Available online : https://doi.org/10.32665/james.v9i1.5067

Journal of Mathematics Education and Science

p-ISSN: 2621-1203 VOL. 9 NO. 1 (2026) : 94-108 e-ISSN: 2621-1211

Combination of the Vigenère Algorithm using RC4 Key Generator and ECB using LFSR Key Generator

Qonita Qurrota A'yun[1*], Auliya Rahman [1], Syaripuddin[1]

[1] Department of Mathematics, Universitas Mulawarman, Indonesia

[*] Corresponding Author. E-mail: qonitaqurrota@fmipa.unmul.ac.id

Keywords

Article History:
Received: 26-Jun. 2025
Revised: 05-Nov. 2025
Accepted: 14-Nov. 2025

Keywords:
Cryptography; Electronic Code Book; Linear Feedback Shift Register; Rivest Cipher 4; Vigenère Cipher

ABSTRACT

This research aims to demonstrate the process of encryption and decryption, as well as the implementation of a program in the form of a Graphical User Interface (GUI), using a combination of the Vigenère Cipher algorithm with an RC4 key generator and the Electronic Code Book (ECB) algorithm with an LFSR key generator. This combination was chosen because the Vigenère Cipher, as a classical algorithm, has a simple substitution structure, while RC4 is able to produce a highly random keystream. Meanwhile, ECB is an efficient modern block encryption method, and LFSR can generate a sequence of random bits based on a polynomial function. Together, these methods form a layered approach to securing messages. The data used in this study consists of plaintext messages in text form chosen by the author. The results show that the encrypted message can be successfully decrypted back into its original form. Therefore, this study proves that combining the Vigenère Cipher algorithm with RC4 and the ECB algorithm with LFSR can be effectively implemented and preserve data authenticity.

How to Cite:
Rahman, A., A'yun, Q. Q., & Syaripuddin. Combination of the Vigenere algorithm using RC4 key generator and ECB using LFSR key generator. *Journal of Mathematics Education and Science*, *15(1)*, 1-5. https://doi.org/10.32665/james.v9i1.5076

INTRODUCTION

Data security is essential and must be preserved in the current era of information technology development. The rapid growth of social media has greatly simplified communication, information exchange, and messages in the form of text, images, audio, or video. However, this advancement has also enabled certain parties to commit cybercrimes such as data theft, information monitoring, manipulation, or misuse of data for specific purposes (Giawa, 2022).

Cryptography is one of the primary techniques employed to address issues related to data security and confidentiality in order to prevent data leaks and misuse (Giawa, 2022). According to Fairuzabadi (2010), cryptography can be defined as hidden writing. The process of concealing messages is carried out through encryption techniques that transform data using an encryption key, rendering it unreadable to unauthorized parties. Conversely, decryption restores the data to its original form through a corresponding decryption key. Cryptography is inherently linked to mathematics, particularly with algebra, number theory, and combinatorics. Various mathematical concepts, including modulo operations, polynomial functions, mathematical logic, and permutations, are applied in the encryption and decryption processes. These concepts play a crucial role in safeguarding data against unauthorized access. Cryptography is generally classified into two main categories: classical and modern. Classical cryptographic methods, such as the Vigenère Cipher, possess security weaknesses despite their ease of implementation (Andayani and Agista, 2017). Attacks on these classical methods can be easily executed if the encryption key lacks sufficient complexity. In contrast, modern cryptographic techniques, such as the Electronic Code Book (ECB), provide stronger and more efficient encryption process through a block-based approach that offers higher resistance to attacks (Widya, 2018).

Cryptographic algorithms require robust key generation mechanism to achieve a high level of security. One approach is the Rivest Cipher 4 (RC4), which generates highly random keys through two main stages: the Key-Scheduling Algorithm (KSA) and the Pseudo-Random Generation Algorithm (PRGA). This algorithm makes the concealed message difficult to decrypt without knowledge of the correct key (Sulaiman and Isnanto, 2018). Alongside RC4, the Linear Feedback Shift Register (LFSR) is also widely used for key generation. According to Nahading and Wowor (2023), the LFSR is a key generation function that accepts an input and produces an output value to generate the keys, enabling the creation of random bit sequences with a maximum period. Classical cryptographic methods, which are no longer utilized today due to their insufficient data security, can be improved through layered approaches that improve data protection. One such approach involves the use of algorithm combinations. Several studies have examined combinations, including the work of Setyawati et al. (2021) who integrated the Vigenère Cipher with Caesar Cipher methods. In addition, Amijaya et al. (2022) conducted a study on a modification of the classical cryptographic algorithm, the Hill Cipher, using two words and 94 ASCII characters. Besides algorithm development, the presentation of cryptographic results in the form of an interactive user interface also plays an important role in facilitating the understanding and application of data security concepts. One form of its presentation is the Graphical User Interface (GUI). According to Lutz (2001), a Graphical User Interface (GUI) is an essential component in software development, including in the field of cryptography. Most users are familiar with GUI elements such as windows, buttons, and menus, which facilitate interaction with software systems.

In line with efforts to enhance security in classical cryptography, various studies have indicated that incorporating key generators and layered encryption methods can significantly strengthen data protection. For instance, Diana and Zebua (2018) as well as Boru et al. (2024) showed that combining the Vigenère substitution cipher with RC4 produced more random ciphertext, making it more difficult for third parties to decrypt. Similarly, the study by Widarma et al. (2019) on data security techniques using the Vigenère Cipher and ECB found that integrating the Vigenère Cipher with ECB yields strong performance and improved security outcomes. Unlike previous studies, this research combines two encryption and decryption algorithms in a layered form (Vigenère Cipher and ECB) and adds separate key generators at each stage, namely RC4 for the Vigenère Cipher and LFSR for ECB.

Based on the previous explanation, the author conducted a study entitled "Combination of the Vigenère Cipher Algorithm using Rivest Cipher 4 Key Generator and Electronic Code Book using Linear Feedback Shift Register Key Generator," which aims to demonstrate the application of the combination of these two cryptographic algorithms in the encryption and decryption process, as well as its implementation in a Graphical User Interface (GUI) as a means of visualizing the cryptographic process. This algorithm is chosen to prove that the combination of classical and modern algorithms in message encryption is capable of maintaining data authenticity.

METHOD

This research used a combination of classical and modern cryptographic techniques, specifically the Vigenère Cipher algorithm integrated with the Rivest Cipher 4 (RC4) key generator and the Electronic Code Book (ECB) algorithm combined with the Linear Feedback Shift Register (LFSR) key generator.

Exclusive OR (XOR)

According to Akmal et al. (2024), the XOR algorithm is a frequently used algorithm in classical cryptography. This algorithm utilizes bit-based operations for its processing and works based on the XOR logic principle. For example, when statements A and B are false (0), then A $\oplus$ B is false (0). For more details, the logic table for the XOR operation can be seen in 1.

Table 1: XOR LOGIC

A	B	A $\oplus$ B
0	0	0
0	1	1
1	0	1
1	1	0

Vigenère *Cipher*

The Vigenère Cipher encrypts plaintext by shifting each letter in the message according to a key value based on its position in the alphabetical sequence. Polyalphabetic substitution, also referred to as compound-alphabetic substitution, applies a distinct key to each character in the message. When the key length is shorter than the plaintext length, the key is repeated until it matches the length of the plaintext (Haris and Ariyus, 2020).

According to Saputra et al. (2017), when the Vigenère Cipher is implemented in computer-based calculations, the characters used correspond to the total number of characters in the ASCII table, namely 256 characters. Consequently, the equation is modified as follows:

1. Encryption

$$C_i = (P_i + K_i) \mod 256 \tag{1}$$

2. Decryption

$$P_i = (C_i - K_i) \mod 256 \tag{2}$$

defined as follows:

C_i : the i^{th} ciphertext value,
P_i : the i^{th} plaintext value,
K_i : the i^{th} key value,

Rivest *Cipher* 4 (RC4)

According to Boru et al. (2024), Rivest Cipher 4 (RC4) is an encryption algorithm developed by Ron Rivest in 1983. Its keystream generation process consists of two main stages: the Key-Scheduling Algorithm (KSA) and the Pseudo-Random Generation Algorithm (PRGA).

As described by Diana and Zebua(2018), the key generation procedure in RC4 is defined as follow. Let *i*, *j* range from 0 to 255, S_i is the value of the S array at the i^{th} position, S_j is the value of the S array at the j^{th} position, T_i is the value of the T array at the i^{th} position, p is the key length, and K_a is the a^{th} key value with $a = i \mod p$.

1. *Key-Scheduling Algorithm* (KSA)

   (a) S array initialization

$$S_i = i \tag{3}$$

   (b) T array initialization

$$T_i = K_a \tag{4}$$

   (c) S array permutation

      i. *j* value

$$j_{baru} = (j_{lama} + S_i + T_i) \mod 256 \tag{5}$$

      ii. Swap S_i with S_j

2. *Pseudo-Random Generation Algorithm* (PRGA)
   In this process, *i* runs from 0 to the length of the plaintext $-\ 1$, with $j = i$.

   (a) *i* value

$$i_{next} = (i_{prev} + 1) \mod 256 \tag{6}$$

   (b) *j* value

$$j_{next} = (j_{prev} + S_i) \mod 256 \tag{7}$$

   (c) Swap $S_{i_{\text{next}}}$ with $S_{j_{\text{prev}}}$

   (d) *t* value

$$t = (S_{i_{\text{next}}} + S_{j_{\text{prev}}}) \mod 256 \tag{8}$$

   (e) The i^{th} generated key value

$$K_i = S_t \tag{9}$$

Electronic Code Book (ECB)

According to Santoso et al. (2024), the Electronic Code Book (ECB) method is a modern cryptographic algorithm that divides plaintext into independent blocks, ensuring that an error in one block does not affect the others. A key characteristic of ECB is that identical plaintext blocks are always encrypted into identical ciphertexts.

The simple encryption and decryption processes are expressed in Equation (10) dan (11).

1. Encryption
$$E(P_i) = (P_i \oplus K_i) \tag{10}$$
C_i, or the ciphertext, is obtained by shifting $E_k(P_i)$ to the left by 1 bit.

2. Decryption
The binary code of the *ciphertext* is first shifted to the right by 1 bit.
$$D(C_i) = (C_i \oplus K_i) \tag{11}$$

Linear Feedback Shift Register (LFSR)

According to Nahading and Wowor (2023),a Linear Feedback Shift Register (LFSR) is a key generation function that accepts an input and returns a function value to a shift register. This mechanism enables the LFSR to produce random bit sequences with a maximum period, making it suitable for various cryptographic applications.

The LFSR can be efficiently represented in polynomial form over the variable x, with n denoting the order of the polynomial, as shown in Equation (12)

$$P(x) = x^n + c_{n-1}x^{n-1} + ... + c_1x + 1 \tag{12}$$

defined as follows:

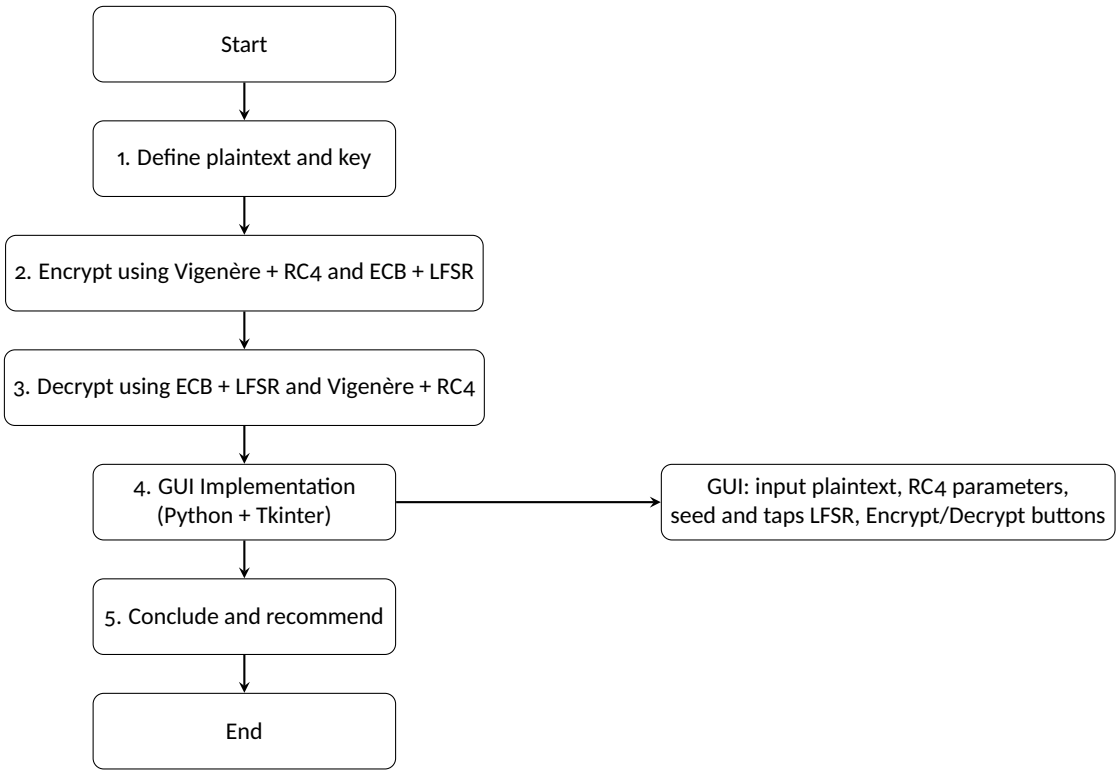
$P(x)$: polynomial function in the variable x,
n : length of the shift register,
$x^n, x^{n-1}, ..., x$: the position of the *bit* in the shift register, and
$c_{n-1}, ..., c_1$: binary coefficient (with a value of 0 or 1).

Graphical User Interface (GUI)

According to Lutz (2001), a GUI is an important component in modern software development that includes elements such as windows, buttons, and menus used to interact with applications. The widespread use of GUIs aims to increase flexibility and ease in operating software for users.

Tkinter is one of the portable and open-source GUI libraries that has become the de facto standard in the development of graphical user interfaces in the Python programming language. Tkinter enables the creation of simple interfaces quickly and flexibly, and can be further developed using other component frameworks available in Python.

The stages of analysis conducted in this study are as follows.



RESULTS

This section presents the encryption and decryption processes of the Vigenère Cipher Algorithm combined with the Rivest Cipher 4 (RC4) Key Generator and the Electronic Code Book (ECB) with the Linear Feedback Shift Register (LFSR) Key Generator, as well as the program implementation in Python.

Encryption Process

At this stage, the encryption process was carried out to convert the original message, known as plaintext, into an unintelligible code by combining the Vigenère Cipher Algorithm with the RC4 key generator and the ECB Algorithm with the LFSR key generator. For instance, the plaintext "KRIPTOGRAFI" was provided, with the initial key "KUNCI."

1. The first step in the encryption process was to convert the plaintext and the initial key into decimal values based on the ASCII table. Let i denote the index, where P_i represented the plaintext in decimal form at the i^{th} position, and K_i represented the key in decimal form at the same index. The results of the plaintext and key conversions are presented in Table 2 dan 3.

Table 2: Initial Plaintext Conversion

i	Character	P_i
0	K	75
1	R	82
2	I	73
3	P	80
4	T	84
5	O	79
6	G	71
7	R	82
8	A	65
9	F	70
10	I	73

Table 3: Initial Key Conversion in the Encryption Process

i	Character	K_i
0	K	75
1	U	85
2	N	78
3	C	67
4	I	73

2. Key generation was performed using the RC4 algorithm, with the input consisting of the initial key's decimal values as shown in 3. The algorithm proceeded through two main stages: the Key-Scheduling Algorithm (KSA) and the Pseudo-Random Generation Algorithm (PRGA).

   (a) *Key-Scheduling Algorithm*
   At this stage, array S was initialized. Let i range from 0 to 255. The values of the S array are modified according to Equation (3) and are presented in Table 4.

Table 4: S Array in the Encryption Process

$i = 0$–7	0	1	2	3	4	5	6	7
$i = 8$–15	8	9	10	11	12	13	14	15
$i = 16$–23	16	17	18	19	20	21	22	23
$i = 24$–31	24	25	26	27	28	29	30	31
$i = 32$–39	32	33	34	35	36	37	38	39
...	...	...	...	...	...	...	...	...
$i = 248$–255	248	249	250	251	252	253	254	255

Array T was then initialized using the decimal values of the keywords from Table 3 as presented in Equation (3), with the results shown in Table 5.

Table 5: T Array in the Encryption Process

$i = 0$–7	75	85	78	67	73	75	85	78
$i = 8$–15	67	73	75	85	78	67	73	75
$i = 16$–23	85	78	67	73	75	85	78	67
$i = 24$–31	73	75	85	78	67	73	75	85
$i = 32$–39	78	67	73	75	85	78	67	73
...	...	...	...	...	...	...	...	...
$i = 248$–255	67	73	75	85	78	67	73	75

Subsequently, array S was permuted based on Equation (5). through iterations of i and j_{prev} from 0 to 255. The final permuted S array is presented in Table 6.

Table 6: Result of S Array Permutation

$i = 0$–7	168	161	229	42	132	20	193	87
$i = 8$–15	207	33	118	214	48	204	240	49
$i = 16$–23	95	194	74	162	21	23	149	84
$i = 24$–31	142	242	40	99	41	143	248	195
$i = 32$–39	218	100	145	1	120	73	3	167
...	...	...	...	...	...	...	...	...
$i = 248$–255	90	79	224	158	191	211	160	78

   (b) *Pseudo-Random Generation Algorithm*
   At this stage, key generation of i and j was executed from iterations 0 to the number of plaintext characters minus one, beginning with i_{prev} = 0 and j_{prev} = 0, as defined in Equations (6) until (9). Upon completion of the PRGA stage, the final RC4 key generation results were obtained, as presented in Table 7.

Table 7: Final Result of RC4 Key Generation in the Encryption Process

i	K_i
0	205
1	3
2	156
3	20
4	221
5	18
6	176
7	173
8	230
9	211
10	67

3. The Vigenère Cipher encryption process was performed based on Equation (1) using the key generated by the RC4 key generator. Iterations were executed from $i = 0$ to $i = 10$, corresponding to the plaintext character indices. The first iteration was conducted at $i = 0$, with P_i derived from Table 2 and K_i from Table 7. Upon completion of the Vigenère Cipher encryption, the first ciphertext was obtained in decimal form, as presented in Table 8.

Table 8: Ciphertext Result of the Vigenère Cipher Encryption Algorithm

i	K_i
0	24
1	85
2	229
3	100
4	49
5	97
6	247
7	255
8	39
9	25
10	140

4. In the subsequent stage, the second key was generated using the LFSR algorithm.

   (a) Parameters Determination
   The parameters were defined as $n = 4$, seed: 1010, and the polynomial equation: $P(x) = x4 + x + 1$, indicating that the Exclusive OR (XOR) operation was performed on the 4^{th} and 1^{st} bit positions.

   (b) LFSR Simulation
   At this stage, a key generation simulation was conducted using the LFSR algorithm. In the first iteration, the seed was initialized as 1010. The XOR operation was applied to the 4^{th} (x_4) and 1^{st} (x_1) bit positions, and the result was stored in the 4^{th} bit for the second iteration. Subsequently, the bit values from the first iteration were shifted one bit to the right to form the 3^{rd} (x_3), 2^{nd} (x_2), and 1^{st} (x_1) bit positions in the next iteration. The results of the LFSR simulation are presented in Table 9.

Table 9: Key Generation using LFSR in the Encryption Process

Iteration	x_4	x_3	x_2	x_1
1	1	0	1	0
2	1	1	0	1
3	0	1	1	0
4	0	0	1	1
5	1	0	0	1
6	0	1	0	0
7	0	0	1	0
8	0	0	0	1

9	1	0	0	0
10	1	1	0	0
11	1	1	1	0
12	1	1	1	1
13	0	1	1	1
14	1	0	1	1
15	0	1	0	1

Thus, the key generated by the LFSR key generation process is 010110010001111.

5. The ciphertext obtained from the first encryption (Table 8) was converted into binary form according to the ASCII table, as presented in Table 10.

Table 10: Conversion of Ciphertext into Binary Form

i	C_i	Binary Form
0	24	00011000
1	85	01010101
2	229	11100101
3	100	01100100
4	49	00110001
5	97	01100001
6	247	11110111
7	255	11111111
8	39	00100111
9	25	00011001
10	140	10001100

6. The key generated using the LFSR was extended from its original 15 bits to a total of 88 bits to match the length of the binary ciphertext in Table 8. The extended key was then divided into blocks, as shown in Table 11.

Table 11: LFSR Key Extension in the Encryption Process

i	K_i
0	01011001
1	00011110
2	10110010
3	00111101
4	01100100
5	01111010
6	11001000
7	11110101
8	10010001
9	11101011
10	00100011

7. The Electronic Code Book (ECB) encryption process was performed using the binary plaintext (P_i) from Table 10 along with the key (K_i) generated and extended through the LFSR key generator, as shown in Table 11. Equation (10) was applied to compute the encryption process, and the detailed steps are presented in Table 12.

Table 12: ECB Encryption Stage

i	0	1	2	3
P_i	00011000	01010101	11100101	01100100
K_i	01011001	00011110	10110010	00111101
$P_i \oplus K_i$	01000001	01001011	01010111	01011001
Bit Shifting	10000010	10010110	10101110	10110010

i	4	5	6	7
P_i	00110001	01100001	11110111	11111111
K_i	01100100	01111010	11001000	11110101
$P_i \oplus K_i$	01010101	00011011	00111111	00001010
Bit Shifting	10101010	00110110	01111110	00010100

i	8	9	10
P_i	00100111	00011001	10001100
K_i	10010001	11101011	00100011
$P_i \oplus K_i$	10110110	11110010	10101111
Bit Shifting	01101101	11100101	01011111

Thus, the ciphertext resulting from the second encryption stage was obtained, as presented in Table 13.

Table 13: Ciphertext Result of the ECB Encryption Algorithm

i	C_i
0	10000010
1	10010110
2	10101110
3	10110010
4	10101010
5	00110110
6	01111110
7	00010100
8	01101101
9	11100101
10	01011111

Proses Dekripsi

At this stage, the decryption process was conducted using the inverse order of the encryption algorithms, namely the ECB algorithm with the LFSR key generator followed by the Vigenère Cipher with the RC4 key generator. The ciphertext utilized was the final encryption result, as presented in Table 14.

1. The first key was generated using the LFSR algorithm.
   (a) Parameters Determination
   $n = 4$, Seed: 1010, Polynomial Equation: $P(x) = x4 + x + 1$, indicating that the bit positions where the Exclusive OR (XOR) operation was performed were the 4th and 1^{st} bits.
   (b) Simulasi LFSR
   In this stage, a key generation simulation was conducted using the LFSR algorithm, following the same procedure applied during the encryption process.

Table 14: Key Generation using LFSR in the Decryption Process

Iteration	x_4	x_3	x_2	x_1
1	1	0	1	0
2	1	1	0	1
3	0	1	1	0
4	0	0	1	1
5	1	0	0	1
6	0	1	0	0

7	0	0	1	0
8	0	0	0	1
9	1	0	0	0
10	1	1	0	0
11	1	1	1	0
12	1	1	1	1
13	0	1	1	1
14	1	0	1	1
15	0	1	0	1

As a result, the key produced by the LFSR was 0101100100001111.

2. The key generated using the LFSR algorithm was extended by adding bits, increasing from the original 15 bits to a total of 88 bits, corresponding to the length of the binary ciphertext in Table 13. The key was subsequently divided into blocks, as shown in Table 15.

Table 15: LFSR Key Extension in the Decryption Process

i	K_i
0	01011001
1	00011110
2	10110010
3	00111101
4	01100100
5	01111010
6	11001000
7	11110101
8	10010001
9	11101011
10	00100011

3. The Electronic Code Book (ECB) decryption process was performed using the binary plaintext (P_i) in Table 13 and the extended key (K_i) generated by the LFSR key generator, as presented in Table 15. Equation (11) was applied for the computation of this algorithm, and the ECB decryption procedure is shown in Table 16.

Table 16: ECB Decryption Stage

i	0	1	2	3
Geser *Bit*	01000001	01001011	01010111	01011001
C_i	01000001	01001011	01010111	01011001
K_i	01011001	00011110	10110010	00111101
$C_i \oplus K_i$	00011000	01010101	11100101	01100100

i	4	5	6	7
Geser *Bit*	01010101	00011011	00111111	00001010
C_i	01010101	00011011	00111111	00001010
K_i	01100100	01111010	11001000	11110101
$C_i \oplus K_i$	00110001	01100001	11110111	11111111

i	8	9	10
Geser *Bit*	10110110	11110010	10101111
C_i	10110110	11110010	10101111
K_i	10010001	11101011	00100011
$C_i \oplus K_i$	00100111	00011001	10001100

Consequently, the ciphertext resulting from the second encryption was obtained, as presented in Table 17.

Table 17: Ciphertext Result of the ECB Decryption Algorithm

i	C_i
0	00011000
1	01010101
2	10101110
3	01100100
4	00110001
5	01100001
6	11110111
7	11111111
8	00100111
9	00011001
10	10001100

4. The second initial key used was the word "KUNCI." The ECB-decrypted plaintext and the second initial key were converted into decimal numbers according to the ASCII table. Let i denote the index, where C_i represents the ciphertext (ECB-decrypted plaintext) in decimal form with the i^{th} index, and K_i denotes the key in decimal form with the i^{th} index. The results of the ciphertext and keyword conversions are presented in Tables 18 and 19.

Table 18: Conversion of Ciphertext from ECB Decryption Results

i	Binary Form	C_i
0	00011000	24
1	01010101	85
2	10101110	229
3	01100100	100
4	00110001	49
5	01100001	97
6	11110111	247
7	11111111	255
8	00100111	39
9	00011001	25
10	10001100	140

Table 19: Second Initial Key Conversion in the Decryption Process

i	Character	K_i
0	K	75
1	U	85
2	N	78
3	C	67
4	I	73

5. Key generation was performed using the RC4 algorithm, with the initial key decimal values from Table 3 as input. This algorithm consisted of two processes: the Key-Scheduling Algorithm (KSA) and the Pseudo-Random Generation Algorithm (PRGA).

   (a) *Key-Scheduling Algorithm*
   At this stage, the array S was initialized, with i ranging from 0 to 255. The values in array S were assigned according to Equation (3), as shown in Table 20.

Table 20: S Array in the Decryption Process

$i = 0$-7	0	1	2	3	4	5	6	7
$i = 8$-15	8	9	10	11	12	13	14	15
$i = 16$-23	16	17	18	19	20	21	22	23
$i = 24$-31	24	25	26	27	28	29	30	31
$i = 32$-39	32	33	34	35	36	37	38	39
...	...	...	...	...	...	...	...	...
$i = 248$-255	248	249	250	251	252	253	254	255

Subsequently, array T was initialized to contain the decimal values of the keywords from Table 3, following the same equation, and is presented in Table 21.

Table 21: T Array in the Decryption Process

$i = 0$-7	75	85	78	67	73	75	85	78
$i = 8$-15	67	73	75	85	78	67	73	75
$i = 16$-23	85	78	67	73	75	85	78	67
$i = 24$-31	73	75	85	78	67	73	75	85
$i = 32$-39	78	67	73	75	85	78	67	73
...	...	...	...	...	...	...	...	...
$i = 248$-255	67	73	75	85	78	67	73	75

Array S was then randomized through permutation based on Equation (5). This permutation process was iterated from $i = 0$ to 255, with corresponding j_{prev} values ranging from 0 to 255. Upon completion of the permutation stage, the resulting S array permutation was obtained, as presented in Table 22.

Table 22: Result of S Array Permutation

$i = 0$-7	168	161	229	42	132	20	193	87
$i = 8$-15	207	33	118	214	48	204	240	49
$i = 16$-23	95	194	74	162	21	23	149	84
$i = 24$-31	142	242	40	99	41	143	248	195
$i = 32$-39	218	100	145	1	120	73	3	167
...	...	...	...	...	...	...	...	...
$i = 248$-255	90	79	224	158	191	211	160	78

(b) *Pseudo-Random Generation Algorithm*
In this stage, key generation was performed with i and j iterated from 0 to the total number of plaintext characters -1, starting with $i_{\text{prev}} = 0$ and $j_{\text{next}} = 0$, as defined by Equations 6 until 9. After completing the PRGA process, the final RC4 key generation results were obtained and are shown in Table 23.

Table 23: Final Result of RC4 Key Generation in the Decryption Proces

i	K_i
0	205
1	3
2	156
3	20
4	221
5	18
6	176
7	173
8	230
9	211
10	67

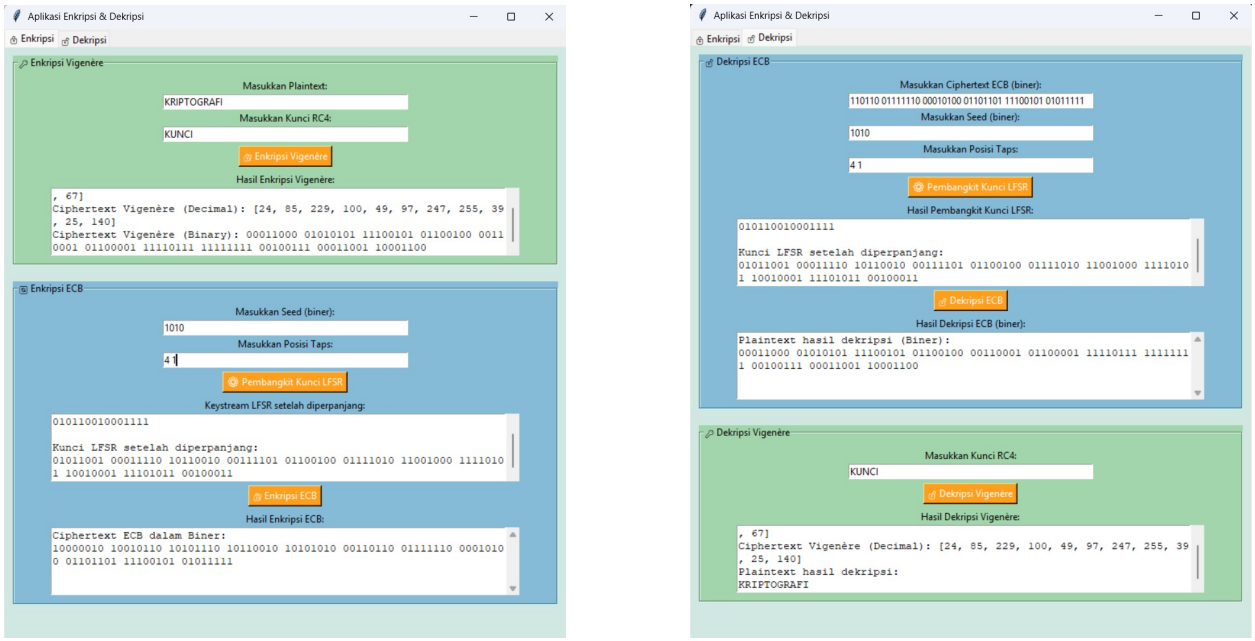
6. The Vigenère Cipher decryption process was performed based on Equation (2), utilizing the key generated through the RC4 key generator. Iterations were executed from $i = 0$ to $i = 10$, corresponding to the plain-

text character indices. The first iteration was conducted at $i = 0$, with C_i derived from Table 18 and K_i from Table 23. Upon completion of the decryption stage, the resulting plaintext in decimal form was obtained, as presented in Table 24.

Table 24: Ciphertext Result of the Vigenère Cipher Decryption Algorithm

i	P_i	Character
0	75	K
1	82	R
2	73	I
3	80	P
4	84	T
5	79	O
6	71	G
7	82	R
8	65	A
9	70	F
10	73	I

7. In this stage, the program implementation of the study was developed, combining the Vigenère Cipher algorithm with the RC4 key generator and the ECB algorithm with the LFSR key generator in a desktop-based application. The Graphical User Interface (GUI) program was implemented and designed in Python using the Tkinter library. The GUI was developed to ensure user-friendly operation and includes two main features: encryption and decryption menus.



(a) Menu Enkripsi

(b) Menu Dekripsi

Figure 1: Tampilan Menu Enkripsi dan Dekripsi

DISCUSSION

The results of the study show that the encryption and decryption process of the plaintext through two stages, namely the combination of the Vigenère Cipher using a key generated from RC4, and the Electronic Code Book (ECB) using a key generated from LFSR, was successfully decrypted back into its original form, which is the word "KRIPTOGRAFI". Thus, the combination of algorithms used in this study is able to maintain data consistency during the process of encoding and recovering the message. This study also successfully implemented the combination of these algorithms in the form of a Graphical User Interface (GUI). This interface displays the stages of the encryption and decryption process in a simple and interactive way, so it can help users understand the layered data security process.

Overall, the results of this study are in line with its objectives. It successfully demonstrates how two cryptographic algorithms can be combined by adding separate key generators at each stage and implementing them in a GUI. This research shows that mathematical concepts such as modulo operations, XOR logic, and binary numbers are not only theoretical but can be directly applied in real encryption and decryption processes. In addition, this study also proves that mathematics can be used to solve real-world problems, especially in securing data.

CONCLUSION

This study successfully demonstrated the combination of a classical cryptographic algorithm, namely the Vigenère Cipher, using a key generated through the RC4 algorithm, and a modern cryptographic algorithm, namely ECB, using a key generated through the LFSR algorithm. The encryption and decryption process of the message shows that the combination of these algorithms was successfully carried out because it was able to maintain the authenticity of the data or message. The results of this study were also implemented in the form of a GUI, thus helping users to understand the working stages of these cryptographic algorithms in data security.

ACKNOWLEDGMENT

The author expresses appreciation to the Mathematics Study Program, Faculty of Mathematics and Natural Sciences, Mulawarman University, for the support provided in the implementation of this research.

AI ACKNOWLEDGMENT

The authors declare that generative AI or AI-assisted technologies were not used in any way to prepare, write, or complete this manuscript. The authors confirm that they are the sole authors of this article and take full responsibility for the content therein, as outlined in COPE recommendations.

INFORMED CONSENT

The authors have obtained informed consent from all participants.

CONFLICT OF INTEREST

The authors declare that there is no conflict of interest.

REFERENCES

Akmal, D. A., Dhani, D. M., & Syahila, F. (2024). Kombinasi kriptografi modern dalam keamanan pesan teks. *Saturnus: Jurnal Teknologi Dan Sistem Informasi*, *2*(3), 119–128. https://doi.org/10.61132/saturnus.v2i3.204

Amijaya, F. D. T., Syaripuddin, S., A'yun, Q. Q., Nasution, Y. N., Wasono, W., & Huda, M. (2022). Hill cipher algorithm using two keywords and 94 ASCII characters. *AIP Conference Proceedings*, *2668*(1). https://doi.org/10.1063/5.0111696

Andayani, S., & Agista, D. S. (2017). Kriptografi klasik teknik subsitusi untuk keamanan data menggunakan VB. Net 2008. *Matrix: Jurnal Manajemen Teknologi dan Informatika*, *4*(2), 75.

Boru, M., Sultani, A. A., Mauko, A. Y., Mola, S. A. S., Letelay, K., & Sihotang, D. M. (2024). Kombinasi cipher subtitusi (Beaufort dan Vigenere) menggunakan pembangkit kunci RC4 pada kriptografi video audio video interlaced (AVI). *Telekontran: Jurnal Ilmiah Telekomunikasi, Kendali dan Elektronika Terapan*, *12*(2), 187–195. https://doi.org/10.34010/telekontran.v12i2.14167

Diana, M., & Zebua, T. (2018). Optimalisasi Beaufort Cipher menggunakan pembangkit kunci RC4 dalam penyandian SMS. *J-SAKTI (Jurnal Sains Komputer dan Informatika)*, *2*(1), 12–22. https://doi.org/10.30645/j-sakti.v2i1.52

Fairuzabadi, M. (2010). Implementasi kriptografi klasik menggunakan Borland Delphi. *Jurnal Dinamika Informatika*, *4*(2), 65–78.

Giawa, Y. N. I. (2022). Implementasi algoritma RC4A dalam pengamanan citra digital. *Journal of Computing and Informatics Research*, *2*(1), 1–9. https://doi.org/10.47065/comforch.v2i1.348

Haris, C. A., & Ariyus, D. (2020). Kombinasi dan modifikasi vigenere cipher dan hill cipher menggunakan metode hybrid kode pos, trigonometri, dan konversi suhu sebagai pengamanan pesan. *Inform. Mulawarman J. Ilm. Ilmu Komput*, *15*(2), 90–96. https://doi.org/10.30872/jim.v15i2.3746

Lutz, M. (2001). *Programming python*. O'Reilly Media, Inc.

Nahading, T. S., & Wowor, A. D. (2023). Desain pembangkit kunci LFSR dengan skema A5/1 menggunakan empat blok bit fungsi XOR. *Kesatria: Jurnal Penerapan Sistem Informasi (Komputer dan Manajemen)*, *4*(2), 409–419. https://doi.org/10.30645/kesatria.v4i2.177

Santoso, K. A., Dhani, E., & Pradjaningsih, A. (2024). Pengaman teks dengan kombinasi metode electronic code book (ecb) dan kode seven segment display. *Jurnal Teknologi Informasi Dan Ilmu Komputer*, *11*(1), 85–94. https://doi.org/10.25126/jtiik.20241117448

Saputra, I., Hasibuan, N. A., & Rahim, R. (2017). Vigenere cipher algorithm with grayscale image key generator for secure text file. *International Journal of Engineering Research and Technology (IJERT)*, *6*(1), 266–269.

Setyawati, N. Y., Khofid, A. N., Rundi, A. U., & Wati, V. (2021). Modifikasi kriptografi klasik kombinasi metode Vigenere Cipher dan Caesar Cipher. *Journal of Smart System*, *1*(1), 1–8. https://doi.org/10.36728/jss.v1i1.1601

Sulaiman, R., & Isnanto, B. (2018). Peningkatan keamanan pesan dengan kriptografi RC4 dan steganografi LSB pada file JPEG. *Konferensi Nasional Sistem Informasi (KNSI) 2018*.

Widarma, A., Siregar, H. F., & Irawan, M. D. (2019). Teknik keamanan data menggunakan Vigenere Cipher dan Electronic Code Book (ECB). *J-SAKTI (Jurnal Sains Komputer dan Informatika)*, *3*(2), 393–400. https://doi.org/10.30645/j-sakti.v3i2.157

Widya, H. (2018). Sistem pembelajaran dan pemahaman algoritma Electronic Code Book (ECB) menggunakan metode Computer Assisted Instruction (CAI). *JET (Journal of Electrical Technology)*, *3*(3), 149–155.