THE 20TH EDITION OF THE INTERNATIONAL CONFERENCE
EUROPEAN INTEGRATION
REALITIES AND PERSPECTIVES

Implementation of a Symmetric Block Cipher Technique in Scilab

Diana-Rodica Munteanu[1], **Ciprian Rusescu**[2], **Mariana Geanina Zaharia**[3]

Abstract: This paper presents an innovative block cipher encryption / decryption technique to enhance data security in digital communications. The proposed algorithm divides the plaintext into blocks applying a series of arithmetic and logical operations to ensure confidentiality. The method's simplicity facilitates its implementation in resource-constrained environments, such as ebedded systems. A practical case study demonstrates the encryption and decryption of a text file, illustrating the algorithm's functionality. Future research will focus on evaluating the method's security strength comparing also its performance with existing encryption standards.

Keywords: Block Cipher; Encryption; Decryption; Data Security; Cryptography

JEL Classification: C63; D83; O33.

1. Introduction

Global economic development, most pronounced in recent decades, would have been inconceiveble without technological advancement. In the digital era, ensuring the confidentiality and integrity of information has become a critical concern, especially given the exponential growth of data exchange through various communication channels such as emails, social media platforms, and online financial services. As cyber threats become increasingly sophisticated, the need of robust encryption techniques, strong enough to protect sensitive data from unauthorized access are becoming more pressing than ever. Block cipher algorithms, known for their efficiency and strength, play a capital role in modern cryptographic systems by securing data transmission and storage. However, designing encryption algorithms able to combine a high security level with computational efficiency remains a significant challenge, particularly for resource-constrained environments.

The present paper's main goal consists in implementing an encryption / decryption innovative technique, the right answer at both security and efficiency concerns. Inspired by classical ciphers such as Vigenère and modern approaches rooted in difference equations, the algorithm combines mathematical rigor with

[1]Senior Lecturer, PhD, Faculty of Psychology and Education Sciences, Ovidius University of Constanta, Romania, Address: 127 Mamaia Blvd., Constanta, Romania, E-mail: diana_munteanu@365.univ-ovidius.ro.

[2] Senior Lecturer, PhD, Department of Informatics and Social Assistence, Danubius International University, Galati. Romania, Address: 3 Galati Blvd., Galati, Romania, Corresponding author: ciprian.rusescu@univ-danubius.ro.

[3] PhD, "Albatros" School Center for Inclusive Education, Constanta, Romania, Address: 55 Barbu Ştefănescu Delavrancea St., Constanta, Romania, E-mail: geanina.zaharia@365.univ-ovidius.ro.

practical applicability. The main objective is to offer a method that can be easily implemented, tested, and adapted for various applications, ranging from secure messaging to data protection in embedded systems. The algorithm has been developed and tested in Scilab, with examples proving its validity and potential utility in safeguarding digital communication.

The best method used to protect the information transmission through various channels (email, social media applications such as Facebook, WhatsApp, Instagram, etc.) is represented by its encryption. This way, an ordinary adversary altering it seems unlikely (in the launched attack scenario) because he can see the code without reading what is hiding behind it (Wael & Hassene, 2016). The encryption process involves various cryptographic techniques, transforming plain text into a ciphered one (cryptogram) using specific keys.

Cryptography uses mathematical methods but also computer science algorithms. Encryption techniques can be generally split into two classes, called “symmetric” and “asymmetric”. First one specific feature is represented by the usage of the same key for both processes (encryption/decryption), whilst in second scenario are necessary two different keys: the public key is well known by any people interested in encrypting a message, but the private one has to be kept secret and used only for decryption (Bazeer Ahamed & Krishnamoorthy, 2024). Symmetric algorithms are easier to use while asymmetric systems offer higher security.

Symmetric algorithms use two different techniques: substitution or transposition. Substitution involves replacing plain text letters with others from the same alphabet. The cipher is further split between a monoalphabetic version (which uses a letter-by-letter codification system, vulnerable to frequency analysis) (Conrad, 2011) and a polyalphabetic one (each plain text character has multiple ciphering possibilities).

An interesting polyalphabetic cipher was invented in 1593 by the baron Blaise de Vigenère, a French mathematician of the 16th century. Due to its resistance at frequency letter attacks, it had considerable importance/usage, being considered safe for centuries, before being kneeled by Friedrich Kasiski (for a long enough text message within which the key appears several times) (Menezes et al., 2018; Shannon, 2001, 1948). Despite that, it was further included in other powerful encryption algorithms as Advanced Encryption Standard (AES) (Sanchez-Avila & Sanchez-Reillol, 2001). Short message service (SMS) encryption represents another utilization of the Vigenère cipher (Fahrianto et al., 2014). Compared with any monoalphabetic cipher, the required crack time is considered long enough to assure the message deletion, after being red by the receiver (Mandal & Deepti, 2016).

2. Literature Review

Various approaches to the Vigenère cipher were used over time. One of them proposes increasing the plain text length by adding a random bit after each reading byte before starting the cryptogram generation (Wilson & Garcia, 2006). This initial message stuffing guarantees Kasiski attack failure, the main shortcoming consisting of ciphered text increasing length. Different other interesting ideas refers at periodically key changing during the encryption process (Kester, 2012), letter substitution with any other specific symbols (Bhardwaj, 2012) and innovative combinations of Vigenere cipher with Linear Feedback Shift Register key (Razzaq et al., 2012), Caesar cipher (Omolara et al., 2014), stream ciphers (Ali and Sarhan, 2014) or Hill system (Touil, 2020).

Modern encryption techniques are based on block ciphers and well-known encryption ciphers like Vigenère. This cipher type includes both an encryption / decryption algorithm, the key being provided for various computations to encrypt a plain text block. The ciphertext block is passed through a decryption operation to retrieve the message sent to the receiver (Qadir & Varol, 2019). Usually, these algorithms are based on mathematical results in number theory. For the same purpose, Flaut (2019) presents several applications of a k degree difference equation, an encrypting / decrypting algorithm being given as example. Additionally, by employing the matrix associated with the k degree ~~k~~ difference equation, an application in the field of Coding Theory has been demonstrated, forming the basis of this article.

Inspired by the Flaut's article, Ragab et al. (2023) propose a symmetric cipher "M-XXTEA" with a simple construction compared to AES, efficient for protecting smart IoT devices (Internet of Things devices). M-XXTEA ensures data confidentiality during transmission over the internet, being a robust version of the XXTEA cipher, using an improved S-box and a chaotic key generator system. The encryption keys are dynamic for each text block, providing stronger security than XXTEA and AES systems. The authors state sustain that Performed experiments allow the authors to sustain that M-XXTEA is to be 60% more efficient respectively 57% faster than AES, making it suitable for protecting IoT devices and innovative systems.

The RSA (Rivest-Shamir-Adleman) algorithms are asymmetric encryption algorithms that use different keys for encryption and decryption (Milanov, 2009), the public one being used for encryption, while the private for decryption. These keys consist in a pair of large prime numbers (mathematically generated) and possess specific properties, able to guarantee communication security. The public key is freely distributed to any interested people in encrypting messages, while the private key allows only to its owner to decrypt those messages. Due to its security and strong mathematical properties, RSA is used in various applications, such as secure communications, user authentification, and digital signatures. However, RSA may prove to be computationally inefficient in a very large message, so faster symmetric algorithms are used for operations involving large data volumes.

The AES (Advanced Encryption Standard) algorithm is one of the world's most widely used and recognized symmetric encryption algorithms (Bani Yassein et al., 2017; Abdullah, 2017; Mali et al., 2005). It has been adopted as standard by governments and organizations worldwide, being able to operate with three different key sizes, 128 bits, 192 bits, and 256 bits. The key plays a central role in the encryption and decryption process, its length being crucial for the algorithm's security. AES uses a symmetric cipher, assuring this way an encryption process high efficiency in the presence of a rigorous key management system able to guarantee its privacy. Due to its features, AES is used in various applications and scenarios, from protecting personal data on mobile devices to securing communications and sensitive data within organizations.

3. Developing the Algorithm

Starting from the cryptographic example provided by Flaut (2019), we implemented an encryption/decryption algorithm in Scilab adding a randomly generated key to make it harder to break. We test the algorithm for different lengths messages to be sure it works properly.

We state first Flaut's algorithms (2019) to offer the reader some technical context of the implementation. In this respect, we need to recall the main results that lead to the cryptography application.

Let $a_1, \dots, a_k$ be arbitrary integers and consider the general k-terms recurrence, $n, k \in \mathbb{N}, k \geq 2, n \geq k$,

$$d_n = a_1 d_{n-1} + a_2 d_{n-2} + \cdots + a_k d_{n-k}, d_0 = d_1 = \cdots = d_{k-2} = 0, d_{k-1} = 1, a_k \neq 0. \quad (1)$$

A sequence of positive integers, $(d_m)_{m\geq 0}, d_m \geq 0$, is complete if and only if each $n \in N$ can be written as $\sum_{i=1}^{t} c_i d_i$, where c_i is either zero or one. Brown (1961) demonstrated that a non-decreasing sequence of natural numbers, with $d_1 = 1$, is complete only if the following relation holds:

$$d_{k+1} \leq 1 + \sum_{i=1}^{k} d_i, \forall k \in 1,2, \ldots$$

Positive integer sequences could be completed or not completed. Zeckendorf (1972) proved that each natural number can be written as a unique sum of non-consecutive Fibonacci numbers, meaning the sequence of Fibonacci numbers is complete. An example of a not complete sequence can be considered the recurrence $(d_m), m \geq 0$:

$$d_m = 4d_{m-1} + 3d_{m-2}, d_0 = 0, d_1 = 1.$$

Considering the set of natural numbers $\mathbb{N} = \{0,1,2, \ldots \ldots\}$, we generalize the notion of complete sequence, and we will provide some applications.

Definition 1. (Flaut, 2019) A sequence of positive integers, $(d_m)_{m\geq 0}, d_m \geq 0$, is called *generally complete* (or *g-complete*) if and only if, for each natural number $n \geq 0$, there is a natural number $q, q \geq 0$, such that n can be written, in a unique way, in the form

$$n = d_0 a_0 + a_1 d_1 + \cdots + a_q d_q, a_0, \ldots, a_q \in \mathbb{N}, a_q \neq 0.$$

Theorem 1. (Flaut, 2019) *The sequence of positive integers $(d_m)_{m\geq 0}, d_m \geq 0$, generated by the difference equation (1), is g-complete.*

The author presents a text encryption / decryption using the above-mentioned result. For context, we chose to rewrite it here as well.

Let $\mathcal{A}$ be an alphabet with N letters, labelled from 0 to $N-1$, m be a plain text and n be a number obtained by using the letter labels from the plain text m. We split the text m in blocks $m_1, m_2, \ldots, m_p$ having the same length, corresponding to the numbers $n_1, n_2, \ldots, n_p$, further encrypted as follows:

Encryption algorithm

1. We consider the natural numbers $k \geq 2, a_1, a_2, \ldots, a_k$;
2. We consider the difference equation of k degree, given by the relation (1), with the above coefficients $a_1, a_2, \ldots, a_k$.
3. We compute the elements $d_k, d_{k+1}, \ldots, d_q$, such that $d_q \leq n_i < d_{q+1}, i \in \{1,2, \ldots, p\}$

$$n = d_q c_q + r_q, 0 \leq r_q < d_q,$$

$$r_q = d_{q-1} c_{q-1} + r_{q-1}, 0 \leq r_{q-1} < d_{q-1},$$

$$r_{q-1} = d_{q-2} c_{q-2} + r_{q-2}, 0 \leq r_{q-2} < d_{q-2},$$

$$\ldots .$$

$$r_{k+1} = d_k c_k + r_k, 0 \leq r_k < d_k,$$

$$r_k = d_{k-1} c_{k-1},$$

with $c_{k-1} = r_k$, since $d_{k-1} = 1$.

We denote $s = q - k + 2$.

4. We obtain the secret encryption and decryption key $(a_1, a_2, \dots, a_k, s)$.
5. We compute the numbers $c_{iq}, c_{i(q-1)}, \dots, c_{i(k-1)}$ such that

$$n_i = d_q c_{iq} + d_{q-1} c_{i(q-1)} + d_{q-2} c_{i(q-2)} + \dots + d_k c_{ik} + d_{k-1} c_{i(k-1)},$$

as in relation (1).

6. We obtain $(c_{iq}, c_{i(q-1)}, \dots, c_{i(k-1)})$, the numeric label for the encrypted text. If all $c_{ij} \leq N - 1$, then we will use the letter L_j, which has the label c_{ij} but if $c_{ij} > N - 1$, then we will put c_{ij} in the cipher text. If there are more than one c_{ij} such that $c_{ij} > N - 1$, namely $\{c_{ij_1}, \dots, c_{ij_t}\}$, we will count the biggest number of decimals of these elements. Assuming that c_{ij_v} has the biggest number of decimals, $0 \leq v \leq t$, we will add zeros to the left side of $c_{ij_p}, p \neq v$, so that all newly obtained c_{ij_l} have the same number of decimals. In this way, we obtain the encrypted text T_i.
7. Using the above steps for all blocks, we obtain the cipher text, by joining the cipher texts $T_1 T_2 \dots T_r$.

Example 1. We consider an alphabet with 27 letters: A, B, C, ..., Z, x, where "x" represents the blank space, labelled 0, 1, 2, ..., 26. We want to encrypt the message "MATHxISxBEAUTIFUL". We split this message into the following blocks: MATH/xISx/BEAU/TIFU/Lxxx and we add three characters at the end to complete the last 4 letters block. We consider a difference equation of degree four, therefore $k = 4$ and $a_1 = 01, a_2 = 11, a_3 = 20, a_4 = 04$. From here, the encoding key is BLUE or 01 11 20 04.

By computation made by Zaharia (2024), the first block MATH is encrypted in BCDCEBDEACHA. The whole message is encrypted as "BCDCEBDEACHA DCBDDCCABCBA CDDCADACAA CCCDDDEABCLA BCBCEAACBBEA".

Decryption algorithm

1. We consider the natural numbers $k \geq 2, a_1, a_2, \dots, a_k$; The last number s is the length of encoded blocks.
2. We split the cipher text into blocks of length s and we consider their labels $(c_{iq}, c_{i(q-1)}, \dots, c_{i(k-1)})$ .We compute the elements $d_k, d_{k+1}, \dots, d_q$, therefore the corresponding plain text for each block is

$$n_i = d_q c_{iq} + d_{q-1} c_{i(q-1)} + d_{q-2} c_{i(q-2)} + \dots + d_k c_{ik} + d_{k-1} c_{i(k-1)}.$$

3. If there are numbers $\{c_{ij_1}, \dots, c_{ij_t}\}$ which are greater than $N - 1$, these numbers appear with the same number of digits or with new labels, therefore it is easy to find the initial message.

4. Implementation of the Algorithm in Scilab

We present the encryption algorithm that can automatically encrypt a message using the method as explained in the previous section.

Table 1. Encryption algorithm using Scilab

Step	
1. Input variable	Input text message and key
2. Initialization	Substitute the spaces in the initial message with '['. Add '[' to obtain equally distributed block in the message. Transform the message into an array of numbers using ASCII and subtracting 65. Call this array v. Compute the n_i-s form v, as the block cipher algorithm presented above. Find the maximum value of n_i. Compute the d_j explained in the 3rd step of the encryption algorithm.
3. Encryption	For every block of data, repeat the following steps. Find the j_0 for which $d_{j_0} < n_i < d_{j_0+1}$. Compute the quotients $c_{iq}, c_{i(q-1)}, \dots, c_{i(k-1)}$ described in the 5th step of the encryption algorithm, obtaining the encrypted-number array c. Transform the array c into text using ASCII and adding 65.
4. Print the encrypted message	Replace '[' with spaces. Print the encrypted message.

We replace the spaces in the original message with '['in the initialization step. We chose this because of the ASCII code of the '[', the next after 'Z', so we could simplify the transformation. If we don't replace the space with the square bracket, ASCII would have a return value 32, and then we had to replace this value with the 26 in our algorithm. Next, we add the square bracket when we add characters at the end of the message to obtain the block of the same length.

Example 2. We take the first example shown with the message 'MATH IS BEAUTIFUL' and key 'BLUE' and feed the data into the program.

We get the exact encrypted text in 0.0142486 seconds. Note that the execution time may vary slightly, between 0.002 and 0.003 seconds, depending on the computer and the CPU processes involved.

"Your message MATH IS BEAUTIFUL was encrypted using the key BLUE as BCDCEBDEACHA DCBDDCCABCBA CDDCADACAA CCCDDDEABCLA BCBCEAACBBEA".

Remark 1. The encrypted message is the same as the one computed by hand by Zaharia (2024).

Example 3. The same message using the key 'KING' is encrypted in 0.0130361 seconds.

"Your message MATH IS BEAUTIFUL was encrypted using the key KING as IAAGKFD BGFJJHBJ HFHKED BBJFDIFG HFGAJBF"

Example 4. Let us now take a key with a different length, for example, 'QUEEN'. The computation is more complicated to do by hand, but with the help of the computer, we have the encrypted message in 0.0123045 seconds.

"Your message MATH IS BEAUTIFUL was encrypted using the key QUEEN as CPKBLQQE BQORAPKK OHHOJP EOQPCNBK"

Example 5. Next, we choose a different key, say 'YELLOW'. The method generates higher numbers as the key's length is longer. However, we still get the encrypted message in 0.0127518 seconds.

"Your message MATH IS BEAUTIFUL was encrypted using the key YELLOW as BATWNTHCC BNUWDUHPI BPNVNTFDS"

Example 6. Taking another message 'MEET ME IN SAINT LOUIS' with the key 'JGARLAND'. The algorithm encrypts the message in 0.0172338 seconds as follows:

"Your message MEET ME IN SAINT LOUIS was encrypted using the key JGARLAND as

CCBHCEDIIBDHEBBC BEICFCGDJFBFJGBF EHJEGFJCBIAEIIIE"

Remark 2. Note that for a key with 6 letters we build 3 blocks of 6 letters. In this case, the message is converted from a string to an array

```
v =

column 1 to 24

12. 4. 4. 19. 26. 12. 4. 26. 8. 13. 26. 18. 0. 8. 13. 19. 26. 11. 14. 20. 8. 18. 26. 26.

n_1=1204041926120426

n_2=813261800081319

n_3=2611142008182626
```

This means that we need to compute the d_j with respect to equation (1) and replicate the divisions presented in the proof of Theorem 1 (Flaut, 2019).

In case of the 4 letters blocks in example 1, the n_i had 8 digits, thus we had to compute some 8 digits d_j-s to establish which d_j has the propriety that $d_j \leq n_i \leq d_{j+1}$. In this case, the computations go up to 16 digits, so it is not easy to do by hand.

Example 7. Let us try a longer text 'ENCRYPTION IS THE PROCESS BY WHICH A READABLE MESSAGE IS CONVERTED TO AN UNREADABLE FORM TO PREVENT UNAUTHORIZED PARTIES FROM READING IT' using 'YELLOW' as key. The encryption takes 0.0543054 seconds.

"Your message ENCRYPTION IS THE PROCESS BY WHICH A READABLE MESSAGE IS CONVERTED TO AN UNREADABLE FORM TO PREVENT UNAUTHORIZED PARTIES FROM READING IT was encrypted using the key YELLOW as IOQCTNLK BPNWPWQBD BNVAEDLTL BHKIVTDTB BNUWEKNVN OSFKCNPA CGDWTVOMV CHOHCOIA BNQVGOQAV RBFKUJJQ BTPXVVQXT CGFABWILX CGFMDGKLU GGCAMBRI KRGNSYEH BFMVIVIRR IOSETQSS KEFTLNP REPMUMGH IQBCVUV CFWDNPTFC BLHTHGHLQ NBCSXPJW"

Remark 3. For a large message like this, we have to transform the message into the array

```
v =

column 1 to 26

26. 4. 13. 2. 17. 24. 15. 19. 8. 14. 13. 26. 8. 18. 26. 19. 7. 4. 26. 15. 17. 14. 2. 4. 18. 18.

column 27 to 53

26. 1. 24. 26. 22. 7. 8. 2. 7. 26. 0. 26. 17. 4. 0. 3. 0. 1. 11. 4. 26. 12. 4. 18. 18. 0. 6.

column 54 to 79

4. 26. 8. 18. 26. 2. 14. 13. 21. 4. 17. 19. 4. 3. 26. 19. 14. 26. 0. 13. 26. 20. 13. 17. 4. 0.

column 80 to 105

3. 0. 1. 11. 4. 26. 5. 14. 17. 12. 26. 19. 14. 26. 15. 17. 4. 21. 4. 13. 19. 26. 20. 13. 0. 20.

column 106 to 132

19. 7. 14. 17. 8. 25. 4. 3. 26. 15. 0. 17. 19. 8. 4. 18. 26. 5. 17. 14. 12. 26. 17. 4. 0. 3. 8.
```

column 133 to 138

13. 6. 26. 8. 19. 26.

This large amount of data is hard to compute, even for a 4-letter key. We encrypted the message using a 6-letter key, which requires computation with 12-digit numbers. Thus, it takes a slightly longer time to operate.

Next, for the decryption part, we want to decrypt an encrypted message. As shown, we need the key to do this. So, the decryption algorithm has two inputs: the encrypted message and the key.

Table 2. Decryption algorithm using Scilab

Steps	
1. Input variables	Input text message and key
2. Initialization	Split the message into blocks using the space separator, ensuring all the blocks have at least the key length. Count the blocks' length. Find the longest block. Compute the d_j explained in the second step of the decryption algorithm.
3. Decryption	For every block of data, repeat the following steps. Transform each block into numbers using ASCII and subtracting 65. Compute the n_i as in the second step of the decryption algorithm. Build an array v by dividing n_i by $10^{2k}, \forall k = 1,2, ..$ and taking the quotients. Transform the array c into text using ASCII and adding 65.
4. Print the decrypted message	Replace '[' with spaces. Print the encrypted message.

We will decrypt the encrypted message to show that the algorithm works correctly.

Example 8. Decrypting 'BCDCEBDEACHA DCBDDCCABCBA CDDCADACAA CCCDDDEABCLA BCBCEAACBBEA' using the key 'BLUE' takes 0.0185837 seconds.

"Your message BCDCEBDEACHA DCBDDCCABCBA CDDCADACAA CCCDDDEABCLA BCBCEAACBBEA with the key BLUE was decrypted as MATH IS BEAUTIFUL"

Example 9. Decrypting the message obtained by the 'KING' key takes 0.0166543 seconds.

"Your message IAAGKFD BGFJJHBJ HFHKED BBJFDIFG HFGAJBF with the key KING was decrypted as MATH IS BEAUTIFUL"

Example 10. Next was the key 'QUEEN', which is decrypted in 0.0165388 seconds.

"Your message CPKBLQQE BQORAPKK OHHOJP EOQPCNBK with the key QUEEN was decrypted as MATH IS BEAUTIFUL"

Example 11. Decrypting the text with 'JGARLAND' key took 0.0237255 seconds.

"Your message CCBHCEDIIBDHEBBC BEICFCGDJFBFJGBF EHJEGFJ-CBIAEIIIE with the key JGARLAND was decrypted as MEET ME IN SAINT LOUIS"

Example 12. The long message decryption using 'YELLOW' key took longer, 0.1030707 seconds, but the result is the same.

"Your message IOQCTNLK BPNWPWQBD BNVAEDLTL BHKIVTDTB BNUWEKNVN OSFKCNPA CGDWTVOMV CHOHCOIA BNQVGOQAV R-BFKUJJQ BTPXVVQXT CGFABWILX CGFMDGKLU GGCAMBRI KRGNSYEH BFMVIVIRR IOSETQSS KEFTLNP REPMUMGH IQBCVUV CFWDNPTFC BLHTHGHLQ NBCSXPJW with the key YELLOW was decrypted as ENCRYPTION IS THE PROCESS BY WHICH A READABLE MESSAGE IS

CONVERTED TO AN UNREADABLE FORM TO PREVENT UNAUTHORIZED PARTIES FROM READING IT”

5. The Block Cipher Algorithm in Scilab with a Randomly Generated Key

We can also randomly generate a key that may be / not be embedded in the text. To this end, we randomly generate an array having 8 real components whose values are included in [0;1] (the second step of the algorithm) further multiplied by 26, before applying integer part function. Using the ASCII transformation, we get 8 random letters we will use as the key. Next, we compute in the same way shown in Table 1.

Example 13. Let’s encrypt the message ‘THE PRICE OF TOMATOES IS GOING UP’ using a randomly generated key. The algorithm will compute the encrypted message.

“Your message THE PRICE OF TOMATOES IS GOING UP was encrypted using the key DQSIANKG as DCBCCBBAEEEAFCEDDEGA EDECAADCEDAEFAEAFDA BBDFDCFDGADDDDEBEB EDEAAAEBBFCFEBFAFAEB CEBCFACCDFBFDDAEEBCB”

Of course, in this case, the recipient must have the key to know how to decrypt the message.

“Your message DCBCCBBAEEEAFCEDDEGA EDECAADCEDAEFAEAFDA BBDFDCFDGADDDDEBEB EDEAAAEBBFCFEBFAFAEB CEBCFACCDFBFDDAEEBCB with the key DQSIANKG was decrypted as THE PRICE OF TOMATOES IS GOING UP”

However, to avoid this, sending the key along with the message will be helpful, as long as we are using the same rule when we make the encryption and decryption.

Example 14. We encrypted the message adding another block of letters containing the length of the key and the key itself.

“Your message THE PRICE OF TOMATOES IS GOING UP was encrypted using the key INOBVPHY as EGDGHDBECDICFFDF BADJCAHIGCCGAB-IA ECDHFBGFHEIBBB GDGBHGAGICIDECFG DHACHHDIIABHGCCE IINOBVPHY”

For the decryption, we have added an additional routine to retrieve the length and the key from the last block of the message and got the initial message transmitted.

“Your message EGDGHDBECDICFFDF BADJCAHIGCCGABIA ECDHFBGFHEIBBB GDGBHGAGICIDECFG *DHACHHDIIABHGCCE* IINOBVPHY with the key INOBVPHY was decrypted as THE PRICE OF TOMATOES IS GOING UP”

6. Conclusion

This paper proposes and implements a symmetric block cipher encryption / decryption technique using modern approaches based on difference equations. The algorithm was developed in Scilab and further tested on multiple examples to demonstrate its functionality, efficiency, and adaptability. The encryption and decryption processes are straightforward, making the method suitable for practical applications requiring simplicity and security.

While the proposed algorithm ensures reasonable data protection and computational efficiency, further research needs to evaluate its resistance against more advanced cryptanalytic attacks. Future work will focus on conducting a comprehensive security analysis, improving the key generation mechanism, and optimizing the algorithm for deployment in real-time communication systems and resource-constrained environments, such as IoT devices. Additionally, comparisons with standard encryption algorithms like AES and RSA will be performed to assess performance, scalability, and robustness.

Acknowledgement

The authors gratefully acknowledge the internal funding that supported this research. All authors have contributed equally to this work. The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

Abdullah, A. (2017). *Advanced Encryption Standard (AES) Algorithm to Encrypt and Decrypt Data*.

Bani Yassein, M., Qawasmeh, E., Khamayseh, Y., & Mardini, W. (2017). *Comprehensive Study of Symmetric Key and Asymmetric Key Encryption Algorithms*.

Bazeer Ahamed, B., & Krishnamoorthy, M. (2024). Image captcha blended with OTP for secured authentication. In P. Vasant, V. Panchenko, E. Munapo, G.-W. Weber, J. J. Thomas, R. Intan, & M. Shamsul Arefin (Eds.), *Intelligent computing and optimization* (pp. 199–209). Springer Nature Switzerland.

Bhardwaj, C. (2012). Modification of Vigenère Cipher by Random Numbers, Punctuations & Mathematical Symbols. *IOSR Journal of Computer Engineering*, *4*, 35–38.

Brown, J. L. (1961). Note on Complete Sequences of Integers. *The American Mathematical Monthly*, *68*(6), 557–560.

Conrad, E. (2011). Domain 3: Cryptography. In *Eleventh Hour CISSP* (pp. 39–54). Elsevier.

Fahrianto, F., Masruroh, S. U., & Ando, N. Z. (2014). Encrypted SMS application on Android with combination of caesar cipher and vigenere algorithm. *2014 International Conference on Cyber and IT Service Management (CITSM)*, 31–33.

Flaut, C. (2019). Some application of difference equations in Cryptography and Coding Theory. *Journal of Difference Equations and Applications*, *25*(7), 905–920.

Kester, Q.-A. (2012). A cryptosystem based on Vigenère cipher with varying key. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, *1*, 108–113.

Mali, M., Novak, F., & Biasizzo, A. (2005). Hardware implementation of AES algorithm. *Journal of Electrical Engineering*, *56*.

Mandal, S. K., & Deepti, A. (2016). A cryptosystem based on vigenere cipher by using mulitlevel encryption scheme. *International Journal of Computer Science and Information Technologies*, *7*(4), 2096–2099.

Menezes, A. J., Van Oorschot, P. C., & Vanstone, S. A. (2018). *Handbook of applied cryptography*. CRC Press.

Milanov, E. (2009). *The RSA algorithm*. RSA Laboratories.

Qadir, A. M., & Varol, N. (2019). A review paper on cryptography. *2019 7th international symposium on digital forensics and security (ISDFS)*, 1–6.

Ragab, A. Ab, M., Madani, A., Wahdan, A. M., & Selim, G. M. I. (2023). Design, analysis, and implementation of a new lightweight block cipher for protecting IoT smart devices. *Journal of Ambient Intelligence and Humanized Computing*, *14*(5), 6077–6094.

Razzaq, A., Mahmood, Y., Ahmed, F., & Hur, A. (2012). Strong key machanism generated by LFSR based Vigenère cipher. *International Arab Conference of Information Technology*, 554–548.

Sanchez-Avila, C., & Sanchez-Reillol, R. (2001). The Rijndael block cipher (AES proposal): A comparison with DES. *Proceedings IEEE 35th Annual 2001 International Carnahan Conference on Security Technology (Cat. No.01CH37186)*, 229–234.

Shannon, C. E. (1948). A Mathematical Theory of Communication. *Bell System Technical Journal*, *27*(3), 379–423.

Shannon, C. E. (2001). A mathematical theory of communication. *ACM SIGMOBILE mobile computing and communications review*, *5*(1), 3–55.

Sher Ali, F., & Sarhan, F. (2014). Enhancing Security of Vigenere Cipher by Stream Cipher. *International Journal of Computer Applications*, *100*, 1–4.

Touil, H., el Akkad, N., & Satori, K. (2020). *Text Encryption: Hybrid cryptographic method using Vigenere and Hill Ciphers*.

Wael, A., & Hassene, S. (2016). A new SMS encryption algorithm based on hyperchaotic system. *2016 2nd International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)*, 57–62.

Wilson, P. I., & Garcia, M. (2006). A modified version of the Vigenère algorithm. *IJCSNS*, *6*(3B), 140.

Zaharia (Tudor), M. G. (2024). *Linear difference equations of degree N and some of their applications*. Ovidius University of Constanța, Doctoral School of Mathematics.

Zeckendorf, E. (1972). Représentation des nombres naturels par une somme de nombres de Fibonacci ou de nombres de Lucas/ Representation of natural numbers by a sum of Fibonacci numbers or Lucas numbers. *Bulletin de la Société Royale des Sciences de Liège*, *41*, 179–182.