IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 25, 2021, accepted February 24, 2021, date of publication February 26, 2021, date of current version March 11, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3062665

General Differential Fault Attack on PRESENT and GIFT Cipher With Nibble

HAOXIANG LUO[1], (Member, IEEE), WEIJIAN CHEN[2], XINYUE MING[1], AND YIFAN WU[1]

[1]Glasgow College, University of Electronic Science and Technology of China, Chengdu 611731, China
[2]School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China

Corresponding author: Weijian Chen (chenweijian@uestc.edu.cn)

This work was supported in part by the Student Science and Innovation Program of the Glasgow College, UESTC, under Grant 2020001, in part by the Innovation and Entrepreneurship Dean Fund of UESTC under Grant 2019007, and in part by the Course Reform Project of Teaching and Assessment Method of UESTC under Grant XJ202075-1.

ABSTRACT Lightweight block cipher PRESENT is an algorithm with SPN structure. Due to its excellent hardware performance and simple round function design, it can be well applied to Internet of things terminals with limited computing resources. As an improved cipher of PRESENT, GIFT is similar in structure to PRESENT and has been widely concerned by academia and industry. This article studies the P permutation law of PRESENT and GIFT, and presents a general differential fault attack(DFA) method with their differential characteristics. For PRESENT, this article chooses to inject a nibble fault before the 30^{th} and 31^{st} rounds of S-box operation. A total of 32 nibble fault ciphertexts are needed to recover the original key. The computational complexity and data complexity are $2^{10.94}$ and 2^{8}, respectively. For GIFT, this article chooses to inject a nibble fault before the 25^{th}, 26^{th}, 27^{th} and 28^{th} rounds of S-box operation. A total of 64 nibble fault ciphertexts are needed to recover the original key. The computational complexity and data complexity are $2^{11.91}$ and 2^{9}, respectively. Compared with other public cryptoanalysis results of PRESENT and GIFT, this general attack method has great advantages. In this article, the DFA of GIFT is experimentally verified and the effectiveness is proved. These experiments have been done on a personal computer and run in a very reasonable time(around 500ms).

INDEX TERMS Internet of Things, lightweight block cipher, PRESENT, GIFT, differential fault attack.

I. INTRODUCTION

DFA [1] is a new cryptanalysis method proposed by E. Beniham and A. Hamir based on a combination of mathematical and physical methods in 1997. This method has been applied to many block ciphers, like FOX [2], SMS4 [3], AES [4], LED [5], SIMON [6] etc. Meanwhile, with the development of the Internet of things(IoT), a large number of various lightweight block ciphers have emerged, and have proved its efficiency in resource-constrained environments such as RFID tags and wireless sensor networks(WSN). Many scholars have also conducted DFA on LEA [7], PRINCE [8], TWINE [9] and other ciphers.

PRESENT [10] is a lightweight cryptography algorithm put forward by A. Bogdanov *et al.* at the CHES 2007 conference in 2007. PRESENT uses SPN structure, which has 80-bit or 128-bit key with 64 bits block length. Since then, the security of PRESENT have been analyzed by many scholars. Wang [11] published a differential attack on PRESENT in 2008 with both computational and data complexity of 2^{64}. Subsequently, in CT-RSA 2009, Collard and Standaert [12] published an Saturation cryptanalysis of PRESENT, with a computational complexity of 2^{20} and a data complexity of 2^{36}. In FSE 2012, Wang *et al.* [13] proposed the Structure attack method. Reference [14] illustrates the Biclique cryptanalysis results of PRESENT by Zheng Gong *et al.*, and the computational complexity and data complexity are $2^{78.9}$ and 2^{64}, respectively [15] illustrates the impossible differential attack on PRESENT by Tezcan. C, with the computational complexity and data complexity of $2^{62.62}$ and $2^{63.86}$, respectively. Reference [16] illustrates that Ya Tian *et al.* analyzed PRESENT by using the methods of multi-differential-input, single-differential-output and single-differential-input, multi-differential-output. Reference [17] illustrates that Liu *et al.* adopted the linear cryptanalysis method in 2016, and the computational complexity and data complexity were 2^{36} and 2^{32}, respectively. In 2017, Liu *et al.* [17] published the multi-differential cryptanalysis for PRESENT, with the

The associate editor coordinating the review of this manuscript and approving it for publication was Zhitao Guan.

computational complexity and data complexity of $2^{57.6}$ and 2^{61}, respectively. In 2019, Chen *et al.* [19] carried out a single-byte DFA on PRESENT, with a computational complexity of 2^{31}. In addition, [20] discussed the hardware performance of PRESENT, and Petr Moucha *et al.* proposed a dummy rounds schemes as a DPA countermeasure in PRESENT [21].

In order to avoid the design defects of PRESENT mentioned above, on the occasion of the 10th anniversary of PRESENT, Banik *et al.* [22] proposed a new lightweight block cipher named GIFT in CHES 2017. GIFT has a key length of 128 bits and is divided into 64 bits or 128 bits according to the block length with the number of encryption rounds of 28 and 40 respectively. GIFT, an improved cipher of PRESENT, has a similar structure of PRESENT and can save a lot of computational resources and improve computing speed. In this article, the author analyzed the resistance effect of GIFT under such attack methods as differential cryptanalysis, linear cryptanalysis, invariant subspace attack and algebraic attack. In addition, GIFT also improves the P permutation layer diffusivity of PRESENT to avoid the DFA in [19]. Currently, there are few cryptanalyses of GIFT in the public literature, mainly including Biclique cryptanalysis [23], [24], differential cryptanalysis [25], [26], side-channel fault attack [27], [28] and so on. In addition, Jati *et al.* [29] analyzed the threshold implementation of GIFT, and Dalmasso *et al.* [30] analyzed the hardware implementation in FPGA.

A. OUR CONTRIBUTION

We proposed a general DFA method for PRESENT and GIFT by analyzing the displacement law of P permutation layer and S-box differential property. Firstly, the differential properties of the two ciphers were analyzed, and the number of differential function solutions was counted by the differential distribution table of the S-boxes. Secondly, we found that when bits are shifted in the P permutation layer, four bits of each nibble will spread into four different nibbles based on the knowledge of PRESENT and GIFT. Then, we found the appropriate fault injection location according to the obtained diffusion law, and then recovered the key by combining the differential property. Furthermore, we calculated the computational complexity and data complexity of this attack method. For PRESENT, the computational and data complexity are $2^{10.94}$ and 2^8, respectively. For Gift, computational and data complexity are $2^{11.91}$ and 2^9, respectively. These results are superior to the existing public literature. Finally, we experimented this attack method on GIFT for 30 times, and it took an average of 87 nibbles faults to recover all the original keys with 500ms attack time.

This attack method mainly studies the PRESENT with a key of 80 bits (PRESENT means PRESENT-80 unless otherwise specified below) and the GIFT with a block length of 64 bits (GIFT means GIFT-64 unless otherwise specified below). It also proposes the optimized DFA. In this article, the remaining organization structure is as follows: Section II makes a brief introduction to the PRSENT and GIFT, and Section III analyzes their S-box differential and P displacement law. Section IV is the DFA principle and steps of PRESENT and GIFT, and Section V part analyses the computational complexity and data complexity of the attack method on these two ciphers. Section VI is the experimental results of DFA on GIFT. At last, Section VII will be a conclusion.

II. PRESENT AND GIFT

A. SYMBOLS AND TERMINOLOGY

In order to better illustrate the encryption and attack process of the two ciphers, the common symbols used in the analysis of the two are defined as follows:

B_i: 64-bit input of the i^{th} round;

x_j^i: 4-bit input value of S-box in position j of the i^{th} round;

Δx_j^i: 4-bit input differential value of S-box in position j of the i^{th} round;

ΔX_i: 64-bit input differential value of S-box of the i^{th} round;

Δy_j^i: 4-bit output differential value of S-box in position j of the i^{th} round;

ΔY_i: 64-bit output differential value of the S-box of the i^{th} round;

K_i: 64-bit round key involved round function calculation of the i^{th} round;

$R_{k_j}^i$: 4-bit round key in position j of the i^{th} round;

C_i: 64-bit correct ciphertext of the i^{th} round;

C_i^*: 64-bit incorrect ciphertext of the i^{th} round;

ΔC_i^*: 64-bit differential value between the correct ciphertext and the incorrect ciphertext;

$c_{j,i}$: 4-bit correct ciphertext in position j of the i^{th} round;

$c_{j,i}^*$: 4-bit incorrect ciphertext in position j of the i^{th} round;

$\Delta c_{j,i}^*$: 4-bit differential value between the 4-bit correct ciphertext and the 4-bit incorrect ciphertext in position j of the i^{th} round;

$S(\cdot)$: S-box operator;

$S^{-1}(\cdot)$: S-box inverse operator;

$P(\cdot)$: operator of P permutation layer;

$P^{-1}(\cdot)$: inverse operator of P permutation layer;

$a \leftarrow b$: value of b is assigned to a;

$a||b$: the cascade of data a and data b;

$a \oplus b$: data a and data b are xor by bit;

$\ll$<: cyclic shift to the left;

$\gg$>: cyclic shift to the right;

RC^i: counter number of the i^{th} round;

B. ENCRYPTION PROCESSES OF PRESENT AND GIFT

PRESENT and GIFT are both block ciphers based on SPN structure, where PRESENT iterates 31 rounds in the whole encryption process while GIFT only iterates 28 rounds. Take the flow chart of PRESENT encryption as an example, as shown in Fig. 1.

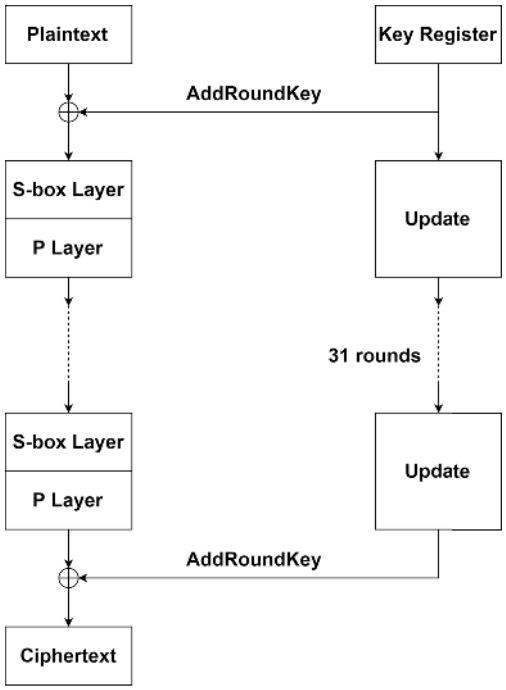

FIGURE 1. Flow chart of PRESENT encryption.

1) PRESENT ENCRYPTION PROCESS

For PRESENT, the encryption process can be divided into three parts: the round key or layer, the nonlinear S-box substitution layer and the linear P permutation layer.

(1) Round key xor layer: the input of the i^{th} round $B_i = b_{63}^i b_{62}^i \cdots b_2^i b_1^i b_0^i$ and the 64-bit round key $K_i = k_{63}^i k_{62}^i \cdots k_2^i k_1^i k_0^i$ of the i^{th} round carries out xor operation, and the output is B_i'.

$$B_i' = B_i \oplus K_i, \quad (0 \le i \le 31) \tag{1}$$

(2) Nonlinear S-box substitution layer: divide the above 64-bit output B_i' into 16 4-bit nibbles represented by $W_{15}W_{14}\cdots W_0$, where $W_j = b_{4j+3}||b_{4j+2}||b_{4j+1}||b_{4j}$, $(0 \le j \le 15)$. PRESENT requires 16 identical 4-bit S-boxes with 4-bit inputs and 4-bit outputs. And W_j were substituted with the 16 S-boxes respectively, to obtain $S(W_j)$. The calculation results of S-box can be obtained from Table 1:

TABLE 1. S-box substitution of PRESENT.

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

(3) Linear P permutation layer: after obtaining the S-box substitution value, each bit is linearly rearranged according to the P permutation table to obtain $P(\cdot)$. The permutation result of the P permutation layer can be obtained the Table 2 below.

2) GIFT ENCRYPTION PROCESS

For GIFT, the encryption process can also be divided into three layers: the nonlinear S-box substitution layer, the linear P permutation layer, the key and constant xor layer. The S-box substitution layer and P permutation layer of GIFT are similar to the PRESENT transformation rule. Except for the transformation value, the S-box substitution table and P layer replacement table of GIFT are given, as shown in Table 3 and Table 4 respectively:

For the key and constant xor layer of GIFT, it contains two parts which are round key and round constant. Round key xor means that 32 bits are selected from the 128-bit key set as the round key for xor operation, and the extracted key is divided into two parts, which are expressed as:

$$K_i = U_i||V_i = u_{15}^i \cdots u_0^i||v_{15}^i \cdots v_0^i \tag{2}$$

Perform xor operation between U_i and V_i on the output of P permutation layer respectively, namely:

$$b_{4j+1}^i \leftarrow b_{4j+1}^i \oplus u_r^i, \quad b_{4j}^i \leftarrow b_{4j}^i \oplus v_r^i, \ 0 \le r \le 15 \tag{3}$$

The cyclic constant xor operation refers to the xor bit 63, bit 23, bit 19, bit 15, bit 11, bit 7 and bit 3 from the P permutation layer between a bit value "1" and a length of 6 bits round constant $L_i = l_5^i l_4^i l_3^i l_2^i l_1^i l_0^i$, namely:

$$\begin{aligned} b_{63}^i &\leftarrow b_{63}^i \oplus 1, \quad b_{23}^i \leftarrow b_{23}^i \oplus l_5^i \\ b_{19}^i &\leftarrow b_{19}^i \oplus l_4^i, \quad b_{15}^i \leftarrow b_{15}^i \oplus l_3^i \\ b_{11}^i &\leftarrow b_{11}^i \oplus l_2^i, \quad b_7^i \leftarrow b_7^i \oplus l_1^i \\ b_3^i &\leftarrow b_3^i \oplus l_0^i \end{aligned} \tag{4}$$

C. ROUND KEY UPDATE SCHEME

1) ROUND KEY UPDATE SCHEME OF PRESENT

The key expansion method of PRESENT includes cyclic shift and S-box substitution. Firstly, the initial key $K_0 = k_{79}^0 k_{78}^0 \cdots k_2^0 k_1^0 k_0^0$ is saved in the shift register. The round key of the i^{th} round is to take the left 64bit in the current register:

$$K_i = k_{63}^i k_{62}^i \cdots k_2^i k_1^i k_0^i = k_{79}^i k_{78}^i \cdots k_{21}^i k_{20}^i k_{19}^i \tag{5}$$

The specific steps are as follows:

(1) First, loop the key to the left 61 bits in the register:

$$K \ll< 61 \tag{6}$$

(2) Then, the left 4-bit value was used for S-box substitution operation:

$$K[79-76] \leftarrow S(K[79-76]) \tag{7}$$

(3) Finally, operate the bitwise xor on$k_{19}k_{18}k_{17}k_{16}k_{15}$ in the round key and the counter number:

$$K[19-15] \leftarrow K[19-15] \oplus RC^i \tag{8}$$

TABLE 2. P permutation layer of PRESENT.

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

TABLE 3. S-box substitution of GIFT.

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

2) ROUND KEY UPDATE SCHEME OF GIFT

a: ROUND KEY

The round key extraction rule of GIFT is extracted first and then updated. The updating method is as follows:

$$t_7||t_6||\cdots||t_1||t_0 \leftarrow t_1 \ggg 2||t_0 \ggg 12||\cdots||t_3||t_2 \quad (9)$$

where t is a 16-bit word.

b: ROUND CONSTANT

The initial value of the 6-bit round constant is 0, which needs to be updated through a 6-bit shift register. The update function is as follows:

$$(l_5, l_4, l_3, l_2, l_1, l_0) \leftarrow (l_4, l_3, l_2, l_1, l_0, l_5 \oplus l_4 \oplus l_1) \quad (10)$$

Round constants of different rounds are in the Table 5.

III. STRUCTRAL PROPERTIES OF PRESENT AND GIFT

A. STRUCTRAL PROPERTIES OF PRESENT

1) S-BOX DIFFERENTIAL PROPERTY

If there are$\Delta\alpha = \Gamma_2^4$, $\Delta\beta = \Gamma_2^4$, $m = \Gamma_2^4$. $\Delta\alpha$ is the differential input of S-box, and $\Delta\beta$ is the differential output of S-box, which satisfy: $S(m \oplus \Delta\alpha) \oplus S(m) = \Delta\beta$. Through calculation, the S-box differential property of PRESNET is obtained as shown in Table 6. $Num(\Delta\alpha, \Delta\beta)$ is the number of m that satisfy the equation $S(m \oplus \Delta\alpha) \oplus S(m) = \Delta\beta$.

When $Num(\Delta\alpha, \Delta\beta) = 0$ is satisfied, $S(m \oplus \Delta\alpha) \oplus S(m) = \Delta\beta$ has no solution. As can be seen from Table 6, of having no solution is 62.1%. Similarly, the probability of the equation having two solutions is 28.2%, the probability of the equation having four solutions is 9.3%, and the probability of the equation having 16 solutions is 0.4%. If we only consider the case where the equation has solutions, namely $Num(\Delta\alpha, \Delta\beta) \neq 0$, we can get that 74.2% of the probability equation has two solutions, 24.1% of the probability equation has four solutions, and 1.1% of the probability equation has 16 solutions. Therefore, the average number of solutions can be calculated to be $2.648 (= 1.484 + 0.998 + 0.276)$.

2) P PERMUTATION LAYER PROPERTY

Every four consecutive nibbles are divided into a group(from left to right are Group1, Group 2, Group 3 and Group 4), then each nibble will spread to four different nibbles after P permutation layer. In addition, 4 nibbles from the same group will be diffused to the same four nibbles, and nibbles diffused by different groups will not cross and repeat, so the position of nibble fault can be determined by positions of the diffusion. The specific diffusion locations for each group of nibbles are shown in Table 7:

According to the characteristics of four nibbles diffusion positions in each group, the import position of the fault can be determined. Therefore, this article proposes a method of nibble DFA on PRESENT in Chapter IV.

B. SRUCTUAL PROPERTIES OF GIFT

1) S-BOX DIFFERENTIAL PROPERTY

In the same way that PRESENT differential properties were analyzed, GIFT's S-box differential distribution law can be obtained, as shown in Table 8.

Similarly, when $Num(\Delta\alpha, \Delta\beta) = 0$ is satisfied, $S(m \oplus \Delta\alpha) \oplus S(m) = \Delta\beta$ has no solution. As can be seen from Table 8, of having no solution is 61.3%. And then, the probability of the equation having two solutions is 30.5%, the probability of the equation having four solutions is 3.1%, the probability of the equation having four solutions is 0.8%, and the probability of the equation having 16 solutions is 0.4%. If we only consider the case where the equation has solutions, namely $Num(\Delta\alpha, \Delta\beta) \neq 0$, we can get that 78.8% of the probability equation has two solutions, 18.2% of the probability equation has four solutions, 2.0% of the probability equation has four solutions, and 1.0% of the probability equation has 16 solutions. Therefore, the average number of solutions can be calculated to be $2.584 (= 1.576 + 0.728 + 0.120 + 0.160)$.

TABLE 4. P permutation layer of GIFT.

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

TABLE 5. Round constants of GIFT.

Number of rounds	Round constant						
1-7	01	03	07	0F	1F	3E	3D
8-14	3B	37	2F	1E	3C	39	33
15-21	27	0E	1D	3A	35	2B	16
22-28	2C	18	30	21	02	05	0B

2) *P* PERMUTATION LAYER PROPERTY

According to Table 4, the displacement law of P permutation layer of GIFT is analyzed. The specific diffusion locations for each group of nibbles are shown in Table 9:

According to the characteristics of four nibbles diffusion positions in each group, the import position of the fault can be determined. Therefore, this article proposes a method of nibble DFA on GIFT in Chapter IV.

IV. NIBBLE DIFFERENTIAL FAUAUL ATTACKS ON PRESENT AND GIFT

A. ATTACK BASIC ASSUMPTION

In order to make a better analysis, two basic assumptions are put forward before introducing concept of attack:

(1) The fault can be imported into nibble of any round with the unknown fault value and position, then the correct ciphertext $C_i$and error ciphertext C_i^* can be obtained.

(2) For same plaintext and round key, random nibble faults can be induced at same times and locations of iterations, then the corresponding error ciphertext C_i^*can be acquired.

B. DFA ON PRESENT

1) ATTACK MODEL AND PRINCIPLE OF PRESENT

Because PRESENT's S-box is a nonlinear substitution of 4-bit input and output, this article chooses to import a nibble fault to construct attack model. Depending on the attack hypothesis and the actual situation, we usually only get the last round of ciphertext. Therefore, we first chose to import random failure in the 31^{th} round before the S-box operation.

After randomly choosing a set of plaintext and key and get the correct ciphertext by encryption. The fault is imported in the any nibble group of the 31^{th} round, and some features of ciphertext are obtained by combining the characteristics of DFA. Furthermore, through the reverse process of the encryption, the partial round key is deduced. By repeating the process several times with the same ciphertext, the partial key will be recovered in a unique value. And then, attack the other three groups of nibbles of the 31^{th} round in the same way until the 64-bit round key is recovered. We can reverse the ciphertext of the 30^{th} round according to the key of 31^{th} round, and then use the same attack method to recover the key of the 30^{th} round. According to the PRESENT key update scheme, the complete 80-bit original key can be derived by two consecutive round keys.

2) SPECIFIC STEPS OF ATTACK

(1) Generate plaintext B and a round key K randomly, and the correct ciphertext C_{31} is obtained after 31 rounds of encryption by PRESENT.

(2) Use the original plaintext B and key K for encryption. In the 31^{th} round of encryption, a random nibble fault is induced in the S-box of PRESENT. The wrong ciphertext C_{31}^* is obtained and the differential operation with the correct ciphertext in the 31^{st} round is performed to obtain the differential error ciphertext:

$$\Delta C_{31}^* = C_{31} \oplus C_{31}^* \tag{11}$$

Import a fault in the first nibble on the left side x_{31}^{15} of Group 1 as an example, and output the differential result in the form of ($\Delta c_{15,31}^*$, 0, 0, 0, $\Delta c_{11,31}^*$, 0, 0, 0, $\Delta c_{7,31}^*$, 0, 0, 0, $\Delta c_{3,31}^*$, 0, 0, 0).

(3) Through the correct ciphertext and the wrong ciphertext, find the fault S-box, and differential output:

$$\Delta Y_{31} = P^{-1}(\Delta C_{31}^*) \tag{12}$$

(4) Because it is a fault attack on the input of the first S-box, ΔY_{31} has only one non-zero nibble Δy_{31}^{15}. List the S-box differential equation:

$$\Delta y_{31}^{15} = S(x_{31}^{15}) \oplus S(x_{31}^{15} \oplus \Delta x_{31}^{15}) \tag{13}$$

Δx_{31}^{15} has $2^4 = 16$ possible values, and then 16 values are exhausted and solve the values of x_{31}^{15}, which satisfy (13). Furthermore, store the calculated x_{31}^{15} in a set M_1.

TABLE 6. S-box differential distribution property of PRESENT.

$Num(\triangle\alpha, \triangle\beta)$	Times of occurrences	Probability of occurrence	Probability of $Num \neq 0$	Mean value
0	159	0.621		
2	72	0.282	0.742	1.484
4	24	0.093	0.241	0.998
16	1	0.004	0.011	0.276

TABLE 7. P layer law of PRESENT.

	Group 1	Group 2
Diffusion location	$x_{15}^i, x_{11}^i, x_7^i, x_3^i$	$x_{14}^i, x_{10}^i, x_6^i, x_2^i$
	Group 3	Group 4
Diffusion location	$x_{13}^i, x_9^i, x_5^i, x_1^i$	$x_{12}^i, x_8^i, x_4^i, x_0^i$

(5) According to the analysis of the S-box property of PRESENT(Table 6), there may be more than one matching result, so attack steps (2)∼(4) will be continuously repeated until set M_1 to reserve only one nibble, which is the correct x_{31}^{15}.

(6) Repeat (2)∼(5), and attack the other three nibbles in Group 1 in the same way to get other 4-bit input differential value of S-box x_{31}^{11}, x_{31}^{7} and x_{31}^{3}.

(7) Repeat (2)∼(6), and attack other nibbles in Group 2, Group 3 and Group 4 in turn to until the 64-bit input differential value of S-box ΔX_{31} is recovered.

(8) Through the following formula, 64-bit round key K_{31} is obtained.

$$K_{31} = P(S(\Delta X_{31})) \oplus C_{31} \tag{14}$$

(9) Similarly, DFA on PRESENT in the 30th round, and steps (2)∼(8) are repeated to recover the key K_{30}.

(10) Based on the recovered two successive rounds of key K_{30} and K_{31}, the PRESENT original 80-bit key is rolled out.

C. DFA ON GIFT

1) ATTACK MODEL AND PRINCIPLE OF GIFT

Because GIFT's S-box is a nonlinear substitution of 4-bit input and output, this article chooses to import a nibble fault to construct attack model. Similar to PRESENT, we first chose to import random failure in the 28th round before the S-box operation.

In the same way, the failure is imported at the 28th round of the GIFT encryption process and the 32-bit round key is recovered using the same method as PRESENT. According to the GIFT key update scheme, a complete 128-bit original key can be derived by four consecutive round keys. Therefore, the 128-bit original key of GIFT is derived by using the same steps to obtain their round keys for the 25th, 26th and 27th rounds respectively.

2) SPECIFIC STEPS OF ATTACK

(1) Generate plaintext B and a round key K randomly, and the correct ciphertext C_{28} is obtained after 28 rounds of encryption by GIFT.

(2) Use the original plaintext B and key K for encryption. In the 28th round of encryption, a random nibble fault is induced in the S-box of GIFT. The wrong ciphertext C_{28}^* is obtained and the differential operation with the correct ciphertext in the 28th round is performed to obtain the differential error ciphertext:

$$\Delta C_{28}^* = C_{28} \oplus C_{28}^* \tag{15}$$

Import a fault in the first nibble on the left side x_{28}^{15} of Group 1 as an example, and output the differential result in the form of ($\Delta c_{15,28}^*$, 0, 0, 0, $\Delta c_{11,28}^*$, 0, 0, 0, $\Delta c_{7,28}^*$, 0, 0, 0, $\Delta c_{3,28}^*$, 0, 0, 0).

(3) Through the correct ciphertext and the wrong ciphertext, find the fault S-box, and differential output:

$$\Delta Y_{28} = P^{-1}(\Delta C_{28}^*) \tag{16}$$

(4) Because it is a fault attack on the input of the first S-box, ΔY_{28} has only one non-zero nibble Δy_{28}^{15}. List the S-box differential equation:

$$\Delta y_{28}^{15} = S(x_{28}^{15}) \oplus S(x_{28}^{15} \oplus \Delta x_{28}^{15}) \tag{17}$$

Δx_{28}^{15} has $2^4 = 16$ possible values, and then 16 values are exhausted and solve the values of x_{28}^{15}, which satisfy (17). Furthermore, store the calculated x_{28}^{15} in a set M_2.

(5) According to the analysis of the S-box property of GIFT(Table 8), there may be more than one matching result, so attack steps (2)∼(4) will be continuously repeated until set M_2 to reserve only one nibble, which is the correct x_{28}^{15}.

(6) Repeat (2)∼(5), and attack the other three nibbles in Group 1 in the same way to get other 4-bit input differential value of S-box x_{28}^{14}, x_{28}^{13} and x_{28}^{12}.

(7) Repeat (2)∼(6), and attack other nibbles in Group 2, Group 3 and Group 4 in turn to until the 64-bit input differential value of S-box ΔX_{28} is recovered.

(8) Through the following formula, round key K_{28} is obtained.

$$K_{28} = P(S(\Delta X_{28})) \oplus C_{28} \tag{18}$$

(9) Similarly, DFA on GIFT in the 27th, 26th and 25th rounds, and steps (2)∼(8) are repeated to recover the keysK_{27}, $K_{26}$and K_{25}.

TABLE 8. S-box differential distribution property of GIFT.

$Num(\triangle\alpha, \triangle\beta)$	Times of occurrences	Probability of occurrence	Probability of $Num \neq 0$	Mean value
0	157	0.613		
2	78	0.305	0.788	1.576
4	18	0.031	0.182	0.728
6	2	0.008	0.020	0.120
16	1	0.004	0.010	0.160

TABLE 9. P layer law of GIFT.

	Group 1	Group 2
Diffusion location	$x^i_{15}, x^i_{11}, x^i_7, x^i_3$	$x^i_{14}, x^i_{10}, x^i_6, x^i_2$
	Group 3	Group 4
Diffusion location	$x^i_{13}, x^i_9, x^i_5, x^i_1$	$x^i_{12}, x^i_8, x^i_4, x^i_0$

(10) Based on the recovered four successive rounds of key K_{25}, K_{26}, K_{27} and K_{28}, the GIFT original 128-bit key is rolled out.

V. COMPLEXITY ANALYSIS

A. COMPLEXITY ANALYSIS OF PRENSENT

The attack mode established based on the fault propagation property is optimized compared with the traditional exhaustive search method of PRESENT. The complexity of recovering the round key is 2^{64} in the traditional exhaustive search method. In this article, according to the propagation property of the fault, possible values of the non-zero nibble position of the differential output are exhausted, which can effectively reduce the complexity required by the attack.

A nibble fault can recover the 4-bit Δx^{15}_{31}, so the minimum number of faults required to recover the 64-bit round key of PRESENT is calculated as follows:

$$\begin{cases} 0, & d = 0 \\ \left\lceil \frac{m}{d} \right\rceil, & 1 \le d \le m \end{cases} \tag{19}$$

In (4), m is the number of round-key bits; d represents the number of round key bits corresponding to a fault. If $d = 0$, the attack does not recover any bits of the round key. For the PRESENT and the attack method in this article, the condition is $m = 64$ and $d = 4$. Therefore, a 64-bit round key need minimum 16 nibble faults to recover itself. According to the key update scheme of PRESENT, 32 nibble faults need to be imported to restore the original 80-bit key. Then, the data complexity required for the attack is the sum of the fault ciphertext and the plaintext at the corresponding fault location. The complexity is calculated as $32 \times 4 \times 2 = 2^8$.

In addition to data complexity, computational complexity is often used to measure the computational time and resources consumed by attack methods, which is also an important metric. For computational complexity, the process would be divided into three steps.

The first step is to guess the 4-bit non-zero nibble of the input differential of S-box, so the complexity is 2^4.

Second, we need to compute x^{15}_{31} based on the exhausted values of Δx^{15}_{31}. According to the S-box differential property of PRESENT, the expected number of candidate values is 2.648 in (13), therefore, the complexity is $2.648 \times 2^4 \approx 2^{5.41}$.

Third, the 4-bit candidate of Δx^{15}_{31}in set M_0 needs to be further screened. So, the computational complexity is 2.648.

Therefore, the sum of computational complexity to recover 4-bit x^{15}_{31} is $2^4 + 2^{5.41} + 2.648 \approx 2^{5.94}$, and the complexity required to find 64-bit ΔX_{31} is $2^{5.94} \times 16 = 2^{9.94}$, which equals the complexity to recover 64-bit round key. According to the key update scheme of PRESENT, the complete 80-bit original key can be obtained after two consecutive rounds of the round key, so the complexity of recovering the full original key is $2^{9.94} \times 2 = 2^{10.94}$.

B. COMPLEXITY ANALYSIS OF GIFT

Use the same approach for the complexity analysis of GIFT. According to the characteristic of GIFT keys and constant xor layers, only 2-bit round key participate in xor for a 1-byte, so only 2-bit round key can be recovered from a 1-byte fault. Therefore, for GIFT, there are $m = 32$ and $d = 2$, which mean 16 nibble faults need to be imported at least. According to the GIFT key update scheme, 64 nibble faults are imported to restore the original 128-bit key. Then, the data complexity required for the attack is the sum of the fault ciphertext and the plaintext of the corresponding fault location. The complexity is calculated as $64 \times 4 \times 2 = 2^9$.

For computational complexity, the process would be divided into three steps.

The first step is to guess the 4-bit non-zero nibble of the input differential of S-box, so the complexity is 2^4.

Second, we need to compute x^{15}_{28} based on the exhausted values of Δx^{15}_{28}. According to the S-box differential property of PRESENT, the expected number of candidate values is 2.584 in (18), therefore, the complexity is $2.584 \times 2^4 \approx 2^{5.37}$.

Third, the 4-bit candidate of Δx^{15}_{28} in set M_1 needs to be further screened. So, the computational complexity is 2.584.

Therefore, the sum of computational complexity to recover 4-bit x^{15}_{28} is $2^4 + 2^{5.37} + 2.584 \approx 2^{5.91}$, and the complexity required to find 64-bit ΔX_{28} is $2^{5.91} \times 16 = 2^{9.91}$, which equals the complexity to recover 64-bit round key. According

to the key update scheme of GIFT, the complete 128-bit original key can be obtained after two consecutive rounds of the round key, so the complexity of recovering the full original key is $2^{9.91} \times 4 = 2^{11.91}$.

VI. EXPERIMENT

The encryption process of GIFT and PRESENT is similar, and the attack method in this article has obvious effect on both. Therefore, only the attack experiment on GIFT is given.

A. EXPERIMENTAL CONFIGURATION

The hardware is configured as a PC (the CPU is Intel Core i5-4200M 2.5GHz, the operating system is 64-bit, and the memory is 4GB), and the programming environment is the C++ language in the platform of Microsoft Visual Studio 2019.

B. EXPERIMENTAL RESULTS

Fault injection is implemented by the programming language modify the encryption process of GIFT. Then, the incorrect ciphertext obtained from the injected random fault was processed. At last, the number of incorrect ciphertext needed for the round key and the running time of the program are recorded. We carried out 30-times experiments, and the results are shown in Table 10.

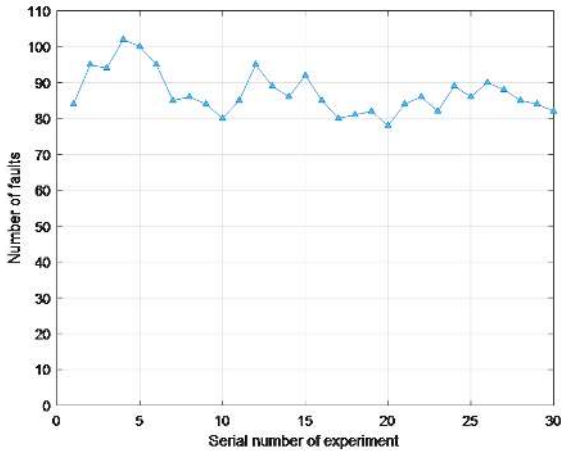

FIGURE 2. Experimental results of DFA on GIFT.

As can be seen from the above table, this attack method requires an average of 87 nibble faults to recover all keys of GIFT, which is higher than the theoretical result of 64 nibble faults mentioned above. Meanwhile, the attack can be completed in about 500ms. In order to show the fluctuation of the results of these 30-times experiments, we drew the line chart as shown in Fig. 2. As can be seen from the figure, the fluctuation of the results decreases with the increase of the number of experiment.

TABLE 10. Experimental result of differential fault analysis on GIFT.

Serial number	Number of faults to attack of the i^{th} round					Time required (*ms*)
	i=25	*i*=26	*i*=27	*i*=28	all	
1	20	18	25	21	84	496
2	22	26	28	19	95	510
3	21	19	22	32	94	508
4	25	24	30	23	102	523
5	24	26	26	24	100	520
6	28	22	23	22	95	511
7	19	21	26	19	85	496
8	24	20	24	18	86	497
9	18	22	22	22	84	493
10	22	22	18	18	80	490
11	25	18	17	25	85	495
12	30	20	25	20	95	510
13	19	22	23	25	89	501
14	22	26	18	20	86	497
15	26	24	23	19	92	504
16	24	19	24	18	85	496
17	21	20	20	19	80	485
18	20	19	20	22	81	486
19	21	21	20	20	82	488
20	18	21	21	18	78	482
21	21	22	19	22	84	492
22	22	21	20	23	86	498
23	20	21	19	23	82	489
24	24	23	22	21	89	500
25	22	23	20	21	86	498
26	21	24	23	22	90	502
27	21	23	22	22	88	499
28	21	20	24	20	85	495
29	23	21	20	20	84	492
30	19	20	21	22	82	490

C. RESULTS DISCUSSION

The number of faults required by the experimental results is larger than that of the theoretical analysis, which we believe is mainly due to the following two reasons:

(1) The round key of is closely related to the solution of the differential equation (18), and according to the differential property of GIFT, the solution of the differential equation is not unique. Therefore, sometimes one round of encryption needs to be attacked multiple times to obtain a unique round of keys.

(2) Because the sample space of this article is limited, so the result is different from the theoretical value. In this article, fault propagation paths effectively used by the sample space are slightly less than the theoretical value. Namely,

TABLE 11. The cryptanalysis results of PRESENT-80 in public literature.

Attack methods	Attack rounds	Computational complexity	Data complexity	Reference
Differential	16	2^{64}	2^{64}	[11]
Saturation	16	2^{20}	2^{36}	[12]
Linear	26	2^{72}	2^{64}	[13]
Multiple Differential	18	2^{79}	2^{64}	[14]
Structure	18	2^{76}	2^{64}	[15]
Biclique	21	$2^{78.9}$	2^{64}	[16]
Improbable Differential	13	$2^{63.86}$	$2^{62.62}$	[17]
Multiple-inputs Single-output Differential	16	$2^{59.16}$	$2^{59.16}$	[18]
Linear	12	2^{36}	2^{32}	[19]
Multiple Differential	16	$2^{57.6}$	2^{61}	[20]
DFA	30,31	$2^{10.94}$	2^{8}	This paper

TABLE 12. The cryptanalysis results of GIFT-64 in public literature.

Attack methods	Attack rounds	Computational complexity	Data complexity	Reference
Star	16	$2^{127.48}$	2	[25]
Biclique	16	$2^{127.36}$	2^{32}	[25]
Unbalanced Biclique	28	$2^{122.88}$	2^{16}	[26]
Differential	16,17	2^{83}	2^{62}	[27]
Related-key Differential	19	-	2^{47}	[28]
DFA	25,26,27,28	$2^{11.91}$	2^{9}	This paper

the impact of each fault on the input of S-box is slightly less than the theoretical value. Therefore, the actual incorrect ciphertext required would be marginally more than the theoretical value. On average, it only takes 87 faults to recover all the key information in one second.

VII. CONCLUSION

In this article, a general DFA is proposed for PRESENT and its improved algorithm GIFT. The effectiveness of this method is also verified on GIFT. By analyzing the permutation law of P permutation layer and combining the differential characteristics, the following conclusions are obtained:

(1) According to PRESENT's P permutation layer, nibble fault is imported into the 30^{th} and 31^{th} rounds respectively. In theory, 32 nibble faults are needed to fully recover the original 80-bit key, with a computational complexity of $2^{10.94}$ and a data complexity of 2^8. The following table lists this method and the existing attack methods for PRESENT.

It can be seen from Table 10 that the PRESENT is attacked by this method, which has great advantages in both computational complexity and data complexity.

(2) According to GIFT's P permutation layer, nibble fault is imported into the 25^{th}, 26^{th}, 27^{th} and 28^{th} rounds respectively. In theory, 64 nibble fault are needed to fully recover the original 128-bit key, with a computational complexity of $2^{11.91}$ and a data complexity of 2^9. The following table lists this method and the existing attack methods for GIFT.

It can be seen from Table 11 that the GIFT is attacked by this method, which has great advantages in both computational complexity and data complexity.

(3) GIFT is improved compared with PRESENT, which can resist differential analysis, linear analysis and algebraic analysis[24], but the nibble differential fault attack proposed in this article has obvious effect. In addition, this method is simple, clear, and has a certain universality, which can be applied to other lightweight cipher that P permutation layer has a certain propagation law. Although different ciphers have different permutation layers, by studying the propagation law, it is possible to recover the original key by this method.

Our proposed a general DFA approach for PRESENT and GIFT performs well in both data complexity and computational complexity. However, the hardware implementation may be limited because it is difficult to induce a nibble fault in the real situation. Therefore, in the future work, we will implement this attack method on FPGA or other hardware, and study the performance of this method in the real environment. In addition, we will continue to explore more DFA approaches to other lightweight block ciphers. Finally, we will discuss the security of various cryptographic application scenarios, such as privacy protection [31], [32] and identity authentication [33].

ACKNOWLEDGMENT

Haoxiang Luo would like to thank the Glasgow College, UESTC, for their support of this research.

REFERENCES

[1] E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems," in *Proc. Annu. Int. Cryptol. Conf. (CRYPTO)*, 1997, pp. 513–525.

[2] L. Breveglieri, I. Koren, and P. A. Maistri, "A fault attack against the FOX cipher family," in *Proc. Int. Workshop Fault Diagnosis Tolerance Cryptogr. (FDTC)*, 2006, pp. 98–105.

[3] Z. Lei and W. Wen-Ling, "Differential fault analysis on SMS4," *Chin. J. Comput.*, vol. 9, no. 9, pp. 1596–1602, 2006.

[4] S. Patranabis, A. Chakraborty, D. Mukhopadhyay, and P. P. Chakrabarti, "Fault space transformation: A generic approach to counter differential fault analysis and differential fault intensity analysis on AES-like block ciphers," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 5, pp. 1092–1102, May 2017.

[5] L. Dong, H. Zhang, L. Zhu, S. Sun, H. Gan, and F. Zhang, "Analysis of an optimal fault attack on the LED-64 lightweight cryptosystem," *IEEE Access*, vol. 7, pp. 31656–31662, 2019.

[6] D.-P. Le, S. L. Yeo, and K. Khoo, "Algebraic differential fault analysis on SIMON block cipher," *IEEE Trans. Comput.*, vol. 68, no. 11, pp. 1561–1572, Nov. 2019.

[7] S. Lim, J. Lee, and D.-G. Han, "Improved differential fault attack on LEA by algebraic representation of modular addition," *IEEE Access*, vol. 8, pp. 212794–212802, 2020.

[8] L. Song and L. Hu, "Differential fault attack on the PRINCE block cipher," in *Proc. Int. Workshop Lightweight Cryptogr. Secur. Privacy (CSP)*, 2013, pp. 43–54.

[9] H. Luo, Y. Wu, and W. Chen, "Differential fault attack on TWINE block cipher with nibble," in *Proc. IEEE 20th Int. Conf. Commun. Technol. (ICCT)*, Nanning, China, Oct. 2020, pp. 1151–1155.

[10] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe, "PRESENT: An ultra-lightweight block cipher," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst. (CHES)*, 2007, pp. 450–466.
[11] M. Wang, "Differential cryptanalysis of reduced-round PRESENT," in *Proc. Int. Conf. Cryptol. Afr. (AFRICACRYPT)*, 2008, pp. 40–49.
[12] B. Collard and F. X. Standaert, "A statistical saturation attack against the block cipher PRESENT," in *Proc. Cryptographers' Track RSA Conf. (CT-RSA)*, 2009, pp. 95–210.
[13] M. Wang, Y. Sun, E. Tischhauser, and B. Preneel, "A model for structure attacks, with applications to PRESENT and Serpent," in *Proc. Int. Workshop Fast Softw. Encryption (FSE)*, 2012, pp. 49–68.
[14] Z. Gong, S.-S. Liu, Y.-M. Wen, and S.-H. Tang, "Biclique analysis on the reduced-round PRESENT," *Chin. J. Comput.*, vol. 36, no. 6, pp. 1139–1148, Mar. 2014.
[15] C. Tezcan, "Improbable differential attacks on present using undisturbed bits," *J. Comput. Appl. Math.*, vol. 259, pp. 503–511, Mar. 2014.
[16] Y. TIAN, S. Z. CHEN, and Y. B. DAI, "Improved differential attack on 16-round present cipher," *J. Cryptologic Res.*, vol. 3, no. 6, pp. 573–583, 2016.
[17] G. Liu, C. Jin, and Z. Kong, "Key recovery attack for PRESENT using slender-set linear cryptanalysis," *Sci. China Inf. Sci.*, vol. 59, no. 3, p. 32110, Mar. 2016.
[18] S. Heyingxiu, "Research on differential security against several typical block cipher," Ph.D. dissertation, School Cryptogr. Eng., PLA Inf. Eng. Univ., Zhengzhou, China, 2017.
[19] C. Wei-Jian, Z. Si-Yu, Z. Rui-Jie, and Z. Xiao-Ning, "The differential fault attack of PRESENT cipher," *J. Univ. Electron. Sci. Technol. China*, vol. 6, pp. 865–869, 2019.
[20] C. A. Lara-Nino, A. Diaz-Perez, and M. Morales-Sandoval, "Lightweight hardware architectures for the present cipher in FPGA," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 9, pp. 2544–2555, Sep. 2017.
[21] P. Moucha, S. Jerabek, and M. Novotny, "Novel dummy rounds schemes as a DPA countermeasure in PRESENT cipher," in *Proc. 23rd Int. Symp. Design Diag. Electron. Circuits Syst. (DDECS)*, Novi Sad, Serbia, Apr. 2020, pp. 1–4.
[22] S. Banik, S. K. Pandey, T. Peyrin, Y. Sasaki, and S. M. Sim, "GIFT: A small present: Towards reaching the limit of lightweight encryption," in *Proc. Cryptograph. Hardw. Embedded Systems. (CHES)*, 2017, pp. 321–345.
[23] G. Weibo, L. Bing, and W. Yang, "Biclique analysis of GIFT-64," *Appl. Res. Comput.*, vol. 5, pp. 1470–1473, Mar. 2020.
[24] G. Han, H. Zhao, and C. Zhao, "Unbalanced biclique cryptanalysis of full-round GIFT," *IEEE Access*, vol. 7, pp. 144425–144432, 2019.
[25] J. Zhao, S. Xu, Z. Zhang, X. Y. Dong, and Z. Li, "Differential analysis of lightweight block cipher GIFT," *J. Cryptologic Res.*, vol. 5, no. 4, pp. 335–343, 2018.
[26] M. Cao and W. Zhang, "Related-key differential cryptanalysis of the reduced-round block cipher GIFT," *IEEE Access*, vol. 7, pp. 175769–175778, 2019.
[27] S. Patranabis, N. Datta, D. Jap, J. Breier, S. Bhasin, and D. Mukhopadhyay, "SCADFA: Combined SCA+DFA attacks on block ciphers with practical validations," *IEEE Trans. Comput.*, vol. 68, no. 10, pp. 1498–1510, Oct. 2019.
[28] J. Breier, D. Jap, X. Hou, and S. Bhasin, "On side channel vulnerabilities of bit permutations in cryptographic algorithms," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 1072–1085, 2020.
[29] A. Jati, N. Gupta, A. Chattopadhyay, S. K. Sanadhya, and D. Chang, "Threshold implementations of GIFT: A trade-off analysis," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 2110–2120, 2020.
[30] L. Dalmasso, F. Bruguier, P. Benoit, and L. Torres, "Evaluation of SPN-based lightweight crypto-ciphers," *IEEE Access*, vol. 7, pp. 10559–10567, 2019.
[31] Y. Liu, H. Wang, M. Peng, J. Guan, and Y. Wang, "An incentive mechanism for privacy-preserving crowdsensing via deep reinforcement learning," *IEEE Internet Things J.*, early access, Dec. 24, 2020, doi: 10.1109/JIOT.2020.3047105.
[32] Y. Liu, T. Feng, M. Peng, J. Guan, and Y. Wang, "DREAM: Online control mechanisms for data aggregation error minimization in privacy-preserving crowdsensing," *IEEE Trans. Dependable Secure Comput.*, early access, Jul. 24, 2020, doi: 10.1109/TDSC.2020.3011679.
[33] H. Luo, W. Chen, C. Chen, Y. Yang, Y. Zhang, and Y. Wu, "Analysis of a multichannel lightweight identity authentication method," in *Proc. IEEE 19th Int. Conf. Commun. Technol. (ICCT)*, Xi'an, China, Oct. 2019, pp. 1285–1290.

HAOXIANG LUO (Member, IEEE) was born in Sichuan, China, in 1999. He is currently pursuing the bachelor's degree in communication engineering with the Glasgow College, University of Electronic Science and Technology of China (UESTC), and has a minor in business administration (direction of innovation and entrepreneurship) with the Institute of Innovation and Entrepreneurship, UESTC.

He has served as a Research Assistant with the Tsinghua Sichuan Energy Internet Research Institute, from July 2020 to November 2020. He currently leads one provincial (in Sichuan) and one university-level (in UESTC) science and technology innovation student program. His research interests include the IoT technology, cryptography theory, and blockchain technology. He is a Reviewer of IEEE Access.

WEIJIAN CHEN was born in Zhejiang, China, in 1956. He received the B.S. degree in electronic engineering and computer science from Shanghai Jiao Tong University (SJTU), Shanghai, in 1982, and the M.S. degree in computer and automation from Chongqing University (CU), Chongqing, in 1988.

From 1982 to 1985, he has worked as an Assistant Engineer with China State Shipbuilding Corporation Ltd. He joined the University of Electronic Science and Technology of China (UESTC) as a Teacher. From 1994 to 1996, he has served as the Deputy Director of the Industrial Division, UESTC. From 1996 to 2003, he has served as the Director and the Chief-Engineer of the Affiliated Factory, UESTC. He is currently a Professor with the School of Information and Communication Engineering, UESTC, where he is also the Chief Professor of key courses such as information theory and analog circuits. In recent five years, he has published more than 30 papers in various well-known academic journals or conferences. His research interests include wireless and mobile communications, cryptography theory and technology, communication networks, and the IoT technology.

XINYUE MING was born in Sichuan, China, in 1999. She is currently pursuing the bachelor's degree in communication engineering with the Glasgow College, University of Electronic Science and Technology of China (UESTC). Her research interests include information security and image processing.

YIFAN WU was born in Jiangsu, China, in 1999. She is currently pursuing the bachelor's degree in communication engineering with the Glasgow College, University of Electronic Science and Technology of China (UESTC). Her research interests include information security and communication engineering.