https://doi.org/10.31436/ijpcc.v11i2.578

A Lightweight Authenticated Cipher For Resource Constrained Environment

Loke Sue Voon

School of Computer Sciences, Universiti Sains Malaysia, Gelugor, Malaysia

*Corresponding author: lsuevoon@gmail.com

(Received: 30th April 2025; Accepted: 3rd July, 2025; Published on-line: 30th July, 2025)

Abstract — In this paper, a new lightweight authenticated encryption scheme that uses a lightweight block cipher is introduced. The cipher supports flexible key sizes starting from 128 bits, with the increment of 64 bits. Security evaluation shows that the cipher passed the evaluation of algebraic, differential and linear cryptanalysis which proved after 11 rounds, the attacker has no advantage over brute force attack. Furthermore, performance analysis is evaluated based on throughput (Mbps), latency (cycle) and area estimation (gate equivalence). Hardware implementations of the cipher require approximately 2526 GE with a throughput of 118Kbps running on Intel i5-8250 CPU using Python implementation.

Keywords — Authenticated encryption, lightweight cryptography, block cipher, flexible key, symmetric encryption

I. Introduction

Cryptography is essential to ensure that the information transacted is not exploited and data are communicated only to those intended parties. Symmetric encryption is one of the ways to secure data as they are being transacted across communication channels. However, conventional symmetric-key cryptography does not prove authenticity. Authentication is crucial to establish trust in the identities of connected devices on the IoT [1]. Authenticated Encryption (AE) is one form of encryption that could simultaneously provide confidentiality, authenticity and integrity [2]. Authenticated Encryption with Associated Data (AEAD) is another form of AE that accepts associated data. In many cases of application settings, the associated data included should be authenticated but left unencrypted. Another problem that conventional cryptosystems are facing is the possible emergence of quantum computers. Quantum computations will likely be able to significantly reduce calculation time from millions of years to only a few hours. With that, there is a need for cryptographic algorithms that are secure in a post-quantum world. Symmetric-key algorithms are expected to be secure in the presence of a quantum adversary by enlarging the key or block size [3].

Resource-constrained environment refers to power limitation, memory limitation and physical limitations [4]. Conventional encryption systems are not suitable to be implemented because they are created for desktop environments or servers that require high memory and computational power. In lightweight applications, the hardware implementation and power requirements also vary according to different situations.

For example, in a closed-circuit television (CCTV) system, images must be encrypted at a high throughput rate, and the system is typically powered by an alternating current (AC) supply. AE is an effective solution in this scenario because it is significantly lighter than public-key cryptography and more efficient, as it performs data authentication and encryption simultaneously. AE techniques require fewer computational resources and avoid potential security weaknesses owing to the implementation flaws. Therefore, a new lightweight authenticated cipher that can accept flexible key sizes is introduced in the paper.

This paper proposes two cryptographic contributions which are the AE Scheme and a new block cipher design. The proposed AE scheme uses an underlying block cipher that can accept flexible key sizes so that the algorithm can be resistant to quantum computing. Preliminary cryptanalysis attacks on the cipher showed that after 11 rounds, the cipher provides sufficient security margin against linear, differential and algebraic cryptanalysis. NIST statistical analysis was used to evaluate the randomness of ciphertext generated by the cipher. This paper is organized as follows; Section II presents the specifications of the AE scheme. Section III shows the security analysis. Section IV details the performance analysis and Section V shows the data availability. Lastly, Section VI concludes the paper.

II. Specifications of AE Scheme

This AE scheme consists of an encryption and decryption algorithm that is parameterized by a block cipher that supports flexible key sizes starting from 128 bits, in increments of 64 bits, k_n and block size 128 bits, x = 128. The cipher is based on substitution and permutation network and the key size is represented by Equation 1:

$$k_n, n = 128 + 64i \text{ where } i = 0, 1, 2, 3\ldots \quad (1)$$

The input parameters for the encryption algorithm are key k_n , associated data A, and a plaintext P . The output of the encryption algorithm is a ciphertext, where the first n bits is the tag, T. The input parameters for the decryption algorithm are key, k_n, associated data, A and ciphertext, C. A successful decryption will output message, M while a failed verification will output the symbol ⊥. Fig. 1 illustrates the entire process of the AE scheme, from initialization and processing of the associated data to the encryption of plaintext, resulting in the ciphertext and authentication tag.

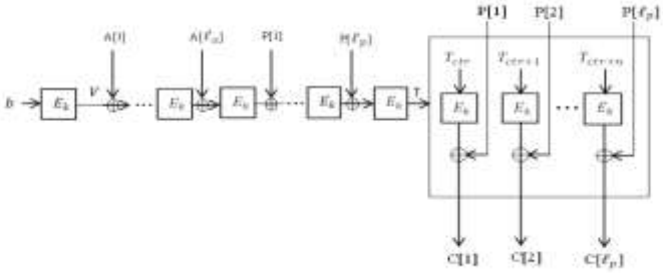


Fig. 1 Overall structure of AE Scheme

A. *Notations*

Let $\{0, 1\}^*$ be the set of all bit strings, $\{0, 1\}^N$ be the set of n-bit strings. The bitwise XOR between x and y is written as x ⊕ y. The concatenation of two bits, x and y is written as x‖y. Bitwise left cyclic shift by y bits is written as x ≪ y. Bitwise right shift by y bits is written as x ≫ y. The length of X is written as |X|.

B. *Initialization*

The algorithm starts with the initialization of a block, b based on the input parameters such as key size |K|, associated data length |A| and plaintext length |P |.

$b = x_{127}x_{126}x_{125}||0^{125}$ is defined in Equation 2.

$$x_{127} = \begin{cases} 0 & \text{if } |K| > 256 = 0 \\ 1 & \text{otherwise} \end{cases}$$
$$x_{126} = \begin{cases} 0 & \text{if } |A| = 0 \\ 1 & \text{otherwise} \end{cases}$$
$$x_{125} = \begin{cases} 0 & \text{if } |M| = 0 \\ 1 & \text{otherwise} \end{cases} \quad (2)$$

The initial block is encrypted to produce a vector, V. If the associated data and plaintext are not passed in, this encryption will produce a tag, T. The algorithm of the block initialization is defined in Algorithm 1.

Algorithm 1

```
Input $K \in K, A \in \{0, 1\}^*, P \in \{0, 1\}^*$
Output $V \in \{0, 1\}^{128}$
$b_1 \leftarrow |K| > 256 ? 1 : 0$
$b_2 \leftarrow |A| > 0 ? 1 : 0$
$b_3 \leftarrow |A| > 0 ? 1 : 0$
$b \leftarrow b_1 || b_2 || b_3 || 0^{125}$
$V \leftarrow E_K(b)$
```

C. *Associated Data and Message Preprocessing*

The associated data needs to be processed before passing through the block cipher call. If the associated data is not empty, it is split into 128-bit subblocks, if the total length of the associated data is not multiple of 128 bits, the last subblock of A will be padded with "1" then followed by "0" until the required length. If the length is in multiple of 128 bits, A will be padded with another block of "0". Then, each of the subblocks are passed through the encryption function, E_k to produce a vector, V. Each subblock is subsequently XOR-ed to update V. Refer to Equation 3.

$$V \leftarrow E_k(V \oplus A[i]) \text{ where } i \in \{1, 2, 3..\ell_A - 1\} \quad (3)$$

The same method is applied to plaintext as well. If the plaintext is not empty, it is split into 128-bit subblocks, if the total length of the plaintext is not multiple of 128 bits, the last subblock of P will be padded with "0" until the required length. Each of the plaintext subblocks is passed through the encryption function, E_k to produce a vector,V . Each subblock is subsequently XOR-ed to update the vector. Refer to Equation 4.

$$V \leftarrow E_k(V \oplus P\,[i]) \text{ where } i \in \{1, 2, 3..\ell_P - 1\} \quad (4)$$

After initialization and preprocessing of associated data and plaintext, the plaintext subblocks are encrypted to produce ciphertext C. The tag is obtained from V. Refer to Equations 5 and 6.

$$T \leftarrow V \quad (5)$$

$$C[i] \leftarrow E_k(V, T_{ctr+i)} \oplus P\,[i] \text{ where } i \in \{1, 2, 3..\ell_P - 1\} \quad (6)$$

D. *Block Cipher Specifications*

The block cipher is a lightweight 128-bit block cipher based on the Substitution-Permutation network. It is an 11-round iterative block cipher. The round function of the block cipher consists of an S-Box layer, a permutation layer, XOR and a left cyclic shift permutation. The architecture of the block cipher and its round function is illustrated in Fig. 2. The message encryption algorithm is defined in Algorithm 2.

Algorithm 2

```
Input $K \in K, P \in \{0, 1\}^{128}$
for i = 1, 2 . . . 13 do
    $K_i \oplus P$
    Perform substitution by byte
    Perform bitwise permutation
    for j = 0, 1 . . . 7 do
        $state_i$ = perform XOR on permutated states
    end for
    Left cyclic shift on $state_i$ and append all states
end for
```

E. S-Box Layer

The S-box used in the cipher has been chosen to ensure the robustness of the design. Since the number of active S-boxes directly impacts the security of the cipher, the S-box design must meet important criteria such as bijectivity, nonlinearity and independence of output bits [5]. The S-Box design is adopted from Serpent cipher [6]. This cipher implements the 4x4 Serpent S3 S-Box which is proven to have the linear and differential probability characteristics of $\frac{1}{4}$ that are within a reasonable bound where linear or differential attacks will be infeasible. The Serpent S3 S-Box is also classified as a Golden S-Box which means it has ideal cryptographic properties such as optimal differential bounds, attractive number of characteristics at differential bound and number of approximations at linear bound [7]. Table I shows S-Box Mapping.

TABLE I
S-BOX MAPPING

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S(x)	0	F	B	8	C	9	6	3	D	1	2	4	A	7	5	E

F. Permutation Layer

The diffusion mechanism employed for this block cipher is a bit-wise shuffle, π. Fig. 3 shows the P-Box, which defines the bit-wise permutation used to achieve diffusion across the cipher state. This block shuffle is used because it has been proven to produce a highly diffusive shuffle based on de Bruijn Graph [8]. The shuffling has been designed to maximize the number of active S-Boxes in each round so that the exploitable characteristics are reduced. The D_{max} value shows that a full diffusion can be achieved at 8 rounds. Table II shows the block shuffle, π, inverse block shuffle, π^{-1} and number of active S-Boxes.

TABLE II
BLOCK SHUFFLE

π	π^{-1}	D_{max}	Active S-boxes(DC&LC)
1,2,9,4,15,6,5,8,13,10,7,14,11,12,3,0	15,0,1,14,3,6,5,10,7,2,9,12,13,8,11,4	8	39

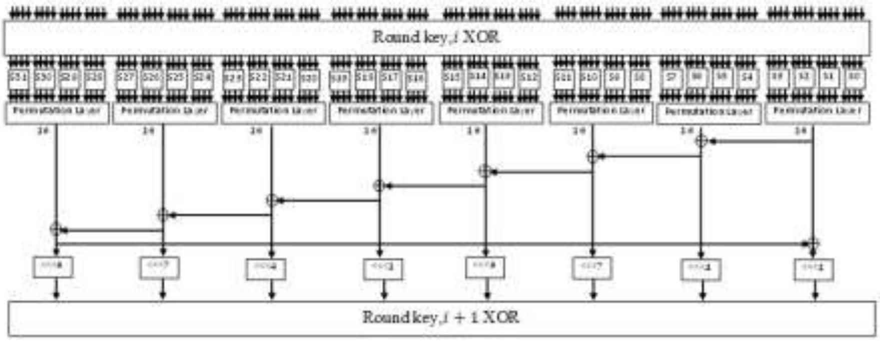


Fig. 3 Single round function in block cipher

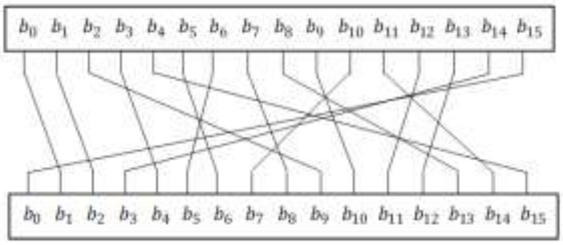

b_0 b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_8 b_9 b_{10} b_{11} b_{12} b_{13} b_{14} b_{15}
b_0 b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_8 b_9 b_{10} b_{11} b_{12} b_{13} b_{14} b_{15}

Fig. 2 P-Box

G. Key Schedule

The key scheduling algorithm implemented in this block cipher is inspired by PRESENT cipher key scheduling. The algorithm has been modified to accommodate the input of flexible key sizes. The key schedule flexibility ensures that the cipher is feasible to be implemented on current devices while catering to the emergence of future quantum computers where larger key sizes are required. Because the key scheduling algorithm used in the cipher is the same regardless of variable length key sizes passed in, the circuit area used by the algorithm remained the same. The only difference is the size of the register that is used to store the master key, which will only increase by a factor of o(n).

First, an n–bit register is initialised with n–bit master key, where $K_{Master} = k_{n-1}k_{n-2}\ldots k_1k_0$. In each round of the key expansion function, the key is XOR-ed to produce round key, K_i (for rounds 1 until 13). K_{Master} is being left shifted by 13 bits,$K_{Master} \ll 13$. The 128–bit LSB of the current key register is taken as the round key, K_i, where $K_i = k_{127}k_{126}...k_1k_0$. Then perform substitution on least significant bits (LSBs) with S-Box defined in Table II, $K_3K_2K_1K_0 = S[K_3K_2K_1K_0]$. The number of LSBs substituted depends on the key size length used for the block cipher. For each multiple of 64–bit keys, a set of substitutions will be performed on the four bits of the key starting from the LSB based on Fig. 4. For each round, the round counter RC^i (5 bits) is XOR-ed with bits 127 to 123 of the key register,

$K[K_{127}K_{126}K_{125}K_{124}K_{123}] = [K_{127}K_{126}K_{125}K_{124}K_{123}] \oplus RC^i$.

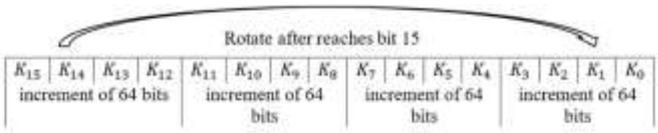


Fig. 4 Rotation Pattern

III. Security Analysis

Preliminary testing and analysis of performance and security are performed on this authenticated block cipher. Differential and linear cryptanalysis are the two most significant ways to analyze security in a block cipher. The security can be determined based on the analysis of active S-Boxes for differential and linear characteristics, and the weight of differential and linear trails. The optimal trails can be obtained by coding the SMT model.

A. *Differential Cryptanalysis*

To conduct a differential cryptanalysis, an open-source tool, CryptoSMT (https://github.com/kste/cryptosmt) has been utilized to find the best differential trails and compute the probability of a differential. The differential behaviour of the block cipher is represented as an SMT model. The script was written based on CVC language and Python was used to create the SMT model. Firstly, the differential behaviour of one round of the block cipher is modelled. The non-linear layer is represented by the S-Box while the linear layer is represented by bit-wise permutation, block XOR and rotation. The equations of linear and non-linear layers are described as follows:

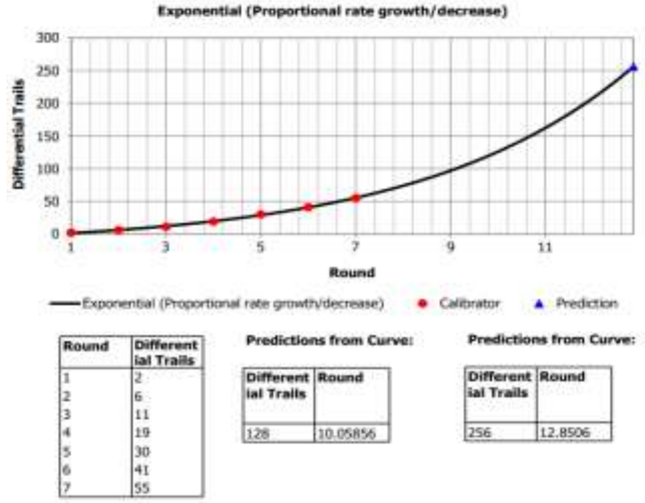
- Assume n number of S-Boxes are activated in the differential trail, for the i–th S-Box, denotes Δ_i^{in} and Δ_i^{out}as input and output differences respectively.
- For block XOR operation, denote Δ_0^{in} and Δ_1^{out} as input and output differences respectively, corresponding equation is $\Delta_0^{in} \oplus \Delta_o^{out} = \Delta^{out}$
- For bit-wise permutation, use P-Box specifications in previous section.

Table III shows the result of differential trails with optimal probability obtained from the automated model. Table IV shows the optimal differential trails and minimum active S-Boxes up to round 7 of the cipher. To compute the optimal differential trail, the corresponding value of difference propagation through the S-Box is divided by 16. Total probability for the entire differential trail is calculated by summing up all p_d as shown in Equation 7.

$$\text{Total}_{pd} = \sum_{i=0\ j=0}^{i=r\ j=31} probability\_i\_j$$

Where i = rounds and j = total S-boxes (7)

Based on the results in Table IV, an extrapolation is done to obtain the number of rounds that produce an optimal differential trail of 2^{-128}. The extrapolation was performed because the computational complexity of search methods such as SAT/SMT and Matsui branch-and-bound algorithm scales exponentially with increasing number of rounds, and is generally slower for large block sizes [9]. Due to limited computational resources, accurate values for the differential trails can only be obtained up to round 7. Subsequently, a regression method is used to estimate the differential trails up to round 11. Fig. 5 shows the regression model of the differential trail. The results indicate that the computational cost of the cipher increases exponentially with the number of rounds. The prediction model also shows a 10-round differential trail with total optimal probability bias, p_d of 2^{-12}.



Round	Differential Trails
1	2
2	6
3	11
4	19
5	30
6	41
7	55

Predictions from Curve:

Differential Trails	Round
128	10.05856

Predictions from Curve:

Differential Trails	Round
256	12.8506

Fig. 5 Regression Model for Differential Trail

In distinguishing attack, the attacker attempts to differentiate the output of a block cipher from that of a random function. If a distinguishing attack is successful, an adversary can deduce secret key information and/or other internal values. To perform a distinguishing attack with a differential trail, the attacker first must encrypt a sufficient number of plaintext pairs to observe at least "one right pair" with respect to the differential trail. The complexity of the differential attack can be computed by calculating the required number of chosen plaintext pairs, N_D using the formula in Equation 8.

$$N_D = \frac{c}{Pd} \text{ where } c=1,\ P_d = 2^{-128} \qquad (8)$$

Hence, the required number of chosen plaintext pairs will be $\frac{1}{2^{-128}} = 2^{128}$. This is equivalent to the number of messages available in the message space. Therefore, at 10 rounds, this is equivalent to a full codebook attack (requiring all 2^{128} possible plaintexts). In addition, the time complexity involved is 2^{128} encryptions, which is equivalent to a brute force attack. Thus, the adversary has no advantage over a brute force search when using a differential distinguisher after 10 rounds. This implies that the cipher is conditionally secure after 10 rounds.

TABLE III
NUMBERS OF OPTIMAL PROBABILITY AND TRAILS UP TO ROUND 7

ROUND	INPUT DIFFERENCE	OUTPUT DIFFERENCE	WEIGHT
1-ROUND DIFFERENTIAL TRAIL WITH PROBABILITY 2^{-2}			
0	0x00000000000000000000000000F00000	0x01600160016001600160016001600160	-2
1	0xC002B000160002C0C002B000160002C0	-	
2-ROUND DIFFERENTIAL TRAIL WITH PROBABILITY 2^{-6}			
0	0x70007000000000000000000000000000	0x00001800000000000000000000000000	-4
1	0x0000000C000000000000000000000000	0x00100010000000000000000000000010	-2
2	0x20000800000000000000000000000020	-	
3-ROUND DIFFERENTIAL TRAIL WITH PROBABILITY 2^{-11}			
0	0x00000000000000400000004000000000	0x00000000000000000060006000000000	-4
1	0x0000000000000000C000300000000000	0x00000000000000000000000800000000	-5
2	0x00000000000000000000040000000000	0x24802480248024802480248000002480	-2
3	0x00494012480249000049401200004900		
4-ROUND DIFFERENTIAL TRAIL WITH PROBABILITY 2^{-19}			
0	0x000000000070003000E000E000000000	0x00000000000000200000800000000000	-8
1	0x00000000000000400000004000000000	0x00000000000000000060006000000000	-4
	0x0000000000000000C000300000000000	0x00000000000000000000000100000000	-5
3	0x00000000000000000000008000000000	0x81208120812081208120812000008120	-2
4	0x41029040120802414102904000000241		
5-ROUND DIFFERENTIAL TRAIL WITH PROBABILITY 2^{-30}			
0	0x00E000E0000000000000300000003000	0x00008000000000000000000000080008	-8
1	0x00000040000000000000000000800010	0x00600060000000000000000000000000	-8
2	0xC0003000000000000000000000000000	0x00000001000000000000000000000000	-5
3	0x00000080000000000000000000000000	0x00600060000000000000000000000060	-3
4	0xC00030000000000000000000000000C0	0x01090108010001000100010001000009	-6
5	0x12028400100002000002800010000012		
6-ROUND DIFFERENTIAL TRAIL WITH PROBABILITY 2^{-41}			
0	0x00070007000000070000000700000000	0x00000006000000000006000600000000	-8
1	0x00000300000000000C00030000000000	0x00000000008000800080040000000000	-8
2	0x00000000080001000001000200000000	0x00000000000000480000000160000000	-12
3	0x000000000000000900000B0000000000	0x00000000000000002000200000000000	-6
4	0x00000000000000000040001000000000	0x00000000000000000000006000000000	-5
5	0x00000000000000000000300000000000	0x00080008000800080008000800000008	-2
6	0x10000400008000101000040000000010		
7-ROUND DIFFERENTIAL TRAIL WITH PROBABILITY 2^{-55}			
0	0x00070007000000070000000700000000	0x00000006000000000006000600000000	-8
1	0x00000300000000000C00030000000000	0x00000000008000800080040000000000	-8
2	0x00000000080001000001000200000000	0x00000000000000480000000160000000	-12
3	0x000000000000000900000B0000000000	0x00000000000000002000200000000000	-6
4	0x00000000000000000040001000000000	0x00000000000000000000006000000000	-5
5	0x00000000000000000000300000000000	0x00080008000800080008000800000008	-2
6	0x10000400008000101000040000010110	0x016009680DE80C888CA884A080208140	-14
7	0xC002B404DE8019105119504202080281		

TABLE IV
NUMBERS OF DIFFERENTIAL TRAILS AND ACTIVE S-BOXES UP TO ROUND 7

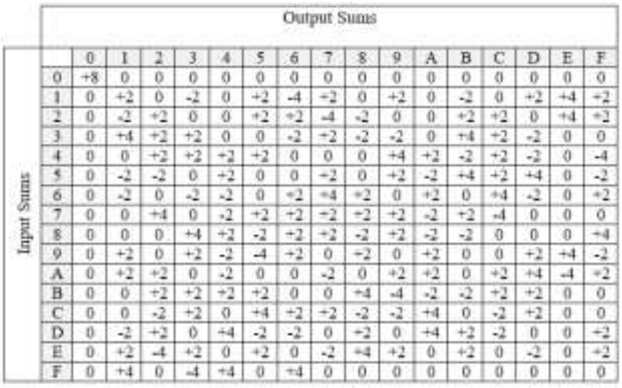
NUMBER OF ROUNDS	OPTIMAL DIFFERENTIAL TRAILS	MINIMUM NUMBER OF ACTIVE S-BOXES
1	2^{-2}	1
2	2^{-6}	3
3	2^{-11}	5
4	2^{-19}	9
5	2^{-30}	13
6	2^{-41}	16
7	2^{-55}	25

B. *Linear Cryptanalysis*

Linear cryptanalysis looks for the probability of linear expressions between plaintext bits and ciphertext bits. An ideal primitive should have a linear expression with the probability of $\frac{1}{2}$. A deviation from that probability results in linear probability bias, ϵ = $|p - \frac{1}{2}|$. To that the S-Box of this cipher is resistant to linear cryptanalysis, the Linear Approximation Table is examined. An extrapolation is also performed based on the result of experiment to obtain the predicted number of active S-Box for round 11. Fig. 6 shows the linear approximation table for the S-box and Fig. 7 shows the exponential regression of active S-Box. The maximum linear probability for the cipher is 2^{-2}. At round 11, the estimated number of active S-Box is 80. The linear approximation for the cipher can be calculated using Equation 9.

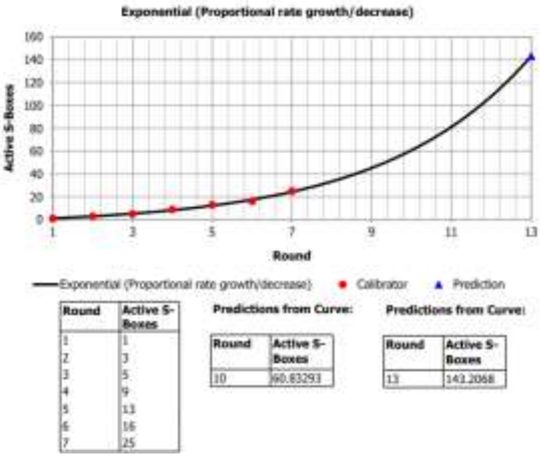
$$\varepsilon_{1,2,..n} = (2n{-}1)\prod_{i=1}^{n}\varepsilon_i \qquad (9)$$
$$= (2^{80\text{-}1} \text{ x } 2^{80\text{x-}2})$$

Therefore, the complexity of a linear attack can be calculated using the equation $N_L = \frac{1}{\varepsilon^2}$. After 11 rounds, the number of plaintexts required for a linear attack is $\frac{1}{(2^{-81})^2} = 2^{162}$, which is more than the available plaintext.



		Output Sums															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Input Sums	0	+8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	+2	0	-2	0	+2	-4	+2	0	+2	0	-2	0	+2	+4	+2
	2	0	-2	+2	0	0	+2	+2	-4	-2	0	0	+2	+2	0	+4	+2
	3	0	+4	+2	+2	0	0	-2	+2	-2	-2	0	+4	+2	-2	0	0
	4	0	0	+2	+2	+2	+2	0	0	0	+4	+2	-2	+2	-2	0	-4
	5	0	-2	-2	0	+2	0	0	+2	0	+2	-2	+4	+2	+4	0	-2
	6	0	-2	0	-2	-2	0	+2	+4	+2	0	+2	0	+4	-2	0	+2
	7	0	0	+4	0	-2	+2	+2	+2	+2	+2	-2	+2	-4	0	0	0
	8	0	0	0	+4	+2	-2	+2	+2	-2	+2	-2	-2	0	0	0	+4
	9	0	+2	0	+2	-2	-4	+2	0	+2	0	+2	0	0	+2	+4	-2
	A	0	+2	+2	0	-2	0	0	-2	0	+2	+2	0	+2	+4	-4	+2
	B	0	0	+2	+2	+2	+2	0	0	+4	-4	-2	-2	+2	+2	0	0
	C	0	0	-2	+2	0	+4	+2	+2	-2	-2	+4	0	-2	+2	0	0
	D	0	-2	+2	0	+4	-2	-2	0	+2	0	+4	+2	-2	0	0	+2
	E	0	+2	-4	+2	0	+2	0	-2	+4	+2	0	+2	0	-2	0	+2
	F	0	+4	0	-4	+4	0	+4	0	0	0	0	0	0	0	0	0

Fig. 6 Linear Approximation Table



Round	Active S-Boxes
1	1
2	3
3	5
4	9
5	13
6	16
7	25

Predictions from Curve:

Round	Active S-Boxes
10	60.83293

Predictions from Curve:

Round	Active S-Boxes
13	143.2068

Fig. 7 Regression Model for Active S-Box

C. *Algebraic Attack*

The proposed AE scheme uses a 4x4 bit S-Box that can be described with at least 21 quadratic equations over the GF(2). Hence, the number of quadratic equations that can be determined in a single round of encryption is described by a = n · 21, in b = n · 8 variables, where n is the total number of S-Boxes in the entire cipher and key scheduling algorithm. In the proposed cipher, there are a total of 32 S-Boxes in a single encryption round and 1 S-Box used in the key scheduling algorithm. In 11-rounds of encryption, there are (11 · 32) + 11 = 363 S-Boxes. The total number of quadratic equations is calculated as a = 363 · 21 = 7623 in 2904 variables. Algebraic attacks are unlikely to threaten the proposed cipher, as even for a reduced two-round version of PRESENT cipher with seven S-Boxes (28-bit block size), no solution could be found within a reasonable time [10].

D. *Statistical Analysis*

The Statistical Test Suite (STS) v2.1.2 was used to analyse binary bitstreams generated by the cipher. The P-value is the probability under the assumption that the sequence is random. Number of bits tested is 10,000,000 bits (1000 Mbit) with a sample size of 1000. The results showed that all P-values are ≥ 0.01, implying that they fall within the expected random sequence. Table V shows NIST statistical test results for the authenticated encryption scheme.

TABLE V
NIST Statistical Test Result

Test	P-Value	Passing Ratio	Result
Block Frequency	0.966	0.990	Pass
Frequency	0.069	0.990	Pass
Runs	0.109	0.996	Pass
Longest Runs of Ones	0.064	0.985	Pass
Binary Matrix Rank(BMR)	0.950	0.986	Pass
Spectral(FFT)	0.783	0.985	Pass
Non-Overlapping Templates	0.494	0.986	Pass
Overlapping Templates	0.399	0.980	Pass
Maurer's Universal	0.499	0.980	Pass
Linear Complexity	0.986	0.986	Pass
Approximate Entropy	0.783	0.989	Pass
Serial	0.683	0.990	Pass
Cumulative Sums	0.078	0.990	Pass
Random Excursion	0.507	0.996	Pass
Variant	0.513	0.998	Pass

IV. Performance Analysis

As for performance analysis, the block cipher is experimented with using Python and its performance is evaluated against lightweight block ciphers that have similar cryptographic properties such as block size and underlying structure. The key size used in the AE cipher for this comparison is 128-bit key. The hardware cost of the cipher was evaluated based on the following steps:

1. Break down cipher into two modules: Registers (key, temp data state and temp key) and round function (substitution box, XOR gate, linear feedback shift register).
2. Determine the gate count needed for one round.
3. Evaluate instruction counts of the cipher based on the number of logic gates using Standard Cells UMCL18G212T3 (CMOS 180 nm technology).

The hardware cost of S-Boxes provided by [2] is based on Serpent S-Box. Table VI shows the breakdown in area estimation for the cipher. The total hardware cost is estimated to be 2527 GE.

TABLE VI
Area Estimation

Component	**Gate Equivalence (GE)**	**Estimated Gate Equivalence (GE)**
Registers		
Key	128-bit D fit flops	128 × 4.25 = 544
Data State	NAND Gate	128 x 4.25 = 544
Key XOR	XOR Gate	128 x 2.67 = 341.76
Round Function		
Permutation	Wired Permutation	0
S-Box Layer	32 4-bit S-box	32 × 28 = 896
XOR Layer	XOR Gate	16 × 2.67 = 42.72
Block Shift	8 4-bit LFSR	8 × 4 × 4.25 = 136
Total		2526.48

Next, the throughput of the cipher is measured by experimenting with encryption of text file size 100Kb. The experiment is run on a laptop with a 1.60 GHz CPU, 12G RAM i5-8250u and Windows 10 Pro (64-bit). The throughput is calculated based on the formula in equation 10. The result obtained is 118.31Kbps using Python implementation. The latency of the cipher is measured by obtaining the clock cycle estimation. This is a fair way to measure the cipher performance is the latency as it is independent of the programming language. In this experiment, the clock cycle estimation is based on Intel Skylake-X Microarchitecture. Lastly, the results of the analysis for area and latency are compared against the current state-of-the-art lightweight authenticated block cipher, Ascon128 [11]. The overall performance analysis result is presented in Table VI. The result shows that the performance of the proposed cipher is on par with Ascon128. Ascon128 is chosen because it is known for its hardware and software efficiency. It is also the primary choice for lightweight authenticated encryption in the final selection of the CAESAR competition.

$$\text{Throughput} = \frac{plaintext,Kb}{encryption\ time,s} \qquad (10)$$

The implementation of a lightweight cryptosystem has to consider several factors such as hardware cost, power consumption, and throughput. From the performance analysis, the cipher is proved to have lightweight features that could be used in the IoT domain. Although various lightweight cryptographic algorithms can be implemented in resource-constrained devices, most of them have fixed acceptable key sizes. This cipher accepts key sizes starting from 128 bits in the increments of 64 bits. The scalability of the key scheduling algorithm enables this cipher to be implemented in various resource-constrained settings depending on the availability of memory space for different devices.

TABLE VII
Performance Analysis

Cipher	Cryptographic Properties				Performance	
	Block Size	Key Size	Structure	Rounds	Area	Latency
AEScheme	128	128	SPN	13	2527	58Kbps (Intel i5-8250u)
ASCON128	128	128	Sponge	6-8	2600	8.6 cycles per byte (AMD Ryzen 7 1700)

V. Data availability

The codes and trails used to support linear and differential cryptanalysis have been deposited in GitHub (https://github.com/suevoon/cryptanalysis-aescheme).

VI. Conclusion

In this paper, a new authenticated encryption scheme with a new block cipher design is proposed. This cipher takes into consideration the existence of future quantum computers by introducing flexible key sizes in its implementation. The cipher is evaluated against differential, linear and algebraic cryptanalysis and is found that there is no feasible distinguishing attack, linear attack and algebraic attack after round 11. Performance analysis has also shown that the block cipher design is comparable with Ascon128 in terms of area and latency. Although baseline security results have been provided, third-party cryptanalysis is still required before the cipher is fit for real-life applications. More security analysis can be done to explore the possibilities of other security vulnerabilities that may exist with this cipher design. Examples of security analysis that can be conducted are key recovery attacks and weak key attacks. The current key scheduling algorithm used in this cipher is inspired by PRESENT key scheduling and modified to accommodate flexible key sizes. Future enhancements can be done by potentially modifying the current design to become a key-alternating cipher which reduces memory usage.

Acknowledgment

The authors hereby acknowledge the review support offered by the IJPCC reviewers who took their time to study the manuscript and find it acceptable for publishing.

Conflict of Interest

The authors declare that there is no conflict of interest

REFERENCES

[1] N. Katuk, R. Vergallo, T. Sugiharto, and R. A. Krisdiawan, "A client-based user authentication scheme for the cloud of things environment," J. Comput. Sci. Technol., vol. 22, no. 2, Art. no. e08, 2022. [Online]. Available: https://doi.org/10.24215/16666038.22.e08

[2] J. Black, "Authenticated Encryption," in Encyclopedia of Cryptography and Security, H. C. A. Tilborg and S. Jajodia, Eds. Boston, MA, USA: Springer, 2011, pp. 52–61. [Online]. Available: https://doi.org/10.1007/978-1-4419-5906-5548

[3] A. Banerjee, T. Reddy.K, D. Schoinianakis, T. Hollebeek, and M. Ounsworth, "Post-Quantum Cryptography for Engineers," Internet Eng. Task Force (IETF), Internet-Draft draft-ietf-pquip-pqc-engineers-09, Work in Progress, Feb. 2025. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-pquip-pqc-engineers/09/

[4] C. I. Rene, N. Katuk, and B. Osman, "A survey of cryptographic algorithms for lightweight authentication schemes in the Internet of Things environment," in Proc. 5th Int. Conf. Comput. Informat. Eng. (IC2IE), 2022, pp. 1–6. [Online]. Available: https://doi.org/10.1109/ic2ie56416.2022.9970015

[5] A. Waheed, F. Subhan, M. M. Suud, M. Alam, and S. Ahmad, "An analytical review of current s-box design methodologies, performance evaluation criteria, and major challenges," Multimedia Tools Appl., vol. 82, no. 19, pp. 29689–29712, 2023.

[6] E. Biham, R. Anderson, and L. Knudsen, "Serpent: A new block cipher proposal," in Fast Software Encryption, vol. 1372, LNCS, Springer, 1998, pp. 222–238.

[7] W. Senpeng, H. Bin, G. Jie, S. Tairong, and Z. Kai, "Research on the security criterion of s-boxes against division property," Chin. J. Electron., vol. 30, no. 1, pp. 85–91, 2021.

[8] T. Suzaki and K. Minematsu, "Improving the generalized Feistel," in Fast Software Encryption, vol. 6147, LNCS, Springer, 2010, pp. 19–39.

[9] W.-Z. Yeoh, J. S. Teh, and J. Chen, "Automated search for block cipher differentials: A GPU-accelerated branch-and-bound algorithm," in Inf. Secur. Privacy, ACISP 2020, vol. 12248, LNCS, Springer, 2020, pp. 160–179.

[10] A. Bogdanov et al., "PRESENT: An ultra-lightweight block cipher," in Cryptographic Hardware and Embedded Systems–CHES 2007, vol. 4727, LNCS, Springer, 2007, pp. 450–466.

[11] C. Dobraunig, M. Eichlseder, F. Mendel, and M. Schl¨affer, "Ascon v1.2: Lightweight authenticated encryption and hashing," J. Cryptol., vol. 34, pp. 1–42, 2021.