International Journal of Safety and Security Engineering
Vol. 15, No. 4, April, 2025, pp. 787-796
Journal homepage: http://iieta.org/journals/ijsse

Modified Lightweight Advanced Encryption Standard for Lightweight Embedded Applications

Andi Sama[1*], Meyliana[2], Yaya Heryadi[1], Taufik Roni Sahroni[3]

[1] Computer Science Department BINUS Graduate Program - Doctor of Computer Science, Bina Nusantara University, Jakarta 11480, Indonesia
[2] Information System Department School of Information System, Bina Nusantara University, Jakarta 11480, Indonesia
[3] Industrial Engineering Department BINUS Graduate Program-Master of Industrial Engineering, Bina Nusantara University, Jakarta 11480, Indonesia

Corresponding Author Email: andi.sama@binus.ac.id

https://doi.org/10.18280/ijsse.150415

Received: 16 February 2025
Revised: 16 March 2025
Accepted: 22 March 2025
Available online: 30 April 2025

Keywords:
advanced encryption standard, AES, AES modification, embedded applications, lightweight cryptography

ABSTRACT

Cryptography enables data integrity, authentication, non-repudiation, and confidentiality. AES, the Advanced Encryption Standard, is part of data confidentiality in cryptography and among the strongest and most effective for implementation difficulty and security for block cipher – the symmetric algorithm that encrypts and decrypts plaintext using the same key. Lightweight AES is a modified AES for lightweight applications like Embedded Systems or the Internet of Things (IoT). Research suggested modifications to 128-bit AES for lightweight applications prioritizing MixColumns, SubBytes, ShiftRows, round reduction, and Key expansion. This paper presents MLAES, the modified lightweight 128-bit AES, by modifying subsets of AES components: the S-box table within the SubBytes function and the MixColumns constants within the MixColumns function, including reductions to number of rounds. Although using a different method, the findings conclude that the MLAES algorithm, which is evaluated on the same dataset consisting of plaintexts and keys as the state-of-the-art, has the result of an average avalanche effect at 53.6719%. MLAES is 0.6250% better than state-of-the-art (53.0469%).

1. INTRODUCTION

ICS, as part of Industry 4.0, is defined as in study [1] "supervisory and regulatory control systems used to control production systems within a local area, such as a factory, using centralized data acquisition".

The focus of IT security for a long time has been the CIA [2-4]. However, integrity and availability are the top priorities for OT security [5], followed by confidentiality, with safety being considered, like ensuring data flow security [6].

As part of OT, such as in manufacturing facilities, ICS consists of specially made devices (embedded systems, such as SoC or hardware-based FPGA [7]) with unique characteristics: low-cost and constrained memory, processing, and I/O. Cryptographic features are generally absent from embedded systems [2].

The Guide for OT Security (Protect, Data Security in Cyber Security framework) suggests encrypting data to guarantee its integrity and confidentiality in transit and at rest. The guide is one of the standards set by the NIST [5].

To prevent unauthorized exposure, ISO/IEC 62443 mandates maintaining the secrecy of information on communication channels (data in transit) and data repositories (data at rest). ISA99, developed by ISA, merged with ISO/IEC 27001 and became ISO/IEC 62443.

Secrecy is the goal of cryptography [8]. The word "cryptos" comes from the Greek word "Kryptos," which means "secret or hidden." Cryptography makes data confidentiality (it's a secret), authentication and access control (truly believes that this is you), non-repudiation (the sender cannot deny that they sent the message), and data integrity (tamper-proof, the information we receive is the same as it was sent) possible.

According to study [9], symmetric encryption encrypts and decrypts plaintext using a single key, i.e., the same key is used for both processes. The symmetric algorithm is sometimes called the shared-key algorithm since the same key is utilized for encryption and decryption.

Asymmetric encryption encrypts and decrypts plaintext using distinct keys [9]. The second key decrypts the ciphertext, while the first encrypts the plaintext. The first and second keys have a mathematical relationship.

1.1 Confusion and diffusion

In paper [10], Shannon described the properties of confusion and diffusion. Encryption is based on these two characteristics.

Confusion means that the ciphertext (the encrypted message) looks very different from the plaintext. The process should be non-linear, with no easily recognizable pattern. Confusion complicates the relationship between the key and ciphertext. A cipher with confusion should work, such as each bit in the ciphertext, depending on the many bits in the key.

Diffusion means that many characters change in the ciphertext if we change a character in the plaintext. The ciphertexts will differ significantly even if there is a change in only one character in the plaintext. Diffusion spreads out the statistical structure of the plaintext. A cipher with diffusion should work such that changing any bit of the plaintext results in a change of about 50% of the ciphertext.

A good cipher should have both confusion and diffusion. "In block ciphers, the common techniques to achieve confusion and diffusion are substitution boxes, permutations, and key expansion" [11].

1.2 Advanced Encryption Standard (AES)

Numerous symmetric and asymmetric encryption techniques exist. When it was discovered to be weak, the widely used symmetric algorithm DES/3DES was replaced [12]. AES is a strong symmetric block cipher [9, 13] with alternative block sizes for key lengths of 128 (16 Bytes), 192 (24 Bytes), or 256 bits (32 Bytes). Many cryptographic services, particularly those that guarantee data confidentiality, are built on block ciphers.

For instance, when using a block cipher with a 128-bit block size, the plaintext and the key must be equally divided into 16-byte blocks for encryption. To finish the last block, the remaining bytes less than 16 will be expanded to 16 bytes using padding, with the selected pattern. Every block goes through independent encryption and decryption [11].

Table 1 summarizes the key lengths of three AES algorithm variations.

Table 1. Three key lengths of AES algorithm

AES Algorithm	Key Length (BITs)	# of Round
AES-128	128	10
AES-192	192	12
AES-256	256	14

In today's environment, when numerous IoT devices need to connect wirelessly, LWC is an important and rapidly expanding field. IoT devices need simple methods for communication security because they have limited resources. LWC is a set of encryption technology solutions that include low-complexity computing devices.

It seeks to increase the use of cryptography on low-resource devices while maintaining high security [11].

AES is regarded as one of the strongest and most effective algorithms in terms of implementation difficulty and security [13]. According to studies [13-15], and others, modifying the current AES algorithm for lightweight applications is possible. This includes hardware implementations like voice message encryption in FPGA [16], compact AES-like S-box implementation in FPGA [17], and compact 8-bit S-box implementation for generic hardware [18].

The top three research topics in experimentation are the IoT as the platform, OT—On-Premises as the environment, and laboratory as experiment deployment. For the AES key length selection, the research trend focused on 128 bits. Out of 23 articles, research trends on AES algorithm component modification include modifying the MixColumns (47.83%) [16, 19-28], SubBytes (30.43%) [21, 24, 26-29], ShiftRows (21.74%) [23, 24, 26, 28, 30], reducing the number of rounds (17.39%) [19, 25, 28, 31], and Key expansion (13.04%) [24, 26, 29].

The top three effects of modifying the AES algorithm to its performance as lightweight AES are speed/time, hardware design efficiency, and the avalanche effect.

An increase in the avalanche effect translates to an increase in security for the AES algorithm [32]. However, this area of research has been underexplored.

The state-of-the-art [32, 33] stated that the average avalanche effect was 53.0469%, 4.2969% better for plaintext bitflip than AES. The author provided the dataset in 10 sets of 16-byte plaintext pairs, in which the second plaintext has a bitflip of the first plaintext. The dataset includes a single 16-byte key.

This study has three research questions (RQ). RQ 1: "What are the AES-128 components to modify for a lightweight and better AES in terms of increased security? RQ2: "What are the metrics to measure such an algorithm?" RQ3: "Can the modified algorithm be better than the state of the art?"

This paper's main theoretical contribution is the Modified Lightweight AES-128 algorithm, or MLAES, aimed at low-power, lightweight computing platforms. MLAES is more secure than the state-of-the-art, outperforming it by 0.6250%. The practical contribution is the possible implementation of MLAES for low-power, lightweight embedded computing systems, such as PLC or the IoT.

The structure of the paper is shown in Figure 1. The introduction comes first, then the method and main results, and finally, the discussion and conclusion parts. There are three steps in the method: reverse engineering of AES-128, restructuring the AES-128 by modifying selected components (S-box table, MixColumns, and round) for increased security and lightweight, and measuring the increase in security with hamming distance and the avalanche effect.

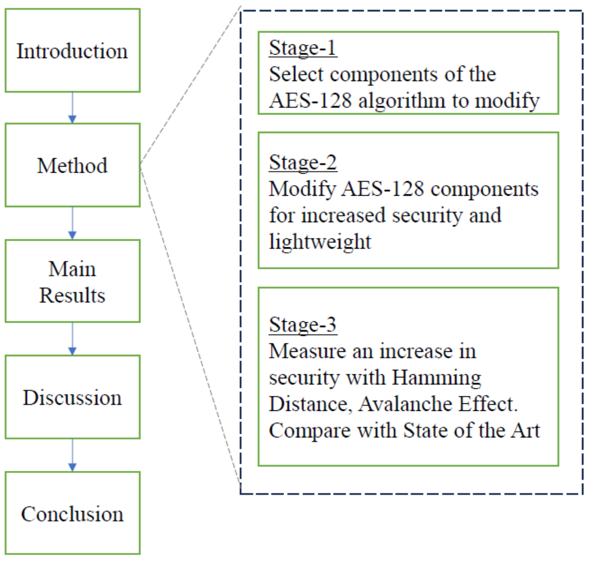
Introduction
Method
Main Results
Discussion
Conclusion
Stage-1
Select components of the AES-128 algorithm to modify
Stage-2
Modify AES-128 components for increased security and lightweight
Stage-3
Measure an increase in security with Hamming Distance, Avalanche Effect. Compare with State of the Art

Figure 1. Paper organization

2. METHOD

Figure 2 illustrates that the experiment involves three stages. First, we seek to understand how the AES algorithm works and the potential modifications of its components to achieve better security and lightweight applications. Second, we modify the selected components.

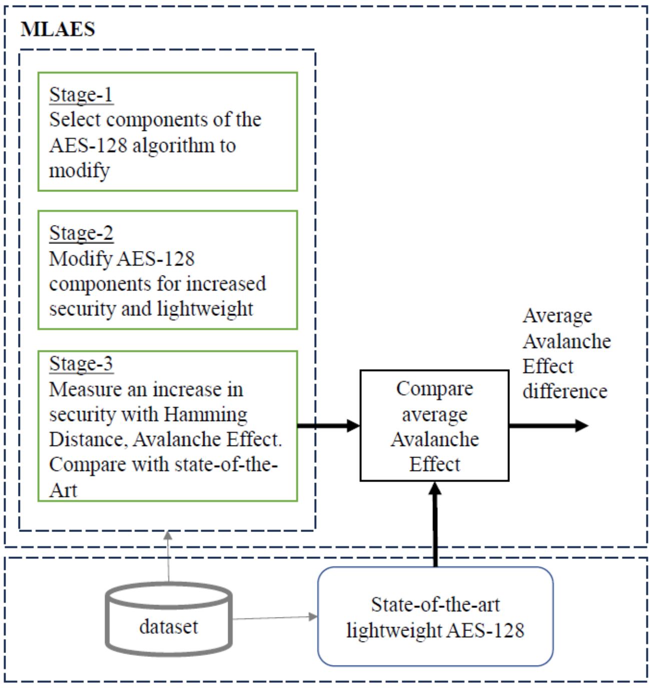
MLAES
Stage-1
Select components of the AES-128 algorithm to modify
Stage-2
Modify AES-128 components for increased security and lightweight
Stage-3
Measure an increase in security with Hamming Distance, Avalanche Effect. Compare with state-of-the-Art
Compare average Avalanche Effect
Average Avalanche Effect difference
dataset
State-of-the-art lightweight AES-128

Figure 2. Experiment method

Finally, using MLAES, we calculate the hamming distance and avalanche effect using the dataset [32, 33]. The average avalanche effect difference over a data series is compared to the state-of-the-art.

The AES-128 source code [34] with the C programming language, a Notepad++ editor, and the GCC compiler on the Windows 11 OS are supporting tools for modifying the AES-128 to MLAES.

2.1 Dataset

The same single 16-byte key, 11111111111111111110 (in hexadecimal), and ten sets of 16-byte plaintexts, 16-byte plaintexts with bitflip, and ciphertexts from the reference paper are used throughout the experiment.

2.2 Stage 1 - Select AES-128 components to modify

The overall structure of AES-128 (encryption, from plaintext to ciphertext; and decryption, from ciphertext to plaintext) is shown in Figure 3 [11].

The AddRoundKey function performs the XOR operation of the plaintext with the corresponding AES key generated from the Key expansion function.

During encryption, the SubBytes function substitutes the current value of each byte taken from the S-box ta, and the 16 by 16 table contains one hexadecimal byte to provide confusion. Decryption uses the InvS-box table [11].

Row 0 of the input state is kept unchanged in the ShiftRows transformation, while the other three rows, r = 1, 2, and 3, are all rotated to the left by r byte(s). This guarantees that one column's four bits are distributed across four distinct columns [11].

Strong diffusion is provided by the MixColumns function, which uses a linear function to work on the state column-by-column during encryption. In the input state, every byte in a column is substituted by twice that byte, three times the subsequent byte, the subsequent byte, and the subsequent byte in the column. Each column circularly uses bytes. During decryption, the InvMixColumns function reverses the MixColumns transformation [11].

2.3 Stage 2 - MLAES, modification of the AES-128 algorithm

The modifications to the AES-128 algorithm are only for selected components: the S-Box table within the SubBytes function, the MixColumns function, and the number of rounds.

The KeyExpansion and Shiftrows functions are kept unchanged in their original form.

2.3.1 Modification to S-box table

The changes are made only to the S-box table. The SubBytes function, which generates confusion using the S-box table, is kept unchanged.

S-box transformation function:

$$Y = (X^2 + \sqrt{(X^{3.14})}) \bmod 16 \qquad (1)$$

Eq. (1) defines the S-box transformation function. By swapping the rows and columns at the center coordinates, the 16-row by 16-column AES S-box table, consisting of 1 byte each (unsigned), is transformed into the MLAES S-box table.

Table 2. The center coordinates (X, Y)

X	Y	X	Y	X	Y
0	0	5	5	10	9
1	2	6	4	11	4
2	6	7	6	12	1
3	14	8	10	13	1
4	8	9	0	14	3
				15	7

Table 3. The center coordinates in the S-box table

	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

Table 2 lists all calculated center coordinates. X represents rows containing unsigned bytes running from 0 to 15 sequentially. Y represents the columns $X, Y \in \{0..15\}$. For X = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, and 15, the formula calculated Y = 0, 2, 6, 14, 8, 5, 4, 6, 10, 0, 9, 4, 1, 1, 3, 7, in sequence, respectively.

Please refer to Table 3, which illustrates all the calculated center coordinates (highlighted).

Each value in the MLAES S-box table (Table 4) contains an unsigned byte consisting of two nibbles (the first 4 bits and the second 4 bits). This is the index to perform byte substitution in the MLAES SubBytes function.

Algorithm 1 illustrates how the S-box transformation function works.

For example, let's see the illustration of the first byte entering the SubBytes function (in AES and MLAES, the input to the SubBytes function is 128-bit, and the output is also 128-bit).

If the first byte is 0x4F (4F in hexadecimal, 01001111 in binary), then the first nibble is 0x4, and the second nibble is 0xF. The SubBytes function will look at the S-box table to find the value in row 0x4 and column 0xF.

For the AES-128 (as in Table 3), the value for the selected row and column is 0x84. Therefore, the first-byte 0x4F input is transformed to 0x84 first-byte output until the 16th byte (128 bits is 16 bytes).

Likewise, applying the same transformation using the MLAES S-box table (Table 4), the SubBytes function transforms 0x4F to 0x41 (row 0x4 and column 0xF in the MLAES S-box table contain 0x41).

2.3.2 Modification to number of rounds

MLAES reduces the number of rounds from 10 (in AES-128) to 8 to make it lightweight.

2.3.3 Modification to MixColumns constant

Following the modification of the S-box table and number of rounds, the MixColumns constant, used by the MixColumns function, is changed from{2, 3, 1, 1, 1, 2, 3, 1, 1, 1, 2, 3, 3, 1, 1, 2} as in the AES-128 to {1, 2, 1, 3, 3, 1, 2, 1, 1, 3, 1, 2, 2, 1, 3, 1}.

Algorithm 1: Swapping rows & columns at center coordinates

Input
AES S-box table as uint8 sboxArray [16];

Process
First, define center coordinates (X, Y) for swapping rows & columns of S-box table according to S-box transformation function (Equation 1), Y = f(X), where X are rows and Y are columns. The S-box table is 16 by 16 array. X, Y are in decimal.

```
For (int i=0; i<16; i++) do
    Y[i] = f(X);
End
centerRow=X; centerCol=Y;
```

Then, for every row in the S-box table, from the first row on the top to the last row at the bottom, switch at the centerRow and centerColumn.

```
For (int i=0; i<16; i++) do
    uint8 temp = sboxArray[centerRow][i];
    sboxArray[centerRow][i] = sboxArray[i][centerCol];
    sboxArray[i][centerCol] = temp;
End
```

Output
MLAES S-box table as uint8 sboxArray [16];

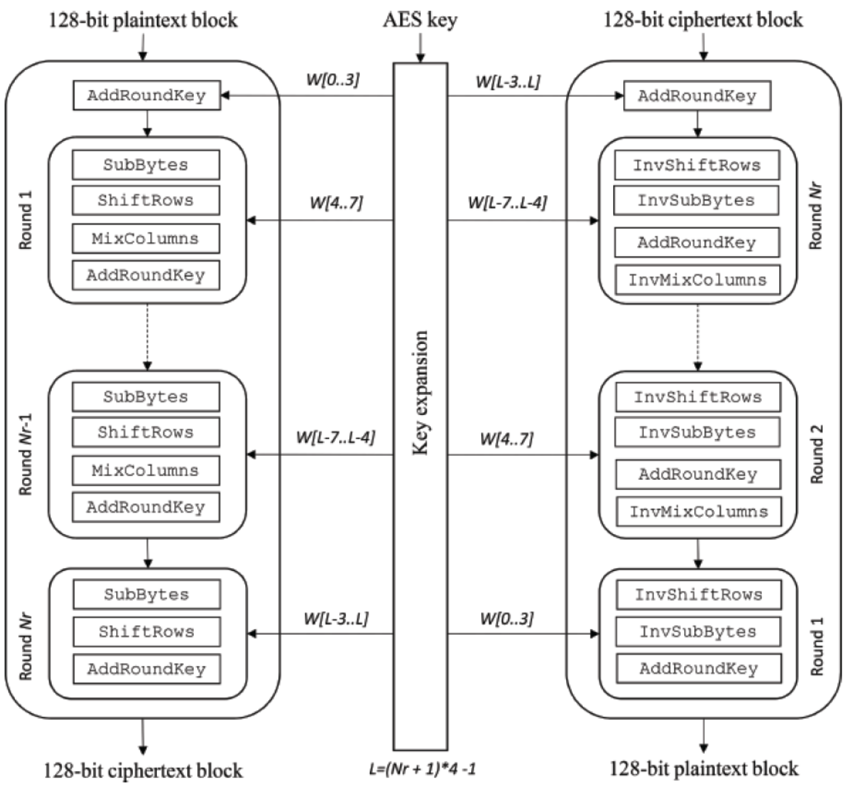

Figure 3. AES-128 algorithm

Table 4. MLAES S-box table

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

Table 5 illustrates the 4×4 matrix for these two MixColumns constants. On the left is the MixColumns constant used by AES 128-bit, and on the right is the MixColumns constant used by MLAES 128-bit.

Table 5. MixColumns constants for AES and MLAES

AES 128-bits				**MLAES 128-bits**			
2	3	1	1	1	2	1	3
1	2	3	1	3	1	2	1
1	1	2	3	1	3	1	2
3	1	1	2	2	1	3	1

2.4 Stage 3 - Measure hamming distance, avalanche effect

2.4.1 Hamming Distance (HD)

Eq. (2) illustrates the formula for HD: the number of bits changed in the ciphertext from the corresponding bit in the plaintext.

Calculating hamming distance:

$$HD = \sum_{n=0}^{NB-1}(Plaintext_n \; XOR \; Ciphertext_n) \qquad (2)$$

NB is the total number of bits in either ciphertext1 or ciphertext2, counted as 1 if the corresponding bit in the ciphertext1 and ciphertext2 are different; otherwise, it is 0. For AES-128, NB is 128, so the counter n runs from 0 to 127. plaintext1 and plaintext2 are the inputs to the cipher (AES or MLAES) before being converted to ciphertext1 and ciphertext2.

Ciphertext1 and ciphertext2 are the encrypted formats of plaintext1 and plaintext2. Because AES is a block cipher, NB (plaintext) = NB (ciphertext).

2.4.2 Avalanche Effect (AE)

The avalanche effect is a desirable cipher effect typically found in block ciphers and cryptographic hash functions. In this way, if the input is changed slightly, the output changes significantly. "Changing a few bits in the plaintext results in a lot of changes in the ciphertext, which is known as avalanche effect, i.e., a small change in either the key or the plaintext should cause a drastic change in the ciphertext" [11].

For example, the output changes significantly if we change one bit in the input. Every single bit of the output depends on every bit of the input.

The formalization of the avalanche effect, the "strict avalanche criterion," states that a change in a single bit of the input results in changes in each output bit with a probability of 50%.

Calculating the avalanche effect:

$$AE = \frac{NBC}{TNB} * 100\% \qquad (3)$$

The AES shows a strong avalanche effect [35]. A significant modification to the algorithm has even been explored, although it deviated significantly from the original AES algorithm [36].

Eq. (3) illustrates the formula to calculate the avalanche effect [33], AE.

NBC is the number of bits changed between two ciphertexts based on the original plaintext and the plaintext after the bitflip. For AES-128, NBC <= NB. NB is 128 for AES-128. NB is the total number of bits in the plaintext. TNB is the total number of bits in the ciphertext. Since AES is a block cipher, the number of bits of plaintext equals to the number of bits of the ciphertext.

3. RESULTS AND DISCUSSION

The hamming distance and average avalanche effect for the AES-128 are illustrated in Table 6. The average avalanche effect of 50.3906% fulfills Shannon's confusion and diffusion properties, in which changing any bit of the plaintext results in a change of about 50% of the ciphertext.

MLAES extends the average avalanche effect to more than what is provided by the AES-128.

3.1 AES component modifications in MLAES

Following the modifications of selected components of the AES-128 (modified the S-box table within the SubBytes function, the MixColumns function with selected MixColumns constant, and the number of rounds), the results for hamming distance and average avalanche effect over a series of datasets for MLAES are listed in Table 7.

The average avalanche effect with the MLAES S-box table, round reduction to 8, and the selected matrix constant is 53.6719%.

Reducing the round to less than eight does not improve the average avalanche effect.

Keeping the same MLAES S-box table and round = 8 and further experimenting with different combinations of MixColumns constants does not improve the average avalanche effect to more than 53.6719%.

There are other MixColumns constants when applied with MLAES, however, resulting in an average avalanche effect precisely the same as the reference paper [33] of 53.0469%, although using a different approach to modify the AES-128 components.

The reference paper chose a different method to modify AES-128. Only one component, the MixColumns function, was modified by replacing it with a permutation function that mapped 128-bit input to 128-bit output. The remaining components, including the number of rounds, were kept unchanged: key expansion, SubBytes, and ShiftRows.

Table 6. Hamming distance and avalanche effect for AES

Test	Plaintext (hex)	Ciphertext (hex)	HD	AE (%)
1	123456789abcdef0123456789abcdef0	171434671d73293b813735a3f0729fbf		
	123456789abcdef0123456789abcdef1	136ed3e12aae2b10c0816c286ba91095	65	50.7813
2	112233445566778899aabbccddeeff00	d0eaf9d89e42dd3997b755aae1fb9ac0		
	112233445566778899aabbccddeeff01	d337dd2f8ed0e59ae5e61e07f886704e	62	48.4375
3	1ee823570972bb0f30d05938c132d612	8a1c6abfb04f7c4f67ec9bbfbabf568c		
	1ee823570972bb0f30d05938c132d613	e4c3e8a6b336533e190a9846d1bc344c	64	50.0000
4	e1172357097244f030d059373ecd2944	ef1c0496e756a5e74a995cdad5063f15		
	e1172357097244f030d059373ecd2945	0471e37a2c75b2eca64c35d58d089054	69	53.9063
5	00112233445566778899aabbccddeeff	a69cc9f963aaf0e581f1bd07c7b6d1ca		
	00112233445566778899aabbccddeefe	edfa2d406e2e423df2dc75a5cc11abc6	60	46.8750
6	5452555354204e4f204f4e4521585858	dd1a152f9c15d48b0f4bf090434e39db		
	5452555354204e4f204f4e4521585859	bd3af1e3898c23a914655ac09b25bd85	57	44.5313
7	4a454e53454e53454154484f41434c41	6ffa9b92f6b843729d7ccb28e626f7cb		
	4a454e53454e53454154484f41434c40	6556e5e782aaa58754e5c9db978c07a9	64	50.0000
8	41636c612c4a616b6520526f756b6500	eed141cd534ac474ab5a030f23de5d64		
	41636c612c4a616b6520526f756b6501	1e38bd4e53000191c888a84ecde77eb1	63	49.2188
9	41434c414a494e44524f414c57594e4e	f79274633d7d5337b043801f752d224f		
	41434c414a494e44524f414c57594e4d	d780b39ae376ad4a5da6692072895520	76	59.3750
10	4d59204d45535341474520495320494e	e5fc7b53f83cdbf3560ce4afb2c6ef87		
	4d59204d45535341474520495320494d	8b7b3a8181ab16ea79eba07583c16931	65	50.7813
		Average Avalanche Effect:		50.3906

Table 7. Hamming distance and avalanche effect for MLAES

Test	Plaintext (hex)	Ciphertext (hex)	HD	AE (%)
1	123456789abcdef0123456789abcdef0	b7009d3694c0979b4e6f33e519de8e3f		
	123456789abcdef0123456789abcdef1	f1ca6c63f27888f12855477285a49508	68	53.1250
2	112233445566778899aabbccddeeff00	c3edfb0c710eda7a67a4fef4ade0797b		
	112233445566778899aabbccddeeff01	26ea0ce0352c591b6ac2adee0635fd96	62	48.4375
3	1ee823570972bb0f30d05938c132d612	1a8d2bba3a26cb54e366e9bb4713319d		
	1ee823570972bb0f30d05938c132d613	57ff371acc4f1cf62abc5328eff62176	65	50.7813
4	e1172357097244f030d059373ecd2944	5c059dfa55fc5c325e363a25ca924589		
	e1172357097244f030d059373ecd2945	87544927888cd008abcbb9ea2038ac7b	76	59.3750
5	00112233445566778899aabbccddeeff	c983e69895638b2655dc0b3058ef33d0		
	00112233445566778899aabbccddeefe	abef8127c23ddf43d8b63ce8bb20d7c1	70	54.6875
6	5452555354204e4f204f4e4521585858	17931e3d9359d2744fbb53a413d072a1		
	5452555354204e4f204f4e4521585859	e92cc179380ca28820f5ee5cad6fed58	87	67.9688
7	4a454e53454e53454154484f41434c41	2e15be6de661170077301dc292755e71		
	4a454e53454e53454154484f41434c40	2258c98f487375d4e4a2e2bdd8d20784	70	54.6875
8	41636c612c4a616b6520526f756b6500	9b07ad9a41f460cd628b5a7a48cad89e		
	41636c612c4a616b6520526f756b6501	1db1d4a8c22758de58f01ecbc27d8bff	62	48.4375
9	41434c414a494e44524f414c57594e4e	231a78b67cd26d76d0b26625b02a29c5		
	41434c414a494e44524f414c57594e4d	9eab4d1df3ba616958c8170cad321183	60	46.8750
10	4d59204d45535341474520495320494e	d02a02cd32c6a259961ba9ade407b1fb		
	4d59204d45535341474520495320494d	e2ed29cf8c01f5e07e579c10a459eab4	67	52.3438
		Average Avalanche Effect:		53.6719

The comparison summary on the experiment, using the dataset from the reference paper (the same set of plaintexts and key), is provided in Table 8. The table compares average hamming distances (AvgHD) and average avalanche effects

(AvgAE) across plaintexts in the dataset. The table presents the results from three categories: the results of the reference paper [33] (for AES, LAES algorithms), results of this paper (for AES, MLAES algorithms), and the additional results by the independent cryptographic tool (for the AES algorithm). The results computed with cryptool2 [37] serve as a reference, as some of the calculated ciphertexts between the reference paper and this paper, for the AES algorithm, have different results. Cryptool2 confirms that the result for the AES algorithm computed in this paper is the same as calculated with Cryptool2, for the given dataset.

Table 8. Average hamming distance and average avalanche effect on plaintext bitflip

Reference Paper	AvgHD[1)]	AvgAE[2)]
AES	62.4000	48.7500
LAES	67.9000	53.0469
This Paper		
AES	64.5000	50.3906
MLAES	68.7000	**53.6719**
Cryptool2		
AES	64.5000	50.3906

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

Figure 4. MLAES Inverse S-box table

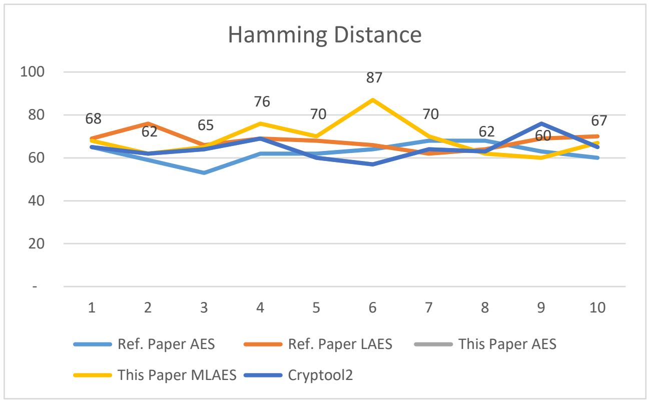

Figure 5. A comparison of hamming distances, AES, and modified AES

The findings summarize that the average avalanche effect in MLAES is 4.9219% and 0.6250% better than calculated AES and LAES in the reference paper, respectively.

MLAES's average avalanche effect is 3.2813% better than the calculated AES in this paper and concurs with the calculated AES in cryptool2, respectively.

Across the dataset, Figure 4 compares hamming distances for AES and LAES in the reference paper with AES and MLAES in this paper, including AES from cryptool2. Figure 5 illustrates the comparison for the avalanche effect, with highlighted values showing for MLAES.

3.2 Decryption in MLAES

Hamming distance and avalanche effect are all derived from the encryption process. We have the key with the first plaintext and, through encryption, produce the first ciphertext. Likewise, we have the key with the second plaintext producing the second ciphertext. Then, we calculate the bit differences between the first ciphertext and the second ciphertext by observing the number of changed bits from the first ciphertext to the second ciphertext.

As AES is a symmetric algorithm, we use the same key to reverse this process to decrypt the ciphertexts and get the plaintexts. This means getting the first plaintext from the first ciphertext and the second plaintext from the second ciphertext.

The sequence of executed functions is also reversed. In encryption, the flow is SubBytes, Shiftrows, and finally, MixColumns run for several rounds. As in AES, MLAES does not use MixColumns in the last round.

In decryption, the sequence is reversed. The process starts from the last round, where no MixColumns function, and proceeds to the next rounds with MixColumns, Shiftrows, and SubBytes in sequence.

Note that, in MLAES, there is no modification to the Shiftrows function.

3.3 Modification to inverse S-box table

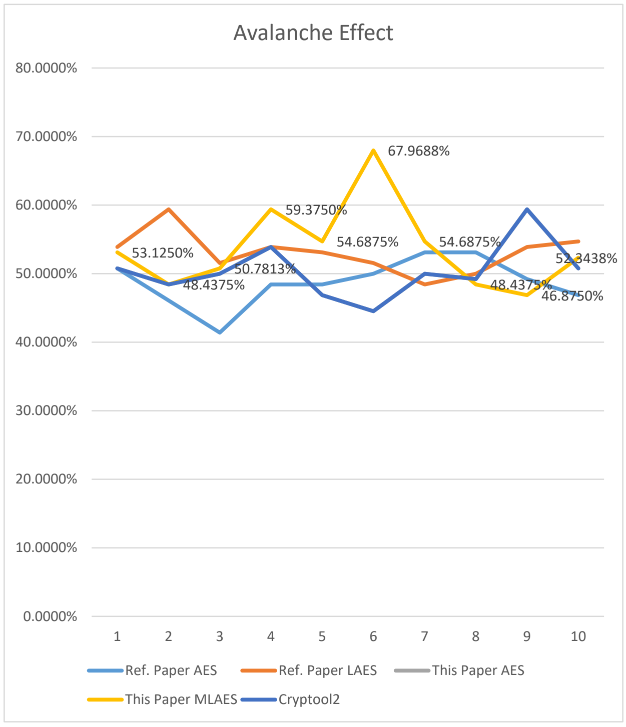

Figure 6. A comparison of avalanche effects, AES, and modified AES

As the encryption process transformed the AES S-box table to the MLAES S-box table, the decryption process for MLAES must use the inverse MLAES S-box table. Figure 6 illustrates the MLAES inverse S-box table.

The inverse S-box table is created from the S-box table. The 16 by 16 byte of S-box table is arranged in rows and columns. The content of each row and column becomes the index of the inverse S-box table. The one-byte content of each cell in the inverse S-box table is the row and column in the S-box table in which the first nibble is the row; the second nibble is the column, hence becoming a byte.

3.4 Modification of the inverse MixColumns constant

The inversed MixColumns Constant for MLAES is derived from the MLAES MixColum Constant. This calculation is not straightforward as we need to find an inverse matrix in the Galois field within the finite set of elements, in this case 2^8, equal to 256 elements. Table 9 illustrates the AES and MLAES inverse MixColumns Constants. AES inverse MixColumns constant has been provided in standard AES, while MLAES MixColums constants are calculated using the Galois library [38] in Python.

3.5 Modification to the number of rounds

The number of rounds is eight, as applied in the encryption process.

Table 9. Inverse MixColumns constants for AES and MLAES

AES 128-bits				**MLAES 128-bits**			
E	B	D	9	1	3	1	2
9	E	B	D	2	1	3	1
D	9	E	B	1	2	1	3
B	D	9	E	3	1	2	1

Table 10 illustrates the dataset with encryption and decryption results. The encryption process encrypts the plaintexts and keys, resulting in ciphertexts. Using the same keys as in the encryption process, the decryption process decrypts the ciphertexts to restore the original plaintexts.

The restored plaintexts are then compared to the original plaintexts, byte-by-byte, to ensure all bytes are matched.

Table 10. MLAES, validating encryption and decryption results

Plaintext (P, hex)	**Ciphertext (C, hex)**	**Decrypted Ciphertext (DC, hex)**	**P=DC?**
123456789ABCDEF0123456789ABCDEF0	B7009D3694C0979B4E6F33E519DE8E3F	123456789ABCDEF0123456789ABCDEF0	Match
123456789ABCDEF0123456789ABCDEF1	F1CA6C63F27888F12855477285A49508	123456789ABCDEF0123456789ABCDEF1	Match
112233445566778899AABBCCDDEEFF00	C3EDFB0C710EDA7A67A4FEF4ADE0797 B	112233445566778899AABBCCDDEEFF00	Match
112233445566778899AABBCCDDEEFF01	26EA0CE0352C591B6AC2ADEE0635FD96	112233445566778899AABBCCDDEEFF01	Match
1EE823570972BB0F30D05938C132D612	1A8D2BBA3A26CB54E366E9BB4713319D	1EE823570972BB0F30D05938C132D612	Match
1EE823570972BB0F30D05938C132D613	57FF371ACC4F1CF62ABC5328EFF62176	1EE823570972BB0F30D05938C132D613	Match
E1172357097244F030D059373ECD2944	5C059DFA55FC5C325E363A25CA924589	E1172357097244F030D059373ECD2944	Match
E1172357097244F030D059373ECD2945	87544927888CD008ABCBB9EA2038AC7B	E1172357097244F030D059373ECD2945	Match
00112233445566778899AABBCCDDEEFF	C983E69895638B2655DC0B3058EF33D0	00112233445566778899AABBCCDDEEFF	Match
00112233445566778899AABBCCDDEEF E	ABEF8127C23DDF43D8B63CE8BB20D7C1	00112233445566778899AABBCCDDEEF E	Match
5452555354204E4F204F4E4521585858	17931E3D9359D2744FBB53A413D072A1	5452555354204E4F204F4E4521585858	Match
5452555354204E4F204F4E4521585859	E92CC179380CA28820F5EE5CAD6FED58	5452555354204E4F204F4E4521585859	Match
4A454E53454E53454154484F41434C41	2E15BE6DE661170077301DC292755E71	4A454E53454E53454154484F41434C41	Match
4A454E53454E53454154484F41434C40	2258C98F487375D4E4A2E2BDD8D20784	4A454E53454E53454154484F41434C40	Match
41636C612C4A616B6520526F756B6500	9B07AD9A41F460CD628B5A7A48CAD89E	41636C612C4A616B6520526F756B6500	Match
41636C612C4A616B6520526F756B6501	1DB1D4A8C22758DE58F01ECBC27D8BFF	41636C612C4A616B6520526F756B6501	Match
41434C414A494E44524F414C57594E4E	231A78B67CD26D76D0B26625B02A29C5	41434C414A494E44524F414C57594E4E	Match
41434C414A494E44524F414C57594E4D	9EAB4D1DF3BA616958C8170CAD321183	41434C414A494E44524F414C57594E4D	Match
4D59204D45535341474520495320494E	D02A02CD32C6A259961BA9ADE407B1FB	4D59204D45535341474520495320494E	Match
4D59204D45535341474520495320494D	E2ED29CF8C01F5E07E579C10A459EAB4	4D59204D45535341474520495320494D	Match

4. CONCLUSION AND FUTURE WORK

4.1 Conclusion

This research paper proposes a lightweight, modified Advanced Encryption algorithm, MLAES, based on the AES algorithm.

The increase in security is measured with the Hamming distance and the avalanche effect. The findings conclude that the average avalanche effect in MLAES (53.6719%), tested with the same dataset, is compared to the state-of-the-art result (53.0469%), showing MLAES is 0.6250% better.

This has answered RQ1: Some possible components of AES-128 to modify are the S-box table within SubBytes and MixColumns, including reducing the number of rounds. RQ2: The metrics to measure for better security are Hamming distance and Avalanche effect. RQ3: Evaluated on the same dataset, MLAES shows a 0.6250% improvement in the measured average avalanche effect (53.6719%), meaning better security than state-of-the-art (53.0469%).

The AES algorithm is the foundation for the MLAES algorithm (128 bits). Its primary design goal is quick, safe, and effective for a lightweight device. Except for Key expansion and ShiftRows, the changes are restricted to the S-box table in SubBytes, MixColumns, and round components.

One practical implication of this research is using MLAES for low-power embedded systems, such as edge computing and Internet of Things devices.

4.2 Limitations

The modification of the AES algorithm is based on AES with a 128-bit key, AES-128. As the standard, AES supports different key variations, AES-192 and AES-256. The algorithm modifications of AES with a 192-bit key and AES with a 256-bit key are not within the scope of this study.

4.3 Future work

Future theoretical research on algorithm enhancements may include exploring transformations in an S-box table within SubBytes with different S-box transformation functions. Modifying the ShiftRows and MixColumns algorithms is another possible area. All algorithm modifications shall consider lightweight computing power and limited resources in embedded systems.

Future practical implementation research may include implementing the MLAES algorithm on edge computing or IoT devices. MLAES can strengthen the security of data transfers between SCADA and MES or SCADA and PLC systems in Industry 4.0, for example, and it can also include connectivity to other external systems such as IoT devices or even the Cloud.

REFERENCES

[1] Buchanan, S.S. (2022). Cyber-attacks to industrial control systems since stuxnet: A systematic review. Capitol Technology University.

[2] Alrumaih, T.N., Alenazi, M.J., AlSowaygh, N.A., Humayed, A.A., Alablani, I.A. (2023). Cyber resilience in industrial networks: A state of the art, challenges, and future directions. Journal of King Saud University-Computer and Information Sciences, 35(9): 101781. https://doi.org/10.1016/j.jksuci.2023.101781

[3] Canonico, R., Sperlì, G. (2023). Industrial cyber-physical systems protection: A methodological review. Computers & Security, 135: 103531. https://doi.org/10.1016/j.cose.2023.103531

[4] Yoo, H., Ahmed, I. (2019). Control logic injection attacks on industrial control systems. In ICT Systems Security and Privacy Protection: 34th IFIP TC 11 International Conference, SEC 2019, Lisbon, Portugal, pp. 33-48. https://doi.org/10.1007/978-3-030-22312-0_3

[5] Stouffer, K., Pease, M., Tang, C., Zimmerman, T., Pillitteri, V., Lightman, S., Hahn, A., Saravia, S., Sherule, A., Thompson, M. (2023). NIST Special Publication NIST SP 800-82r3 Guide to Operational Technology (OT) Security. NIST: Gaithersburg, MD, USA. https://doi.org/10.6028/NIST.SP.800-82r3

[6] Jayalaxmi, P., Saha, R., Kumar, G., Kumar, N., Kim, T.H. (2021). A taxonomy of security issues in Industrial Internet-of-Things: Scoping review for existing solutions, future implications, and research challenges. IEEE Access, 9: 25344-25359. https://doi.org/10.1109/ACCESS.2021.3057766

[7] Patnala, T.R., Jayanthi, D., Majji, S., Valleti, M., Kothapalli, S., Karanam, S.R. (2020). A modernistic way for KEY generation for highly secure data transfer in ASIC design flow. In 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, pp. 892-897. https://doi.org/10.1109/ICACCS48705.2020.9074200

[8] Mishra, R., Bhanodiya, P. (2015). A review on steganography and cryptography. In 2015 International Conference on Advances in Computer Engineering and Applications, Ghaziabad, India, pp. 119-122. https://doi.org/10.1109/ICACEA.2015.7164679

[9] Rahmani, M.K.I. (2021). Cryptographic Algorithms and Protocols. A Step Towards Society 5.0: Research, Innovations, and Developments in Cloud-Based Computing Technologies, pp. 11-20. CRC Press. https://doi.org/10.1201/9781003138037-2

[10] Shannon, C.E. (1949). Communication theory of secrecy systems. The Bell System Technical Journal, 28(4): 656-715. https://doi.org/10.1002/J.1538-7305.1949.TB00928.X

[11] Mammeri, Z. (2024). Cryptography: Algorithms, Protocols, and Standards for Computer Security. Wiley. https://doi.org/10.1002/9781394207510

[12] Dworkin, M.J. (2023). Advanced Encryption Standard (AES). https://doi.org/10.6028/NIST.FIPS.197-upd1

[13] John, S. (2023). Advanced encryption standard modified for cryptographic applications. International Research Journal of Modernization in Engineering Technology and Science, 5(8): 2188-2197. https://doi.org/10.56726/IRJMETS44294

[14] Cecchinato, N., Toma, A., Drioli, C., Oliva, G., Sechi, G., Foresti, G.L. (2022). A secure real-time multimedia streaming through robust and lightweight AES encryption in UAV networks for operational scenarios in military domain. Procedia Computer Science, 205: 50-57. https://doi.org/10.1016/J.PROCS.2022.09.006

[15] Lavanya, R., Karpagam, M. (2020). Enhancing the security of AES through small scale confusion operations for data communication. Microprocessors and Microsystems, 75: 103041. https://doi.org/10.1016/J.MICPRO.2020.103041

[16] Kumar, K., Ramkumar, K.R., Kaur, A. (2022). A lightweight AES algorithm implementation for encrypting voice messages using field programmable gate arrays. Journal of King Saud University-Computer and Information Sciences, 34(6): 3878-3885. https://doi.org/10.1016/J.JKSUCI.2020.08.005

[17] Malal, A., Tezcan, C. (2024). FPGA-friendly compact and efficient AES-like 8 × 8 S-box. Microprocessors and Microsystems, 105: 105007. https://doi.org/10.1016/J.MICPRO.2024.105007

[18] Rashidi, B. (2021). Compact and efficient structure of 8-bit S-box for lightweight cryptography. Integration, 76: 172-182. https://doi.org/10.1016/J.VLSI.2020.10.009

[19] Illy, A., Yélémou, T., Tall, H., Dandjinou, T.M. (2022). An improvement of the AES protocol to optimize energy consumption in IoT. In 2022 IEEE Multi-conference on Natural and Engineering Sciences for Sahel's Sustainable Development (MNE3SD), Ouagadougou, Burkina Faso, pp. 1-5. https://doi.org/10.1109/MNE3SD53781.2022.9723173

[20] Gamido, H.V. (2020). Implementation of a bit permutation-based advanced encryption standard for securing text and image files. Indonesian Journal of Electrical Engineering and Computer Science, 19(3): 1596-1601. https://doi.org/10.11591/ijeecs.v19.i3.pp1596-1601

[21] Lakshmi, M.B., Murthy, P.K. (2024). Modified advanced encryption standard for more secure communication. Journal of Emerging Technologies and Innovative Research, 7(11): 226-233.

[22] Devi, K.Y., Chakravarthy, V.V.S.S.S. (2020). Design of light weight encryption algorithm for data privacy in light weight devices using 128 bit key. International Journal of Research, 7(5): 221-229.

[23] Mohammad, H.M., Abdullah, A.A. (2022).

Enhancement process of AES: A lightweight cryptography algorithm-AES for constrained devices. TELKOMNIKA (Telecommunication Computing Electronics and Control), 20(3): 551-560. https://doi.org/10.12928/telkomnika.v20i3.23297

[24] Fadhil, M.S., Farhan, A.K., Fadhil, M.N. (2021). A lightweight AES algorithm implementation for secure IoT environment. Iraqi Journal of Science, 62(8): 2759-2770. https://doi.org/10.24996/ijs.2021.62.8.29

[25] Qabajeh, L., Tahboub, R., AbuJoodeh, M. (2023). A new lightweight AES for IoT. 2023 International Conference on Information Technology (ICIT), Amman, Jordan, pp. 397-404. https://doi.org/10.1109/ICIT58056.2023.10226005

[26] Fadhil, M.S., Farhan, A.K., Fadhil, M.N., Al-Saidi, N.M. (2020). A new lightweight AES using a combination of chaotic systems. In 2020 1st. Information Technology to Enhance E-Learning and Other Application (IT-ELA, Baghdad, Iraq, pp. 82-88. https://doi.org/10.1109/IT-ELA50150.2020.9253099

[27] Vaz, Y.S., Mattos, J.C., Soares, R.I. (2023). Improving an ultra lightweight AES for IoT applications. In 2023 IEEE 9th World Forum on Internet of Things (WF-IoT), Aveiro, Portugal, pp. 1-6. https://doi.org/10.1109/WF-IoT58464.2023.10539597

[28] Vimalkumar, J., Babu, H.R., Bhaskar, M. (2023). FPGA implementation of modified lightweight 128-Bit AES algorithm for IoT applications. In 2023 IEEE International Symposium on Smart Electronic Systems (iSES), Ahmedabad, India, pp. 306-309. https://doi.org/10.1109/iSES58672.2023.00069

[29] Purohit, S., Deshpande, V., Ingale, V. (2023). FPGA implementation of the AES algorithm with lightweight LFSR-based approach and optimized key expansion. In 2023 IEEE International Conference on Public Key Infrastructure and its Applications (PKIA), Bangalore, India, pp. 1-7. https://doi.org/10.1109/PKIA58446.2023.10262697

[30] Hammod, D.N. (2022). Modified lightweight AES based on replacement table and chaotic system. In 2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), Ankara, Turkey, pp. 1-5. https://doi.org/10.1109/HORA55278.2022.9799804

[31] Abdalrazzaq, A.S., Alabady, S.A. (2022). Design and implementation of a lightweight and fast tiny advanced encryption standard algorithm. Jordan Journal of Electrical Engineering, 8(4): 340. https://doi.org/10.5455/jjee.204-1658696772

[32] Salman, R.S., Farhan, A.K., Shakir, A. (2022). Lightweight modifications in the Advanced Encryption Standard (AES) for IoT applications: A comparative survey. In 2022 International Conference on Computer Science and Software Engineering (CSASE), Duhok, Iraq, pp. 325-330. https://doi.org/10.1109/CSASE51777.2022.9759828

[33] Acla, H.B., Gerardo, B.D. (2019). Security analysis of lightweight encryption based on advanced encryption standard for wireless sensor networks. In 2019 IEEE 6th International Conference on Engineering Technologies and Applied Sciences (ICETAS), Kuala Lumpur, Malaysia, pp. 1-6. https://doi.org/10.1109/ICETAS48360.2019.9117387

[34] Wu, M. Understanding AES & Rijndael. https://github.com/matt-wu/AES/tree/master, accessed on Dec. 14, 2024.

[35] Webster, A.F., Tavares, S.E. (2000). On the design of S-boxes. In Advances in Cryptology-CRYPTO'85. Lecture Notes in Computer Science, 523-534. https://doi.org/10.1007/3-540-39799-X_41

[36] Kalaiselvi, R.C., Vennila, S.M. (2021). Security enhancement using custom-AES and its performance evaluation on avalanche effect-A new approach. INDIAN Journal of Computer Science and Engineering, 12(3): 591-597. https://doi.org/10.21817/indjcse/2021/v12i3/211203092

[37] Esslinger, B., Kopal, N., Wacker, A. CrypTool 2: An Open-Source E-Learning Project for Cryptography and Cryptanalysis. https://www.cryptool.org/en/ct2/, https://github.com/CrypToolProject/CrypTool-2, accessed on Dec. 16, 2024.

[38] Python Software Foundation, "galois PyPI," https://pypi.org/project/galois/, accessed on Dec. 22, 2024.

NOMENCLATURE

3DES	Triple DES
AES	Advanced Encryption Standard
CIA	Confidentiality, Integrity, and Availability
CPU	Central Processing Unit
DES	Data Encryption Standard
FPGA	Field Programmable Gate Array
GCC	Gnu Compiler Collection
HD	Hamming Distance
ICS	Industrial Control Systems
I/O	Input/Output
IoT	Internet of Things
IIoT	Industrial IoT
IEC	International Electrotechnical Commission
ISA	Industrial Automation and Control Systems Security
ISA99	International Society of Automation
ISO	International Standard Organization
LWC	Lightweight Cryptography
IT	Information Technology
LAES	Lightweight AES
MES	Manufacturing Execution System
MLAES	Modified LAES
NBC	Number of bits changed
NIST	National Institute of Standards and Technology
OT	Operational Technology
OS	Operating System
PLC	Programmable Logic Controller
RQ	Research Question
SCADA	Supervisory Control and Data Acquisition
SoC	System on Chip
TNB	Total number of bits
XOR	eXclusive OR