Journal of Technology and Computer (JOTECHCOM)
Vol. 2, No. 3, August 2025, pp. 146~156
ISSN: 3048-0477 (Media Online)

Implementation of the Hill Cipher Algorithm with a Random Generator in Key Validation for File Security

Fauzul Azmi[1], Tommy[2]
[1,2]Department of Informatic Engineering, Faculty of Computer and Engineering, University of Harapan Medan, Medan, Indonesia
[1]fauzulazmi28032000@gmail.com, [2]tomshirakawa@gmail.com

Article Info

Article history:

Received August 07, 2025
Revised August 08, 2025
Accepted August 09, 2025

Keywords:

Cryptography
Hill cipher
Data security
Digital files
Encryption
Key validation
Matrix

ABSTRACT

Digital file security plays a crucial role in the modern era of rapid information exchange and growing data threats. This study presents the development of a desktop-based file security system that utilizes a random key matrix approach for secure encryption and decryption. The core innovation lies in the validation of randomly generated key matrices to ensure they possess a valid inverse under modulo 256 arithmetic. This validation is critical, as it guarantees that encrypted data can be accurately decrypted using the same key. The system converts digital files into binary blocks, which are then encrypted through matrix multiplication with the validated key. Comprehensive testing was carried out using various file types and sizes to evaluate the system's robustness, performance, and reliability. The results demonstrate that the application provides an effective method for safeguarding digital files, with improved security due to the randomness and strength of the key generation process. This research contributes significantly to the ongoing advancement of cryptography-based solutions for file protection in desktop environments.

Corresponding Author:

Fauzul Azmi
University of Harapan Medan
Email: fauzulazmi28032000@gmail.com

1 INTRODUCTION

The security and confidentiality of information is extremely important. [1], especially sensitive or personal information that should only be accessed by authorized parties, both when stored on storage media (hard drives) and when being transmitted. [2]. What's more, if the transmission is carried out via a public network, if the data is not secured first, it will be very easy for unauthorized parties to intercept and access the information. [3], [4], [5]. One technique that can be used is cryptography. [6], [7].

Cryptography is a science of message encoding used to improve security in message delivery or data communication [6]. Cryptography has now become an important requirement in information technology security for the delivery of important and confidential messages. The transmission of important and confidential messages is highly vulnerable to attacks by third parties, such as eavesdropping, communication disruption, and message alteration [8], [9]. The purpose of cryptography is to protect the privacy or confidentiality of information, whether in the form of text or other formats, when it is shared so that it reaches its intended destination through the same channel [2] [10].

To date, Hill Cipher is one of the classic cryptography techniques that is still widely studied and is one of the matrix-based cryptography algorithms. Hill Cipher was discovered by Lester Hill in 1929 and can encrypt and decrypt messages using the principle of modular arithmetic. However, in its initial implementation, the Hill Cipher used a conventional alphabet with a modulo of 26, which is limited for contemporary applications that use characters beyond the alphabet. In the study [11], the modulo 26 operation was successfully modified to modulo 256, covering all ASCII characters (0-255), including letters, numbers, symbols, and special characters. The strength of the algorithm used in data security does not only depend on the complexity of the algorithm used, but also lies in the randomness and complexity of the key used [11].

One way to improve the security of the Hill Cipher algorithm is to use a random generator for the key. The random generator will produce an unpredictable key, thereby increasing the level of security. However, not all random keys can be used as keys in the Hill Cipher algorithm, as the keys must meet certain criteria, namely the key matrix must have an inverse modulo 256. Nevertheless, key validation can be performed before file encryption. Therefore, this study aims to implement the Hill Cipher algorithm using a random generator for key validation in file encryption.

2. METHOD

2.1. Hill Cipher

In its application, the Hill Cipher uses matrix multiplication and matrix inversion techniques. The key in the Hill Cipher is an n x n matrix, where n is the block size. The key matrix K must be invertible, meaning it has a multiplicative inverse K-1. Therefore, the key must have an inverse because the matrix K-1 is the key used for decryption [12], [13]. The encryption process in the Hill Cipher is performed on each block of plaintext. The block size is the same as the size of the key matrix. Before dividing the text into a series of blocks, the plaintext is first converted into numbers, such that A=0, B=1, up to Z=25 [12], [13]. In this study, matrix multiplication uses the range (0–255), commonly referred to as a byte file.

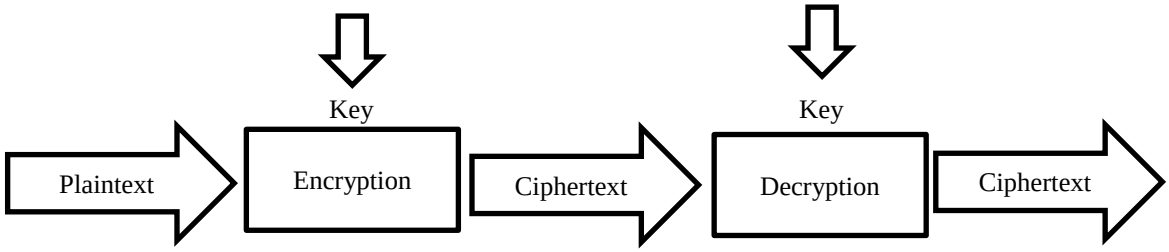


Figure 1. Encryption and Decryption Process

2.2. Steps of Hill Cipher

In the encryption process, plaintext is divided into sequential blocks corresponding to the size of the key matrix used. The plaintext matrix P and ciphertext matrix C in the Hill Cipher algorithm are shown in equations 1 and 3. Meanwhile, the key matrix K in the Hill Cipher algorithm is shown in equation 2. If there is a plaintext P that has a key matrix K to be encrypted into ciphertext C, then mathematically the encryption process can be formulated in equation 4 and the decryption process in equation 5. Then, for the key determinant matrix and key inverse matrix, they can be formulated in equation 6 and equation 7.

$$P = \begin{bmatrix} P_{0,0} & P_{0,1} \\ P_{1,0} & P_{1,1} \end{bmatrix} \quad \dots (1)$$

$$K = \begin{bmatrix} K_{0,0} & K_{0,1} \\ K_{1,0} & K_{1,1} \end{bmatrix} \quad \dots (2)$$

$$C = \begin{bmatrix} C_{0,0} & C_{0,1} \\ C_{1,0} & C_{1,1} \end{bmatrix} \quad \dots (3)$$

$$C = K \times P \quad \dots (4)$$

$$P = K^{-1} \times C \quad \dots (5)$$

$$|K| = \begin{vmatrix} K_{0,0} & K_{0,1} \\ K_{1,0} & K_{1,1} \end{vmatrix} = K_{0,0}.K_{1,1} - K_{0,1}.K_{1,0} \neq 0 \quad \dots (6)$$

$$K^{-1} = |K|^{-1} \begin{pmatrix} K_{1,1} & -K_{0,1} \\ -K_{1,0} & K_{0,0} \end{pmatrix} \quad \dots (7)$$

Decryption is performed by reversing the encryption process. The inverse of the same key matrix is used to perform decryption, which allows the ciphertext to be returned to plaintext. However, ensuring that the

key matrix has an invertible determinant is a major problem with the Hill Cipher. This must be done by ensuring that the determinant is not zero and that the result of the modulo used is 1 [11].

2.3. Random Number Generator

True Random Number Generator (TRNG)

TRNG extracts entropy from unpredictable physical sources to generate nondeterministic random numbers. However, it takes more effort to obtain nondeterministic random numbers derived from physical activities outside the computer [14].

TRNG generates pure random numbers from truly random physical phenomena, such as thermal fluctuations, radioactive noise, or time variability. Examples of its use include high-level cryptography, lotteries, or games that require true randomness. Based on its operating principle, TRNG relies on unpredictable physical phenomena, such as electronic noise, radioactivity, or time variations between user button clicks. The process involves collecting physical data, such as signal fluctuations, then processing the data into random numbers using digital methods, followed by bias correction (if the physical source has an imbalance). The advantages and disadvantages of TRNG are that it can generate true randomness that is independent of the seed. Its disadvantage is that it is slower because it depends on physical hardware [15].

Pseudo-Random Number Generator (PRNG)

PRNG generates random number sequences based on deterministic algorithms using computers. The sequences generated have repeating patterns over a certain period of time and can be predicted if the initial conditions and algorithms are known [14].

PRNG uses mathematical algorithms to generate numbers that appear random but are actually determined by the initial value (seed). Examples of its use include computer simulations, genetic algorithms, games, or general programming. According to its working principle, PRNG is generated by a deterministic algorithm. The PRNG process begins with an initial value called a seed, then uses mathematical functions to generate a sequence of numbers from the seed. The resulting pattern appears random, but if the seed is the same, the sequence of numbers generated will also be the same. For example, the Linear Congruential Generator (LCG) algorithm can be seen in Equation 8.

$$X_{n+1} = (aX_n + c)\ mod\ m \quad \ldots (8)$$

2.4. Representation of Byte Files to ASCII Characters

In applying the Hill Cipher algorithm to digital files, the data to be encrypted is not limited to text alone, but can include various types of files such as documents, images, videos, music, or other binary files. In order for these files to be processed mathematically by the algorithm, the contents of the files must first be converted into numerical form. Every digital file essentially consists of a series of bytes with values ranging from 0 to 255. These values represent the binary form of the file data, which in the Hill Cipher encryption implementation are treated as elements of a vector or matrix to be processed using modulo 256 operations. With this approach, the Hill Cipher algorithm can be applied directly to the file contents, where each byte of the file is treated as an integer between 0 and 255. This process ensures that all data can be encrypted and decrypted correctly without losing any information.

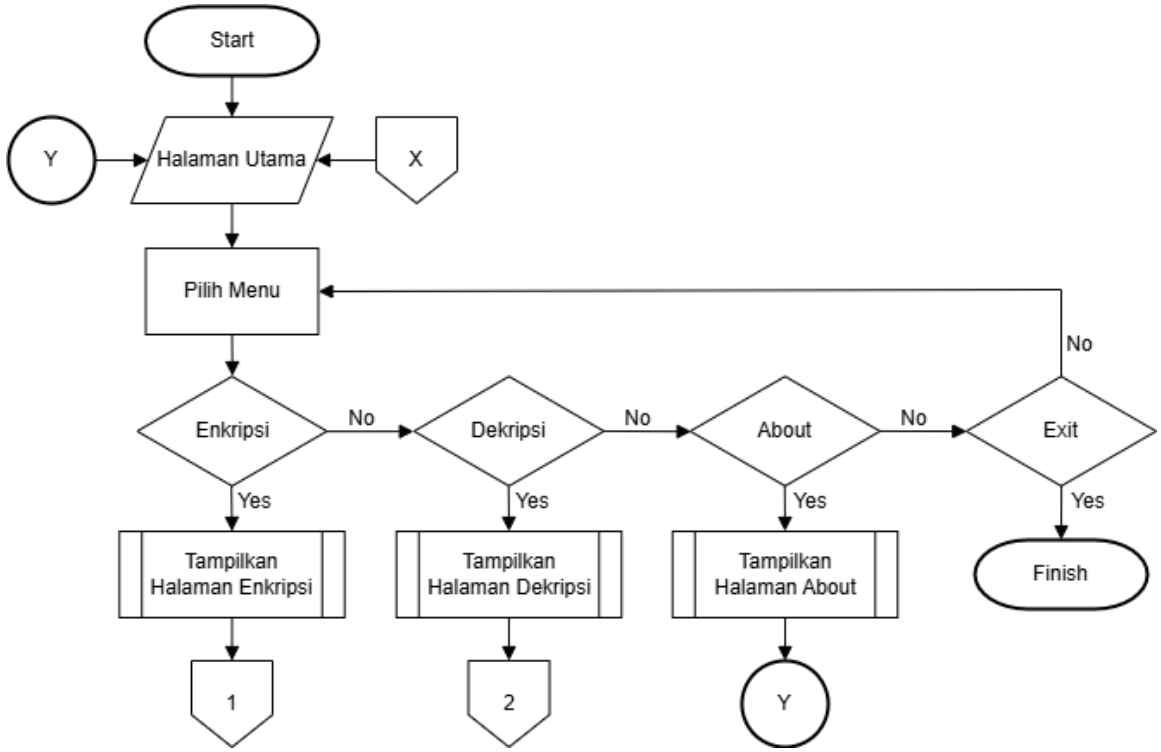
Table 1. Representation of Byte Files to ASCII Characters

Dec	Char	Dec	Char	Dec	Char	Dec	Char	Dec	Char	Dec	Char
0	NUL	43	+	86	V	129	ü	172	¼	215	╫
1	SOH	44	,	87	W	130	é	173	¡	216	╪
2	STX	45	-	88	X	131	â	174	«	217	┘
3	ETX	46	.	89	Y	132	ä	175	»	218	┌
4	EOT	47	/	90	Z	133	à	176	░	219	█
5	ENQ	48	0	91	[	134	å	177	▒	220	▄
6	ACX	49	1	92	\	135	ç	178	▓	221	▌
7	BEL	50	2	93	]	136	ê	179	│	222	▐
8	BS	51	3	94	^	137	ë	180	┤	223	▀
9	TAB	52	4	95	_	138	è	181	╡	224	α
10	LF	53	5	96	`	139	ï	182	╢	225	ß
11	VT	54	6	97	a	140	î	183	╖	226	Γ
12	FF	55	7	98	b	141	ì	184	╕	227	π
13	CR	56	8	99	c	142	Ä	185	╣	228	Σ
14	SO	57	9	100	d	143	Å	186	║	229	σ

15	SI	58	:	101	e	144	É	187	╗	230	µ
16	DLE	59	;	102	f	145	æ	188	╝	231	τ
17	DC1	60	<	103	g	146	Æ	189	╜	232	Φ
18	DC2	61	=	104	h	147	ô	190	╛	233	Θ
19	DC3	62	>	105	i	148	ö	191	┐	234	Ω
20	DC4	63	?	106	j	149	ò	192	└	235	δ
21	NAK	64	@	107	k	150	û	193	┴	236	∞
22	SYN	65	A	108	l	151	ù	194	┬	237	φ
23	ETB	66	B	109	m	152	ÿ	195	├	238	ε
24	CAN	67	C	110	n	153	Ö	196	─	239	∩
25	EM	68	D	111	o	154	Ü	197	┼	240	≡
26	SUB	69	E	112	p	155	¢	198	╞	241	±
27	ESC	70	F	113	q	156	£	199	╟	242	≥
28	FS	71	G	114	r	157	¥	200	╚	243	≤
29	GS	72	H	115	s	158	₧	201	╔	244	⌠
30	RS	73	I	116	t	159	ƒ	202	╩	245	⌡
31	US	74	J	117	u	160	á	203	╦	246	÷
32	Space	75	K	118	v	161	í	204	╠	247	≈
33	!	76	L	119	w	162	ó	205	═	248	°
34	"	77	M	120	x	163	ú	206	╬	249	∙
35	#	78	N	121	y	164	ñ	207	╧	250	·
36	$	79	O	122	z	165	Ñ	208	╨	251	√
37	%	80	P	123	{	166	ª	209	╤	252	ⁿ
38	&	81	Q	124	|	167	º	210	╥	253	²
39	'	82	R	125	}	168	¿	211	╙	254	■
40	(	83	S	126	~	169	⌐	212	╘	255	ÿ
41	)	84	T	127	DEL	170	¬	213	╒		
42	*	85	U	128	Ç	171	½	214	╓		

2.5. Flow Chart

A flowchart is a diagram that logically and structurally depicts the workflow or process within a system, from input to process to output. Flowcharts use standard graphic symbols to visualize how data or information flows through various stages within a system, including decision making, manual or automated processes, and interactions between system components.



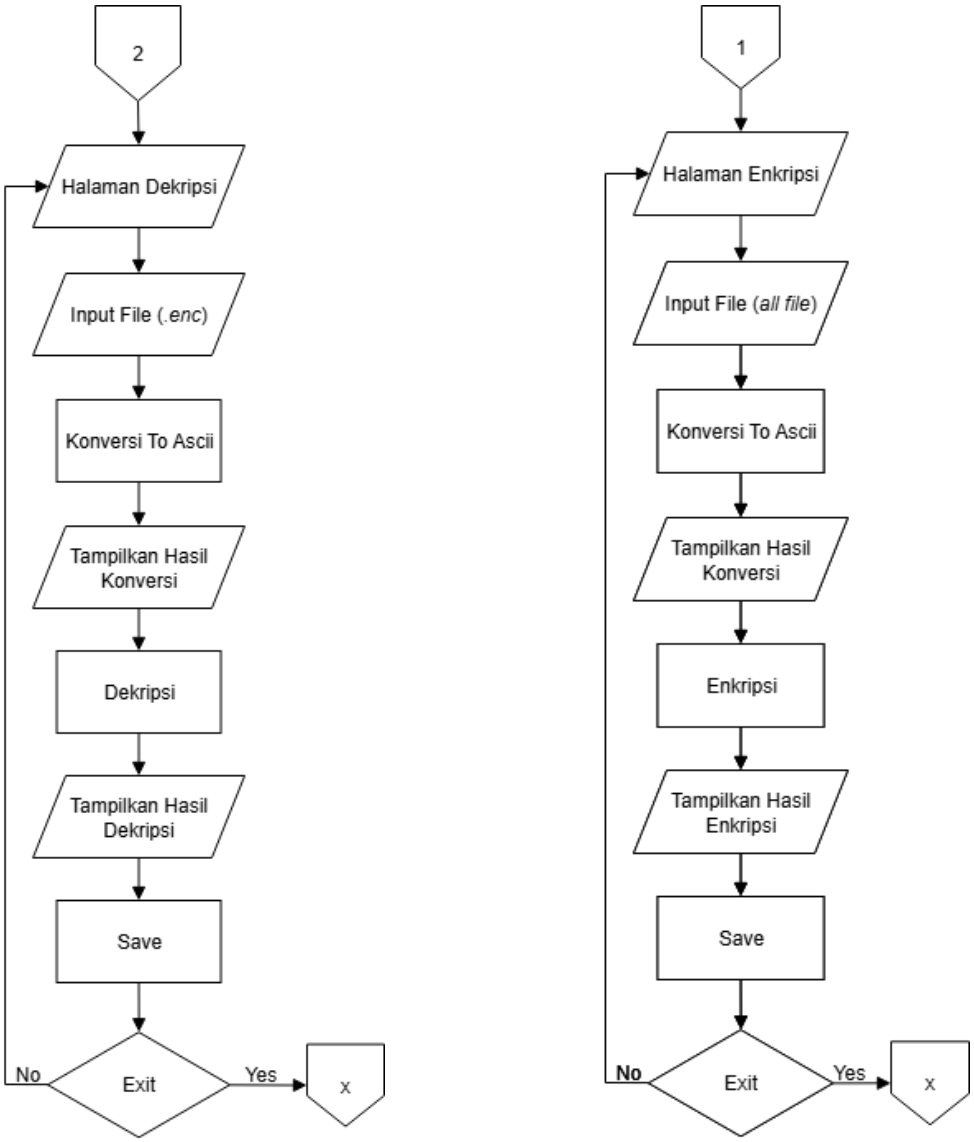


Figure 2. Encryption and Decryption Process Flowchart

The flowchart above is as follows:

a. Start: The process has begun.
b. The user selects the encryption, decryption, or about page.
c. If the user selects the encryption page, they are asked to enter a key that will be validated using the PRNG method and re-shuffle the user's key to generate a new key that will be validated to have an inverse modulo 256.
d. The user enters the file to be encrypted with the Hill Cipher algorithm key, then the new file and key file are saved.
e. If the user selects the decryption page, they are asked to enter the saved key file and then enter the encryption file to be decrypted using the Hill Cipher algorithm, which will then be saved in its original file format.
f. If the user selects the “About” page, the user's personal information will be displayed.
g. Finish: The end point of the process.

3. RESULTS AND DISCUSSION

3.1. Manual Calculations in the Hill Cipher Algorithm

This manual calculation of the Hill Cipher algorithm will explain how to manually calculate the Hill Cipher algorithm in detail. Starting from key generation, the encryption process, and the decryption process.

a. Encryption and Decryption Processes with the Hll Cipher Algorithm

1. Key Making:
   a) Enter 4 key characters. For example, “AZMI” without quotation marks.
   b) The key is converted into a byte file format as “65 90 77 73”.
   c) The key is randomized using the PRNG method with the following conditions: (a) = 65, (seed) = 90, (c) = 77, and modulus (m) = 255.
   d) Calculate using equation 8.
      $x_1 = \big((65 \times 90) + 77\big)\ mod\ 255 = 62$.
      $x_2 = \big((65 \times 62) + 77\big)\ mod\ 255 = 27$.
      $x_3 = \big((65 \times 27) + 77\big)\ mod\ 255 = 47$.
      $x_4 = \big((65 \times 47) + 77\big)\ mod\ 255 = 72$.
      Generating new keys “62 27 47 72”
   e) The key is validated to have an inverse modulo 256 with equations 6 and 7.
   f) Calculate $|K| = \begin{vmatrix} 62 & 27 \\ 47 & 72 \end{vmatrix} = \big((62 \times 72) - (27 \times 47)\big) mod\ 256 = 123$
   g) Calculate $K^{-1} = |123|^{-1} \begin{pmatrix} 72 & -27 \\ -47 & 62 \end{pmatrix} mod\ 256$
   h) Perform a multiplication experiment of the key matrix determinant with numbers in the range 1–255, then modulus the result by 256 and it must be equal to 1.
      $(123 \times 1)\ mod\ 256 = 123$
      $(123 \times 2)\ mod\ 256 = 246$
      $(123 \times 3)\ mod\ 256 = 113$
      $(123 \times 179)\ mod\ 256 = 1$
      So the inverse determinant is 179
   i) Calculate the inverse of the key matrix. $K^{-1} = 179 \begin{pmatrix} 72 & -27 \\ -47 & 62 \end{pmatrix} mod\ 256$
      $= \begin{pmatrix} 12888 & -4833 \\ -8413 & 11098 \end{pmatrix} mod\ 256 = \begin{pmatrix} 88 & 31 \\ 35 & 90 \end{pmatrix}$
   j) The key validation result is $\begin{pmatrix} 88 & 31 \\ 35 & 90 \end{pmatrix}$ used in the decryption process.
2. File Encryption:
   a) Enter the file to be encrypted. For example, a jpg file or photo.
   b) Then convert it into a byte file format as “255 216 255 224 0 16 74 70 73 70”.
   c) Then the plaintext vector is divided into several matrix blocks.
      $P_1 = \begin{bmatrix} 255 \\ 216 \end{bmatrix}\ P_2 = \begin{bmatrix} 255 \\ 224 \end{bmatrix}\ P_3 = \begin{bmatrix} 0 \\ 16 \end{bmatrix}\ P_4 = \begin{bmatrix} 74 \\ 70 \end{bmatrix}\ P_5 = \begin{bmatrix} 73 \\ 70 \end{bmatrix}$
   d) Encryption for matrix blocks P_1: $(K \times P_1)$ mod 256 = $\left(\begin{bmatrix} 62 & 27 \\ 47 & 72 \end{bmatrix} \times \begin{bmatrix} 255 \\ 216 \end{bmatrix}\right)$ mod 256
      $C_1 = \begin{bmatrix} 21642 \\ 27537 \end{bmatrix}$ mod 256 = $\begin{bmatrix} 138 \\ 145 \end{bmatrix}$
   e) Encryption for matrix blocks P_2: $(K \times P_2)$ mod 256 = $\left(\begin{bmatrix} 62 & 27 \\ 47 & 72 \end{bmatrix} \times \begin{bmatrix} 255 \\ 224 \end{bmatrix}\right)$ mod 256
      $C_2 = \begin{bmatrix} 21858 \\ 28113 \end{bmatrix}$ mod 256 = $\begin{bmatrix} 98 \\ 209 \end{bmatrix}$
   f) Encryption for matrix blocksiyaa P_3: $(K \times P_3)$ mod 256 = $\left(\begin{bmatrix} 62 & 27 \\ 47 & 72 \end{bmatrix} \times \begin{bmatrix} 0 \\ 16 \end{bmatrix}\right)$ mod 256
      $C_3 = \begin{bmatrix} 432 \\ 1152 \end{bmatrix}$ mod 256 = $\begin{bmatrix} 176 \\ 128 \end{bmatrix}$
   g) Encryption for matrix blocks P_4: $(K \times P_4)$ mod 256 = $\left(\begin{bmatrix} 62 & 27 \\ 47 & 72 \end{bmatrix} \times \begin{bmatrix} 74 \\ 70 \end{bmatrix}\right)$ mod 256
      $C_4 = \begin{bmatrix} 6478 \\ 8518 \end{bmatrix}$ mod 256 = $\begin{bmatrix} 78 \\ 70 \end{bmatrix}$
   h) Encryption for matrix blocks P_5: $(K \times P_5)$ mod 256 = $\left(\begin{bmatrix} 62 & 27 \\ 47 & 72 \end{bmatrix} \times \begin{bmatrix} 73 \\ 70 \end{bmatrix}\right)$ mod 256
      $C_5 = \begin{bmatrix} 6416 \\ 8471 \end{bmatrix}$ mod 256 = $\begin{bmatrix} 16 \\ 23 \end{bmatrix}$
   i) Ciphertext result: “138 145 98 209 176 128 78 70 16 23” which will be saved in file format (.enc) along with the validated key in format (.key).
3. File Decryption:
   a) Enter the key file in (.key) format and the file to be decrypted. For example, (inc file).

b) Then the key file is converted into a byte file in the form of "88 31 35 90".
c) Then convert it into a byte file format as "138 145 98 209 176 128 78 70 16 23".
d) Then the plaintext vector is divided into several matrix blocks.

$$C_1 = \begin{bmatrix}138\\145\end{bmatrix} \quad C_2 = \begin{bmatrix}98\\209\end{bmatrix} \quad C_3 = \begin{bmatrix}176\\128\end{bmatrix} \quad C_4 = \begin{bmatrix}78\\70\end{bmatrix} \quad C_5 = \begin{bmatrix}16\\23\end{bmatrix}$$

e) Encryption for matrix blocks C_1: $(K^{-1} \times C_1) \bmod 256 = \left(\begin{bmatrix}88 & 31\\35 & 90\end{bmatrix} \times \begin{bmatrix}138\\145\end{bmatrix}\right) \bmod 256$

$$P_1 = \begin{bmatrix}16639\\17880\end{bmatrix} \bmod 256 = \begin{bmatrix}255\\216\end{bmatrix}$$

f) Encryption for matrix blocks C_2: $(K^{-1} \times C_2) \bmod 256 = \left(\begin{bmatrix}88 & 31\\35 & 90\end{bmatrix} \times \begin{bmatrix}98\\209\end{bmatrix}\right) \bmod 256$

$$p_2 = \begin{bmatrix}15103\\22240\end{bmatrix} \bmod 256 = \begin{bmatrix}255\\224\end{bmatrix}$$

g) Encryption for matrix blocks C_3: $(K^{-1} \times C_3) \bmod 256 = \left(\begin{bmatrix}88 & 31\\35 & 90\end{bmatrix} \times \begin{bmatrix}176\\128\end{bmatrix}\right) \bmod 256$

$$p_3 = \begin{bmatrix}19456\\17680\end{bmatrix} \bmod 256 = \begin{bmatrix}0\\16\end{bmatrix}$$

h) Encryption for matrix blocks C_4: $(K^{-1} \times C_4) \bmod 256 = \left(\begin{bmatrix}88 & 31\\35 & 90\end{bmatrix} \times \begin{bmatrix}78\\70\end{bmatrix}\right) \bmod 256$

$$P_4 = \begin{bmatrix}9034\\9030\end{bmatrix} \bmod 256 = \begin{bmatrix}74\\70\end{bmatrix}$$

i) Encryption for matrix blocks C_5: $K^{-1} \times C_5) \bmod 256 = \left(\begin{bmatrix}88 & 31\\35 & 90\end{bmatrix} \times \begin{bmatrix}16\\23\end{bmatrix}\right) \bmod 256$

$$P_5 = \begin{bmatrix}2121\\2630\end{bmatrix} \bmod 256 = \begin{bmatrix}73\\70\end{bmatrix}$$

j) Plaintext result: "255 216 255 224 0 16 74 70 73 70".

b. Conclusion

Based on the analysis of the encryption process above, it can be concluded that the security of data files is highly dependent on the key used. With the use of a random generator, this can produce truly random encrypted files, and based on the final result of the decryption process, the original data file is recovered, which is exactly the same as the original data file before encryption.

3.2. Implementation of Digital File Encryption with the Hill Cipher Algorithm

Encryption implementation is carried out to test whether the system built can perform its functions as expected. Encryption implementation is carried out in three stages, namely the process of converting keys into byte files and then validating them, the process of converting data files into byte files during data file input, and finally the process of encrypting files with the Hill Cipher algorithm key into ciphertext.

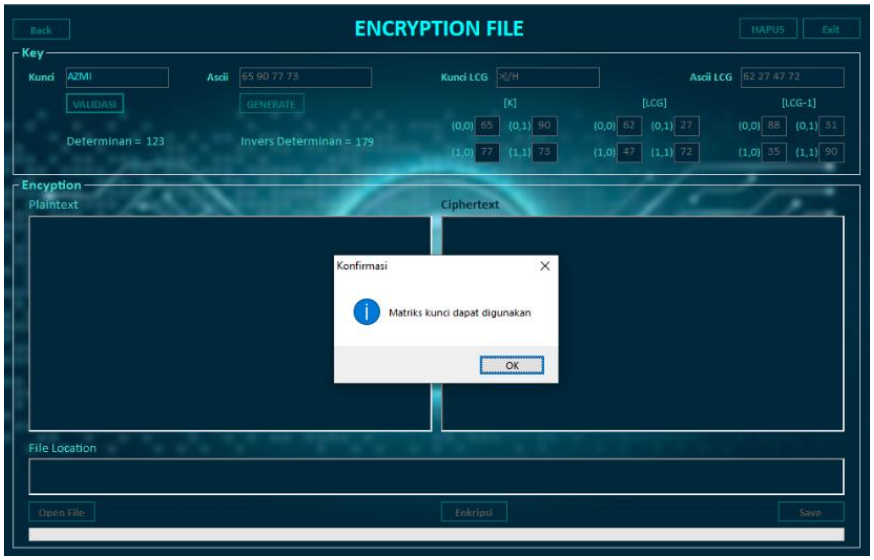


Figure 3. Key Validation

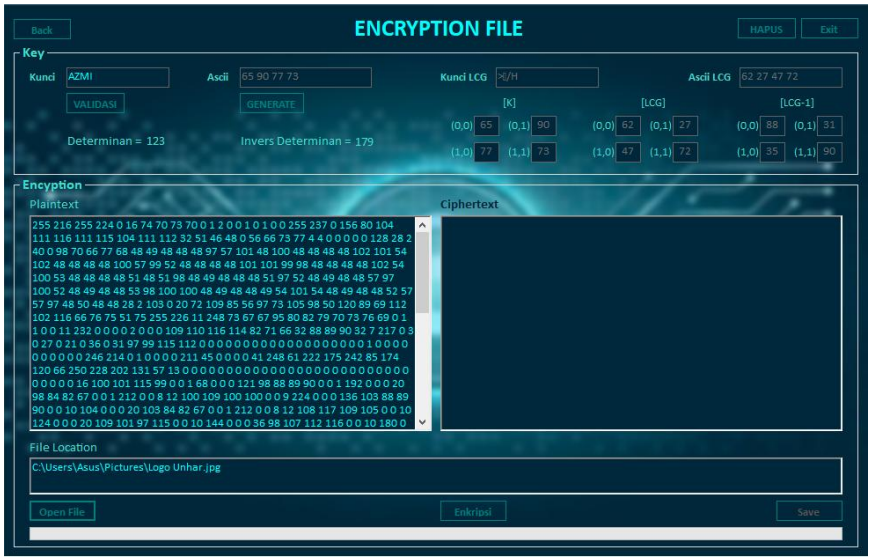


Figure 4. Convert Digital Files to Byte Files

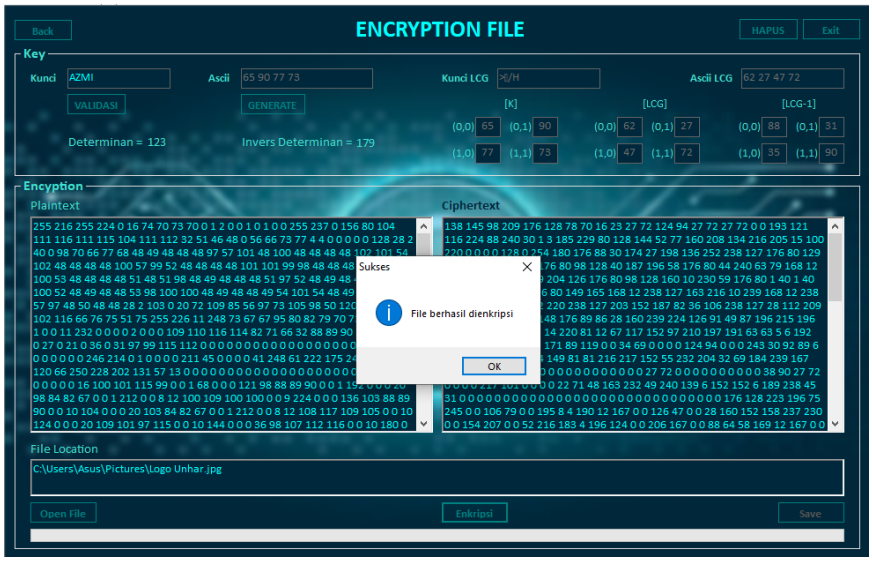


Figure 5. File Encryption Results

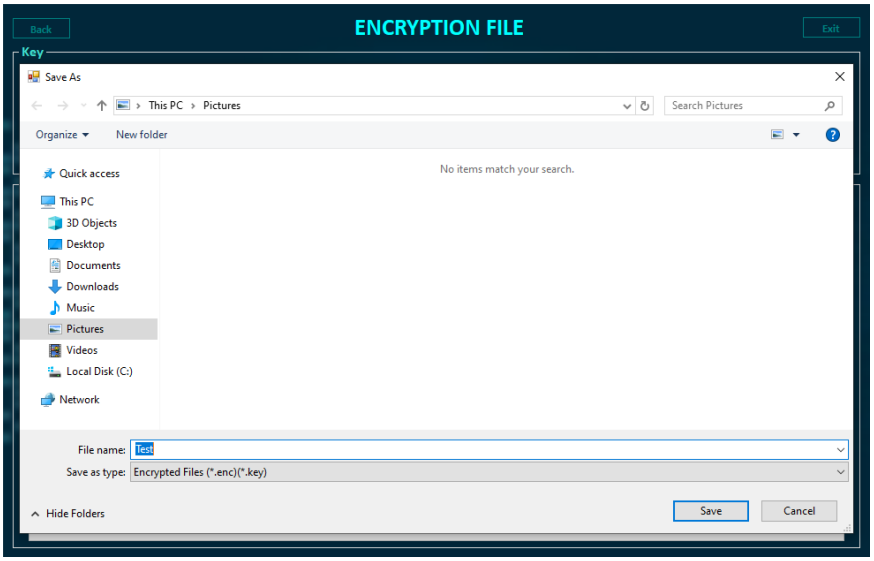


Figure 6. Save Encrypted Files and Key Files

3.3. Implementation of Digital File Decryption with the Hill Cipher Algorithm

Decryption is implemented to test whether the system can perform its functions as expected. Decryption implementation is the reverse of the previous encryption process, where decryption implementation also goes through three stages: the conversion of the key file into a byte file format, the conversion of the encrypted file into a byte file format during the input of the encrypted file, and finally the decryption of the file using the Hill Cipher algorithm key to restore the decryption results so that they can be converted back into the original data file.

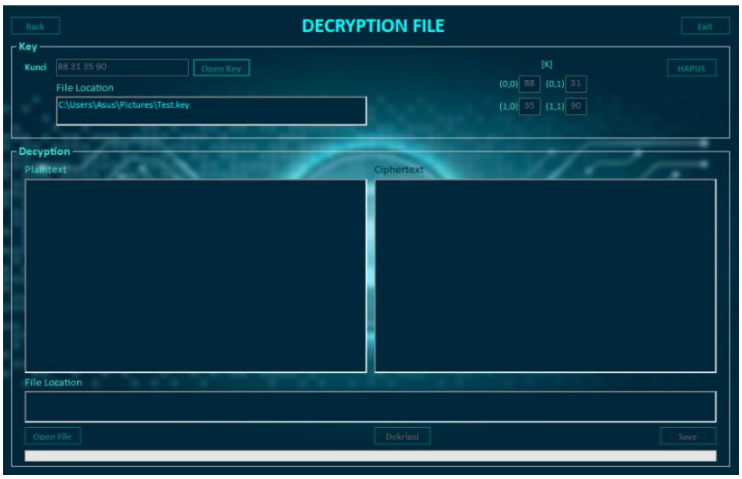


Figure 7. Convert Key Files to Byte Files

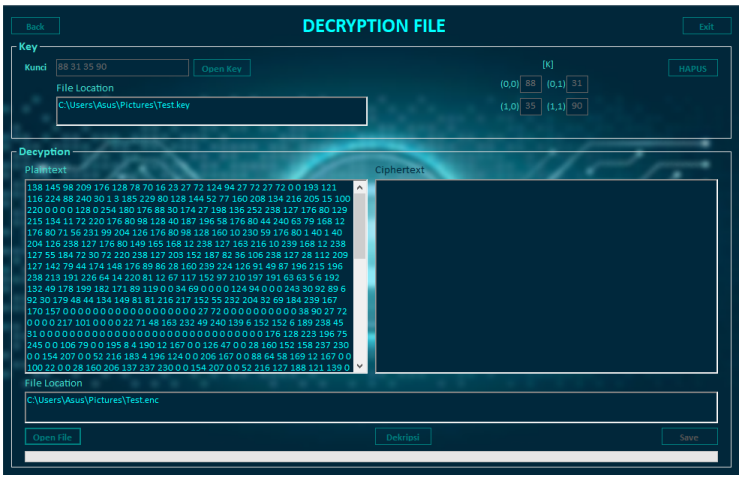


Figure 8. Convert Encrypted Files to Byte Files

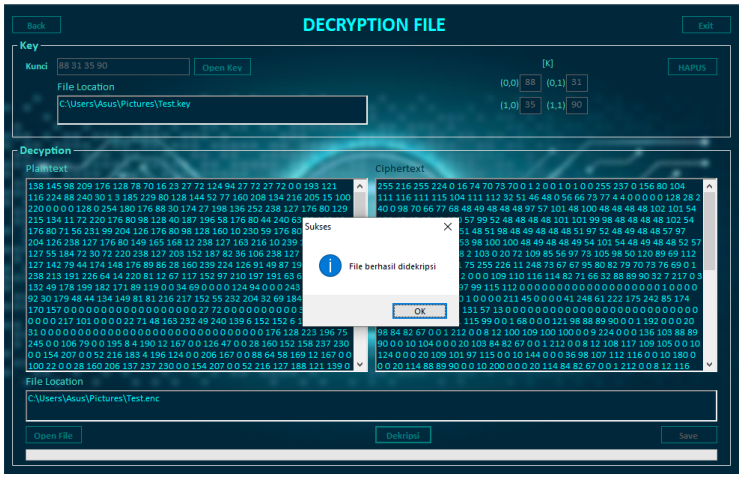


Figure 9. File Decryption Results

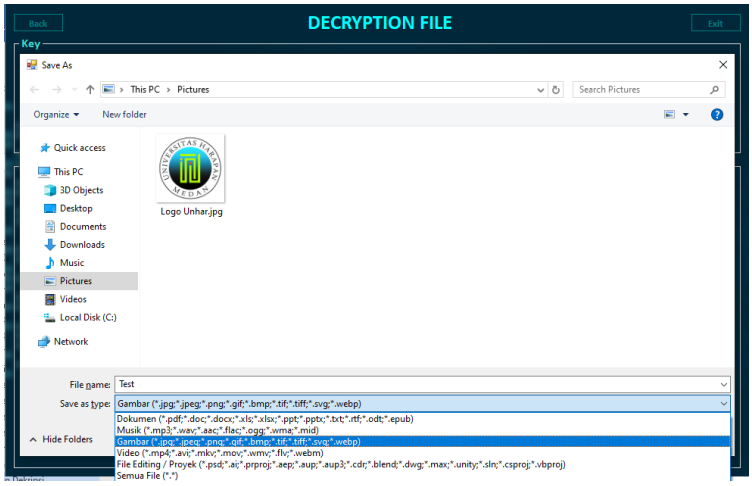


Figure 10. Save Decryption File

4. Conclusion

The conclusions that can be drawn after implementing and testing the Hill Cipher algorithm system using a random generator for key validation for file security are as follows:

1. The Hill Cipher algorithm has been proven to be effectively implemented to enhance the security of digital files, using a matrix key approach based on modular operations.
2. The use of random generators (both True Random Number Generators and Pseudo Random Number Generators) in key generation adds value to the security of the algorithm because it produces a random and unpredictable key matrix.
3. The key validation process is a crucial factor in ensuring that the key matrix has an inverse modulo 256 so that the encryption and decryption processes can run perfectly.
4. Applications built with the Visual Basic.NET programming language are capable of encrypting various types of files and restoring them to their original form through a highly accurate decryption process.
5. Test results show that encrypted data cannot be read without the appropriate key, and the decryption results are completely identical to the original file, proving the successful implementation of the designed system.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my supervisor for providing guidance and advice during the process of writing this journal. With his help, I was able to complete this writing successfully.

REFERENCES

[1] R. Dunnill and C. Barham, "Confidentiality and security of information," *Anaesthesia & Intensive Care Medicine*, vol. 8, no. 12, pp. 509–512, Dec. 2007, doi: 10.1016/J.MPAIC.2007.09.008.

[2] E. Prasetyo and Y. F. A. Lubis, "Optimasi Keamanan Hasil Enkripsi Algoritma Playfair Cipher ke dalam Kode Morse," *JiTEKH*, vol. 11, no. 1, pp. 41–50, 2023, doi: 10.35447/jitekh.v11i1.703.

[3] W. Rajeh and W. Rajeh, "Hadoop Distributed File System Security Challenges and Examination of Unauthorized Access Issue," *Journal of Information Security*, vol. 13, no. 2, pp. 23–42, Feb. 2022, doi: 10.4236/JIS.2022.132002.

[4] S. Shukla, J. P. George, K. Tiwari, and J. V. Kureethara, "Data Security," *SpringerBriefs in Applied Sciences and Technology*, pp. 41–59, 2022, doi: 10.1007/978-981-19-0752-4_3.

[5] V. Babak, S. Babak, V. Eremenko, Y. Kuts, and A. Zaporozhets, "Protection of Measurement Information from Unauthorized Access," pp. 409–458, 2025, doi: 10.1007/978-3-031-89406-0_10.

[6] F. Ruziq, "Analisis Kombinasi Algoritma Data Encryption Standard (DES) dan Algoritma Luc pada Pengamanan File," 2020, Accessed: Aug. 04, 2025. [Online]. Available: https://repositori.usu.ac.id/handle/123456789/24679

[7] B. K. Laia, "Modifikasi Algoritma Lucifer Dengan Menerapkan Pembangkitan Kunci Berdasarkan Naive Shuffle," *Journal of Informatics, Electrical and Electronics Engineering*, vol. 1, no. 3, pp. 110–118, 2022, doi: 10.47065/jieee.v1i3.355.

[8] M. Soni and D. K. Singh, "New directions for security attacks, privacy, and malware detection in WBAN," *Evol Intell*, vol. 16, no. 6, pp. 1917–1934, Dec. 2023, doi: 10.1007/S12065-022-00759-2/METRICS.

[9] Ö. Aslan, S. S. Aktuğ, M. Ozkan-Okay, A. A. Yilmaz, and E. Akin, "A Comprehensive Review of Cyber Security Vulnerabilities, Threats, Attacks, and Solutions," *Electronics 2023, Vol. 12, Page 1333*, vol. 12, no. 6, p. 1333, Mar. 2023, doi: 10.3390/ELECTRONICS12061333.

[10] F. Azmi and R. Anugrahwaty, "Analisis Matriks 5x7 Pada Kriptografi Playfair Cipher," *Jurnal & Penelitian Teknik Informatika*, vol. 1, no. 2, pp. 27–30, 2017.

[11] M. N. Sutoyo *et al.*, "Pengamanan Data Berbasis Hill Cipher dengan Operasi Modulo pada Karakter ASCII," vol. 23, no. 4, pp. 786–795, 2024.

[12] D. Laoli, B. Sinaga, and A. S. R. M. Sinaga, "Penerapan Algoritma Hill Cipher Dan Least Significant Bit (LSB) Untuk Pengamanan Pesan Pada Citra Digital," *JISKA (Jurnal Informatika Sunan Kalijaga)*, vol. 4, no. 3, p. 1, 2020, doi: 10.14421/jiska.2020.43-01.

[13] A. H. Hasugian, "Implementasi Algoritma Hill Cipher Dalam Penyandian Data," *Pelita Informatika Budi Darma*, vol. IV, no. 2, pp. 115–122, 2013.

[14] D. A. Risdianto and B. N. Prastowo, "Pengembangan True Random Number Generator berbasis Citra menggunakan Algoritme Kaotis," *IJEIS (Indonesian Journal of Electronics and Instrumentation Systems)*, vol. 10, no. 1, p. 87, 2020, doi: 10.22146/ijeis.36517.

[15] Moh. Wasil, "Implementasi Matriks Dalam Kriptografi Hill Cipher Dalam Mengamankan Pesan Rahasia," *Zeta - Math Journal*, vol. 8, no. 2, pp. 71–78, 2023, doi: 10.31102/zeta.2023.8.2.71-78.