pmatrix}$	-223	-72
7	$\begin{pmatrix} 52 & 84 \\ 46 & 44 \end{pmatrix}$	$\begin{pmatrix} 6 & 49 \\ 11 & 33 \end{pmatrix}$	-1576	-56
8	$\begin{pmatrix} 15 & 1 \\ 43 & 16 \end{pmatrix}$	$\begin{pmatrix} 43 & 27 \\ 21 & 70 \end{pmatrix}$	197	68
9	$\begin{pmatrix} 78 & 80 \\ 37 & 41 \end{pmatrix}$	$\begin{pmatrix} 82 & 30 \\ 21 & 61 \end{pmatrix}$	238	2
10	$\begin{pmatrix} 1 & 48 \\ 64 & 90 \end{pmatrix}$	$\begin{pmatrix} 90 & 9 \\ 12 & 77 \end{pmatrix}$	-2982	-18

"AwnJ*nTUEe0bmE*Xf(/31c9e0bm".

which is then encrypted into ciphertext:

Then the ciphertext is decrypted again to produce plaintext:

"Pasukan datang jam 03 petang".

The application display that illustrates how the Hill Cipher cryptography process is carried out is shown in Figure 1.

In the display example above, the matrix used as the key is a 2x2 matrix, namely:

$$\begin{bmatrix} 1 & 48 \\ 64 & 90 \end{bmatrix}$$

The 2x2 matrix is not entered manually by the user but is generated randomly through the application developed in this study. To get the value, press the "Set Matrix" button so that it displays the display as in Figure 2.

The steps for generating the 2x2 matrix are carried out by pressing the "Randomize!" button and then the algorithmic sequence carried out by the program is as follows:

1. Generate 4 random numbers between 1 and 95 as elements of matrix A identified as A11, A12, A21, and A22

$$A = \begin{bmatrix} 1 & 48 \\ 64 & 90 \end{bmatrix}$$

2. Defines the adjoint of the matrix A whose elements are identified as AdjA11, AdjA12, AdjA21, and AdjA22

$$\text{Adjoint}(A) = \begin{bmatrix} 90 & -48 \\ -64 & 1 \end{bmatrix}$$

3. Calculate the value of Det, which is the determinant of matrix A

$$\begin{aligned} \text{Det} &= A11 \times A22 - A12 \times A21 \\ &= 1 \times 90 - 64 \times 48 \\ &= 90 - 3072 \\ &= -2982 \end{aligned}$$

4. Calculates the value of Det_Inv, which is the inverse modulo 95 of Det

   IF Det < 0 THEN Det_Inv = $-$Invers_Modulo($-$Det, 95)
   IF Det > 0 THEN Det_Inv = Invers_Modulo(Det, 95)

   Since, Det = -2982 then:

$$\begin{aligned} \text{Det\_Inv} &= -\text{Invers\_Modulo}(-\text{Det}, 95) \\ &= -\text{Invers\_Modulo}(-(-2982), 95) \\ &= -18 \end{aligned}$$

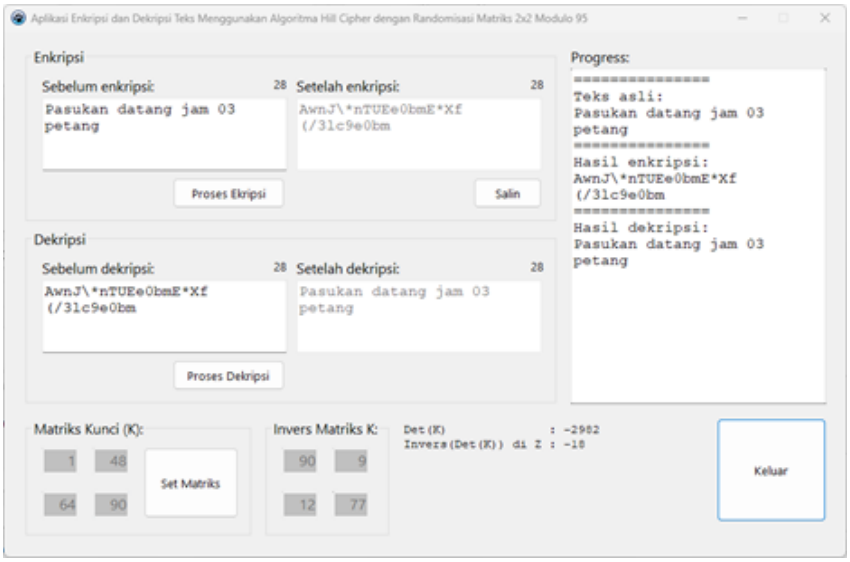

Figure 1. Encryption and Decryption Application Interface using Hill Cipher Algorithm

Figure 2. 2x2 Key Matrix Randomization Page Interface

5. If the value of Det_inv = 0 then the process is repeated from no. 1, but if Det_inv ≠ 0 proceed to step 6.
6. Calculate the inverse of matrix A, namely B = Det_Inv * Adjoin(A)
   Calculated the inverse matrix modulo 95, obtained:

$$B = \begin{bmatrix} 90 & 9 \\ 12 & 77 \end{bmatrix}$$

7. End.

In this research, 10 experiments were carried out to generate similar key matrices, the results of which are shown in Table 1.

5. CONCLUSIONS

This research introduced a method for generating key matrices in Hill Cipher cryptography, specifically by generating random numbers within the range of 1 to 95. The implementation of this automated generation process incorporates several functions, including those for generating random numbers, verifying prime numbers, identifying relatively prime numbers, and performing modulo, inverse modulo, determinant, and inverse matrix calculations. This study employed a 2x2 matrix as an illustrative example; however, the methodology can be extended to accommodate larger matrix dimensions, such as 3x3 or 4x4. To simulate the encryption and decryption processes, a desktop-based cryptography application was developed using the Lazarus Development IDE Version 3.6. The application demonstrated favorable results in generating 10 distinct key matrices.

6. ACKNOWLEDGEMENT

I would like to express my deepest gratitude to The Faculty of Science and Mathematics, Diponegoro University, for providing computational laboratory facilities for this research.

REFERENCES

[1] Padmaksha Roy, Jaganmohan Chandrasekaran, Erin Lanus, Laura Freeman, and Jeremy Werner. A survey of data

security: Practices from cybersecurity and challenges of machine learning. *arXiv preprint arXiv:2310.04513*, 2023.

[2] Nor Natasha Ashira Shamsudin, Saiful Farik Mat Yatin, Nurul Fadhlin Mohd Nazim, Amie Witiza Talib, Mohammad Afiq Mohamed Sopiee, and Fifi Natasya Shaari. Information security behaviors among employees. *International Journal of Academic Research in Business and Social Sciences*, 9(6):560–571, 2019.

[3] Hari Purwanto. Penerapan keamanan basis data dengan teknik enkripsi. *Jurnal Sistem Informasi Universitas Suryadarma*, 1(1):12–25, 2014.

[4] Mulder. *Trends in Data Protection and Encryption Technologies*. Springer, 2023.

[5] Fajar Sudana Putra and Dony Ariyus. Enkripsi & dekripsi teks menggunakan hill cipher dengan matriks ordo 3 x 3. *JURTI (Jurnal Teknologi Informasi)*, 5(1):17–22, 2021.

[6] H. Touil, N.E. Akkad, and K. Satori. Text encryption: Hybrid cryptographic method using vigenere and hill ciphers. In *2020 International Conference on Intelligent Systems and Computer Vision (ISCV)*, pages 1–6, 2020.

[7] Wafiqah Yasmin Azhar and Supriyadi. Kriptanalisis hill cipher terhadap known plaintext attack menggunakan metode determinan matriks berbasis android. *Jurnal SIMETRIS*, 8(2):579–586, 2017.

[8] Rui Xu, Chi Cheng, Yue Qin, and Tao Jiang. Lighting the way to a smart world: Lattice-based cryptography for internet of things. *arXiv preprint arXiv:1805.04880*, 2018.

[9] Lester S. Hill. Cryptography in an algebraic alphabet. *The American Mathematical Monthly*, 36(6):306–312, 1929.

[10] Rinaldi Munir. *Kriptografi*. Penerbit Informatika, Bandung, edisi kedua edition, 2019.

[11] American National Standards Institute. *ISO-IR-6: ASCII Graphic Character Set*. 1975. Accessed on 29 May 2025.

[12] American National Standards Institute. *American National Standard Code for Information Interchange (ASCII) (ANSI X3.4-1968)*. 1968.

[13] Eric S. Raymond. *The Jargon File (version 4.4.8)*. Accessed on 29 May 2025.

[14] David A. Patterson and John L. Hennessy. *Computer Organization and Design: The Hardware/Software Interface*. Morgan Kaufmann, fifth edition edition, 2014.

[15] Nikken Prima Puspita and Nurdin Bahtiar. *Kriptografi Hill Cipher dengan Menggunakan Operasi Matriks*. Jurusan Matematika UNDIP, 2010.

[16] A. Mishra, P. Johri, and A. Mishra. Audio steganography using ascii code and ga. In *2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions) (ICTUS)*, pages 646–651, 2017.

[17] A. Elmogy, Y. Bouteraa, R. Alshabanat, and W. Alghaslan. A new cryptography algorithm based on ascii code. In *2019 19th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*, pages 626–631, 2019.

[18] S. Guilley and Y. El Housni. Random numbers generation: Tests and attacks. *2018 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pages 49–54, 2018.

[19] A. Kumar and A. Mishra. Evaluation of cryptographically secure pseudo random number generators for post quantum era. In *2022 IEEE 7th International Conference for Convergence in Technology (I2CT)*, pages 1–5, 2022.

[20] M.M. Jacak, P. Jóźwiak, J. Niemczuk, and J.E. Jacak. Quantum generators of random numbers. *Scientific Reports*, 11(1):16108, 2021.

[21] Y. Oohama. Performance analysis of the interval algorithm for random number generation based on number systems. *IEEE Transactions on Information Theory*, 57(3):1177–1185, 2011.

[22] Young-Seob Jeong, Kyojoong Oh, Cheol-Koo Cho, and Hyun-Joo Choi. Pseudo random number generation using lstms and irrational numbers. In *2018 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 541–544, 2018.

[23] R.H. Sianipar. *Java untuk Kriptografi*. Penerbit Andi, Yogyakarta, 2017.

[24] E. Setyaningsih. *Kriptografi dan Implementasinya Menggunakan MATLAB*. Penerbit Andi, Yogyakarta, 2015.

[25] Free Pascal Team. *Free Pascal Language Reference Guide*. Accessed on 29 May 2025.

