Propose a Lightweight Block Cipher Algorithm for Securing Internet of Things

♦ Recepción/ 27 junio 2019
♦ Aceptación/ 25 agosto 2019

Propose A Lightweight Block Cipher Algorithm For Securing Internet of Things

Propuesto De Un Algoritmo De Cifras De Bloque Ligero Para Asegurar Internet De Las Cosas

Seddiq Q. Abd Al-Rahman[a,*],
Ali Makki Sagheer[b],
Omar A. Dawood[c]

[a] College of Computer Science and Information Technology, University of Anbar Anbar, Iraq, co.sedeikaldossary@uoanbar.edu.iq, ORCID: 0000-0002-6917-7352

[b] Al-Qalam University College Kirkuk, Iraq, ali.m.sagheer@gmail.com
[c]College of Computer Science and Information Technology, University of Anbar Anbar, Iraq. the_lionofclub@yahoo.com, Omar-Abdulrahman@uoanbar.edu.iq, ORCID: 0000-0003-3276-602X, ResearcherID: S-1401-2018

ABSTRACT/ This paper introduces a modern lightweight symmetric algorithm for securing the Internet of Things (IoT) applications. The lightweight cipher is considered a new trend of cryptographic design that aims at developing an elegant cipher with low cost construction for embedded devices. The proposed cipher designed to support a 64-bit secret key and encrypt a 64-bit block of electronic data. The proposed algorithm iterated with 24-rounds of Feistel network structure. The main round works as a product cipher that encompasses different layers similar to the round transformation of substitution-permutation network SPN structure. The main round includes four elementary stages S-box stage, shifting stage, MixColumn stage and adding keys stage. The first is a non-linear stage that undertakes the responsibility of increasing the confusion property. The shifting stage is designed in an easy and simple way to be convenient for restricted resources. The MixColumn stage is designed in accordance with half-byte oriented structure. The produced cipher has been tested according to several metrics and it has given accepted results. Also, it has passed the NIST statistical randomness tests with a typical time of implementation.
Keywords: Lightweight Cryptography, Block Cipher, Internet of Thing, Symmetric Cipher, Feistel Structure.
RESUMEN/ Este artículo presenta un algoritmo simétrico ligero y moderno para proteger las aplicaciones de Internet de las cosas (IoT). El cifrado ligero se considera una nueva tendencia de diseño criptográfico que tiene como objetivo desarrollar un cifrado elegante con una construcción de bajo costo para dispositivos integrados. El cifrado propuesto diseñado para admitir una clave secreta de 64 bits y cifrar un bloque de datos electrónicos de 64 bits. El algoritmo propuesto con 24 rondas de estructura de red Feistel. La ronda principal funciona como un cifrado de producto que abarca diferentes capas similares a la transformación redonda de la estructura SPN de la red de sustitución-permutación. La ronda principal incluye cuatro etapas elementales de pasantías S-box, pasantías cambiantes, pasantías MixColumn y agregar etapas de pasantías. La primera es una etapa no lineal que asume la responsabilidad de aumentar la propiedad de confusión. La etapa de desplazamiento está diseñada de una manera fácil y sencilla para ser conveniente para recursos restringidos. MixColumn está diseñado con una estructura orientada a medio byte. El producto ha sido probado de acuerdo con varias métricas. Además, ha pasado el NIST con un tiempo típico de implementación. Palabras clave: Criptografía ligera, Cifrado en bloque, Internet de las cosas, Cifrado simétrico, Estructura Feistel.

Introduction

The rapid development of communication and electronic applications has transformed the world into a digital village. Many contemporary services and applications are interleaved with our daily lives. Sensors and radiofrequency identification (RFID) technology are embedded in many aspects of modern techniques (e.g., access control, parking management, identification and goods tracking). Accordingly, a new cryptography trend is required in this kind of lightweight applications. Lightweight cryptographic protocols are characterised by limited computation ability, small storage space and constrained power consumption[1]. Therefore, traditional block ciphers (AES and Data Encryption Standard) are unsuitable for this kind of extremely constrained environment. Hence, the increasing demands for lightweight ciphers have received a lot of attention in recent years. Lightweight algorithms have prominent differences in numerous terms of design aspects[2]: Firstly, the algorithm should be suitable for constrained devices with small or accepted block size. Secondly, the design process must have hardware and software compatibility. Lastly, lightweight ciphers are generally executed in a hardware environment, and a small part of them are also executed on software platforms, like 8 bits microcontroller[3],[4]. Thus, performance of hardware is the essential consideration for lightweight ciphers. The efficiency of hardware could be calculated in various ways: the length of the critical path, clock cycles, latency, throughput, power consumption, and area requirements. Between them, the requirement of area is the most important parameter because small area requirements can efficiently minimise power consumption and the cost[5].

This paper is organised as follows: Section 2 explains the idea of the Internet of things (IoT). Section 3 introduces the lightweight block cipher. Section 4 presents the highlights of previous studies. Section 5 demonstrates the proposed algorithm design. Section 6 explains the analysis of the proposed algorithm.

The Internet of Things (IoT)

A developing variety of physical objects are being connected at an unparalleled charge to realise the concept of the IoT. The internet of different objects and network connectivity lets in objects to communicate and transformation information, consisting of sensors, smart phones, smart meters, smart vehicles, personal digital assistants, RFID tags and other items (software and actuators, embedded with electronics)[6]. The interconnection of those devices can be advanced IoT implementation (e.g., environment monitoring, product tracking, energy management and patient's surveillance) and develops the automation of our daily lifestyle. Example of the IoT applications is smart home, that can be inhabitants to open automatically their garage at access, prepare coffee start the air, condition and TV, control lights and other appliances[2]. IoT likewise make a progressively, substantial role in other fields (e.g., smart grid, smart city, electronic healthcare, industrial automation, intelligent transportation and disaster response). IoT is enable innovations which ease new interactions between human and 'things' and supply new application opportunities, services and infrastructures which get better our lifestyle quality. (Fig. 1) shows the potential applications of IoT[7]. As a significant expansion of cloud computing, some problems and questions will continue to arise, specially privacy and security issues. The fog computing is circulated by different untrusted fog computing service suppliers, the devices is putting at high risk. Fog computing nodes are faced with different privacy and security threats[8]. The devices of IoT have limited computing, battery and storage resources and are readily hacked, stolen or broken. The current solutions in cloud computing can be fared to address several privacy and security affairs in fog computing. However, specific security and privacy challenges remain due to distinctive features, such as decentralised infrastructure, mobility support, location awareness and low latency[9].

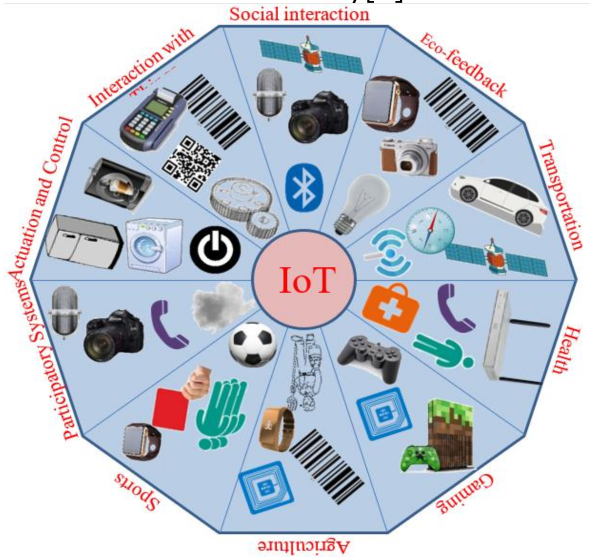

Figure 1. Potential IoT applications

Lightweight Block Ciphers

Lightweight cryptography is a modern approach that endeavour to provide solutions and develop efficient and fast security mechanisms for cruel limited resources environments. These solutions contain new designs in cryptographic protocols and primitives, as well as the adaption and modification of contemporary cryptosystems[10]. Three aspects are required to be optimised to design a lightweight cryptography: performance, security and cost[2],[11],[12]. Performance is calculated of the gross number of clock cycles to complete an operation, that is commensurate to the energy and throughput. Security is calculated through the bit number of a key. Increasing the size of the key results in higher security. Cost (expressed in terms of power or area) rely on the used architecture. Among the three aspects, a trade-off that create optimizing all of aspects with each other in one design extremely difficult is shown in Fig. 2 [13]. For instance, security is in a trade-off with cost and performance. To get high security have requires an increased cost or number of rounds. Cost and performance are two vertexes of this triangle. Serialised architecture harvests area and low power and results in low performance.

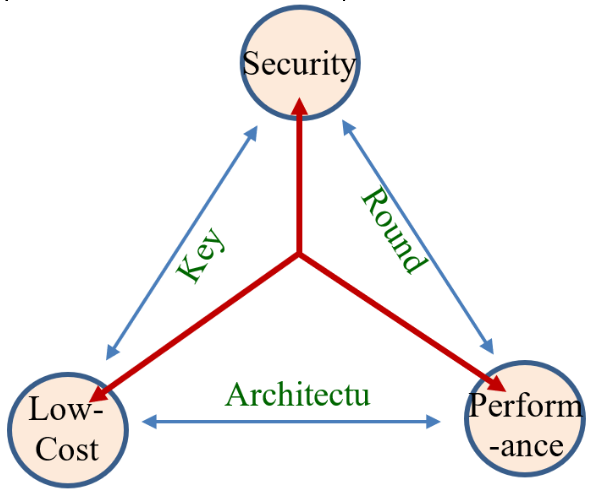

Figure 2. Triangle for lightweight cryptography

The technologies of cryptographic are advancing, and novel methods on attack, implementation and design are studied. One of the state of the art techniques is 'lightweight cryptography'[10],[14]. Lightweight cryptography is similarly designed as traditional cryptography in terms of two types: substitution–permutation network (SPN) structure and generalised Feistel network structure (GFN). SPN is modelled by including Galois field (GF) operations in each round, where GF operations produce apparent gibberish but can be mathematically inverted. It uses all data for encryption process in round functions (F). The F function includes a substitution box as a nonlinear layer, permutations as transformations of linear in the form of mappings based on maximum distance separable (MDS) and exclusive-OR (XOR) of key and data bits[15],[16]. In GFN, the input data is divided into two equally sized blocks (called left (L) and right (R)), frequently cycled through the algorithm. An F function is applied to the right block and the key at each cycle, and the result of the F function is XORed into the L. The blocks are exchanged between R and L. The XOR operation result makes the new R block, and the unaltered right block makes the L block[15],[17]. The procedures are repeated in a rounds number.

Related Works

Many scholars have examined lightweight block ciphers in the cryptography field, focusing on the design of lightweight block ciphers, security analysis and performance evaluation. These studies have achieved favorable results in the aspect of lightweight block cipher design. A number quantity of lightweight block ciphers has also been proposed. This section presents the findings of some significantly related works that deal with data encryption:

- DES[18] is a block cipher that uses 64 bits as the plaintext block data. It is based on a symmetric key that supports 56 bits as a secret key and uses 16 rounds for data encryption. DES is the base in the Feistel structure. The first step is expanding the F function to 48 bits by using a fixed table. The second step is utilising XOR data with a key round. The third step uses eight S-boxes, and the final step is permutation to return the size in 32 bits. The eight S-boxes are designed to change to 6 bits, sized 4 × 6 bits each. The 56-bit secret key is selected from the initial 64 bits as the input in the key schedule. The 56 bits are halved (i.e., 28 bit each), and both halves are rotated left by one or two bits and then merged and shrunken to 48 bits as fixed tables.
- DESL[19] is DES in a lightweight type, that utilized one S-box mapping only, making the DES more compact (which uses eight S-box mappings). DESXL is a DESL type with a new improvement using a 184 bits as key size. However, the efficient key size is approximately 118 bits. In addition, 1,850 GEs are used in DES, compared with the 2,310 GEs in DESXL.
- ITUbee[20] is a block cipher that uses 20 rounds of a Feistel structure. Its use of a Feistel function based on a small SPN is reminiscent of the GFN. The key length

and block size of ITUbee are 80 bits. The whitening key is added to the top and bottom parts of the cipher. The key scheduling includes rounding a constant in a fashion similar to LED or PRINCE. The F function consists of an S-box layer that is the S-box of AES and uses the XOR between fixed bits as the permutation layer.

- LBlock[21] is a GFN structure that uses an 80-bit key size to encrypt 64 bits in 32 rounds. The F function utilizes a SPN, which confusion layer composed of 4 × 4 S-boxes and diffusion layer composed of a simple 4 bits word permutation in the left part. The right part makes an 8-bit left cyclic shift operation. The key scheduling generated 32 subkeys at 32-bit size. The operation consists of 28-bit left cyclic shifting, S-boxes, and XOR with previous subkeys to generate it. The software implementation on the 8-bit microcontroller requires approximately 3,955 clock cycles to encrypt a plaintext block.
- TWINE[22] is a GFN structure with 16 4-bit branches. The F function (8 times per round) is XOR-mixed with a subkey and a 4 × 4 S-box. The key schedule itself is also a GFN. The permutation requires only half as much rounds as a circular shift for one subblock difference to diffuse to all the subblocks. The S-box is based on the inverse function of GF
- GRANULE is used the GFN, it is encrypt data with 64-bit as size by secret key (80/128)-bit size with 32 rounds. The function is a merge of three layers permutation, an S-box and shift. The first layer is permutation, it is doing in the F function that is block permutation sized 4 bits each. The S-box layer (nonlinear layer) is utilize a single 4 bits S-box. The right and left circular shifts are used in the F function. The key scheduling execute is stimulated by using key schedule in the PRESENT algorithm.
- QTL is a 64-bit block cipher supported by 64/128 bit keys and used similar GFN structure in iterative rounds as 16/20. QTL structure has a rapid diffusion of the SPN structure, that does not use a key schedule. QTL has two active S-boxes on easy bounds through the encryption process. The data are divided into four equal parts: the first and third parts were inputted to the F function and then switched each two adjacent and second switch between the first and third parts. The function includes AddConstants → AddRoundKey → S-box layer → P-permutation layer → S-box layer.
- RoadRunneR is based on GFN structure, it is encrypted data block with a 64-bit by secret key length 80/128-bit. The 80-bit and 128-bit keys require 10 and 12 rounds, respectively. XOR is used to add a whitening key before and after the left part of the divided data. The F function consist of shifting bits with a fixed table, an S-box with one 4 × 4 bit, and shifting bits with inverse first shifting, diffusion layer and XOR with key. The key schedules use the same simple characterization for both key sizes: Start from the beginning of the master key whenever a new 32-bit key material is wanted and then continue on the master key in a circular way.
- KLEIN is a lightweight algorithm based on SPN structure for encryption / decryption data block 64-bit as size by secret key size 64/80/96-bit with 12/16/20 as rounds number. It merges the operations of PRESENT and AES. The nonlinear layer uses a S-box as size 4 × 4 bits. In the design of the key schedule, it is using a Feistel structure to avoid possibility key attacks. A chosen-plaintext key-recovery attack is present up to 8 rounds of KLEIN-64.
- HISEC is a Feistel lightweight block cipher. HISEC is use a 64-bit as a data block and 80 bits as a secret key with 15 rounds number. The features of HISEC are like as those of PRESENT algorithm, but the permutation layer in HISEC is various from PRESENT algorithm. HISEC is utilized on two sides, each one is equal 32-bit. There are four layers in HISEC, it is in each one as XOR with the key. It enables confusion, the first layer, by performing nonlinearity to the algorithm and utilizing a one 4 × 4 bits S-box, this layer is repeat the S-box 16 times. The following layers are bit permutation as diffusion, rotation bits and XOR. The ultimate right 64-bit of secret key are used for the encryption / decryption algorithm. HISEC cipher is safe against integral, differential and boomerang attacks.

- DLBCA[28] a new lightweight algorithm using GFN structure .DLBCA is designed with a 32-bit block size and supported a secret key size 80 bits with 15 rounds. DLBCA stratifies four layers: S-box, bit-permutation, EX-OR rotation and Adding key. DLBCA supplies resistance to boomerang and differential attacks.

The proposed cipher depends mainly on several published algorithms that paved the road in front of design the main stages. The proposed cipher also inherited some good features from these successors ciphers that involved a new trend in cryptographic algorithms.

Definition and Design Rationale of the Proposed Algorithm

The structure of the developed cipher is a balance Feistel network with 24 rounds. The proposed algorithm is utilizing 64-bit as length of secret key to encrypt / decrypt 64-bit as a block data. In each round of the proposed cipher, it is used a 32-bit as round key Rki, which is generated from a 64-bit master key register and is used XOR operation with the half plaintext, (see Fig. 3). The introduced cipher uses the following notations:

PT	64-bit input plaintext block
CT	64-bit output ciphertext block
K	64-bit key register
Rk_i	32-bit sub keys for round i
F	Function
$\oplus$	Bitwise exclusive-OR operation
<<<<n	Left cyclic shift by n bits
>>>>n	Right cyclic shift by n bits
$[i]^2$	Binary representation of integer i
||	Concatenation of two strings
!	Bitwise NOT operation

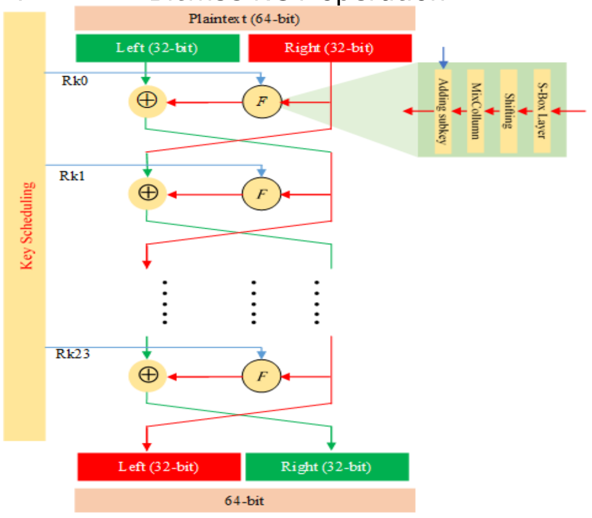

Figure 3. The Proposed Lightweight Cipher with Feistel structure

The F function in the main round is a combination of the layers of the SPN structure operators. The layers are S-box, shifting, MixColumn and adding a subkey Rki operators. The nonlinear operator is a single 4-bit S-box. The shifting operator uses fixed equations, and MixColumn is dependent on GF(2). The right and left circular shifts of the F function are used. A pseudocode for the cipher is given below:

```
Generate Round Subeys()
$PT = PT^L || PT^R$
for $i = 0$ to 23 do
S-Box($PT^R$)
Shifting($PT^R$)
MixCollumn($PT^R$)
Adding_Subkey($PT^R$,$Rk_i$)
$PT^L = PT^R$
$PT^R = PT^L$
end for
$CT = PT^R || PT^L$
```

Encryption Algorithm

The encryption process works as follows, a 64-bit plaintext PT is halved (PT^L and PT^R), each of which has a 32-bit size:

$PT \leftarrow PT^R \,||\, PT^L$

The encryption process is described as sequences of smart steps that can be summarized below:

1. Apply F function on the 32-bit word PT^R for i = 0 to 23 and XOR with PT^L and key Rk_i:

$PT \leftarrow PT^L \oplus F\,(PT^R) \oplus Rk_i$

2. The plaintext will be interchanged as shown in Fig. 3:

$PT^L \leftarrow PT^R$

$PT^R \leftarrow PT^L$

When finished 32 rounds, the cipher text (CT) 64-bit is acquired, which is a series of PT^R and PT^L:

$CT \leftarrow PT^R \,||\, PT^L$

F Function

Fig. 4 presents the F function of the round transformation. The function uses 32 bits input PT^R and makes 32 bits output:

$F: \{0, 1\}^{32} \leftarrow \{0, 1\}^{32}$

Four various operations in the F function are used: S-box, shifting, MixColumn and adding a subkey Rki operators. All these operations take 32-bit as input and makes 32-bit as output. These functions are described in the following sections.

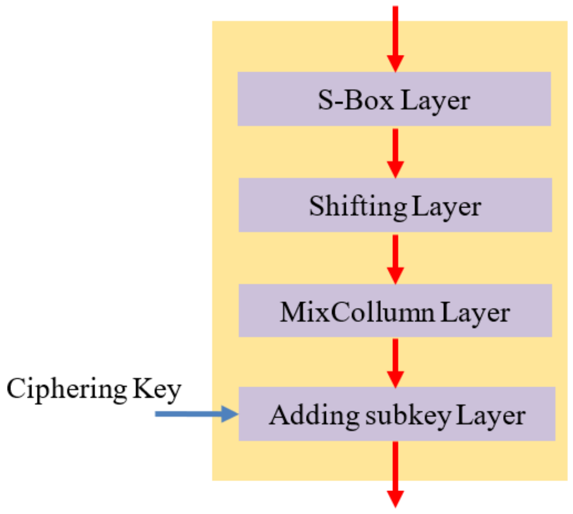

Figure 4. The Main F function of the Proposed Cipher

1. S-box Operation

The energy per cycle is evident in the analysis provided by[29]. The design that used a 4 × 4 bits S-box is found more effective than that of the 8 bits S-box because the latter will have a higher signal postponement than the former. However, 4-bit S-boxes give low nonlinearity and high rates of the distribution panel / lighting panel (DP/LP) coefficient. A design utilizing a 4 × 4 bits S-box requests an implementation of the F function to save security margins similarly. All 4 bits are converted into one hexadecimal and inserted into the S-box to deal with the hexadecimal numbers. The one type of objective 4 × 4 bits S-boxes in Midori algorithm[30] (As in Table 1.) can be utilized in accordance with the explanation above:

Sb: $\{0,1\}^4 \rightarrow \{0,1\}^4$

Table 1: The Hexadecimal Number of 4 × 4 bits S-box SB

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

2. Shifting Operation

The purpose of the shift bit in shifting operation is to distribute the bits of each input bits to various output bits (i.e., linear diffusion process). The shifting operation is applied on the row of PTR 32 bit. The rotation uses a fixed change location, (see Fig 5). Table 2 discusses fixed changing bits between the input and output.

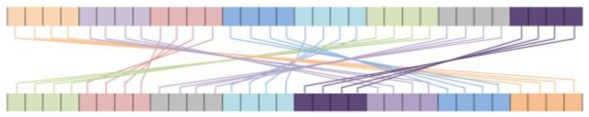

Figure 5. Shifting operation

Table 2.
Change of bits in the shifting operation

New Bit	0	1	2	3	4	5	6	7
Old Bit	31	30	29	28	23	22	21	20
New Bit	8	9	10	11	12	13	14	15
Old Bit	7	6	5	4	27	26	25	24
New Bit	16	17	18	19	20	21	22	23
Old Bit	15	14	13	12	3	2	1	0
New Bit	24	25	26	27	28	29	30	31
Old Bit	11	10	9	8	19	18	17	16

3. MixColumn Operation

MixColumn is a bricklayer permutation operating on the state column by column. This operation has the property that makes linear and differential attacks harder. The PT^R 32 bits must be converted to hexadecimal number for the MixColumn operation to work. Eight hexadecimal values can make a 4 × 2 matrix. Now, MDS matrix, which is a matrix representing a function with certain diffusion properties that have useful applications over $GF(2^n)$ in finite field, is used by the MixColumn operation. MDS contains a half byte. It returns to its original size as the input in the output matrix: (4 × 4) × (4 × 2) = (4 × 2).

$$\begin{bmatrix} S'_{0,x} \\ S'_{1,x} \\ S'_{2,x} \\ S'_{3,x} \end{bmatrix} = \begin{bmatrix} 7\,5\,6\,5 \\ 5\,7\,5\,6 \\ 6\,5\,7\,5 \\ 5\,6\,5\,7 \end{bmatrix} \otimes \begin{bmatrix} S_{0,x} \\ S_{1,x} \\ S_{2,x} \\ S_{3,x} \end{bmatrix}$$

Key Scheduling

The key schedule takes a 64-bit key as input, which is stored in a key state register K = (k_0 … k_{63}). The round key Rk_i at every round is derived from three steps to be generated: left cyclic shift, S-box and bitwise NOT. The pseudocode for the key scheduling is given below:

```
K = k0 … k63
for i = 0 to 23 do
<<<12
S-Box(K)
```

```
! (K)
Rki ← K
end for
```

The first step in the key is a left cycle shifting with 12 bits and then sending to S-box to convert bits into a nonlinearity criterion. The final step is a bitwise NOT to reverse bit value. Thereafter, K is saved as a subkey and used to generate the other keys.

Decryption Algorithm

The decryption algorithm is the invert of the encryption execution. The generation subkeys are created in invert arrangement through a key schedule utilizing the operation round keys. The decryption process step could be performed to have a similar the encryption process as architecture for it. The rounds of algorithm must be similar to the encryption algorithm where the processes are implement in each round. The operations are performed on the left side of the plaintext, unlike the encryption process. The operations in the F function are made in the same order and values as in the encryption process but different processes in each operation. The same S-box is used in the S-box operation because it is self-inverse. The bits in shifting operation must be returned with the lines used in the encryption operation. Table 3 shows the new location bits.

Table 3.
Shifting operation in the decryption process

New Bit	0	1	2	3	4	5	6	7
Old Bit	23	22	21	20	11	10	9	8
New Bit	8	9	10	11	12	13	14	15
Old Bit	27	26	25	24	19	18	17	16
New Bit	16	17	18	19	20	21	22	23
Old Bit	31	30	29	28	7	6	5	4
New Bit	24	25	26	27	28	29	30	31
Old Bit	15	14	13	12	3	2	1	0

MixColumn operation uses a reversible MDS. The process is self-reversible, and the equation is given as follows:

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

Algorithm Analyses

The proposed algorithm is designed with the size in plaintext and key in proportion to the specific hardware resources where IoT is considered one of its most important applications. The 64-bit key that makes a powerful analytical process required an ample time (high key agility). This cipher utilizes an model, in the form of key dependent on S-boxes, that make an abnormal dependency among the structure of the key schedule and the security of the algorithm. Thus, possible attacks are prevented. An efficient combination of S-box and MixColumn operations was proposed and used in the same forward and backward manners, to use less resources than separated block implementations. The proposal that used Feistel structure is time efficient and projects better quality of operation to save the strength of the algorithm. NIST statistical tests are used to know the strength of the proposed performance. The NIST test suite is a statistical package be composed of several tests which were developed to test the randomness of (arbitrarily long) binary sequences produced by either hardware- or software-based cryptographic random or pseudorandom number generators. These tests focus on different types of nonrandomness that can exist in a sequence. All the tests applied to the algorithm successfully passed the ciphertext test, and Table 4 shows the test results.

Table 4: NIST test results

Test Name		Proposed Algorithm
Frequency (monobit) test		0.073100
Frequency test within a block		0.815961
Runs test		0.134389
Cumulative sums (Cusum) test	REVERSE	0.137790
	FORWARD	0.196863
Test for the longest run of ones in a block		1.000000
Serial test	P-v1	0.113246
	P-v2	0.419500
Approximate entropy test		0.297463

The algorithm is achieved through the criterion tests by computer Intel Core i7-4600u at 2.10 GHz processor using Visual Studio .NET C# language with a total execution time of 0.002003 ms. The proposed algorithm is worked with 64-bit, that meaning the probability of (2^{64}) is equal to 18446744073709551616. So, the attackers will spend much time and they will have complexity to broken the ciphertext.

Conclusion

A new symmetric lightweight algorithm of 64-bit ciphering secret key has been proposed. The suggested cipher encrypts the electronic data of 64-bit across an iterated of 20-rounds feistel structure. The internal round transformation designed with four involutional stages that encrypt and decrypt data with the same invertible operations. The proposed structure works to balance among the triples constraints of the lightweight design Cost, Performance and Security factors. The proposed cipher submitted a reasonable security level and high-speed implementation with low amount of memory requirement. The developed structure design to work with restricted environment and low-cost of hardware to fit the IoT applications. The evolved cipher has been tested and measured according to several impact metrics and it gave satisfactory results.

References

[1] E. Jacinto, H. Montiel, and F. Martinez, “High Performance Optimization Function for 32-Bits Microcontrollers in Key Scheduling of the Lightweight Cipher Algorithm CLEFIA,” Indian J. Sci. Technol., vol. 10, no. 15, pp. 1–5, 2017.

[2] S. Singh, P. K. Sharma, S. Y. Moon, and J. H. Park, “Advanced lightweight encryption algorithms for IoT devices: survey, challenges and solutions,” J. Ambient Intell. Humaniz. Comput., vol. 0, no. 0, pp. 1–18, 2017.

[3] B. J. Mohd and T. Hayajneh, “Lightweight block ciphers for IoT: Energy optimization and survivability techniques,” IEEE Access, vol. 6, no. c, pp. 35966–35978, 2018.

[4] S. N. Swamy, D. Jadhav, and N. Kulkarni, “Security threats in the application layer in IOT applications,” Proc. Int. Conf. IoT Soc. Mobile, Anal. Cloud, I-SMAC 2017, pp. 477–480, 2017.

[5] S. Li, T. Tryfonas, and H. Li, “The Internet of Things: a security point of view,” Internet Res., vol. 26, no. 2, pp. 337–359, 2016.

[6] A. Whitmore, A. Agarwal, and L. Da Xu, “The Internet of Things—A survey of topics and trends,” Inf. Syst. Front., vol. 17, no. 2, pp. 261–274, 2015.

[7] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, “Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications,” IEEE Commun. Surv. Tutorials, vol. 17, no. 4, pp. 2347–2376, 2015.

[8] H. E. Hudson, V. Forsythe, and S. G. Burns, “Fog Computing for the Internet of Things Security and Privacy Issues,” IEEE Internet Comput., vol. 21, no. 2, pp. 34–42, 2017.

[9] J. Ni, K. Zhang, X. Lin, and X. S. Shen, “Securing Fog Computing for Internet of Things Applications: Challenges and Solutions,” IEEE Commun. Surv. Tutorials, vol. 20, no. 1, pp. 601–628, 2018.

[10] P. Nandhini, V. Vanitha, and P. Scholar, “A Study of Lightweight Cryptographic Algorithms for IoT,” Int. J. Innov. Adv. Comput. Sci. IJIACS ISSN, vol. 6, no. 1, pp. 2347–8616, 2017.

[11] C. Pei, Y. Xiao, W. Liang, and X. Han, “Trade-off of security and performance of lightweight block ciphers in Industrial Wireless Sensor Networks,” Eurasip J. Wirel. Commun. Netw., vol. 2018, no. 1, 2018.

[12] M. A. Philip and V. Vaithiyanathan, “A survey on lightweight ciphers for IoT devices,” Proc. 2017 IEEE Int. Conf. Technol. Adv. Power Energy Explor. Energy Solut. an Intell. Power Grid, TAP Energy 2017, pp. 1–4, 2018.

[13] S. B. Sadkhan and A. O. Salman, “A survey on lightweight-cryptography status and future challenges,” Int. Conf. Adv. Sustain. Eng. Appl. ICASEA 2018 - Proc., pp. 105–108, 2018.

[14] T. Eisenbarth, L. Uhsadel, S. Kumar, C. Paar, and A. Poschmann, “A Survey of Lightweight- Cryptography Implementations,” pp. 0–11, 2007.

[15] D. Sehrawat and N. S. Gill, “Lightweight Block Ciphers for IoT based applications : A Review,” Int. J. Appl. Eng. Res., vol. 13, no. 5, pp. 2258–2270, 2018.

[16] O. A. Dawood, A. M. S. Rahma, A. Mohssen, and J. A. Hossen, "The Euphrates Cipher," vol. 12, no. 2, pp. 154–160, 2015.
[17] M. Kumar, S. Pal, and A. Panigrahi, "FeW: A Lightweight Block Cipher.," IACR Cryptol. ePrint Arch., 2014.
[18] E. Biham and A. Biryukov, "An Improvement of Davies' Attack on DES," \ifnum\shortbib=1{EUROCRYPT}\else{ Advances Cryptol. -- {EUROCRYPT}}\fi'94, vol. 950, no. 10, pp. 461–467, 1997.
[19] S. Jana, J. Bhaumik, and M. K. Maiti, "Survey on Lightweight Block Cipher," Int. J. Soft Comput. Eng., vol. 3, no. 5, pp. 183–187, 2013.
[20] A. Baysal and S. Şahin, "RoadRunneR: A Small And Fast Bitslice Block Cipher For Low Cost 8-bit Processors," in LightSec 2015, 2016, vol. LNCS 9542, pp. 58–76.
[21] J. M. Kizza, "LBlock: A Lightweight Block Cipher," Guid. to Comput. Netw. Secur., no. 1, pp. 339–354, 2013.
[22] T. Suzaki, K. Minematsu, S. Morioka, and E. Kobayashi, "Twine: A lightweight, versatile block cipher," ECRYPT Work. pn Light. Cryptogr. LC11, pp. 146–169, 2011.
[23] G. Bansod, A. Patil, and N. Pisharoty, "GRANULE: An Ultra lightweight cipher design for embedded security," IACR Cryptol. ePrint Arch. 2018, vol. 600, 2018.
[24] L. Li, B. Liu, and H. Wang, "QTL: A new ultra-lightweight block cipher," Microprocess. Microsyst., vol. 45, pp. 45–55, 2016.
[25] A. Baysal and S. Şahin, "RoadRunneR: A Small And Fast Bitslice Block Cipher For Low Cost 8-bit Processors," in LightSec 2015, 2016, vol. LNCS 9542, pp. 58–76.
[26] Z. Gong, S. Nikova, and Y.-W. Law, "KLEIN: A New Family of Lightweight Block Ciphers," 7th Work. {RFID} Secur. Priv. {(RFIDSec '11)}, vol. 7055, pp. 1–18, 2011.
[27] S. S. M. Aldabbagh and I. F. T. Al Shaikhli, "HISEC: A new lightweight block cipher algorithm," Proc. - 3rd Int. Conf. Adv. Comput. Sci. Appl. Technol. ACSAT 2014, pp. 15–20, 2014.
[28] S. Salim and M. Aldabbagh, "Design 32-bit Lightweight Block Cipher Algorithm (DLBCA)," Int. J. Comput. Appl., vol. 166, no. 8, pp. 17–20, 2017.
[29] G. Leander and A. Poschmann, "On the Classification of 4 Bit S-Boxes," Arith. Finite Fields, pp. 159–176, 2007.
[30] S. Banik et al., "Midori: A block cipher for low energy," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 9453, pp. 411–436, 2015.