Efficient Security for Resource-Constrained Devices with LPM: A Lightweight Protocol Utilizing Chaotic Map-Based S-Box

M. Prakash*, K. Ramesh

Department of Computer Science and Engineering, National Institute of Technology Warangal, Telangana, 506004, India

---

Abstract

The widespread adoption of resource-constrained devices, including lightweight devices such as RFID devices, necessitates the development of encryption algorithms that effectively balance security and efficiency. In this study, we conducted a comprehensive evaluation of various lightweight block ciphers, focusing on their suitability for lightweight devices. Through our assessment, we identified vulnerabilities in the LRBC cipher, highlighting the need for improved solutions. To address these challenges, we propose the Lightweight Securing Protocol (LSP), a novel approach that leverages a chaotic map-based S-Box. Our protocol incorporates a sophisticated key scheduling scheme, integrating a permutation box and function to enhance resistance against key-related attacks while maintaining lightweight characteristics. Furthermore, we meticulously designed a highly efficient S-Box, yielding promising metric results, including an average Avalanche effect of 50%, Signal to Noise Ratio of 2.024, the sum of square indicator of 1048, and an absolute indicator of 16. Thorough analysis and detailed results are presented in the results section, affirming the efficacy and robustness of our proposed protocol.

Keywords: IoT,Lightweight Algorithms,Encryption,Chaotic Equation,RFID.

---

1. Introduction

Conventional cryptographic algorithms are often time-consuming and memory-intensive, which can be problematic for a category of lightweight devices known as resource-constrained devices. These devices typically have low memory and battery life or no battery at all. Therefore, efficient algorithms that meet certain underlying conditions are necessary to secure these devices[1].

IoT devices are informally divided into two main categories based on the resources they use, The first category, rich in resources, includes devices that are fully equipped with high memory and high power, allowing for the use of conventional cryptographic algorithms. The second category, poor in

---

Email addresses: `mp721084@student.nitw.ac.in` (M. Prakash*), `kramesh@nitw.ac.in` (K. Ramesh)

Preprint submitted to International Journal of Communication Systems *October 15, 2023*

resources[2], includes lightweight devices[3], which are also known as resource-constrained devices. Conventional cryptographic algorithms are not suitable for these devices, as they have low memory and battery resources. This is where lightweight cryptographic algorithms come in.

In this paper, we focus on the second type of IoT devices, which require strong security due to the significant amount of real-world data being transacted over the internet every second. [4].

Despite the availability of some good designs, they fall short in meeting all the requirements[5] for a lightweight algorithm. The necessary requirements include performance, cost, and security. However, achieving all three simultaneously is challenging, as demonstrated by conventional cryptographic algorithms. While they exhibit good performance and security, their cost is high. If we attempt to decrease the cost, performance and security will inevitably suffer. It is also challenging to achieve all three requirements[6] for lightweight algorithms. This paper reviews some recent lightweight algorithms mentioned in the literature survey[7].

1.1. Contributions

In this paper, we make the following contributions:

1. **New S-Box Design Method:** We introduce a novel method for designing S-Boxes using logistic map equation, and apply it to the proposed lightweight block cipher, LPM. In, Previous research such as LRBC introduced by Biswas et al.[8], suffered from vulnerabilities to key-related attacks and known plaintext attacks. Our new S-Box design method enhances the security of the cipher.
2. **Enhanced Key Scheduling Scheme:** We introduce a new key scheduling scheme that takes a 32-bit input and produces a processed 16-bit output, utilized as pre-key-whitening. This enhancement strengthens the key management and makes the cipher more resilient against related key attacks.
3. **Chaotic Logistic Map for S-Box Design:** To design the S-Box, we utilize a chaotic logistic map, which offers higher non-linearity. In the LRBC scheme, the S-Box was constructed using only XOR operations, lacking non-linear functions. This design flaw weakened the entire scheme. With the chaotic logistic map, we address this weakness.
4. **Secure and Efficient Protocol:** The proposed LPM cipher has been rigorously tested and proven to be secure, space-efficient, and time-efficient. It addresses the drawbacks identified in the LRBC scheme, providing a more robust encryption protocol.

This paper is structured as follows: Section.2 provides an overview of related work on lightweight block ciphers. Section.3 reviews the LRBC scheme and analyzes it. Section.4 introduces the proposed scheme and explains how the newly developed key scheduling and S-Box generation were designed, along with an explanation of the encryption process. Section.5 provides results and analysis, including a comparison with standard lightweight block ciphers. Finally, the last Section.6 concludes the paper and outlines future work.

2. Related Work

Lightweight securing protocols are critical for dealing with small data sizes, low device power[9], and low computing equipment costs. So, while building a cryptography protocol for a compact computer device, the key goal should be to make it as light as possible in terms of memory use[10], chip size, and power consumption. Though there are many cryptographic secure protocols, all of them lack one among security, performance, and cost[11], an ideal lightweight protocol should possess these requirements.

Two types of structures are there to build block ciphers, these are not mandatory to be followed to design the block ciphers[9] but can be combinedly used those two structures or else can be introduced a new structure. In this way, SFN[12] is a lightweight block cipher which is operated using both the structures such as the substitution-permutation network and Feistel network. The author has taken the SP network to make it used as a Feistel network. Encryption/Decryption has been done by the substitution-permutation network and key scheduling has been done by the Feistel network. The sizes of the key and block are 96 and 64 bits respectively. And, they use 32 rounds of encryption and decryption. Security has been analysed on differential cryptanalysis, linear cryptanalysis and algebraic attacks.

CHAM is a family of lightweight block ciphers[13] suggested by a group of researchers. It is based on a 4-branch Feistel structure that performs ARX[14] (Addition, Rotation, XOR) operations (Koo et al. 2017). They used a simple key schedule that was implemented without altering key status, resulting in lower hardware implementation areas. Different round functions were set up in this technique so that the reduced set of round keys may be reused repeatedly.

Lightweight cryptography system for IoT devices using DNA, this scheme[15] is a lightweight block cipher the operation of the algorithm is quite different from the regular block ciphers. It uses DNA sequences for genuinely random keys. And the block size is variable so that the algorithm can be performed by different systems. No specific S-Box is used in the cipher.

The cipher mcrypton[16] operates the data by taking the byte blocks to array nibbles. Here they take an 8byte block and convert them to a 4*4 array nibble. The round function is being operated in the following steps

- Transposing the columns as rows.
- Substitution is respective to nibbles.
- Bit-permutation is respective to column-wise.
- Key scheduling.

In this algorithm, the round function is repeated for 12 rounds to get the final cipher text and to get back to the plaintext the process is almost the same but with a different key scheduling algorithm.

CipherName	**BlockSize**	**KeySize**	**Rounds**	**Cipher Weakness**
SFN[12]	64bit	96bit	32	Differential and Linear attacks
PRESENT[17]	64bit	80,128bit	31	Differential cryptanalysis with limited rounds
LEA[18]	128bit	128/192/256bit	24/28/32	Related Key Attack
KLIEN[19]	64/80/96bit	64/80/96bit	12/16/20	Related Key Attack
SLIM[20]	32bit	80bit	32	Differential and Linear attacks
M-XXTEA[21]	64bit	128bit	-	Key storage Problem
CHAM[13]	128bit	128/192/256bit	80/96/112	Side Channel Attacks
DNA[15]	-	-	-	Key storage Problem
Bflexγ[22]	32bit	32bit	8	Side Channel Attacks

Table 1: Lightweight Block Ciphers and their Weaknesses.

The modified XXTEA block cipher[21]. It has simple construction compared to AES block cipher but efficient algorithm for constrained devices like IoT. It uses a chaotic key generator so that the key for each block of plain text will be changed dynamically, providing more security[23]. The XXTEA algorithm was proven to be Chosen Plaintext Attack(CPA) and key-related issues. They have been solved in the M-XXTEA protocol.

There is a protocol called PRESENT[17]. It is one of the standardised and NIST-recommended lightweight algorithms now. PRESENT is a simple algorithm which consists of S-box and P-Box only. It is very popular for its simplicity in algorithm and because of the strong S-Box. It has been the strongest among all lightweight block ciphers. In this scheme, they used the XOR operation in the key scheduling scheme as the total rounds are 32, the 32nd round key is used as the post-whitening key[24]. This scheme uses the Substitution-Permutation Network, so as per the structure it is using confusion is created by the S-Box and diffusion is created by the P-Box.

There is an operation called ARX[14] which consists of addition, rotation and XOR operations. This ARX operation is largely used in block ciphers. The block cipher LEA uses this ARX operation The round function of LEA[18] consists of ARX operations. 32-bit ARX operation is used in the Lightweight Encryption Algorithm(LEA). The rotations and the block permutations are used for diffusion. Its simplicity makes it so unique and its encryption speed is very less. It has high-speed performance also. The diffusion and the nonlinearity it provides give good performance.

SLIM[20] is a symmetric encryption algorithm in which the symmetric algorithm uses the same key for both encryption and decryption. The only difference is that in the decryption process the algorithm uses sub-keys of the key in reverse order. Security and simplicity are two main considerations and themes in the SLIM algorithm. It has immunity against the exhaustive search attack. Because the key length used in the protocol is 80bit. This is actually the recommendation of the NIST report. Using the NIST recommendations report for key length (key length ≤ 80). SLIM achieves both confusion and diffusion properties with S-box and permutation box respectively.

Lblock block cipher[25], In This work, they introduced a new lightweight block cipher. In which the block size of the cipher is 64bit and the size of the key is 80bit. The cipher takes 1320 of Gate Equivalents(GE). Which is enough for a group of lightweight devices. And they are also proven that it is immune to differential cryptanalysis, related key attacks and impossible differential attacks. It has been implemented in both software and hardware.

Prince is another block cipher, the speciality of this block cipher is it's been implemented in hardware with reduced latency. Also, it has got that the time taken for the decryption is very less when it is implemented on top of the encryption process[26], to be short, it holds that decryption with a key is directly related to the encryption key[27]. The author represented this property as α reflection which is beneficial to save the cost in time at the time of decryption.

Bflex[22] γ is a recently published article on a lightweight block cipher, in this cipher they used the linear feedback register for key management. And they made use of variably changed ciphertext in the intermediate rounds.

3. Review and Analysis of the LRBC Scheme

Recently, A. Biswas et al.[8] proposed a lightweight block cipher scheme with improved security and resource efficiency. We have reviewed the scheme and identified some vulnerabilities in the design.

Upon examining the scheme's design, it is apparent that the plaintext is split into four-bit blocks, with PT^1 consisting of the first, second, ninth, and tenth bits, and so on for PT^2, PT^3, and PT^4. The first and last four bits of the plaintext are then subject to an XNOR operation with k^a and k^d, respectively, where the key is divided into four sub-parts of four bits each: k^a, k^b, k^c, and k^d. The middle four bits of the plaintext are XORed with k^b and k^c, respectively.

```
Algorithm 1: LRBC Encryption Algorithm
  Data: 16-bit Intermediate Cipher Text
  Result: Final Cipher Text
    1. S-Box Computation
       (a) $S_i^1 = I_i^1 \odot I_i^3$
       (b) $S_i^2 = I_i^1 \oplus 1$
       (c) $S_i^3 = I_i^2 \odot I_i^4$
       (d) $S_i^4 = I_i^2 \oplus 1$
    2. P-Box Computation.
       (a) $PB_i^1 = S_i^1[1]||S_i^2[4]||S_i^1[2]||S_i^2[3]$
       (b) $PB_i^2 = S_i^3[3]||S_i^2[2]||S_i^1[4]||S_i^2[1]$
       (c) $PB_i^3 = S_i^3[1]||S_i^4[4]||S_i^3[2]||S_i^4[3]$
       (d) $PB_i^4 = S_i^3[3]||S_i^4[2]||S_i^3[4]||S_i^4[1]$
    3. L-Box Computation
       (a) $P_i[1] = (PB_i^1[1] \oplus PB_i^2[4]); Q_i[1] = (PB_i^1[1] \odot 0)$
       (b) $P_i[2] = (PB_i^1[2] \oplus PB_i^2[3]); Q_i[2] = (PB_i^1[2] \oplus 1)$
       (c) $P_i[3] = (PB_i^1[3] \oplus PB_i^2[2]); Q_i[3] = (PB_i^1[3] \odot 0)$
       (d) $P_i[4] = (PB_i^1[4] \oplus PB_i^2[1]); Q_i[4] = (PB_i^1[4] \oplus 1)$
       (e) $P_i[5] = (PB_i^3[1] \oplus PB_i^4[4]); Q_i[5] = (PB_i^2[1] \odot 0)$
       (f) $P_i[6] = (PB_i^3[2] \oplus PB_i^4[3]); Q_i[6] = (PB_i^2[2] \oplus 1)$
       (g) $P_i[7] = (PB_i^3[3] \oplus PB_i^4[2]); Q_i[7] = (PB_i^2[3] \odot 0)$
       (h) $P_i[8] = (PB_i^3[4] \oplus PB_i^4[1]); Q_i[8] = (PB_i^2[4] \oplus 1)$
    4. $L_i(1) = P_i[1]||Q_i[4]||P_i[2]||Q_i[3]||P_i[3]||Q_i[2]||P_i[4]||Q_i[1]$
    5. $L_i(2) = P_i[5]||Q_i[8]||P_i[6]||Q_i[7]||P_i[7]||Q_i[6]||P_i[8]||Q_i[5]$
    6. $Z = L_i(1)||L_i(2)$
```

The resulting intermediate cipher text (ICT) is then sent through a function as shown in the Algorithm.1, in which the first and third four-bit blocks of the ICT are operated on. This function includes an S-box, P-box, and L-box, which are composed of a series of operations such as XOR, XNOR, and concatenation.

3.1. Security analysis of LRBC scheme

3.1.1. Key-Related Attacks

A key-related attack is a cyber attack that specifically targets the cryptographic keys used for secure communication, with the intention of compromising the confidentiality, integrity, or authenticity of the key. These attacks can subsequently be exploited to gain unauthorized access to sensitive information or impersonate legitimate users. In LRBC cipher they propose a key scheduling design that permutes the round keys k^a, k^b, k^c, and k^d in a manner that makes any one of the round keys vulnerable to revealing the original key. With only $2^4 = 16$ possible combinations, even an exhaustive search algorithm can easily compromise the scheme with present computing power, requiring a search of just 2^{16} combinations.

3.1.2. Known Plaintext Attacks

A known plaintext attack is a cryptanalytic attack in which an attacker has access to both the ciphertext (encrypted message) and the corresponding plaintext (original message). By analyzing the relationship between the plaintext and the ciphertext, the attacker can determine the encryption key and subsequently use it to decrypt other messages encrypted with the same key. This type of attack is typically used against symmetric encryption algorithms, where the same key is used for both encryption and decryption.

In the case of the LRBC scheme, attacking it using a known plaintext attack only requires the plaintext itself. This is because we can observe that in the LRBC cipher algorithm, the plaintext is directly XOR/XNORed with the initial key, and this operation can be repeated with the resulting output to obtain the key itself, as shown in Example 1

Example 1. Let's take,
Plaint-text = 0x00000000
$k^a = 0110$, $k^b = 1011$, $k^c = 1110$, $k^d = 0111$
After randomizing the plaintext and dividing it into 4 sub-parts,
$PT^1 = 0000$, $PT^2 = 0000$, $PT^3 = 0000$, $PT^4 = 0000$
As per in the initial round of the LRBC scheme,
$PT^1 \odot k^a = 0000 \odot 0110 = 1001$
$PT^2 \oplus k^b = 0000 \oplus 1011 = 1011$
$PT^3 \oplus k^c = 0000 \oplus 1110 = 1110$
$PT^4 \odot k^d = 0000 \odot 0111 = 1000$
So, the intermediate cipher text is = 1001101111101000
If we just negate the 1st 4bits and last 4bits then we get,

ICT = 0110101111100111 which is equal to the key. This means we are simply getting the key by taking the plaintext as all 0s or all 1s.

3.2. Hardware Analysis

In this section, we analyze the hardware requirements of the LRBC algorithm in terms of gate equivalents. To estimate the total number of gate equivalents that the algorithm occupies, we assume the number of gate equivalents per operation and use the following equation:

$$S_{total} = 39S_{concat} + 24S_{xor} + S_{DR} + S_{KSR} + 4S_{mux}. \tag{1}$$

We assume that the number of gate equivalents (GE) for the concatenate operation (S_{concat}), exclusive OR/ exclusive NOR operation (S_{xor}), data registers (S_{DR}), key-store-registers (S_{KSR}), and multiplexers (S_{mux}) are 3, 7, 80, 80, and 1, respectively. Using these assumptions, the total number of gate equivalents required for the system is calculated as follows:

$$S_{TotalOperations} = 39 \times 3 + 24 \times 7 + 80 + 80 + 4 \times 1 \tag{2}$$

$$S_{TotalOperations} = 117 + 168 + 80 + 80 + 4 \tag{3}$$

$$S_{TotalOperations} = 449. \tag{4}$$

Based on these assumptions, the calculated total gate equivalents for the algorithm are 449, which will be further analyzed with the proposed scheme.

3.3. LRBC Cipher Weakness

The LRBC cipher employs only XOR/XNOR operations, which are linear operations. Linear operations pose a security risk as they compromise the confidentiality and integrity of the data being encrypted. The primary disadvantage of using only XOR/XNOR operations in designing block ciphers is limited diffusion and confusion. XOR/XNOR operations provide only a limited degree of diffusion, meaning that small changes in the plaintext may result in limited changes in the ciphertext. This can make the cipher vulnerable to certain types of attacks, such as differential cryptanalysis. Additionally, XOR/XNOR operations alone do not provide adequate confusion, which means the relationship between the key and the ciphertext is not sufficiently obscured. This can make the cipher vulnerable to certain types of attacks, such as linear cryptanalysis.

To mitigate these limitations, XOR/XNOR operations are typically used in combination with other operations in the design of block ciphers. It is well known that XOR/XNOR operations are linear operations, which means they do not provide sufficient security. This weakness of XOR/XNOR operations has been demonstrated in previous research.

Here is a theorem to prove that the XOR operation is linear: The XOR operation is a linear operation.

To show that the XOR operation ($\oplus$) is linear, we need to prove that it satisfies the following two properties:

1. Closure under addition: For any binary vectors $\mathbf{a}$ and $\mathbf{b}$, $\mathbf{a} \oplus \mathbf{b}$ is also a binary vector.
2. Homogeneity: For any binary vector $\mathbf{a}$ and scalar c, $c \cdot (\mathbf{a} \oplus \mathbf{b}) = c \cdot \mathbf{a} \oplus c \cdot \mathbf{b}$.

To show that $\oplus$ is closure under addition, we use Boolean algebra to demonstrate that the XOR operation satisfies the following properties:

- Commutativity: $\mathbf{a} \oplus \mathbf{b} = \mathbf{b} \oplus \mathbf{a}$
- Associativity: $(\mathbf{a} \oplus \mathbf{b}) \oplus \mathbf{c} = \mathbf{a} \oplus (\mathbf{b} \oplus \mathbf{c})$
- Identity: $\mathbf{a} \oplus \mathbf{0} = \mathbf{a}$
- Inverse: $\mathbf{a} \oplus \mathbf{a} = \mathbf{0}$

These properties, along with the distributive property of $\oplus$ over $\wedge$, demonstrate that $\oplus$ is a binary operation that is closed under addition.

To show that $\oplus$ is homogeneous, we use the fact that $c \cdot \mathbf{0} = \mathbf{0}$ and $c \cdot \mathbf{1} = c$ for any scalar c, and demonstrate that:

$$c \cdot (\mathbf{a} \oplus \mathbf{b}) = c \cdot \mathbf{a} \oplus c \cdot \mathbf{b}$$

This completes the proof that $\oplus$ is a linear operation.

4. Proposed Scheme

4.1. Overview of the scheme

The proposed scheme is truly a new way of a block cipher as depicted in Fig.2 and has a stronger key scheduling scheme with the use of a newly introduced function, which transforms 32bit key to 16bit key. In any cryptographic algorithm, it is necessary that key management and key should be the important element, because if one is having the key then it is a matter of seconds to decode any cryptographic algorithm. So, we must be very much concerned with the key scheduling and key whitening in block ciphers. So, in this regard, we chose key scheduling to be very strong. In Fig.2, we can see the diagram of the whole design. We have used two new concepts in the design. One is using key scheduling and the other is a new way of designing an S-Box using a chaotic equation. Designing of S-Box algorithm can be seen in Algorithm.2 . Basically, the whole design consists of key management, S-Box and P-Box. S-Box has been compared and proved to be as strong as the Present cipher's S-Box and also compared with the LRBC cipher's S-Box it is proved that the avalanche effect of our S-Box is more than the LRBC cipher's S-Box. All these results are shown in the result and analysis part. And the encryption scheme can be seen in Algorithm.3

4.2. Key Scheduling

The design of the key scheduling is very unique compared to the other existing lightweight algorithms. The function is illustrated below. The function is used for the initial round as a key whitening technique, also to give more security to the scheme and to overcome the related key attacks. The strength of this key scheduling is shown in the result and analysis section.

The Key Management is a process of converting a 32-bit input into a 16-bit output. The algorithm consists of two parts: the Initial Round function and the Round Key generation process.

The Initial Round function takes the 32-bit input and transforms it into a 16-bit key. The function performs the following steps:

1. Computes "First" (denoted by $\sim$) to the first 4 bits of the 32-bit input and keeps remaining bits as it is: [First = $\sim$(first 4 bits of 32-bit input)]
2. Computes "Second" by applying the bitwise NOT operation to the third 4 bits of the 32-bit input and keeps remaining bits as it is: [Second = $\sim$(third 4 bits of 32-bit input)]
3. Computes "Third" by applying the bitwise NOT operation to the fifth 4 bits of the 32-bit input and keeps remaining bits as it is: [Third = $\sim$(fifth 4 bits of 32-bit input)]
4. Computes "Fourth" by applying the bitwise NOT operation to the seventh 4 bits of the 32-bit input and keeps remaining bits as it is: [Fourth = $\sim$(seventh 4 bits of 32-bit input)]
5. Calculates the "Final-key" by performing a bitwise XOR operation (denoted by $\oplus$) on the results of two other bitwise XOR operations. The first XOR operation is performed on the odd-numbered bits of "First" and the even-numbered bits of "Second", while the second XOR operation is performed on the odd-numbered bits of "Third" and the even-numbered bits of "Fourth": [Final-key = (odd-numbered bits of First $\oplus$ even-numbered bits of Second) $\oplus$ (odd-numbered bits of Third $\oplus$ even-numbered bits of Fourth)]
6. Returns the "Final-key".

The Round Key generation process uses the output of the Initial Round function to generate

16 round keys. And in each round the key will be further divided into 4bits each as part of the encryption process. It performs the following steps:

1. Sets "Final-key" equal to the output of the Initial Round function.
2. Repeats the following step 16 times:
    (a) Applies the P-Box operation to "Final-key": [Final-key = P-Box(Final-key)]

4.3. S-Box Generation

S-Boxes play a critical role in modern symmetric-key cryptographic algorithms, such as the Advanced Encryption Standard (AES). They introduce confusion and non-linearity, enhancing the overall security of the encryption process. Traditional S-Box generation methods rely on algebraic constructions, such as lookup tables or mathematical operations. In this work, we present a novel approach to S-Box generation using chaotic systems, specifically the logistic map.

The logistic map is a simple non-linear dynamic equation given by:

$$x_{n+1} = r \cdot x_n(1 - x_n) \tag{5}$$

where x represents the population, and r represents the growth rate. The behavior of the logistic map can vary significantly based on the value of r. For values greater than 3.5 and within the range $0 < x < 1$, the equation exhibits chaotic behavior. This chaotic nature makes the logistic map a promising candidate for generating S-Boxes with enhanced cryptographic properties.

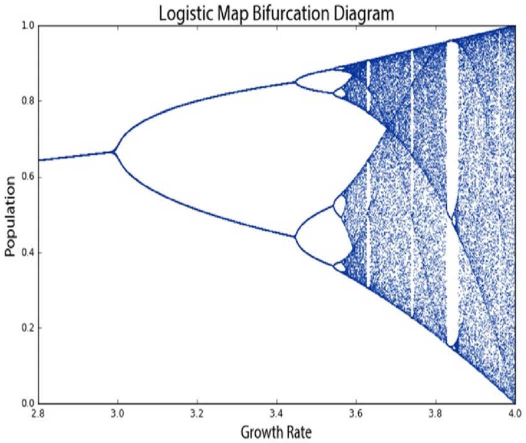

Figure 1: Behavior of the logistic map

Algorithm 2: S-Box Generation Algorithm

```
Data: $0 < x < 1$, $3.5699 < r < 4$ Result: Chaotic numbers of $x$ (S-Box)
    1. Initialize parameters $N$, $x_0$, and a list to store the generated numbers.
    2. For $n$ in the range 0 to $N - 1$:
        (a) Calculate $x_{n+1}$ using the logistic map equation $x_{n+1} = r \cdot x_n(1 - x_n)$.
        (b) Store $x_{n+1}$ in the list.
        (c) Set $x_n = x_{n+1}$.
    3. Sort the list of generated numbers.
    4. Initialize a counter to 0.
    5. While the counter is less than $N$:
        (a) Assign the counter as the serial number for the number in the current position of the list.
        (b) Increment the counter by 1.
        (c) Move to the next position in the list.
    6. The final table with serial numbers as indices and sorted random numbers as values will be
       the S-Box.
```

The S-Box generation algorithm, shown in Algorithm 2, utilizes the logistic map to produce chaotic numbers that are then used to construct the S-Box.

The S-Box generation process starts by iterating the logistic map equation with a random initial value x_0 and the growth rate r within the chaotic range. The resulting sequence of chaotic numbers is then sorted, and each number is assigned a serial number. The sorted sequence, with serial numbers as indices, forms the final S-Box.

The S-Box generated using this method exhibits strong cryptographic properties due to the chaotic behavior of the logistic map. The inherent unpredictability and non-linearity of chaotic systems contribute to the diffusion and confusion properties required for a robust cryptographic S-Box.

4.3.1. Practical Use Case

The chaotic S-Box generated through this method finds application in various cryptographic algorithms, particularly those requiring strong data encryption. For instance, it can be integrated into block ciphers like AES or used in stream ciphers to enhance their security. The non-linearity introduced by the chaotic S-Box further fortifies the resistance against various attacks, such as differential cryptanalysis and linear cryptanalysis

4.4. Encryption and Decryption Process

The proposed scheme's encryption process is presented in Algorithm 3. It involves several steps, beginning with pre-whitening the 32-bit key using a special function outlined in the key scheduling section. The output of this function is a 16-bit key, which is XORed with the plaintext. The resulting intermediate ciphertext is then XORed with the round-1 key and passed through the S-Box, followed by the P-Box and block permutation. This process is repeated for 16 rounds to generate the final ciphertext. A block diagram of the proposed scheme is depicted in Fig.2. Overall, the proposed encryption process is straightforward.

The decryption process is the same as the encryption process but in reverse operations.

```
Algorithm 3: Encryption Algorithm
  Data: 16-bit Plain-text (PT), 32-bit Key (key-32)
  Result: Cipher-text of the Plain-text
    1. Key of 32-bit will be processed using the function Processed-key.
        (a) Key = Processed-key(key-32)
    2. Key Whitening.
        (a) $ICT_1$ = PT $\oplus$ key
    3. Initial round.
        (a) Divide $ICT_1$ into 4 sub-parts of each 4-bit.
        (b) Divide round key-1 into 4 sub-parts of each 4-bits.
        (c) XOR operation with key.
              i. $ICT_{1a} = ICT_{1a} \oplus k_a$
             ii. $ICT_{1b} = ICT_{1b} \oplus k_b$
            iii. $ICT_{1c} = ICT_{1c} \oplus k_c$
             iv. $ICT_{1d} = ICT_{1d} \oplus k_d$
    4. S-Box calculation.
        (a) $ICT_{1a}$ = S-Box($ICT_{1a}$)
        (b) $ICT_{1b}$ = S-Box($ICT_{1b}$)
        (c) $ICT_{1c}$ = S-Box($ICT_{1c}$)
        (d) $ICT_{1d}$ = S-Box($ICT_{1d}$)
    5. P-Box calculation.
        (a) $ICT_1$ = P-Box($ICT_1$)
    6. Block-Permutation.
        (a) $ICT_{1a} = ICT_{1b}$
        (b) $ICT_{1b} = ICT_{1a}$
        (c) $ICT_{1c} = ICT_{1d}$
        (d) $ICT_{1d} = ICT_{1c}$
    7. Repeat the steps from Initial round to Block-Permutation for more 15 rounds.
```

5. Results and analysis

5.1. Related key attack

Related key attacks[22] are that where the relationship between keys while managing the keys in round functions can reveal the key itself. In the LRBC scheme, the key is managed by simply block

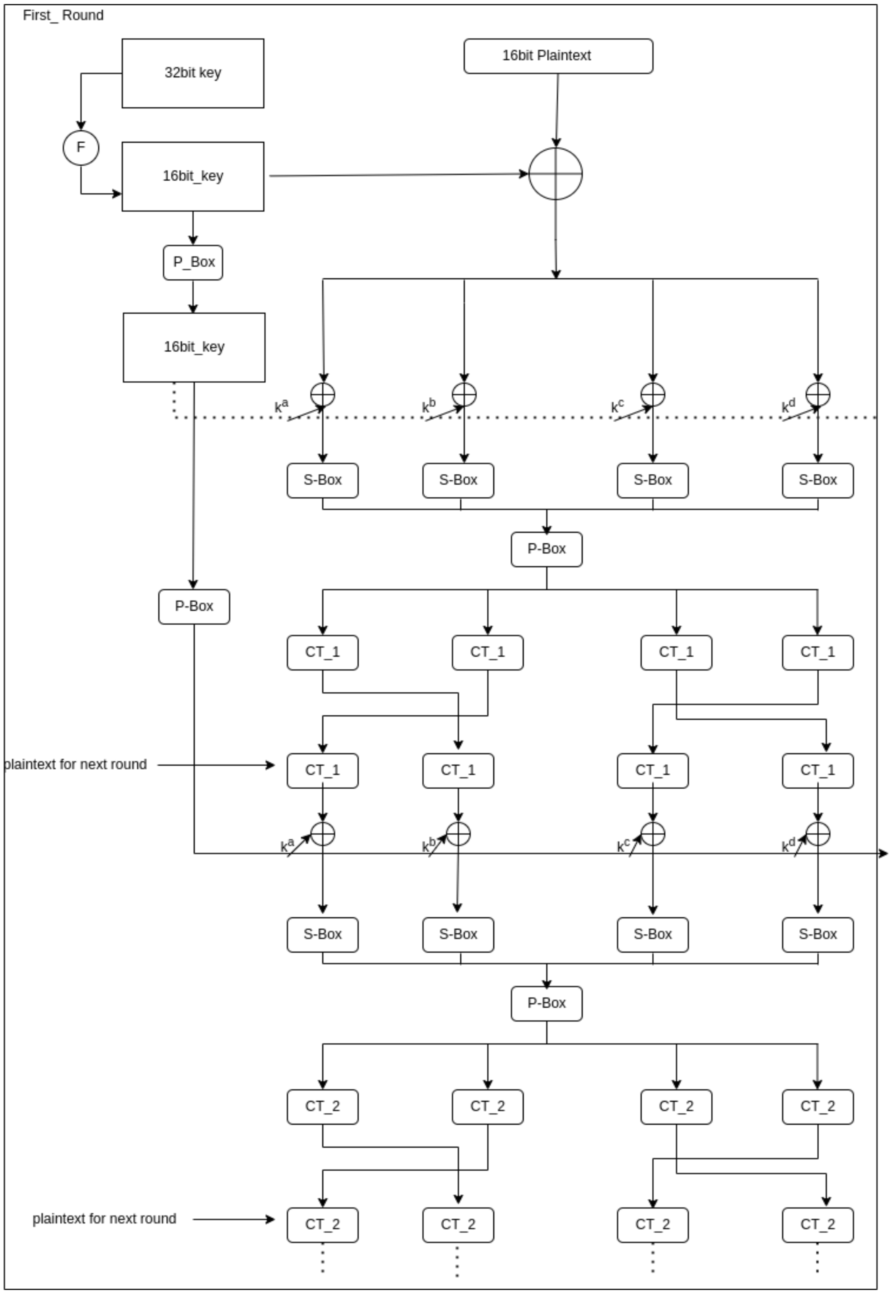

Figure 2: Encryption Process

permuting the master key. As the scheme has 16bit keys and divides the key into 4 parts, we have 4 blocks of keys and we get 16 number of permutations for a total of 16 rounds, which is an easy relation to get attacked. Before knowing how the LRBC scheme is prone to related-key attacks, The LRBC scheme is very much attackable in the initial stage only because key whitening has not been done in the scheme. And just the key is XORed with the plaintext directly, if an attacker just puts all zeroes in the plain text then we get the key that is used as shown in the Eg.1.

We have tested our proposed key scheduling scheme, and the results are very much robust towards key-related attacks. The results are brought by fixing the plain text as 0xffff and the key as 0xaa55aa55, we have calculated the hamming distance(D_{hamm}) of each round key, master key and plaintext, each round cipher text and then calculated the difference between these two D_{hamm}. So, till we reach 9 rounds of the algorithm we didn't get the zero as the difference, which means that at least for 9 rounds the proposed key scheduling scheme is robust towards key attacks. The results are shown in the Table.2

Round Number	1	2	3	4	5	6	7	8	9	10
$D_{hamm}(K, K_r)$	8	10	8	6	8	10	4	4	8	10
$D_{hamm}(P, C_r)$	6	4	7	7	12	9	8	10	9	7
Δ	2	6	1	1	4	1	4	6	1	3

Table 2: Related- key testing table

5.2. Avalanche effect

The Avalanche effect is another property which is important property in cryptographic algorithms because it defines that a small change in the input should reflect a larger difference in the output. We can also recognise if the algorithm is having any nonlinear properties by testing the percentage of the avalanche effect in the algorithm. A block cipher or cryptographic hash function has inadequate randomization if the avalanche effect is not present to a substantial extent. In this case, a cryptanalyst may anticipate the input using only the output. The algorithm might be broken in part or entirely by this. Therefore, from the perspective of the creator of the cryptographic algorithm or device, the avalanche effect is a desirable circumstance. The analysis has been plotted in the graph shown in Fig.3

Change in Bits	Proposed	LRBC	Twine	Present	Lblock	SLIM
1	53.75%	0%	45.6	52.5	49.5	45.6
2	61.25%	0%	47.5	56.4	48.6	43.5
3	52.5%	43.75%	52.6	52.6	49.2	48.6
4	63.75%	50%	52.8	52.8	48.9	49.1
5	45%	43.75%	49.6	49.6	47.6	52.8

Table 3: Avalanche effect

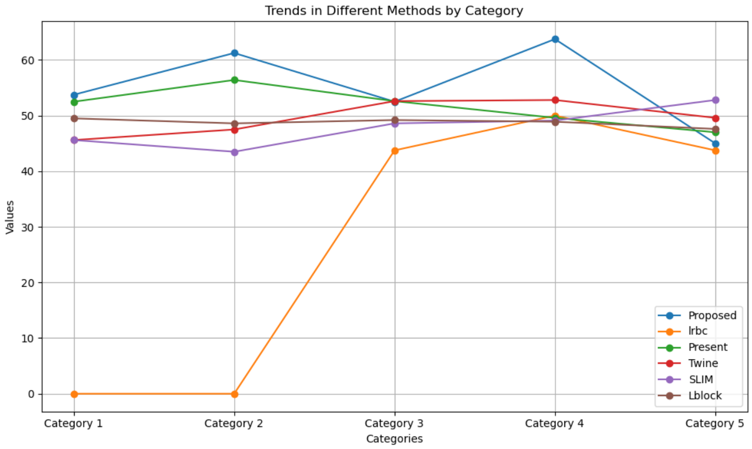

Figure 3: comparison of the avalanche effect

5.3. Encryption time

Since the algorithm is intended for lightweight devices, we regarded encryption time as a critical factor in designing the algorithm. We analyzed the encryption time of the proposed scheme and obtained significantly improved results compared to those of the LRBC scheme. These results are presented in Fig. 4, providing assurance of the scheme's efficiency.

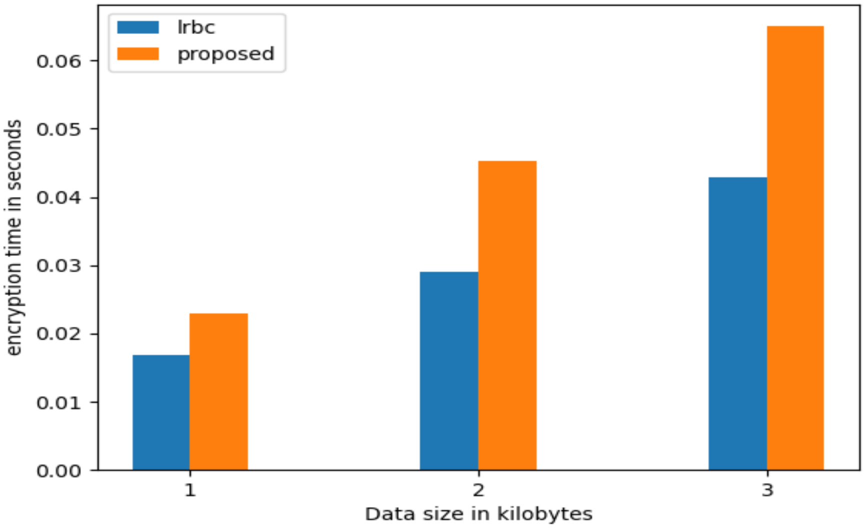

Figure 4: comparison of the speeds

5.4. S-Box analysis

As we have seen the generation of the s-box. The S-Box we are using for the cipher has been tested over PRESENT cipher's S-Box, the results are very promising as it is observed that the confusion coefficient variance value is more than the PRESENT cipher's S-Box and also B-Flex-γ's(a block cipher) S-Box. The results are shown in the Table.4.

The Confusion Coefficient Variance(CCV) is a metric that any ideal S-Box must hold higher value, so that it gets immune to side channel attacks. Because higher value of the confusion coefficient variance gives better resistance from side channel attacks. In Table.4 we showed that our proposed scheme holds the highest value even compared to the standard present cipher's S-Box. Confusion Coefficient Variance used to calculate as follows,

$$K(k_a, k_b) = E[L(F(k_a \oplus P)) - L(F(k_b \oplus P)))^2] \tag{6}$$

where, k_a, k_b are the sub keys and p stands for arbitrary inputs, L stands for the leakage function, and E is the mean to be calculated. Additionally, a larger confusion coefficient variance (CCV) value in the S-Box results in a stronger resistance to SCAs. Formally, the value of CCV of an S-Box is determined as follows for all key pairings k_a, k_b, and $k_a = k_b$:

$$CCV(F) = var(E[H(F(k_a \oplus P)) - H(F(k_b \oplus P)))^2]) \tag{7}$$

This means the cipher is very much robust towards side channel attacks.

Scheme	Confusion coefficient variance
proposed scheme	0.757222.
PRESENT	0.657222.
BFlex-γ	0.382222.
Lblock0	0.207222.
Lblock1	0.257222.
Lblock2	0.257222.
Twine	0.357222.

Table 4: Confusion Coefficient Variance

5.5. Linear Cryptanalysis

Linear cryptanalysis is basically to find high probability instances of linear expressions utilising subkey bits, plaintext bits, and "ciphertext" bits. It is a known plaintext attack, meaning the attacker must be aware of a set of plaintexts and the accompanying ciphertexts in order for it to succeed. The available plaintexts (and related ciphertexts) cannot be chosen by the attacker. It is plausible to suppose that the attacker is aware of a random set of plaintexts and the matching ciphertexts in many circumstances and applications. The fundamental concept is to approximate a section of the cipher's operation with a linear expression where the linearity refers to a mod-2 bit-wise operation (exclusive-OR, indicated by " $\oplus$ "). Such an expression has the following form:

$$P_{a_1} \oplus P_{a_2} \oplus ....P_{a_x} \oplus Q_{b_1} \oplus Q_{b_2} \oplus ....Q_{b_y} = 0 \tag{8}$$

the a^{th} bit of the input P = [P_1, P_2,...] is represented by P_i, while the b^{th} bit of the output Q = [Q_1, Q_2,...] is represented by Q_b. The "sum" of the exclusive-OR of the x input bits and the y output bits is represented by the Eq.8

Determining expressions of the aforementioned kind that have a high or low likelihood of occurring is the strategy used in linear cryptanalysis. (No evident linearity like the one described above shouldn't apply for all input and output values; otherwise, the cipher would be incredibly vulnerable.) It is a sign of weak randomization skills when a cipher has a tendency for Eq.8, to hold with high probability or not hold with high probability. Consider that the likelihood that the statement would hold is exactly 1/2 if we randomly choose values for the x + y bits and enter them into the Eq.8. In linear cryptanalysis, the departure or bias from the likelihood of 1/2 for an expression to hold is what is exploited; the farther distant a linear expression is from the probability of 1/2, the more effectively the cryptanalyst can use linear cryptanalysis. The degree by which the probability of a linear expression holding deviates from 1/2 is referred to as the linear probability bias in the remaining sections of the paper. Therefore, the probability bias is $p_L - 1/2$ if the expression above holds with probability p_L for randomly selected plaintexts and the accompanying ciphertexts. With fewer known plaintexts needed in the attack, linear cryptanalysis is more applicable the larger the probability bias, $|pL - 1/2|$

The number of matches between the linear equation denoted by "Input Sum" and the sum of the output bits denoted by "Output Sum" minus 8 is represented by each entry in the table. To determine the probability bias for a specific linear combination of input and output bits, divide an element value by 16. When seen as a binary number, the hexadecimal value of a sum identifies the variables that went into the sum. When seen as a binary number, the hexadecimal value of a sum identifies the variables that went into the sum. The hexadecimal value reflects the binary value $x_1x_2x_3x_4$, where x_1 is the most significant bit, for a linear combination of input variables represented as $x_1.A_1 \oplus x_2.A_2 \oplus x_3.A_3 \oplus x_4.A_4$ where $x_i \in$ 0,1 and "." indicates binary AND. Similar to this, the hexadecimal value represents the binary vector $y_1y_2y_3y_4$ for a linear combination of output bits $y_1.B_1 \oplus y_2.B_2 \oplus y_3.B_3 \oplus y_4.B_4$ where $y_i \in$ 0,1. The bias of the linear equation $A_3 \oplus A_4 = B_1 \oplus B_4$ (hex input 3 and hex output 9) is -2/16 = -1/8, and the likelihood that the correct linear equation (1/2)-(1/8)= 3/8 = 0.375. It is observed that the Linear Approximation table is as similar to present cipher's LAT values so, it is as strong as Present cipher.

5.6. Differential Cryptanalysis

Differential cryptanalysis works by taking the changes in the plaintext and records the value changed in ciphertext[28], the probability that the 1 bit change and 2 bit change and so on. Let's Consider a system. For example, the inputs for the system are $[A_1A_2A_3...A_n]$ and the outputs are $[B_1B_2...B_n]$, Say the the considered system has two inputs, A_v and A_w, and two outputs, B_v and B_w, respectively, calculating the XOR of taken inputs and XOR of respective outputs is the input difference and output difference respectively.

$$\Delta A = [\Delta A_1 \Delta A_2 .... \Delta A_n] \tag{9}$$

$$\Delta B = [\Delta B_1 \Delta B_2 .... \Delta B_n] \tag{10}$$

Where, ΔA_1 is $A_v \oplus A_w$

Theoretically an ideal cipher holds the probability of $1/2^n$, where n is represented as the number of bits in A. 1/2n, it is the probability that a specific output difference Δ B will occur given a specific input difference Δ A. The probability of input difference of 5 and the output difference of 5 is 2/16 = 1/8 = 0.125, the highest probability of any cell in the table is 4/16 = 1/4 = 0.25, which is as equal as the standard present cipher.

5.7. Strength of the S-Box

We compared our S-Box with the Present[6] S-Box and some other well known standard lightweight ciphers. Our S-Box is giving the best result with respect to some properties and almost equal in some of the other properties that should hold for any S-Box. The analysis is shown in Table.5, and the plotted graph is shwon in Fig.5

Sbox/property	Absolute Indicator	SNR	Sum of Square Indicator	Composite Algebraic Immunity
Present	16	2.024	1024	2
Lblock0	16	2.008	1024	2
Lblock1	16	2.018	1024	2
Lblock2	16	2.0152	1024	2
twine	16	2.014	640	2
Proposed S-box	16	2.024	1408	2

Table 5: S-Box Property Analysis

From the Table.5, it is observed that for any property that is mentioned in the table our proposed S-Box is showing the better result or at-least the same result. This means our proposed S-Box is secure and efficient for lightweight devices. And if we observe the signal to noise ratio this metric is showing the highest value for our proposed scheme, The more Signal to Noise Ratio(SNR) value the more immune to the side channel attacks.

5.8. Hardware analysis compared to LRBC Scheme

As previously analyzed in the section.3.1 of the LRBC scheme, it was found that it requires 449 Gate Equivalents (GE) to implement. The proposed scheme can be represented by the following equation, describing its operations:

$$S_{total} = 4S_{NotOperation} + 4S_{xor} + S_{DR} + S_{KSR} + 4S_{SBox} + S_{PBOX} + 4S_{replace} \tag{11}$$

We assume that the number of gate equivalents(GE) for the Not operation($S_{NotOpearation}$), XOR operation(S_{xor}), Data Registers(S_{DR}), Key-Store-Registers(S_{KSR}, SBox(S_{SBox}), PBoxS_{PBox} and replace operation$S_{replace}$ are 2, 7, 80, 80, 4, 4 and 1 respectively. Then, the total number of GEs for this system will be,

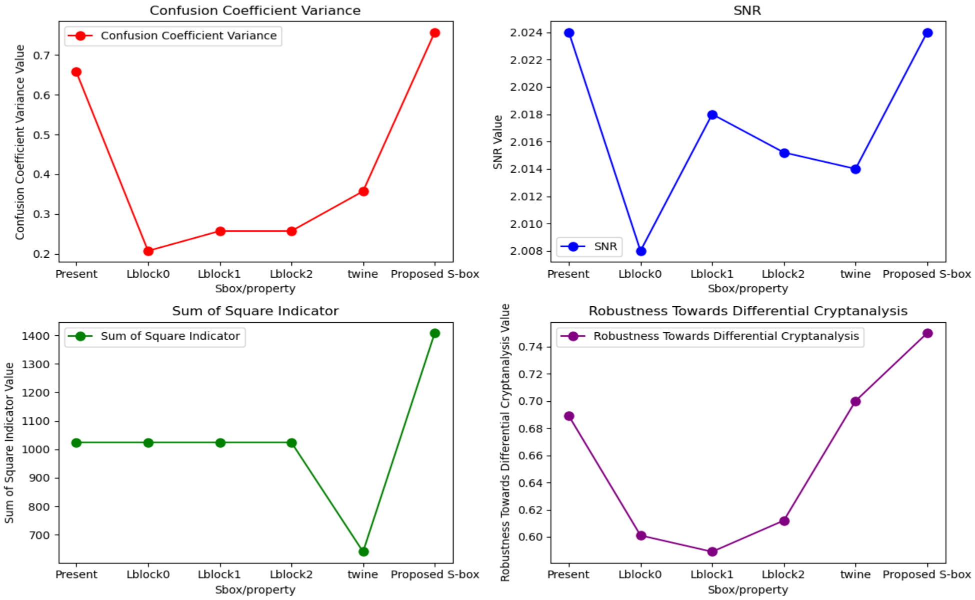

Figure 5: Comparison of Proposed S-Box with Some Popular Existing S-Boxes

$$S_{TotalOperations} = 4*2 + 4*7 + 80 + 80 + 4*4 + 4 + 4*1 \tag{12}$$

$$S_{TotalOperations} = 8 + 28 + 160 + 16 + 4 + 4 \tag{13}$$

$$S_{TotalOperations} = 220. \tag{14}$$

So, compared to the LRBC scheme, our proposed scheme is good in number of GE's.

6. Conclusion and Future work

In conclusion, the scheme proposed in this paper provides a secure and cost-efficient solution for lightweight devices and systems to ensure the protection of information. The scheme is resistant to basic to advanced attacks, and it guarantees the confidentiality and integrity of the data. As technology continues to evolve, it is essential to ensure that the security measures keep pace with the new threats. Therefore, it is crucial to continue exploring and enhancing the security aspects of lightweight systems to provide a more robust solution.

Future research should continue to advance and improve the security measures to meet the growing demands of the technology. One of the potential future works is to implement the proposed scheme in hardware. This would allow for a real-world test and evaluation of the scheme's practicality, performance, and security. Additionally, future research could focus on expanding the scheme to address more advanced attacks, such as side-channel attacks, and to investigate its scalability to support larger-scale deployments.

References

[1] H. Noura, O. Salman, R. Couturier, A. Chehab, Lesca: Lightweight stream cipher algorithm for emerging systems, Ad Hoc Networks 138 (2023) 102999.

[2] W. Chen, L. Li, Y. Guo, Y. Huang, Sand-2: An optimized implementation of lightweight block cipher, Integration (2023).

[3] V. Panchami, M. M. Mathews, A substitution box for lightweight ciphers to secure internet of things, Journal of King Saud University-Computer and Information Sciences (2023).

[4] R. Mishra, M. Okade, K. Mahapatra, Novel substitution box architectural synthesis for lightweight block ciphers, IEEE Embedded Systems Letters (2023).

[5] M. Rana, Q. Mamun, R. Islam, Lightweight cryptography in iot networks: A survey, Future Generation Computer Systems 129 (2022) 77–89.

[6] V. A. Thakor, M. A. Razzaque, M. R. A. Khandaker, Lightweight cryptography algorithms for resource-constrained iot devices: A review, comparison and research opportunities, IEEE Access 9 (2021) 28177–28193. `doi:10.1109/ACCESS.2021.3052867`.

[7] V. Rao, K. Prema, A review on lightweight cryptography for internet-of-things based applications, Journal of Ambient Intelligence and Humanized Computing 12 (9) (2021) 8835–8857.

[8] A. Biswas, A. Majumdar, S. Nath, A. Dutta, K. Baishnab, Lrbc: a lightweight block cipher design for resource constrained iot devices, Journal of Ambient Intelligence and Humanized Computing (2020) 1–15.

[9] S. S. Dhanda, B. Singh, P. Jindal, Lightweight cryptography: a solution to secure iot, Wireless Personal Communications 112 (3) (2020) 1947–1980.

[10] K. Chatterjee, R. R. K. Chaudhary, A. Singh, A lightweight block cipher technique for iot based e-healthcare system security, Multimedia Tools and Applications (2022) 1–30.

[11] B. J. Mohd, T. Hayajneh, Lightweight block ciphers for iot: Energy optimization and survivability techniques, IEEE Access 6 (2018) 35966–35978. `doi:10.1109/ACCESS.2018.2848586`.

[12] L. Li, B. Liu, Y. Zhou, Y. Zou, Sfn: A new lightweight block cipher, Microprocessors and Microsystems 60 (2018) 138–150.

[13] B. Seok, C. Lee, Fast implementations of arx-based lightweight block ciphers (sparx, cham) on 32-bit processor, International Journal of Distributed Sensor Networks 15 (9) (2019) 1550147719874180.

[14] N. Mouha, Arx-based cryptography, Online: https://www. cosic. esat. kuleuven. be/ecrypt/courses/albena11/slides/nicky_mouha_arx-slides. pdf (2011).

[15] M. A. F. Al-Husainy, B. Al-Shargabi, S. Aljawarneh, Lightweight cryptography system for iot devices using dna, Computers & Electrical Engineering 95 (2021) 107418.

[16] C. H. Lim, T. Korkishko, mcrypton–a lightweight block cipher for security of low-cost rfid tags and sensors, in: International workshop on information security applications, Springer, 2005, pp. 243–258.

[17] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. Robshaw, Y. Seurin, C. Vikkelsoe, Present: An ultra-lightweight block cipher, in: International workshop on cryptographic hardware and embedded systems, Springer, 2007, pp. 450–466.

[18] D. Hong, J.-K. Lee, D.-C. Kim, D. Kwon, K. H. Ryu, D.-G. Lee, Lea: A 128-bit block cipher for fast encryption on common processors, in: International Workshop on Information Security Applications, Springer, 2013, pp. 3–27.

[19] Z. Gong, S. Nikova, Y. W. Law, Klein: a new family of lightweight block ciphers, in: International Workshop on Radio Frequency Identification: Security and Privacy Issues, Springer, 2011, pp. 1–18.

[20] B. Aboushosha, R. A. Ramadan, A. D. Dwivedi, A. El-Sayed, M. M. Dessouky, Slim: a lightweight block cipher for internet of health things, IEEE Access 8 (2020) 203747–203757.

[21] A. Ragab, A. Madani, A. Wahdan, G. Selim, Design, analysis, and implementation of a new lightweight block cipher for protecting iot smart devices, Journal of Ambient Intelligence and Humanized Computing (2021) 1–18`doi:10.1007/s12652-020-02782-6`.

[22] A. K. Das, N. Kar, S. Deb, M. Singh, bflex-\ $/gamma$ $/gamma$: A lightweight block cipher utilizing key cross approach via probability density function, Arabian Journal for Science and Engineering (2022) 1–16.

[23] A. Castiglione, F. Palmieri, F. Colace, M. Lombardi, D. Santaniello, G. D'Aniello, Securing the internet of vehicles through lightweight block ciphers, Pattern Recognition Letters 135 (2020) 264–270.

[24] D. Dinu, Y. L. Corre, D. Khovratovich, L. Perrin, J. Großschädl, A. Biryukov, Triathlon of lightweight block ciphers for the internet of things, Journal of Cryptographic Engineering 9 (3) (2019) 283–302.

[25] W. Wu, L. Zhang, Lblock: a lightweight block cipher, in: International conference on applied cryptography and network security, Springer, 2011, pp. 327–344.

[26] L. Sliman, T. Omrani, Z. Tari, A. E. Samhat, R. Rhouma, Towards an ultra lightweight block ciphers for internet of things, Journal of information security and applications 61 (2021) 102897.

[27] J. Borghoff, A. Canteaut, T. Güneysu, E. B. Kavun, M. Knezevic, L. R. Knudsen, G. Leander, V. Nikov, C. Paar, C. Rechberger, et al., Prince–a low-latency block cipher for pervasive computing applications, in: International conference on the theory and application of cryptology and information security, Springer, 2012, pp. 208–225.

[28] J. S. Teh, L. J. Tham, N. Jamil, W.-S. Yap, New differential cryptanalysis results for the lightweight block cipher boron, Journal of Information Security and Applications 66 (2022) 103129.