A Lightweight Block Encryption Algorithm for Narrowband Internet of Things

guosheng zhao
Harbin Normal University
huan chen (luckych2021@163.com)
Harbin Normal University
Jian Wang
Harbin University of Science and Technology

Research Article

Keywords: NB-IOT, ARX structure, SPN structure, Lightweight block encryption

Posted Date: September 22nd, 2022

DOI: https://doi.org/10.21203/rs.3.rs-2033728/v1

A Lightweight Block Encryption Algorithm for Narrowband Internet of Things

Guosheng Zhao[1], Huan Chen[1*], Jian Wang[2]

Abstract

Aiming at the security of terminal data information in the application architecture of narrowband internet of things (NB-IOT), this paper proposes a lightweight block encryption algorithm SPNRX based on variant ARX (And-Rotate-XOR) structure and Substitute Permutation Network (SPN) structure. The proposed algorithm takes into account the unique security requirements of the NB-IOT terminal, such as simplifying the encryption process as far as possible, not too long encrypted data and high security level, and reduces the number of encryption rounds. It overcomes the shortcoming that half of the block information of ARX structure does not change during one round of encryption, so as to improve the diffusion speed. In addition, considering the security of keys and the cost of hardware implementation, a key schedule based on matrix transformation and P-box permutation is proposed. Through the security analysis of SPNRX algorithm, the analysis shows that SPNRX algorithm has resistance to differential analysis, linear analysis and so on. Finally, the hardware and software performance of SPNRX algorithm is tested. The experimental results show that the hardware implementation cost of the proposed algorithm is low, only 1093 GE based on 0.13 micron ASIC process is required. And the software implementation of the proposed algorithm performs well. The encryption and decryption of the proposed algorithm on the 64-bit processor takes about 0.763 milliseconds and 0.772 milliseconds. And it requires 41.06MB of memory resources. In addition, the security strength and encryption effect of the algorithm are tested through key sensitivity, information entropy, etc. The results show that the proposed algorithm has high enough security.

Keywords: NB-IOT; ARX structure; SPN structure; Lightweight block encryption

1 Introduction

With the rapid development of Internet of Things technology, mobile communication is developing from the connection between people to the realization of the interconnection between people and things and between things. The development between the Internet of Things and wireless communication technologies is interrelated. With the continuous growth and diversification of the business requirements of the Internet of Things, the NB-IOT emerges as the times require.

NB-IOT is a wireless communication technology for Low Power Wide Area Network (LPWA) proposed by the 3rd Generation Partnership Project (3GPP) standard organization[1]. It is a new Internet of Things access technology based on the Internet of Things, which now has become an important branch of the Internet of everything. Since it was proposed, NB-IOT has been characterized by low cost, low power consumption, large connection, wide coverage and deep penetration[2], which makes up for the shortcomings of existing LPWA technology that cannot be deployed over long distances and large areas, and meets the increasingly complex market requirements of the Internet of Things.

Therefore, NB-IOT technology has developed rapidly and is widely used in many application fields, such as eight typical industries, including healthcare, public utilities, consumers, smart cities, agricultural environments, smart buildings, logistics and warehousing, and manufacturing[3]. However, security issues often appear after large-scale applications. Due to the lack of security awareness of device manufacturers and the lag of security standards, personal privacy data and corporate private information generated by NB-IOT-based applications have been greatly threatened.

In the application process of NB-IOT, the application architecture process is abstracted as: the terminal connected to the NB device is expected to generate a large amount of data, and the data is sent to the cloud platform through the network communication center, and then the cloud platform forwards it to the human-machine Interactive platform, people can use tablet computers, general-purpose computers and other devices to obtain terminal data in the interactive system.

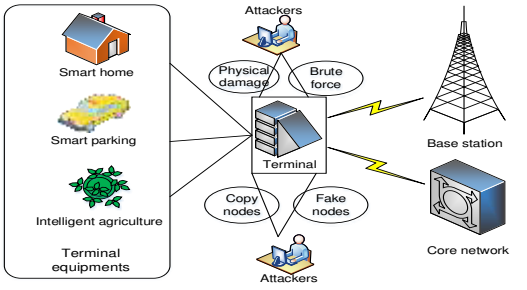


Fig. 1 NB-IOT terminal security issues.

Because NB-IOT is an open network, it has the characteristics of large number of connected devices, diversified business and wide service scope [4]. Therefore, NB-IOT terminals need a large number of terminal sensing nodes to connect these diverse devices.

☒ Huan Chen
luckych2021@163.com

1 College of Computer Science and Information Engineering, Harbin Normal University Harbin 150025, China

2 School of Computer Science and Technology, Harbin University of Science and Technology 150001, China

However, sensing nodes lack self-protection measures and are vulnerable to malicious external attacks such as message theft, data tampering and node replication, which poses a great threat to data security, as shown in Fig. 1. It can be concluded that the diversity and heterogeneity of NB-IOT terminal devices can greatly affect the overall security level of the Internet of Things [5].

From the perspective of application development, developers can only design the software and hardware of the NB-IOT terminal and human-computer interaction platform. Due to the limited computing capacity of NB-IOT terminals, and the large number of low-power devices involved in NB-IOT terminals, the processing capacity and memory capacity of terminals are very limited due to the cost limitation. If the traditional security algorithm is used, it requires a large computation cost and occupies a high storage resource, which damages the performance of these devices. Therefore, consider using a lightweight block cipher with a low resource footprint. The main contributions of this paper are as follows:

(1) In order to ensure the security of NB-IOT terminals in application scenarios, a lightweight block cipher algorithm SPNRX is proposed. SPNRX algorithm has the same length of plaintext and ciphertext, so it does not cause extra overhead for NB-IOT network traffic. The number of encryption rounds is 16 rounds, and the encryption process is simple, which can reduce the computational burden of the NB-IOT terminal. In addition, when evaluating the security and operational efficiency of the proposed algorithm, the results show good performance.

(2) In order to enhance the diffusion effect of the cipher, the proposed encryption algorithm combines the variant ARX structure and the SPN structure. This combination method can overcome the shortcomings of the ARX structure that half of the block information remains unchanged during one round of encryption and the security is limited. In addition, it can accelerate the diffusion speed and reduce the number of encryption rounds. And the variant ARX structure replaces the modulo-add component in the ARX structure with a simple logic AND operation, which can reduce the computational complexity and cost.

(3) In order to make it difficult for attackers to crack the key, a new key schedule is proposed. This scheme obtains the round key in the encryption process through matrix transformation and P-box permutation. The sub-keys of each round are different, making the key not easy to be attacked. And in terms of hardware implementation, P replacement only needs to be completed by wiring, and does not require hardware consumption resources, which is relatively easy to implement and reduces hardware consumption resources.

The rest of this paper is organized as follows: Section 2 summarizes the research work related to NB-IOT; Section 3 introduces the proposed lightweight encryption algorithm SPNRX; Section 4 provides a security analysis of the proposed algorithm; Section 5 is the performance test results and analysis of the proposed algorithm; Section 6 is the summary of this paper and the prospect of future research.

2.Related work

The NB-IoT application architecture consists of four parts: the terminal, the communication center, the cloud platform and the human-computer interaction platform. Data transmission from the terminal to the human-computer interaction platform needs to pass through the NB-IoT base station, core network, and various NB-IoT networks. Therefore, there may be attackers in every link of the data transmission process, and these security issues will restrict the development of NB-IOT. However, because the security threats of data in the process of NB-IoT network transmission are mainly solved by operators, and the application server has a certain generality, the relevant security mechanisms can be directly applied by them. Therefore, the object that needs to be focused on is the NB-IOT terminal. This chapter will mainly analyze the potential security problems of terminals in NB-IOT application scenarios, the encryption algorithms used and their unique requirements for encryption algorithms, and analyze the existing lightweight encryption algorithms.

2.1 NB-IOT security issues

The establishment of the NB-IOT standard in 2016, there have been many studies on the technical analysis, characteristics, energy consumption, application prospects, and development trends of NB-IOT on its own application technology and communication characteristics [6-7]. But there are only a handful of security studies on NB-IOT. Sun Zhixin et al. studied the security requirements of the perception layer, network layer and processing layer respectively, and then proposed a three-layer security architecture based on NB-IOT [8]. Shi Jianbing et al. analyzed the security issues of NB-IOT at the technical level and beyond technical issues [9]. According to the characteristics and security requirements of NB-IOT, Wang Baocheng et al. studied the possible security risks in the process of data transmission from NB-IOT devices to the platform and then to the management system [10].

Because NB-IOT terminal involves a large number of devices, such as smart meters, gas meters, smart street lights, etc., and terminal devices are generally deployed in application scenarios such as electricity meter reading, gas meter reading, and smart cities. In these scenarios, the terminal nodes of NB-IOT are often easily exposed, and the terminal devices cannot be effectively protected and monitored by humans. Therefore, NB-IOT terminals are prone to many security problems, such as terminal nodes are vulnerable to physical damage, brute force cracking, node replication, etc. If the NB-IOT terminal device is attacked and damaged, the secret interface inside the device will be exposed, making the key data information of the device at risk of being stolen. The data transmitted by the device is often private data or important information generated by NB-IOT users.

Many device manufacturers often only consider the realization function of the device when designing and manufacturing the device, but do not consider the protection of data security. Therefore, in order to ensure the security of the data transmission process and prevent the private data of the device from being stolen and analyzed by an attacker, the data needs to be encrypted.

2.2 NB-IOT encryption algorithm

In the field of security, it is a common security method to encrypt data by using a key encryption algorithm. Using encryption algorithms to encrypt plaintext data into chaotic ciphertext data can greatly reduce the possibility of data being stolen or leaked, thereby ensuring secure data communication. As we all know, encryption algorithm has always been one of the key technologies to ensure data security in IoT applications, and it is a research hotspot of terminal security [11-12]. Although NB-IOT is an emerging technology based on the Internet of Things, it has many similarities with the traditional Internet of Things, but NB-IOT terminals have their own unique performance and working methods. Therefore, it is possible to directly misappropriate the previous encryption algorithm. will not work well.

Through the synthesis of a large amount of data, it is learned that the NB-IOT terminal has the characteristics of limited energy and computing storage capacity, limited data length, periodicity and asymmetry of data transmission. Therefore, NB-IOT has its own unique requirements for the design and selection of encryption algorithms, including the encryption process being as simple as possible, the encrypted data packets being short and the security level being sufficiently high [14]. At present, there are only a few lightweight encryption algorithms proposed based on NB-IOT. For example, Jia Rongyuan et al. proposed a lightweight encryption algorithm based on AES and chaotic sequence, which improved the operation efficiency by reducing the number of AES encryption rounds and merging round functions [15]. It can be seen that the existing research is not enough to meet the security requirements of NB-IOT.

2.3 Lightweight encryption algorithm

Nowadays, many scholars in the field of cryptography have carried out research on lightweight block ciphers. In the design and implementation of lightweight block ciphers, researchers have achieved good results, and proposed some lightweight block cipher algorithms that can not only better ensure device security, but also achieve faster execution efficiency. These lightweight block cipher algorithms can usually be attributed to the following structures: SPN, Feistel network, generalized Feistel network (GFN), Addition-Rotation-Xor(ARX) and nonlinear feedback shift register (NLFSR) , as shown in Table 1.

Table 1 Lightweight block cipher structure types.

structure	Algorithm
types	names
SPN	AES、PRESENT、KLEIN、PRINT、Rectangle、mCrypton、Midori、PRINCE
Feistel	DES、TEA、Simon、LBlock、SEA、MIBS、ITUbee、Few、GOST
GFN	CLEFIA、Piccolo、TWINE
ARX	HIGHT、IDEA、Speck、LEA
NLFSR	KATAN/KTANTAN

The SPN structure can transform the data through the S box and the P box, and the diffusion speed is fast, but the structure of encryption and decryption is inconsistent, which increases the cost of decryption. The Feistel structure divides the data into two blocks of the same size, and then encrypts them. The generalized Feistel structure is a generalization of Feistel, which can divide data into several blocks. Feistel has the same encryption and decryption structure, but all blocks information cannot be changed in each round of encryption, and more rounds are required, which will increase the energy consumption of the Feistel structure [16]. The ARX structure only uses addition, rotation and XOR for encryption and decryption. The structure is relatively simple, and the implementation is fast and compact, but it is not as secure as the SPN and Feistel structures [17]. NLFSR is derived from stream ciphers, which use stream ciphers to construct blocks.

In the application of the Internet of Things, considering the limited resources of terminal nodes, its security research has always focused on lightweight encryption algorithms that occupy less resources [18-19], such as RC5, Skipjack, TEA, and improved algorithms based on AES. Among them, RC5 algorithm has an obvious deficiency, which takes up a large amount of storage space when it performs key initialization [20]. Although Skipjack algorithm meets the requirement that NB-IOT encrypted data should not be too long, its security level is low [21]. TEA algorithm can be applied to the equipment with low cost and weak computing power. It is simple but vulnerable to attack and has been cracked [22]. Zhang Yao et al. and Zhao Yuehua et al. proposed an improved AES algorithm with the number of encryption rounds reduced, which improved the operation efficiency of the algorithm, but reduced the security [23-24]. Chen D et al. and Yan Lele et al. proposed an improved AES algorithm combined with chaotic sequences, which used chaotic sequences to generate round keys, thus improving the security of the algorithm [25-26]. However, none of the lightweight encryption algorithms mentioned above can simultaneously meet the special requirements of NB-IOT algorithm.

3.Algorithm structure

Considering the cost and computational complexity of addition component in traditional ARX structure, this paper designs a hybrid lightweight block cipher SPNRX by replacing the addition component with simple logic operation, and then combining the variant ARX structure and SPN network structure. SPNRX is a 64-bit lightweight block cipher with a 96-bit key length and 16 iteration rounds. In the following, we will

introduce the encryption and decryption algorithm and key extension algorithm of SPNRX in detail.

3.1 Symbol Description

The interpretation of the symbols involved in the text is shown in Table 2.

Table 2 Symbol Explanation.

Symbol name	meaning
X	64-bit plaintext
X_j	the j-th plaintext
C	64-bit ciphertext
K	64-bit master key
K_i	i-th round sub-key
F	round function
$<<< i$	rotate left by i bits
$\oplus$	bitwise XOR
$\|\|$	concatenation of two binary strings
P	P -box
S	S-box

3.2 Encryption Algorithm

SPNRX is a symmetric key block cipher based on the ARX structure, in which the encryption and decryption processes use the same key. It encrypts the plaintext of block length 64 bits with an initial key of length 96 bits. In order to keep the encryption process strong enough, the average design of the number of encryption rounds is 10 to 20 rounds. In this paper, the number of encryption rounds is set to 16 rounds. The encryption process is shown in Fig. 2. The 16 rounds of data processing of SPNRX can be expressed as ENC_{16}. ENC_{16} inputs plaintext data X and 128-bit master key K, and outputs ciphertext data C. Then the definition of ENC_{16} is shown in the following Formula (1).

$$ENC_{16}:\begin{Bmatrix}\{0,1\}^{64}\times\{\{0,1\}^{16}\}^{8}\to\{0,1\}^{64}\\(X,K_i,CON)\to C\end{Bmatrix} \quad (1)$$

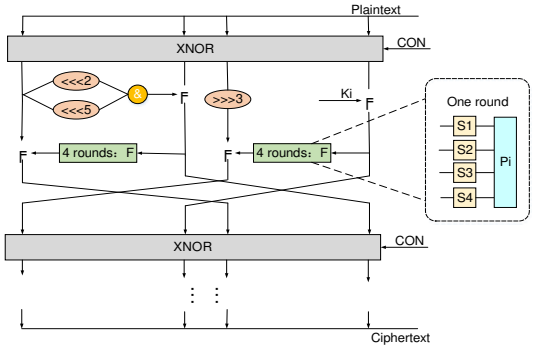


Fig. 2 Encryption structure of SPNRX

The one round of encryption process of SPNRX:

(1) Input 64-bit plaintext data X and round constant CON, and $X, CON \in \{0,1\}^{64}$, then divide the 64-bit plaintext into 4 16-bit parts: $X = X_0 \| X_1 \| X_2 \| X_3$.

(2) Enter a 16-bit sub-key $K_i (1 \le i \le 16)$ generated by the key schedule.

(3) Output 64-bit ciphertext data $C \in \{0,1\}^{64}$. (see Algorithm 1 for details)

Algorithm1 SPNRX encryption algorithm

Input: *plaintext X, 64bits ; key K, 96bits ; round constant CON, 64bits*

Output: *ciphertext C, 64bits*

1. $X \leftarrow X \odot CON$
2. $X \to X_0||X_1||X_2||X_3$
3. **for** *i=1* **to** *16* **do** *the following*
4. $X_1 \leftarrow ((X_0 <<< 2)\&(X_0 <<< 5)) \oplus X_1$
5. $X_0 \leftarrow X_0 \oplus F(X_1)$
6. $X_3 \leftarrow X_3 \oplus K_i$
7. $X_2 \leftarrow (X_2 >>> 3) \oplus F(X_3)$
8. $X_2||X_3||X_0||X_1 \leftarrow X_0||X_1||X_2||X_3$
9. **end for**
10. $C_0||C_1||C_2||C_3 \leftarrow X_0||X_1||X_2||X_3$
11. $C \leftarrow C_0||C_1||C_2||C_3$
12. **return** *C*

3.2.1 round function

Based on the consideration of the number of iterative rounds of the algorithm and the cost of hardware implementation, the algorithm uses two identical round functions in each round of the iterative structure. In addition, considering the security of the algorithm, both round functions are based on the SPN structure, using the S-box transformation and P-box permutation of the four rounds to perform nonlinear and linear transformations, thereby realizing the design principles of confusion and diffusion. The round function realization of the proposed algorithm SPNRX in each round is only composed of S-box transformation and P permutation layer, as shown in Fig. 3. The process of the F function is as follows. However, in each round function execution process, the S-box layer and the P permutation layer are repeated 4 times, and the S-box layer and the P permutation layer that iterates 4 rounds are called SP4 structure. The execution process of the round function is shown in Algorithm 2.

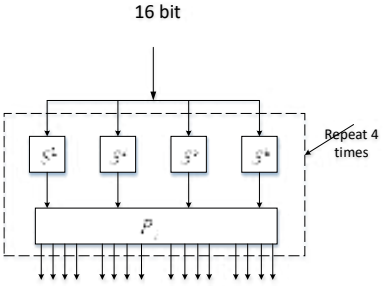


Fig. 3 Round function structure

Algorithm2 round function execution algorithm

Input: *plaintext, 16bits*

Output: *middle state, STATE*

1. $state \leftarrow X_i$
2. **for** *i=1* **to** *4* **do** *the following*
3. *Sbox(state)*
4. *P(state)*
5. **end for**
6. $STATE \leftarrow state$
7. **Return** *STATE*

3.2.2 XNORconstant

XNORconstant is to update a 64-bit plaintext data input.

The round constant is represented by a hexadecimal number, and the integer constant of the i-th round is defined as Con_i, the combination rule is as Formula (2). According to the combination rule, 16 combinations can be obtained, corresponding to 16 rounds of encryption process. Table 3 shows the round constant CON of each round of SPNRX.

$$Con_i = abcd \rightarrow \begin{cases} a = 2n-1 \\ b = 7n \\ c = 5n+1 \\ d = 6n+3 \end{cases} \quad n = 1,2 \tag{2}$$

Table 3 Round constant for SPNRX

Round i	1	2	3	4
Con_i	0x1769	0x37b9	0x176f	0x37bf
Round i	5	6	7	8
Con_i	0x17b9	0x3e69	0x17bf	0x3e6f
Round i	9	10	11	12
Con_i	0x1e69	0x3eb9	0x1e6f	0x3ebf
Round i	13	14	15	16
Con_i	0x1e69	0x3769	0x1ebf	0x376f

After determining the round function, the round constant of each round is used to perform the XNOR operation with the high 8 bits and low 8 bits of the plaintext data. The XNOR operation of the round constant is simple, the calculation is small, and the calculation takes up a small amount of resources, and it can update the input plaintext data in each round.

3.2.3 Matrix transform sub-key

The algorithm SPARK uses a key length of 96 bits. First introduce the 96-bit seed key K, which is recorded as $K_{96} = k_{95} \| k_{94} \| \Lambda \| k_2 \| k_1 \| k_0$ .Then the key value obtained by the key schedule is expressed as a nibble array as in Formula (3).

$$\begin{bmatrix} K_0 & K_1 & K_2 & K_3 \\ K_4 & K_5 & K_6 & K_7 \\ K_8 & K_9 & K_{10} & K_{11} \\ K_{12} & K_{13} & K_{14} & K_{15} \end{bmatrix} \tag{3}$$

Among them, the value of the main diagonal of the matrix is taken as the sub-key of the odd-numbered rounds of encryption, and the value of the matrix of the sub-diagonals is taken as the sub-key of the even-numbered round of encryption, as shown in Formula (4).

$$subkey = \begin{cases} K_0 \ \mathrm{P} \ K_5 \ \mathrm{P} \ K_{10} \ \mathrm{P} \ K_{15} & i = 2n-1 \\ K_3 \ \mathrm{P} \ K_6 \ \mathrm{P} \ K_9 \ \mathrm{P} \ K_{12} & i = 2n \end{cases} \tag{4}$$

The round key addition is the XOR of the 16-bit plaintext data input in the round function and the 16-bit round key. The operation formula is shown in Formula (5):

$$X_i \leftarrow X_i \oplus subkey \tag{5}$$

3.2.4 S-box transformation

S-boxes can realize nonlinear transformation of data and are nonlinear components in encryption algorithms. Their selection plays an important role in the security of SPNRX. For the S-box layer, in addition to its security, it is also necessary to consider the efficiency of implementation, that is, there are certain requirements for its cost and area. Since the proposed algorithm needs to be suitable for resource-constrained NB-IOT terminals and needs to have efficient hardware performance, 4×4 S-boxes are chosen in SPNRX. Because the input and output bits of the 3×3 S-box are too small, the security of the algorithm cannot be well guaranteed; while the 8×8 S-box can achieve better security, it will increase the storage and computation cost of the algorithm due to its higher number of input and output bits, which may have a negative impact in hardware implementation. Currently, the 4×4 S-box not only provides appropriate security, but also implements low complexity on the hardware. In this paper, the S-box of lightweight block cipher algorithm ANU is selected. The design of S-box fully considers the characteristics of bijection, linearity and difference, which can better balance the implementation efficiency of software and hardware and the security of the algorithm. In the nonlinear transformation layer, four identical S-boxes are used in parallel, where the S-box transformation is shown in Table 4 below, and the input and output of the S-box are represented by four-bit binary hexadecimal numbers.

Table 4 S-box transformation of SPNRX

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

3.2.5 P-box transformation

P-box transformation is essentially a permutation, which satisfies the principles of packet diffusion and chaos. P-box transformation is the linear layer structure of the proposed algorithm, which can enhance the diffusion characteristics of the algorithm and reduce the hardware implementation cost of the linear layer to a certain extent. In general, P permutation is actually a process of rearrangement. Input 16-bit plaintext data in P-box, and then according to a certain rule of rearrangement, finally produce 16-bit output data. In the SPNRX algorithm, the permutation layer chooses a simple bitwise rearrangement due to its current simplicity and efficiency, and the rearrangement rules are shown in Table 5 and Table 6 below. In other words, each time the round function is executed, the 16-bit input of each round function goes through the data transformation process as shown in Fig. 4 and Fig. 5.

Table 5 P_0-box of SPNRX

x	0	1	2	3	4	5	6	7
P_0	11	4	2	13	7	0	14	9
x	8	9	10	11	12	13	14	15
P_0	3	12	10	5	15	8	6	1

Table 6 P_1-box of SPNRX

x	0	1	2	3	4	5	6	7
P_1	8	6	1	15	4	2	13	11
x	8	9	10	11	12	13	14	15
P_1	0	14	9	7	12	10	5	3

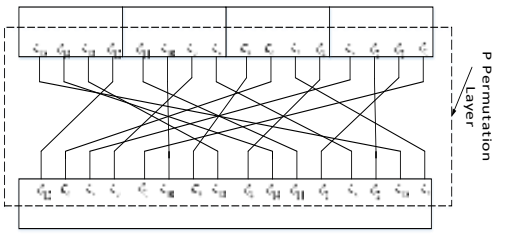


Fig. 4 P_0-box of SPNRX

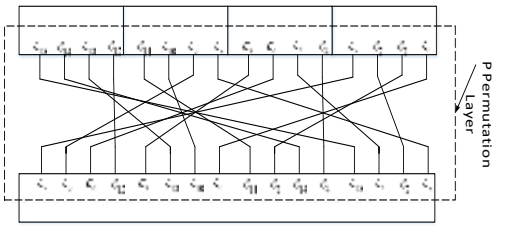


Fig. 5 P_1-box of SPNRX

In the process of P-box transformation, we can find that each time the round function is executed, the output of one S-box enters the 4 S-boxes of the next S-box transformation, that is, the input of one S-box comes from 4 different S-boxes.

3.3 Key Schedule of round permutation

The key is the basic component in the encryption and decryption process, and the security of the data is closely related to the security of the key. If an attacker knows the key used for encryption, the confidentiality of the data is at risk of being lost. Therefore, necessary measures need to be taken to make the key disclosure as difficult as possible. In order to prevent exhaustive key search, the key should be long enough. But the longer the key, the slower the encryption. Therefore, this paper uses a key with a length of 96 bits in the key schedule. In the SPNRX cipher, the key schedule generates a total of 16 sub-keys, each with a length of 16 bits. By using Addconstant to break the regularity between the keys, and then using matrix transformation and P-box replacement to schedule the keys, the complexity of key disclosure is increased to a certain extent.

The key schedule of SPNRX is to use the 96-bit master key $K_{96} = k_{95} \| k_{94} \| \Lambda \| k_2 \| k_1 \| k_0$ to generate 16 16-bit round keys. The process of generating the round keys is shown in Fig. 6.

The 96bit master key K is first divided into keys K^1 and K^2 with lengths of 64bit and 32bit, and then K^1 and K^2 are XNORed to obtain a new key value K', as shown in Formula (6):

$$\begin{aligned} K^1 &= k_{95}\ \mathrm{P}\ k_{94}\ \mathrm{P}\ \mathrm{L}\ \ \mathrm{P}\ k_{64} \\ K^2 &= k_{63}\ \mathrm{P}\ k_{62}\ \mathrm{P}\ \mathrm{L}\ \ \mathrm{P}\ k_0 \\ K' &= K^1 \bullet K^2 \end{aligned} \tag{6}$$

After that, when each round of encryption process is executed, the key schedule also executes one round accordingly. The specific execution steps are as follows:

(1) Perform the Addconstant operation between the key value K' obtained after the XOR of the Formula (6) and the round constant in Table 3. As shown in Formula (7).

$$K_{(1)} = K' \oplus Con_{16-i} \tag{7}$$

where Con_{16-i} represents the round constant of the 16th-i round

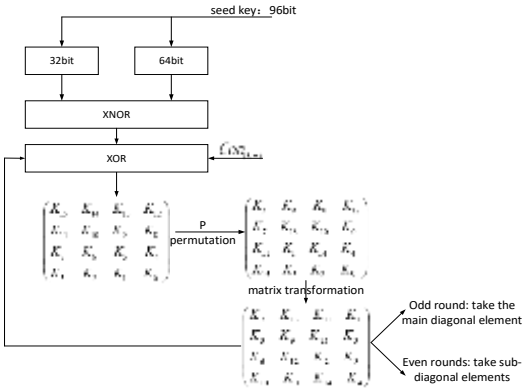


Fig. 6 Generation of sub-keys

(2) The obtained new key value K′ is represented by the nibble matrix of Formula (8) as,

$$K_{(2)} = \begin{bmatrix} K_{15} & K_{14} & K_{13} & K_{12} \\ K_{11} & K_{10} & K_9 & K_8 \\ K_7 & K_6 & K_5 & K_4 \\ K_3 & K_2 & K_1 & K_0 \end{bmatrix} \tag{8}$$

where $K_i = k_{4i+3} \| k_{4i+2} \| k_{4i+1} \| k_{4i}\ (0 \le i \le 15)$.

(3) Then the P_1 permutation box is applied to the key matrix to reorder it, and the result of the reordered key matrix is obtained as Formula (9):

$$K_{(3)} = \begin{bmatrix} K_3 & K_9 & K_6 & K_{12} \\ K_7 & K_{13} & K_{10} & K_0 \\ K_{11} & K_1 & K_{14} & K_4 \\ K_{15} & K_5 & K_2 & K_8 \end{bmatrix} \tag{9}$$

(4) The above matrix is decomposed into four 2*2 sub-matrices as shown in Formula (10).

$$\begin{aligned} K_0' &= \begin{bmatrix} K_3 & K_9 \\ K_7 & K_{13} \end{bmatrix} \quad K_1' = \begin{bmatrix} K_6 & K_{12} \\ K_{10} & K_0 \end{bmatrix} \\ K_2' &= \begin{bmatrix} K_{11} & K_1 \\ K_{15} & K_5 \end{bmatrix} \quad K_3' = \begin{bmatrix} K_{14} & K_4 \\ K_2 & K_8 \end{bmatrix} \end{aligned} \tag{10}$$

Then swap the positions of K_1' and K_2', while K_3' and K_4' swap their own upstream and downstream elements, and the result is as shown in Formula (11).

$$K_{(4)} = \begin{bmatrix} K_7 & K_{13} & K_{11} & K_1 \\ K_3 & K_9 & K_{15} & K_5 \\ K_6 & K_{12} & K_2 & K_8 \\ K_{10} & K_0 & K_{14} & K_4 \end{bmatrix} \tag{11}$$

(5) In the odd rounds of the encryption algorithm, the main diagonal elements in the step (3) matrix are selected to be concatenated as wheel keys, while in the even rounds of the encryption algorithm, the sub-diagonal elements in the step (3) matrix are selected. The specific expression is shown in Formula (12).

$$subkey = \begin{cases} K_7 \ \mathrm{P}\ K_9 \ \mathrm{P}\ K_2 \ \mathrm{P}\ K_4 & i = 2n-1 \\ K_1 \ \mathrm{P}\ K_{15} \ \mathrm{P}\ K_{12} \ \mathrm{P}\ K_{10} & i = 2n \end{cases} \quad (12)$$

3.4 decryption process

While designing a strong encryption process, a successful decryption process is undoubtedly very important. Since the cryptographic algorithm proposed in this paper uses an ARX-type network structure, the decryption algorithm and encryption algorithm of SPNRX are structurally consistent. Therefore, when the decryption algorithm is executed, the steps in the execution process of the encryption algorithm only need to be reversed, which also causes the round sub-keys to be used in the reverse order. In addition, in the decryption process, the S-box and P permutations need to be reversed before use.

4. Security Analysis

The attempt to analyze the cipher is an attack. Successful cryptanalysis can resolve the plaintext or key corresponding to the legitimate ciphertext. A good cryptographic algorithm can resist cryptanalysis to a certain extent. In the following, the security analysis of the proposed algorithm will be carried out to verify whether it has the ability to resist linear analysis, differential analysis, algebraic analysis and other cryptanalysis.

4.1 Linear cryptanalysis

Linear cryptanalysis is a new attack method against DES proposed by Matsui [27] in the 1993 Eurocrypt. Its basic idea is to find the linear relationship between the plaintext, ciphertext and key, and give the linear expression of the cryptographic algorithm, so as to reveal the information about the key to achieve the effect of decryption. Linear cryptanalysis is one of the most basic attack methods, which is essentially a known plaintext attack method, that is, the attacker actually has a set of plaintext and corresponding ciphertext, but the attacker cannot distinguish which of these plain and ciphertext information is available, so it needs to use a linear equation to approximate the password. This linear expression is shown in Formula (13) [13].

$$X_{i_1} \oplus X_{i_2} \oplus \mathrm{L} \oplus X_{i_m} \oplus Y_{j_1} \oplus Y_{j_2} \oplus \mathrm{L} \oplus Y_{j_n} = 0 \quad (13)$$

where X_i represents the ith bit of the input $X=[X_1, X_2, \Lambda, X_m]$, and Y_j represents the jth bit of the input $Y=[Y_1, Y_2, \Lambda, Y_n]$. The attacker can use equation (13) to calculate the probability when the equation holds. In the above equation, the value of m+n bits is randomly selected and placed, then the probability of calculating this equation is exactly 1/2. Linear cryptanalysis is to determine the resistance to cryptanalysis by using the deviation of the probability of 1/2 when the equality is true. According to the Piling-Up Lemma stacking lemma, the deviation of X_i is defined as Formula (14):

$$\varepsilon_i = \Pr[X_i = 0] - \frac{1}{2} \quad (14)$$

Then $\varepsilon_{i_1, i_2, \Lambda, i_k}(i_1 < i_2 < \Lambda < i_k)$ represents the deviation of the random variable $X_{i_1} \oplus X_{i_2} \oplus \Lambda \oplus X_{i_k}$, as shown in Formula (15).

$$\varepsilon_{i_1, i_2, \Lambda\, i_k} = 2^{k-1} \prod_{j=1}^{k} \varepsilon_j \quad (15)$$

The smaller the probability deviation (|Pr- 1/2|), the better the algorithm is resistant to linear cryptanalysis. That is to say, if the probability that the equation is established is always 1/2, it can be considered that the proposed algorithm has good randomness and the ability to resist linear cryptanalysis.

In this paper, the S-box of 4*4 satisfies the rule $\{0,1\}^4 \to \{0,1\}^4$. If the input $X=[X_1, X_2, \Lambda, X_m]$ is selected uniformly and randomly from the set $\{0,1\}^4$, and similarly the output random variable Y_1, Y_2, Λ, Y_n is valued at $\{0,1\}$, then the linear linear method can be developed between the set of input bits and output bits of the S-box. Table 7 gives examples of the linear analysis probability part of an S-box considering three sample linear expressions. From these examples, it can be observed that most linear expressions have a probability of 1/2 and a probability deviation of 0 under the condition that formula (13) is satisfied. Therefore, it can be shown that SPNRX algorithm can resist linear analysis.

4.2 Differential cryptanalysis

Eli Biham and Adi Shamir [28] first published differential cryptanalysis research in 1990, which is a selective plaintext attack. Its basic idea is to obtain the most likely key by analyzing the influence of specific plaintext difference on the resulting ciphertext difference. Differential cryptanalysis is one of the most effective security analysis methods for iterative block ciphers.

To understand how the proposed algorithm resists differential cryptanalysis, it is necessary to calculate the differential probability of S-boxes and the number of active S-boxes. S-box is the only nonlinear component in block cipher algorithm, which plays a key role in security analysis. The minimum number of active S-boxes determines whether the cipher can resist differential cryptanalysis. When calculating the difference probability of S-box, it is necessary to calculate the difference distribution table of S-box first, as shown in Table 8.

Definition 1: For an S-box, assuming that the value after the XOR of an input message pair X_1 and X_2 is non-zero, that is, the input difference is non-zero, then this S-box is called an active S-box.

Definition 2: (Difference probability) For S-box $S_i : GF(2)^m \to GF(2)^m$, any given difference Δx and Δy, the probability of each difference S-box is shown in Formula (16).

$$DP^{S_i}(\Delta x \to \Delta y) = \frac{\#\{x \in GF(2)^m \mid S_i(x) \oplus S_i(x \oplus \Delta x) = \Delta y\}}{2^m} \quad (16)$$

Then, the maximum difference probability is shown in Formula (17).

$$MaxDP^{S_i} = \max_i \max_{\Delta x \neq 0, \Delta y} DP^{S_i}(\Delta x \to \Delta y) \qquad (17)$$

Table 7 Examples of linear analysis probability

X				Y				$x_2 \oplus x_4$	$x_3 \oplus x_4$	$x_1 \oplus x_2$	$x_2 \oplus y_3$	$x_1 \oplus x_4$	$y_3 \oplus y_4$
x_1	x_2	x_3	x_4	y_1	y_2	y_3	y_4	$\oplus y_2$	$\oplus y_2$	$\oplus y_1$	$\oplus y_4$		
0	0	0	0	0	0	1	0	0	0	0	1	1	0
0	0	0	1	1	0	0	1	1	1	1	1	1	1
0	0	1	0	0	1	1	1	1	0	0	0	0	0
0	0	1	1	1	1	1	0	0	1	1	1	1	1
0	1	0	0	0	0	0	1	1	0	1	0	1	0
0	1	0	1	1	1	0	0	1	0	0	1	0	1
0	1	1	0	1	0	1	0	1	1	0	0	1	0
0	1	1	1	0	0	0	0	0	0	1	1	0	1
1	0	0	0	0	1	0	0	1	1	1	0	0	1
1	0	0	1	0	0	1	1	1	1	1	0	0	0
1	0	1	0	1	0	1	1	0	1	0	0	0	1
1	0	1	1	1	1	0	1	0	1	0	1	1	0
1	1	0	0	1	1	1	1	0	1	1	1	0	1
1	1	0	1	0	1	1	0	1	0	0	0	1	0
1	1	1	0	0	1	0	1	0	0	0	0	1	1
1	1	1	1	1	0	1	1	0	0	1	1	0	0

Table 8 Differential distribution table of S-box of SPNRX

Input differential	output differential															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	2	0	2	0	4	2	2	0	2	2	0
2	0	0	0	0	0	2	0	2	0	0	2	2	4	2	2	0
3	0	2	0	2	2	0	2	0	0	2	0	2	2	0	2	0
4	0	0	0	2	0	4	2	0	0	0	0	2	0	4	2	0
5	0	0	2	2	2	0	0	2	4	0	0	0	2	0	2	0
6	0	2	2	0	0	0	2	2	4	2	0	2	0	0	0	0
7	0	0	4	2	0	0	2	0	0	0	0	2	0	0	2	4
8	0	0	0	2	0	0	2	0	0	0	4	2	0	0	2	4
9	0	4	0	2	0	2	2	2	0	0	2	0	0	2	0	0
A	0	0	0	2	4	2	2	2	0	0	2	0	0	2	0	0
B	0	2	0	0	2	0	0	0	0	2	4	0	2	0	0	4
C	0	0	4	0	0	4	0	0	0	0	0	0	0	4	0	4
D	0	0	2	0	2	0	2	2	4	0	0	2	2	0	0	0
E	0	2	2	2	0	0	0	2	4	2	0	0	0	0	2	0
F	0	4	0	0	4	0	0	0	0	4	0	0	4	0	0	0

According to the above difference probability formula and the difference distribution table of S-box, the maximum difference probability of S-box is calculated as 2^{-2}. In addition, it can be proved that in the process of a round of encryption, the minimum number of active S-boxes of SPNRX algorithm is 6. The proof process is as follows:

Suppose that in the round of encryption process shown in Figure 3, there is an active S-box (S^1) in the first round of S-box transformation in SP4 structure, that is, the input difference is nonzero. In order to make the minimum number of active s-boxes exist in this round of encryption process, the input difference in this active s-box has the least number of non-zero bits, that is, there are only four possible input differences in the first active s-box: 0001,0010,0100,1000. According to the differential distribution table of S-box, the possible input difference and possible output difference of the first S-box can be obtained as shown in Table 9. According to the rule of displacement layer by position, the possible input difference of P-box transformation and its corresponding output difference can be obtained as shown in Table 10.

Table 9 Active S-box differential distributions of the first round S-box transformation of the round function

S-box possible input differential	S-box possible output differential
0001	0101,0111,1001,1010,1011,1101,1110
0010	0101,0111,1010,1011,1100,1101,1110
0100	0011,0101,0110,1011,1101,1110
1000	0011,0110,1010,1011,1110,1111

Table 10 Active S-box differential distribution of P-box transformation in the first round of the round function

P-box possible	**Corresponding P-box output**	**Second S-box possible non-zero**

input		input
0011	1000,0001,0000,0000	1000,0001
0101	1000,0000,0100,0000	1000,0100
0110	0000,0001,0100,0000	0001,0100
0111	1000,0001,0100,0000	1000,0001,0100
1001	1000,0000,0000,0010	1000,0100
1010	0000,0001,0000,0010	0001,0010
1011	1000,0001,0000,0010	1000,0001,0010
1100	0000,0000,0100,0010	0100,0010
1101	1000,0000,0100,0010	1000,0100,0010
1110	0000,0001,0100,0010	0001,0100,0010
1111	1000,0001,0100,0010	1000,0001,0100,0010

Assuming that during one round of encryption, only the left round function has non-zero differential bit input, and in the round function, the S-box transformation and the P permutation layer are repeated 4 times. In the first round of S-box transformation of the round function, it is assumed that there is an active S-box. According to the possible input difference of the second S-box in Table 8, it can be seen that in the second round of transformation, there are at least two active S-boxes. And the non-zero bits of the input difference of the two active S-boxes are only 1 bit, so it can be seen from Table 10 that in the third structural transformation, there are at least two active S-boxes as well. In the fourth round of transformation, it can be determined that there must be an S-box whose input differential is non-zero. It can be concluded that in one round of encryption, there are at least 1+2+2+1=6 active S-boxes. Fig. 7 shows the input and output differential paths that may be obtained when searching for the minimum number of active S-boxes during the execution of a round function.

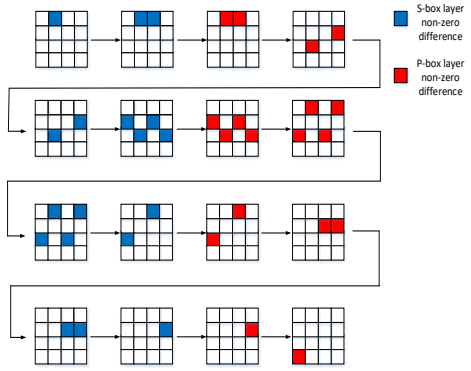


Fig. 7 Possible differential paths of least active S-boxes in one-round function

Therefore, there are at least 6 differentially active S-boxes in any round of encryption algorithm, so there will be at least $6\times10=60$ active S-boxes in the entire encryption process of the first 10 rounds of the SPNRX algorithm. For this algorithm, the maximum differential probability of its S box is 2^{-2}, then the differential probability of the entire encryption algorithm is 2^{120}. Therefore, if a differential attack is performed on the proposed algorithm, a successful attack requires at least a data complexity of 2^{120}, which exceeds the security threshold of 2^{64} [29]. Therefore, SPNRX has the ability to resist differential analysis.

4.3 Algebraic attack

Algebraic attack was first proposed by Courtois and Pieprzyk [30-31]in 2002, and has now developed into one of the important methods of cryptanalysis. The algebraic attack of cryptography is essentially the reverse derivation of the key by solving the nonlinear equation system including plaintext, ciphertext and key. The attacker achieves the purpose of deciphering by solving the corresponding multivariate algebraic equation problem on the finite field. It can be seen from the literature [30] that the S-box can be represented by a multivariate algebraic equation composed of many variables. When performing algebraic analysis, the main purpose is to analyze the nonlinear component S-box layer. And any 4*4 S-box can be represented by at least 21 quadratic equations, and these equations are composed of eight input/output variables on $GF(2)$. In the SPNRX algorithm, there are $4\times4\times2=32$ S-boxes in one round of encryption, and there are a total of $32\times16=512$ S-boxes in the entire encryption algorithm. That is to say, the entire SPNRX algorithm has $512\times21=10752$ multivariate algebraic equations and $512\times8=4096$ input/output variables, and the number of equations is greater than the number of variables. It can be seen that this is an over-determined multivariate equation system, and solving such a system of equations is an NP-hard problem. Then we compare the number of algebraic equations and variables of SPNRX, PRESENT-80, MIBS-80, KLEIN-64 and QTL-64, and the comparison results are shown in Fig. 8. It is proved that algebraic attacks cannot threaten SPNRX under a certain space-time complexity.

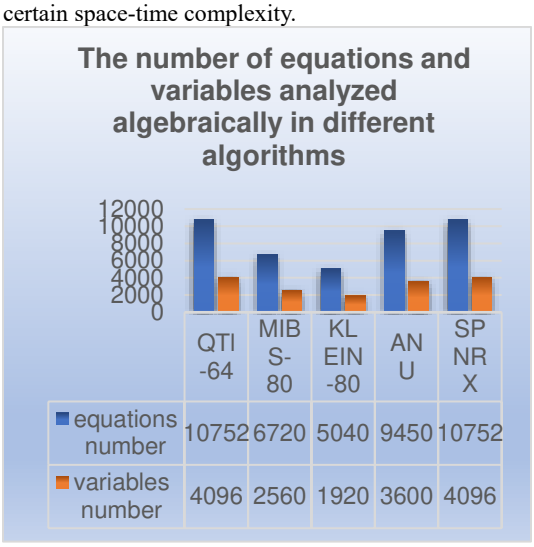


Fig. 8 Number of equations and variables for algebraic analysis in different algorithms

4.4 XSL attack

Extended Sparse Linearization (eXtended Sparse Linearization, XSL) attack is an analysis method for block cipher algebraic structures proposed by N. T. Courtois and J. Pieprzyk et al. in 2002. It is known from [30] that the complexity of XSL attacks is calculated by exploiting the work factor. Defining the complexity of the attack as WF, for the algorithm proposed in this paper, the complexity is calculated as Formula (18-19):

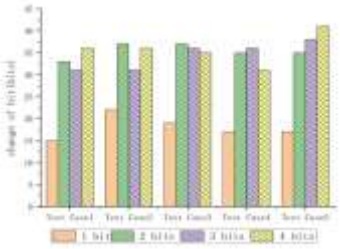
$$\Gamma = \left(\frac{t-r}{s}\right)^{\frac{t-r}{s}} = 2^8 \tag{18}$$

$$\begin{aligned} WF &= \Gamma^{\omega} \cdot (Block Size)^{\omega t - r/s} \cdot (Number\ of\ rounds)^{2\omega t - r/s} \\ &= \left(2^8\right)^{2.376} \times 64^{2.376\times 4} \times 16^{2\times 2.376\times 4} = 2^{171.072} \end{aligned} \tag{19}$$

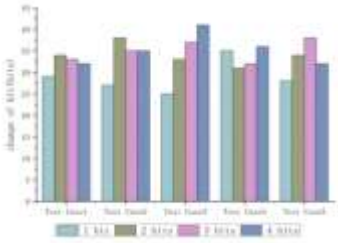
Among them, t represents the number of monomials in the linear equation, r represents the number of S-box linear equations, s represents the number of input bits of the S-box, and $\omega = 2.376$. According to the calculation result, the complexity of XSL attack is $2^{171.072}$, and the range of exhaustive search is 2^{64}. SPNRX's WF is large enough to resist XSL attacks.

4.5 Avalanche effect

Avalanche effect is an important security analysis that measures the diffusivity of block ciphers. It means that when the input changes by one bit, it will cause a significant change in the output. According to the strict avalanche criterion SAC [32], the test is considered perfect if a change of one bit results in a change of 50% of the bits. The greater the avalanche effect, the greater the security factor of the encryption algorithm. Therefore, we keep the key unchanged, select a specific plaintext, and then randomly change one, two, three, and four of them to obtain some test cases. Similarly, when a specific key is selected, the plaintext is kept unchanged. The obtained results are shown in Fig. 9, and Fig. 9(a) and (b) show the avalanche effect of plaintext and key under different test cases, respectively.



(a) Avalanche effect of plaintext test cases



(b) Avalanche effect of key test cases
Fig. 9 Avalanche effect test case results

By observing the test case results, it can be seen that the change of one bit of plaintext can only change the 20 bits of ciphertext, and the avalanche effect of plaintext is not good. However, changing one key can change half of the ciphertext, and the avalanche effect of the key is good. In conclusion, the security of the algorithm can be guaranteed.

5. performance analysis

This part is divided into two sections. The first section calculates the number of GE consumed by the hardware implementation of the cryptographic algorithm, and the second section uses different evaluation parameters to test the encryption effect and security strength of the proposed cryptographic algorithm.

5.1 hardware implementation

Lightweight block ciphers are mostly used in resource-constrained devices, which often have strict requirements on cost and storage space. Therefore, when evaluating the hardware implementation performance of the algorithm, it is necessary to measure the circuit area of the algorithm. As we all know, the area of hardware implementation is proportional to its required cost. Therefore, when the designed cryptographic algorithm is implemented in hardware, the area occupied by it should be reduced as much as possible. The number of Gate Equivalences(GEs) is commonly used to evaluate the performance of the algorithm in hardware platform implementation, This paper uses the ARM standard cell library using 0.13 micron ASIC process [33]. And the GE number required for some basic Gate implementation of the library is shown in Table 11.

Table 11 Number of GEs required for basic gate implementation

basic unit	AND	XOR	2-1 MUX	D flip-flop	XNOR
GEs	1.25	2.00	2.25	4.25	2.00

The data path of SPNRX is shown in Fig. 10. In one round of encryption, operations such as Addroundkey, XNORconstant, logical AND, S-box transformation, P-box permutation, and simple rotation are included.

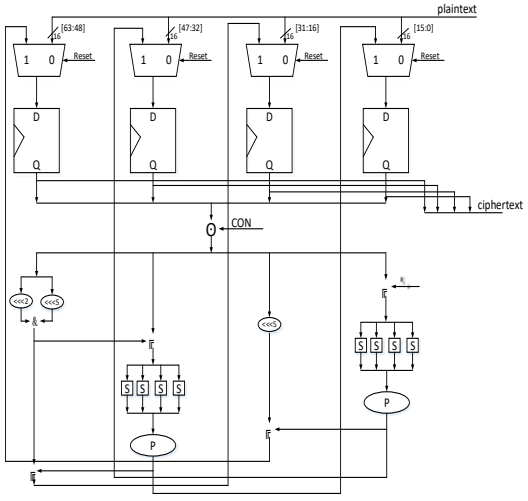


Fig. 10 Data path of SPNRX

It is known from literature [34] that it takes 4.25GEs to store 1-bit data, 408GEs to store 64-bit plaintext data in SPNRX algorithm and 272GEs to store 96-bit key, respectively. While P-box replacement and shift operations of data state can be obtained through wiring without consuming hardware resources. An s-box

hardware implementation of the nonlinear layer of the SPNRX algorithm requires a gate number of 28GEs. Table 12 summarizes the number of GEs required for the hardware implementation of each component module of the SPNRX block cipher.

Table 12 Number of GEs required by SPNRX hardware to realize each module

component modules	GEs
data storage	64×4.25=272
key storage	96×4.25=408
XNORconstant	32×2=64
AND	16×1.25=20
2-1 MUX	4×2.25=9
nonlinear layer	2×4×24=192
AddRoundKey(XOR)	16×2=32
Other data XOR	3×16×2=96
Total	1093

According to the results in Table 12, it can be seen that the hardware implementation of SPNRX requires about 1093GE. By comparing it with other lightweight block ciphers, the results are shown in Table 13 and Fig. 11.

Table 13 Comparison between SPNRX and other lightweight block ciphers

algorithm name	Block size(bits）	key length （bits）	Logic process （μm）	Algorithm structure	GEs consumed	Ref.
KLEIN	64	64	0.13	SPN	1432	[33]
QTL	64	64	0.18	GFN	1026	[35]
Piccolo	64	80	0.13	GFN	1496	[36]
PRESENT	64	80	0.18	SPN	1570	[36]
RECTANGLE	64	80	0.13	SPN	1599.5	[36]
TWINE	64	80	0.09	GFN	1799	[35]
SFN	64	96	0.18	GFN	1877	[35]
PRINCE	64	128	0.13	SPN	2963	[33]
CLEFIA	128	128	0.13	GFN	2488	[33]
SPNRX	64	96	0.13	ARX+SPN	1093	This paper

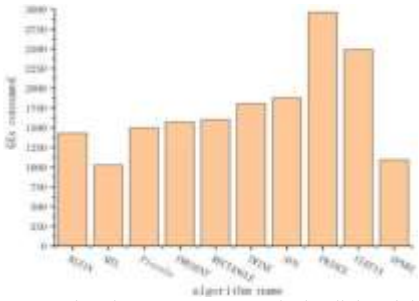


Fig. 11 Comparison between SPNRX and other lightweight block ciphers

The proposed algorithm combines ARX structure, which requires low hardware resources for simple logic AND and XOR operation, and the Rotation does not consume hardware resources. In addition, the matrix operation and P-layer diffusion in the key schedule of the proposed algorithm can be completed by wiring, which also reduces the cost of hardware implementation. Therefore, according to the comparison results, it can be seen that the hardware resource consumption of the proposed algorithm is second only to QTL, only 1093GE. Since it is stipulated that the number of GEs required for hardware implementation is 2000-3000, the algorithm is called a lightweight implementation, and the number of GEs required for hardware implementation of the proposed algorithm is much less than 2000. It can be seen that the proposed algorithm has reached the lightweight standard.

5.2 Software Implementation

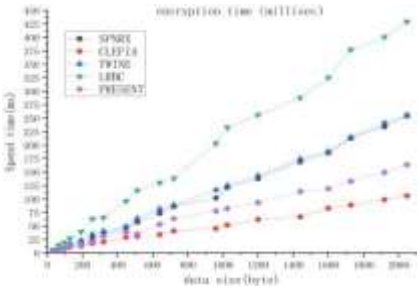
In order to evaluate the software performance and security strength of the proposed algorithm, this section uses Python to test the encryption and decryption time, algorithm correctness, key sensitivity, histogram, correlation coefficient, image entropy and etc. on Intel(R) Core(TM) I5-7200U CPU @ 2.50GHz 2.71 GHz processor.

5.2.1 The execution time

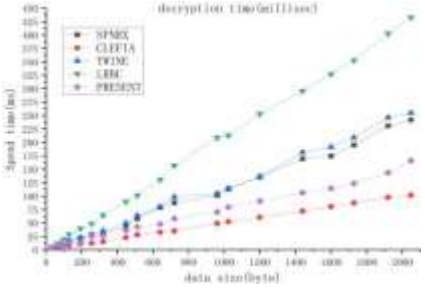
The basic parameter for evaluating the performance of an algorithm is the time it takes to perform encryption and decryption of data of a certain size. In addition to considering its security and low consumption, the algorithm proposed for resource-constrained devices should also have a certain efficiency. The efficiency of a cryptographic algorithm refers to the speed at which the algorithm encrypts and decrypts data. This paper uses the encryption and decryption execution time to measure it. The less time consumed, the faster the encryption and decryption speed.

In this experiment, the proposed algorithm is implemented, and the encryption and decryption times are 0.763 milliseconds and 0.772 milliseconds, respectively. Then the lightweight block cipher algorithms CLEFIA, TWINE, LRBC and PRESENT are selected to compare with the proposed algorithm in the case of different data sizes, the encryption (decryption) time consumption. The PRESENT algorithm is SPN structure, the block size and key size are 64bit and 80bit respectively, and the number of encryption rounds is 31. Both CLEFIA and TWINE algorithms have GFN structure. When CLEFIA block and key size are both 128bit, the number of encryption rounds is 18 rounds, while TWINE encryption rounds are 36 rounds, and the block and key lengths are 64bit and 80bit respectively. The LRBC algorithm is a combination of Feistel and SPN structure, and the block size and key length are both 16bit, requiring 24 rounds of iteration. Fig. 12(a) and (b) show the execution times of data encryption and decryption by five encryption algorithms, respectively.

It can be seen from Fig. 12. that the encryption (decryption) time of the algorithm increases with the increase of the amount of data. When the data size is less than 500Byte, the encryption and decryption time of the other four algorithms except LRBC algorithm is very close. When the data size is higher than 500Byte, the encryption (decryption) execution time of the proposed algorithm is slightly higher than CLEFIA and PRESENT algorithm, lower than LRBC algorithm, and close to TWINE algorithm. Compared with PRESENT, the key length of the proposed algorithm is longer, so the encryption speed is slower. Although the length of the plaintext and the key of CLEFIA algorithm is larger, compared with SPNRX algorithm, CLEFIA algorithm needs less rounds and consumes less time when encrypting data of the same size. The LRBC algorithm is the opposite. Although the number of rounds of TWINE is more than that of the proposed algorithm, the key length of the proposed algorithm is larger, so the encryption and decryption rates of TWINE algorithm and the proposed algorithm are similar. In summary, the proposed algorithm performs well in the execution efficiency of encryption and decryption.



(a) Encryption time of the algorithm



(b) Decryption time of the algorithm

Fig. 12 Encryption and decryption time of the algorithm in different data sizes

5.2.2 Memory usage

One of the major problems faced by resource-constrained devices is memory limitation. Therefore, for lightweight encryption algorithms, in addition to the time consumed by the encryption and decryption of the password algorithm, memory resources occupied by the algorithm should also be considered. Through several encryption and decryption tests on the algorithm, the test results are shown in Table 14.

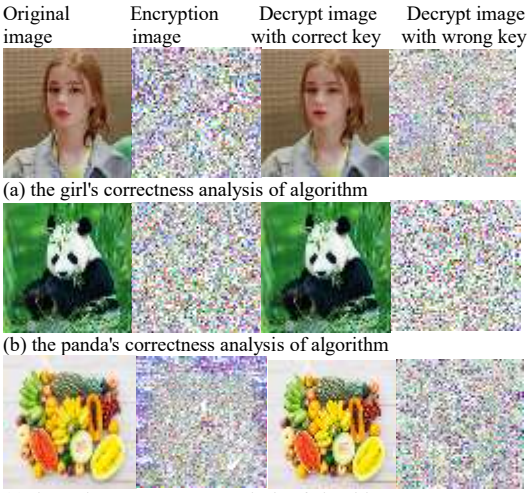
Table 14 Memory usage comparison

algorithm name	Block size	Key length	memory size	Ref
AES	64bit	128bit	48MB	[37]
SDN_IIOT_EN	64bit	128bit	52MB	[37]
SPNRX	64bit	96bit	41.06MB	This paper

According to the data in the table, it can be found that the encryption algorithm proposed in this paper reduces the memory resource occupation and can control it in a small range.

5.2.3 Correctness of algorithm

For the encryption algorithm, it is important to ensure the correctness of the algorithm. This experiment verifies the encryption and decryption of the three original images in Fig. 13. The results show that only the correct key can be used to accurately decrypt the image, while the original image cannot be obtained if the wrong key is used.



(a) the girl's correctness analysis of algorithm

(b) the panda's correctness analysis of algorithm

(c) the melon's correctness analysis of algorithm

Fig. 13 Algorithm correctness analysis

5.2.4 key sensitivity

A strong block cipher yields a very different encryption result when the key changes by one bit. According to Fig. 13. if the key is changed by one bit, the correct decryption result cannot be obtained, indicating that the algorithm has a certain sensitivity to the key. In addition, this experiment also used the number of pixels change rate (NPCR) and the unified average change intensity (UACI) to quantitatively assess the sensitivity. NPCR and UACI represent the number of pixels and the average intensity of changes between two encrypted images, and their ideal values are 99.6094% and 33.4635%, respectively. In the key sensitivity test, K^' is obtained by changing one bit of the key value K, and then the NPCR and UACI values of the encrypted images with these two key values are calculated to quantify the difference between the two images. The NPCR and UACI values calculated in Fig. 13. are shown in Table 15.

Table 15 the results of key sensitivity analysis

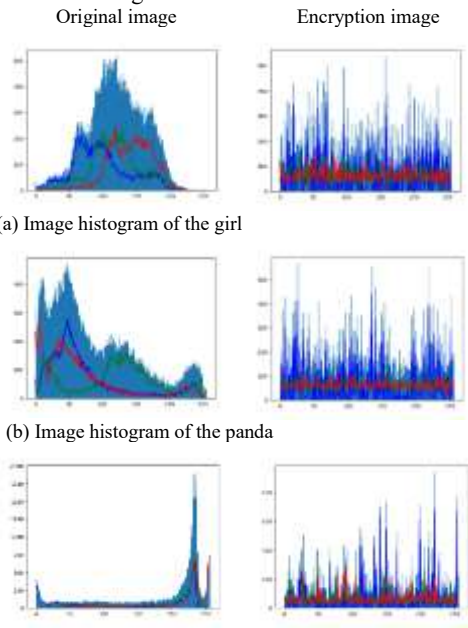
	NPCR	**UACI**
Fig. 13(a)	99.5687	34.0757

Fig. 13(b)	99.5748	34.4624
Fig. 13(c)	99.5544	32.6993

Since the key of each round is diffused and confused by P-box permutation and matrix transformation, and the round constant is used to eliminate its similarity, the one-bit change of the key will lead to the change of the key in each round, and thus the ciphertext will change. As shown in Table 15, NPCR and UACI values are close to the ideal values, indicating that the algorithm is sensitive to key and the encryption algorithm has certain security.

5.2.5 Histogram Analysis

The histogram analysis can obtain the statistical characteristics of the encrypted image and visualize the security strength of the algorithm. In this experiment, the histogram analysis of the original image and encrypted image in Fig. 13. is carried out, and the result is shown in Fig. 14.



(a) Image histogram of the girl

(b) Image histogram of the panda

(c) Image histogram of the melon

Fig. 14 Histogram analysis of the corresponding original and encrypted images

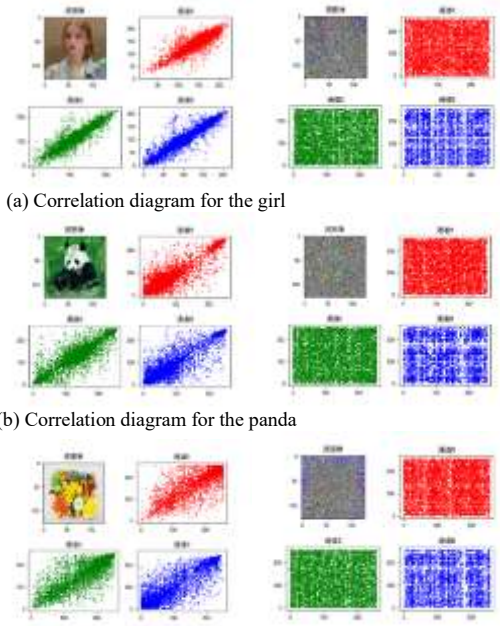
According to the experimental results, it can be seen that the histogram of the image before and after encryption changes greatly, indicating that the statistical characteristics of the two have almost no corresponding relationship, and the histogram before encryption fluctuates greatly, while the histogram of the encrypted image still fluctuates in a small part, the general distribution tends to be uniform. It can be seen that the P-box replacement layer in the algorithm plays a role in diffusion, the statistical properties of the plaintext are generally uniformly diffused into the ciphertext, which greatly reduces the correlation between the plaintext and the ciphertext, indicating that the algorithm is secure.

5.2.6 Correlation

Correlation is a method used to measure the strength of a cryptographic algorithm, and the correlation coefficient represents the dependence between two values. In this experiment, the original image and encrypted image in Fig. 13. are selected to analyze their correlation, and the obtained results are shown in Fig. 15. In addition, the correlation coefficient between adjacent pixel pairs of the selected image is calculated, and the calculation formula is Formula (20).

$$r_{xy} = \frac{\frac{1}{N}\sum_{i=1}^{N}(x_i - E(x))(y_i - E(y))}{\sqrt{D(x)}\sqrt{D(y)}} \tag{20}$$

Among them, x, y represent two adjacent image pixel values, E(x), E(y) and D(x), D(y) represent the expectation and variance of variables x, y, respectively. The obtained calculation results are shown in Table 16.



(a) Correlation diagram for the girl

(b) Correlation diagram for the panda

(c) Correlation diagram for the melon

Fig. 15 Correlogram analysis of different channels

Table 16 Correlation coefficients of different channels

	Original image		
The girl	R	G	B
	0.8932	0.9092	0.9345
The panda	R	G	B
	0.9042	0.9112	0.8878
The melon	R	G	B
	0.8621	0.8751	0.9437
	Encryption image		
The girl	R	G	B
	0.0383	0.0604	0.0122
The panda	R	G	B
	0.0053	0.0107	0.0156
The melon	R	G	B
	0.0244	0.0262	0.0981

For an ideal block cipher, the ciphertext obtained after encryption by the algorithm does not depend on the corresponding plaintext, that is, the information of the plaintext cannot be obtained from the ciphertext. The algorithm proposed in this paper uses a four-round SPN structure in the round function to achieve confusion and diffusion, and also transforms the key in the key schedule. Therefore, according to the results in Fig. 15. and Table 16, it can be seen that each channel

of the original image has a high correlation, while the correlation value of each channel of the encrypted image is very small. It shows that after the image is encrypted by the encryption algorithm, the ciphertext has no obvious dependence on the plaintext, which means that it is difficult for analysts to infer the relationship between the plaintext and the ciphertext according to the statistical law between the plaintext and the ciphertext. It can be seen that the proposed algorithm has higher strong nonlinearity to resist statistical analysis attacks.

5.2.7 image entropy

Entropy can measure the degree of chaos in the amount of information. For cryptographic algorithms, the greater the entropy of the encrypted image, the better the encryption effect. In this experiment, the three original pictures and encrypted pictures in Fig. 13. are analyzed, and the entropy values of the images before and after encryption are calculated as shown in Formula (21), and the obtained results are shown in Table 17.

$$Entropy = \sum p_i(i)\log_2\left(\frac{1}{p_i(i)}\right) \quad (21)$$

where P_i represents the probability of the i-th pixel value of the image.

Table 17 Image entropy of original and encrypted images

	Original image	Encryption image
The girl	6.7768	7.5786
The panda	6.6249	7.6358
The melon	6.7213	7.5062

According to the results in Table 17, the entropy value of the encrypted image is significantly increased compared with the image entropy value before encryption, which indicates that the encryption effect of the algorithm is good.

6 Conclusion

In this paper, a lightweight block cipher algorithm SPNRX based on ARX and SPN structure is proposed to protect the security of NB-IOT resource-constrained terminal data transmission. It can not only meet the special needs of NB-IOT terminals, but also meet the lightweight encryption algorithm for lightweight standards. It has a compact performance in hardware implementation, requiring only 1093GE, far less than 2500GE. Through the security analysis of the proposed algorithm, the results show that SPNRX has high resistance to different cryptanalysis, including linear, differential, and algebraic analysis. In addition, the cryptographic algorithm is tested in terms of key sensitivity, histogram, correlation coefficient and image entropy. The experimental results show that the encryption effect and security strength of the algorithm can be guaranteed. And use the encryption and decryption time to test the operation efficiency of the algorithm, and think that SPNRX has a higher operation efficiency. However, the avalanche effect of plaintext is slightly insufficient. In the future research, the encryption process can be studied and improved to enhance the diffusion effect of plaintext and further improve the operation efficiency of encryption and decryption.

Declarations

-Ethical Approval

I would like to declare on behalf of my co-authors that the work described was original research that has not been published previously, and not under consideration for publication elsewhere, in whole or in part.

-Human and Animal Ethics

Not applicable.

-Conflict of interests

No conflict of interest exits in the submission of this manuscript, and manuscript is approved by all authors for publication.

-Data Availability

The datasets generated during the current study are available from the corresponding author on reasonable request.

-Authors' contributions

Guosheng Zhao and Huan Chen contributed equally to this work.

-Funding

This present research work was supported by the National Natural Science Foundation of China (No.61202458, 61403109), the Natural Science Foundation of Heilongjiang Province of China (No.LH2020F034).

-Consent for publication

We would like to submit the manuscript entitled “A Lightweight Block Encryption Algorithm for Narrowband Internet of Things”, which we wish to be considered for publication in “Peer-to-Peer Networking and Applications”.

References

1. Wang Y. P. Eric,Lin Xingqin,Adhikary Ansuman,Grovlen Asbjorn,Sui Yutao,Blankenship Yufei,Bergman Johan,Razaghi Hazhir S. (2017) A Primer on 3GPP Narrowband Internet of Things[J]. IEEE Communications Magazine, 55(3): 117-123.
2. Oh Sung Min,Shin JaeSheung (2017) An Efficient Small Data Transmission Scheme in the 3GPP NB-IoT System. IEEE Communications Letters, 21(3): 660-663.
3. Zou Yulong, Ding Xiaojin, Wang Quanquan (2017) Key Technologies and Application Prospects of NB-IoT. ZTE Technology, 23(01): 43-46.
4. Xu Furong, Weng Weiwen, Zhang Chao, Bian Tingting, Ma Xinli, Gao Xiangdong, Li Xin, Cao Lei (2020) Research status and development trend of NB-IoT enhancement technology. Telecommunications Science, 36(02): 130-136.
5. Xiao Zhengrong, Zhang Yanlian (2017) Analysis of NB-IoTT and its development strategy[J]. World Telecom, (01):64-66+72.
6. Ratasuk, R., Mangalvedhe, N., Zhang, Y., Robert, M., & Koskinen, J. P. (2016). Overview of narrowband IoT in LTE Rel-13. In 2016 IEEE conference on standards for communications and networking (CSCN), pp 1-7.
7. Andres-Maldonado, P., Ameigeiras, P., Prados-Garzon, J., Navarro-Ortiz, J., & Lopez-Soler, J. M. (2017) Narrowband IoT

data transmission procedures for massive machine-type communications. Ieee Network, 31(6): 8-15.
8. Zhixin, S., & Hanshu, H. (2017) Some thoughts on security issues in NB-IoT [J]. ZTE technology, 1: 47-50.
9. Jianbing, S. (2017). Narrow band Internet of things (NB-IoT) application and security [J]. Information security and communication security, (06): 27-31.
10. Wang Baocheng and Li Shan. (2020) The Research of Security in NB-IoT. In Proceedings of the 2020 4th International Conference on Electronic Information Technology and Computer Engineering (EITCE 2020). Association for Computing Machinery, New York, NY, USA, pp 275–279.
11. Mo Jianhua (2010) An Encryption Algorithm for WSN Data Security. Zhejiang University of Technology.
12. Li Wenfeng, Du Yanhui (2009) The application of cryptography in network security [J]. Information Network Security, (04): 40-42+47.
13. Heys, H. M. (2002) A tutorial on linear and differential cryptanalysis. Cryptologia, 26(3): 189-221.
14. Kewen Zhan. (2019) Research on anti-power attack of lightweight encryption algorithm applied to narrowband Internet of Things. Southeast University.
15. Jia Rongyuan, Wang Yihuai, Wang Xiaoning. (2018) A Lightweight Encryption Scheme for Narrowband Internet of Things. Computer Engineering and Design, 39(10): 3039-3044.
16. Poschmann A. (2009) Lightweight cryptography-cryptographic engineering for a pervasive world. Cryptology ePrint Archive.
17. Thakor, V. A., Razzaque, M. A., & Khandaker, M. R. (2021). Lightweight cryptography algorithms for resource-constrained IoT devices: A review, comparison and research opportunities. IEEE Access, 9, pp 28177-28193.
18. Park, G. T., Han, H. J., & Lee, J. H. (2014). Design and implementation of lightweight encryption algorithm on OpenSSL. The Journal of Korean Institute of Communications and Information Sciences, 39(12): 822-830.
19. Alshamsi, A. Z., Barka, E. S., & Serhani, M. A. (2016, November). Lightweight encryption algorithm in wireless body area network for e-health monitoring. In 2016 12th International Conference on Innovations in Information Technology (IIT) pp 1-7.
20. Ruhan Bevi, A., Malarvizhi, S., & Saxena, S. (2014). RC5 implementation in ARM platform for cryptographic applications. International Journal of Information and Communication Technology, 6(3-4): 251-259.
21. Grossschadl, J., Tillich, S., Rechberger, C., Hofmann, M., & Medwed, M. (2007). Energy evaluation of software implementations of block ciphers under memory constraints. In 2007 Design, Automation & Test in Europe Conference & Exhibition. IEEE pp. 1-6.
22. Ge, Y. T., Liu, X. M., & Yin, X. T. (2014). Study on TEA Encryption for Public Communication Network Wireless Remoter. In Applied Mechanics and Materials (Vol. 565, pp. 179-182). Trans Tech Publications Ltd.
23. Zhang Yao, Ye Ling. (2015) WSN encryption algorithm based on AES [J]. Computer Engineering and Design, 36(03): 619-623.
24. Zhao Yuehua, Ma Linlin. (2015) Research on lightweight implementation of AES algorithm. Computer Engineering and Applications, 51(06): 79-83.
25. Chen, D., Qing, D., & Wang, D. (2012). AES key expansion algorithm based on 2D logistic mapping. In 2012 Fifth International Workshop on Chaos-fractals Theories and Applications. IEEE pp. 207-211.
26. Yan Lele, Li Hui. (2017) Dynamic key AES encryption algorithm based on complex chaotic sequence. Computer Science, 44(06): 133-138+160.
27. Matsui, M. (1993, May). Linear cryptanalysis method for DES cipher. In Workshop on the Theory and Application of of Cryptographic Techniques, Springer, Berlin, Heidelberg, pp. 386-397.
28. Biham, E., & Shamir, A. (1991) Differential cryptanalysis of DES-like cryptosystems. Journal of CRYPTOLOGY, 4(1): 3-72.
29. Yue, L., Wei, L., Yanqin, C., & Jiajin, L. (2016). Performance analysis of several lightweight block cipher [J]. Computer application and software, 33(10): 317-320.
30. Courtois, N. T., & Pieprzyk, J. (2002). Cryptanalysis of block ciphers with overdefined systems of equations. In International conference on the theory and application of cryptology and information security, Springer, Berlin, pp 267-287.
31. Courtois, N. T., & Bard, G. V. (2007). Algebraic cryptanalysis of the data encryption standard. In IMA International Conference on Cryptography and Coding, Springer, Berlin, Heidelberg, pp. 152-169.
32. Webster, A. F., & Tavares, S. E. (1985). On the design of S-boxes. In Conference on the theory and application of cryptographic techniques, Springer, Berlin, Heidelberg, pp. 523-534.
33. Ramadan, R. A., Aboshosha, B. W., Yadav, K., Alseadoon, I. M., Kashout, M. J., & Elhoseny, M. (2021). Lbc-iot: lightweight block cipher for iot constraint devices. CMC-COMPUTERS MATERIALS & CONTINUA, 67(3): 3563-3579.
34. Bansod, G., Patil, A., Sutar, S., & Pisharoty, N. (2016). ANU: an ultra lightweight cipher design for security in IoT. Security and Communication Networks, 9(18): 5238-5251.
35. Li, L., Liu, B., Zhou, Y., & Zou, Y. (2018). SFN: A new lightweight block cipher. Microprocessors and Microsystems, 60, 138-150.
36. Zhang W, Bao Z, Lin D, et al. (2015) RECTANGLE: a bit-slice lightweight block cipher suitable for multiple platforms. Science China Information Sciences, 58(12): 1-15.
37. Ma, D., & Shi, Y. (2019, December). A lightweight encryption algorithm for edge networks in software-defined industrial Internet of Things. In 2019 IEEE 5th International Conference on Computer and Communications (ICCC), IEEE pp 1489-1493.

Guosheng Zhao, born in 1977. PhD, professor, Master supervisor. Senior member of CCF. His main research interests include cognitive network, and trusted computing.

Huan Chen, born in 1999, postgraduate. Her main research interests include network security and lightweight block cipher.

Jian Wang, born in 1979. PhD, professor, Doctoral supervisor. Member of CCF. Her main research interests include crowd sensing, cognitive network, SDN and survivability.