BAREKENG: Journal of Mathematics and Its Applications
March 2024 Volume 18 Issue 1 Page 0517–0528
P-ISSN: 1978-7227 E-ISSN: 2615-3017
https://doi.org/10.30598/barekengvol18iss1pp0517-0528

CONSTRUCTION OF SUBSTITUTION BOX (S-BOX) BASED ON IRREDUCIBLE POLYNOMIALS ON $GF(2^8)$

Faldy Tita [1*], Adi Setiawan[2], Bambang Susanto[3]

[1]*Department of Mathematics, Faculty of Sciences and Mathematics, Universitas Kristen Satya Wacana*
[1,2,3]*Master of Data Sciences Department, Faculty of Sciences and Mathematics, Universitas Kristen Satya Wacana*
Diponegoro Street No 52-60. Salatiga, 50711, Indonesia

*Corresponding author's e-mail: * **faldy.tita@uksw.edu***

Article History:

Received: 28th October 2023
Revised: 6th January 2024
Accepted: 28th January 2024

Keywords:

Cryptography;
Galois Field;
Hamming;
Substitution Box.

ABSTRACT

In the field of modern encryption algorithms, the creation of S-Box is an essential element that plays an important role in maintaining data security in various industries. This article provides a comprehensive review of various S-Box designs, with particular emphasis on essential parameters such as "Average d", "Average H_C" and "Non-linearity value". The main goal is to determine the most optimal S-Box structure to minimize correlation, thereby improving the security and unpredictability of the cryptographic system. Research results indicate that the S-Box characterized by the 1BD hexadecimal code is superior to its counterparts. It has an average d value of 4.1953 and an average H_C value of 0.4756. In contrast, the S-Box represented by hexadecimal code 169 displays a relatively lower level of security, with an average d value of 3.8750 and an average H_C value of 0.5156. These results enable security experts and cryptographers to make the correct choice when selecting the S-Box with the minimum correlation value, thereby strengthening cryptographic systems against emerging cyber threats.

How to cite this article:

F. Tita, A. Setiawan, B. Susanto., "CONSTRUCTION OF SUBSTITUTION BOX (S-BOX) BASED ON IRREDUCIBLE POLYNOMIALS ON $GF(2^8)$," *BAREKENG: J. Math. & App.*, vol. 18, iss. 1, pp. 0517-0528, March, 2024.

Journal homepage: *https://ojs3.unpatti.ac.id/index.php/barekeng/*
Journal e-mail: *barekeng.math@yahoo.com; barekeng.journal@mail.unpatti.ac.id*

Research Article · Open Access

1. INTRODUCTION

In an increasingly advanced digital age, sensitive data protection has become a significant concern in various sectors, including banking, health, communications, and more. Strong and reliable data security protects personal information, industrial secrecy, and national interests. Encryption algorithms play a crucial role in modern data security systems. One of the most commonly used encryption algorithms is the Advanced Encryption Standard Algorithm (AES). AES uses a symmetrical cryptographic approach to protect sensitive data with high encryption strength [1]. The Substitution Box (S-Box) is essential for encrypting and decrypting data in the AES structure. S-Box converts input bytes into output bytes through non-linear substitution. A good design of the S-Box is critical to generating an optimal level of security in the data security system. However, in the face of new challenges in data security and increasingly sophisticated crypto-analysis attacks, it is crucial to continue to develop and enhance the construction of S-Boxes used in security systems. A more substantial and efficient S-box design can increase the security level of the data system and reduce the risk of attacks [2]. Previous research [3]–[12] has proposed various methods and approaches to S-Box construction for data security systems.

Moreover, a substantial selection of S-boxes can provide additional strength against crypto-analysis attacks such as differential and linear attacks. Various S-box constructions have been proposed by researchers using mathematical, heuristic, and algebraic approaches. An effective S-box design is a critical component of a cryptographic system and must stick to strict security standards such as proliferation, nonlinearity, and resilience to potential attacks. In this context, an irreducible polynomial (IP) in $GF(2^8)$ is a nonconstant polynomial that resists decomposition into a product of two nonconstant polynomials with binary coefficients. These polynomials play an important role in constructing finite field operations and designing algorithms for cryptographic primitives such as error-correcting codes and the Advanced Encryption Standard (AES). Their importance extends to the creation of finite field extension fields, thereby increasing the efficiency and safety of algorithms operating on these fields. Despite these advances, continued research is needed to explore and improve the design, evaluation, and improvement of S-Boxes to meet the ever-increasing demands for security and efficiency in cryptographic applications.

This article introduces a new perspective to the S-box design of cryptographic systems by emphasizing the distinctive role of irreducible polynomials in $GF(2^8)$. While prior studies have explored essential criteria like diffusion, nonlinearity, and resistance to attacks in S-box design, this work contributes originality by underscoring the importance of these polynomials in constructing finite field arithmetic and designing algorithms for cryptographic primitives such as error-correcting codes and the Advanced Encryption Standard (AES). Additionally, we highlight the pervasive effects of irreducible polynomials in creating finite field extension fields, enhancing the efficiency and security of algorithms within these domains. Despite the progress in S-box design discussed in previous literature [4], [9], [10], and [13], our article identifies areas for further investigation and improvement, taking into account the evolving landscape of security and efficiency requirements in cryptographic systems. Consequently, this work not only builds on existing knowledge but also identifies potential avenues for future research, bringing a forward-looking perspective to the current state of S-box design in cryptographic applications.

The goal of the research is to determine the optimal S-Box design for data security systems, considering factors such as security, computational efficiency, and resistance to crypto-analysis attacks based on some criteria like Hamming distance, Hamming correlation and non-linearity function. By digging deeper into S-Box's design, data security systems are expected to become more robust, reliable, and effective in protecting sensitive information from the growing threat of external attacks.

2. RESEARCH METHODS

In this study, the research methods used include identifying the purpose of the research, understanding S-box construction on cryptographic algorithms, searching for literature, literature analysis, development of new construction methods, simulation, and analysis, as well as conclusions and discussions. In the phase of literary search, a comprehensive literature search is carried out through various sources of information such as academic databases, scientific journals, and related conferences. Relevant keywords are used to obtain literature related to S-box construction. Existing methods are evaluated based on observed strengths, weaknesses, and safety criteria. Furthermore, a new S-box construction will be developed using an irreducible

polynomial in Field. This new approach is based on cryptographic theory, mathematics, optimization, or other techniques. The proposed method is then tested through simulation or implementation to evaluate security strength, efficiency, and resistance to attack.

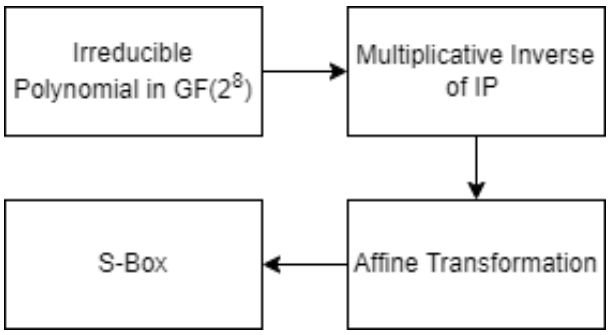

Figure 1. Step by step to Construct Substitution box (S-Box)

The Substitution Box (S-Box) is essential in cryptographic algorithms for data encryption and decryption. The construction process involves the following steps as shown in **Figure 1**. That is:

1. Finds the inverse multiplication of each element on $GF(2^m)$ of an irreducible polynomial in $GF\ (2^m)$.
2. Taps the affine matrix and affine key vector of the operation to find the S-Box. Several conditions define affine matrices that are non-linear, unique, and non-singular.
3. Conduct affine transformation by multiplying the vectors of each inverse element on the $GF(2^m)$ with the affine matrix and then adding them to a given affine key vector. In the calculation process, the XOR operation is used to determine the result.
4. After performing affine transformations on each element, the entire S-box is obtained.

2.1 Galois Field (GF)

Galois Field is also known as a finite field, containing a limited number of elements. Évariste Galois, a French mathematician, introduced the field and significantly contributed to polynomial and field theories. Galois Field has essential roles in various mathematics and computer science fields, including cryptography, code theory, communication system design, and polynomial theories.

Definition 1. A Galois field in order q, denoted as $GF(q)$, is a field containing q elements. In this case, q is a prime rank $(q = p^m)$ with p, a prime number and m, a positive integer. The elements of Galois Field $GF(p^m)$ is defined as

$$\begin{aligned} GF(p^m) = {} & (0, 1, 2, \dots, p-1) \cup \\ & (p, p+1, p+2, \dots, p+p-1) \cup \\ & (p^2, p^2+1, p^2+2, \dots, p^2+p-1) \cup \dots \cup \\ & (p^{m-1}, p^{m-1}+1, p^{m-1}+2, \dots, p^{m-1}+p-1) \end{aligned}$$

The integers of modulo q represent elements from the Galois Field, so the mathematical operations used in this field also use modulo q [14]. Various studies related to the Galois Field or finite and current fields can be seen. Aidoo & Gyamfi [15], suggested how to construct an irreducible polynomial in $GF(2^m)$ using Normal Bases. Similarly, Nithya & Ramadoss [16], review how an expansion field is formed from smaller fields, as is discussed about finite fields in the Galois Theory. In addition, Dey & Ghosh [17], suggests how to find monic and irreducible polynomial over Gallois Field $GF(p^q)$.

2.2 Substitution Box (S-Box)

The S-box (Substitution Box) is crucial in various symmetrical cryptographic algorithms. The primary function is to replace or change input values with different output values. S-Box aims to provide diffusion or

confusion in cryptographic algorithms by creating a complex relationship between input and output. Usually, the S-Box operates on data blocks of a certain length, such as 8bit or 32bit, and can be applied repeatedly in cryptographic algorithms. Each input value in the data block will be replaced with a corresponding output value following the substitution function specified by the s-Box.

Using the S-Box primarily creates complicated non-linear relationships between input and output, making it difficult for crypto-analysis attacks to find patterns or weaknesses in cryptographic algorithms. In some algorithms, such as AES, the S-Box is used in conjunction with other linear functions, such as substitution and bit shift, to provide a higher level of security. A well-designed S-Box must have the following characteristics [7]:

1. Non-linearity: The S-box must produce outputs that are difficult to predict or show a linear relationship pattern between input and output.
2. Diffusion: A one-bit change in the input of the S-Box must cause multiple-bit changes in the output so that the effect of the shift spreads evenly in the cryptographic algorithm.
3. Resistance to attack: The S-Box should be designed to withstand various crypto-analysis attacks that attempt to disclose classified information or reduce the complexity of the crypto algorithm.

3. RESULTS AND DISCUSSION

3.1 S-box construction on $GF(2^8)$

The first step in constructing the S-box is to find the inverse multiplication of each element in the $GF\ (2^8)$. Mathematically, an inverse calculation of an element of an irreducible polynomial can be done using the Extended Euclid Algorithm [18].

The extended Euclidean algorithm is used to find a Bézout coefficient that meets the Bézout identity, namely:

$$ax\ +\ by\ = \gcd(a, b) \tag{1}$$

where a and b are the given integer numbers. x and y are the Bézout coefficients to be solved, and $\gcd(a, b)$ is the greatest common divisor (GCD) of a and b.

Here are the stages in the Extended Euclidean Algorithm:

1. Initialization: Start with two nonnegative integers a and b, where $a \geq b$. If $b = 0$. then GCD is a, and the corresponding Bézout coefficient is $(1, 0)$.
2. Division: Give a by b and note the remaining result. Thus, we can write $a = bq + r$, where q is the result and r is the remainder.
3. Update: Replace a with b and b with r obtained from the previous step. In other words, do $a \leftarrow b$ and $b \leftarrow r$.
4. Repeat: Repeat steps 2 and 3 until the rest of r is 0.
5. Bézout identity: After the rest is 0. The last step produces the linear equation $ax\ +\ by\ = \gcd(a, b)$, where x and y are the desired Bézout coefficients, and $\gcd(a, b)$ is the GCD of a and b.

Example 1. Suppose you want to find the inverse of 83 on module 283. Then just count

$$83x + 283y = \gcd(83, 283)$$

With the calculation of Euclid's algorithm

$$\begin{aligned} 283 &= 83\ (3) + 34 \\ 83 &= 34\ (2) + 15 \\ 34 &= 15\ (2) + 4 \\ 15 &= 4\ (3) + 3 \\ 4 &= 3\ (1) + 1 \\ 3 &= 1\ (3) + 0 \end{aligned}$$

obtained the value $\gcd(83,283) = 1$.
Then calculate the value that meets the Bézout identity

$$83x + 283y = 1$$

Using the reverse calculation

$$\begin{aligned} 1 &= 4 - 3 \\ &= 4 - \big(15 - 4(3)\big) \\ &= 4(4) - 15 \\ &= \big(34 - 15(2)\big)(4) - 15 \\ &= 34(4) - 15(9) \\ &= 34(4) - \big(83 - 34(2)\big)(9) \\ &= 34(22) - 83(9) \\ &= \big(283 - 83(3)\big)(22) - 83(9) \\ &= 283(22) - 83(75) \end{aligned}$$

So, we get $1 = 83(-75) + 283(22)$. Based on Bézout's identity, the inverse of 83 on module 283 is -75 or 208 in class 283.

The Extended Euclid algorithm used to calculate the greatest divisor polynomial (GCD) of two polynomials can be done by searching for two Polynomials, $U(x)$ and $W(x)$, as Bézout identities, that meet the following equation [19]:

$$\gcd(A(x), B(x)) = U(x) \times A(x) + W(x) \times B(x) \quad (2)$$

The Extended Euclid algorithm works as follows: the symbol "÷" indicates the operation that counts the result for a polynomial. The algorithm for calculating the inverse polynomial is as follows:

1. Initializing $R_{-1}(x) \coloneqq B(x)$; $U_{-1}(x) \coloneqq 0$; $W_{-1}(x) \coloneqq 1$; $R_0(x) \coloneqq A(x)$; $U_0(x) \coloneqq 1$; $W_0(x) \coloneqq 0$; and $j \coloneqq 0$;
2. Calculate the following values:
   a. $Q_j(x) \coloneqq R_{j-2}(x) \div R_{j-1}(x)$;
   b. $R_j(x) \coloneqq R_{j-2}(x) - Q_j(x) \times R_{j-1}(x)$;
   c. $U_j(x) \coloneqq U_{j-2}(x) - Q_j(x) \times U_{j-1}(x)$;
   d. $W_j(x) \coloneqq W_{j-2}(x) - Q_j(x) \times W_{j-1}(x)$;
3. Repeats the calculation at step (2) for the value $j \coloneqq j + 1$, until the value $R_j \coloneqq 0$;
4. When the value $R_j \coloneqq 0$. The algorithm stops, and the value U_{j-1} is the inverse of the polynomial $A(x)$ on an irreducible polynomial.

Example 2. Suppose the irreducible polynomial $x^8 + x^4 + x^3 + x + 1$ of $GF(2^8)$. Find the inverse of $x^6 + x^4 + x + 1$.

Using the calculation of Extended Euclid's algorithm for polynomials obtained the inverse multiplication of the polynomial $x^6 + x^4 + x + 1$ with the module of an irreducible polynomial $x^8 + x^4 + x^3 + x + 1$ of $GF(2^8)$ is $x^7 + x^6 + x^3 + x$ as follows in **Table 1**.

Table 1. Extended Euclid's Algorithm for Polynomials Example 2

j	$Q_j(x)$	$R_j(x)$	$U_j(x)$	$W_j(x)$
-1	$-$	$x^8 + x^4 + x^3 + x + 1$	0	1
0	$-$	$x^6 + x^4 + x + 1$	1	0
1	$x^2 + 1$	x^2	$x^2 + 1$	1
2	$x^4 + x^2$	$x + 1$	$x^6 + x^2 + 1$	$x^4 + x^2$
3	x	x	$x^7 + x^3 + x^2 + x + 1$	$x^5 + x^3 + 1$
4	1	1	$x^7 + x^6 + x^3 + x$	$x^5 + x^4 + x^3 + x^2 + 1$
5	x	0	$x^8 + x^4 + x^3 + x + 1$	$x^6 + x^4 + x + 1$

Furthermore, polynomials in $GF(2^8)$ are also represented as hexadecimal numbers to facilitate the formation of the Substitution Box. For the process of changing a polynomial into a hexadecimal number, the process is done by converting a polynomial into a binary with a length of 8 bits obtained from the coefficient of a variable in the polynomial, namely

$$b_7b_6b_5b_4b_3b_2b_1b_0$$

and then every four string bits of binary values are converted to hexadecimal or vice versa by using **Table 2**, as follows:

Table 2. Conversion Table Binary to Hexadecimal

Binary	Hex	Binary	Hex	Binary	Hex	Binary	Hex
0000	0	0100	4	1000	8	1100	C
0001	1	0101	5	1001	9	1101	D
0010	2	0110	6	1010	A	1110	E
0011	3	0111	7	1011	B	1111	F

Example 3. Suppose the given polynomial is $x^6 + x^4 + x + 1$, then the binary 8-bit string of the polynomial is 01010011, and the hexadecimal value is 53. The inverse is $x^7 + x^6 + x^3 + x$ with the 8-bit binary string 11001010. and the value is hexadecimal CA. So, the inverse of 53 is CA. Furthermore, for every polynomial in $GF(2^8)$ the inverse element as follows in **Table 3.** All of polynomials must be written in hexadecimal form.

Table 3. Inverse polynomial in GF(2^8)

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***A***	***B***	***C***	***D***	***E***	***F***
0	00	01	$8D$	$F6$	CB	52	$7B$	$D1$	$E8$	$4F$	29	$C0$	$B0$	$E1$	$E5$	$C7$
1	74	$B4$	AA	$4B$	99	$2B$	60	$5F$	58	$3F$	FD	CC	FF	40	EE	$B2$
2	$3A$	$6E$	$5A$	$F1$	55	$4D$	$A8$	$C9$	$C1$	$0A$	98	15	30	44	$A2$	$C2$
3	$2C$	45	92	$6C$	$F3$	39	66	42	$F2$	35	20	$6F$	77	BB	59	19
4	$1D$	FE	37	67	$2D$	31	$F5$	69	$A7$	64	AB	13	54	25	$E9$	09
5	ED	$5C$	05	CA	$4C$	24	87	BF	18	$3E$	22	$F0$	51	EC	61	17
6	16	$5E$	AF	$D3$	49	$A6$	36	43	$F4$	47	91	DF	33	93	21	$3B$
7	79	$B7$	97	85	10	$B5$	BA	$3C$	$B6$	70	$D0$	06	$A1$	FA	81	82
8	83	$7E$	$7F$	80	96	73	BE	56	$9B$	$9E$	95	$D9$	$F7$	02	$B9$	$A4$
9	DE	$6A$	32	$6D$	$D8$	$8A$	84	72	$2A$	14	$9F$	88	$F9$	DC	89	$9A$
A	FB	$7C$	$2E$	$C3$	$8F$	$B8$	65	48	26	$C8$	12	$4A$	CE	$E7$	$D2$	62
B	$0C$	$E0$	$1F$	EF	11	75	78	71	$A5$	$8E$	76	$3D$	BD	BC	86	57
C	$0B$	28	$2F$	$A3$	DA	$D4$	$E4$	$0F$	$A9$	27	53	04	$1B$	FC	AC	$E6$
D	$7A$	07	AE	63	$C5$	DB	$E2$	EA	94	$8B$	$C4$	$D5$	$9D$	$F8$	90	$6B$
E	$B1$	$0D$	$D6$	EB	$C6$	$0E$	CF	AD	08	$4E$	$D7$	$E3$	$5D$	50	$1E$	$B3$
F	$5B$	23	38	34	68	46	03	$8C$	DD	$9C$	$7D$	$A0$	CD	$1A$	41	$1C$

Before performing an affine transformation, each polynomial on $GF(2^8)$ will be represented in the form of a column vector containing the 8-bit binary value of the string of the polynomial with the following rule, assuming given a polynomial in $GF(2^8)$ i.e.

$$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + b_0x^0$$

Then the 8-bit binary string is $b_7b_6b_5b_4b_3b_2b_1b_0$ and the vector representation is

$$b = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix}$$

Example 4. Based on **Example 2** and **Example 3**, inverse multiplication for the given polynomial $x^6 + x^4 + x + 1$ with the module of an irreducible polynomial $x^8 + x^4 + x^3 + x + 1$ of $GF(2^8)$ is $x^7 + x^6 + x^3 + x$, which can be represented as 11001010 and the vector as follows:

$$b = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

After obtaining the inverse for each value on $GF(2^8)$ and further determining the affine matrix and affine key vector used in affine transformation, this study used an affine matrix M of size 8×8 and an affine vector v of length 1×8 as follows:

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}; v = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Further, from the affine matrix and the defined affine vector, an affine transformation is carried out by performing the module 2 matrix multiplication and the operation XOR with the affine vector by the result of the multiplication of the affine matrix. These values are then entered in the Substitution Box (S-Box) in **Table 4.** and the affine transformation can be written as follows:

$$s_{ij} = M \cdot b_{ij} \oplus v \tag{3}$$

where s, b, and v are column vectors 1×8 with s being the result vector for the hexadecimal value ij; b is a vector of the inverse polynomial value ij; and also v is the affine vector, M is the affine matrix, and the "$\oplus$" is the XOR operation, with i being the hexadecimal value in the S-Box row and j being the value hexadecimal in the S-box column. Constructing the S-box using an affine matrix in order to improve image encryption security.

Example 5. Suppose we find the entry of S-Box of an irreducible polynomial $x^8 + x^4 + x^3 + x + 1$ of $GF(2^8)$ in cell hexadecimal 53 (row of hexadecimal 5 and column of hexadecimal 3), then the formula of transformation affine (3) is used by

$$s_{53} = M \cdot b_{53} \oplus v$$

Table 4. Substitution Box Format

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***A***	***B***	***C***	***D***	***E***	***F***
0	s_{00}	s_{01}	s_{02}	s_{03}	s_{04}	s_{05}	s_{06}	s_{07}	s_{08}	s_{09}	s_{0A}	s_{0B}	s_{0C}	s_{0D}	s_{0E}	s_{0F}
1	s_{10}	s_{11}	s_{12}	s_{13}	s_{14}	s_{15}	s_{16}	s_{17}	s_{18}	s_{19}	s_{1A}	s_{1B}	s_{1C}	s_{1D}	s_{1E}	s_{1F}
2	s_{20}	s_{21}	s_{22}	s_{23}	s_{24}	s_{25}	s_{26}	s_{27}	s_{28}	s_{29}	s_{2A}	s_{2B}	s_{2C}	s_{2D}	s_{2E}	s_{2F}
3	s_{30}	s_{31}	s_{32}	s_{33}	s_{34}	s_{35}	s_{36}	s_{37}	s_{38}	s_{39}	s_{3A}	s_{3B}	s_{3C}	s_{3D}	s_{3E}	s_{3F}
4	s_{40}	s_{41}	s_{42}	s_{43}	s_{44}	s_{45}	s_{46}	s_{47}	s_{48}	s_{49}	s_{4A}	s_{4B}	s_{4C}	s_{4D}	s_{4E}	s_{4F}
5	s_{50}	s_{51}	s_{52}	s_{53}	s_{54}	s_{55}	s_{56}	s_{57}	s_{58}	s_{59}	s_{5A}	s_{5B}	s_{5C}	s_{5D}	s_{5E}	s_{5F}
6	s_{60}	s_{61}	s_{62}	s_{63}	s_{64}	s_{65}	s_{66}	s_{67}	s_{68}	s_{69}	s_{6A}	s_{6B}	s_{6C}	s_{6D}	s_{6E}	s_{6F}
7	s_{70}	s_{71}	s_{72}	s_{73}	s_{74}	s_{75}	s_{76}	s_{77}	s_{78}	s_{79}	s_{7A}	s_{7B}	s_{7C}	s_{7D}	s_{7E}	s_{7F}
8	s_{80}	s_{81}	s_{82}	s_{83}	s_{84}	s_{85}	s_{86}	s_{87}	s_{88}	s_{89}	s_{8A}	s_{8B}	s_{8C}	s_{8D}	s_{8E}	s_{8F}
9	s_{90}	s_{91}	s_{92}	s_{93}	s_{94}	s_{95}	s_{96}	s_{97}	s_{98}	s_{99}	s_{9A}	s_{9B}	s_{9C}	s_{9D}	s_{9E}	s_{9F}
A	s_{A0}	s_{A1}	s_{A2}	s_{A3}	s_{A4}	s_{A5}	s_{A6}	s_{A7}	s_{A8}	s_{A9}	s_{AA}	s_{AB}	s_{AC}	s_{AD}	s_{AE}	s_{AF}
B	s_{B0}	s_{B1}	s_{B2}	s_{B3}	s_{B4}	s_{B5}	s_{B6}	s_{B7}	s_{B8}	s_{B9}	s_{BA}	s_{BB}	s_{BC}	s_{BD}	s_{BE}	s_{BF}
C	s_{C0}	s_{C1}	s_{C2}	s_{C3}	s_{C4}	s_{C5}	s_{C6}	s_{C7}	s_{C8}	s_{C9}	s_{CA}	s_{CB}	s_{CC}	s_{CD}	s_{CE}	s_{CF}
D	s_{D0}	s_{D1}	s_{D2}	s_{D3}	s_{D4}	s_{D5}	s_{D6}	s_{D7}	s_{D8}	s_{D9}	s_{DA}	s_{DB}	s_{DC}	s_{DD}	s_{DE}	s_{DF}
E	s_{E0}	s_{E1}	s_{E2}	s_{E3}	s_{E4}	s_{E5}	s_{E6}	s_{E7}	s_{E8}	s_{E9}	s_{EA}	s_{EB}	s_{EC}	s_{ED}	s_{EE}	s_{EF}
F	s_{F0}	s_{F1}	s_{F2}	s_{F3}	s_{F4}	s_{F5}	s_{F6}	s_{F7}	s_{F8}	s_{F9}	s_{FA}	s_{FB}	s_{FC}	s_{FD}	s_{FE}	s_{FF}

Based on **Table 2**, hexadecimal 53 has an 8-bit binary string is 01010011, so the vector is

$$b_{53} = [0\ 1\ 0\ 1\ 0\ 0\ 1\ 1]^T$$

Therefore, the calculation we have

$$s_{53} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

The result in an 8-bit string is 11101101, and the hexadecimal is ED. As the same step for every cell, we have the Substitution Box of an irreducible polynomial $x^8 + x^4 + x^3 + x + 1$, as follows in **Table 5**:

Table 5. Substitution Box of $x^8 + x^4 + x^3 + x + 1$

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***A***	***B***	***C***	***D***	***E***	***F***
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Example 6. Construction of the Substitution Box of an irreducible polynomial $x^8 + x^6 + x^4 + x^3 + x^2 + x + 1$ of $GF(2^8)$.

Using the same calculation as **Example 5,** for every cell, we obtain the substitution box as follows in **Table 6**.

Table 6. Substitution Box of $x^8 + x^6 + x^4 + x^3 + x^2 + x + 1$

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***A***	***B***	***C***	***D***	***E***	***F***
0	63	7C	AA	ED	C1	E6	24	88	32	3A	A1	3F	86	33	96	64
1	CB	E5	CF	14	44	45	0B	AC	D7	3C	4B	54	DF	A8	A6	80
2	37	B9	20	A5	73	BC	D8	5E	F0	1D	70	A9	11	0A	84	2D
3	7F	A2	8A	65	31	4E	F8	99	7B	D9	C0	09	81	29	92	FA
4	0F	EB	48	69	C2	41	00	DE	6B	B8	8C	8E	BE	BA	FD	4D
5	EC	BF	5C	A7	EA	9E	40	CC	1C	CA	91	62	D6	C4	02	78
6	2B	35	C5	AE	97	21	26	82	4A	F3	F5	36	E8	FE	1E	52
7	6F	59	3E	3B	B2	03	10	BB	12	2E	46	B6	9B	25	E9	27
8	55	A0	61	30	B0	98	66	DA	B3	D0	34	58	94	AB	FB	72
9	67	EF	C8	75	D2	2F	D3	17	8D	D4	C9	CE	2C	E7	74	43
A	A4	F4	0D	51	FC	A3	01	E2	E1	C3	DB	D1	B4	68	F2	5D
B	DC	F7	B7	16	1A	39	E3	6C	FF	3D	F6	13	95	50	EE	5A
C	47	2A	0E	1B	76	9A	85	57	5F	08	42	B5	87	90	93	7D
D	B1	79	6D	56	28	9F	8F	AF	E0	19	AD	D5	DD	C7	BD	71
E	23	6A	38	0C	8B	77	4F	7A	CD	7E	15	04	9C	18	49	E4
F	9D	05	83	53	F1	5B	89	C6	1F	F9	06	22	60	6E	07	04

Example 7. Construction of the Substitution Box of an irreducible polynomial $x^8 + x^7 + x^6 + x^5 + x^4 + x^2 + 1$ of $GF(2^8)$.

Using the same calculation as **Example 5**, for every cell, we obtain the substitution box as follows in **Table 7**.

Table 7. Substitution Box of $x^8 + x^7 + x^6 + x^5 + x^4 + x^2 + 1$

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	***A***	***B***	***C***	***D***	***E***	***F***
0	63	7C	FF	8B	2D	3B	17	7E	57	97	4F	C8	59	F4	FE	49
1	79	98	0A	45	75	72	B6	F3	6D	EE	A8	D2	AD	18	65	58
2	6E	68	9E	90	C4	E8	70	4A	7B	39	EB	E7	89	0C	38	D5
3	77	80	A5	D7	95	9C	BB	E9	04	13	CD	54	73	20	ED	93
4	F6	33	E6	5F	8E	AF	9A	40	A3	A7	B5	AE	EA	6C	E4	23
5	6F	DD	4E	DF	34	F0	21	61	16	07	D4	F2	CE	D1	2B	36
6	7A	37	92	87	00	32	2A	E2	0B	99	8F	BE	1C	22	35	1A
7	D0	EC	48	84	27	25	F8	46	6B	C0	C2	55	24	1F	1B	96
8	A9	CF	4B	B7	A1	81	7D	AA	86	BF	05	FD	9F	71	E1	C6
9	10	69	01	FB	08	3E	85	F9	B4	9D	F7	FC	A0	14	43	BC
A	76	1D	2F	4D	F5	3F	2E	30	DB	E3	B9	19	42	E0	62	B1
B	D9	56	51	74	AB	FA	B8	5C	A6	D6	3A	CA	47	28	DA	5E
C	EF	29	5A	AC	9B	88	02	0F	C1	60	CB	8C	C7	91	B0	D8
D	44	41	1E	3C	06	C9	8D	66	DC	03	C3	3D	5B	C5	CC	DE
E	BA	50	A4	2C	E5	82	83	0D	52	31	53	A2	BD	0E	F1	6A
F	67	64	B2	09	B3	15	78	26	D3	7F	5D	94	4C	11	8A	12

3.2 Criteria

3.2.1 Hamming Distance & Hamming Correlation

Hamming distance is a metric used to measure the difference between two strings of equal length. It is defined as the number of positions at which the corresponding symbols are different. In other words, it measures the minimum number of substitutions required to change one string into another, or the minimum number of errors that could have transformed one string into another **[20]**, **[21]**. Hamming distance is one of several string metrics for measuring the edit distance between two sequences. The Hamming distance between two 8-bit strings is a measure of the difference or dissimilarity between them. It's calculated by comparing each pair of corresponding bits in the two strings and counting the number of positions at which they differ. The XOR (exclusive OR) operation is commonly used to calculate the Hamming distance because it results in a 1 (true) when two bits are different and 0 (false) when they are the same.

Here's how you can calculate the Hamming distance between two 8-bit strings using XOR:

1. Make sure the two 8-bit strings have the same length. If they don't, you may need to pad the shorter string with leading zeros.
2. Perform a bitwise XOR operation between the two 8-bit strings. This means comparing each pair of corresponding bits and applying the XOR operation to them.
3. Count the number of 1s (true values) in the result of the XOR operation. This count represents the Hamming distance between the two 8-bit strings.

In form of mathematics the formula of Hamming distance between two n-bit string $x = (x_{n-1} \dots x_2 x_1 x_0)$ and $y = (y_{n-1} \dots y_2 y_1 y_0)$ using operation XOR ($\oplus$) is given by:

$$d(x,y) = \sum_{j=0}^{n} x_j \oplus y_j \tag{4}$$

Example 8. Hamming distance between strings 11001100 and 01110100 is 4.
In this case, both bit strings have the same length. Perform the XOR operation:

$$11001100 \oplus 01110100 = 10111000$$

So, the Hamming distance is:

$$d = \sum(11001100 \oplus 01110100) = 1 + 0 + 1 + 1 + 1 + 0 + 0 + 0 + 0 = 4$$

In this study, we utilize the Hamming distance to measure how many bit positions differ between two binary values (the input and output values in an S-box) as shown in **Table 8**. This metric is valuable for evaluating the behavior and properties of an S-box. Additionally, we explore Hamming correlation, a concept closely related to the Hamming distance, but with a focus on measuring the similarity or correlation between two binary sequences rather than their dissimilarity. Hamming correlation is not a commonly used term, but it appears to refer to the correlation between two binary strings based on their Hamming distance **[20]**, **[21]**. Maximizing the Hamming distance between two binary strings is not the same as minimizing their correlation. Hamming distance is frequently encountered in the analysis of block codes. It is also used in character recognition using template matching schemes. In addition, Hamming distance is used to compare two binary strings of the same length. Hamming correlation finds applications in various fields, such as information theory, error detection and correction, and signal processing. Mathematically, the Hamming correlation (H_c) can be expressed as follows:

$$H_c(x, y) = \frac{d(x, y)}{n} \tag{5}$$

The result is a value between 0 and 1, where a value of 0 indicates that the two strings are identical and value of 1 indicates that the two strings have no matching bits.

In **Table 8**, the "S-Box Code" column labels each S-box constructed by its hexadecimal polynomial code. The "Average d" column represents the average Hamming distance value between all input and output bits in the S-Box. The last column, "Average H_C" indicates the Hamming correlation value of the S-Box. A higher value of "d" corresponds to a smaller correlation value, signifying distinctions between the S-box's input and output. Consequently, a higher "d" value implies that the S-box possesses superior scrambling capabilities. Using a correlation test between "Average d" and "Average H_C" we obtain a significant negative correlation, because this implies that since the S-boxes have a higher average Hamming distance (more dispersion and diffusion), they tend to have a lower Hamming correlation (less linear relationships between input and output). This is generally considered favorable for cryptographic S-boxes as it indicates a stronger level of security.

3.2.2 Non-Linearity (NL)

The non-linearity of a Boolean function can be defined as the smallest distance between the function and the set of affine functions. It is denoted by Ψ is mathematically represented as:

$$\Psi = \min(N_\mu) \tag{6}$$

with

$$N_j = \min[d(f, g)], \text{ where } g \in A_n \tag{7}$$

where A_n is the set of all the affine functions. And

$$d(f, g) = 2^{n-1} - 2^{-1}(\langle \eta, \beta \rangle) \tag{8}$$

where η, β represent the binary sequence of f, g respectively and $\langle \eta, \beta \rangle$ define the scalar product of sequence, Hence, for a function $f: B^n \to B$

$$N_f = 2^{n-1} - 2^{-1}[\max(\langle \eta, \beta_j \rangle)] \tag{9}$$

Where β_j belongs to sequence of all linear function.

3.3 Discussion and Future Research

In comparison, the results from **Table 8** in this study provide a comprehensive assessment of various S-boxes used in encryption algorithms. The focus here is on evaluating S-boxes based on "Average d" and "Average H_C" to enhance security and unpredictability. The S-box with the hexadecimal code 1BD stands out as the most favorable choice, boasting an average d value of 4.1953 and an average H_C value of 0.4756. This suggests a high level of security. Conversely, the S-box with hexadecimal code 169 is identified as the least desirable option, with an average d value of 3.8750 and an average H_C value of 0.5156, indicating a lower level of security. Interestingly, when comparing these results to those in **[22]**, which focuses on image security, and in this article **[9]**, which explores dynamic S-boxes in the AES algorithm, it becomes apparent

that the emphasis in this study is on cryptographic system security rather than image encryption or dynamic S-box flexibility. While all three studies aim to enhance security, the specific focus areas and evaluation criteria differ, highlighting the adaptability of S-boxes in various contexts.

For future research, it will be valuable to explore additional metrics beyond Hamming distance and Hamming correlation. The research should aim to bridge the gap between theoretical analysis and practical security concerns, ultimately contributing to the development of more robust encryption algorithms across a range of domains. Several other algebraic structures can also be found to construct such S-boxes to improve their robustness and can also be implemented in any size of S-Boxes, like 4×4, 8×8, and 12×12 [23]–[26]. Furthermore, different crypto-analysis techniques can be explored for such S-boxes.

4. CONCLUSIONS

In conclusion, Table 8 provides a comprehensive overview of various S-boxes, crucial components in encryption algorithms. These S-boxes undergo evaluation based on three key parameters: "Average d", "Average H_C" and “NL (Non-linearity value).” The overarching objective is to select S-boxes with minimal correlation, ideally approaching a value of 0. To maximize security and minimize predictability in cryptographic systems. Upon closer scrutiny, it becomes apparent that specific S-boxes consistently demonstrate lower correlations, nearing 0, signifying superior performance in security and unpredictability.

Based on the result, the best S-Box is constructed by a polynomial with the hexadecimal 1BD, boasting an average d value of 4.1953 and an average H_C value of 0.4756. Conversely, the least desirable S-Box is constructed by hexadecimal 169, with an average d value of 3.8750 and an average H_C value of 0.5156. The original S-Box or Rijndael version possesses an average d value of 3.9922 and an average H_C value of 0.5010.

And for the non-linearity of a Boolean function, quantified by the smallest distance between the functions, varies among S-boxes, with some reaching a maximum of 112 and a minimum of 85. In addition, the higher correlations observed in certain S-boxes highlight their increased susceptibility to cryptographic vulnerabilities. As a result, it is crucial for cryptographers and security professionals to conduct thorough analyses and potentially avoid using these S-boxes in security-sensitive applications. By prioritizing S-boxes with minimal correlations, cryptographic systems can be significantly strengthened, providing greater resilience against potential attacks and ensuring the protection of sensitive data in the ever-evolving digital landscape.

Table 8. Average Hamming Distance and Hamming Correlation of S-Box

S-Box Code	Average d	Average H_C	NL	S-Box Code	Average d	Average H_C	NL
11B	3.9922	0.5010	112	**18B**	3.9141	0.5107	110
11D	4.0156	0.4980	95	**18D**	3.9141	0.5107	108
12B	3.9844	0.5020	103	**19F**	4.0312	0.4961	85
12D	4.1172	0.4854	105	**1A3**	4.0469	0.4941	102
139	3.8984	0.5127	111	**1A9**	3.9141	0.5107	109
13F	4.0000	0.5000	96	**1B1**	4.0234	0.4971	101
14D	4.1016	0.4873	105	**1BD**	4.1953	0.4756	111
15F	4.0781	0.4902	100	**1C3**	4.0234	0.4971	92
163	3.9062	0.5117	112	**1CF**	4.0000	0.5000	105
165	4.1250	0.4844	106	**1D7**	3.9453	0.5068	106
169	3.8750	0.5156	107	**1DD**	4.0938	0.4883	105
171	4.0156	0.4980	110	**1E7**	4.0703	0.4912	89
177	4.0781	0.4902	98	**1F3**	4.0547	0.4932	112
17B	4.0156	0.4980	110	**1F5**	4.0547	0.4932	104
187	3.9297	0.5088	98	**1F9**	3.9844	0.5020	108

ACKNOWLEDGMENT

This research was supported by Universitas Kristen Satya Wacana through the internal research scheme in 2023 with Decree No. 039/RIK-RPM/8/2023. We are grateful for their contribution to this research. Additionally, we would like to express our appreciation to the anonymous reviewers for their valuable feedback that significantly enhanced the quality of our paper.

REFERENCES

[1] M. S. de Alencar, *Cryptography and Network Security*. 2022. doi: 10.1201/b11517-4.
[2] C. Paar and J. Pelzl, *Understanding Cryptography*. 2010. doi: 10.1007/978-3-642-04101-3.
[3] J. Daemen and V. Rijmen, "The design of Rijndael: The advanced encryption standard (AES): Second Edition," in *Information Security and Cryptography*, 2020.
[4] Alamsyah, B. Prasetyo, and Y. Muhammad, "S-box Construction on AES Algorithm using Affine Matrix Modification to Improve Image Encryption Security," *Sci. J. Informatics*, vol. 10, no. 2, 2023, doi: 10.15294/sji.v10i2.42305.
[5] C. Christopher, A. Gunawan, and S. Prima, "Encrypted Short Message Service Design Using Combination of Modified Advanced Encryption Standard (AES) and Vigenere Cipher Algorithm," *Eng. Math. Comput. Sci. J.*, vol. 4, no. 2, 2022, doi: 10.21512/emacsjournal.v4i2.8273.
[6] A. Nakashima, R. Ueno, and N. Homma, "AES S-Box Hardware With Efficiency Improvement Based on Linear Mapping Optimization," *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 69, no. 10, 2022, doi: 10.1109/TCSII.2022.3185632.
[7] C. Cid, S. Murphy, and M. Robshaw, "Algebraic aspects of the advanced encryption standard," *Algebr. Asp. Adv. Encryption Stand.*, pp. 1–145, 2006, doi: 10.1007/978-0-387-36842-9.
[8] R. H. Prayitno, S. A. Sudiro, S. Madenda, and S. Harmanto, "Hardware Implementation of Galois Field Multiplication for Mixcolumn and Inversemixcolumn Process in Encryption-Decryption Algorithms," *J. Theor. Appl. Inf. Technol.*, vol. 100, no. 14, 2022.
[9] H. Susanto, Alamsyah, and A. T. Putra, "Security Improvement of the 256-BIT AES Algorithm With Dynamic S-Box Based on Static Parameter as the Key for S-Box Formation," *J. Adv. Inf. Syst. Technol.*, vol. 4, no. 1, pp. 33–41, 2022, doi: 10.15294/jaist.v4i1.59976.
[10] N. Angraini and Y. Suryanto, "Modification Advanced Encryption Standard (AES) Algorithm with Perfect Strict Avalanche Criterion S-Box," *J. Tek. Inform.*, vol. 3, no. 4, 2022, doi: 10.20884/1.jutif.2022.3.4.352.
[11] C. Blondeau, G. Leander, and K. Nyberg, "Differential-Linear Crypto-analysis Revisited," *J. Cryptol.*, vol. 30, no. 3, 2017, doi: 10.1007/s00145-016-9237-5.
[12] H. Kim *et al.*, "A New Method for Designing Lightweight S-Boxes with High Differential and Linear Branch Numbers, and its Application," *IEEE Access*, vol. 9, 2021, doi: 10.1109/ACCESS.2021.3126008.
[13] B. Susanto, A. D. Wowor, and V. B. Liwandouw, "Desain S-Box Fleksibel: Regenerasi Konstanta dan Koefisien Fungsi Linier Berbasis CSPRNG Chaos," *J. Nas. Tek. Elektro dan Teknol. Inf.*, vol. 8, no. 1, 2019, doi: 10.22146/jnteti.v8i1.484.
[14] C. Planteen, "Primitive elements and irreducible polynomials of GF (256)," 2019, pp. 1–6.
[15] A. Aidoo and K. B. Gyam, "Construction of Irreducible Polynomials in Galois elds, GF(2m) Using Normal Bases," *Asian Res. J. Math.*, vol. 14, no. 3, pp. 1–15, Jul. 2019, doi: 10.9734/arjom/2019/v14i330131.
[16] B. Nithya and V. Ramadoss, "Extension fields and Galois Theory," *Int. J. Math. Trends Technol.*, vol. 65, no. 7, 2019, doi: 10.14445/22315373/ijmtt-v65i7p507.
[17] S. Dey and R. Ghosh, "Mathematical Method to Search for Monic Irreducible Polynomials with Decimal Equivalents of Polynomials over Galois Field GF(pq)," *Circ. Comput. Sci.*, vol. 2, no. 11, 2017, doi: 10.22632/ccs-2017-252-68.
[18] A. Chandoul and A. M. Sibih, "Note on irreducible polynomials over finite field," *Eur. J. Pure Appl. Math.*, vol. 14, no. 1, 2021, doi: 10.29020/NYBG.EJPAM.V14I1.3898.
[19] K. Kobayashi, N. Takagi, and K. Takagi, "An algorithm for inversion in GF(2m) suitable for implementation using a polynomial multiply instruction on GF(2)," in *Proceedings - Symposium on Computer Arithmetic*, 2007. doi: 10.1109/ARITH.2007.9.
[20] T. Kaida and J. Zheng, "Hamming distance correlation for q-ary constant weight codes," in *2010 International Symposium On Information Theory & Its Applications*, Oct. 2010, pp. 842–845. doi: 10.1109/ISITA.2010.5649591.
[21] G. S. Shehu, A. M. Ashir, and A. Eleyan, "Character recognition using correlation & hamming distance," in *2015 23nd Signal Processing and Communications Applications Conference (SIU)*, May 2015, pp. 755–758. doi: 10.1109/SIU.2015.7129937.
[22] A. Alamsyah, B. Prasetiyo, and Y. Muhammad, "S-box Construction on AES Algorithm using Affine Matrix Modification to Improve Image Encryption Security," *Sci. J. Informatics*, vol. 10, no. 2, pp. 69–82, Apr. 2023, doi: 10.15294/sji.v10i2.42305.
[23] T. ul Haq and T. Shah, "12×12 S-box Design and its Application to RGB Image Encryption," *Optik (Stuttg).*, vol. 217, p. 164922, Sep. 2020, doi: 10.1016/j.ijleo.2020.164922.
[24] D. K. Sushma and M. Devi, "Design of S-box and IN V S -box using Composite Field Arithmetic for AES Algorithm," vol. 6, no. 13, pp. 1–4, 2018.
[25] T. Shah and A. Qureshi, "S-box on subgroup of galois field," *Cryptography*, vol. 3, no. 2, pp. 1–9, 2019, doi: 10.3390/cryptography3020013.
[26] S. Dey and R. Ghosh, "Irreducible or Reducible Polynomials over Gal- ois Field GF (p q) for Smart Applications .," pp. 1–18.