Unraveling the MixColumns Operation of the AES Cipher Function

José Gladistone da Rocha
Ministério da Defesa
Secretaria de Planejamento Baseado em Capacidade
Brasília, DF, Brasil
jgladistone@gmail.com

Carlo Kleber da Silva Rodrigues
Center for Mathematics, Computing and Cognition
Federal University of ABC (UFABC)
Santo André, São Paulo, Brazil
carlo.kleber@ufabc.edu.br

ABSTRACT

The AES symmetric cryptographic algorithm is one of the most widely used nowadays. Its encryption process is quite complex and difficult to understand. Among encryption operations, one particularly is so much complicated to understand: the MixColumns. The goal of this work is to describe in detail, including an example, how AES MixColumns works. Thus, a bibliographical research was carried out to verify what the literature deals with regarding this subject in order to contribute positively to this work. As results and conclusion the objective of this article was reached out, by present all the steps involved in the operation of the AES algorithm MixColumn, explained each one of them in detail and even demonstrating a complete example of the state transformation process, which is the block of input bits of the process.

CCS Concepts

• **Security systems→Cryptography**

Keywords

Chiper; Cryptography; Encryption; MixColumns; Process.

1. INTRODUCTION

The AES symmetric cryptographic algorithm is one of the most widely used nowadays [7]. It has four basic operations for encrypting and decrypting data, using a very complex and difficult cryptographic architecture to understand [5].

Particularly, the MixColumns operation is present in the encryption function and it is the third and penultimate operation to be performed. The four operations performed in AES encryption function are executed in this sequence: SubBytes, ShiftRows, MixColumns and finally AddRoundKey. MixColumns is a lot of substitution that uses arithmetic over GF(2^8). MixColumns is one of the most complex and difficult operations to explain and understand. This was the motivation for carrying out this paper and unravels how this operation works. In addition, it was created a complete example of its cryptographic round.

Within this context. this article aims to detail the entire process of the AES MixColumns operation encryption algorithm and provides a complete example of this operation, in order to make it easier to understand and for its future implementation in software. To this end, this work is then organized as follows: Section 1 deals with the introduction; literature background is addressed in Section 2; Section 3 presents the methodology applied to the paper; Section 4 deals with the detailed explanation of the MixColumns operation functioning and finally, Section 5 refers to the conclusion and indication of future work.

2. BACKGROUND

Several research literature works are discussed in what follows. The main goal is to provide the reader with a general state-of-the-art view of the research theme of this present research.

To make the cloud data stored more secure according to the characteristics of cloud computing, in [7], the authors studied the modified data encryption algorithm in cloud technology. First, the traditional advanced encryption standard AES is analyzed. Then, a modified advanced encryption standard for data security in cloud computing is proposed by introducing random disturbance information to improve data security. Furthermore, the column mixing operation and key choreography in AES are improved. Formal security analysis and performance comparisons conducted by the authors indicate that the proposed solutions simultaneously ensure attribute privacy and improve decryption efficiency for outsourced data storage in mobile cloud computing.

Security measures such as data encryption often result in reduced performance speed. In [2], the authors performed a work to improve the 128-bit version of AES by replacing the MixColumns function with a permutation-based approach and decreasing the overall number of rounds. The evaluation results indicated a substantial improvement in encryption and decryption speed, with a 76.76% improvement in encryption time and a 55.46% improvement in decryption time. Furthermore, the authors report that it is important to mention that the modifications implemented in the standard AES did not compromise its security respecting to the avalanche effect criterion, which for the modified AES is 52.92%, exceeding the minimum requirement of 50%. Finally, the modified AES demonstrated a 31.12% increase in throughput for encryption and a 25.50% increase for decryption when compared to the original AES, using the sample dataset.

In [9], the authors carried out a study to show how the Mixcolumn operation works using examples, but beyond the example, it does not really explain how this AES operation works. In [5], the authors proposed a technique for the modification of MixColumns using Very Large Scale Integration (VLSI) system design. The modified MixColumns transformation improves the performance of the AES algorithm. In the proposed technique, the door counts in the MixColumns process have been reduced. When compared to existing AES encryption and Decryption using MixColumns-based Xtime multiplication, the proposed optimized MixColumns-based AES decryption provides better performance.

Block ciphers, particularly Substitution-Permutation Network (SPN) such as AES, are widely used in contemporary cryptography. However, they face strong cryptanalysis including differential, linear and algebraic cryptanalysis. Therefore, increasing the security of block ciphers, particularly AES, is an urgent area of research. In addition to security, the execution cost of block ciphers is crucial. In [8], the authors conducted a research work that elucidates how Maximum Distance Separable (MDS) matrices increase the number of diffusion layer branches in block ciphers, increasing their security. The authors propose a method to increase the security of AES by altering its Mixcolumns transformation using efficient MDS matrices of various sizes. Furthermore, they created a technique to evaluate the fixed point coefficients of D(A) and fixed points in the modified AES diffusion layers. They demonstrated the number of branches of modified AES diffusion layers with MDS matrices of sizes 8 and 16, analyzing their security, statistical patterns, and execution speed. Their discovery indicates a significant improvement in AES security through their proposed approach.

In [3], the authors modified AES MixColumns based on cellular automata functions. AES has no compression, but it has good accessibility compared to other algorithms. The modified AES hardware implementation provides efficient memory space and area consumption. Comparative study of architecture Security analysis as Fast Walsh Transform method is followed to verify the security in modified AES algorithm. Traditional Mixcolumns and Cellular Automata-based Mixcolumns architecture is done through hardware simulation in Xilinx tool, to show the Field-Programmable Gate Array (FPGA) implementation of AES results as a lightweight cipher, in terms of memory requirement.

In [1], the authors proposed an implementation of the AES MixColumns operation where a compact architecture for the AES MixColumns operation and its inverse with hardware implementation is presented. The authors show that the design has a lower gate count than other designs that implement both the forward and inverse MixColumns operation. Comparisons indicate that the proposed MixColumns design has less complexity than previous relevant work in gate size and number of clock cycles. This compact design can help implement AES for smart cards, RFID tags, and wireless sensors.

Finally, in [10], the authors presentes in their work an efficient method to calculate the circulating matrices in the AES MixColumns transformation to accelerate encryption. Using the multiplication of 8×8 involutional matrices, 64 multiplications and 56 additions are required in the Mix-Columns transformation. The authors proposed a method with diversity of 8×8 circulating matrices and in which only 19 multiplications and 57 additions are required for both encryption and decryption operations. Therefore, the 8×8 circulant matrix operation with AES key sizes of 128-bit, 192-bit, and 256-bit is over 33.5%, 33.7%, and 33.9% faster than using the 4×4 involutory matrix operation (16 multiplications, 12 additions), respectively. The encryption/decryption speed of 8×8 circulating matrix is over 79% faster than that of 8×8 involutional matrix operation. Finally, the proposed method for evaluating matrix multiplication can be made regular, simple and suitable for software implementations in embedded systems.

3. METHODOLOGY

To construct this work, a bibliographical survey was carried out in renowned books, articles and websites on the subject to obtain the necessary information to formulate the knowledge base of this article core.

This work is heavily dependent on modern mathematical concepts that are essential to understanding what this article proposes, which is a complete understanding of how the AES MixColumns operation works.

The following section is entirely dedicated to these mathematical concepts that will serve as a basis for understanding the subsequent sections.

4. PRELIMINARY CONCEPTS

The MixColumns operation makes extensive use of the concept of finite fields and in this way, what this mathematical knowledge is and how to use it, in practice, in cryptographic algorithms such as AES will be presented below. This transformation provides good diffusion properties in the AES algorithm [3].

Several cryptographic algorithms rely widely on properties of finite fields, notably the Advanced Encryption Standard (AES) and elliptic curve cryptography [6]. Other examples include the CMAC message authentication code, which is an authentication technique that uses a cipher algorithm such as AES to generate an authentication code for a message, and the GCM authenticated encryption scheme, which is an authenticated encryption mode of operation that combines symmetric encryption with message authentication, providing an efficient and secure solution for data encryption and authentication.

Finite fields is a subset of fields, consisting of those fields with a finite number of elements. These are the classes of fields that are found in cryptographic algorithms.

The most important class of finite fields, for cryptography, comprises within 2^n elements described as fields of the form GF(2^n). They are used in a wide variety of cryptographic algorithms. However, before discussing these fields, it is necessary to analyze the topic of polynomial arithmetic.

The finite field of pn order is usually written GF(p^n); GF stands for Galois field, named after the mathematician who first studied finite fields. Two special cases are of this purpose interest. For n = 1, is called finite field GF(p); this finite field has a different structure from the finite fields with n > 1. For finite fields of GF(p^n) form, GF(2^n) fields are of particular cryptographic interest [6].

For a given prime, p, we define the finite field of order p, GF(p), as the set Z_p of integers {0, 1, ... , p - 1} with the arithmetic operations modulo p. Note, therefore, that ordinary modular arithmetic is being used to define the operations on these fields.

Some transformations of the AES algorithms specified in Section 5, each byte in the state array is interpreted as one of the 256 elements of a finite field denoted by GF(2^8) [4].

In addition and multiplication defined in GF(2^8), each byte {b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0} is interpreted as a polynomial, denoted by b(x), as follows:

$$b(x)=b_7x^7+b_6x^6+b_5x^5+b_4x^4+b_3x^3+b_2x^2+b_1x+b_0 \qquad (1)$$

For example: {10110011} is equivalent to the polynomial: x7 + x5 + x4 + x + 1.

4.1 Addition in GF(2^8)

Addition in this field is the performance of exclusive or (XOR) operations, which is represented by the symbol $\oplus$.

The elementary operations are:

$1 \oplus 1 = 0$; $1 \oplus 0 = 1$; and $0 \oplus 0 = 0$.

As examples of this operation, three representations are presented that are identical and indicate the same results.

$(X^4 + X^2 + 1) \;+\; (X^7 + X^3) \;=\; X^7 + X^4 + X^3 + X^2 + 1$ (polinomial)

$\{00010101\} \oplus \{10001000\} = \{10011101\}$ (binary) (2)

$\{15\} \oplus \{F8\} = \{9F\}$ (hexadecimal)

4.2 Multiplication of a word by a fixed Matrix

Algorithms for AES block ciphers can be expressed in terms of matrix multiplication. In particular, a distinct fixed matrix is specified for each transformation. For both matrices, each of the 16 entries of the matrix is a byte of a single specified word, denoted here by $[a_0,a_1,a_2,a_3]$ (STALLING, 2017).

In a given input word $[b_0,b_1,b_2,b_3]$ for the transformation, the output word $[d_0,d_1,d_2,d_3]$ is determined by finite field arithmetic as follows:

$$
\begin{aligned}
d0 &=(a0 \bullet b0)\oplus(a3 \bullet b1)\oplus(a2 \bullet b2)\oplus(a1 \bullet b3)\\
d1 &=(a1 \bullet b0)\oplus(a0 \bullet b1)\oplus(a3 \bullet b2)\oplus(a2 \bullet b3) \qquad (3)\\
d2 &=(a2 \bullet b0)\oplus(a1 \bullet b1)\oplus(a0 \bullet b2)\oplus(a3 \bullet b3)\\
d3 &=(a3 \bullet b0)\oplus(a2 \bullet b1)\oplus(a1 \bullet b2)\oplus(a0 \bullet b3)
\end{aligned}
$$

These two mathematical operations are used in the AES MixColumns operation.

5. MIXCOLUMNS OPERATION

MixColumn is equivalent to matrix multiplication of each column of the state. A fixed matrix is multiplied to each column vector. In this operation, bytes are taken as polynomials instead of numbers. The state transformation that takes all the columns in the state and shuffles their data (independently of each other) to produce new columns.

Each column of the state matrix is represented as a vector with coefficients in GF (2^8). Thus, this vector is multiplied by a constant 4x4 matrix over GF(2^8) as indicated in Figure 2.

Figure 1 presents the AES encryption function, so that it can be identified in the algorithm where the MixColumns operation is found in the general context.

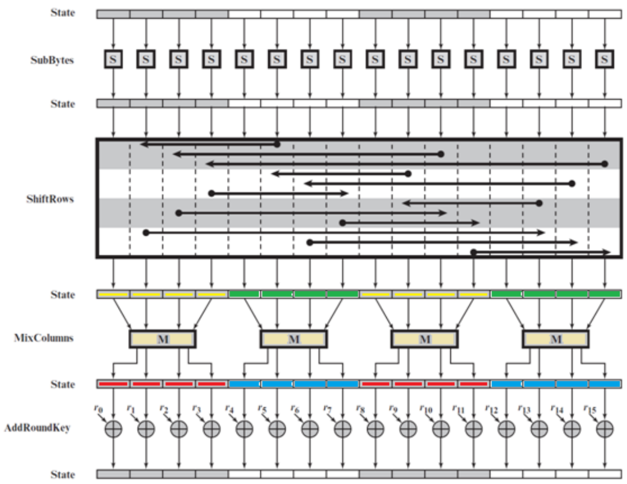

Fig. 1: AES encryption function. Source: [6].

Note in Figure 1 the initial vector of 128 bits is called state, or 4 groups of MixColumns (yellow and green blocks in Figure 1) of 4 words with 8 bits (1 byte) each, which is input to the AES encryption function and used throughout the process. The transformation of this vector is also called state. Thus, the MixColumns operation input is a 128-bit vector originating from the transformation of the state that came from the previous operation, ShiftRows.

MixColumns operates with each column individually. Each byte in a column is mapped into a new value that is a function of all four bytes in that column. The transformation can be defined by the following matrix multiplication in state (Figure 2):

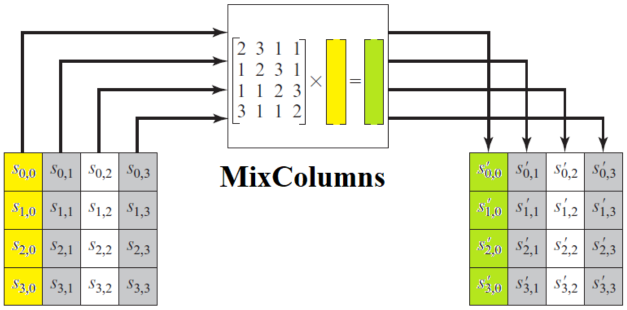
2 3 1 1
1 2 3 1
1 1 2 3
3 1 1 2
×
=
MixColumns
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,1}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$

Fig. 2: MixColumns Illustration. Source: adapted from [6].

As Figure 2 indicates, the first column of state multiplies the first row of the constant matrix resulting in the first column of the new state; the second column of state multiplies the second row of the constant matrix resulting in the second column of the new state, and so on.

Thus, each element in the product matrix is the sum of the products of the elements in a row and a column. In this case, individual additions and multiplications are performed in GF(2^8).

$$
\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix}
\begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix}
=
\begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix} \qquad (4)
$$

One way of visualizing this matrix multiplication is presented in the four equations (5), where **s'** represents the output of the 4x4 matrix, which is the new state of the next stage input. Figure 2 illustrates this MixColumns matrix multiplication. Where in the equations (5) the index j varies according to $0 \leq j < 4$. The MixColumns transformation on a single state column can be expressed as indicated in the equation (5).

$$
\begin{aligned}
s'_{0,j} &= (2 \cdot s_{0,j}) \oplus (3 \cdot s_{1,j}) \oplus s_{2,j} \oplus s_{3,j}\\
s'_{1,j} &= s_{0,j} \oplus (2 \cdot s_{1,j}) \oplus (3 \cdot s_{2,j}) \oplus s_{3,j} \qquad (5)\\
s'_{2,j} &= s_{0,j} \oplus s_{1,j} \oplus (2 \cdot s_{2,j}) \oplus (3 \cdot s_{3,j})\\
s'_{3,j} &= (3 \cdot s_{0,j}) \oplus s_{1,j} \oplus s_{2,j} \oplus (2 \cdot s_{3,j})
\end{aligned}
$$

In order to make it easy to understand, an example of this multiplication of matrices will be presented, which results in the execution of the equations (5).

Let the following matrix (4x4) be the input state of MixColumns:

5F	13	46	17
22	2C	19	21
A0	1B	30	09
57	11	FE	20

Adjusting the matrix multiplication to suit the MixColumns process, where the first matrix is composed of constants and the second matrix is the state, will look like this:

2	3	1	1
1	2	3	1
1	1	2	3
3	1	1	2

Constants

5F	13	46	17
22	2C	19	21
A0	1B	30	09
57	11	FE	20

State

=

$S'_{0,0}$	$S'_{0,1}$	$S'_{0,2}$	$S'_{0,3}$
$S'_{1,0}$	$S'_{1,1}$	$S'_{1,2}$	$S'_{1,3}$
$S'_{2,0}$	$S'_{2,1}$	$S'_{2,2}$	$S'_{2,3}$
$S'_{3,0}$	$S'_{3,1}$	$S'_{3,2}$	$S'_{3,3}$

New state

Applying formulas (5) to the first column of the state we have:

$S'_{0,0} = (\{02\} . \{5F\}) \oplus (\{03\} . \{22\}) \oplus \{A0\} \oplus \{57\}$

$S'_{1,0} = \{5F\} \oplus (\{02\} . \{22\}) \oplus (\{03\} . \{A0\}) \oplus \{57\}$

$S'_{2,0} = \{5F\} \oplus \{22\} \oplus (\{02\} . \{A0\}) \oplus (\{03\} . \{57\})$

$S'_{3,0} = (\{03\} . \{5F\}) \oplus \{22\} \oplus \{A0\} \oplus (\{02\} . \{57\})$

In $S'_{0,0}$ transforming the numbers from hexadecimal to binary we have:

$S'_{0,0} = (\{02\} . \{5F\}) \oplus (\{03\} . \{22\}) \oplus \{A0\} \oplus \{57\}$

$S'_{0,0} = (\{10\} . \{01011111\}) \oplus (\{11\} . \{00100010\}) \oplus \{10100000\} \oplus \{01010111\}$

$S'_{0,0} = \{10111110\} \oplus \{01100110\} \oplus \{10100000\} \oplus \{01010111\}$ where,

$S'_{0,0} = \{11010111\}$, passing to hexadecimal we have: D7

$S'_{0,0} = D7$

Calculating for $S'_{1,0}$ we have:

$S'_{1,0} = \{5F\} \oplus (\{02\} . \{22\}) \oplus (\{03\} . \{A0\}) \oplus \{57\}$

In $S'_{1,0}$ transforming the numbers from hexadecimal to binary we have:

$S'_{1,0} = \{01011111\} \oplus (\{10\} . \{00100010\}) \oplus (\{11\} . \{10100000\}) \oplus \{01010111\}$

$S'_{1,0} = \{01011111\} \oplus \{01000100\} \oplus \{11100000\} \oplus \{01010111\}$

$S'_{1,0} = \{01001100\}$, passing to hexadecimal we have: 4C

$S'_{1,0} = 4C$

Calculating for $S'_{2,0}$ we have:

$S'_{2,0} = \{5F\} \oplus \{22\} \oplus (\{02\} . \{A0\}) \oplus (\{03\} . \{57\})$

In $S'_{2.0}$ transforming the numbers from hexadecimal to binary we have:$S'_{2,0} = \{01011111\} \oplus \{00100010\} \oplus (\{10\} . \{10100000\}) \oplus (\{11\} . \{01010111\})$

$S'_{2,0} = \{01011111\} \oplus \{00100010\} \oplus \{01000000\} \oplus \{11110101\}$

$S'_{2,0} = \{01001010\}$, passing to hexadecimal we have: 4A

$S'_{2,0} = 4A$

Calculating for $S'_{3.0}$ we have:

$S'_{3,0} = (\{03\} . \{5F\}) \oplus \{22\} \oplus \{A0\} \oplus (\{02\} . \{57\})$

In $S'_{3.0}$ transforming the numbers from hexadecimal to binary we have:

$S'_{3,0} = (\{11\} . \{01011111\}) \oplus \{00100010\} \oplus \{10100000\} \oplus (\{10\} . \{01010111\})$

$S'_{3,0} = \{00011101\} \oplus \{00100010\} \oplus \{10100000\} \oplus \{10101110\}$

$S'_{3,0} = \{00110001\}$, passing to hexadecimal we have: 31

$S'_{3,0} = 31$

Now the output matrix will be updated, looking like this:

D7	$S'_{0,1}$	$S'_{0,2}$	$S'_{0,3}$
5C	$S'_{1,1}$	$S'_{1,2}$	$S'_{1,3}$
4A	$S'_{2,1}$	$S'_{2,2}$	$S'_{2,3}$
31	$S'_{3,1}$	$S'_{3,2}$	$S'_{3,3}$

Calculating for the other columns as indicated for column 1 and applying the formulas (5) we have the complete matrix below as the new state for input in the next operation of the AES encryption algorithm, which is AddRoundKey.

D7	83	09	64
C5	8F	1A	6E
4A	04	45	44
31	32	01	29

Another way to characterize the MixColumns transformation is in terms of polynomial arithmetic. In the standard, MixColumns is defined by considering each column of State as a four-term polynomial with coefficients in GF(2^8). Each column is multiplied modulo ($x^4 + 1$) by the fixed polynomial a(x), given by:

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\} \qquad (6)$$

This expression will not be applied in this article, but it was presented here because it actually represents the finite field for GF(2^8), that is, in polynomial form as indicated in Formula (6).
As noted, the initial state of MixColumn is heavily modified with the finite field transformation GF(2^8) giving the AES algorithm the nonlinearity to make it even more secure against linear and differential cryptanalysis attacks.

6. CONCLUSION

The present study aimed to elucidate, in a detailed and systematic manner, the functioning of the MixColumns operation in the Advanced Encryption Standard (AES) algorithm. To achieve this purpose, a bibliographical and theoretical investigation was conducted, grounded in the mathematical framework of finite fields (GF(2^8)) and matrix algebra. This methodological approach enabled a rigorous reconstruction of the MixColumns procedure, supplemented by a fully worked numerical example that illustrates each computational step involved in the transformation.

The results obtained demonstrate the complete execution of the MixColumns operation, encompassing the conversion between hexadecimal, binary, and polynomial representations, as well as the successive applications of finite field arithmetic that yield the final state matrix. Through this systematic exposition, the work provides a transparent and comprehensive understanding of how diffusion and nonlinearity are introduced within AES, both of which are fundamental to the cipher's resistance against differential and linear cryptanalysis.

The principal contribution of this study resides in its didactic and analytical value. By articulating the mathematical underpinnings of the MixColumns operation in a clear and structured manner, the research enhances the accessibility of complex cryptographic concepts and supports the correct implementation of AES in software and hardware environments. Furthermore, the discussion reinforces the importance of finite field operations as a cornerstone of contemporary symmetric cryptography, linking theoretical abstraction to practical security mechanisms.

Building upon the findings presented herein, several avenues for further investigation are suggested: (1) a comparative and detailed study of the remaining AES transformations—SubBytes, ShiftRows, and AddRoundKey; (2) the exploration of hardware-optimized architectures for MixColumns to improve performance and energy efficiency; (3) the examination of modified AES variants incorporating alternative diffusion matrices or reduced computational complexity; and (4) the development of educational frameworks, visualization tools, or simulation environments aimed at facilitating the teaching and comprehension of finite field–based cryptographic operations.

7. REFERENCES

[1] AHMED, Eslam Gamal; SHAABAN, Eman; HASHEM, Mohamed. Ligthweight Mix Columns Implementation for AES. International Conference on APPLIED INFORMATICS AND COMMUNICATIONS. 2009.

[2] BALADHAY, Jerico S.; REYES, Edjie M. De Los. AES-128 reduced-round permutation by replacing the MixColumns function. Indonesian Journal of Electrical Engineering and Computer Science Vol. 33, No. 3, March 2024, pp. 1641~1652. DOI: https://www.doi.org/10.11591/ijeecs. v33.i3.pp1641-1652. 2024.

[3] KUMAR, K.J. Jegadish; BALASUBRAMANIAN, R. Lightweight Mixcolumn Architecture for Advanced Encryption Standard. International Journal of Computer Applications (0975 – 8887), Volume 136 – No.11, February, 2016.

[4] NIST – National Institute of Standards and Technology. Advanced Encryption Standard (AES). Federal Information Processing Standards Publication. 2023.

[5] RAMESH, Singh Poja e SINGH, Santhosh Kumar. Improvement of Data Security Using Mixcolumn. Indonesian Journal of Electrical Engineering and Computer Science. Vol. 9, No. 2, February 2018, pp. 361~364. ISSN: 2502-4752, DOI: https://www.doi.org/10.11591/ijeecs. v9.i2.pp361-364. 2018.

[6] STALLINGS W. Criptografia e Segurança de Redes: Princípios e Práticas. 7ª Ed, São Paulo, Editora Pearson Education do Brasil, ISBN: 978-85-430-1450-0, 2017.

[7] TENG, L., LI, H., YIN, S. e SUN, Y. A Modified Advanced Encryption Standard for Data Security. International Journal of Network Security, Vol.22, No.1, p.112-117, DOI: HTTPS://www.doi.org/10.6633/IJNS.202001. 22(1).11. 2020.

[8] THANG, Chien; TRIEU, Tan; TRI, Thanh; NOI, Ha; NAM, Viet. ENHANCING THE SECURITY OF AES BLOCK CIPHER BASED ON MODIFIED MIXCOLUMN TRANSFORMATION. Journal of Computer Science and Cybernetics, V.40, N.2 (2024), 186–202 DOI no. https://www.doi.org/10.15625/1813-9663/18058. 2024.

[9] XINTONG, Kit Choy. Understanding AES Mix-Columns Transformation Calculation. University of Wollongong. 2014.

[10] WANG, Jeng-Jung; CHEN, Yan-Haw; CHEN, Yan-Wen; LEE, Chong-Dao. DIVERSITY AES IN MIXCOLUMNS STEP WITH 8×8 CIRCULANT MATRIX. International Journal of Engineering Technologies and Management Research, V. 8 N.9, p. 19–35. DOI: https://www.doi.org/10.29121/ijetmr.v8.i9.2021.1037. 2021.