CHAPTER 4

DESIGN AND EVALUATION OF SIC-CHAIN SECURITY LAYER

4.1 INTRODUCTION

The primary contribution of this chapter includes proposing an algorithm for generating 16*16 S-Boxes utilizing Chaotic Logistic Maps, which is analysed via performance parameters like Non-linearity, Differential and Linear approximation probability. In static S-Boxes pre-defined substitution tables are fixed during the encryption or decryption process. Static S-Boxes are fixed in nature and susceptible to attacks that need to be considered in the design of secure cryptographic systems. Therefore, the selection and design of S-Boxes on a dynamic basis are crucial aspects of constructing a secure cryptographic algorithm for IoT devices. This chapter also contributes in proposing a newly defined dynamic key dependent algorithm based on the 16*16 non-linear S-Boxes for securing data between IoT devices and web-based platforms. Finally, the chapter describes the overall experimental evaluation results. Performance analysis for SiC-Chain security layer has been conducted on strict avalanche characteristics, encryption time, decryption time, throughput and memory consumption, besides, encryption and decryption of IoT data to validate its suitability for cold chain applications.

Section 4.2. describes the need for non-linear S-Boxes, Section 4.3 explains the algorithm for development of non-linear S-Boxes and Section 4.4 explain the performance evaluation of non-linear S-Boxes. Section 4.5 presents the need for dynamic key dependent algorithm. Proposed algorithm, is described in Section 4.6 with its subsections explaining the key components of the algorithm and selection of dynamic key dependent S-Box. The overall experimental evaluation of the algorithm for SiC-Chain security layer are presented in Section 4.7.

4.2 Need for Non-Linear S-Boxes

A S-Box is a fixed lookup table that performs substitution of elements from one representation to another in cryptographic algorithms. S-Boxes introduce confusion and non-linearity into the encryption process, increasing the algorithm's resistance to cryptanalysis. Modern block ciphers, like AES, DES and PRESENT are based on Shannon principle of confusion and diffusion. This non-linear auxiliary table S-Box becomes the confusion component.

However, some limitations and vulnerabilities persist that need to be considered in the design of secure cryptographic systems:

(i) **Vulnerability to attacks:** Static S-Boxes could be susceptible to varied attacks due to the S-Box properties through which information about encryption key or plaintext may be gained. If the S-Box lacks strong cryptographic properties, it may lead to the compromise of the entire cryptographic system**.**

(ii) **Fixed nature**: Static S-Boxes are predetermined and remain fixed during the entire encryption process. Attackers can perform exhaustive search attacks by analysing the fixed S-Box, making it easier to find weaknesses in the system. Static S-Boxes are fixed, pre-defined substitution tables that do not change during the encryption or decryption process. They are typically chosen during the design phase of a cryptographic algorithm and are part of its specifications. Static S-Boxes are often implemented as constants in the algorithm's source code or stored in hardware components. When designing symmetric key algorithms, the properties of S-Boxes, such as their objectiveness and the level of non-linearity they provide, are carefully considered to enhance the security of the algorithm. One example of a well-known static S-Box is the one used in the AES, which is a widely used symmetric key encryption algorithm. The AES S-Box is fixed and used in both the encryption and decryption processes. It is designed to have certain cryptographic properties that make

AES a secure block cipher. The use of static S-Boxes allows for faster encryption and decryption since the lookup tables are fixed and do not require any additional computational overhead to calculate them on the fly during the encryption process. However, it also means that the power of the algorithm heavily relies on quality of chosen S-Box. If an adversary manages to find weaknesses or patterns in the S-Box, it could potentially lead to vulnerabilities in the algorithm's security. Therefore, the selection and design of S-Boxes are crucial aspects of constructing a secure cryptographic algorithm for our proposed architecture-"SiC-Chain".

4.3 DEVELOPMENT OF NON-LINEAR S-BOXES

This section elucidates the mode for generating a non-linear 16*16 S-Box utilising chaotic logistic maps, as discussed in Sect.4.3.1 and Sect.4.3.2.

4.3.1. S-Box Generation

A logistic map, often referred to as the logistic chaotic map or logistic equation, is a mathematical model used to describe a chaotic system. It is one of the simplest nonlinear differential equations that exhibits chaotic behaviour. The map was popularized by the biologist Robert May in 1976 and has since been widely studied in various fields of science and engineering. Logistic map is popular a single dimensional chaotic map is iterated to generate S-Box.

In mathematics, the definition is:

$$x_{n+1} = r x_n (1 - x_n) \qquad (4.1)$$

where $0 < r \leq 4$, $x_n \in (0, 1)$. According to value of x_0, the x_n will decrease when r is assigned values between (0,1). The behavioural aspects of logistic map can vary significantly depending on the value of the parameter r. For certain values of r, the system shows ordered, stable behaviour, while for other values, the system exhibits chaotic behaviour.

For $0 \leq r < 1$, the system converges to a stable fixed point. As r increases, the system undergoes a period-doubling bifurcation and transitions into periodic behaviour. At around $r \approx 3.57$, the system enters chaotic behaviour and exhibits aperiodic, unpredictable dynamics. As r increases further, the chaotic behaviour becomes more complex.

The logistic map is an essential example in the study of chaos theory and has uses in diverse spheres, like population dynamics, cryptography, and random number generation. It has also been used to generate pseudorandom numbers due to its chaotic properties, although dedicated algorithms like the Mersenne Twister are more commonly used for this purpose nowadays.

4.3.2. Algorithm for 16*16 S-Box Generation

Choose initial values,$0 < r \leq 4$ (r=3.99), $x_0 \in (0, 1)$. A combined chaotic map is prescribed for producing chaotic sequences. Its data range may escalate the diversity owing to uniform distribution within [0,1] interval. Chaotic Logistic map can help in the generation of non-linear S-Boxes. The algorithm generates a 16 * 16 S-Box with random integers using logistic chaotic map. This matrix is stored in an array to make dynamic selection of one of the S-Box for every encryption round.

Algorithm for 16 * 16 S-Box Generation

Start

1. Let x=0
2. Let u = 3.99, d = a random number between [0,1]
3. Let ary [] = []
4. While x<16
    a. d = u*d*(1-d)
    b. r = rounded value of (d*16) %16

c. If r is not in ary

i. Append r to ary

d. x= x+1

5. End Loop

Stop

4.4 PERFORMANCE EVALUATION

As fixed or static S-Box of cryptographic algorithms can allow any attacker in investigating S-Box and identifying inherent weakness, this non-linear S-Box is generated using Logistic Chaotic Map. Non-linearity, Differential and Linear Approximation Probability are considered in the course of the analysis.

4.4.1. Non-Linearity

The non-linearity is a critical property that enhances the security of cryptographic algorithms. For symmetric key cryptography, S-Boxes are commonly used in block ciphers to perform substitution operations, introducing confusion and non-linearity to the encryption process.

The non-linearity property is typically evaluated by considering the Walsh-Hadamard transform of the S-Box. These analyses help to measure how much the S-Box behaves like a random function and how well it resists linear cryptanalysis attacks.

To ensure sufficient non-linearity in S-Boxes, cryptographic designers often use mathematical constructions, such as the S-Boxes used in AES, which are derived from finite field arithmetic and carefully chosen to meet specific criteria, including non-linearity, differential uniformity, and resistance to other cryptanalytic techniques.

Overall, the non-linearity of S-Boxes plays a crucial role in building secure cryptographic systems, preventing attackers from exploiting linear relationships and enhancing the overall resistance against various cryptanalytic attacks.

Let Boolean function f be a bent function on F_2^4 then function f has maximum Nonlinearity (Shah T. et al., 2019) achieved when n is even:

$$N_f\ (max) = 2^{n-1} - 2^{n/2-1} \quad (4.2)$$

This shows that maximum non linearity attained in GF (2^4) is 6 and for Dynamic S-Box obtained this is 5.

For example:

$$N_{f\,(}max) = 2^{4-1} - 2^{4/2\,-1}$$

$$= 2^3 - 2^1$$

$$= 8 - 2$$

$$= 6$$

Figure 4.1 shows the 3D representation of the 16*16 S-Box generated using Chaotic Logistic Map. Each of the S-Box contains 16 integer values where a line is plotted for a single S-Box value. Being perfectly non-linear, the points do not render a straight line where the relationship from one point to the other is non-linear.

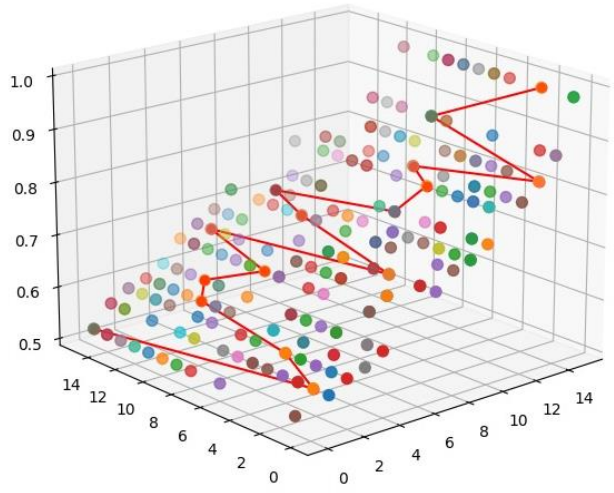

Figure 4.1 3D Non-Linear Representation of 16 * 16 SBox generated with Chaotic Logistic map

Figure 4.2 shows non-linearity analysis of SPN structure lightweight algorithms. The recommended S-Box has maximum Nonlinearity of 5. To tackle linear cryptanalysis, the nonlinearity must be high. This proposed S-Box offers maximum nonlinearity of 5, higher than that of other lightweight S-Boxes and minimum nonlinearity of 3, lesser for other S-Boxes.

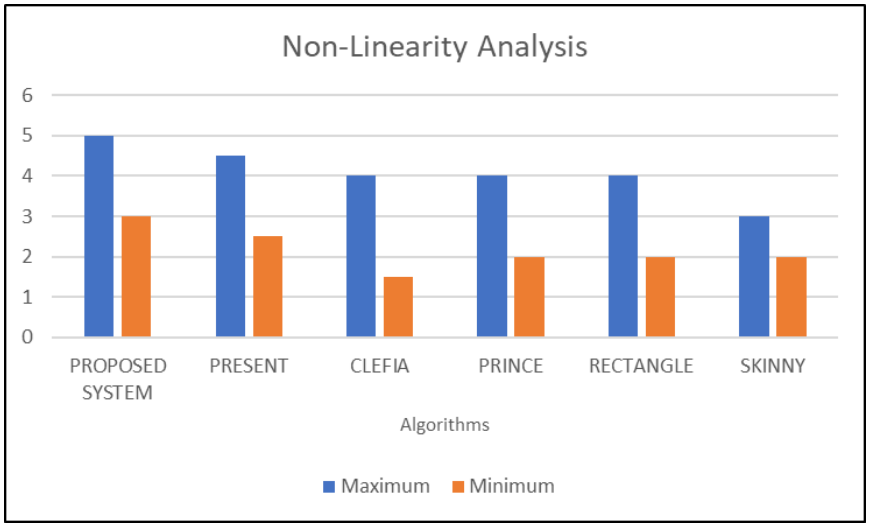

Figure 4.2 Non-linearity analysis of SPN structure lightweight algorithms

4.4.2. Differential Approximation Probability

Also known as Differential Probability (DP) or differential probability distribution, this is a measure used in the context of differential cryptanalysis to analyse the behaviour of S-Boxes in block ciphers. It quantifies the likelihood of a specific input plaintext difference propagating to a specific output ciphertext difference through the S-Box.

In the context of symmetric-key block ciphers, differential cryptanalysis is a technique used by cryptanalysts to break the cipher by observing how differences in the plaintext propagate through the encryption process to produce differences in the ciphertext. DP increases S-Box strength against these types of attacks. The concept works as below:

- Input Difference: Consider two plaintext inputs, denoted as P1 and P2, where the input differences, denoted as ΔP is defined as the bitwise XOR of P1 and P2.

- Output Difference: Corresponding to the inputs P1 and P2, the S-Box produces two ciphertext outputs, denoted as C1 and C2. The output difference, denoted as ΔC is the bitwise XOR of C1 and C2.

- Differential Probability: It measures the likelihood that the input difference ΔP will result in the output difference ΔC through the S-Box.

In general, lower DP value indicates stronger S-Box against differential cryptanalysis. A good S-Box should produce output differences that are uniformly distributed for all possible input differences, making it harder for cryptanalysts to exploit any specific input-output difference patterns.

When designing or analysing S-Boxes for cryptographic algorithms, it is essential to evaluate their differential approximation probabilities and other differential properties to ensure they provide the necessary resistance against differential cryptanalysis attacks. Cryptographers aim to design S-Boxes with low differential probabilities to enhance the security of the overall cryptographic system.

Differential approximation probability of an S-Box (Panchami, V. et al., 2023, Chen, J.,et al., 2022) is measure for Differential Uniformity as given below

$$DP(\Delta x \rightarrow \Delta y) = \frac{\#\{x \epsilon \frac{X}{S(x)} \oplus S(x \oplus \Delta x) = \Delta y\}}{2^n} \qquad (4.3)$$

Where Δx and Δy are input and output differentials for all inputs 2^n

Differential uniformity is the greatest value in the Differential distribution table. The ideal differential bound (highest differential within each S-Box) is differential probability = 0.25 for 4 * 4 S-Boxes. Figure 4.3 shows a sample differential approximation probability table of the proposed S-Boxes. In Figure 4.4, the application

results of input - output differentials to most likely output XOR of S-Box are depicted. Matrix maximum is 4/16 = 0.25 validates immense resilience of this S-Box for supporting differential technique.

Differential Uniformity of SBox : 4

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	2	0	2	0	2	2	0	2	2	4	0
2	0	0	0	2	0	0	2	4	0	0	0	2	0	0	2	4
3	0	2	4	2	0	0	0	0	2	2	0	0	0	2	2	0
4	0	0	0	2	0	0	2	0	0	0	4	2	0	0	2	4
5	0	2	0	4	2	2	0	2	0	0	2	0	0	2	0	0
6	0	4	0	0	4	0	0	0	0	4	0	0	4	0	0	0
7	0	0	4	2	2	0	0	0	2	0	0	0	2	2	2	0
8	0	0	0	0	0	2	0	2	0	2	2	4	2	2	0	0
9	0	2	0	2	2	0	2	0	0	2	0	2	2	0	2	0
10	0	0	4	0	2	0	2	0	2	0	0	2	2	2	0	0
11	0	0	0	2	0	4	2	0	4	0	0	2	0	0	2	0
12	0	2	0	0	2	2	4	2	0	0	2	0	0	2	0	0
13	0	2	0	0	2	0	0	0	0	2	4	0	2	0	0	4
14	0	2	4	0	0	0	2	0	2	2	0	2	0	2	0	0
15	0	0	0	0	0	4	0	4	4	0	0	0	0	0	0	4

Figure 4.3 Differential Approximation Table

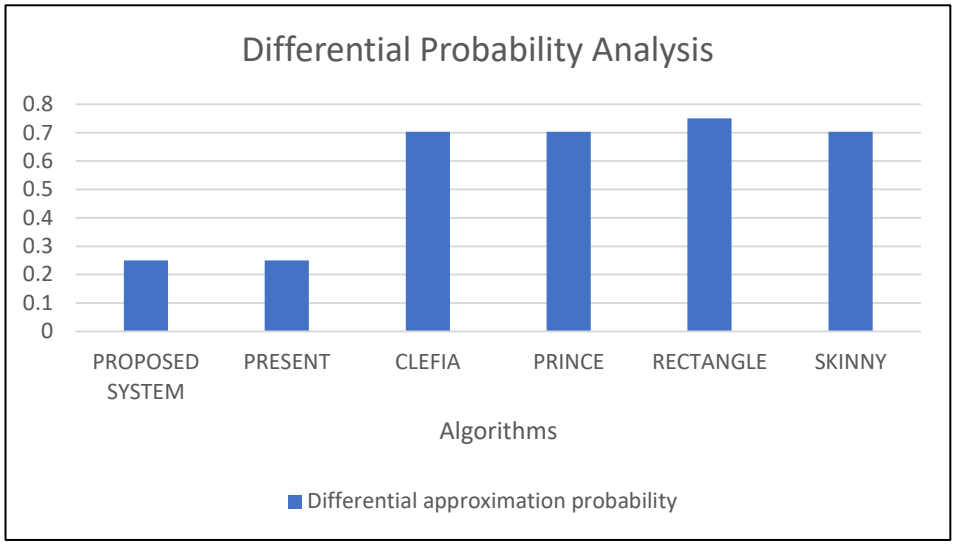

Figure 4.4 Differential Approximation Probability Analysis

4.4.3. Linear Approximation Probability

Linear Probability (LP) or linear approximation distribution, is a measure used in linear cryptanalysis for analysing behaviour of S-Box in block ciphers.In the context

of symmetric-key block ciphers, linear cryptanalysis is another technique used by cryptanalysts to break the cipher by observing linear approximations between the plaintext and ciphertext bits. LP helps assess strength of an S-Box against this type of attack. The concept works as below:

- Linear Approximation: A linear approximation represents a linear relationship between certain parts of plaintext and ciphertext, or vice versa. It is a probabilistic correlation between input and output bits, which means that input difference XORed with some mask, denoted as ΔP is statistically likely to equal the output difference XORed with another mask, denoted as ΔC. This correlation is denoted as ΔP ≈ ΔC, where "≈" represents the approximation.

- Linear Approximation Probability: This measures the likelihood that the linear approximation ΔP ≈ ΔC holds for a given S-Box.

In general, a lower value of LP(ΔP ≈ ΔC) indicates a stronger S-Box against linear cryptanalysis. A good S-Box should produce output differences that are statistically uncorrelated with any specific input differences, making it harder for cryptanalysts to exploit linear relationships.

When designing or analysing S-Boxes for cryptographic algorithms, it is essential to evaluate their linear approximation probabilities and other linear properties to ensure they provide the necessary resistance against linear cryptanalysis attacks. Cryptographers aim to design S-Boxes with low linear approximation probabilities to enhance the security of the overall cryptographic system.

The maximum degree of dissimilarity is represented by (Panchami, V. et al., 2023):

$$\text{LP} = \max \left| \frac{\#\{x/x.\Gamma x = S(x).\Gamma y\}}{2^n} - \frac{1}{2} \right| \qquad (4.4)$$

Where Γ_x is input mask; Γ_y output mask and x input vector of all inputs 2^n

The S-Box's nonlinear property is stronger when chances of linear approximation are less. Figure 4.5 displays the sample LP table of this proposed S-Boxes. Figure 4.6 exhibits that LP value is less than values for other S-Boxes, enabling greater resistance to linear cryptanalysis.

```
######## LAT of SBox-2 ########
       |0    1    2    3    4    5    6    7    8    9    A    B    C    D    E    F
-------------------------------------------------------------------------------------
0.0    |8    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
1.0    |0    0    0    0    0    4    0    4    0   -4    0    4    0    0    0    0
2.0    |0    0    2    2    2   -2    0    4    0    0   -2   -2    2   -2    4    0
3.0    |0    0    2    2    2    2    0    0    0    4   -2    2    2   -2   -4    0
4.0    |0    0    2   -2   -2   -2    4    0    0    0   -2    2   -2   -2    0   -4
5.0    |0    0   -2    2   -2    2    0    0   -4    0   -2   -2    2    2    0   -4
6.0    |0    0    0   -4    0    4    0    0    0    0    0   -4    0   -4    0    0
7.0    |0    0    4    0    0    0   -4    0   -4    0    0    0   -4    0    0    0
8.0    |0    0   -2    2    0    0    2   -2   -2   -2   -4    0   -2   -2    0    4
9.0    |0    0   -2    2    4    0   -2   -2    2   -2    0    0   -2   -2    0   -4
10.0   |0    4    0    0   -2   -2   -2    2    2   -2   -2   -2    0    0   -4    0
11.0   |0   -4    0    0    2   -2    2    2   -2   -2    2   -2    0    0   -4    0
12.0   |0    0    0    0    2    2    2    2    2    2   -2   -2   -4    4    0    0
13.0   |0    0    4    4   -2    2    2   -2    2   -2    2   -2    0    0    0    0
14.0   |0    4   -2    2    0    0    2    2   -2    2    4    0   -2   -2    0    0
15.0   |0    4    2   -2    4    0    2   -2   -2   -2    0    0    2    2    0    0
```

Figure 4.5 Linear Approximation Table

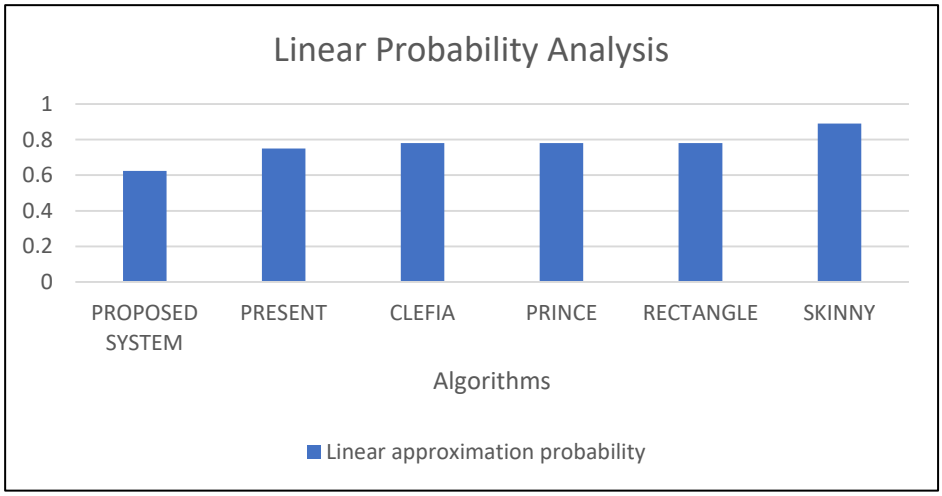

Figure 4.6 Linear Approximation Probability Analysis

Through experiments and comparisons, the conclusion is these generated S-Boxes have strong nonlinearity, and differential & linear approximation probability. So, this can resist any attack and is applicable for lightweight cryptographic algorithms.

4.5. Need for Dynamic key dependent algorithm

Before choosing a lightweight algorithm, it is recommended to assess its security features, performance characteristics, and any available analyses or evaluations. The usefulness of the simple lightweight algorithms depends on the demands and limitations of the application.

In majority of the lightweight cryptographic algorithms, the substitution layer as shown in Figure 4.7, is one of the core components of the round function used during both encryption and decryption processes. Its main purpose is to introduce confusion in the data by substituting each byte of the 64-bit state with a corresponding byte from a fixed 4-bit S-Box. This substitution is done independently for each nibble, and the mapping is fixed throughout the encryption and decryption process. Together with the permutation layer which introduces diffusion, the Substitution Layer forms the round function used in each round of the existing lightweight cryptographic algorithms.

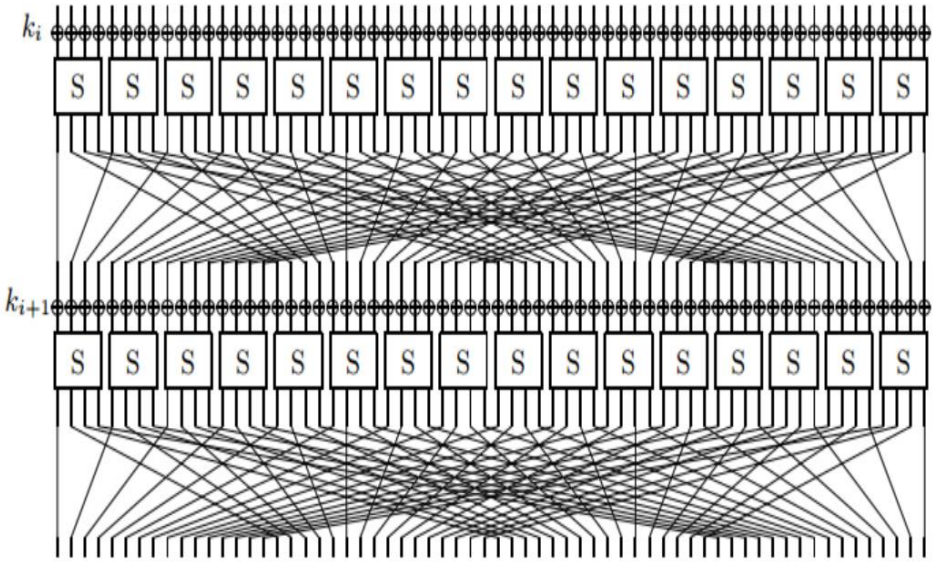

Figure 4.7 Mapping in Substitution Layer

The drawbacks of static S-Boxes are, it is vulnerable to differential and linear attacks. It does not possess a good degree of diffusion or a sound rate of avalanche effect. The key dependent approaches are superior as they impede attackers from offline

analysis of attacks on specific S-Box sets. Hence selection and design of dynamic key dependent S-Boxes and algorithms is important for enhancing security of IoT systems.

4.6. PROPOSED ALGORITHM WITH DYNAMIC KEY DEPENDENT S-BOX

A lightweight algorithm with dynamic key depenedent S-Box is proposed to secure the IoT data in SiC-Chain security layer.

Cryptographic Algorithm with Dynamic Key Dependent S-Box

Start

1. Read data
2. Let key = master key
3. Master Key, K = K =$K_{79,}$ K_{78} .....K_0
4. Import key, rounds, newly generated Dynamic S-Box from main function
5. i = 1
6. While i<rounds+1
   a. Right shift key by 16 and append it to round keys

      [Round keys are $K_{i=}K_{63,}K_{62,..}K_{0=}K_{79,}K_{78\ldots..}K_{16}$]

   b. Let k be the last round key

   c. Let x=0, i=0

   d. Extract the 8, 16, 24, 32, 40, 48, 56, 64 bits and store it in x as a 4-bit number that has bit 0 = (8th XOR 16th), bit 1 = (24th XOR 32nd), bit 2 = (40th XOR 48th), bit 3 = (56th XOR 64th)

   e. Store Sbox[x] in Sbox_temp

   f. Append Sbox_temp to Sbox_n

g. For each round the key register is updated as follows:

i. $[K_{79}K_{78}\ldots\ldots K_1K_0] = [K_{18}K_{17}\ldots\ldots.. K_{20}K_{19}]$

ii. $[K_{79}K_{78}K_{77}K_{76}] = S[K_{79}K_{78}K_{77}K_{76}]$

iii. $[K_{19}K_{18}K_{17}K_{16}K_{15}] = [K_{19}K_{18}K_{17}K_{16}K_{15}] \oplus$ *round_counter*

7. End Loop
8. Output round keys to main function
9. For each line, ti in data
    a. Append padding to ti and name it plain text
    b. Encrypt plain text, encrypted
    c. Output encrypted
    d. Decrypt it and output
    e. Remove padding

Stop

The flowchart below, Figure 4.8 depicts the overall workflow of the proposed algorithm. Initially the data and master key has been imported. Using one dimensional logistic chaotic map a 16*16 S-Box is generated. The selection of S-Box for every round of the algorithm is grounded on a dynamic basis. This is done using the round key generated. Finally, the encryption is performed.

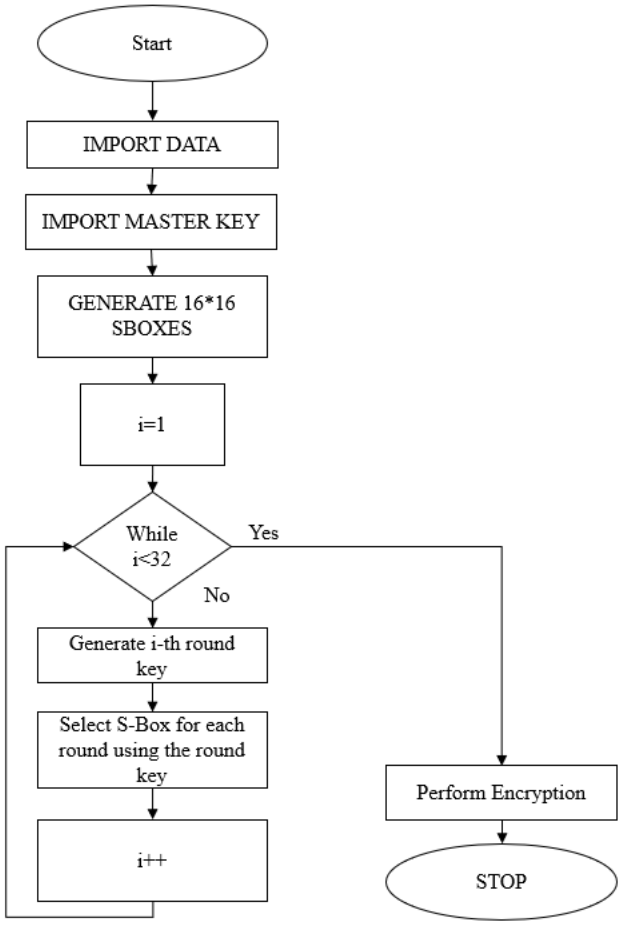

Figure 4.8 Flow-chart of the proposed algorithm

Key components and steps in the algorithm are:

4.6.1. Key Initialization

A master key of 80 bit or 128 bit is initialized as its first step. A round key schedule, which is an array of 64-bit round keys, is created using the key. There are 31 rounds for keys with 80 bits and 128 bits, respectively. In the proposed work, a master key of 80 bit is considered as input key.

4.6.2. Encryption Process

A 64-bit block of input plaintext can be encrypted using the key schedule. There are 31 rounds of encryption before the final stage.

4.6.2.1. Round Function

Each round applies a round function to the input data and the current round key. The round function consists of the following steps:

- Substitution Layer: Each byte of the 64-bit input is substituted with a corresponding byte from a 4-bit, 16* 16 S-Box. For each round, the selection of S-Box is dynamic, round key dependent.
- Permutation Layer: The bits of the 64-bit intermediate state are shuffled according to a fixed permutation.
- Key Mixing: The intermediate state is bitwise XORed with the round key.

Round key generation is same as that of NIST approved PRESENT algorithm. But for each round, the S-Box is different. As illustrated in the section 4.3, 16 * 16 non-linear S-Boxes are generated using chaotic logistic map.

- Key Schedule : 80 bit key
  - Master Key is K = $K_{79}K_{78}\ldots.\ K_0$
  - Round Key are Ki = $K_{63}K_{62}\ldots\ldots\ K_0 = K_{79}K_{78}\ldots.\ K_{16}$
  - Each round the key register is updated as follows:
    - $[K_{79}K_{78}\ldots\ldots K_1K_0] = [K_{18}K_{17}\ldots\ldots..\ K_{20}K_{19}]$

      (Right rotate 19 bits)
    - $[K_{79}K_{78}K_{77}K_{76}] = S[K_{79}K_{78}K_{77}K_{76}]$
    - $[K_{19}K_{18}K_{17}K_{16}K_{15}] = [K_{19}K_{18}K_{17}K_{16}K_{15}] \oplus$ *round_counter*

- Key Schedule : 128 bit key
    - Master Key is K = $K_{127}K_{126}\ldots.\ K_0$
    - Round Key are Ki = $K_{63}K_{62}\ldots\ldots\ K_0 = K_{127}K_{126}\ldots.\ K_{64}$
    - Each round the key register is updated as follows:
        - $[K_{127}K_{126}\ldots\ldots K_1K_0] = [K_{66}K_{65}\ldots\ldots\ K_{68}K_{67}]$
        - $[K_{127}K_{126}K_{125}K_{124}] = S[K_{127}K_{126}K_{125}K_{124}]$
        - $[K_{123}K_{122}K_{121}K_{120}] = S[K_{123}K_{122}K_{121}K_{120}]$
        - $[K_{66}K_{65}K_{64}K_{63}K_{62}] = [K_{66}K_{65}K_{64}K_{63}K_{62}] \oplus$ *round_counter*

4.6.2.2. Selection of Dynamic Key Dependent S-Box

From the generated 16 * 16 non- linear S-Boxes as discussed in section 4.3, the S-Box is selected for every round of algorithm on a dynamic basis as shown in Figure 4.9. The round key generation continues to be the same, but for every round, this S-Box is different.

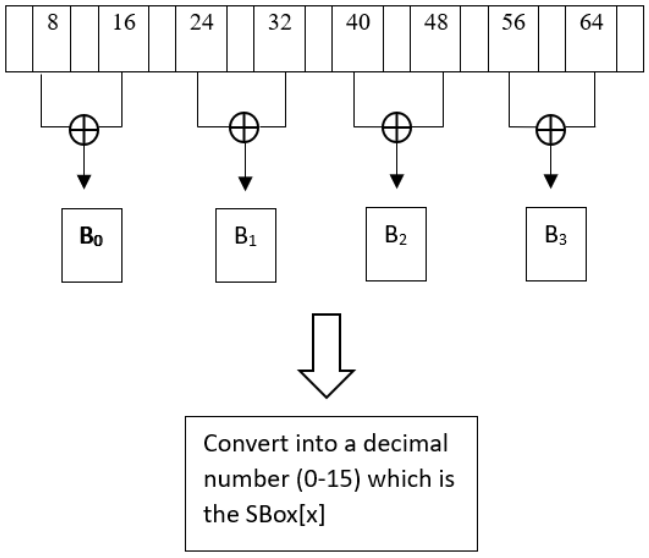

Figure 4.9 Selection of Dynamic Key Dependent S-Box

The round key consists of 64 bits for an 80 bit and 128-bit master key. From the 64-bit round key, extract 8,16,24,32,40,48,56 and 64 bit key and store it as a 4-bit number that has bit 0 = (8th XOR 16th), bit 1 = (24th XOR 32nd), bit 2 = (40th XOR 48th), bit 3 = (56th XOR 64th). Convert the 4 bit number into a decimal number which is the index of the current round S-Box. The selection is depicted in Figure 4.9.

The algorithm is described below:

Algorithm for Dynamic S-Box Selection

Start

1. Master Key, K =$K_{79,}$ K_{78} .....K_0
2. Import key, rounds, Sbox_temp, Sbox_n, Sbox from main function
3. i = 1
4. While i<rounds+1
   a. Right shift key by 16 and append it to round keys

      [Round keys are $K_{i=}K_{63,}$ $K_{62,..}K_{0=}K_{79,}K_{78....}K_{16]}$

   b. Let k be the last round key

   c. Let x=0, i=0

   d. Extract 8,16, 24, 32, 40, 48, 56, 64 bits and store it in x as a 4-bit number that has bit 0 = (8th XOR 16th), bit 1 = (24th XOR 32nd), bit 2 = (40th XOR 48th), bit 3 = (56th XOR 64th)

   e. Store Sbox[x] in Sbox_temp

   f. Append Sbox_temp to Sbox_n

   g. For each round the key register is updated thus:

      i. $[K_{79}K_{78}\ldots\ldots K_1K_0] = [K_{18}K_{17}\ldots\ldots.. K_{20}K_{19}]$

      ii. $[K_{79}K_{78}K_{77}K_{76}] = S[K_{79}K_{78}K_{77}K_{76}]$

iii. $[K_{19}K_{18}K_{17}K_{16}K_{15}] = [K_{19}K_{18}K_{17}K_{16}K_{15}] \oplus round\_counter$

5. End Loop

6. Output round keys to main function

Stop

4.6.3. Decryption Process

The final stage does not include the Permutation Layer. Decryption follows a similar procedure to encryption but is carried out in the opposite direction. The ciphertext is decrypted using the same round keys in the opposite order.

The intricacy of the Substitution Box and the P-Layer, which offer confusion and diffusion features, contributes to the algorithm's security. The less number of rounds and the simplicity of the algorithm enable efficient hardware implementations with low power consumption and small area requirements. As with any cryptographic algorithm, the security of algorithm depends on keeping the key secret and ensuring that no vulnerabilities or weaknesses are discovered that could be exploited to break the encryption.

A secure real- time data transfer between sensors and application is incorporated using dynamic key dependent cryptographic algorithm, to avoid the eavesdropping and manipulation of temperature and humidity values in SiC-Chain architecture. Here we have discussed the drawbacks of static S-Box in existing algorithm. It also describes the algorithm for selection of dynamic S-Boxes and the novel dynamic key dependent algorithm.

4.7. Overall Experimental Evaluation

This section describes the overall experimental evaluation results. Performance analysis has been conducted on strict avalanche characteristics, encryption time, decryption time, throughput and memory consumption, besides, encryption and decryption of IoT data to validate its suitability for SiC-Chain Architecture.

The performance evaluation has been conducted on the following parameters with respect to NIST approved lightweight cryptographic algorithm and proposed algorithm, which is described in the following sections.

4.7.1. Strict Avalanche effect

This is a crucial property of cryptographic algorithms, particularly symmetric-key block ciphers and hash functions. It describes the behaviour where even minute changes within input plaintext or key can produce notable and unpredictable changes within output ciphertext or hash value. In other words, a slight modification in the input should lead to drastic and widespread changes throughout the encryption or hashing process. The name "avalanche effect" is derived from the similarity to the behaviour of an actual avalanche, where a small disturbance in a snowpack can trigger a massive, cascading impact as the snow slides down the slope.

The avalanche effect is desirable in cryptographic algorithms for several reasons:

(i) Security against attacks: A strong avalanche effect makes the output highly sensitive to any changes in the input, thereby thwarting any attempt to deduce patterns, correlations, or relationships between the plaintext, ciphertext, or hash values. It adds confusion and complexity, making the algorithm resistant to differential cryptanalysis, linear cryptanalysis, and other cryptanalytic techniques.

(ii) Increased diffusion: The avalanche effect helps to propagate the effect of any single bit of the input across multiple bits of the output. This property is important for achieving confusion and diffusion, two essential aspects of secure encryption.

(iii) Error propagation: In data transmission or storage, a small error in the input should lead to a completely different output to prevent error propagation. The avalanche effect ensures that errors in the input are not easily predictable in the output, enhancing the robustness of the cryptographic system.

(iv) Enhancing randomness: The avalanche effect is related to the concept of pseudorandomness. A good cryptographic algorithm should produce ciphertext or hash values that appear statistically indistinguishable from random sequences. The avalanche effect contributes to the algorithm's pseudorandomness.

To evaluate the avalanche effect of a cryptographic algorithm, various statistical measures are used, such as the Hamming distance between related input and output, as well as the correlation between input and output bit patterns. A high avalanche effect is an indicative of a strong and secure cryptographic algorithm, while a weak or non-existent avalanche effect could indicate vulnerabilities that might be exploited by attackers.

The strict avalanche effect is a more stringent version of the regular avalanche effect in cryptographic algorithms. It refers to the property where any single alteration within input plaintext or key causes exactly half the bits in output ciphertext or hash value to flip, that is, change from 0 to 1 or from 1 to 0. In other words, each output bit should have a balanced probability of flipping for any single bit change in the input.

The strict avalanche effect is a desirable property in cryptographic algorithms for the following reasons:

(i) Enhanced security: By ensuring that each output bit changes with a probability close to 50% for any single input bit change, the strict avalanche effect contributes to a more balanced, chaotic behaviour. This characteristic increases the confusion and diffusion properties of the cryptographic algorithm, making it harder for attackers to exploit any bias or pattern in the encryption process.

(ii) Resistance against differential and linear cryptanalysis: Cryptographic algorithms exhibiting strict avalanche effect tend to be more resilient against differential and linear cryptanalysis attacks, which are common techniques used by attackers to analyze the statistical behaviour of ciphers and find vulnerabilities.

(iii) Error propagation and robustness: The strict avalanche effect helps to minimize error propagation. Even if a small error occurs in the input, the output is significantly altered, preventing the error from affecting subsequent computations.

Ensuring the strict avalanche effect in cryptographic algorithms is a challenging task. Designers of secure ciphers and hash functions carefully select S-Boxes, permutation layers, and other components to achieve this property. Additionally, they may employ various testing and evaluation techniques, such as using cryptographic test suites and statistical analyses, to verify and quantify the strict avalanche effect.

It's important to note that while the strict avalanche effect is a desirable characteristic, achieving perfect strict avalanche for all possible input bit changes in a cryptographic algorithm may be impractical due to the complexity of the algorithms and the necessary trade-offs between security and performance. As such, designers aim to achieve a high level of avalanche effect while considering other cryptographic properties and requirements.

Strict Avalanche Criterion (SAC) was first advocated in 1986 by Webster and Tavares (Panchami, V., et al., 2023). Figure 4.10 displays Strict Avalanche Effect. Over 77% output probability bit gets changed, for a one bit, two bit and four-bit change in plaintext.

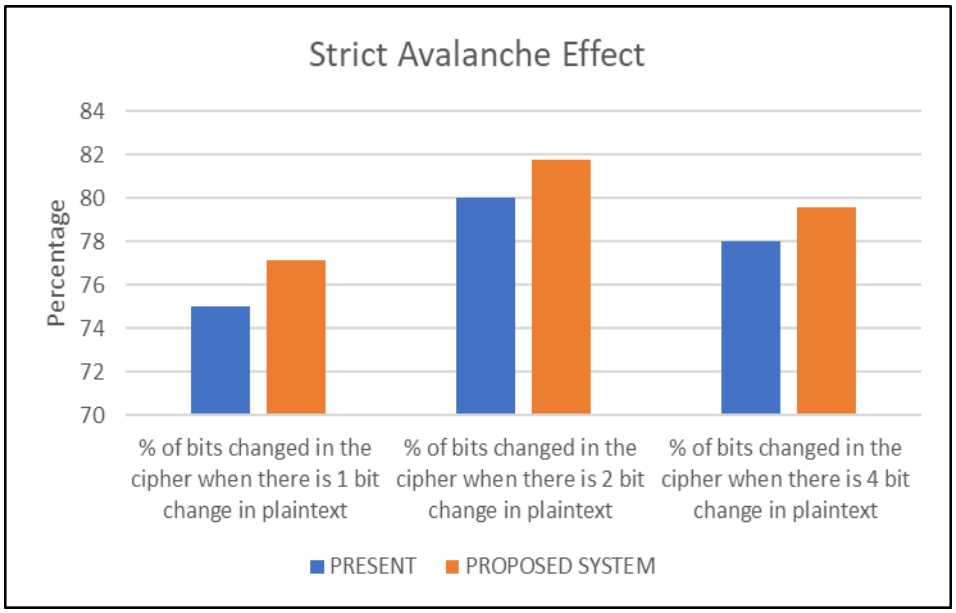

Figure 4.10 Strict Avalanche Effect w.r.t Plain text to Cipher Text

Figure 4.11 displays the Strict Avalanche Effect with respect to Key to Cipher Text. Over 78% output probability bit gets changed, for a one bit, two bit and four-bit change in key.

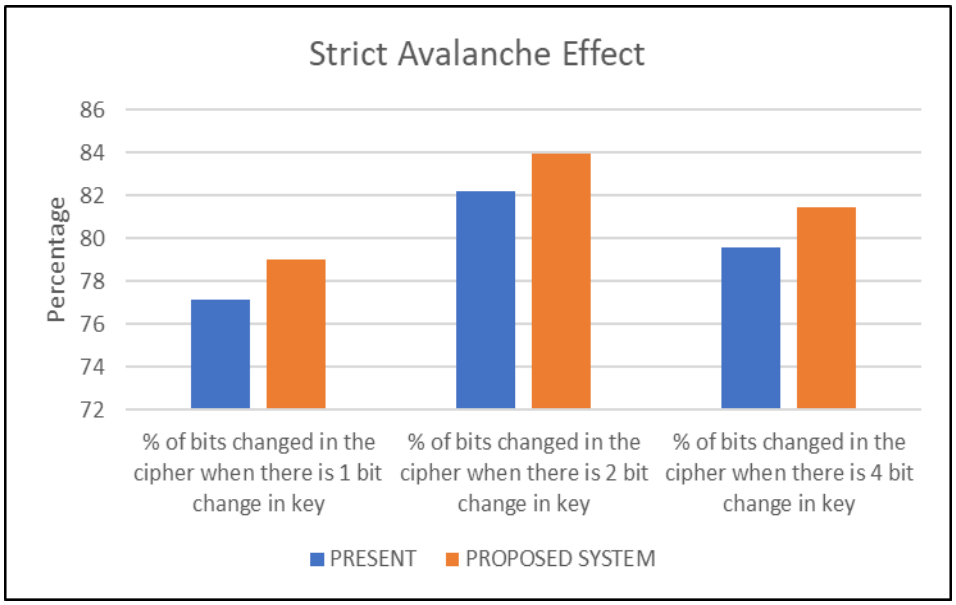

Figure 4.11 Strict Avalanche Effect w.r.t Key to Cipher Text

4.7.2. Average Encryption/Decryption Time

This is the basis for evaluating the proposed algorithms. The length of time it takes an encryption algorithm to convert plaintext into ciphertext is encryption time. The average encryption - decryption time of existing and proposed algorithms are shown in Figure 4.12 and 4.13.

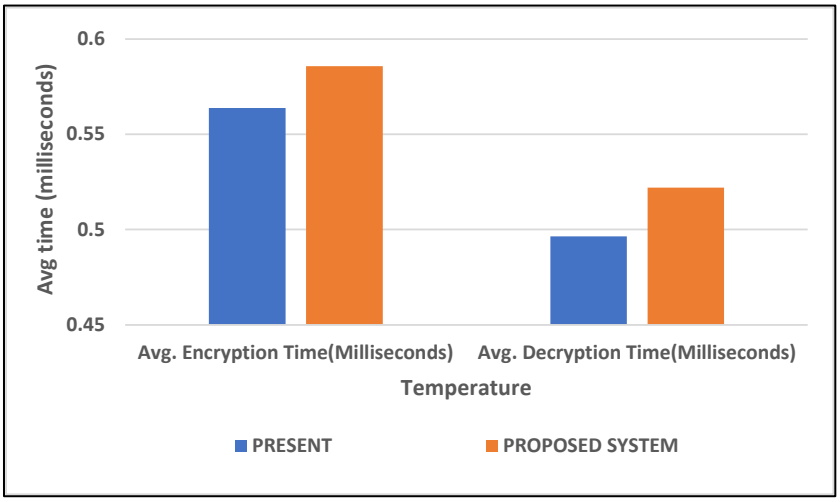

Figure 4.12 Average Encryption/Decryption Time for Temperature

The average encryption and decryption time for temperature and humidity parameters for the existing and proposed algorithms has negligible differences.

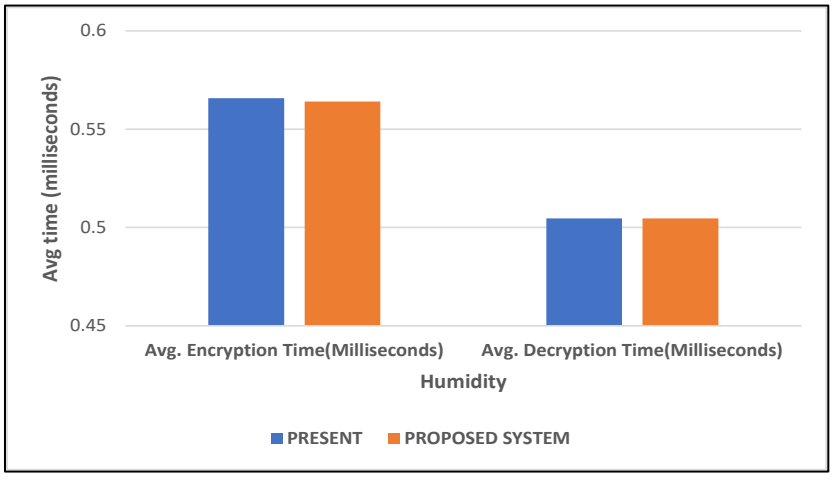

Figure 4.13 Average Encryption/Decryption Time for Humidity

4.7.3. Average Throughput and Memory Consumption

The throughput of an encryption technique explains the encryption's speed and computed as follows:

Throughput = No of bytes/Encryption Time

The unit of throughput is bytes/second. The average throughput of the existing and proposed algorithms are shown in Figure 4.14 and Figure 4.15.

The graphical representation of the above are shown below:

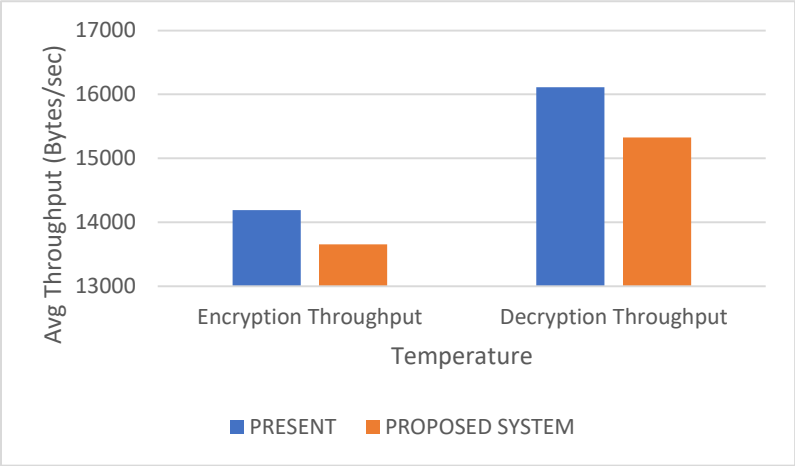

Figure 4.14 Average Encryption/Decryption Throughput for Temperature

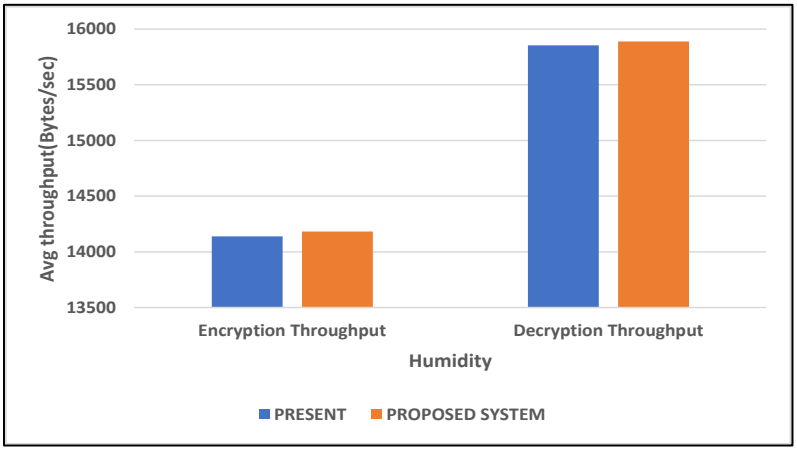

Figure 4.15 Average Encryption/Decryption Throughput for Humidity

The average encryption and decryption throughput for temperature and humidity parameters does not vary much for the existing and proposed algorithms. Figure 4.16 explains the memory consumption of existing and proposed algorithms. The RAM consumption is measured in kilobytes. For example, the peak RAM usage for existing is around 9.64 Kilobytes and 10.12 Kilobytes for the proposed algorithm. It also shows that proposed algorithm has negligible differences compared to existing algorithm. Figure 4.16 shows that the existing and proposed algorithms can take varying amounts of memory.

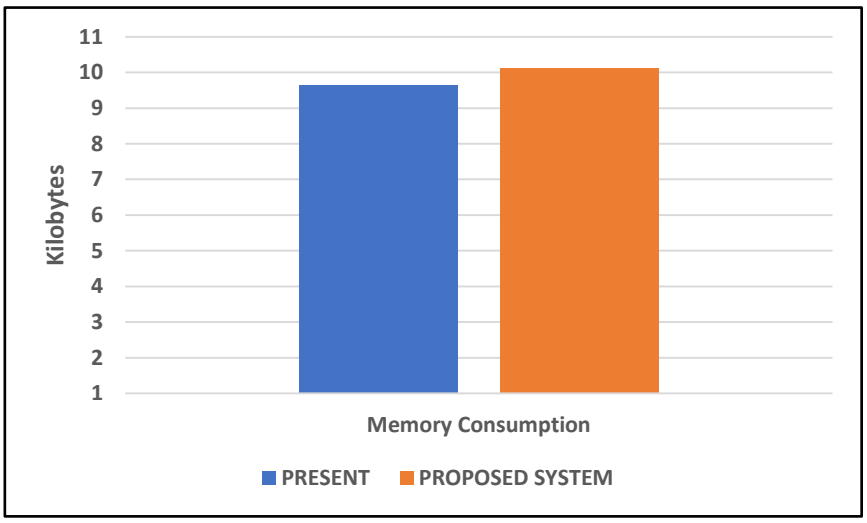

Figure 4.16 Memory Consumption

4.8. CHAPTER SUMMARY

The SiC-Chain security layer aims to address the security challenges in the SiC-Chain architecture. This chapter recommends non-linear S-Boxes generation mode based on chaotic map and its performance evaluation on non-linearity, differential approximation probability and linear approximation probability. It also describes the algorithm for selection of dynamic S-Boxes and the dynamic key dependent algorithm. This chapter provides overall experimental evaluation like strict avalanche effect, average encryption and decryption time, throughput and memory consumption. The algorithm outperforms better than other algorithms for resource constrained devices.