International Journal of Computing and Digital Systems
ISSN (2210-142X)

LWLRD: New lightweight encryption algorithm for Low recourse devices

Shayma shakeeb mohammed[1], Yaseen Ismaiel[2]

[1, 2] *Department, of Computer Science, Mosul University, Mosul, Iraq*

Abstract: In resource-constrained situations, the demand for secure yet lightweight cryptographic algorithms is particularly high, especially for applications such as mobile payments. This work offers a unique encryption algorithm that aims to find a balance between strong security and economic performance. Our method employs a unique 5-bit S-box derived from the Cube tent chaotic function, a previously unexplored source of S-box production. We carefully chose an S-box with ideal cryptographic features, including low differential approximation probability (DAP), linear probability (LP), and high nonlinearity, which improves resistance to various attacks. To improve security, we use a dynamic P-box formed by a logistic map, with the initial value taken from the secret key. This assures that the P-box configuration is unique to each key, removing fixed patterns that attackers could exploit. Furthermore, we offer a better key generation approach based on the PRESENT algorithm, but with increased randomness and complexity to make the system more resistant to key recovery attacks. Our extensive security and performance study proves the algorithm's efficacy. The encryption procedure has a relatively short execution time of 1.3 milliseconds, and the memory footprint is small at 0.003969 MB. These findings demonstrate the algorithm's applicability for resource-constrained situations, making it a suitable choice for protecting sensitive data in mobile and embedded devices.

Keywords: ***lightweight encryption, s_box, p_box, chaotic functions***

1. INTRODUCTION

Cryptography is the foundation for the secure exchange of information, protecting individuals, organizations, and governments from the dangers posed by malevolent actors. As our reliance on digital communication and data storage grows, so will the need for strong cryptographic solutions.[1] [2]

Lightweight cryptography is a novel method that seeks to address the difficulty of Creating fast and efficient security solutions in resource-constrained environments. These options include creating new cryptographic primitives and protocols. as well as adapting and changing existing cryptosystems. When designing lightweight cryptography, three critical characteristics must be optimized: security, performance, and cost. The number of bits in a cryptographic key is commonly used to measure security. Increasing the key size can improve security. Performance is assessed by the total number of clock cycles required to complete an operation, which is proportional to throughput and energy consumption. The cost, evaluated in terms of energy or space, is dictated by the specific hardware structure utilized. However, because of the trade-offs between these three aspects, optimizing all three at the same time in a single design is extremely challenging. Designers must carefully balance competing security, performance, and cost criteria when creating successful lightweight cryptographic solutions for resource-constrained contexts.[3] [4]

Low-resource computer devices have limited hardware capabilities, including CPU, power, memory storage, and energy. These devices are commonly used in embedded systems, Internet of Things (IoT) applications, and other constrained settings where cost, size, and power consumption are critical factors. The key challenges in developing software for low-resource systems include optimizing code for efficient memory and CPU usage, reducing battery consumption, and managing limited connectivity and peripheral

E-mail: Shayma.21csp70@student.uomosul.edu.iq

resources. Ciphers developed for resource-constrained devices are lightweight and can be implemented using software or hardware to maximize resource consumption.[5]
This research offers a new substitution-permutation network (SPN) cipher optimized for low-resource devices. The goal is to find the right balance between rigorous security and lightweight performance.

2. RELATED WORK

Many lightweight algorithms have been proposed over the years. Thakor 2023 [6] proposed a new lightweight cryptographic method, AUM, specifically for resource-constrained IoT devices. It solves the issues of cost, performance, and security by introducing a 32-bit block size and key size method with a novel 5-bit S-box architecture. Using the simple algorithm to generate random subkeys. AUM intends to efficiently encrypt short communications (<2Kb) in IoT devices such as RFID tags, smart cards, sensors, and actuators.
Khompysh et.al.2023 [7] proposed ISL-LWS lightweight encryption algorithm. It processes a 64-bit input with an 80-bit key, using a 4-bit s_box. The SL-LWS algorithm outperforms other popular lightweight algorithms such as Present and Speck in terms of encryption speed and key generation time. It offers excellent data security on resource-constrained devices by providing a high level of diffusion and confusion through its linear and non-linear transformations.
Abd Al-Rahman et.al.2022 [3] proposed a Hybrid Lightweight Cipher Algorithm that includes two types of encryption Feistel or SPN the chosen encryption type depends on the secret key. The Hybrid Lightweight Cipher Algorithm's SPN component analyzes 64-bit input data blocks with a 64-bit secret key and a 4-bit S-box for replacement. The algorithm customizes the number of rounds in the SPN structure (10 to 20) to meet security and performance requirements.
Aboushosha et.al.2020 [8] A Feistel lightweight cipher algorithm called SLIM has been suggested. It features a 32-bit block size and an 80-bit key size via 32 rounds, with 32 subkeys of 16 bits each created from the 80-bit key. It similarly uses four 4-bit S-box in each round; the cipher is simple to develop and execute.
BANSOD et.al.2017 [9] proposed (BORON), a new well-designed ultra-lightweight cipher with strong cryptographic properties. It prosses a 64-bit input block with 80/128-bit key size, it utilizes 4-bit S-boxes and consists of 25 rounds.
Bansod et.al.2016 [10] proposed A PICO is an ultra-lightweight, low-power encryption that uses a 64-bit plaintext and a 128-bit key length.
It consists of 32 rounds with a 4-bit s_box
Bogdanov et al. (2007)[11] introduced the most hardware and software-efficient method, PRESENT. There are 32 rounds. It has a 64-bit input block with an 80-bit/128-bit key size and a 4-bit s_box. Table 1 summarizes the related works displayed in this paper.

3. CHAOS THEORY

Chaotic systems have aperiodic, seemingly random activity even though deterministic rules govern them. This is known as the "butterfly effect" where small differences in initial conditions can lead to large variations over time. Discrete-time chaotic systems are often modeled using "chaotic maps".
A major benefit of chaotic maps is that mathematical equations fully determine their behavior while they produce complex, unpredictable outputs. Researchers have leveraged this deterministic chaos property in cryptography.
Incorporating chaotic map outputs into cipher design can enhance properties like confusion and diffusion. This helps strengthen security by making the relationship between ciphertext and plaintext more difficult to discern without the key. When combined judiciously with standard cryptographic primitives, the sensitivity to initial conditions inherent in chaos theory introduces additional unpredictability compared to conventional ciphers. The resulting hybrid ciphers retain cryptography's desired qualities like resistance to known plaintext attacks, while gaining potential robustness from chaos' ability to amplify minor perturbations in the key, IV, or plaintext exponentially over iterations. This makes the systems even harder to analyze or break using traditional cryptanalysis techniques.[12]

A. logistic map

A mapping of discrete time that depicts how a population changes over time is called the logistic map function. The following formula (equation 1)

Table 1: lightweight algorithm summary

Algorithm	SPN/Feistel	Key size	Input size	S_box	No. of rounds
AUM [6]	SPN	32 bits	32 bits	5 bits	16
Khompysh [7]	Feistel	80 bits	64 bits	4 bits	16
Hybrid [3]	SPN/Feistel	64 bits	64 bits	4 bits	10 to 20
SLIM [8]	Feistel	32 bits	80 bits	4 bits	32
BORON [9]	SPN	64 bits	80/128 bits	4 bits	25
PICO [10]	SPN	64 bits	128 bits	4 bits	32
PRESENT [11]	SPN	64 bits	80/128 bits	4 bits	32

defines it.

$$x_{n+1} = r x_n (1 - x_n)$$

where x_n represents the population size at time n , xn+1 represents the population size at the following time step, and r is a growth rate parameter ranging from 3.5 to 4. Figure 1 illustrates the logistic map's bifurcation diagram.[13] [14]

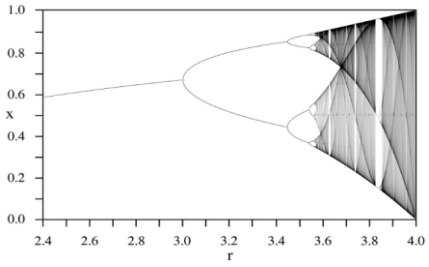

Figure 1: logistic map bifurcation diagram [13]

B.Cubic-Tent map

Aouissaoui.et.al [15] The proposed one-dimensional chaotic map is the piecewise Cubic-Tent (CT) map. It consists of the Cubic map and the Tent map, as shown below (equation 2):

$$x_{n+1} = \begin{cases} \left(\left(4 - \frac{3}{4}r\right) x_n (1 - x_n^2) + \frac{r}{2} x_n\right) mod\ 1 & x_n < 0.5 \\ \left(\left(4 - \frac{3}{4}r\right) x_n (1 - x_n^2) + \frac{r}{2}(1 - x_n)\right) mod\ 1 & x_n \geq 0.5 \end{cases}$$

The control parameter (r), iteration number (n), and modulo operation (mod) are all used. The modulo operation ensures output data falls inside the range of [0, 1]. Figure 2 depicts the CT map's bifurcation diagram, which demonstrates chaotic behavior over the full interval [0,4], with minor breaks.

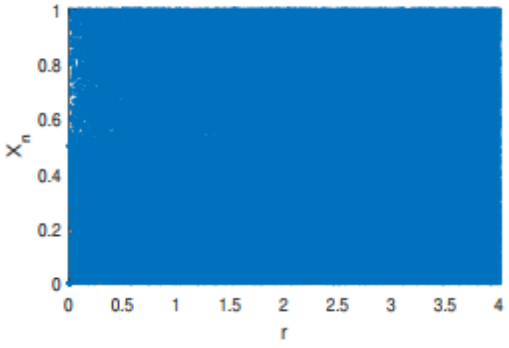

Figure 2: bifurcation diagram of the Cubic-Tent map[15].

C. Generate a random sequence

To generate a random sequence for constructing s_box and p_box using chaotic functions, the steps proposed by [16] are as follows:

1. Choose the original number x0, which is the seed of the chaotic functions. And it is another key of the algorithm.
2. using the chaotic function n-1 times to create a sequence $\{x_1, x_2, \ldots\ldots\ldots\ldots x_{n-1}\}$
3. Sorting the previous sequence and creating a new sequence $\{x'_0, x'_1 \ldots\ldots\ldots\ldots x'_{n-1}\}$
4. Find out the position of every element of the sequence $\{x_0, x_2, \ldots\ldots\ldots\ldots x_{n-1}\}$ In the sequence $\{x'_0, x'_1 \ldots\ldots\ldots\ldots x'_{n-1}\}$, then create a transform sequence $T = \{t_0, t_1, \ldots\ldots\ldots\ldots t_{n-1}\}$, sequence T produced from these steps contain values from 0 to n-1 sorted at random and not serial.

4. PROPOSED LWLRD BLOCK CIPHER STRUCTURE

The block cipher design operates as a Substitution-Permutation Network (SPN) and consists of 16 rounds.

It encrypts a 64-bit input block using an 80-bit secret key. As shown in the block diagram(Figure 3), each round begins by XORing the round input with a subkey derived from the main key via the key generation function which is an improvement of the Present key generation algorithm, choosing the 80-bit key size according to the NIST recommendation report[8]. Next, a confusion step applies a 5-bit substitution box (S-box) to the middle 60 bits of the state. These bits are divided into 12 sections which each undergo S-box substitution. Additionally, the first and last 2 bits are swapped with each other, choosing 5-bit s_box because it is moderate in security and cost between 8-bit s_box (high cost with high security) and 4_bit (low cost and low security)[17]. This layer is followed by a diffusion layer where a permutation box (P-box) rearranges the entire 64-bit state, the p_box sequence is different every time the initial secret key is changed.

These round operations of subkey XOR, S-box substitution, and P-box diffusion are repeated 16 times on the evolving ciphertext state. After the final round, another subkey is XORed with the output to produce the resulting encrypted ciphertext block. The generate round key's function handles expanding the main key into the required set of 16 round keys plus one final key. The block diagram of LWMP is illustrated in Figure 3. This fully specifies the proposed cipher as an iterative SPN utilizing cryptographic primitives like s_boxes, permutations, and key additions across multiple rounds.

The operation of each round is described in detail in the following sections.

A. s_box generation

The proposed S-box for the cipher has a size of 5 bits (1D matrix of 32 elements) and is constructed using a deterministic chaotic generator. Specifically, the Cubic-Tent map (equation 2) with parameter r=2 is utilized to generate the S-box values randomly. As described in section 3. A.

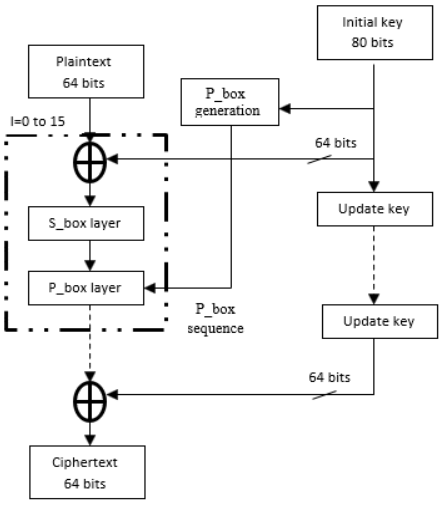

Figure 3 block diagram of LWLRD

The chaotic map iterates through 2.63×10^35 possible permutations of the 32 unique 5-bit values from inputs to outputs. This provides an enormous key space that helps obscure the relationship between plaintext and ciphertext.

Ninety randomly generated S-boxes were analyzed to evaluate their cryptographic properties and security characteristics. The optimal choice balancing factors like nonlinearity, strict avalanche criterion, and resistance to differential and linear cryptanalysis were selected for use in the cipher's substitution layer.

This S-box, presented in Table 2, will introduce confusion into the cipher by mapping each 5-bit input block to a pseudo-randomly determined 5-bit output value according to the fixed but secret S-box table. Its generation via chaotic dynamics adds another layer of complexity compared to a traditional lookup-based S-box design.

Table 2: the proposed s_box

X	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
S(x)	19	16	1A	B	4	1E	10	1C	2	14	17	A	F	1B	01	13
X	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
S(x)	09	0E	00	12	08	0D	07	0C	06	05	1F	11	1D	03	15	18

B. *p_box generation:*

The P-box used in the diffusion layer of the cipher algorithm is a 64-element bit permutation generated through a deterministic chaos function. Specifically, the logistic map (equation 1) with a parameter of r=4, which produces fully chaotic behavior, is used to randomly construct the P-box values.

The initial condition x0 input to the logistic map is derived directly from the cipher initial key. This ties the generation of the P-box to the secret key and ensures it will be different for any change to the key. Using the key as x0 provides an additional source of randomness compared to a fixed initial value.

For an x0 value of 0, Table 3 shows the resulting 64-bit P-box generated according to the process above. This P-box will serve to diffuse the bits in the ciphertext state after each round by rearranging their positions according to the fixed but key-dependent mapping defined in the table. When combined with the mixing provided by the S-box, this diffusion layer enhances the cryptographic strength of the algorithm.

Table 3: the p_box for $x_0 = 0$

X	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
P(X)	63	13	14	15	38	47	16	39	34	48	26	52	17	58	40	0
X	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
P(X)	7	23	10	35	49	27	53	18	3	43	30	59	41	1	8	56
X	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
P(X)	24	61	32	45	11	36	50	5	21	28	54	19	20	4	44	31
X	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
P(X)	60	55	29	42	2	9	22	6	57	51	25	62	33	46	37	12

C. *key schedule:*

Bogdanov et. al.(2007)[11] Proposed a lightweight block cipher(PRESENT) with a strong key generation algorithm. PRESENT Key update steps are summarized as follows

1. [k79k78 . . . k1k0] =[k18k17... . .k20, k19]
2. [k79k78k77k76] = S[k79k78k77k76]
3. [k19k18k17k16k15] = [k19k18k17k16k15] ⊕ round counter

The PRESENT key generation algorithm begins with Applying 61 bits shift to the left, applying PRESENT s_box substitution to the left-most four bits and the round_counter value i is exclusive-ored with bits k19k18k17k16k15 of K. Figure 4 depicts these steps.

Patel. Lamkuche. (2021) [18] designed a deep learning model to attack the PRESENT key, and try to retrieve the main key from the final round key, Approximately half of the final round key bits may be predicted properly. This implies that these key schedules are quite adept at evading some sort of deep-learning analysis.

To increase the security in balance with time. Therefore, the proposed LWLRD key generation algorithm is derived from the PRESENT key updating algorithm with some changes as follows:

1. [k79k78 . . . k1k0] = [k50,k51.K2,k1,K0,K79,K78,……..K52,K51]
2. [k79…….K75] = S [k79……K75]
3. [K55……K50] = S [K55……K50]
4. [K30…….K25]=S [K30……..…K25]
5. [k19……..…..k15] = [k19…….16k15] ⊕ round counter

The proposed algorithm begins with the right circular shift of the initial key by 50 bits, the 5-bit s_box is called 3 times in different places, and finally XOR between 4 bits (K19…K15] with the round counter. Figure 5 depicts the LWLRD key-updating algorithm

E-mail: Shayma.21csp70@student.uomosul.edu.iq

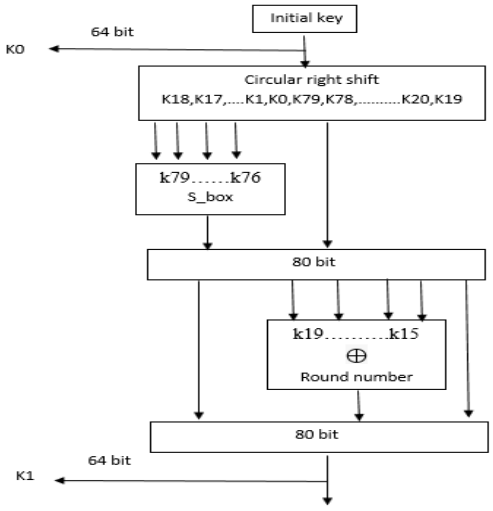

Figure 4: PRESENT key update algorithm[11]

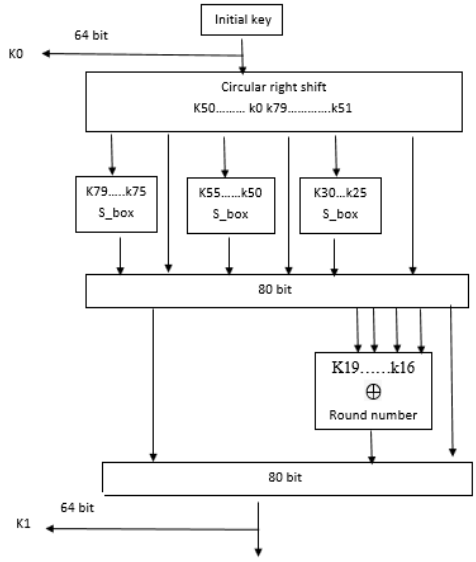

Figure 5: proposed LWLRD key update algorithm

5. *SECURITY ANALYSIS:*

This section describes an analysis of the suggested algorithm's strength.

A. S_box security analysis:

The S-box is a critical component that gives confusion property and further ensures nonlinearity in any SPN-based cryptography method; hence it receives major attention while developing any cryptography algorithm. This section compares security analysis to another.
5 bits s_box which includes, ASCON [19], SHAMASH [20], ICEPOLE[21], and Thakor [17]. ASCON, SHAMASH, and ICEPOLE are authenticated encryption algorithms that use 5-bit s_boxs. Thakor proposed a new 5-bit s_box using chaotic functions.

- *Nonlinearity:*

The S-box serves as the cipher's nonlinear component, causing confusion through its transformations. An S-box with strong nonlinearity (NL) causes significant data diffusion.
The proposed S-box design employs a chaotic function to generate a fully random substitution structure, making it extremely difficult to determine any correlating relationship between input and output values in algebraic or analytic form. Nonlinearity is measured using Hamming distances (H_d) between input-output pairs, where H(x_i,y_i)=#(x_i≠y_i) [6]. Higher H_d values indicate higher nonlinearity. Table 4 shows the hamming distance for the proposed s_box. Figure 6 shows that the proposed s_box has an average hamming distance of 2.78125 when compared to the other s_boxes. As can be observed, the proposed s_box has a higher nonlinearity than the others.

Table 4 Hamming distance of proposed s_box

input	output	Hamming distance (H_d)
0 00000	25 11001	3
1 00001	22 10110	4
2 00010	26 11010	2
3 00011	11 01011	1
4 00100	4 00100	0
5 00101	30 11110	4
6 00110	16 10000	3
7 00111	28 11100	4
8 01000	2 00010	2
9 01001	20 10100	4
10 01010	23 10111	4
11 01011	10 01010	1
12 01100	15 01111	2
13 01101	27 11011	3
14 01110	1 00001	4
15 01111	19 10011	3
16 10000	9 01001	3
17 10001	14 01110	5
18 10010	0 00000	2
19 10011	18 10010	1
20 10100	8 01000	3

21 10101	13 01101	2
22 10110	7 00111	2
23 10111	12 01100	4
24 11000	6 00110	4
25 11001	5 00101	3
26 11010	31 11111	2
27 11011	17 10001	3
28 11100	29 11101	1
29 11101	3 00011	4
30 11110	21 10101	3
31 11111	24 11000	3
Average		2.78125

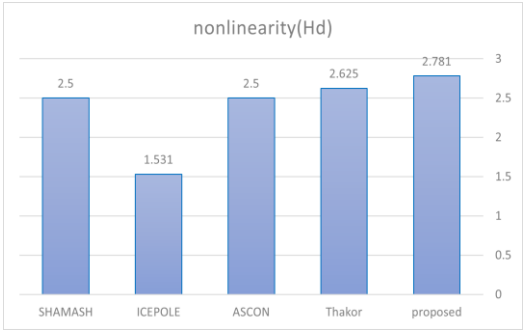

Figure 6 The nonlinearity comparison between s_boxs

- *Linear approximation probability (LP):*

In cryptography, the Linear Approximation Probability (LP) metric is used to measure the nonlinearity of a substitution box (S-box). It quantifies the maximum imbalance in the S-box's input and output bits using linear approximations. The LP is calculated by evaluating all potential input differentials (Δx) and output differentials (Δy) and calculating the highest likelihood of a linear relationship between them.
Specifically, the LP is defined as:

$$LP = \max_{\Delta x, \Delta y \neq 0} \left| \frac{\#\{x \in X | x.\Delta x = S(x).\Delta y\}}{2^n} - \frac{1}{2} \right|$$

Where:

- X is the set of all possible n-bit inputs
- S(x) is the S-box function that maps each input to an output
- |#| denotes the cardinality (number of elements) of a set
- n is the number of input bits

A lower LP value indicates greater nonlinearity in the S-box, making it more resistant to linear cryptanalysis attacks. The LP metric provides a method to objectively evaluate the security and effectiveness of an S-box design in maintaining the nonlinearity required for strong encryption and decryption against such attacks.[6] [22] [23]. The LP for the suggested s_box is 0.25. Figure 7 displays the LP of the proposed and other existing s_boxes; as can be seen, the proposed s_box has the same LP value as the other existing s_boxes.

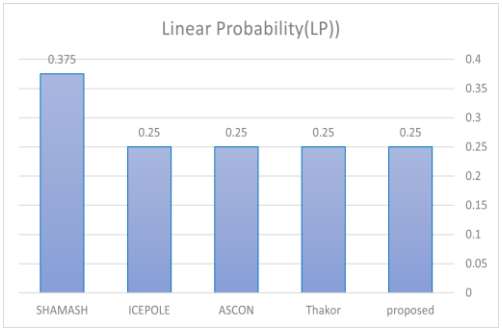

Figure 7 the LP of the proposed and others s_boxs

- *Differential approximation probability (DAP):*

Differential Approximation Probability (DAP) is a statistic used to assess a substitution box's (S-box) susceptibility to differential cryptanalysis. It measures the S-box's differential uniformity by calculating the highest likelihood of a specific output difference given an input difference. To calculate the DAP, all conceivable input differences (Δx) and output differences (Δy) are considered, and the highest probability of seeing a specific output difference for a given input difference is determined. The DAP can be defined mathematically as follows:

$$DAP = \max_{\Delta x 0, \Delta y} \left(\frac{\#\{x \in X | S(x) \oplus (S(x \oplus \Delta x) = \Delta y)\}}{2^n} \right)$$

Where:

- X is the set of all possible n-bit inputs
- S(x) is the S-box function
- $\oplus$ denotes bitwise XOR
- |#| represents the cardinality (number of elements) of a set

A lower DAP value indicates higher resistance to differential cryptanalysis since fewer predictable

output differences correspond to an input difference. S-boxes with low DAP exhibit better uniformity in their input-output behavior, making them more robust against attacks exploiting differential characteristics. Therefore, cryptographic algorithms prefer S-boxes with low DAP to enhance security.[6] [22] [24], Table 5 displays the differential distribution table of the proposed s_box, as seen the maximum value is 6 so, the DAP is equal to 0.187 which is considered a pretty good value: It indicates that the S-box is relatively unpredictable. Figure 8 displays the comparison between the proposed s_box and the DAP value of other s_boxs

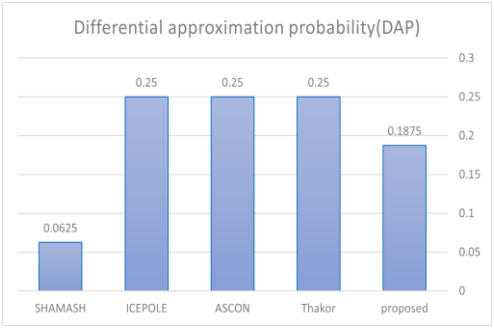

Figure 8 The DAP comparison

- *strict avalanche criterion (SAC):*

The avalanche effect happens when a little change in input bits causes a substantial change in output bits. This attribute is critical for cryptographic functions such as block ciphers because it improves diffusion - the spreading of the influence of particular input bits across many output bits. The strict avalanche criteria (SAC) is a measure of a function's avalanche properties. It requires that at least half of the output bits change on average when a single input bit is flipped. In other words, the output must have at least n/2 bits that differ from the input.

Efficient S-box design plays a key role in helping block ciphers achieve strict avalanches. S-boxes that satisfy SAC on their own provide strong diffusion to ciphertext bits from any input change. This improves the overall diffusion characteristic of the cipher and makes it significantly harder for attackers to deduce relationships between plaintext and ciphertext through differential analysis. To calculate the SAC value, assume a 5-bit input X and a sequence of input vectors, $X1$, $X2$, …,$X5$, obtained by modifying the jth bit exclusively.

The equivalent 5-bit output vectors, $Y1$, $Y2$, …, $Y5$, can be assigned using a substitution function, $Yj = S(Xj)$. To compute an avalanche vector, Vj, just XOR the output vectors Y and Yj. To generate a 5 × 5 dependency matrix, A, add the ith bit of Vj to ai, j, where ai,j is the ith member of the matrix A. Repeat the previous procedures for each vector X, then divide each element of matrix A by $2n$ (where n is the number of input/output bits) to calculate the SAC matrix. [6] [25] [26]. Table 6 displays the SAC matrix for the proposed S-box. The proposed S-box has an average SAC of 0.52 (52%), close to the optimal value of 0.5, indicating a strong avalanche impact. This signifies that the proposed S-box meets the SAC condition. Figure 9 compares the proposed S-box's SAC to that exists. Table 7: Security Analysis Comparison of the Exciting S_Boxes.

Table 6: SAC matrix of the proposed s_box

0.5	0.625	0.625	0.5	0.5
0.5	0.375	0.5	0.625	0.5
0.625	0.625	0.5	0.625	0.375
0.75	0.625	0.5	0.5	0.375
0.375	0.375	0.25	0.5	0.75

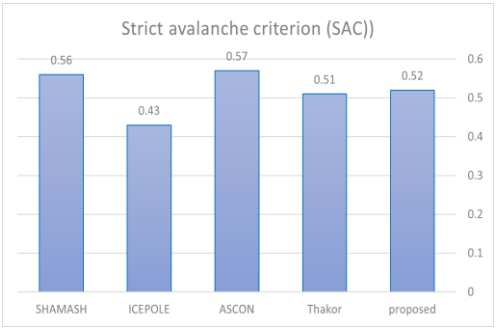

Figure 9: SAC

Table 7 The exciting s_boxes security analysis

	nonlinearity	LP	DAP	SAC
Proposed	2.781	0.25	0.1875	0.52
Thakor	2.625	0.25	0.25	0.51
ASCON	2.5	0.25	0.25	0.57
SHAMASH	2.5	0.375	0.25	0.43
ICEPOLE	1.531	0.25	0.0625	0.56

B. Key schedule evaluation:

The proposed key updating algorithm is derived from the present key updating algorithm. Therefore, in this section comparisons are made between the two algorithms

Table 5: differential distribution table (DDT) of proposed s_box

32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	2	0	2	0	2	0	0	0	2	2	2	2	2	0	2	4	0	2	0	2	0	0	0	2	0	0	2	2	0
0	2	2	2	0	0	0	0	4	2	0	0	0	0	2	2	0	0	0	0	4	2	0	0	0	2	0	2	2	2	2	0
0	0	0	2	2	2	0	0	2	0	2	0	2	0	4	0	0	0	2	0	0	0	2	2	2	0	4	2	2	0	0	0
0	2	0	2	0	0	2	2	2	2	4	0	0	2	0	2	0	0	0	0	0	0	0	2	2	0	2	0	2	0	2	2	0
0	0	0	0	6	2	4	4	0	0	0	2	2	0	0	0	0	0	2	0	0	2	0	0	2	2	0	4	0	0	0	0
0	0	4	2	0	0	0	2	2	2	2	0	0	0	2	0	0	2	2	2	0	2	0	0	2	0	0	0	0	2	2	2
0	0	0	0	2	6	2	0	0	2	0	0	4	2	0	2	2	2	0	0	0	2	0	0	0	0	2	0	2	0	2	0
0	2	2	2	0	2	0	0	0	0	0	4	0	2	2	4	0	2	2	0	2	2	0	0	0	0	0	2	0	0	0	2
0	0	0	2	0	0	0	0	2	0	0	2	2	4	0	0	4	4	0	0	2	0	0	0	0	2	0	0	2	2	0	4
0	0	0	0	0	2	2	2	0	0	0	0	0	2	2	2	0	0	0	0	0	2	2	2	2	0	2	0	2	2	0	6
0	2	0	0	2	2	0	0	0	2	0	2	0	0	2	0	2	4	0	4	2	0	0	2	4	0	0	0	0	0	0	2
0	0	0	0	0	0	2	2	2	0	4	0	0	4	2	0	0	0	0	0	2	2	4	0	4	0	0	2	0	2	0	0
0	0	2	0	0	0	0	2	0	2	4	4	0	2	0	0	2	0	0	4	0	0	2	0	2	2	2	0	2	0	0	0
0	2	0	0	0	2	0	0	2	2	0	0	0	0	0	0	2	2	2	2	2	2	2	2	2	0	0	0	4	2	0	0
0	2	2	2	4	0	0	0	0	2	4	0	0	0	2	2	0	2	2	0	0	0	0	0	2	0	2	0	2	0	0	2	0
0	0	0	0	2	0	0	0	0	2	0	0	2	2	0	0	0	4	2	2	2	2	0	0	2	4	2	2	2	0	0	0	0
0	0	0	0	0	0	6	2	2	2	0	2	2	0	0	0	0	0	2	0	0	2	2	2	0	2	0	2	2	0	0	2
0	0	0	4	2	4	0	0	0	0	0	0	0	0	0	2	2	4	2	2	0	0	0	0	2	2	2	0	2	2	0	0
0	0	4	0	0	0	0	0	2	0	0	4	2	0	4	0	0	0	2	2	4	0	2	2	0	2	0	0	0	2	0	0
0	0	4	0	0	0	0	2	0	2	0	0	0	2	2	0	4	2	2	0	0	0	0	2	0	0	0	0	2	0	2	4	2
0	2	2	0	0	0	0	0	0	0	2	4	0	4	0	0	2	2	0	0	0	0	2	0	2	2	0	0	0	0	2	2	2
0	0	0	0	2	0	2	2	0	2	4	0	4	0	0	0	2	0	4	0	0	0	2	2	0	2	2	0	0	0	2	0
0	2	0	2	4	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	2	6	2	4	0	0	2	0	0	0	6	0
0	0	0	0	2	4	2	2	0	0	0	2	0	0	0	0	0	0	0	2	0	0	2	2	2	2	4	0	0	2	0	4
0	0	2	2	0	2	0	2	2	2	2	2	2	2	0	0	2	0	0	2	4	0	0	0	0	0	0	0	2	2	0	0
0	0	2	0	2	0	6	2	2	2	0	0	0	2	2	0	0	2	0	0	0	0	0	2	0	0	0	0	2	0	4	2
0	2	0	4	0	0	0	0	2	2	0	2	2	2	0	0	2	0	0	2	2	0	0	0	0	2	0	2	4	0	0	2
0	4	2	0	2	0	4	0	0	0	2	0	0	2	0	4	2	0	0	2	0	4	0	0	0	2	0	2	0	0	0	0

- *Bit Differences between Round Keys*

The goal of this test is to evaluate the relationship. between the round keys in a cryptographic algorithm. The test involves XORing consecutive round keys to determine the number of bit differences between them[27]. The test uses an initial key, k0, which is an 80-bit value set to 0x00000000000000000000000000000000000000. The table below shows the number of bit differences between consecutive round keys for the "PRESENT" and the "proposed LWLRD" keys. The results (Table 8, Table 9) indicate that the "PRESENT" key schedule algorithm has a low number of bit differences (Below 10) in the first 5 rounds. However, the " proposed LWLRD " key schedule algorithm shows a greater number of bit differences, with at least 10 bits differing between consecutive round keys starting from the first round.

This analysis is important because the round keys in a strong cryptographic algorithm should exhibit a high degree of avalanche effect. That is, small changes in the initial key should result in large, unpredictable changes in the subsequent round keys. A higher number of bit differences between consecutive round keys is a desirable property, as it helps to ensure the key schedule is resistant to attacks that exploit weaknesses in the key expansion process. The observed differences between the "PRESENT" and " proposed LWLRD " key schedules suggest that the " proposed LWLRD " key schedule may provide better security properties in terms of key schedule strength.

C. Block cipher security analysis

this section contains the analysis of 16 round block cipher

Table 8: the key generated from the PRESENT key updating algorithm with the bit difference

no	Generated key (present key schedule algorithm)	No of bit difference $(k_i XOR\ k_{i+1})$ I=0,1,…..n-1
0	00000000000000000000000000000000000000000000000000000000000000000000	
1	11001000000000000001000000000000000000000000000000000000000000000000	4
2	11001000000000000011100100000000000001000000000000000000000000000000	4
3	11001000000000000010100100000000000011100100000000000001000000	5
4	11001000000000000101100100000000000010100100000000000011100100	7
5	01001000000000010100100100000000000101100100000000000010100100	7
6	01001000000000111110100100000000010100100100000000000101100100	8
7	01001000000000101111100100000000111110100100000000010100100100	7
8	01001000000001010000100100000000101111100100000000111110100100	12
9	01001000000101000001100100000000101000010010000000010111100100	12
10	01001000001111100010100100000010100000110010000000010100000100100	15
11	01001000001011110011100100000111110001010010000001010000011100100	11
12	01001000010100000100100100000101111001110010000011111000010100100	18
13	01001001010000010101100100001010000010010010000010111100011100100	17
14	01001011111000100110100100101000001010110010000101000000100100100	21
15	01001010111100110111100101111100010011010010010100000101011100100	15

Table 9: the key generated from the LWLRD key updating algorithm with the bit difference

no	Generated key (present key schedule algorithm)	No of bit difference $(k_i XOR\ k_{i+1})$ I=0,1,…..n-1
0	00000000000000000000000000000000000000000000000000000000000000000000	
1	11001000000000000010000011001000000000000000000000011001000000000	10
2	00101000000000000010000001100111001000000000000000111001110010000	12
3	00101110010000000011000001100100101000000000000001011001110011100	11
4	00101110011100100100000001011000101110010000000011110011100100 10	17
5	00101110010010100101000001101000101110011100100100110011101100010	16
6	00101101100010111111000000101100101110010010100101110011101100010	15
7	00101110100010111110110010010000101101100010111111110010101100 10	19
8	00101010110010110001001011111000101110100010111110001010010000010	22
9	11110001000010111111001011100000101010110010110001111101111001000 10	26
10	10101111100010110000001011010111110001000010111111111110110000010	26
11	10101110000010100000001010111010101111100010110000111101010111111	23
12	10101101011111001000001011100010101110000010100000111100111010 10	20
13	10101011101010110011001010100110101101011111001000111101100010 10	24
14	10101110001010110110001011100110101011101010110011111100100110 10	19
15	10101010011010111010111100001010101110001010110110111101100110 10	18

- *avalanche criteria*

The avalanche test is satisfied if at least 50% of the output bits change when changing one bit of the input (either the plaintext or the key). This threshold is necessary to ensure that the cryptographic function is sensitive enough to detect slight changes in the input, rendering it resistant to various cryptanalysis approaches. To be considered

E-mail: Shayma.21csp70@student.uomosul.edu.iq

secure, a cryptographic function must meet two important qualities known as the avalanche criterion. [28] [29]. These criteria are:

1. Fixed Key and Differing Plaintext:
   This criterion uses a fixed key, and multiple plaintext inputs are tested.
   Each pair of input plaintexts should differ in exactly one bit. The resulting ciphertexts for these input pairs should have a significant number of bits (at least 50%) that differ.

2. Fixed Plaintext and Differing Key:
   In this criterion, a fixed plaintext is used, and multiple keys are tested.
   Each pair of keys should differ in exactly one bit.
   The resulting ciphertexts for these key pairs should have a significant number of bits (at least 50%) that differ.

To satisfy these criteria choose the initial plaintext = 0x0000000000000000(64-bit)
An initial key = 0x00000000000000000000000000000000000 (80 bit), using hamming distance to calculate the difference between two ciphertexts. Table 10 and Table 11 display the result of hamming distance after performing the two avalanche criteria. As seen the proposed cipher algorithm (LWLRD) satisfies the avalanche test after 10 rounds.

Table 10: fixed key with different plaintext

	After 10 rounds	After 16 rounds
Minimum bit difference	22	24
Maximum bit difference	41	41
Average bit difference (after 64-bit flipping)	0.488	0.497
Average bit difference (after 80-bit flipping)	0.503	0.508

Table 11: fixed plain with different key

	After 10 rounds	After 16 rounds
Minimum bit difference	25	26
Maximum bit difference	39	40

- *National Institute of Standards and Technology (NIST) test group:*

The NIST (National Institute of Standards and Technology) test suite is a widely used collection of statistical tests for determining the randomness of binary sequences, such as ciphertext in cryptography. The NIST test suite consists of numerous statistical tests, each of which returns a p-value.
The p-value expresses the likelihood that a perfectly random sequence would have a test statistic at least as extreme as the one observed for the sequence under test. If the p-value is greater than or equal to 0.01, the sequence is regarded to have passed the test, implying that it is most likely random. In contrast, if the p-value is less than 0.01, the sequence is judged to have failed the test, implying that it is not random enough. [27]
The suggested algorithm was tested using the NIST test suite, with the results shown in Table 12. The table demonstrates that the proposed algorithm passed all 12 random tests with p-values greater than the 0.01 criterion. These findings demonstrate that the ciphertext generated by the proposed method is extremely random, passing all 12 statistical tests in the NIST test suite.

6. PERFORMANCE EVALUATION:

This section evaluates the proposed LWLRD algorithm in terms of execution time, throughput, and memory consumption. We compare LWLRD to the PRESENT cipher block to determine its relative efficiency.
All experiments were conducted on a Laptop having Intel(R) Core(TM) i5-3427U CPU @ 1.80GHz, 2301 MHz, 2 Core(s), 4 Logical Processor(s) with 12 GB RAM on Windows 10 Pro, 64-bit operating system. The two algorithms are developed using Python 3.12.3

A. Execution time

The execution time for cipher blocks refers to the time necessary to process a block of data using an encryption algorithm. The execution time depends on the cipher type, block size, and the hardware or software platform employed.[30]. Table 13 compares the proposed algorithm to the

PRESENT algorithm; the difference is rather minimal.

B. *Memory usage*

The memory required to store encrypted data, known as the cipher block's memory usage, depends on factors such as the encryption method, the data size, and the system's hardware and software [31]. Table 14 shows that the proposed LWLRD algorithm has a relatively small memory footprint, indicating efficient memory usage. While there is a slight difference in memory consumption compared to an existing PRESENT algorithm, this difference is not significant.

Table 12: NIST suit test

Test name	P_value	Conclusion
Frequency (Monobit) Test	0.2040841777655733	pass
Frequency Test within a Block	0.2040841777655733	pass
Runs Test	0.9597443417058545	pass
Test for the Longest Run of Ones in a Block	0.5390898267391488	pass
Discrete Fourier Transform (Spectra) Test	0.5999691396040943	pass
Non-overlapping Template Matching Test	0.9999999999999261	pass
Serial Test	0.4989610874592239	pass
Approximate Entropy Test	1.0	pass
Cumulative Sums Test (Forward)	0.2550069993732869	pass
Cumulative Sums Test (Backward)	0.4078904265537182	pass
Random Excursions Test	0.9996100613790039	pass
Random Excursions Variant Test	0.4795001221869535	pass

Table 13: time needed to perform LWMP algorithm compared with PRESENT

Algorithm	Encryption + Key Generation (millisecond)	Decryption + Key Generation (millisecond)	Key Generation (millisecond)	Encryption +decryption +key generation
LWLRD (16 rounds)	1.3	2.31	0.21	3.03
PRESENT (32 rounds)	1.06	1.83	0.14	2.85

Table 14: memory usage (megabyte)

	Encryption +key generation	Decryption +key generation	Key generation	Encryption +decryption +key generation
Proposed LWMP (16 rounds)	0.003969 MB	0.003969 MB	0.001068 MB	0.004005 MB
PRESENT (32 rounds)	0.002162 MB	0.002162 MB	0.001782 MB	0.002198 MB

C. Throughput

Throughput is a measure of how many units of information a system can handle in a particular period. The throughput of the encryption can be calculated as in equation [32] [33]:

$$\text{Throughput} = \frac{\text{Tp}}{\text{Et}}$$

were
Tp: Total plain text encrypted
Et: Encryption time (second)

Table 15 shows the throughput of the proposed LWLRD compared with the PRESENT algorithm The throughput values shown in the table were calculated by averaging the results of 300 separate runs of the algorithm. Each run consisted of 100 iterations, and the average throughput was calculated across all 300 runs

Table 15 throughput of LWLRD and PRESENT

	Blocks(64bit) /seconds	Bits/seconds	Kilobits/seconds
LWLRD (16 rounds)	890.86	57015.27	55.679
PRESENT (32 rounds)	1434.57	91812.9	90.129

7. CONCLUSION:

This paper has presented LWLRD, a novel lightweight cipher block designed to address the growing demand for secure and efficient cryptographic solutions in resource-constrained environments. LWLRD leverages the inherent properties of chaotic functions to achieve a robust balance between security and performance.
The meticulously designed 5-bit S-box, derived from a comprehensive analysis of cubic tent functions, exhibits high nonlinearity while minimizing Linear Probability (LP) and Differential Approximation Probability (DAP) values, enhancing resistance against linear and differential cryptanalysis. The dynamic diffusion layer, generated from the initial secret key, further strengthens the cipher's security by ensuring a non-fixed structure that further enhances security by introducing unpredictability and complexity.
The suggested key generation technique, an improved variant of the PRESENT approach, provides flexibility in accommodating keys of varied sizes while incurring low-performance overhead. LWLRD's robustness and randomness have been confirmed by rigorous testing, including avalanche and NIST statistical randomness tests. The performance evaluation reveals its efficiency in terms of encryption/decryption execution time, throughput, and memory use.

8. REFERENCES:

[1] M. Sajjad, T. Shah, and R. J. Serna, “Designing Pair of Nonlinear Components of a Block Cipher over Gaussian Integers,” Comput. Mater. Contin., vol.75, no.3, ISSN:1546-2226(online), 2023, doi: 10.32604/cmc.2023.035347.

[2] J. H. Zadeh and A. G. Bafghi, “Evaluation of Lightweight Block Ciphers in Hardware Implementation: A Comprehensive Survey,” 2016.

[3] S. Q. Abd Al-Rahman, O. A. Dawood, and A. M. Sagheer, “A Hybrid Lightweight Cipher Algorithm,” Int. J. Comput. Digit. Syst., vol. 11, no. 1, pp. 463–475, Jan. 2022, doi: 10.12785/ijcds/110138.

[4] C. E. Shannon, “Communication Theory of Secrecy Systems*,” Bell Syst. Tech. J., vol. 28, no. 4, pp. 656–715, Oct. 1949, doi: 10.1002/j.1538-7305.1949.tb00928.x.

[5] C. Pei, Y. Xiao, W. Liang, and X. Han, “Trade-off of security and performance of lightweight block ciphers in Industrial Wireless Sensor Networks,” EURASIP J. Wirel. Commun. Netw., 2018, doi: 10.1186/s13638-018-1121-6.

[6] H. M. Z. Al Shebli and B. D. Beheshti, “Light Weight Cryptography for Resource Constrained IoT Devices,” in Proceedings of the Future Technologies Conference (FTC)

E-mail: Shayma.21csp70@student.uomosul.edu.iq

2018, vol. 880, K. Arai, R. Bhatia, and S. Kapoor, Eds., in Advances in Intelligent Systems and Computing, vol. 880. , Cham: Springer International Publishing, 2019, pp. 196–204. doi: 10.1007/978-3-030-02686-8_16.

[7] A. Khompysh, N. Kapalova, O. Lizunov, D. Dilmukhanbet, and S. Kairat, "Development of a New Lightweight Encryption Algorithm," Int. J. Adv. Comput. Sci. Appl., vol. 14, no. 5, 2023.

[8] B. Aboushosha, R. A. Ramadan, A. D. Dwivedi, A. El-Sayed, and M. M. Dessouky, "SLIM: A Lightweight Block Cipher for Internet of Health Things," IEEE Access, vol. 8, pp. 203747–203757, 2020, doi: 10.1109/ACCESS.2020.3036589.

[9] G. Bansod, N. Pisharoty, and A. Patil, "BORON: an ultra-lightweight and low power encryption design for pervasive computing," Front. Inf. Technol. Electron. Eng., vol. 18, no. 3, pp. 317–331, Mar. 2017, doi: 10.1631/FITEE.1500415.

[10] G. Bansod, N. Pisharoty, and A. Patil, "PICO : An Ultra Lightweight and Low Power Encryption Design for Ubiquitous Computing," Def. Sci. J., vol. Vol. 66, no. No. 3, 2016, doi: DOI : 10.14429/dsj.66.9276.

[11] A. Bogdanov et al., "PRESENT: An Ultra-Lightweight Block Cipher," in Cryptographic Hardware and Embedded Systems - CHES 2007, vol. 4727, P. Paillier and I. Verbauwhede, Eds., in Lecture Notes in Computer Science, vol. 4727. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 450–466. doi: 10.1007/978-3-540-74735-2_31.

[12] R. S. Salman, A. K. Farhan, and A. Shakir, "Creation of S-Box based One-Dimensional Chaotic Logistic Map: Colour Image Encryption Approach," Int. J. Intell. Eng. Syst., vol. 15, no. 5, pp. 378–389, Oct. 2022, doi: 10.22266/ijies2022.1031.33.

[13] S. Chen, S. Feng, W. Fu, and Y. Zhang, "Logistic Map: Stability and Entrance to Chaos," J. Phys. Conf. Ser., vol. 2014, no. 1, p. 012009, Sep. 2021, doi: 10.1088/1742-6596/2014/1/012009.

[14] M. S. Fadhil, A. K. Farhan, and M. N. Fadhil, "Designing Substitution Box Based on the 1D Logistic Map Chaotic System," IOP Conf. Ser. Mater. Sci. Eng., vol. 1076, no. 1, p. 012041, Feb. 2021, doi: 10.1088/1757-899X/1076/1/012041.

[15] I. Aouissaoui, T. Bakir, A. Sakly, and S. Femmam, "Improved One-Dimensional Piecewise Chaotic Maps for Information Security," J. Commun., vol. Vol. 17, No. 1, Jan. 2022, doi: 10.12720/jcm.17.1.11-16.

[16] Yang, G., & Zhou, Y. (2009). LSB Algorithm Research Based on Chaos. 2009 Ninth International Conference on Hybrid Intelligent systems. doi:10.1109/his.2009.201

[17] V. A. Thakor, M. A. Razzaque, A. D. Darji, and A. R. Patel, "A novel 5-bit S-box design for lightweight cryptography algorithms," J. Inf. Secur. Appl., vol. 73, p. 103444, Mar. 2023, doi: 10.1016/j.jisa.2023.103444.

[18] N. K. Patel and H. S. Lamkuche, "Deep Learning Based Analysis of Key Scheduling Algorithms of Advanced Ciphers," Cryptol. EPrint Arch., May 2021, [Online]. Available: https://eprint.iacr.org/2020/981.

[19] C. Dobraunig, M. Eichlseder, and F. M. Schläffer, "ASCON v1.2: Lightweight Authenticated Encryption and Hashing," Journal of Cryptology. Accessed: Jul. 01, 2021. [Online]. Available: https://doi.org/10.1007/s00145-021-09398-9

[20] D. Penazzi and M. Montes, "Shamash (and Shamashash) (version 1)," Lightweight Cryptogr. Stand. Process Round, vol. 1, 2019.

[21] P. Morawiecki et al., "ICEPOLE: High-Speed, Hardware-Oriented Authenticated Encryption," in Advanced Information Systems Engineering, vol. 7908, C. Salinesi, M. C. Norrie, and Ó. Pastor, Eds., in Lecture Notes in Computer Science, vol. 7908. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 392–413. doi: 10.1007/978-3-662-44709-3_22.

[22] N. Siddiqui, "A Novel Approach to Building Substitution-Boxes with Dihedral Group," vol. 4, no. 12, 2023.

[23] N. F. Mohd Esa, S. F. Abdul-Latip, and N. A. Abu, "A New Design of Substitution Box with Ideal Strict Avalanche Criterion," Malays. J. Math. Sci., vol. 16, no. 4, pp. 697–715, Dec. 2022, doi: 10.47836/mjms.16.4.04.

[24] M. Usama, O. Rehman, and S. Rizvi, "An efficient construction of key-dependent substitution box based on chaotic sine map" 2019, International Journal of Distributed Sensor Networks. Available:https://doi.org/10.1177/1550147719895957

[25] J. C. H. Castro, J. M. ´ ´ıa Sierra, and A. Seznec, “The strict avalanche criterion randomness test,” Mathematics and Computers in Simulation., vol. 68, no. 1, 2005, DOI: 10.1016/j.matcom.2004.09.001.

[26] D. Upadhyay, N. Gaikwad, M. Zaman, and S. Sampalli, “Investigating the Avalanche Effect of Various Cryptographically Secure Hash Functions and Hash-Based Applications,” *IEEE Access*, vol. 10, pp. 112472–112486, 2022, doi: 10.1109/ACCESS.2022.3215778.

[27] M. Imdad, S. N. Ramli, and H. Mahdin, “An Enhanced Key Schedule Algorithm of PRESENT-128 Block Cipher for Random and Non-Random Secret Keys,” *Symmetry*, vol. 14, no. 3, Art. no. 3, Mar. 2022, doi: 10.3390/sym14030604.

[28] K. Mohamed, M. N. M. Pauzi, F. H. H. M. Ali, and S. Ariffin, “Analyse "On Avalanche Effect In Cryptography Algorithm",” *Eur. Proc. Multidiscip. Sci. EpMS*, 2021, doi: 10.15405/epms.2022.10.57.

[29] R. Verma and A. K. Sharma, “Cryptography: Avalanche effect of AES and RSA,” *Int. J. Sci. Res. Publ. IJSRP*, vol. 10, no. 4, 2020, doi: 10.29322/IJSRP.10.04.2020.p10013.

[30] M. MANAA and R. H. JWDHA, “A proactive data security scheme of files using minhash technique,” *J. Theor. Appl. Inf. Technol.*, vol. 96, 2018.

[31] B. A. Buhari, A. A. Obiniyi, K. Sunday, and S. Shehu, “Performance Evaluation of Symmetric Data Encryption Algorithms: AES and Blowfish,” *Saudi J. Eng. Technol.*, vol. 04, no. 10, pp. 407–414, 2019, doi: 10.36348/SJEAT.2019.v04i10.002.

[32] S. Kansal and M. Mittal, “Performance evaluation of various symmetric encryption algorithms,” *International Conference on Parallel, Distributed and Grid Computing*, Solan, India: IEEE, 2014, pp. 105–109. doi: 10.1109/PDGC.2014.7030724.