Microprocessors and Microsystems 60 (2018) 138–150

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

SFN: A new lightweight block cipher

Lang Li[a,b,*], Botao Liu[a,b,c,*], Yimeng Zhou[b], Yi Zou[a,b]

[a] College of Computer Science and Technology, Hengyang Normal University, Hengyang 421002, China
[b] Hunan Provincial Key Laboratory of Intelligent Information Processing and Application, Hengyang 421002, China
[c] College of Computer Science and Technology, Guizhou University, Guiyang 550025, China

ARTICLE INFO

Keywords:
Lightweight
Block cipher
SP network
Feistel network

ABSTRACT

During past few years, some lightweight block ciphers have been proposed. These lightweight block ciphers take single encryption method that either uses Substitution-Permutation (SP) network structure or Feistel network structure to encrypt. In this paper, we have designed a different encryption method that takes both SP network structure and Feistel network structure to encrypt. Current SP network has a limitation that the encryption and decryption processes are dissimilar. To solve this problem, we have employed involution related properties of the nonlinear and linear components to modify SP network structure. The modified one enables the encryption and decryption program or circuit to work as the Feistel network structure. Additionally, we have implemented a MixRows in SP network structure. Then we instantiate these three novel ideas into the lightweight block cipher called SFN. We have carried out the security evaluation and the hardware and software experiments to it. The result shows that compared to other lightweight block ciphers, SFN has more advantages in terms of being immune to attacks. Also, SFN is not only compact in hardware environment but also efficient in software platforms.

1. Introduction

Nowadays, embedded devices of IOT have been deployed to daily life. These devices are extremely stringent on power consumption and cost. The application of these devices has been implicated in some governmental, financial and health institutions. While the information of these devices is highly sensitive, and lightweight block ciphers are the main protective measurements of such information, which are getting more and more attention of researchers in related field.

At present, many scholars study on lightweight block cipher in the field of cryptography. These studies have focused on the design of lightweight block cipher, security analysis and performance evaluation, etc. In the aspect of the design of lightweight block cipher those research have achieved favorable results. A great quantity of lightweight block ciphers are proposed, such as QTL [1], LILLIPUT [2], KLEIN [3], LED [4], Midori [5], Zorro [6], mCrypton [7], PRESENT [8], PRIDE [9], PRINCE [10], Rectangle [11], GOST revisited [12], ITUbee [13], LBlock [14], RoadRunneR [15], SEA [16], SIMECK [17], SIMON and SPECK [18], Piccolo [19], TWINE [20], MIBS [21]. Lightweight block cipher is intended to consider its application to the target platform. Among these lightweight block ciphers, some are designed for the hardware implementation, some for the software, and some are comprehensively designed for both hardware and software implementations. For hardware design, the goal is to reduce the required amount of logic gates, like how QTL, Midori, mCrypton, PRESENT, PRINCE, GOST revisited, SIMECK, SIMON, Piccolo, MIBS do. For software design, the lightweight block cipher considers the typical implementation platform as the microprocessor, and the goal is to save more storage space, eliminate computation, and maximize throughput, like how Zorro, PRIDE, ITUbee, RoadRunneR, and SPECK function. In combining the advantages of both software and hardware design, the lightweight block cipher is often used in smart card and other devices, such as KLEIN, LED, Rectangle, LBlock, SEA, TWINE. Then the trade-off between safety, cost and efficiency are key problems to be resolved in the design of lightweight block cipher. In particular, the key length of lightweight block cipher, the total number of rounds, and algorithm of structure affect safety, cost and efficiency, but the security of cryptographic algorithm itself plays the most essential role.

Lightweight block cipher is facing the following problems: (1) Some lightweight block ciphers are pursuing smaller areas and higher efficiency. Their round functions are too simple or without key expansion, which leaves their security a big problem. Recent studies have shown that some lightweight block ciphers were threatened by some attack methods; for example, KLEIN-64 and Zorro are susceptible to the threats of Truncated Differential [22,23]; Piccolo-80 and TWINE are susceptible to the threats of Biclique [24,25], and PRINCE is susceptible

* Corresponding author at: College of Computer Science and Technology, Hengyang Normal University, Hengyang 421002, China.
E-mail addresses: lilang911@126.com (L. Li), teslal0505@foxmail.com (B. Liu).

https://doi.org/10.1016/j.micpro.2018.04.009
Received 2 September 2016; Received in revised form 5 December 2017; Accepted 21 April 2018
Available online 30 April 2018
0141-9331/

to the threats of Multiple differentials [26], and so on. (2) For most of dedicated chips, there is only a fixed cryptography mode; that is a chip embedded in a cipher algorithm. This implementation is difficult to meet the security requirements of different user levels. Although there are reconfigurable designs for two different cipher algorithms in chips, the safety of chips need to be improved. However, in the design process, between different algorithms we can extract some with the same or similar operation, but algorithms are various, which occupy considerable resources of the overhead area, resulting high cost in hardware implementation and largely reducing the efficiency of the performance. Meanwhile, flexibility is negatively affected. (3) Block cipher algorithm, with two types of typical structures within which one is a SP network structure, the other is a Feistel network structure. Compared with Feistel network structure, SP network has a considerable disadvantage that the encryption and decryption processes are dissimilar. In implementation, the SP network costs more resource. However, the benefit of applying SP network is the level diffusion. Its round function is able to change all block messages in an iterative round; therefore, its security is relatively high.

In order to solve the above problems, we propose a different encryption method and some components. We instantiate this method and these components into a lightweight block cipher.

1.1. Our contributions

In this paper, we present a different encryption method which uses both SP network structure and Feistel network structure to encrypt. The latest 32-bit primary key is the control signal which controls SP network structure and Feistel network structure for key expansion and encryption or decryption. When Feistel network structure is utilized for key expansion, SP network structure is used for encryption or decryption. This method utilizes a network structure to carry out not only key expansion but also encryption or decryption; thus, the key expansion is more advanced. The SP network structure and Feistel network structure contain some distinct operation components, in order to ensure key expansion and encryption or decryption operation of speed synchronization as far as possible.

The latest 32-bit primary key takes as the control signal to select one of the two structures to encrypt or decrypt while the other is selected to operate key expansion. There are $2^{32} = 4{,}294{,}697{,}296$ different ways of operation. Comparing to fixed structure in encryption or decryption and key expansion, the proposed encryption method can improve the immunity with a coefficient of 2^{32} against attacks, such as Differential and linear attacks, Meet-in-the-Middle attacks, Impossible Differential attack and Integral attack and so on.

Because the diffusion and confusion speeds of the two structures are different, we operate them separately, presenting different levels of diffusion and confusion. The new encryption is enabled to satisfy the security requirements of different user levels.

We aim to address the dissimilar encryption and decryption processes based on SP network structure. Involution related properties of the nonlinear and linear components are employed into the design of SP network structure. The modified SP network structure enables the encryption and decryption program or circuit to work as the Feistel network structure. Also, implementing the inverse of SP network structure becomes easier. The modified SP network structure, on one hand, saves the resource of decryption; on the other hand, it changes entire block messages in an iterative round. The block cipher providing a SP network structure for both encryption and decryption is one of the research purposes in the future.

In SP network structure, we apply a MixRows. The round function process of the SP network structure is illustrated as: AddRoundKey → S_1-box layer → MixColumns → MixRows → S_1-box layer. Assuming that we input 4 × 4 array (the 64-bit plaintext is equally divided into 16 4-bit, which are arranged a 4 × 4 array) where the first 4-bit is active and the others are not. After a round encryption of SFN, 4 × 4 array becomes all active. And AES achieves fast diffusion. After a round encryption of AES, the first column of the 4 × 4 array is active but the other three remain not. AES needs to be fully confused with 4 × 4 array data like SFN, requiring two rounds of encryption. In a round encryption, the confusion effect of SFN is faster than the AES's, and the number of active S-box is also more than its. Because MixRows and MixColumns use the same fixed diffusion matrix, implementation of MixRows requires less hardware and software resources. Thus, the round function of SP network structure achieves a higher security and a compact implementation in software and hardware.

Then we instantiate the above novel idea into the lightweight block cipher called SFN (this name is the abbreviation of SP network structure and Feistel network structure). The SFN fixes 64-bit block with 96-bit key. We have carried out the security evaluation to SFN. The maximum linear characteristic probability and the maximum differential characteristic probability is 2^{-488}. And the SFN is depicted with 32,256 quadratic equations of 12,288 variables. These results prove that SFN is more secure than other lightweight block ciphers. Due to the variety of 32-bit control keys, there are 2^{32} different ways of operation. This method is extremely effective against attacks, such as Differential and linear attacks, Meet-in-the-Middle attack, Impossible Differential attack and Integral attack. We have conducted the hardware and software experiments to SFN. SFN requires 1876.04 GE which is less than 2000 GE in hardware implementation, and it only takes 4736 cycles in software implementation. Thus, SFN is not only operation-efficient in software platforms but also compact in hardware environment, so that it can be applied to a constrained environment.

1.2. Structure of the paper

The rest of paper is organized as the following. In Section 2, the specification of SFN is introduced. In Section 3, we explain the design rationale for SFN. In Section 4, we present the result of security analysis. Section 5 contains the performance evaluation of its hardware and software implementations. And a conclusion is presented in Section 6.

2. Specification of SFN

For the cipher structure and each component of SFN, we give full consideration to make it achieves high security as well as high-performance and compact implementation in hardware environment. In the following part, we will depict SFN encryption and decryption algorithms as well as KeyExpansion algorithm in details.

2.1. Notations

In the paper, we use five notations as below.

P	permutation layer
⊕	bitwise exclusive-OR operation
||	concatenation of two binary strings
S	S-box layer
CON	round Constants

SFN is a special structure which includes a SP network and a Feistel network. The block length is 64-bit, and the key length is 96-bit. The 96-bit key is divided into two parts where one is the front 64-bit to carry out AddRoundKey and KeyExpansion, and the other is rest 32-bit to act as control signal. Additionally, this signal controls two network structures where one is responsible for encryption or decryption, and the other is for KeyExpansion. Each bit of the control signal conducts one round operation, and the SFN consists of 32 rounds. Details of the operation (see Fig. 1): when the bit of the control signal is 0, SFN selects SP network structure to run encryption or decryption while Feistel network structure carries out KeyExpansion. When the bit of the signal is 1, SFN selects Feistel network structure to run encryption or

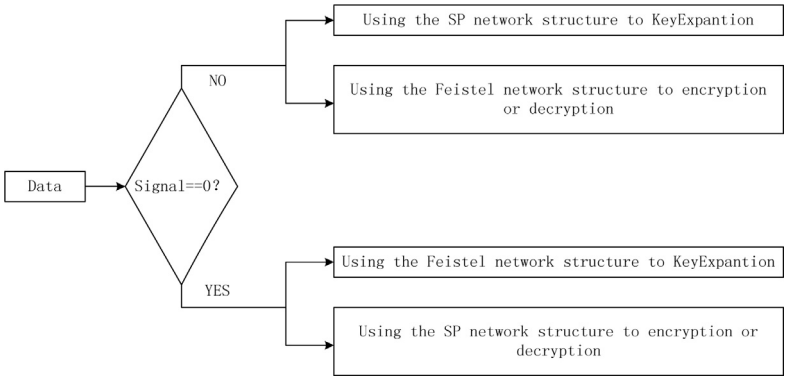


Fig. 1. The control signal of operating.

decryption while the SP network structure works for KeyExpansion.

When SP network structure carries out encryption or decryption, its round function is composed of four: AddRoundKey, S_1, MixColumns and MixRows. When the SP network structure carries out KeyExpansion, its round function includes 4 components: AddConstants, S_2, MixColumns and MixRows. Simultaneously, when the Feistel network structure carries out encryption or decryption, the round function of it includes 4 components: AddRoundKey, P, MixXors and S_2, and when it carries out KeyExpansion, its round function consists of four: AddConstants, P, MixXors and S_1.

The 64-bit plaintext and intermediate state is displayed as $STATE \in \{0, 1\}^{64}$, and the 16 4-bit nibbles $STATE_j \in \{0, 1\}^4 (0 \le j \le 15)$ are arranged in the following 4 × 4 array:

$$STATE = \begin{bmatrix} STATE_{0(4)} & STATE_{1(4)} & STATE_{2(4)} & STATE_{3(4)} \\ STATE_{4(4)} & STATE_{5(4)} & STATE_{6(4)} & STATE_{7(4)} \\ STATE_{8(4)} & STATE_{9(4)} & STATE_{10(4)} & STATE_{11(4)} \\ STATE_{12(4)} & STATE_{13(4)} & STATE_{14(4)} & STATE_{15(4)} \end{bmatrix}$$

The 96-bit primary key is divided into the 64-bit round key and the 32-bit control key, where the 64-bit round key is expressed as $RK \in \{0, 1\}^{64}$ and the 32-bit control key is expressed as $CK \in \{0, 1\}^{32}$. The RK is carried out AddRoundKey and KeyExpansion, and the 16 4-bit nibbles $RK_j \in \{0, 1\}^4$ $(0 \le j \le 15)$ are also arranged in the following 4 × 4 array:

$$RK = \begin{bmatrix} RK_{0(4)} & RK_{1(4)} & RK_{2(4)} & RK_{3(4)} \\ RK_{4(4)} & RK_{5(4)} & RK_{6(4)} & RK_{7(4)} \\ RK_{8(4)} & RK_{9(4)} & RK_{10(4)} & RK_{11(4)} \\ RK_{12(4)} & RK_{13(4)} & RK_{14(4)} & RK_{15(4)} \end{bmatrix}$$

The $CK = CK_{0(1)} \| CK_{1(1)} \| ... \| CK_{30(1)} \| CK_{31(1)}$ is regarded as the control signal, and each bit of the control signal conducts one round operation.

2.2. Encryption algorithm

We will further describe the encryption algorithm steps of SFN. The number of SFN iterative rounds is 32. Additionally, Fig. 2 illustrates the encryption procedure.

The 32 rounds data processing of SFN can be expressed as ENC_{32}. ENC_{32} inputs a plaintext data $STATE$, a primary key (64-bit RK and 32-bit CK), and outputs a ciphertext data. Then ENC_{32} is defined as below:

$$ENC_{32}: \left\{ \begin{array}{c} \{0, 1\}^{64} \times \{0, 1\}^{96} \to \{0, 1\}^{64} \\ (Plaintext_{(64)}, RK_{(64)}, CK_{(32)}) \to Ciphertext_{(64)} \end{array} \right\}$$

ENC_{32} is also described as Algorithm 1. Specifically, the KeyExpansion illustrates RK and CK in details.

The encryption of SFN is involved with a SP network structure and a Feistel network structure. We will describe the function components of both structures in details as below.

When SP network structure carries out encryption or decryption, the round function process of it is illustrated as: AddRoundKey → S_1-box layer → MixColumns → MixRows → S_1-box layer (see Fig. 3).

Here, AddRoundKey updates a 64-bit data in the round function of SP network structure: $\{0, 1\}^{64} \times \{0, 1\}^{64} \to \{0, 1\}^{64}$.

AddRoundKey: Given a 64-bit $RK_{(64)}$ ($RK_{0(4)}$, $RK_{1(4)}$, ..., $RK_{14(4)}$, $RK_{15(4)}$) and a 64-bit $STATE_{(64)}$ ($STATE_{0(4)}$, $STATE_{1(4)}$, ..., $STATE_{14(4)}$, $STATE_{15(4)}$), then

$$STATE_{j(4)} \leftarrow STATE_{j(4)} \oplus RK_{j(4)} \ (0 \le j \le 15)$$

The S_1-box layer is different from the S_2-box layer, but they are both 4 × 4 S-boxes. Table 1 shows the values of S_1-box. Then the S_1-box layer updates a 64-bit $STATE_{(64)}$ ($STATE_{0(4)}$, $STATE_{1(4)}$, ..., $STATE_{14(4)}$, $STATE_{15(4)}$) as below:

$$STATE_{j(4)} \leftarrow S_1(STATE_{j(4)}) \ (0 \le j \le 15)$$

The MixColumns: Given the diffusion matrix M in hexadecimal form as follows:

$$M = \begin{pmatrix} 1 & 2 & 6 & 4 \\ 2 & 1 & 4 & 6 \\ 6 & 4 & 1 & 2 \\ 4 & 6 & 2 & 1 \end{pmatrix}$$

The matrix M has the involutive property. The multiplications between the matrix M and the $STATE_{(64)}$ 4 × 4 array are performed over $GF(2^4)$ as below, where the $GF(2^4)$ is expressed as the an irreducible polynomial $x^4 + x + 1$. That is, the MixColumns is the matrix M multiplied by $STATE_{(64)}$ 4 × 4 array.

$$STATE_{(64)} \leftarrow \begin{pmatrix} 1 & 2 & 6 & 4 \\ 2 & 1 & 4 & 6 \\ 6 & 4 & 1 & 2 \\ 4 & 6 & 2 & 1 \end{pmatrix} \cdot \begin{pmatrix} STATE_{0(4)} & STATE_{1(4)} & STATE_{2(4)} & STATE_{3(4)} \\ STATE_{4(4)} & STATE_{5(4)} & STATE_{6(4)} & STATE_{7(4)} \\ STATE_{8(4)} & STATE_{9(4)} & STATE_{10(4)} & STATE_{11(4)} \\ STATE_{12(4)} & STATE_{13(4)} & STATE_{14(4)} & STATE_{15(4)} \end{pmatrix}$$

The MixRows: The diffusion matrix M is also used in the MixRows. Then the multiplications between the $STATE_{(64)}$ 4 × 4 array and matrix M are performed over $GF(2^4)$ as the following. That is, the MixRows is the $STATE_{(64)}$ 4 × 4 array multiplied by matrix M.

$$STATE_{(64)} \leftarrow \begin{pmatrix} STATE_{0(4)} & STATE_{1(4)} & STATE_{2(4)} & STATE_{3(4)} \\ STATE_{4(4)} & STATE_{5(4)} & STATE_{6(4)} & STATE_{7(4)} \\ STATE_{8(4)} & STATE_{9(4)} & STATE_{10(4)} & STATE_{11(4)} \\ STATE_{12(4)} & STATE_{13(4)} & STATE_{14(4)} & STATE_{15(4)} \end{pmatrix} \cdot \begin{pmatrix} 1 & 2 & 6 & 4 \\ 2 & 1 & 4 & 6 \\ 6 & 4 & 1 & 2 \\ 4 & 6 & 2 & 1 \end{pmatrix}$$

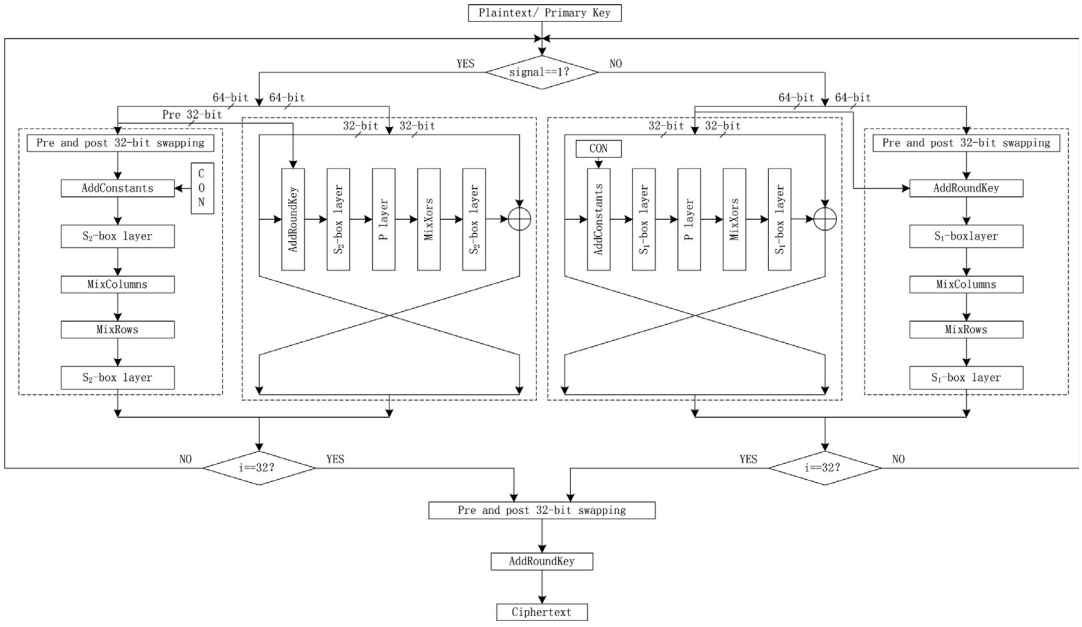


Fig. 2. The encryption procedure of SFN.

Algorithm 1
The encryption routine of SFN.

ENC_{32}
Input: $Plaintext_{(64)}$, $RK_{(64)}$, $CK_{(32)}$;
Output: $Ciphertext_{(64)}$;
$STATE^0_{(64)} \leftarrow Plaintext_{(64)}$;
for i = 0 to 31 do
  if ($CK_{i(1)} = 0$)
    $M^i_{(64)} \leftarrow STATE^i_{8(4)}||...||STATE^i_{15(4)}||STATE^i_{0(4)}||...||STATE^i_{7(4)}$;
    $STATE^i_{(64)} \leftarrow M^i_{(64)}$;
    AddRoundKey($STATE^i_{(64)}$, $RK^i_{(64)}$);
    S_1-box layer ($STATE^i_{(64)}$);
    MixColumns ($STATE^i_{(64)}$);
    MixRows ($STATE^i_{(64)}$);
    S_1-box layer ($STATE^i_{(64)}$);
    $STATE^{i+1}_{(64)} \leftarrow STATE^i_{(64)}$;
  end if
  if ($CK_{i(1)} = 1$)
    $X^{2i}_{(32)} \leftarrow STATE^i_{0(4)}||...||STATE^i_{7(4)}$, $X^{2i+1}_{(32)} \leftarrow STATE^i_{8(4)}||...||STATE^i_{15(4)}$;
    $K^i_{(32)} \leftarrow RK^i_{0(4)}||...||RK^i_{7(4)}$;
    AddRoundKey($X^{2i}_{(32)}$, $K^i_{(32)}$);
    S_2-box layer ($X^{2i}_{(32)}$);
    P layer ($X^{2i}_{(32)}$);
    MixXors ($X^{2i}_{(32)}$);
    S_2-box layer ($X^{2i}_{(32)}$);
    $X^{2i}_{(32)} \leftarrow X^{2i}_{(32)} \oplus X^{2i+1}_{(32)}$, $X^{2i+1}_{(32)} \leftarrow STATE^i_{0(4)}||...||STATE^i_{7(4)}$;
    $STATE^{i+1}_{(64)} \leftarrow X^{2i}_{(32)}||X^{2i+1}_{(32)}$;
  end if;
end for;
$M^{32}_{(64)} \leftarrow STATE^{32}_{8(4)}||...||STATE^{32}_{15(4)}||STATE^{32}_{0(4)}||...||STATE^{32}_{7(4)}$;
$STATE^{32}_{(64)} \leftarrow M^{32}_{(64)}$;
AddRoundKey($STATE^{32}_{(64)}$, $RK^{32}_{(64)}$);
$Ciphertext_{(64)} \leftarrow STATE^{32}_{(64)}$;
Return $Ciphertext_{(64)}$;

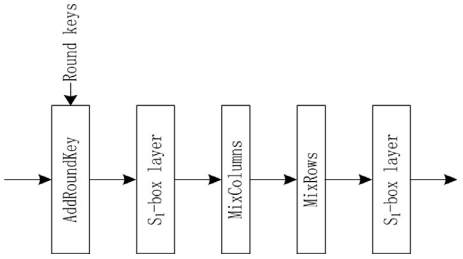


Fig. 3. The SP network structure of encryption.

Table 1
4-bit bijective S-box S_1 in hexadecimal form.

X	0	1	2	3	4	5	6	7
S_1[x]	C	A	D	3	E	B	F	7
X	8	9	A	B	C	D	E	F
S_1[x]	8	9	1	5	0	2	4	6

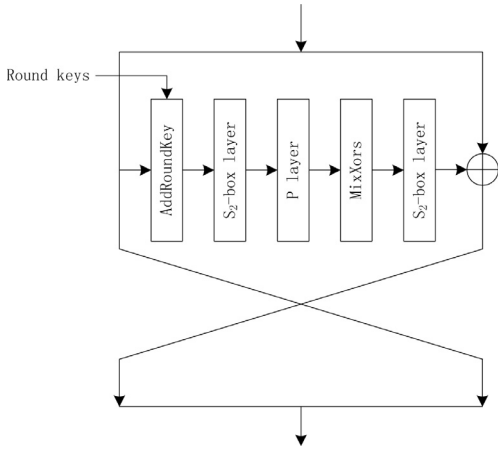


Fig. 4. The Feistel network structure of encryption.

Table 2
The bit permutation of P.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P(j)$	9	28	7	13	8	12	29	6	0	2	17	23	30	24	18	11
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$P(j)$	31	4	15	19	5	1	25	27	3	10	22	21	26	16	20	14

When Feistel network structure carries out encryption or decryption, the round function process of it is illustrated as: AddRoundKey → S_2-box layer → P layer → MixXors → S_2-box layer (see Fig. 4).

The AddRoundKey updates a 32-bit data in the round function of the Feistel network structure: $\{0, 1\}^{32} \times \{0, 1\}^{32} \rightarrow \{0, 1\}^{32}$. AddRoundKey: Given a 32-bit $(RK_{0(4)}, ..., RK_{7(4)})$ and a 64-bit $(STATE_{0(4)}, ..., STATE_{7(4)})$, then

$$STATE_{j(4)} \leftarrow STATE_{j(4)} \oplus RK_{j(4)} \ (0 \le j \le 7)$$

where P layer is bit permutations. Then the P layer can be expressed as P: $\{0, 1\}^{32} \rightarrow \{0, 1\}^{32}$. Bit j $(0 \le j \le 31)$ of 32-bit $(STATE_{0(4)}, ..., STATE_{7(4)})$ moves to bit located at $P(j)$. The P is a fixed set of random sequences by drawing lots, which is shown in Table 2.

The MixXors: the MixXors layer is a linear transformation. Then the MixXors layer is $(GF(2)^4)^8 \rightarrow (GF(2)^4)^8$ and is an exclusive OR operation among $STATE_{0(4)}, ..., STATE_{7(4)}$. The MixXors updates $STATE_{j(4)} (0 \le j \le 7)$ as follows (see Fig. 5).

$$\begin{aligned}
STATE_{0(4)} &\leftarrow STATE_{1(4)} \oplus STATE_{2(4)} \oplus STATE_{3(4)} \oplus STATE_{4(4)} \\
&\quad \oplus STATE_{5(4)} \oplus STATE_{6(4)} \\
STATE_{1(4)} &\leftarrow STATE_{0(4)} \oplus STATE_{2(4)} \oplus STATE_{3(4)} \oplus STATE_{5(4)} \\
&\quad \oplus STATE_{6(4)} \oplus STATE_{7(4)} \\
STATE_{2(4)} &\leftarrow STATE_{0(4)} \oplus STATE_{1(4)} \oplus STATE_{3(4)} \oplus STATE_{4(4)} \\
&\quad \oplus STATE_{6(4)} \oplus STATE_{7(4)} \\
STATE_{3(4)} &\leftarrow STATE_{0(4)} \oplus STATE_{1(4)} \oplus STATE_{2(4)} \oplus STATE_{4(4)} \\
&\quad \oplus STATE_{5(4)} \oplus STATE_{7(4)} \\
STATE_{4(4)} &\leftarrow STATE_{0(4)} \oplus STATE_{1(4)} \oplus STATE_{3(4)} \oplus STATE_{4(4)} \\
&\quad \oplus STATE_{5(4)} \\
STATE_{5(4)} &\leftarrow STATE_{0(4)} \oplus STATE_{1(4)} \oplus STATE_{2(4)} \oplus STATE_{5(4)} \\
&\quad \oplus STATE_{6(4)} \\
STATE_{6(4)} &\leftarrow STATE_{1(4)} \oplus STATE_{2(4)} \oplus STATE_{3(4)} \oplus STATE_{6(4)} \\
&\quad \oplus STATE_{7(4)} \\
STATE_{7(4)} &\leftarrow STATE_{0(4)} \oplus STATE_{2(4)} \oplus STATE_{3(4)} \oplus STATE_{4(4)} \\
&\quad \oplus STATE_{7(4)}
\end{aligned}$$

The S_2-box layer: Table 3 shows the values of S_2-box. Then the S_2-box layer updates a 32-bit $(STATE_{0(4)}, ..., STATE_{7(4)})$ as follows.

Table 3
4-bit bijective S-box S_2 in hexadecimal form.

X	0	1	2	3	4	5	6	7
S_2[x]	B	F	3	2	A	C	9	1
X	8	9	A	B	C	D	E	F
S_2[x]	6	7	8	0	E	5	D	4

Algorithm 2
The KeyExpansion routine of SFN.

KEYE$_{32}$
Input: $RK^0_{(64)}$, $CK_{(32)}$;
Output: $RK^{i+1}_{(64)}$;
1: for $i = 0$ to 31 do
2: if $(CK_{i(1)} = = 0)$
3: $X^{2i}_{(32)} \leftarrow RK^i_{0(4)} || ... || RK^i_{7(4)}$, $X^{2i+1}_{(32)} \leftarrow RK^i_{8(4)} || ... || RK^i_{15(4)}$;
4: AddConstants $\left(X^{2i}_{(32)}, i\right)$;
5: S_1-box layer $(X^{2i}_{(32)})$;
6: P layer $(X^{2i}_{(32)})$;
7: MixXors$(X^{2i}_{(32)})$;
8: S_1-box layer $(X^{2i}_{(32)})$;
9: $X^{2i}_{(32)} \leftarrow X^{2i}_{(32)} \oplus X^{2i+1}_{(32)}$, $X^{2i+1}_{(32)} \leftarrow RK^i_{0(4)} || ... || RK^i_{7(4)}$;
10: $RK^{i+1}_{(64)} \leftarrow X^{2i}_{(32)} \ \| X^{2i+1}_{(32)}$;
11: end if
12: if $(CK_{i(1)} = = 1)$
13: $K^i_{(64)} \leftarrow RK^i_{8(4)} || ... || RK^i_{15(4)} \ || RK^i_{0(4)} || \cdots || RK^i_{7(4)}$;
14: $RK^i_{(64)} \leftarrow K^i_{(64)}$;
15: AddConstants $\left(RK^i_{(64)}, i\right)$;
16: S_2-box layer $(RK^i_{(64)})$;
17: MixColumns $(RK^i_{(64)})$;
18: MixRows $(RK^i_{(64)})$;
19: S_2-box layer $(RK^i_{(64)})$;
20: $RK^{i+1}_{(64)} \leftarrow RK^i_{(64)}$;
21: end if;
22: **Return** $RK^{i+1}_{(64)}$;
23: end for;

$$STATE_{j(4)} \leftarrow S_2(STATE_{j(4)}) \ (0 \le j \le 7)$$

2.3. KeyExpansion algorithm

The KeyExpansion produces 32-round keys from $RK^1_{(64)}$ to $RK^{32}_{(64)}$ via the 64-bit of the primary key. In order to improve security, we make the KeyExpansion algorithm structure the same with encryption and

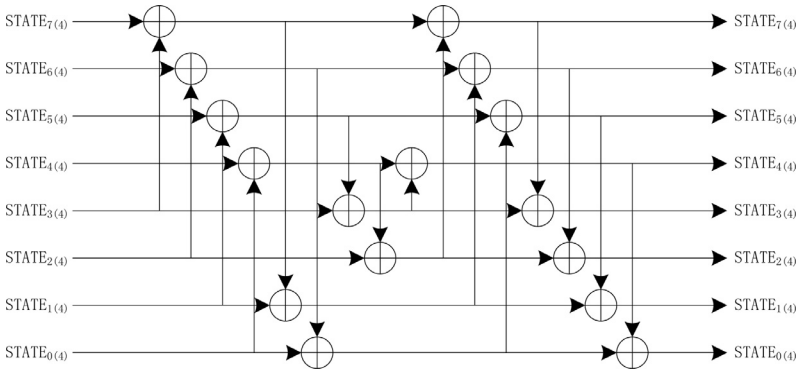


Fig. 5. The operation of MixXors.

Table 4
The *CON* in hexadecimal form.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CON	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CON	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F

decryption algorithm structure, and KeyExpansion is also consisted of the SP and the Feistel network structure. The 32-round key expansion process of SFN can be expressed as $KEYE_{32}$. Then $KEYE_{32}$is defined as below:

$$KEYE_{32}: \begin{Bmatrix} \{0,1\}^{96} \rightarrow \{0,1\}^{64} \\ (RK^{0}_{(64)}, CK_{(32)}) \rightarrow RK^{i+1}_{(64)} (0 \le i \le 31) \end{Bmatrix}$$

$KEYE_{32}$ is also described as Algorithm 2.

When Feistel network structure runs KeyExpansion, the round function process of it is illustrated as: AddConstants → S_1-box layer → P layer → MixXors → S_1-box layer. When SP network structure takes over KeyExpansion, the round function process is illustrated as: AddConstants → S_2-box layer → MixColumns → MixRows → S_2-box layer. Because KeyExpansion algorithm structure has the same encryption and decryption algorithm structure, we only need to explain AddConstants of the round function as below.

The AddConstants: to save hardware and software resource, we define *i*-th ($0 \le i \le 31$) as round constant *CON*. Table 4 shows the *CON*. Then AddConstants updates an 8-bit ($RK_{0(4)}$, $RK_{1(4)}$) as follows.

$$K^{i}_{(8)} \leftarrow RK^{i}_{0(4)} \; \| RK^{i}_{1(4)},$$

$$K^{i}_{(8)} \leftarrow K^{i}_{(8)} \oplus i \; (0 \le i \le 31)$$

2.4. Decryption algorithm

In this paper, SP network structure is designated to make the encryption and decryption processes into one program or circuit as the Feistel network structure does. The decryption algorithm of SFN is quite similar to the encryption of it. Because the decryption is the inverse of the encryption procedure, the round keys are used in a reverse order. We show the test vectors of SFN in Appendix I.

3. Design rationale

3.1. Algorithm structure

It has been stated that the AES replaces the DES in NIST as the security of DES is not enough. When we design SFN, the goal is to improve the security and have a more flexible lightweight block cipher. Moreover, the SFN has a balanced performance in the mixed environments of resource-constrained hardware and software. We propose a different encryption method to realize these purposes. The latest 32-bit primary key is the control signal, controlling SP network structure and Feistel network structure for key expansion and encryption or decryption. Thus, this explains that SFN contains two different encryption algorithms. When Feistel network structure is used for key expansion, SP network structure is used for encryption or decryption. This method utilizes a network structure to run not only key expansion but also encryption or decryption; thus, the key expansion is more advanced. And this method operates efficiently in diffusion and confusion, because weak key will not exist, so that this kind of key expansion will be protected from relating key attacks. Hence, compared with some other lightweight block ciphers (LED, ITUbee) without key expansions, SFN improves the security. In order to keep a stable implementation performance under the mixed environments of resource-constrained hardware and software, the key length is 96-bit (the 64-bit round key and the 32-bit control key) where only the 64-bit round key operates in the algorithm. Specifically, the 32-bit control key takes as control signal which selects one of the two structures to encrypt or decrypt and the other to operate key expansion. There are $2^{32} = 4{,}294{,}697{,}296$ different ways of operation. Comparing to fixed structures of encrypting or decrypting and operating key expansion, the proposed method can improve the immunity with a coefficient of 2^{32} from attacks. Because the speed of the diffusion and confusion structures is different, for an iterative round, the SP network structure can change all block messages, but the Feistel network structure only changes half of them. We run a random encryption, showing different levels of diffusion and confusion. The new encryption method can satisfy the security requirements of different user levels. We take two special cases as examples. On one hand, if we select the 32-bit control key as 32 one-bit 0 data, the SFN only uses the SP network structure to encrypt. On the other, if we select the 32-bit control key as 32 one-bit 1 data, the SFN takes advantage of the Feistel network structure to encrypt. Thus, compared with the second case, the first case is about double the levels of diffusion and confusion. We do not need to consider a variety of lengths for the other lightweight block ciphers, such as Midori, LED, Piccolo and PRESENT of two key lengths, KLEIN and mCrypton of three key lengths.

3.2. The modified SP network structure

In current block ciphers, the SP network has a better diffusion effect compared with Feistel network structure, but the SP network has an obvious limitation that the encryption and decryption processes are dissimilar. We aim to address the dissimilar encryption and decryption processes of SP network structure. We apply involution related properties of the nonlinear and linear components to reform SP network structure. The modified one makes encryption and decryption program or circuit work as the Feistel network structure. The round function of encryption and decryption of the SP modified network structure is illustrated as: AddRoundKey → S_1-box layer → MixColumns → MixRows → S_1-box layer. These components have the involution properties. The modified SP network structure, on one hand, can save the resource of decryption while on the other, it achieves extremely fast diffusion at each round. The round function uses several linear layers between two S-box layers. The design method has been applied to F function of Piccolo and ITUbee which belongs to Feistel network structure. Specifically, we know that the maximum differential probability (MDP) of an S-box is 2^{-2}, and the maximum linear probability (MLP) of an S-box is 2^{-2}. Moreover, compared with round function of Piccolo, we have approximately doubled the number of S-boxes. What is more important, in the SP network structure, we apply a MixRows. Assuming we input 4 × 4 array where the first 4-bit is active and the others are not, after a round encryption of SFN, 4 × 4 array becomes all active. The confusion effect of SFN is faster than that of AES, and the number of active S-box is also more than its. Because MixRows and MixColumns use the same fixed diffusion matrix, implementation of MixRows does not need to cost a lot of hardware and software resources. Hence, the design of modified SP network structure is sufficiently secure, and SFN has enough linearly and differentially active S-boxes over a certain number of rounds. Furthermore, the round function of encryption and decryption uses two S_1-box layers where the Delay of the S_1-box layer is 0.24 ns, and the sum of two S_1-box layers is 0.48 ns. Compared to S-box layer of PRESENT (0.47 ns), the sum of these two S_1-box layers is just more than 0.01 ns. Thus, the modified SP network structure has higher security and remains sufficient efficiency.

3.3. P layer and addconstants

When designing the P layer, our focus is improving the diffusion and reducing the hardware implementation cost. The P layer is bit permutations of 32-bits. And it is fixed sets of random sequences from drawing

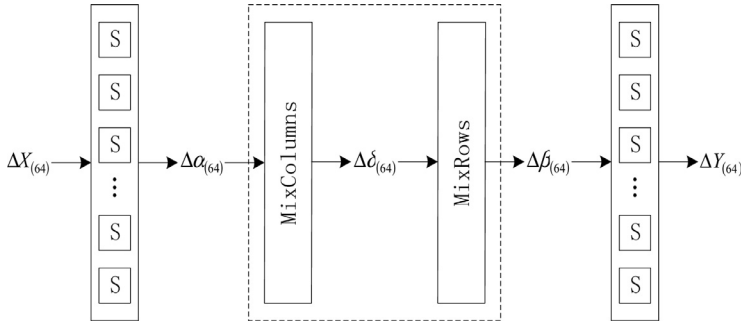


Fig. 6. The differential of *F* function in SP network structure.

lots. The method has random and equal probability. In hardware and software environments, it is extremely easy to be implemented and help improve the security. We use the AddConstants in KeyExpansion of SFN. In order to save more hardware and software resource, we regard the *i*-th ($0 \le i \le 31$) as *CON*. The AddConstants enables SFN to resist self-similarity attacks.

3.4. S-box layer

The proposed SFN uses two different 4 × 4 S-box layers. The S_1-box is the S-box of Midori-64, and the S_2-box is the S-box of PRINCE. The S-box layer is the nonlinear component. Their choice is of crucial importance for the SFN security. Furthermore, to improve the SFN security, we use two high-security S-boxes in SFN. The two S-boxes fulfill two basic requirements: (1) The maximal absolute bias of a linear approximation is 2^{-2}. (2) The maximal probability of a differential is 2^{-2}. The two S-boxes are proved to be highly secure by their designers. The S_1-box layer has an involution property. We take into account that the SFN is applied to resource-constrained environments. We should seek to save hardware resources, but we use two different S-box layers in the SFN. The S_1-box only needs 13.3 GE while the S_2-box only needs 16 GE, and the sum is 29.3 GE. Comparing to S-box of PRESENT (24.33 GE), the sum of these two S-boxes is only more than 4.97 GE. But we use two different S-box layers to improve security for SFN. As a result, we decide to use these two S-box layers in respect of improving sufficient security and at the same time remaining moderate hardware efficiency.

3.5. MixColumns,MixRows and MixXors

The MixColumns and the MixRows have a good linear diffusion property, so we choose the 4 × 4 MDS matrix *M* which is suitable for compact implementation. To improve security of SFN, we give up the circulant-type almost MDS matrices which are adopted in Midori, PRINCE and FIDES [27]. The MDS matrices have more branch numbers, so their diffusion speed is faster, and the maximum number of active S-boxes in each round is larger than almost MDS matrices. The MDS matrix *M* is an involution. The MixColumns is the matrix *M* multiplied by the $STATE_{(64)}$ 4 × 4 array, but the MixRows is the $STATE_{(64)}$ 4 × 4 array multiplied by the matrix *M*.The confusion mode of the MixColumns is different from the MixRows.

The MixXors is also a good linear diffusion transformation, which is used also in round function of MIBS. The branch number of MixXors is 5 and the computational complexity of MixXors is very low. So the MixXors is suitable for compact implementation and uses the round function of Feistel of SFN that is more secure against differential and linear attacks.

4. Security analysis

4.1. Differential and linear attacks

Differential and linear attacks [28,29] are the most typical approaches that have been applied to attack iterative block cipher, and they are also important indicators to measure the security of block ciphers. Therefore, when we design a high-security lightweight block cipher, we should consider these two attacks. In order to assess that the SFN is able to resist differential and linear attacks, we choose to count the number of active *F* functions which are involved in the number of linear and differential active S-boxes. The more the number of linear and differential active S-boxes is, the less the linear and differential probabilities are. Additionally, the security against the differential cryptanalysis and linear attacks can be estimated in similar ways [30,31]. The round function of modified SP network structure uses some linear layers between two S-box layers, and these linear layers do not contain the AddRoundKey. We define this design of round function as *F* function.

Definition 1. A differentially active *F* function is defined as a *F* function given a non-zero input difference. A linearly active *F* function is defined as a *F* function given a non-zero output mask value.

Definition 2. We define *n* as the block length, for any given $\Delta X, \Gamma X, \Delta Y, \Gamma Y \in \{0, 1\}^n$, and the differential and linear probabilities of *F* function are defined as:

$$\underset{X}{DP}(\Delta X, \Delta Y) = \underset{X}{\mathrm{Prob}}\{F(X) \oplus F(X \oplus \Delta X)\} = \Delta Y$$

$$\underset{X}{LP}(\Gamma X, \Gamma Y) = \left(2 \cdot \underset{X}{\mathrm{Prob}}\{\Gamma X \cdot X = \Gamma Y \cdot F(X)\} - 1\right)^2$$

Definition 3. We assume MDP_F and MLP_F are the maximum differential and linear probabilities of all *F* functions $\{F_1, F_2, F_3, \ldots\}$.

$$MDP_F = \max_i \max_{\Delta X \neq 0, \Delta Y} \underset{X}{DP}(\Delta X, \Delta Y)$$

$$MLP_F = \max_i \max_{\Gamma Y \neq 0, \Delta X} \underset{X}{LP}(\Gamma X, \Gamma Y)$$

With a novel encryption method taking both SP and Feistel network structures to encrypt and decrypt, Fig. 6 shows the differential of *F* function in SP network structure, and Fig. 7 shows the differential of *F* function in Feistel network structure.

When operating the SP network structure, for one round, if it is active in 64-bit input, there will be 2 active *F* functions. But when operating Feistel network structure, for one round, if it is active in 32-bit input, 1*F* function is active.

In *F* function, the maximal branch number of the linear layer is 5. When S_1-box is the S-box of Midori-64 and the S_2-box is the S-box of PRINCE, these two S-boxes have been proved to be secure. Thus, we

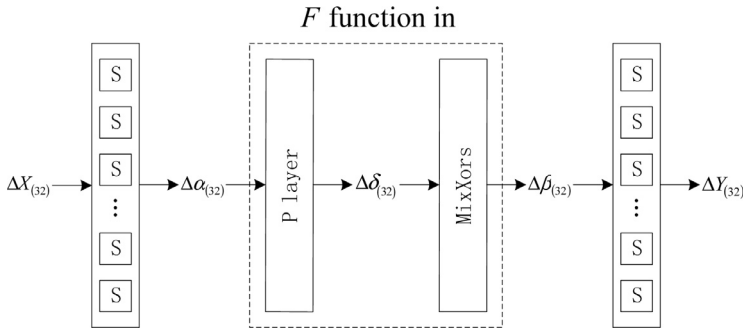


Fig. 7. The differential of *F* function in Feistel network structure.

Table 5
Active *F* functions for 16 consecutive rounds of the SP network structure.

Rounds	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Active *F* functions	0	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29

obtain that both *MDP* and *MLP* of an S-box are 2^{-2}.

The maximum differential and linear probabilities of the *F* function are

$$MDP_F = (MDP_S)^{B-1} = (2^{-2})^{5-1} = 2^{-8}$$

$$MLP_F = (MLP_S)^{B-1} = (2^{-2})^{5-1} = 2^{-8}.$$

Table 5 shows the minimum numbers of active *F* functions for 16 consecutive rounds of the SP network structure. The maximum differential characteristic probability is $(2^{-8})^{29} = 2^{-232}$, and the maximum linear characteristic probability is $(2^{-8})^{29} = 2^{-232}$.

Table 6 shows the minimum numbers of active *F* functions for 16 consecutive rounds of the Feistel network structure. The maximum differential characteristic probability is $(2^{-8})^{15} = 2^{-120}$, and the maximum linear characteristic probability is $(2^{-8})^{15} = 2^{-120}$.

Table 7 shows the minimum numbers of active *F* functions for 8 consecutive rounds of the Feistel network structure and 8 consecutive rounds of the SP network structure. The maximum differential characteristic probability is $(2^{-8})^{23} = 2^{-184}$, and the maximum linear characteristic probability is $(2^{-8})^{23} = 2^{-184}$.

Based on above results, we confirm that 16 rounds SFN can resist the differential and linear attacks. Furthermore, the SFN has 32 rounds of iterative operation, whereas it is secure enough against differential and linear attacks.

4.2. Related-key attack

Related-key attack [32] is a form of cryptanalysis. We are able to prove that SFN has great resistance to this attack. We divide the 96-bit key into two parts: the 64-bit *RK* and 32-bit *CK*. The *RK* carries out AddRoundKey and KeyExpansion, and each individual unit of the *RK* is not directly relied on each other. Lately, there are some lightweight block ciphers which either exclude KeyExpansion or have too simple KeyExpansion. Without extra protection, these ciphers are vulnerable to related-key attacks. Thus, we prove that SFN offers immunity to related-key attacks with high related-key differential characteristic probability. We have evaluated the number of active S-boxes in KeyExpansion. We

Table 6
Active *F* functions for 16 consecutive rounds of the Feistel network structure.

Rounds	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Active *F* functions	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Table 7
Active *F* functions for each 8 consecutive rounds of these two structures.

Rounds	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Active *F* functions	0	1	2	3	4	5	6	7	9	11	13	15	17	19	21	23

then propose that KeyExpansion algorithm structure is the same with encryption and decryption algorithm structure in SFN, as it was explained in Section 4.1 where we have obtained the number of active S-boxes in the encryption. In a similar way, the number of active S-boxes in KeyExpansion has been attained. SFN has 32 iterative rounds. In the first case, when we use SP network structure for KeyExpansion, the minimum number of active *F* functions for 32 consecutive rounds is 61, and the maximum related-key differential character is $(2^{-8})^{61} = 2^{-488}$, 64-bit round key data can be fully confused after two rounds. In the second case, when Feistel network structure is used for KeyExpansion, the minimum number of active *F* functions for 32 consecutive rounds is 31, and the maximum related-key differential character is $(2^{-8})^{31} = 2^{-248}$, and 64-bit round key data can be fully confused after four rounds. Outputs are affected by all inputs of full diffusion. Thus, each round key *RK* is different $(RK^0_{(64)} \neq RK^1_{(64)} \neq RK^2_{(64)} \neq ... \neq RK^{31}_{(64)} \neq RK^{32}_{(64)})$. Under the condition of round key, the plaintexts are different from the ciphertexts $(Y = SFN(X, RK) \neq X)$. As a consequence, the results of above two cases constitute the maximum related-key differential character of SFN. We conclude that SFN is secure enough against the related-key attack and does not have a weak key.

4.3. Algebraic attack

Algebraic attack [33] is a powerful approach which can be combined with other attacks to attack ciphers. The attackers usually calculate some multivariate algebraic equations to recover the key. However, these complex multivariate algebraic equations come from the nonlinear S-boxes. Any 4 × 4 S-box can be described by 21 quadratic equations of 8 input/output-bit variables over *GF*(2) [31]. We use two different high-security S-box layers in SFN. If SFN has 32 iterative rounds and consists of $m = (32 \times 32) + (32 \times 16) = 1536$ S-boxes. The round function of the SP network structure uses two S-box layers, so there are $(32 \times 32) = 1024$ S-boxes in the encryption and KeyExpansion. And the round function of Feistel network structure uses

Table 8
The number of S-boxes, quadratic equations and variables in different ciphers.

Algorithm	S-boxes	Quadratic equations	Variables
PRESENT-80	527	11,067	4216
MIBS-80	320	6720	2560
KLEIN-64	240	5040	1920
SFN	1536	32,256	12,288

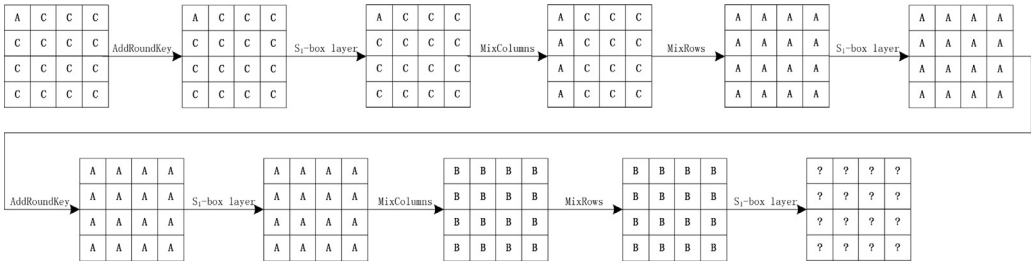


Fig. 8. Two-round integral distinguisher of SFN.

one S-box layer, so there are $(32 \times 16) = 512$ S-boxes in the encryption and the KeyExpansion. Thus, SFN can be described as $32256(=1536 \times 21)$ quadratic equations of $12288(=1536 \times 8)$ variables. However, we have compared SFN with PRESENT, MIBS and KLEIN on the number of S-boxes and quadratic equations of variables. Table 8 shows a comparison among them. In this experiment, SFN has better results than others. Thus, the SFN is proved to resist algebraic attack very well.

4.4. Slide attack

Slide attack [34] takes such advantages of some weaknesses of KeyExpansion as related-key attack does. As we have explained in Section 4.2, the KeyExpansion algorithm structure is in common with encryption and decryption algorithm structure. Moreover, we use the AddConstants in KeyExpansion of SFN, and the AddConstants makes the round function of KeyExpansion adding a round constant described as *CON*; thus, we ensure the operation of each round function is different. We design the KeyExpansion which uses both SP network structure and Feistel network structure to operate, so the KeyExpansion is irregular, which provides the method a good performance against slide attack.

4.5. Meet-in-the-Middle attacks

Meet-in-the-Middle attacks are to increase the memory complexity and to reduce the time complexity. Compared with other attacks, these attacks require less number of plaintext and ciphertext pairs. 32-bit key of SFN is used as the control signal to select different structures (SP or Feistel) to perform encryption and do key expansion. When SP is used to carry out encryption/decryption and KeyExpansion, the round functions for them are different. This also happens when using Feistel structure. Due to the difference of 32-bit control keys, there are 2^{32} different ways of operation. Thus, encrypted and decrypted data cannot be matched in the Meet-in-the-Middle attacks, and attackers are not able to guess partial information of the key. If the control key is fixed to 32-bit 0 or 1, we can analyze the Meet-in-the-Middle attacks for SFN. When the control key is fixed to 32-bit 0, SFN selects the SP network structure to carry out encryption and decryption while the Feistel network structure carries out KeyExpansion. 64-bit encrypted data can be fully diffused after two rounds, and the upper bounded rounds of the partial matching [35] are $3 (= (2 - 1) + (2 - 1) + 1)$. And the 64-bit round key is utilized for each round in the encryption process. In the worst case, there may be $7 (= 2 + 2 + 3)$ rounds of Meet-in-Middle Attacks. When the control key is fixed to 32-bit 1, SFN selects the Feistel network structure to run encryption and decryption while the SP network structure runs KeyExpansion. 64-bit encrypted data can be fully diffused after four rounds, and the upper bounded rounds of the partial matching are $7 (= (4 - 1) + (4 - 1) + 1)$. But the 64-bit data of the round key can only be fully diffused by two rounds. In the worst case here, there may be $15 (= 4 + 4 + 7)$ rounds of Meet-in-Middle Attacks. In practical application of SFN, the control key cannot be fixed to 32-bit 0 or 1. As a result, SFN has high security against Meet-in-the-Middle attacks.

4.6. Integral attack

Integral attack is a chosen plaintext attack which is one of the most aggressive attacks. Integral attack mainly targets on the structure of the encryption algorithm than the algorithm component. For integral attack, we consider the case where the control key is 32-bit 0, and the others are similar. When SFN selects the SP network structure to carry out encryption and decryption, we constitute two-round integral distinguisher of SFN. We define an active 4-bit as *A*, and *C* denotes a constant 4-bit. *B* denotes a balanced 4-bit, and ? denotes an uncertain 4-bit. Assuming that we input 4 × 4 array where the first item is *A* and the others are 15 *C*. In the first round encryption of SFN, the first item is also *A* and the others are 15 *C* after the calculation of AddRoundKey and the S_1-box layer. The first column of the 4×4 array is 4 *A* while the other three columns are 12 *C* after the calculation of MixColumns. Then the 4×4 array becomes 16 *A* being calculated by MixRows. At last, the 4×4 array is also 16 *A* after another S_1-box layer calculation. In the second round encryption of SFN, the 4 × 4 array becomes 16 *A* after the calculation of AddRoundKey and the S_1-box layer. Then the 4×4 array becomes 16 *B* after the calculation of MixColumns and MixRows. At last, the 4×4 array becomes 16. Fig. 8 illustrates two-round integral distinguisher of SFN.

We can attack up to 4 rounds of SFN based on the two-round integral distinguisher with two rounds of encrypted operation. Due to the fourth round of encrypted data being fully diffused, we need to guess 16 4-bit key in the 4 rounds of integral attack. We define the 64-bit key on the fourth round as $K^4_{i(4)}$ $(0 \le i \le 15)$, and $K^4_{i(4)}$ has 16 different values. The probability is about 2^{-4} for a wrong $K^4_{i(4)}$. Because the number of wrong round key is $(2^{64} - 1) \times (2^{-4})^{17} \approx 2^{-4} < 1$, we need to analyze 17 groups of plaintext and ciphertext pairs to find the one correct key. We analyze that the attack time complexity as the following. The first group of plaintext and ciphertext pairs processing time complexity is equivalent to $(2^{64} + 2^8) \div (4 \times 16) \approx 2^{58}$, and there are $(2^{64} \times 2^{-4}) = 2^{60}$ wrong round keys left. The second group of plaintext and ciphertext pairs processing time complexity is equivalent to $(2^{60} + 2^8) \div (4 \times 16) \approx 2^{54}$, and there are $(2^{60} \times 2^{-4}) = 2^{56}$ wrong round keys left. In the same way, the third group to the 16th group of plaintext and ciphertext pairs is analyzed. The 17th group of plaintext and ciphertext pairs processing time complexity is equivalent to2^4.

Therefore, when the control key of SFN is 32-bit 0, the complexity of 4 rounds integral attack can be described as follows. The data complexity is $17 \times 2^{64} \approx 2^{68}$, the time complexity is $2^{58} + 2^{54} + ... + 2^4 \approx 2^{58}$, and the memory space is 2^{60} in order to store the key. Similarly, when the control key of SFN is 32-bit 1, the data and time complexities of 8 rounds integral attack have a similar result with the above. When the control key is not fixed to 32-bit 0 or 1, the full-round of SFN is sufficiently secure against integral attack.

4.7. Impossible differential attack

Impossible differential attack is a variant of differential attacks, which has been employed to analyze many block ciphers and structures. It attacks the weaknesses of the algorithm, such as the poor avalanche

properly. The miss-in-the-middle and the key sieving approaches are required during impossible differential attack. We apply the method of meet-in-the-middle attack for the evaluation of SFN against impossible differential attack. The 32-bit key of SFN is used as the control signal to select different structures (SP or Feistel) to perform encryption and do key expansion. There are 2^{32} different ways of operating the key. This method makes the miss-in-the-middle approach not able to constitute an impossible differential distinguisher. In meet-in-the-middle attack of SFN, we have described the diffusion of the SFN. If the control key is fixed to 32-bit 0, SFN selects the SP network structure to carry out encryption and decryption. SFN has a maximum 7 rounds impossible differential attack. With the fact that the one round diffusion of SFN is faster than AES, these two algorithms have similar round function to encrypt. AES has 7 rounds of impossible differential attack [36]. If the control key is fixed to 32-bit 1, SFN selects the Feistel network structure to carry out encryption and decryption. SFN has a maximum 9 rounds of impossible differential attack. Piccolo has 9 rounds of impossible differential attack [19]. These two algorithms have similar round functions to encrypt. Therefore, we conclude that when the control key is not fixed to 32-bit 0 or 1, the full-round of SFN is sufficiently secure against impossible differential attack.

5. Performance evaluation

5.1. Hardware implementation

We propose that SFN is compact in hardware implementation and is guaranteed to have high security. We have conducted a hardware experiment with SFN and implemented SFN with Verilog-HDL. SFN is simulated with the ModelSim SE PLUS 6.1f Evaluation. We also use a standard cell library based on SMIC 0.18 μm CMOS technology to synthesize SFN by Synopsys Design Compiler version B-2008.09. The circuit area size is measured in Gate Equivalences (GEs), from which a GE is equivalent to the area of a two-input NAND gate.

In Fig. 9, the datapath of SFN is shown when SFN has been implemented in hardware. Our design requires two 64-bit wide registers, a 32-bit wide register, two 1-to-2 DEMUX and two 2-to-1 MUX. These registers are composed of D-flip-flops. A D-flip-flop, a 1-to-2 DEMUX and a 2-to-1 MUX need 4.5 GE, 2.0 GE and 2.0 GE, respectively. The SP network structure consists of key addition, S-box layers, MixColumns and MixRows. The Feistel network structure includes constant addition, right data addition, P layer, MixXors and S-box layer. The 64-bit plaintext inputs are stored in the 64-bit wide register, and the 96-bit primary key has been divided into two parts which are 64-bit *RK*and 32-bit *CK*. The 64-bit *RK* is also stored in the 64-bit wide register, and the 32-bit *CK* is stored in the 32-bit one. In the round constant addition, right data and key additions are installed as bit-wire XORs. The MixXors is installed as 4-bit-wire XORs. Two different 4 × 4 S-boxes layers are calculated in the round function. To preserve the 4 × 4 MDS matrix, the MixClomns and the MixRows require some registers. The P layer is bit wiring.

SFN encrypts a 64-bit plaintext with 96-bit key, and the encryption process takes 32 clock cycles. SFN requires 1876.04 GE in hardware implementation. The simulated power consumption of SFN is 1.97 μW with a frequency of 100 KHz. We compare the implementations of SFN with other lightweight block ciphers in Table 9.

How components of SFN occupy hardware resources is explained in the following. Stored data costs some GEs, and the 64-bit plaintext needs approximately 344 GE while the 96-bit key requires 476 GE. SFN has 6 components. The AddRoundKey has two parts: one is the AddRoundKey of SP network structure, the other is the AddRoundKey of Feistel network structure. The AddRoundKey of SP network structure

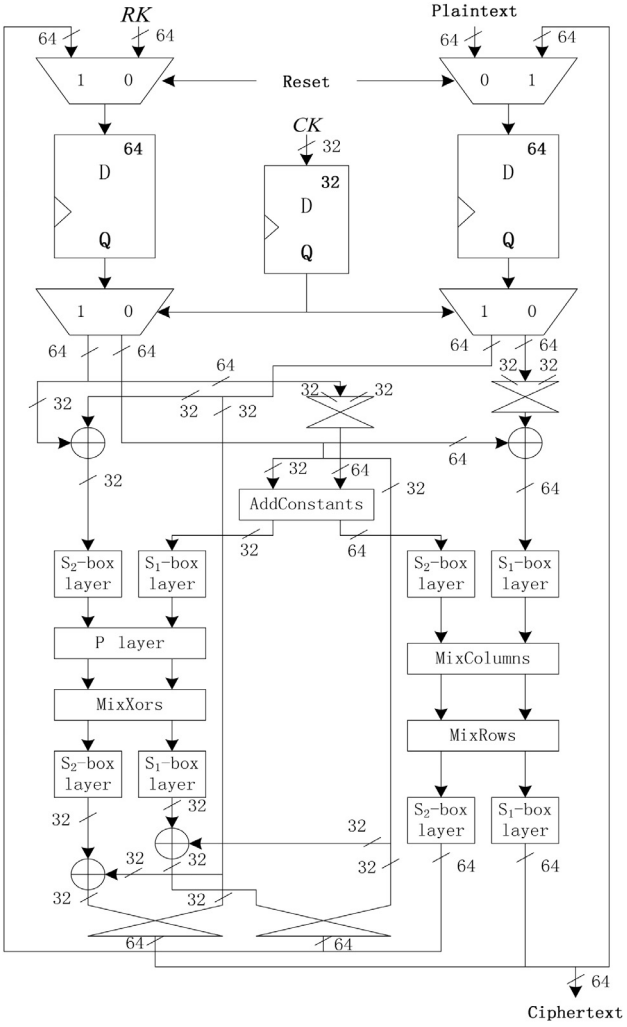


Fig. 9. The datapath of SFN.

runs 64-bit XOR operation, and the AddRoundKey of Feistel network structure runs 32-bit XOR operation. The AddRoundKey requires 169.8 GE. The right data addition also needs 32-bit XOR operation, requiring 56.6 GE. SFN has two different S-box layers where the S_1-box layers and the S_2-box layers cost 212.8 GE and 256 GE, respectively. The MixColumns and the MixRows require 162 GE. The AddConstants consists of an 8-bit Constants XOR operation, costing 16 GE. The MixXors requires 170.84 GE. The P layer is implemented by wiring, without costing any area resource. Furthermore, the control logic and other counters of SFN require 12 GE. Thus, the area of SFN is 1876.04GE. Table 10 summarizes the experiment results of SFN.

5.2. Software implementation

SFN also works efficiently in software implementation. The encryption of SFN is suitable for both hardware and software platforms. When we design SFN, we exclude some components which need to be operated by complex computers. The S-box layers and the P layers can be implemented by lookup tables. When SFN is implemented on 8-bit microcontroller, the encryption and KeyExpansion need use 28 table lookups, and 24 × 32 = 896 8-bit words of memory are required. We also need to analyze the number of clock cycles. The S-box layers

Table 9
Comparison of lightweight block cipher implementations.

Algorithm	Structure	Rounds	Block size (bits)	Key size (bits)	Area (GE)	Speed (kbps@100 KHz)	Logic Process
Piccolo	GFN	25	64	80	1136	237.04	–
PRESENT	SPN	31	64	80	1570	200	0.18 μm
KLEIN	SPN	12	64	64	2032	–	0.18 μm
LBock	Feistel	32	64	80	1320	200	0.18 μm
Twine	GFN	36	64	80	1799	178	0.09 μm
LED	SPN	32	64	128	1265	3.4	0.18 μm
MIBS	Feistel	32	64	64	1396	200	0.18 μm
RECTANGLE	SPN	25	64	80	1599.5	246	0.13 μm
QTL	GFN	16	64	64	1026	200	0.18 μm
mCrypton	SPN	12	64	64	2420	482.3	0.13 μm
Midori	SFN	16	64	64	1542	–	0.09 μm
PRINCE	SPN	10	64	128	3286	529.9	0.09 μm
SIMON	Feistel	44	64	128	1484	133.3	0.13 μm
LILLIPUT	EGFN	30	64	80	1581	213	LP0.065 μm
SFN	SPN, Feistel	32	64	96	1876.04	200	0.18 μm

Table 10
Area requirement of SFN.

Module	GE	%
Data Register	344	18.37
Constants Xor	16	0.85
Key Xor	169.8	9.05
P layer	0	0
Two Data Xor	56.6	3.02
S-box layers	468.8	24.99
MixColumns and MixRows	162	8.64
MixXors	170.84	9.11
Key Register	476	25.37
Control logic and other counters	12	0.64
Total	1876.04	100

require 40 clock cycles, the MixColumns and the MixRows require 40 clock cycles, the MixXors requires 20 clock cycles, however, other linear layers require 48 clock cycles. Altogether, SFN only requires 4736 clock cycles to encrypt a plaintext block.

6. Conclusion

In this paper, we have introduced a different encryption method taking advantages of both SP network structure and Feistel network structure. The 32-bit *CK* controls SP network structure and Feistel network structure for KeyExpansion and encryption or decryption. This method utilizes a network structure to operate not only KeyExpansion but also encryption or decryption. There are $2^{32} = 4{,}294{,}697{,}296$ different ways of operation. Comparing with fixed structure of encrypting or decrypting and operating KeyExpansion, the encryption method improves the immunity with a coefficient of 2^{32} against attacks, such as Differential and linear attacks, Meet-in-the-Middle attack, Impossible Differential attack and Integral attack and so on. Additionally, this novel encryption method satisfies the security requirements of different user levels. Secondly, we have exploit involution related properties of the nonlinear and linear components to design SP network structure. The modified SP network structure makes encryption and decryption program or circuit as the Feistel network structure. Specifically, the round function of modified SP network structure uses some linear layers between two S-box layers. This design has enough differentially and linearly active S-boxes over a certain number of rounds. Thirdly, in the SP network structure, we have employed a MixRows. The confusion effect of SFN is faster than the AES's, and the number of active S-box is also more than that of AES. Because MixRows and MixColumns use the same fixed diffusion matrix, implementation of MixRows spends little hardware and software resources. The round function of SP network structure achieves a high security and a compact implementation in software and hardware.

Finally, we instantiate these three novel ideas into the lightweight block cipher called SFN. The SFN fixes 64-bit block with 96-bit key. We have conducted the security evaluation to SFN. The maximum differential characteristic probability and the maximum linear characteristic probability have achieved 2^{-488}. And SFN can be described with 32,256 quadratic equations of 12,288 variables. The full-round of SFN is sufficiently secure against Meet-in-the-Middle attacks, Integral attack and Impossible Differential attack. These results prove that SFN is more secure than other lightweight block ciphers. We have also run hardware and software experiments on SFN. It requires 1876.04 GE which is less than 2000GE in hardware implementation and only needs 4736 cycles in software implementation. Thus, SFN is not only compact in hardware environments, but also efficient in software platforms. This study has also proved that SFN can be applied to constrained environments. In the end, we strongly encourage researchers to further study the security challenges of SFN.

Acknowledgments

This research is supported by the National Natural Science Foundation of China under grant no.61572174, supported by Hunan Provincial Natural Science Foundation of Chinawith grant no.2017JJ4001, the Scientific Research fund of Hengyang Normal University with grant no 16CXYZ01, the Science and Technology Plan Project of Hunan Province (2016TP1020).

Appendix I. : test vectors

Test vectors of SFN are given such as Table 11. The data are expressed in hexadecimal form.

Table 11
Test vectors for SFN.

Plaintext	Key	Ciphertext
0000-0000-0000-0000	0000-0000-0000-0000-0000-0000	308D-4520-E35A-E7B2
0000-0000-0000-0000	0000-0000-0000-0000-FFFF-FFFF	220A-AED8-D79A-2BA0
0000-0000-0000-0000	FFFF-FFFF-FFFF-FFFF-0000-0000	BB4F-0883-F2D4-2D10
FFFF-FFFF-FFFF-FFFF	0000-0000-0000-0000-FFFF-FFFF	6311-D733-42E7-CABF
6736-05E0-856A-91FB	1695-29AC-FD59-B08B-F85A-2130	CE28-4415-9C1E-E46F

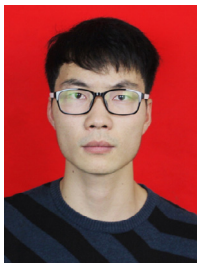
Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.micpro.2018.04.009.

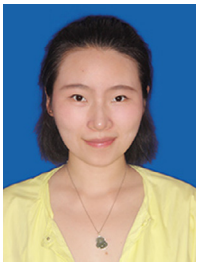
References

[1] L. Li, B.T. Liu, H. Wang, QTL: a new ultra-lightweight block cipher, Microprocess. Microsyst. 45 (2016) 45–55.
[2] T.P. Berger, J. Francq, M. Minier, G. Thomas, Extended generalized Feistel networks using matrix representation to propose a new lightweight block cipher: LILLIPUT, IEEE Trans. Comput. 65 (7) (2016) 2074–2089.
[3] Z. Gong, S. Nikova, Y.W. Law, KLEIN: a new family of lightweight block ciphers, Proceeding of RFIDSec 2011, Springer, 2012, pp. 1–18.
[4] J. Guo, T. Peyrin, A. Poschmann, M. Robshaw, The LED block cipher, Proceeding of Cryptographic Hardware and Embedded Systems-CHES 2011, Springer, 2011, pp. 326–341.
[5] S. Banik, A. Bogdanov, T. Isobe, K. Shibutani, H. Hiwatari, T. Akishita, F. Regazzoni, Midori: a block cipher for low energy, Proceeding of Advances in Cryptology-ASIACRYPT 2015, Springer, 2015, pp. 411–436.
[6] B. Gérard, V. Grosso, M. Naya-Plasencia, F.X. Standaert, Block ciphers that are easier to mask: how far can we go? Proceeding of Cryptographic Hardware and Embedded Systems-CHES 2013, Springer, 2013, pp. 383–399.
[7] C.H. Lim, T. Korkishko, mCrypton – a lightweight block cipher for security of low-cost RFID tags and sensors, Proceeding of WISA 2005, Springer, 2005, pp. 243–258.
[8] A. Bogdanov, L.R. Knudsen, G. Leander, C. Paar, A. Poschmann, M.J.B. Robshaw, Y. Seurin, C. Vikkelsoe, PRESENT: an ultra-lightweight block cipher, Proceeding of Cryptographic Hardware and Embedded Systems-CHES 2007, Springer, 2007, pp. 450–466.
[9] M.R. Albrecht, B. Driessen, E. Kavun, G. Leander, C. Paar, T. Yalçın, Block ciphers—focus on the linear layer (feat. PRIDE), Proceeding of Advances in Cryptology-CRYPTO 2014, Springer, 2014, pp. 57–76.
[10] J. Borghoff, A. Canteaut, T. Güneysu, E.B. Kavun, M. Knezevic, L.R. Knudsen, G. Leander, V. Nikov, C. Paar, C. Rechberger, P. Rombouts, S.S. Thomsen, T. Yalçın, PRINCE- a low-latency block cipher for pervasive computing applications, Proceeding of ASIACRYPT 2012, Springer, 2012, pp. 208–225.
[11] W.T. Zhang, Z.Z. Bao, D.D. Lin, V. Rijmen, B.H. Yang, I. Verbauwhede, RECTANGLE: a bit-slice ultra-lightweight block cipher suitable for multiple platforms, Sci. China Inf. Sci. 58 (12) (2015) 1–15.
[12] A. Poschmann, S. Ling, H. Wang, 256 bit standardized crypto for 650 GE—GOST revisited, Proceeding of Cryptographic Hardware and Embedded Systems-CHES 2010, Springer, 2010, pp. 219–233.
[13] F. Karakoç, H. Demirci, A.E. Harmancı, ITUbee: a software oriented lightweight block cipher, Proceeding of Lightweight Cryptography for Security and Privacy-LightSec2013, Springer, 2013, pp. 16–27.
[14] W.L. Wu, L. Zhang, LBlock: a lightweight block Cipher, Proceeding of Applied Cryptography and Network Security-ACNS 2011, Springer, 2011, pp. 327–344.
[15] A. Baysal, S. Şahin, RoadRunneR: a small and fast bitslice block cipher for low cost 8-bit processors, Proceeding of Lightweight Cryptography for Security and Privacy-LightSec2015, Springer, 2015, pp. 58–76.
[16] F.X. Standaert, G. Piret, N. Gershenfeld, J.J. Quisquater, SEA: a scalable encryption algorithm for small embedded applications, Proceeding of CARDIS 2006, Springer, 2006, pp. 222–236.
[17] G.Q. Yang, B. Zhu, V. Suder, M.D. Aagaard, G. Gong, The Simeck family of lightweight block ciphers, Proceeding of Cryptographic Hardware and Embedded Systems-CHES 2015, Springer, 2015, pp. 307–329.
[18] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, L. Wingers, The SIMON and SPECK families of lightweight block ciphers, 2013. http://eprint.iacr.org/2013/404.pdf>.
[19] K. Shibutani, T. Isobe, H. Hiwatari, A. Mitsuda, T. Akishita, T. Shirai, Piccolo: an ultra-lightweight blockcipher, Proceeding of Cryptographic Hardware and Embedded Systems-CHES 2011, Springer, 2011, pp. 342–357.
[20] T. Suzaki, K. Minematsu, S. Morioka, E. Kobayashi, TWINE: a lightweight, versatile block cipher, Proceeding of ECRYPT Workshop on Lightweight Cryptography 2011, 2011, pp. 146–169.
[21] M. Izadi, B. Sadeghiyan, S.S. Sadeghian, H.A. Khanooki, MIBS: a new lightweight block cipher, Proceeding of Cryptography and Network Security-CANS 2009, Springer, 2009, pp. 334–348.
[22] V. Lallemand, M. Naya-Plasencia, Cryptanalysis of KLEIN, Proceeding of Fast Software Encryption-FSE 2014, Springer, 2014, pp. 451–470.
[23] J. Guo, I. Nikolic, T. Peyrin, L. Wang, Cryptanalysis of zorro, 2013, http://eprint.iacr.org/2013/713.pdf>.
[24] Y.F. Wang, W.L. Wu, X.L. Yu, Biclique cryptanalysis of reduced-round piccolo block cipher, Proceeding of Information Security Practice and Experience-ISPEC 2012, Springer, 2012, pp. 337–352.
[25] M. Çoban, F. Karakoç, Ö. Boztaş, Biclique cryptanalysis of TWINE, Proceeding of Cryptology and Network Security-CANS 2012, Springer, 2012, pp. 43–55.
[26] A. Canteaut, T. Fuhr, H. Gilbert, M. Naya-Plasencia, J.R. Reinhard, Multiple differential cryptanalysis of round-reduced PRINCE, Proceeding of Fast Software Encryption-FSE 2014, Springer, 2014, pp. 591–610.
[27] B. Bilgin, A. Bogdanov, M. Knežević, F. Mendel, Q. Wang, Fides: Lightweight authenticated cipher with side-channel resistance for constrained hardware, Proceeding of Cryptographic Hardware and Embedded Systems-CHES 2013, Springer, 2013, pp. 142–158.
[28] E. Bilam, A. Shamir, Differential cryptanalysis of the full 16-round DES, Proceeding of Advances in Cryptology-CRYPTO 1992, Springer, 1993, pp. 487–496.
[29] M. Matsui, Linear cryptanalysis method for DES cipher, Proceeding of Advances in Cryptology – EUROCRYPT 1993, Springer, 1994, pp. 386–397.
[30] K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, T. Tokita, Camellia: a 128-bit block cipher suitable for multiple platforms —design and analysis, Proceeding of Selected Areas in Cryptography-SAC 2000, Springer, 2000, pp. 39–56.
[31] M. Matsui, On correlation between the order of S-boxes and the strength of DES, Proceeding of Advances in Cryptology-EUROCRYPT 1994, Springer, 1994, pp. 366–375.
[32] E. Biham, New types of cryptoanalytic attacks using related keys (extended abstract), Proceeding of Advances in Cryptology-EUROCRYPT 1993, Springer, 1993, pp. 398–409.
[33] N.T. Courtois, J. Pieprzyk, Cryptanalysis of block ciphers with overdefined systems of equations, Proceeding of Advances in Cryptology-ASIACRYPT 2002, Springer, 2002, pp. 267–287.
[34] A. Biryukov, D. Wagner, Slide attacks, Proceeding of Fast Software Encryption-FSE 1999, Springer, 1999, pp. 245–259.
[35] K. Aoki, Y. Sasaki, Preimage attacks on one-block MD4, 63-step MD5 and more, Proceeding of Selected Areas in Cryptography-SAC 2008, Springer, 2008, pp. 103–119.
[36] W.T. Zhang, W.L. Wu, D.G. Feng, New results on impossible differential cryptanalysis of reduced AES, Proceeding of International Conference on Information Security and Cryptology-ICISC 2007, Springer, 2007, pp. 239–250.

Lang Li received his Ph.D. and Master's degrees in computer science from Hunan University, Changsha, China, in 2010 and 2006, respectively, and earned his B.S. degree in circuits and systems from Hunan Normal University in 1996. Since 2011, he has been working as a professor in the College of Computer Science and Technology at the Hengyang Normal University, Hengyang, China. He has research interests in embedded system and information security.



Botao Liu received his B.S. degree in Hengyang Normal University, Hengyang, China, in 2016. He has been working towards his Master's degree in Guizhou University, Guizhou, China, since 2016. His main research interest lies in embedded system and information security.



Yimeng Zhou received her Master of Science degree in international hospitality and tourism management from the University of South Carolina, Columbia, South Carolina, USA in May 2014. Her industry experience included working with the team of information system security and integrated information technology for Marriott Vacations Worldwide and Marriott International from June, 2014 to February, 2016. She has joined Hengyang Normal University since March 2016. Her research interests lie in the application of integrated information technology and information system security in hospitality and tourism management, tourism sustainability, etc.

Yi Zou received her Master's degree in computer science from Hunan University, Changsha, China, in 2008. Since 2010, she has been working as a lecturer in the College of Computer Science and Technology at Hengyang Normal University, Hengyang, China. Her main research interest lies in embedded system and information security.