International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 11, No.1 (Jan-2022)

https://dx.doi.org/10.12785/ijcds/110138

A Hybrid Lightweight Cipher Algorithm

Seddiq Q. Abd Al-Rahman[1], **Omar A. Dawood**[2] **and Ali Makki Sagheer**[3]

[1]*Department of Computer Network Systems, University Of Anbar, Anbar, Iraq.*
[2]*Department of Information Systems, University Of Anbar, Anbar, Iraq*
[3]*Al-Qalam University College, Kirkuk, Iraq*

Received 28 Dec. 2020, Revised 25 Jun. 2021, Accepted 8 Jul. 2021, Published 20 Jan. 2022

Abstract: The speed development of the Internet applications and the explosive technology growth changed the lifestyle radically. The amount of data is increasing rapidly. Many of these data are sensitive and private. To protect these data, the lightweight encryption was developed. The paper proposing a new lightweight algorithm for securing the important data. The main objective of this mode is to achieve a trusted data transport throughout an open environment. The proposed algorithm involves design a modern lightweight symmetric block cipher algorithm that encrypts the sensitive data across unsecure channel. The designed algorithm works with 64-bit of encrypted data under 64-bit of secret ciphering key. The structure of the proposed algorithm is based on the combining the Substitution Permutation network and the Feistel network structures with 10, 16 or 20 rounds. The structure selection process depends on the binary bit-value for the ciphering key where zero for Feistel and one for SPN. The submitted cipher encompasses of three main layers: non-linear layer, linear layer and key addition layer. The proposed algorithm is applied for image encryption. Many of evaluation measure are computed according to several important metrics in terms of the randomness tests, avalanche effect test, time of implementation test, correlation, entropy, the Number of Modifying Pixel Rate (NPCR) and Unified Average Adjusted Intensity (UACI). All of them give us qualified results.

Keywords: Lightweight, Cryptography, Block Cipher, Symmetric, SPN, Feistel, S-Box

1. Introduction

Lightweight cryptography is a revolutionary approach which goals at supplying solutions to satisfy the challenge of developing speedy and efficient security mechanisms for harsh resource limited environments. These solutions consist of new designs in cryptographic primitives and protocols similarly to adapting and modifying contemporary cryptosystems [1][2][3]. To design lightweight cryptography, there are three aspects which might be required to be optimized: security, performance and cost. Security is measured via the number of bits of the key. Through increasing the size of the key, the supplied security might be higher. Performance is considered in terms of the total number of clock cycles to finish an operation that's proportional to the throughput and energy. The cost, expressed in terms of energy or area, depends on the used structure. Among these three factors, there is a trade-off which makes optimizing all of them collectively in one design very difficult, as shown in " Figure 1" [4][5][6]. For example, security is in tradeoff with performance and cost. The high security has be require to increasing the cost or the number of rounds. The cost and the performance are two other summits of this triangle. Serialized architecture yields lower power and area while it results in lower performance [7][8][9].
A block cipher is one in which a block of plaintext is dealts

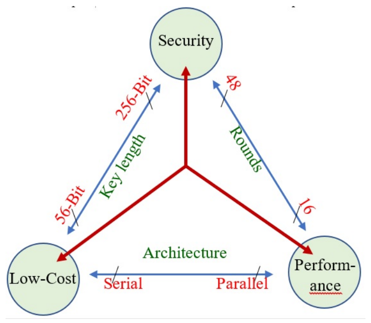


Figure 1. Design Trade-Offs for Lightweight Cryptography

with as an entire and used to supply a ciphertext block of equal length. Usually, a block length of 64 or 128 bits is used. A block cipher takes an input of plaintext block with secret key and produces a ciphertext block. Generally, the block length is fixed, and the block of ciphertext produced through the block cipher is normally the identical length as the plaintext block. So, a block cipher to be secure it ought to be a pseudorandom permutation (PRP), which means that so long as the key's secret, an attacker shouldn't be capable of compute an output of the block cipher from any input. That is, as long as key's secret and random from an

E-mail address: co.sedeikaldossary@uoanbar.edu.iq

attacker's perspective, they should haven't any clue about what encryption. This is because block ciphers vary so much in their technical design [10][11][2][12].
The block cipher design based fundamentally on the structure use that different between algorithms. The important structures can be used in traditional ciphers that can be used with lightweight ciphers. The structures are determining the general form of the action of the algorithm [13]. **Substitution-Permutation Network (SPN)** is perform dependent on chain of mathematical operations using substitutions and permutations in round function. SPN represents a repeat cipher that have a uniform round transformation by using all data block. The merging of these two layers represents a form of ciphers that substitutions supply confusion and permutations supply diffusion of input data block. **The Generalized Feistel Network structure (GFN)** is an oldest structure and most common construction. GFN is receive data block then divide into two parts, one of these parts using round function including substitutions and permutations. GFN use the same round function for encryption and decryption that making it a suitable structure for low-cost implementation of hardware. **The Involution structure** is work operations of involutional properties, that process in the forward is same as the backward, which means the same algorithm used for encryption and decryption. It differentiated by decreased cost of hardware and software implementation. The only difference between the encryption process and decryption process is the reverse subkey from key scheduling [10][11].
The residual paper is in good order as follows: Section 2 presents the most important studies and previous work related to the topic. The implementation of the proposed algorithm is performed in section 3. In section 4, we have presented the results obtained from the implementation of the proposal with a comparison with previous work. Finally, work is concluded in the last section.

2. Related Works

The lightweight block cipher is the most significant cryptographic of limited resources devices that have been proposed since its inception. So, numerous of researchers in the world suggested several distinct of lightweight ciphers, that can be explained as following:

- In 2019, Jagdish Patil et al. [14] proposed DoT algorithm which it has employed with block data 64-bit by used SPN structure. So, it accepts 80/128-bit as a secret key and 31 rounds. The round function includes: Adding subkey, S-box, shifting, permutation bits and other different shifting operation. DoT uses a 4×4 S-box and all permutation shifting bits using a different fixed table. DoT assumes the power design in the permutations of bits that do not use large memory and are quick in the process.

- In 2018, R Shantha Mary Joshitta et al. [15] proposed a lightweight block cipher for securing data in the healthcare environment. They work with 32-bit as a data block using a 64-bit secret key, the encryption used FN structure in 16 rounds. The round function F has three actions: firstly left part employ shifting operation, secondly the result of the first action is executed bitwise AND operation with left part data, finally left part performs the left circular shift. They require less computation power and memory size with small key size, this may reduce the security level something.

- In 2018, Lang Li et al. [16] proposed Substitution Feistel Network (SFN) as new lightweight cipher algorithm. It has designed in different methods that takes both SP and Feistel network structure. It encrypts 64-bit block data by using 96-bit as a secret key. The secret key is dividing into 64-bit that used to generated subkey and 32-bit of a control signal. So, these bits control conducts for network structure that find 32 rounds. Additionally, the round function for SPN structure consist: AddRoundKey, S1-Box layer, MixColumns, and MixRows, As well as, round function for Feistel network structure consists of: AddRoundKey, P-layer, MixXors and S2-box layer. Whilst, Key Expansion for SPN structure include: AddConstants, S2-box layer, MixColumns and MixRows, So, Key Expansion for Feistel network structure include: AddConstants, P-layer, MixXors and S1-box layer, as the S1-box layer is varying of S2-box layer.

- In 2017, Muhammad Usman et al. [17] proposed new algorithm of lightweight cipher which is called Secure IoT (SIT). It is a hybrid algorithm of Feistel network structure and SPN structure, it is supported with 64-bit secret key to encrypted 64-bit data block. SIT works with just five rounds and average 10 to 20 rounds. The encryption process at the first divides the data into four blocks of 16-bit. The first and last segments are adding the same subkey then log to round function. Other segments are change location then XOR with near segment, to finish round function change location between pair segments, and the round function is the same with round function for key expansion. While, the key expansion is divided into several segments of 4-bit, all 4 segments making 4×4 matrix that go to function block after nonlinear layer by two 4×4 S-Box, then XOR both of segments to generated subkey.

The aim at this study is to implement a hybrid lightweight algorithm of SPN with FN structures. The proposed cipher dedicated for securing the transfer of data between limited resources sector. The hybrid structure algorithm produces a unique lightweight algorithm that can be used for trusting network channels. The proposed cipher characterized by an elegant structure with light internal mathematical operations. The proposed cipher was built with involutional hybrid structure that uses the same operations in encryp-

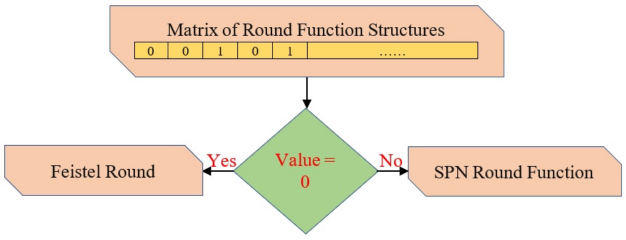


Figure 2. Choosing The Round Function Structures

tion and decryption processes. The main idea beyond the adaptation the involution notation is to reduces the number of Gate Equivalent (GE) on the electronic board design.

3. Implementation of Proposed Algorithm

The proposed cipher uses encryption/decryption data with a 64-bit secret key. For all cipher algorithms, the most fundamental pillars to building a strong and secure encryption/decryption algorithm are a powerful and secure S-box, key scheduling and appropriate structure use. The proposed cipher is designed to operate with 10, 16 or 20 rounds. Relying on this cipher, the proposed algorithm is designed by using the proved mathematical methods whilst conserving limited IoT device resources.
The proposed cipher is based on a combination of the SPN and Feistel structures that preserves the properties of both structures. Each structure is characterised by the size of the data used in the round function and by the order of layers within the round function in encryption and decryption. The round function in both structures contains four layers which are arranged as follows: S-box, Shifting, MixColumn and Adding Subkey layers. The proposed algorithm chooses a round function structure depending on the values of the master key bits.

A. *Forward Encryption Algorithm*

The proposed algorithm receives 64-bit length as a plaintext and a secret key. The secret key is used to generate the subkeys and round function structures according to the value of bits based on required length. The plaintext is converted into bits in accordance with the American Standard Code for Information Interchange (ASCII) code; the state for the block data is presented as a matrix of one dimension of 64 elements. The round function structures are returned in a 1×10, 1×16 or 1×20 matrix. The basis of the structure of this algorithm is as follows. If the value of elements for the round function structures is equal to zero, then the state of data s is sent to the Feistel round function; otherwise, it is sent to the SPN round function, as shown in "Figure 2".
Each structure receives 64-bit data and returns 64-bit state. The ciphertext is an end state when it finishes processing all round functions, which will be represented by the bits and can be returned to the text using ASCII code. "Figure 3" shows the algorithm structure.
The complexity of integrating the equations into a single structure is the motivation for proposing the merging of

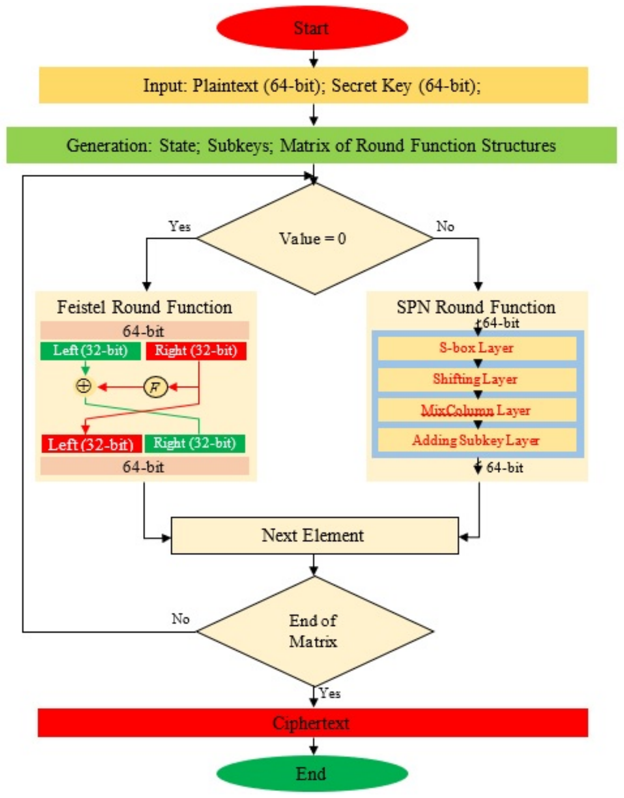
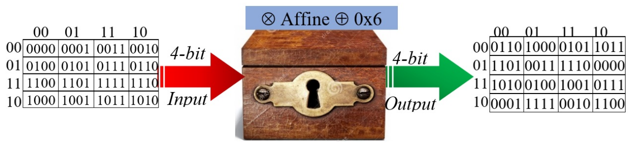


Figure 3. Cipher Structure Flowchart



	00	01	11	10
00	0000	0001	0011	0010
01	0100	0101	0111	0110
11	1100	1101	1111	1110
10	1000	1001	1011	1010

	00	01	11	10
00	0110	1000	0101	1011
01	1101	0011	1110	0000
11	1010	0100	1001	0111
10	0001	1111	0010	1100

Figure 4. Proposed S-box Design

the two structures to establish the proposed algorithm for securing data. Thus, the proposed algorithm includes several methods built in a powerful way. The subkeys are generated by key scheduling, the new S-box, MixColumn layer, the integration of the two structures and the method of selecting the round function structure. All these factors can make the proposed cipher powerful in resisting attacks.

1) *Nonlinear Layer (S-box) Design*

The substitution layer represents the nonlinearity level in the construction of any algorithm and complex relationship amongst the plaintext, key and ciphertext. The basic components of block ciphers are mappings $S : V_4 \rightarrow W_4$, known as S-boxes.
The proposed 4-bit S-box is built using modular arithmetic. Based on this forward and backward S-box, it is a self-inverse S-box. The multiplication is performed in a $GF(2)$ on the finite field in polynomial numbers, as shown in "Figure 4".

- An affine transformation matrix is generated by using the binary number 0111, and a rotational matrix is created based on the value 0001. The matrix is inversible, that is, equal to 1, when multiplied by itself, as explained in Equation 1.

$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \bigotimes \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

- Perform an XOR with a palindrome binary number of $4 - bit$ 0110 $0x6$ in hexa notation.
- For all elements to be calculated, the all elements between 0 to 24 $S = (s_0, \ldots, s_(n - 1))$ must apply Equation 2.

$$\begin{bmatrix} S'_0 \\ S'_1 \\ S'_2 \\ S'_3 \end{bmatrix} = \begin{bmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \end{bmatrix} \bigotimes \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \bigoplus \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \tag{2}$$

- The 4-bit is input into the S-box, and the proposal converts it into a vertical matrix by rotating it towards the clock. The bits are converted into horizontal opposite the clockwise to present 4-bit output form. Table I shows the results obtained from the entry of values of 2^4 into S-box, and Algorithm 1 shows the design steps.

Algorithm 1 Design of S-box

```
    Input: $Val$
Output: $S(Val)$
Start
Read $Val$
If $Val$ size is equal 4 and it's a binary number
M← $M \leftarrow Val_i \mid Val_{i+1} \mid Val_{i+2} \mid Val_{i+3}$
$P1 \leftarrow 0 \oplus ((M_1 \otimes 0) \oplus ((M_2 \otimes 1) \oplus ((M_3 \otimes 1) \oplus ((M_4 \otimes 1))$
$P2 \leftarrow 1 \oplus ((M_1 \otimes 0) \oplus ((M_2 \otimes 1) \oplus ((M_3 \otimes 1) \oplus ((M_4 \otimes 1))$
$P3 \leftarrow 1 \oplus ((M_1 \otimes 1) \oplus ((M_2 \otimes 1) \oplus ((M_3 \otimes 0) \oplus ((M_4 \otimes 1))$
$P4 \leftarrow 0 \oplus ((M_1 \otimes 0) \oplus ((M_2 \otimes 1) \oplus ((M_3 \otimes 1) \oplus ((M_4 \otimes 0))$
$S(Val) \leftarrow P1 \mid P2 \mid P3 \mid P4$
$Endif$
$Output : S(Val)$
$End.$
```

The S-box in proposed cipher is a good description of handling the nonlinearity level of design because it introduces N of the number bits corresponding to the same number of outputs. All output values of the S-box are subject to strict avalanche criteria, where each output value is different from the input value by half the number of bits. Every value previously designed from the self-inverse S-box in the standard way cannot achieve this criterion. Thus, the S-box has several criteria: independence of bits, nonlinearity and balance with vectors. These are achieved despite the lack of any link between bit values for inputs with bit values for output, subject to the NIST.

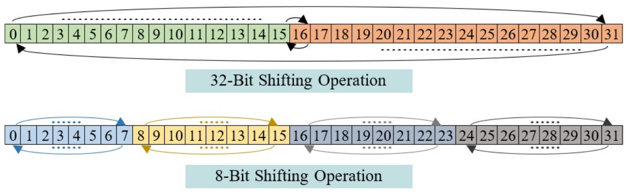


Figure 5. Shifting Layer Form

2) *Shifting Layer*

This transformation is one of permutation bits based on linearity operation by cyclical bits. It works by changing the bit locations through a fixed style. In the SPN structure, the state of ciphering data block is length 64-bit, which is divided into two equals in length parts $two32 - bitparts$. Each 32-bit part rotates on its own, and the rotation process has two similar phases with different sizes. The first $32 - bit$ size is divided into four equal sections of $8bits$, which represent the second size. The total size rotates then segments to rotating each part individually. The first size is $32bits$, the first bit rotates instead of the last bit, the second bit is interchanged with the previous one and so on. The second size is $8 - bit$, the bits rotate in the same form, the first bit is exchanged instead of the last bit and so on, as shown in Figure 5. In the Feistel structure, the state of the 32-bit ciphering data block is equal to one part of SPN structure, and the shifting operation works directly without needing division into parts.

The shifting layer is one of the cipher stages used to scatter bits and increase the complexity of the attackers by changing the bit locations.

3) *MixColumn Layer*

MixColumn is bricklayer permutation operating on the state column by column. A branch number is defined to quantify the relevant diffusion power. The larger the branch number, the more relevant the diffusion power. This transformation makes linear and the differential attacks difficult. The state of the ciphering block data of the proposed algorithm is represented in a one-dimensional matrix consisting of bits. For MixColumn transformation, the state must convert elements into hexadecimal numbers which can be handled within MixColumn. In the SPN structure, the $64-bit$ length of the state represents $16hexadecimal$ number values. The 16 numbers can be represented by a matrix with two dimensions 4×4 which can implement MixColumn transformation. The important function of the MDS matrix is to initiate MixColumn transformation, which is a matrix representing a function with certain diffusion properties with useful applications over $GF(2^n)$ in finite fields. The matrix columns of (MDS) are considered polynomials over $GF(2^4)$, and the modulo of Equation 3 is multiplied with a fixed polynomial Equation 4 as shown in Figure 6. The polynomial takes a half-byte (not full byte) value, which is composed of a single hexadecimal number, because the state matrix values consist of hexadecimal numbers, making multiplying and addition with one another easy. The

TABLE I. Bijective of 4-Bit S-Box in Hexadecimal Form

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

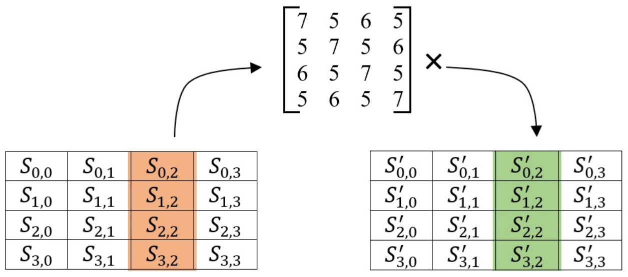


Figure 6. Encryption Process of MixColumn Stage of 4 × 4 State Matrix

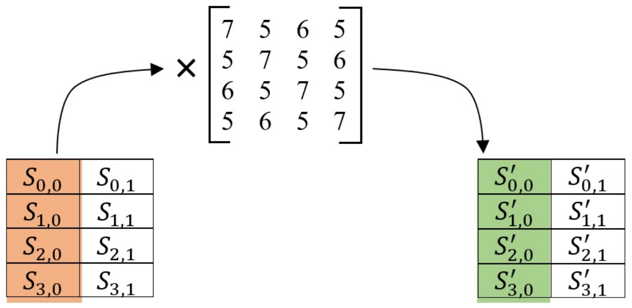


Figure 7. Encryption Process of MixColumn Stage of 4 × 2 State Matrix

MixColumn transformation is shown in Equation 5.

$$x^4 + x^3 + x + 1 \quad (3)$$

$$a(i) = ' 5' x^3 +' 6' x^2 +' 5' x + 7 \quad (4)$$

$$\begin{bmatrix} S'_{0,x} \\ S'_{1,x} \\ S'_{2,x} \\ S'_{3,x} \end{bmatrix} = \begin{bmatrix} 7 & 5 & 6 & 5 \\ 5 & 7 & 5 & 6 \\ 6 & 5 & 7 & 5 \\ 5 & 6 & 5 & 7 \end{bmatrix} \bigotimes \begin{bmatrix} S_{0,x} \\ S_{1,x} \\ S_{2,x} \\ S_{3,x} \end{bmatrix} \quad (5)$$

The inverting (feedback) MixColumn transformation, the MDS matrix is the same because it is designed in a self-inverse matrix. Equation 6 finds feedback values in this transformation.

$$\begin{bmatrix} S_{0,x} \\ S_{1,x} \\ S_{2,x} \\ S_{3,x} \end{bmatrix} = \begin{bmatrix} 7 & 5 & 6 & 5 \\ 5 & 7 & 5 & 6 \\ 6 & 5 & 7 & 5 \\ 5 & 6 & 5 & 7 \end{bmatrix} \bigotimes \begin{bmatrix} S'_{0,x} \\ S'_{1,x} \\ S'_{2,x} \\ S'_{3,x} \end{bmatrix} \quad (6)$$

In the Feistel structure, the 32-bit length of state data represents 8*hexadecimal* number values. The 8 numbers can be represented by a matrix with two dimensions 4×2 which can implement MixColumn transformation. Therefore, the multiplying operation is employed between the MDS matrix 4×4 and the state matrix 4×2 to obtain the output matrix 4×2 by using Equation 7. It returns the same size as the input in the output matrix, as shown in Figure 7.

$$\begin{bmatrix} S'_{0,x} \\ S'_{1,x} \\ S'_{2,x} \\ S'_{3,x} \end{bmatrix} = \begin{bmatrix} 7 & 5 & 6 & 5 \\ 5 & 7 & 5 & 6 \\ 6 & 5 & 7 & 5 \\ 5 & 6 & 5 & 7 \end{bmatrix} \bigotimes \begin{bmatrix} S_{0,x} \\ S_{1,x} \\ S_{2,x} \\ S_{3,x} \end{bmatrix} \quad (7)$$

The inverting (feedback) MixColumn transformation, that is, Equation 8, is employed to find feedback values in this transformation.

$$\begin{bmatrix} S_{0,x} \\ S_{1,x} \\ S_{2,x} \\ S_{3,x} \end{bmatrix} = \begin{bmatrix} 7 & 5 & 6 & 5 \\ 5 & 7 & 5 & 6 \\ 6 & 5 & 7 & 5 \\ 5 & 6 & 5 & 7 \end{bmatrix} \bigotimes \begin{bmatrix} S'_{0,x} \\ S'_{1,x} \\ S'_{2,x} \\ S'_{3,x} \end{bmatrix} \quad (8)$$

In both structures, the outputs of MixColumn transformation will be hexadecimal numbers, which must be returned to bits. These bits represent the state of the ciphering data block, which the proposed cipher can handle in the rest of the layers.

The polynomial values are self-inverse, and the values are used in half-byte for compatibility with the requirements of IoT device applications (limited resources), speeding up the process time whilst maintaining the necessary criteria in parts, calculation, storage and time. The half-byte handling in MixColumn is one contribution to proposed algorithm.

4) *Key Scheduling*

Key generation is a main part of cipher algorithms and other parts in cryptographic algorithms. Therefore, key scheduling is an independent algorithm regardless of the original encryption algorithm because it is based on mathematical methods that attackers cannot predict. This algorithm is one-way, and it should be more complex and more secure in its construction.

The proposed algorithm (Algorithm 2) encrypts the data with 64-bit secret key with 10/16/20 rounds. The key generation process includes three complex sub-functions which take the responsibilities of subkey generation steps. The steps of key scheduling algorithm are shown in Figure 8.

1) The state array of the ciphering key can be treated as a matrix of one-dimension form [0–63 bit] after converting the secret key as text by using ASCII code.

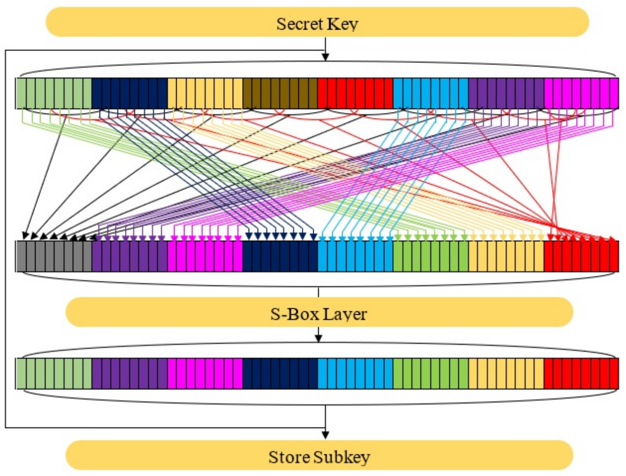


Figure 8. Key Scheduling Steps

TABLE II. FIRST PERMUTATION PART (XOR OPERATION)

Location New Bits	XOR between Location Bit 1	Location Bit 2
0	0	9
1	8	17
2	16	25
3	24	33
4	32	41
5	40	49
6	48	57
7	56	63
5	56	59
57	51	60
58	43	52
59	35	44
60	27	36
61	19	28
62	11	20
63	3	12

2) Bitwise XORed operation is taken to the whole stream of the ciphering key according to certain bit permutation to discover the first 8 bits and last 8 bits, as shown in Table II.
3) To fill other bits of the state array of the ciphering key, initial permutation is performed for bit location from location 8 to location 55 as shown in Table III.

4) The state is sent to the S-box to change bits by nonlinearity criterion while maintaining the size.

The end state array of the ciphering key represents the first subkey which is generated and must be returned to Step (2) to generate other subkeys. In this proposed algorithm, the state is treated with bits because it represents the most powerful point in the change. It is added to the logical operations gates for a quick option in dealing with

Algorithm 2 Design of Key Scheduling

```
    Input: $K^0_{63}$
Output: $SubK^0_{63}[20]$
Start
Read $K^0_{63}$
SubK$^0_{63}$[1] $\leftarrow K^0_{63}$
for i = 1 to 20
XOR between bits to generate new bits and padding state by shifting other bits
SubK$^0_{63}[i] \leftarrow updateitbysendingtoSBox$
$Store$ SubK$^0_{63}[i]$
End for
Output $SubK^0_{63}[20]$
End.
```

encryption and decryption. Owing to limited resources, the most powerful processes and the fastest in implementation must be selected, and the security and performance levels must be maintained. To increase the complexity for the attackers, the shifting operation is used, and the S-box gives it strength. The key scheduling process represents the basic structure of any encryption and decryption algorithm, so it must work in a more confrontational way for attackers.

5) *Selective Round Function Structure*

Repeating the operations on a text to encrypt will increase its complexity. Therefore, the more complex the ciphertext on the attacks is, the more powerful the algorithm. The proposed cipher can operate with several open rounds; for the text to be more secure in encryption, the number of rounds will be a minimum of 10 and can increase to 20. The proposed cipher will be employed on 10, 16 and 20 rounds, and the results will consider distinguishing the best from them. The results and the distinction between them will be explained to the system in three cases. The number of rounds is not determined randomly. The number of rounds in the AES encryption algorithm is 10. The Data Encryption Standard (DES) algorithm operates with 16 rounds. According to previous studies, the most lightweight cryptographic algorithms uses 20 rounds, ranging from 16 to 35 rounds.
The determination of the round function structure is generally based on the values of the secret key bits. The stage receives the secret key in the form of a one- dimensional matrix of 64 elements. The key is divided into eight 8-bit sections, the stage takes the bit values for third location of each section and stores them in the new matrix. Two bit values are taken from the first and second sections for three and five locations in case the work is on 10 rounds. All values of the bit are taken to fill 16 values if the work has 16 rounds. If the work has 20 rounds, four values are taken from seven location bits of four sections. The result is a matrix containing values taken from the master key, which is used to determine the direction of structure processing in each round, as shown in Figure 9. The selective round function structure chooses to maintain the speed of the

TABLE III. SECOND PERMUTATION PART (SHIFTING OPERATION)

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

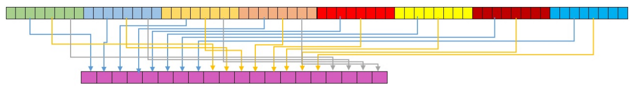


Figure 9. Formation Selective Round Function

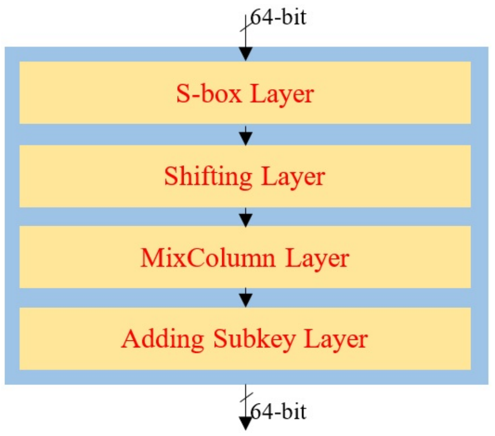


Figure 10. SPN Round Function in Proposed Algorithm

performance cipher to suit the resources of devices. It is an important step which combines two structures in a heterogeneous manner to be more complex and confuse the attackers.

6) *SPN Function Round*

The SPN function receives a 64-bit state of ciphering block data and produces 64 bits. The four basic operations are S-box, Shifting, MixColumn and Adding Subkey. The S-box layer replaces all four bits as a hexadecimal number, corresponding to the results obtained from the S-box algorithm. The round function sends the state to the rotation layer; it works as previously stated in the work of this layer. Similar to the MixColumn layer, the state is sent to the work of this layer. The Adding Subkey layer is required to the round number where it represents the subkey number to be added with the state by the bitwise XOR process, as shown in Figure 10.
The two layers of the S-box and MixColumn are more complex in the performance of SPN structure, which is built based on complex mathematical methods, can repel attacks, and shift and add the subkey.

7) *Feistel Function Round*

The round function of the Feistel structure receives the 64-bit state of ciphering block data and returns it with the same size. The basis work of the structure is to divide the state into two parts, deal with one of the parts, add it to the other, switch between them and reintegrate. The work will be on the balanced generalised Feistel network structure, and the round function is employed in the right part. The round function consists of four basic processes derived from the layers of the SPN structure: S-box, Shifting, MixColumn and Adding Subkey. The S-box layer interchanges bits of the state according to the resulting S-box algorithm. The shifting layer return changes bit locations according to the layer performance. The MixColumn layer receives 32 bits and returns the same bit size. The Adding Subkey layer requires the round number to bring the correct subkey to perform XOR operation. The subkeys generated from the key scheduling are 64 bits, so only 32 bits can be used. To finish the work of this function, the new right part results from adding the right part after processing with the left part by used XOR operation. The new left part is equal to the original right part before it performs any process, and the two parts are merged to return 64 bits, as shown in Figure 11.
The layers in the Feistel function are integrated to form a new form of structure which can be more complex and can repel the attacks.

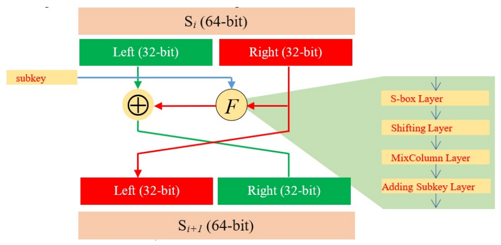


Figure 11. Feistel Round Function in Proposed Algorithm

8) *Encryption (Forward) Algorithm*

B. *Backward Decryption Algorithm*

The decryption process is the reverse process of encryption. The ciphertext and secret key are entered to obtain plaintext. The subkeys are created from the same key scheduling algorithm with the same number. Through the secret key, the selective round function structure creates the matrix with the same methods. The most important part of the work is invoked inversely for the values of the selective round function structure matrix. The subkeys must also be handled in reverse in terms of indexing. Now, the decryption algorithm is applied with the same structural

Algorithm 3 The Proposed Encryption Algorithm

```
Input: $Plaintext, SecretKey, NofRound$
Output: $Ciphertext$
Start
Read $Plaintext, SecretKey, NofRound$
Generate $subkey[NofRound] fromSecretKey$
Convert $PlaintexttoState[64] asbinarypresentation$
Generate $RondStruc[NofRound] fromSecretKey$
For i=1 to NofRound by step 1
if (RondStruc[i] = 0) then
Begin
    Divide State to $Left$ and $Right$ parts
    Send $Right$ to S-Box layer and return in $Right$
    Send $Right$ to Shifting layer and return in $Right$
    Send $Right$ to MixCollumn layer and return in $Right$
    Add $subkey[i]$-used 32-bit only- to $Right$
    $NewRight$ is Right with adding $Left$
    $NewLeft$ is Right in $step(8)$
    Update $State$ by merge $NewLeft$ and $NewRight$
End
Else
Begin
    Send $State$ to S-Box layer and return in $State$
    Send $State$ to Shifting layer by divided in two 32-bit and return in $State$
    Send $State$ to MixCollumn layer and return in $State$
    Add subkey[i] to $State$
End
End if
The $State$ is presented $Ciphertext$ in binary Number
Convert $State$ to string and printed it Output string $State$
End.
```

encryption algorithm in reverse to return the plaintext from the ciphertext. The details of the round functions and layers that must be considered in decryption follow.

1) Inverse of S-box, MixColumn and Shifting Layers

It is designed to be an inverted output of its inputs, and it is self-inverse. This feature is characterised by most IoT application algorithms to be agile algorithms with fast performance and quality of security.

2) Inverse SPN Transformation Round

The inverse function is the same original function but inverted by indexing layers, that is, it includes Adding Subkey, MixColumn, Shifting and S-box layers.

3) Inverse Feistel Transformation Round

The data are changed in sending to function, and the left part will work on them. The parts are positions reversed in end encryption/decryption processes.

4. The Results

A. Plaintext and Ciphertext Implementation

A similar secret key is used to encrypt 64-bit plaintext for the AES, DES, PRESENT, DoT, GRANULE, SFN and SIT algorithms. All ciphertexts are the same size as plain text, and the proposed cipher is encrypted with 10, 16 and 20 rounds. The devices of IoT applications deal with the texts of short-bodied generated from them. These devices may deal with image transformation considered long text.

B. Effect Implementation of Proposed Cipher on Images

The images are important information which can be handled by IoT applications. The images may be sent via the Internet. Therefore, the IoT devices may deal with sensitive systems or places and must achieve high security for these images. The images are encrypted with the proposed algorithm with 10 rounds, and original and encrypted images are sent in social networking and e-mail programmes. The encryption results are good, so they can be implemented with 16 and 20 rounds. For the purpose of resisting the brute-force attack, the key-space that is utilized for securing the image cryptosystem has to be large enough. This section includes a discussion of the proposed encryption algorithm results in terms of the statistical accuracy and the encrypted image. The cryptographic block cipher's base is the confusion; the blocks plain image are converted to cipher-image blocks, building on the 2-D Hénon map key of grey color image for XOR operation. The small changes in the initial parameters or conditions result in a variety of the changes in the final encryption image. After that, the diffusion is utilized for shifting the cryptographic block cipher; just several digits in the ciphertext can affect every one of the digits of the plain image and every one of the digits of the hidden key. In the suggested method, various image sizes and quality tests have been used in order to evaluate the proposed system's efficiency and security. Particularly, Picture Quality Evaluation (PQE) has been utilized [18], in addition to the Image Quality by Entropy evaluation and the randomness tests. The pixel strength diffusion measures for an image are seen in a histogram. The secure encryption system has to provide histograms that are evenly balanced for the purpose of surviving the statistical attacks. The histogram images in Figure 12 are for sample images in the original as well as the encrypted form. It has been observed that the regular image histograms aren't well-balanced, while histograms that are created from encrypted digital images are evenly balanced.

1) Picture Quality Evaluation (PQE) Metrics

Regarding the measurements of the quality of the encrypted and decrypted images, PQE has to be implemented. As seen in the image that has been indicated below, those metrics were utilized in the present proposal. Table IV lists the twelve measurements of the MSE. A value of the MSE has to be large, due to the fact that it indicates a difference between the plain and the cipher images [19], and every image has large values. The formula of the PSNR will then use those values of the MSE for the calculation of the ratios of the maximum probable

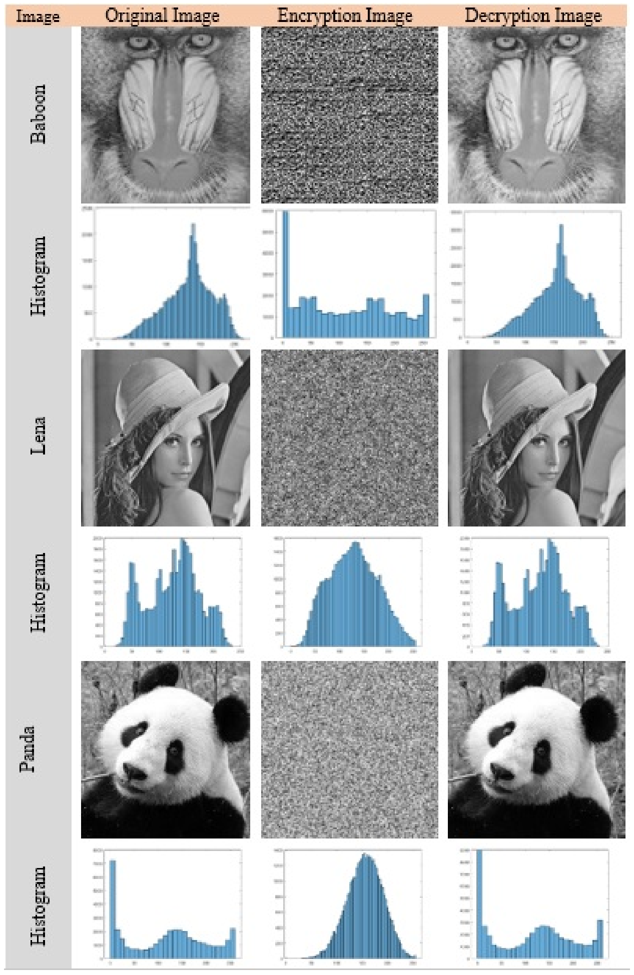


Figure 12. Histograms of Original and Encrypted Images

signal and noise powers. In addition to that, AD exhibits difference between the cipher and the plain images, and in the case where it has been divided by MSE, the resultant MD exhibits maximal error between the plain image and the cipher image after both images were transformed into the gray level, ranging from 0 to 255. The value of the NC between the decrypted and plain images have to be large, due to the fact that it shows difference between the plain and cipher images, while the MAE provides the same information that has been provided by the MSE. None-the-less, rather than utilizing squared differences between the plain image and the decrypted image, the MAE value will be equal to 1 in the case where the plain image and the decrypted image do not have any differences and less than 1 otherwise. The SNR includes all electric signals between the plain image and encrypted image, SIM carries the same information that has been conveyed by the MSE, and the EQ measures the quality of the encryption. All the images showed good results.

2) *Encryption and Decryption Running Time*

As can be seen in Table V, it takes a few milliseconds to encrypt and decrypt every one of the images. Various image sizes have been utilized for all images 200 x 200.

3) *Differential Attack Analysis*

The Number of Modifying Pixel Rate (NPCR) and Unified Average Adjusted Intensity (UACI) [20] are utilized for the estimation of the image encryption algorithm/code strength in terms of the differential attacks. The results have been listed in Table VI.
Table VII lists the values of the NPSR for the suggested system and other systems [21] [22] [23] [24], and Table VIII lists the values of the UACI for the proposed system as well as the other systems [22] [23] [24].

4) *Information Entropy*

The information entropy is important as well when computing the cipher file randomness. The $H(s)$ entropy can be calculated from the equation9.

$$H(s) = -\sum_{i=0}^{2^{n-1}} p(s_i)log_2 p(s_i) \tag{9}$$

Here, $p(si)$ represents the value of the probability of s. $H(s)$ should be 8 for a $2-1$ grey cipher-8 image that displays the random knowledge. As can be seen from in table IX.
The entropy values in Table X are close to this ideal value. Therefore, it is assumes that the algorithm proposed in this paper is of a strong randomness.

5) *Image Correlation*

The correlation between two values is a statistical relationship which depicts the dependency of one value to another. Data points that hold substantial dependency have a significant correlation value. A good cipher is expected to remove the dependency of the ciphertext on the original message. Therefore, no information can be extracted from the cipher alone, and no relationship can be drawn between the plaintext and the cipher text. In this experiment, the correlation coefficient is calculated for original and encrypted images by fixed direction in MATLAB. Table XI present the results calculated between encryption images with original images.
As basis for evaluating results, the range of values for the correlation coefficient bounded is by 1.0 on an absolute value basis or between -1.0 and 1.0. If the correlation coefficient is greater than 1.0 or less than -1.0, then the correlation measurement is incorrect. A correlation of -1.0 shows a perfect negative correlation, while a correlation of 1.0 shows a perfect positive correlation. A correlation of 0.0 shows zero or no relationship between the movement of the two variables.

TABLE IV. PQE METRICS FOR ENCRYPTED IMAGES

Name	MSE	PSNR	AD	MD	NC	MAE	NAE	SC	NSR	SIM	CC	EQ
Baboon	8145.14	0.006	6174.16	217	0.231	10509.13	0.316	1.396	1.358	128.7	0.0059	8138
Lena	9027.22	0.007	5224.18	252	0.930	10983.43	0.415	1.021	1.265	106.1	0.0019	6767
Panda	8891.74	0.006	5935.92	209	0.624	10059.72	0.510	1.009	1.387	140.4	0.0063	5952

TABLE V. ENCRYPTION AND DECRYPTION RUNNING TIME

Image	Encryption Time	Decryption Time
Baboon	510	616
Lena	502	612
Panda	303	407

TABLE VI. RANDOMNESS TESTS

Image	Encryption Time	Decryption Time
Baboon	0.982	0.31
Lena	0.989	0.33
Panda	0.989	0.35

TABLE VII. NPSR COMPARISONS AMONGST VARIOUS ALGORITHMS

Proposed Work	Lena Image
Our proposed	0.999
[21]	0.993
[22]	0.992
[23]	0.999
[24]	0.934

TABLE VIII. UACI COMPARISONS AMONGST VARIOUS ALGORITHMS

Proposed Work	Lena Image
Our proposed	0.338
[21]	0.334
[22]	0.335
[23]	0.999
[24]	0.335

TABLE IX. INFORMATION ENTROPY

Name	Baboon	Lena	Panda
Inf. Entropy	7.6431	7.6386	7. 6062

TABLE X. INFORMATION ENTROPY COMPARISONS AMONGST VARIOUS ALGORITHMS

Image	Our Proposed	[20]	[25]
Lena	7.6386	7.997	7.997

TABLE XI. CORRELATION FOR ENCRYPTION IMAGES WITH ORIGINAL ONE

Image	Correlation
Baboon	0.0039
Cameraman	0.0035
Lena	0.0114
Panda	0.0053

The result shows that the correlation of coefficients for the encrypted images is very small, indicating that the attacker cannot obtain any valuable information by exploiting a statistic attack.

C. Avalanche Effect

In cryptography, the avalanche effect is the desirable property of cryptographic algorithms, typically for block ciphers and cryptographic hash functions, wherein if an input is changed slightly (for example, flipping a single bit), the output changes significantly. In the case of high-quality block ciphers, such a small change in either the key or the plaintext should cause a drastic change in the ciphertext.
In the proposed algorithm, the change of one bit in the plaintext which has been encrypted results in changing over half the number of bits for the previous case. This observation was made in the case of encryption 10, 16 and 20 rounds; the difference between them is very minimal. In the proposed algorithm with 10 round, avalanche effect is calculated in all changes to the plaintext. The first bit is changed to the plaintext and encrypted. The ciphertext with the origin ciphertext calculates the first avalanche effect. The second bit of the original plaintext is changed and encrypted to calculate the second avalanche effect for the resulting ciphertext with the original ciphertext. On this basis, all bits are changed one by one, encrypted and calculate the avalanche effect. The plaintext and ciphertext are $64bits$ in block size, so these operations are performed 64 times. The results range from 25 to 33 bits changed or a 45% rate for all changes to measure the avalanche effect with the proposed cipher in 10 rounds. The previous operations are repeated to measure the avalanche effect with the proposed cipher in 16 rounds. The rate of bit change of in the range of 28 to 33 and at a rate of 49% for all $64bits$. The criteria are re-calculated using the proposed cipher of 20 rounds. The change of bits range from 30 to 38, and the ratio in this encryption achieves a level exceeding half, which is 51%.
The results show that the proposed algorithm is close to strict avalanche criteria, which can define the transformation of approximately half of the plaintext bits into the output

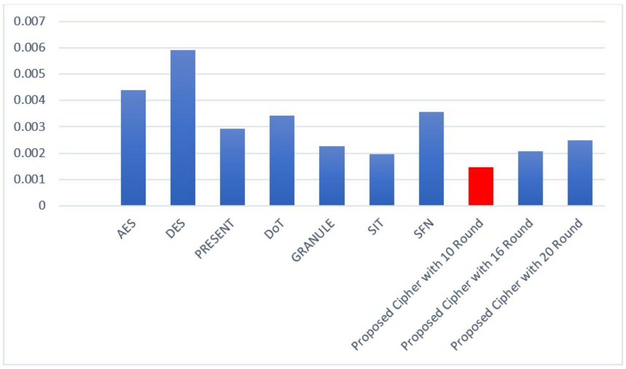


Figure 13. Compare Execution Time

block. Performing an analysis is difficult. These results confirm the strength of the work of the S-box, which is the key measure in the success of this system.

D. *Randomness Test for NIST Statistical*

The NIST test suite is composed of 15 tests, as explained in [26]. These tests focus on different types of non-randomness that could exist in a sequence. Each test deals with a minimum number of bits.
Depending on the values of tests in [26], handling of IoT applications can not be done more than the first seven tests because the remaining tests need a large number of bits. The same $128 - bit$ plaintext and the secret key are applied to the proposed algorithm and other algorithms. The probabilistic measure of the randomness of the sequence is under test. If the significance level (α) is chosen to be 0.01 (common values of in cryptography are about 0.01), then approximately 1% of the sequences are expected to be non-random. The results of the proposed cipher with 10 rounds obtains the highest values in four tests of the seventh. The proposed cipher with 16 and 20 rounds achieves success in all tests. Table XII explains the results obtained by the proposed algorithm with the values obtained in the tests.

E. *Execution Time*

One of the most important factors in the implementation of applications of limited devices (IoTs) is the time taken to implement the algorithm. These devices are used to connect to the Internet continuously and the number of attackers attacked it, so time plays a prominent role in the performance of its work. Figure 13 explains the execution time of the proposed algorithm on an HP laptop with processor (Intel® Core™ i7 -4600U) and a RAM (4096 MB).
The data Figure 13 shows that the proposed algorithm with 10 rounds achieves the least execution time. The proposed algorithm ranks third when implemented in 16 rounds. The proposed cipher with 20 rounds is fifth. The standard lightweight cryptography phrase 'Having high security requires increasing either the number of rounds or the cost', which means that a high degree of security cannot be achieved without sacrificing one of the pillars of the triangle for lightweight cipher for security IoT devices.

F. *Strength of Proposed Algorithm*

The security of the block cipher depends on the encryption function. Software implementation of block cipher is stronger than software implementation of stream cipher, and any error transmitting in one block generally does not affect other block. Thus, the proposed cipher uses a word $8bytes$ as data, and each word $8 - byte$ block is encrypted separately but using the same key. Suppose an entry of encryption plaintext as long $225bytes$, the proposed cipher takes the first word of data and encrypts them using the key scheduling. The ciphertext has the same length of word. After the first block, the key scheduling does not change and takes the next block.
Plaintext = 225 bytes divided by word (8 bytes -64-bit-) as block size.
= 28 word as 8 bytes with 1 byte as residue.
For encryption, the last 1 byte uses the padding technique. Extra bytes of zeros are added to make the last block size of $8bytes$. In decryption, the cipher text of the proposed can not recognise the padding. Owing to the handling of limited resource devices, storing encrypted texts to override their encryption more than once in the same event is not possible. The time and speed factors do not allow for a memory check if data have been previously encrypted.
The $64 - bit$ size for the key and plaintext is linked to the XOR process. Thus, the chance of breaking the probability of the proposed algorithm is 2^64 equal 18446744073709551616, which means time complexity is required. This digit needs many supercomputers over the speed of data transfer in Internet because of dealing with IoT devices. Furthermore, KIRCHHOFF law states, the secrecy of an algorithm lies in the key, not the algorithm. On this basis, the key scheduling imposes more complexity on the attackers. The process of XOR is used, changing the value of the bit depending on other bit values, which means a significant increase in probability to double. With the repetition of operation in half the size of the key and using the rotation with the other half, the attackers find difficulty in using two operations on the same level, maintaining complexity in design. The substitution of values in the S-box phase and the power of their design result in change data fragmentation, making predicting their potential difficult.
The proposed structure and the method of selecting each round function are very complex stand-alone methods because they depends on the bits of secret key. The proposed structure of the algorithm in encryption and decryption is specified based on the secret key in addition to the Kirchhoff law. Maintaining the security key mining is the basis for achieving the security of the algorithm. The system is more complex not in encryption and decryption, but in maintaining the key, which represents the structure of encryption operations in addition to its value. All mathematical complexities in the proposed algorithm are implemented in an easy and simplified way in encryption and decryption. The

TABLE XII. PROPOSED ALGORITHM RESULTS UNDER NIST TESTS

Test Name	Proposed Cipher with 10 Round	Proposed Cipher with 16 Round	Proposed Cipher with 20 Round
Frequency test	1.000000	0.376759	0.077100
Frequency test within a block	0.753143	0.511861	0.615961
Runs test	0.859684	0.913690	0.234389
Cumulative REVERSE	0.984155	0.431439	0.067790
Cumulative FORWARD	0.984155	0.431439	0.126863
Test for the longest run of ones in a block	1.000000	1.000000	1.000000
Serial test P-v1	1.000000	0.635639	0.163246
Serial test P-v2	1.000000	0.723674	0.479500
Approximate entropy test	0.036782	0.544026	0.300463

degree of complexity is easy to implement in the encryption and decryption phases, repelling attackers depending on the strength of the key generated in the system. Thus, the proposed cipher can be designed to run fast with more complexity for analysis. A compact trade-off exists between the strong performance in implementation and flexibility with increasing the number of rounds.

5. CONCLUSIONS AND SUGGESTIONS FUTURE WORK

In this proposal, a lightweight cipher algorithm was designed based on the incorporation of the substitution permutation network and the Feistel network. Hybridization depends on the use of the main key bits in determining the structure of each round within the encryption/decryption processes. Our proposed encrypts $64 - bit$ plaintext with a master key of the same length text. The proposal was implemented in several different courses, which were 10, 16, or 20 rounds. The layers in the two structures had similar operations with the different text size input to them. The S-Box is designed in a new way to have the output results differently from their inputs. We proposed a new approach for creating subkeys generated from the master key. Our proposal has achieved good results. Image encoding through histogram, entropy, and correlation accounts were sent to it via various social media. The results of the Avalanche Effect were equal to 50% for rounds 16 and more for rounds 20. The results of our proposal were successful in Randomness Tests for NIST Statistical and the implementation time for our proposal was fast compared to other algorithms. As for future research, we suggest implementing the proposed lightweight algorithm cipher on embedded devices likewise FPGA, ASIC, and others, and compute the required number of GE and the power consumption, related to the proposed algorithm cipher operations.

References

[1] A. Khattab, Z. Jeddi, E. Amini, and M. Bayoumi, *RFID security: a lightweight paradigm*. Springer, 2016.

[2] C. L. C. W. Group *et al.*, "Cryptrec cryptographic technology guideline (lightweight cryptography)," *CRYPTREC Report March*, 2017.

[3] H. Song, G. A. Fink, and S. Jeschke, *Security and privacy in cyber-physical systems: Foundations, principles, and applications*. John Wiley & Sons, 2021.

[4] R. Roman, P. Najera, and J. Lopez, "Securing the internet of things," *Computer*, vol. 44, no. 9, pp. 51–58, 2011.

[5] B. J. Mohd, T. Hayajneh, K. M. A. Yousef, Z. A. Khalaf, and M. Z. A. Bhuiyan, "Hardware design and modeling of lightweight block ciphers for secure communications," *Future Generation Computer Systems*, vol. 83, pp. 510–521, 2018.

[6] S. B. Sadkhan and A. O. Salman, "A survey on lightweight-cryptography status and future challenges," in *2018 International Conference on Advance of Sustainable Engineering and its Application (ICASEA)*. IEEE, 2018, pp. 105–108.

[7] K. McKay, L. Bassham, M. Sönmez Turan, and N. Mouha, "Report on lightweight cryptography," National Institute of Standards and Technology, Tech. Rep., 2016.

[8] K. Georgiou, S. Xavier-de Souza, and K. Eder, "The iot energy challenge: A software perspective," *IEEE Embedded Systems Letters*, vol. 10, no. 3, pp. 53–56, 2017.

[9] D. E. Kouicem, A. Bouabdallah, and H. Lakhlef, "Internet of things security: A top-down survey," *Computer Networks*, vol. 141, pp. 199–221, 2018.

[10] W. Stallings, "Cryptography and network security principles and practice, global edi," 2017.

[11] K. M. Martin, "Everyday cryptography," *The Australian Mathematical Society*, vol. 231, no. 6, 2012.

[12] J.-P. Aumasson, *Serious cryptography: a practical introduction to modern encryption*. No Starch Press, 2017.

[13] A. M. Sagheer, A. A. Abdulhameed, and M. A. AbdulJabbar, "Sms security for smartphone," in *2013 Sixth International Conference on Developments in eSystems Engineering*. IEEE, 2013, pp. 281–285.

[14] J. Patil, G. Bansod, and K. S. Kant, "Dot: A new ultra-lightweight sp network encryption design for resource-constrained environment," in *Proceedings of the 2nd International Conference on Data Engineering and Communication Technology*. Springer, 2019, pp. 249–257.

[15] R. S. M. Joshitta and L. Arockiam, "A novel block cipher for enhancing data security in healthcare internet of things," in *Journal*

of Physics: Conference Series, vol. 1142, no. 1. IOP Publishing, 2018, p. 012002.

[16] L. Li, B. Liu, Y. Zhou, and Y. Zou, "Sfn: A new lightweight block cipher," *Microprocessors and Microsystems*, vol. 60, pp. 138–150, 2018.

[17] M. Usman, I. Ahmed, M. I. Aslam, S. Khan, and U. A. Shah, "Sit: a lightweight encryption algorithm for secure internet of things," *arXiv preprint arXiv:1704.08688*, 2017.

[18] M. Mrak, S. Grgic, and M. Grgic, "Picture quality measures in image compression systems," in *The IEEE Region 8 EUROCON 2003. Computer as a Tool.*, vol. 1. IEEE, 2003, pp. 233–236.

[19] A. Kadhim *et al.*, "New image encryption based on pixel mixing and generating chaos system," *Al-Qadisiyah Journal Of Pure Science*, vol. 25, no. 4, pp. 1–14, 2020.

[20] H. Liu, B. Zhao, and L. Huang, "Quantum image encryption scheme using arnold transform and s-box scrambling," *Entropy*, vol. 21, no. 4, p. 343, 2019.

[21] A. S. Saljoughi and H. Mirvaziri, "A new method for image encryption by 3d chaotic map," *Pattern Analysis and Applications*, vol. 22, no. 1, pp. 243–257, 2019.

[22] P. Ramasamy, V. Ranganathan, S. Kadry, R. Damaševičius, and T. Blažauskas, "An image encryption scheme based on block scrambling, modified zigzag transformation and key generation using enhanced logistic—tent map," *Entropy*, vol. 21, no. 7, p. 656, 2019.

[23] M. Essaid, I. Akharraz, A. Saaidi, and A. Mouhib, "A new image encryption scheme based on confusion-diffusion using an enhanced skew tent map," *Procedia Computer Science*, vol. 127, pp. 539–548, 2018.

[24] Y. Zhang, "The unified image encryption algorithm based on chaos and cubic s-box," *Information Sciences*, vol. 450, pp. 361–377, 2018.

[25] H. Liu and C. Jin, "A novel color image encryption algorithm based on quantum chaos sequence," *3D Research*, vol. 8, no. 1, p. 4, 2017.

[26] L. E. Bassham III, A. L. Rukhin, J. Soto, J. R. Nechvatal, M. E. Smid, E. B. Barker, S. D. Leigh, M. Levenson, M. Vangel, D. L. Banks *et al.*, "Sp 800-22 rev. 1a. a statistical test suite for random and pseudorandom number generators for cryptographic applications," 2010.

Seddiq Q. Abd Al-Rahman Seddiq Qais Abd Al-Rahman is a lecturer in Computer Science Information Technology where he has been a college employee since 2007. He is now Rapporteur of the Computer Network Systems Department. From 2007–2016, He taught many laboratory materials and programming languages. Abd Al-Rahman completed his MSc degree at the University of Anbar at 2019 and his undergraduate studies at the same University. He teaches Computer Science in general, Cryptographic, Visual Programing and structure Programing. His areas of expertise interests are focused on the approaches for securing transfer data and store it with intelligent methods. He is also interested in handling with the smartphone applications.

Omar A. Dawood Omar A. Dawood was born in Habanyah, Anbar City, Iraq (1986). He got B.Sc. (2008), M.Sc. (2011) in Computer Science from the College of Computer and Information Technology, University of Anbar - Iraq. He was ranking the first during his B.Sc. and M.Sc. studies. He got the Ph.D. from the Computer Science Department, University of Technology-Baghdad - Iraq (2015). He is a teaching staff member and Director of Avicenna Division – Center for Continuing Education – University of Anbar. His research interests are: Data and Network Security, Coding, Number Theory and Cryptography.

Ali Makki Sagheer Ali Makki Sagheer was born in Iraq-Basrah 1979. He got on B.Sc. of Information System in Computer Science Department at the University of Technology (2001)-Iraq, M.Sc. in Data Security from the University of Technology (2004)-Iraq and Ph.D. in Computer Science from the University of Technology (2007)-Iraq. He is interesting in Cryptology, Information Security, Cyber Security Number Theory, Multimedia Compression, Image Processing, Coding Systems and Artificial Intelligence. He published more than seventy paper in different scientific journals and conferences twenty-four of them are published in Scopus. Finally, he obtained Professor scientific degree in Cryptography and Information Security since 18 Jul 2015.