Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2014, Article ID 842675, 10 pages
http://dx.doi.org/10.1155/2014/842675

Research Article

The Security Weakness of Block Cipher Piccolo against Fault Analysis

Junghwan Song, Kwanhyung Lee, and Younghoon Jung

Department of Mathematics, Hanyang University, Seoul 133-791, Republic of Korea

Correspondence should be addressed to Younghoon Jung; sky1236@hanyang.ac.kr

Received 23 December 2013; Accepted 21 January 2014; Published 13 March 2014

Academic Editor: Jongsung Kim

Piccolo is a 64-bit lightweight block cipher which is able to be implemented in constrained hardware environments such as a wireless sensor network. Fault analysis is a type of side channel attack and cube attack is an algebraic attack finding sufficiently low-degree polynomials in a cipher. In this paper, we show a fault analysis on the Piccolo by using cube attack. We find 16 linear equations corresponding to a round function F by cube attack, which are used to fault analysis. Our attack has the complexity of $2^{8.49}$ and $2^{9.21}$ encryptions with fault injections of target bit positions into Piccolo-80 and Piccolo-128, respectively. And our attack needs $2^{20.86}$ and $2^{21.60}$ encryptions with *random* 4-bit fault injections for Piccolo-80 and Piccolo-128, respectively.

1. Introduction

Fault analysis is a type of side channel attack. This analysis was introduced by Boneh et al. in [1]. Differential fault analysis (DFA), which is an improved method of fault analysis, was introduced by Biham and Shamir in [2]. DFA is applied to various block ciphers such as AES [3, 4], ARIA [5], SEED [6], CLEFIA [7], LED [8], Piccolo [9–12], PRESENT [13], and KATAN32 [14]. Cube attack was introduced by Dinur and Shamir in [15]. This attack is an algebraic attack by finding sufficiently low-degree polynomials in a cipher. Cube attack is applied to various cryptosystems such as block cipher [16] and stream cipher [15, 17].

In CHES 2011, Piccolo was introduced by Shibutani et al. in [18]. Piccolo is a block cipher which supports 80-bit and 128-bit secret key size. In this paper, we analyze two versions of Piccolo [18] with fault analysis by using cube attack. In ISPEC 2012, fault analysis using cube attack was introduced by Abdul-Latip et al. in [16]. In this paper, we apply this method on Piccolo-80 and Piccolo-128. As a result, we find 16 linear equations corresponding to a round function F by cube attack, which are used to fault analysis.

Piccolo is analyzed by various techniques. In ISPEC 2012, Wang et al. suggest biclique cryptanalysis of reduced round Piccolo in [19]. They analyze reduced version of Piccolo-80 without postwhitening keys XOR and reduced 28-round Piccolo-128 without prewhitening keys XOR. In 2013, Song et al. suggest biclique cryptanalysis of full rounds of Piccolo [20]. And also Jeong suggests a differential fault analysis of full rounds of Piccolo [9].

In this paper, we show a fault analysis on the Piccolo by using cube attack. We find 16 linear equations corresponding to a round function F of Piccolo by using cube attack. These equations are used to our attack. In this paper, we describe the case that an adversary injects random 4-bit faults. Our attack has the complexity of $2^{20.86}$ and $2^{21.60}$ encryptions for Piccolo-80 and Piccolo-128, respectively, while the assumption of [9] is an adversary that injects random byte faults. Reference [9] has the complexity of 2^{24} and 2^{40} encryptions for Piccolo-80 and Piccolo-128, respectively. Our attack has a lower computational complexity than [9], even though the assumption of fault injection in our attack differs from [9].

In Section 2, we briefly describe the procedures of cube attack and cube tester. And then we describe the brief specifications of Piccolo in Section 3. In Section 4, a method of fault analysis of Piccolo by using cube attack is presented. Finally, our conclusions are in Section 5.

2. Cube Attack and Cube Tester

Algebraic attack is to find a solution, which is the key, of a system of equations that represent target cipher with given plaintext and the corresponding ciphertext, that is representing cipher as a system of equations with multiple variables defined over finite field where each key bit is represented as a variable in the system. Solving the system is equivalent to finding the secret key of the target cipher. Cube attack is an algebraic attack finding sufficiently low-degree polynomials in cipher.

2.1. Cube Attack. Cube attack was introduced by Dinur and Shamir in [15]. Cube attack is a chosen plaintext attack. The main idea of cube attack is to find linear equations consisting of secret variables by using cube sum. Let $p(v_1, \ldots, v_n, k_1, \ldots, k_m)$ be a polynomial derived from a cipher, where $v_1, \ldots, v_n$ are public variables and $k_1, \ldots, k_m$ are secret variables. In other words, each secret variable is considered a bit in secret key and each public variable is considered a bit in plaintext or internal state. Let $I = \{I_1, \ldots, I_s\} \subseteq \{1, \ldots, n\}$ be a set and let t_I be the monomial $x_{I_1} x_{I_2} \ldots x_{I_s}$. Note that the set in terms of I is called cube index. Then the polynomial p is represented by three polynomials t_I, $p_{S(I)}$, and q as the following form:

$$\begin{aligned} &p(v_1, \ldots, v_n, k_1, \ldots, k_m) \\ &\quad = t_I \cdot p_{S(I)} + q(v_1, \ldots, v_n, k_1, \ldots, k_m), \end{aligned} \tag{1}$$

where q is not consisting of a monomial which has a factor t_I.

Cube attack is required to check the linearity of $p_{S(I)}$ which is called superpoly. A superpoly $p_{S(I)}$ is called a maxterm if $p_{S(I)}$ is linear. We use the following cube sum to find a $p_{S(I)}$:

$$p_{S(I)} = \sum_{(v_{I_1}, \ldots, v_{I_s}) \in \mathrm{GF}(2)^s} p(v_1, \ldots, v_n, k_1, \ldots, k_m), \tag{2}$$

where plaintext bits except cube index $(v_i, i \in \{1, \ldots, n\} - I)$ are fixed as constants.

As the above representation, cube is completed with the sum total 2^S pairs of plaintext and ciphertext for a cube index $I = \{I_1, \ldots, I_s\}$. To check whether $p_{S(I)}$ is a maxterm, linearity test is required. Let $p_{S(I)}(k_1, \ldots, k_m)$ be a polynomial of m variables over GF(2). Let t be the number of tests. The following is a procedure of linearity test.

Step 1. Choose 2 random vectors $x, y \in \mathrm{GF}(2)^m$.

Step 2. If $p_{S(I)}(x) \oplus p_{S(I)}(y) \oplus p_{S(I)}(0) \neq p_{S(I)}(x \oplus y)$, then $p_{S(I)}$ is not linear. Stop the test.

Step 3. Repeat Steps 1 and 2, t times.

Step 4. $p_{S(I)}$ is linear. Stop the test, where $0 = (0, \ldots, 0) \in \mathrm{GF}(2)^m$.

If $p_{S(I)}(k_1, \ldots, k_m)$ is linear, the above equation in Step 2 is always correct for all inputs $x, y \in \mathrm{GF}(2)^m$. Because checking all inputs is impossible, an upper bound of number of linearity tests has to be set. If there are at most d_1 elements in a cube index for testing linearity, at most $2^{d_1} \times (3 \times t + 1)$ pairs of plaintext and ciphertext are needed. Cube attack consists of preprocessing phase and online phase. Preprocessing phase is to find a system of linear equations by using cube sum and linearity test. Online phase is recovering the master key stored by using an encryption oracle. The following are details for the two phases.

Preprocessing Phase. After finding a polynomial from a cipher, find a cube, that is, a maxterm, by using linearity test. Since we know output after all plaintext bits are entered in encryption oracle, fix plaintext except cube index as a constant. Fixed constants of every cube do not have to be equal. Let f_i be a maxterm which consists of only secret variables $k_1, \ldots, k_m$ and let b_i be the value of the maxterm f_i which is found from online phase. We consider the following system of equations:

$$\begin{gathered} f_1(k_1, \ldots, k_m) = b_1 \\ \vdots \\ f_l(k_1, \ldots, k_m) = b_l. \end{gathered} \tag{3}$$

In the preprocessing phase, find enough maxterms to recover the master key and precalculate this system of equations by using Gaussian elimination. If we find m linear independent maxterms, then recover all the master keys with m^3 operations for recovery by using Gaussian elimination. In general, it is lower than complexity $l \times 2^{d_1} \times (3 \times t + 1)$ for finding l maxterms. Let $f_1, \ldots, f_m$ be linearly independent. Then the master keys are represented as the following system:

$$\begin{gathered} k_1 = \sum_{i=1}^{m} a_{1,i} \cdot b_i \\ \vdots \\ k_m = \sum_{i=1}^{m} a_{m,i} \cdot b_i, \end{gathered} \tag{4}$$

where $a_{i,j} \in \mathrm{GF}(2)$.

Online Phase. In online phase, calculate the value of cube sum from an encryption oracle by using the cube that has been found in the preprocessing phase. Let each plaintext bit not in the cube be constant. The calculated value is b_i, that is, the value of the maxterm. By substituting the value b_i into (4), we recover the master key. Let cube index found at preprocessing phase have at most d_2 elements. Then the complexity of online phase is $m \times 2^{d_2}$.

2.2. Cube Tester. Cube attack finds a maxterm by testing linearity of $p_{S(I)}$ of a given polynomial p and cube index I. Cube tester distinguishes a polynomial from a random polynomial by many tests including linearity test. There are some other tests using cube sum in [21]. In cube attack, a plaintext bit not in the cube is fixed as a constant. However all bits not in the cube have to be considered variables in the cube tester. Since the purpose of using the cube tester is getting information, which are properties of polynomial, we use the

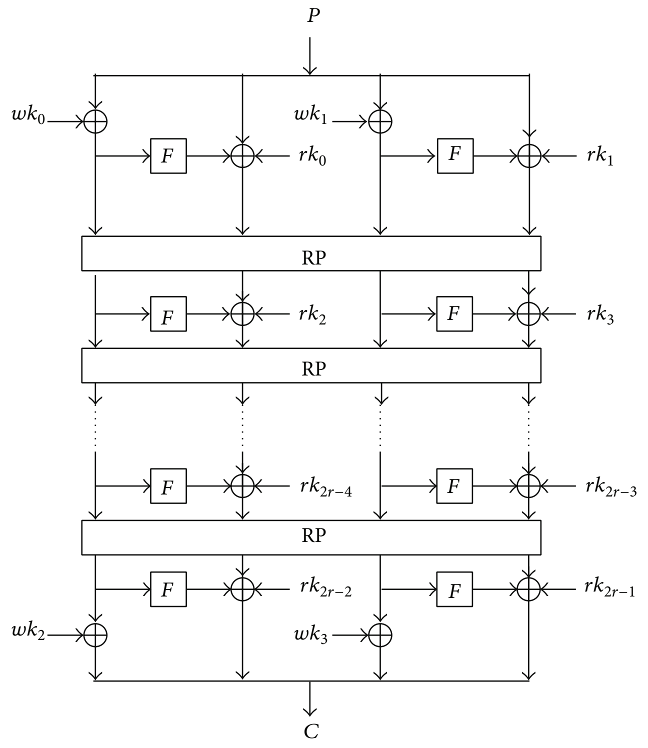

Figure 1: Encryption process of Piccolo.

low-degree test that is in [21]. The degree N is determined by low-degree test. Let I be a cube index, let J be the number of bits not in the cube index (i.e., $J = n + m - S$; $p_{S(I)}$ consists of J variables), and t is the number of tests. Since low-degree test is valid only when $p(0) = 0$ for the given polynomial p, we define $p^*_{S(I)}(x) = p_{S(I)}(x) + p_{S(I)}(0)$. Then low-degree test for the polynomial $p^*_{S(I)}(x)$ is as follows.

Step 1. Choose $N+1$ random vectors $y_1, \dots, y_{N+1} \in \mathrm{GF}(2)^J$.

Step 2. If $\sum_{\phi \neq S \subset \{y_1,\dots,y_{N+1}\}} p^*_{S(I)}(\sum_{y_i \in S} y_i) \neq 0$, then degree of $p_{S(I)} > N$. Stop the test.

Step 3. Repeat Steps 1 and 2, t times.

Step 4. Degree of $p_{S(I)} \leq N$. Stop the test.

If $N = 1$, then the low-degree test is similar to the linearity test. We use the idea of the cube tester which uses every bit not in the cube index (consisting of plaintext and the master key) as a variable.

3. Description of Piccolo

Piccolo is a 64-bit block cipher with 80- and 128-bit key size. The structure of Piccolo is a Feistel network. Piccolo-80 consists of 25 rounds and Piccolo-128 consists of 31 rounds. Figure 1 illustrates the working processing of Piccolo. Each round consists of two functions, round function F and round permutation RP. The round functions F and RP are as follows.

Table 1: S-box of Piccolo.

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$S(x)$	e	4	b	2	3	8	0	9	1	a	7	f	6	c	5	d

Round Function F. The round function F is defined by

$$F(x_0, x_1, x_2, x_3) = (S(x_0), S(x_1), S(x_2), S(x_3)) \cdot M \cdot (S(x_0), S(x_1), S(x_2), S(x_3))^t, \quad (5)$$

where X^t is the transposition of X.

$S(x)$ is the 4-bit S-box and M is the diffusion matrix as follows (see Table 1):

$$M = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix}. \quad (6)$$

The multiplications between M and vectors are defined by an irreducible polynomial $x^4 + x + 1$ over $\mathrm{GF}(2^4)$.

Round Permutation RP. The round permutation RP is defined by

$$\mathrm{RP}(x_0, x_1, \dots, x_7) = (x_2, x_7, x_4, x_1, x_6, x_3, x_0, x_5), \quad (7)$$

where x_i is byte.

For description of Piccolo and our attack, we denote intermediate variables before r-round as $X^r = X^r_0 \mid X^r_1 \mid X^r_2 \mid X^r_3 = (x^r_0, \dots, x^r_{63})$ and intermediate variables before r-round's RP function as $Y^r = Y^r_0 \mid Y^r_1 \mid Y^r_2 \mid Y^r_3$ (i.e., $\mathrm{RP}(Y^r) = X^{r+1}$). Let the round function F in r-round be $F(X^r_i) = (F_0(X^r_i), \dots, F_{15}(X^r_i))$ and let the round key be $rk_r = (k^r_0, \dots, k^r_{15})$. The other notations are as follows:

$(X^r_i)^L$: left 8 bits of X^r_i and $(X^r_i)^R$: right 8 bits of X^r_i;

K^L_i: left 8 bits of K_i and K^R_i: right 8 bits of K_i;

$A \mid B$: concatenation of A and B.

Let the 64-bit plaintext and ciphertext be P and C, respectively. Encryption of Piccolo is defined as follows:

(1) $P_0 \mid P_1 \mid P_2 \mid P_3 \leftarrow P$ (P_i is the 16-bit plaintext);

(2) $X^1_0 \leftarrow P_0 \oplus wk_0$, $X^1_1 = P_1$
$X^1_2 \leftarrow P_2 \oplus wk_1$, $X^1_3 = P_3$;

(3) for $i = 1$ to $r-1$

$$Y^i_0 \leftarrow X^i_0, Y^i_1 \leftarrow X^i_1 \oplus F(X^i_0) \oplus rk_{2i-2}$$
$$Y^i_2 \leftarrow X^i_2, Y^i_3 \leftarrow X^i_3 \oplus F(X^i_2) \oplus rk_{2i-1}$$
$$X^{i+1} \leftarrow \mathrm{RP}(Y^i);$$

(4) $Y^r_0 \leftarrow X^r_0 \oplus wk_2, Y^r_1 \leftarrow X^r_1 \oplus F(X^r_0) \oplus rk_{2r-2}$
$Y^r_2 \leftarrow X^r_2 \oplus wk_3, Y^r_3 \leftarrow X^r_3 \oplus F(X^r_2) \oplus rk_{2r-1}$;

(5) $C \leftarrow Y^r_0 \mid Y^r_1 \mid Y^r_2 \mid Y^r_3$ (Y^r_i is the 16-bit ciphertext).

Key schedule of Piccolo consists of the following.

Piccolo-80:

$$wk_0 \longleftarrow K_0^L \mid K_1^R, wk_1 \longleftarrow K_1^L \mid K_0^R,$$
$$wk_2 \longleftarrow K_4^L \mid K_3^R, wk_3 \longleftarrow K_3^L \mid K_4^R, \quad (8)$$

for $i \leftarrow 0$ to 24 do

if $i \mod 5 = 0$ or 2, then

$$(rk_{2i}, rk_{2i+1}) \longleftarrow (\text{con}_{2i}^{80}, \text{con}_{2i+1}^{80}) \oplus (K_2, K_3) \quad (9)$$

if $i \mod 5 = 1$ or 4, then

$$(rk_{2i}, rk_{2i+1}) \longleftarrow (\text{con}_{2i}^{80}, \text{con}_{2i+1}^{80}) \oplus (K_0, K_1) \quad (10)$$

if $i \mod 5 = 3$, then

$$(rk_{2i}, rk_{2i+1}) \longleftarrow (\text{con}_{2i}^{80}, \text{con}_{2i+1}^{80}) \oplus (K_4, K_4), \quad (11)$$

where con_i^{80} is the round constant.

Piccolo-128:

$$wk_0 \longleftarrow K_0^L \mid K_1^R, wk_1 \longleftarrow K_1^L \mid K_0^R,$$
$$wk_2 \longleftarrow K_4^L \mid K_7^R, wk_3 \longleftarrow K_7^L \mid K_4^R, \quad (12)$$

for $i \leftarrow 0$ to 61 do

if $(i+2) \mod 8 = 0$, then

$$(K_0, K_2, K_6, K_4) \longleftarrow (K_2, K_6, K_4, K_0)$$
$$(K_3, K_7, K_5) \longleftarrow (K_7, K_5, K_3) \quad (13)$$
$$rk_i \longleftarrow rk_{(i+2) \bmod 8} \oplus \text{con}_i^{128},$$

where con_i^{128} is the round constant.

Since key schedule of Piccolo is just performing XOR determined constants to the master key, recovering the round key and recovering the master key are the same. Table 2 is showing the master key used for the round key of Piccolo. Detailed descriptions of Piccolo are in [18].

4. Fault Analysis on the Piccolo

In this section, we show the fault analysis for Piccolo-80 and Piccolo-128. We assume that an adversary is able to make 4-bit errors in a maximum at a time on a round during an encryption process. By using cube sum, find system of linear equations in the common F of Piccolo-80 and Piccolo-128. And use the system to represent the phase recovering the master key of Piccolo-80 and Piccolo-128. Analysis of a round function F in Section 4.1 is corresponding to the preprocessing phase of cube attack. The attack in Sections 4.2 and 4.3 is the case of an encryption oracle that is given and is corresponding to online phase.

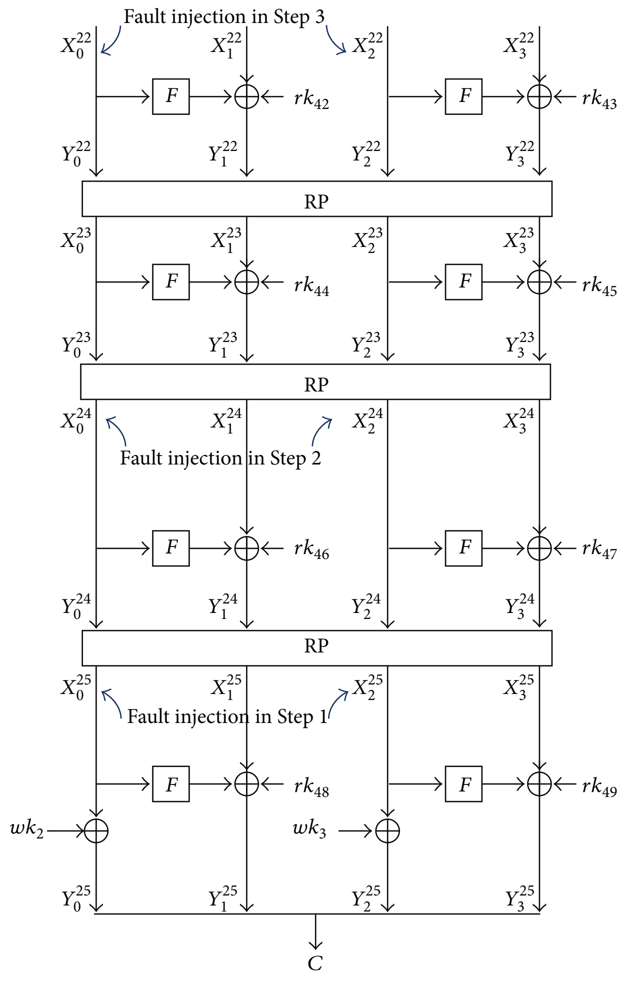

FIGURE 2: Fault analysis of Piccolo-80.

4.1. Equations of Round Function F. Since round function F is the same for Piccolo-80 and Piccolo-128, the results of fault injection attack on F are the same in the both algorithms. Let $F(X) = (F_0(X), \ldots, F_{15}(X))$, where $X = (x_0, x_1, \ldots, x_{15})$ is an 16-bit intermediate value and each $F_j(x)$ is a bit ($F : \text{GF}(2)^{16} \rightarrow \text{GF}(2)^{16}$). We test all possible cubes of degree 1 to degree 4 and all possible inputs for each cube. We get many linear polynomials and choose 16 appropriate polynomials for recovering the master key. Table 3 shows our selected 16 polynomials, cube index, and output bit (F_i).

4.2. Analysis on the Piccolo-80. We explain how to recover all the master keys of Piccolo-80. By key schedule of Piccolo-80, recovering wk_2, wk_3, rk_{44}, rk_{48}, and rk_{49} is equal to recovering all the master keys of Piccolo-80. Let plaintext P be given and let X_i^j, Y_i^j be intermediate values for plaintext P. In this paper, we recover the master key of Piccolo-80 by recovering some X_i^js. Figure 2 is for the last 4 rounds of Piccolo-80. The following is the attack on Piccolo-80.

Step 1. First, we analyze the last round (i.e., round 25). Perform cube sum by using the cube in Table 3 for F which

Table 2: Round key of Piccolo.

Piccolo-80			Piccolo-128		
Round	Round key	Master key	Round	Round key	Master key
First	wk_0, wk_1	$K_0^L \mid K_1^R, K_1^L \mid K_0^R$	First	wk_0, wk_1	$K_0^L \mid K_1^R, K_1^L \mid K_0^R$
1	rk_0, rk_1	K_2, K_3	1	rk_0, rk_1	K_2, K_3
2	rk_2, rk_3	K_0, K_1	2	rk_2, rk_3	K_4, K_5
3	rk_4, rk_5	K_2, K_3	3	rk_4, rk_5	K_6, K_7
4	rk_6, rk_7	K_4, K_4	4	rk_6, rk_7	K_2, K_1
5	rk_8, rk_9	K_0, K_1	5	rk_8, rk_9	K_6, K_7
6	rk_{10}, rk_{11}	K_2, K_3	6	rk_{10}, rk_{11}	K_0, K_3
7	rk_{12}, rk_{13}	K_0, K_1	7	rk_{12}, rk_{13}	K_4, K_5
8	rk_{14}, rk_{15}	K_2, K_3	8	rk_{14}, rk_{15}	K_6, K_1
9	rk_{16}, rk_{17}	K_4, K_4	9	rk_{16}, rk_{17}	K_4, K_5
10	rk_{18}, rk_{19}	K_0, K_1	10	rk_{18}, rk_{19}	K_2, K_7
11	rk_{20}, rk_{21}	K_2, K_3	11	rk_{20}, rk_{21}	K_0, K_3
12	rk_{22}, rk_{23}	K_0, K_1	12	rk_{22}, rk_{23}	K_4, K_1
13	rk_{24}, rk_{25}	K_2, K_3	13	rk_{24}, rk_{25}	K_0, K_3
14	rk_{26}, rk_{27}	K_4, K_4	14	rk_{26}, rk_{27}	K_6, K_5
15	rk_{28}, rk_{29}	K_0, K_1	15	rk_{28}, rk_{29}	K_2, K_7
16	rk_{30}, rk_{31}	K_2, K_3	16	rk_{30}, rk_{31}	K_0, K_1
17	rk_{32}, rk_{33}	K_0, K_1	17	rk_{32}, rk_{33}	K_2, K_7
18	rk_{34}, rk_{35}	K_2, K_3	18	rk_{34}, rk_{35}	K_4, K_3
19	rk_{36}, rk_{37}	K_4, K_4	19	rk_{36}, rk_{37}	K_6, K_5
20	rk_{38}, rk_{39}	K_0, K_1	20	rk_{38}, rk_{39}	K_2, K_1
21	rk_{40}, rk_{41}	K_2, K_3	21	rk_{40}, rk_{41}	K_6, K_5
22	rk_{42}, rk_{43}	K_0, K_1	22	rk_{42}, rk_{43}	K_0, K_7
23	rk_{44}, rk_{45}	K_2, K_3	23	rk_{44}, rk_{45}	K_4, K_3
24	rk_{46}, rk_{47}	K_4, K_4	24	rk_{46}, rk_{47}	K_6, K_1
25	rk_{48}, rk_{49}	K_0, K_1	25	rk_{48}, rk_{49}	K_4, K_3
Final	wk_2, wk_3	$K_4^L \mid K_3^R, K_3^L \mid K_4^R$	26	rk_{50}, rk_{51}	K_2, K_5
			27	rk_{52}, rk_{53}	K_0, K_7
			28	rk_{54}, rk_{55}	K_4, K_1
			29	rk_{56}, rk_{57}	K_0, K_7
			30	rk_{58}, rk_{59}	K_6, K_3
			31	rk_{60}, rk_{61}	K_2, K_5
			Final	wk_2, wk_3	$K_4^L \mid K_7^R, K_7^L \mid K_4^R$

takes X_0^{25}. For example, consider 6th equation of Table 3. Suppose that inject fault into x_4^{25} to x_8^{25}. Then, since fault is injected into only X_0^{25}, value of X_1^{25} or rk_{48} is not changed. We notate the following to explain our attack:

$X_0^{25} = (x_0^{25}, \ldots, x_{15}^{25}), X_1^{25} = (x_{16}^{25}, \ldots, x_{31}^{25})$;

$Y_0^{25} = (y_0^{25}, \ldots, y_{15}^{25})$;

$y_{12}^{25}[x_4^{25}]$: y_{12}^{25} when fault is injected into x_4^{25};

$y_{12}^{25}[x_8^{25}]$: y_{12}^{25} when fault is injected into x_8^{25};

$y_{12}^{25}[x_4^{25}, x_8^{25}]$: y_{12}^{25} when fault is injected into both x_4^{25}, x_8^{25}.

We calculate cube sum for cube index {4, 8} like the following:

$$\begin{aligned}\text{Cube sum} &= \sum_{x_4, x_8 \in \{0,1\}} F_{12}\left(x_0^{25}, \ldots, x_{15}^{25}\right) \\ &= \sum_{x_4, x_8 \in \{0,1\}} \left[F_{12}\left(x_0^{25}, \ldots, x_{15}^{25}\right) \oplus x_{28}^{25} \oplus k_{12}^{48}\right] \\ &= y_{12}^{25} \oplus y_{12}^{25}\left[x_4^{25}\right] \oplus y_{12}^{25}\left[x_8^{25}\right] \oplus y_{12}^{25}\left[x_4^{25}, x_8^{25}\right].\end{aligned} \tag{14}$$

y_{12}^{25} is not the output of F. But since cube sum does XOR even times, x_{28}^{25} and rk_{12}^{48} are offset. That is, we know value of cube sum cause of $Y_0^{25} \mid Y_1^{25} \mid Y_2^{25} \mid Y_3^{25} = C$. In the same way, cube sum using fault injection in this paper is performed. By performing cube sum for every cube in Table 3, we get 16 systems of equations. Recover input X_0^{25}.

Table 3: Cube sum result of $F(x_0, \ldots, x_{15})$.

Cube index	Outbit (F_i)	Polyequation
1, 5, 6	8	$x_0 + 1$
0, 8, 9, 11	10	$x_1 + 1$
1, 5, 6	12	$x_0 + x_2$
0, 8, 9, 11	7	x_3
0, 1, 5, 6	4	$x_4 + 1$
4, 8	12	$x_5 + x_9$
4, 5, 8, 9	4	$x_6 + 1$
4, 8, 9, 11	7	$x_5 + x_7 + 1$
5, 6, 9	12	$x_8 + 1$
4, 5, 7, 8	6	x_9
5, 6, 9	8	$x_{10} + 1$
4, 5, 7, 8	3	x_{11}
5, 6, 13	8	$x_{12} + x_{14}$
0, 12	4	$x_1 + x_{13}$
5, 6, 13	12	$x_{14} + 1$
0, 8, 11, 12	7	$x_3 + x_{15}$

Similarly, we recover input X_2^{25} using F which takes X_2^{25}. Since $X_0^{25} \oplus wk_2 = Y_0^{25}$, $X_2^{25} \oplus wk_3 = Y_2^{25}$, we recover wk_2, wk_3 (i.e., K_3, K_4).

Step 2. Since we know wk_2 and wk_3, calculate intermediate value X_0^{25}, X_2^{25} for given ciphertext. Round permutation RP in round 24 is as follows:

$$\begin{aligned} Y_1^{24} &= \left(X_0^{25}\right)^L \mid \left(X_2^{25}\right)^R, \qquad & Y_3^{24} &= \left(X_2^{25}\right)^L \mid \left(X_0^{25}\right)^R \\ Y_0^{24} &= \left(X_3^{25}\right)^L \mid \left(X_1^{25}\right)^R, \qquad & Y_2^{24} &= \left(X_1^{25}\right)^L \mid \left(X_3^{25}\right)^R. \end{aligned} \tag{15}$$

Therefore, we calculate Y_1^{24}, Y_3^{24} for given ciphertext. By using this, analyze round 24. In a similar way with Step 1, recover X_0^{24}, X_2^{24} by using the cube in Table 3 for F which takes X_0^{24}, X_2^{24}. Since $X_0^{24} = Y_0^{24}$, $X_2^{24} = Y_2^{24}$, $Y_0^{24} = (X_3^{25})^L \mid (X_1^{25})^R$, and $Y_2^{24} = (X_1^{25})^L \mid (X_3^{25})^R$, we recover X_1^{25}, X_3^{25}. Then we recover rk_{48}, rk_{49} (i.e., K_0, K_1) since $X_1^{25} \oplus F(X_0^{25}) \oplus rk_{48} = Y_1^{25}$, $X_3^{25} \oplus F(X_2^{25}) \oplus rk_{49} = Y_3^{25}$.

Step 3. We recover K_0, K_1, K_3, and K_4 so far. Given ciphertext C, we calculate X_0^{24}, X_1^{24}, X_2^{24}, and X_3^{24}. That is, we recover X_0^{23}, X_2^{23}, Y_1^{23}, and Y_3^{23}. We want to recover X_1^{23}. Since $X_0^{22} = Y_0^{22} = (X_3^{23})^L \mid (X_1^{23})^R$, $X_2^{22} = Y_2^{22} = (X_1^{23})^L \mid (X_3^{23})^R$, if we recover right 8 bits of X_0^{22} and left 8 bits of X_2^{22}, then we recover X_1^{23}. To recover right 8 bits of X_0^{22}, inject fault into bits corresponding to cube index of last 8 equations in Table 3. For recovering left 8 bits of X_2^{22}, inject fault into bits corresponding to cube index of first 8 equations in Table 3. Then we recover X_1^{23}. Since $X_1^{23} \oplus F(X_0^{23}) \oplus rk_{44} = Y_1^{23}$,we recover rk_{44} (i.e., K_2).

Use the above 3 steps to recover all the master keys used for Piccolo-80. This needs the assumption that we inject fault into at most 4 bits in the same time. Thus in this paper we consider the following as an analyzing way.

Table 4: Attack complexity of Piccolo-80.

Assumption	Required fault	Complexity
Assumption 1	$132 \approx 2^{7.04}$	2^{48}
Assumption 2	$264 \approx 2^{8.04}$	$2^{16.01}$
Assumption 3	$347 \approx 2^{8.44}$	$359.1 \approx 2^{8.49}$

Assumption 1. We inject fault into at most 4 bits in the same time. But, apply this to only last round.

Assumption 2. We inject fault into at most 4 bits in the same time. But, apply this to only rounds 24 and 25.

Assumption 3. We inject fault into at most 4 bits in the same time.

That is, suppose that we analyze only Step 1 or Steps 1 and 2. Even though we analyze only Steps 1 and 2, since at least 32 bits of 80-bit master key are recovered, we recover all the master keys with less operations than brute-force attack. To recover X_i^r, inject fault into 11 bits among 16 bits of internal state X_i^r by using injections 66 times. To recover left 8 bits and right 8 bits of X_i^r (i.e., $(X_i^r)^L, (X_i^r)^R$), we inject fault into 8 bits and 10 bits among 16 bits of X_i^r, respectively. Then $(X_i^r)^L, (X_i^r)^R$ are recovered by using injections 44 and 39 times, respectively.

Under Assumption 1, we need 133 encryptions for recovering 32-bit master key (wk_2, wk_3). We exclusively search to recover remaining 48-bit master key. Therefore, we need total $133 + 2^{48} \approx 2^{48}$ encryptions for recovering the master key under Assumption 1.

Under Assumption 2, we need 133 encryptions for recovering 32-bit master key (wk_2, wk_3). For recovering 32-bit round keys rk_{48} and rk_{49}, we have to calculate Y_1^{24}, Y_3^{24}. Given ciphertext, calculating Y_1^{24} is equivalent to 0.5 round encryption. So is Y_3^{24}. Hence, we need $132 + (132 \times 0.5 + 1)/25$ encryptions for recovering 32-bit round keys rk_{48} and rk_{49}. We exclusively search to recover remaining 16-bit master key. Therefore, we need total $133 + (132 + 67/25) + 2^{16} \approx 2^{16.01}$ encryptions for recovering the master key under Assumption 2.

Under Assumption 3, we need $133+(132+67/25)$ encryptions for recovering wk_2, wk_3, rk_{48}, and rk_{49}. Given ciphertext, calculating Y_1^{22} is equivalent to 2.5 round encryption. We need $44 + 39 + \{(44 + 39) \times 2.5 + 1 \times 3\}/25$ encryptions for recovering 32-bit round keys rk_{44}, rk_{45}. Therefore, we need total $133+(132+67/25)+(83+210.5/25) \approx 2^{8.49}$ encryptions for recovering the master key under Assumption 3. Table 4 is showing encryption complexity needed for recovering the master key of each assumption.

4.3. Analysis on the Piccolo-128. Piccolo-128 recovers the master key in a similar way to Piccolo-80. Figure 3 is for the last 5 rounds of Piccolo-128. The following is how Piccolo-128 recovers the master key.

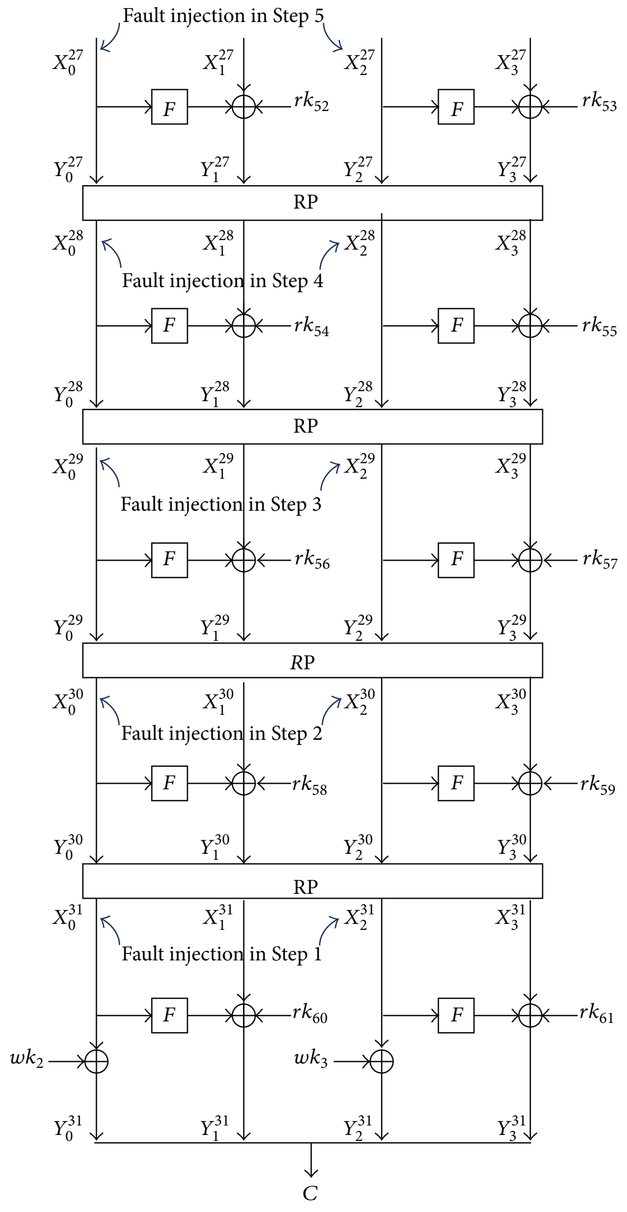
Fault injection in Step 5
Fault injection in Step 4
Fault injection in Step 3
Fault injection in Step 2
Fault injection in Step 1
F
RP
rk_{52} rk_{53} rk_{54} rk_{55} rk_{56} rk_{57} rk_{58} rk_{59} rk_{60} rk_{61}
wk_2 wk_3
C

Figure 3: Fault analysis of Piccolo-128.

Step 1. First, we analyze the last round (i.e., round 25). In a similar way with Step 1 in analysis of Piccolo-80, recover X_0^{31}, X_2^{31} by using the cube in Table 3 for F which takes X_0^{31}, X_2^{31}. Since $X_0^{31} \oplus wk_2 = Y_0^{31}$, $X_2^{31} \oplus wk_3 = Y_2^{31}$, we recover wk_2, wk_3 (K_4, K_7).

Step 2. Since we know wk_2 and wk_3, calculate intermediate values X_0^{31}, X_2^{31} for given ciphertext. Since $Y_1^{30} = (X_0^{31})^L \mid (X_2^{31})^R$, $Y_3^{30} = (X_2^{31})^L \mid (X_0^{31})^R$, we calculate Y_1^{30}, Y_3^{30} for given ciphertext. In a similar way with Step 1 in analysis of Piccolo-80, recover X_0^{30}, X_2^{30} by using the cube in Table 3 for F which takes X_0^{30}, X_2^{30}. We recover X_1^{31}, X_3^{31}, since $X_0^{30} = Y_0^{30}$, $X_2^{30} = Y_2^{30}$, and $Y_0^{30} = (X_3^{31})^L \mid (X_1^{31})^R$, $Y_2^{30} = (X_1^{31})^L \mid (X_3^{31})^R$. Since $X_1^{31} \oplus F(X_0^{31}) \oplus rk_{60} = Y_1^{31}$, $X_3^{31} \oplus F(X_2^{31}) \oplus rk_{61} = Y_3^{31}$, we recover rk_{60}, rk_{61} (K_2, K_5).

Step 3. Since we know wk_2, wk_3, rk_{60}, and rk_{61}, calculate intermediate values X_0^{30}, X_2^{30}, Y_1^{30}, and Y_3^{30} for given ciphertext. Since $Y_1^{29} = (X_0^{30})^L \mid (X_2^{30})^R$, $Y_3^{29} = (X_2^{30})^L \mid (X_0^{30})^R$, we calculate Y_1^{29}, Y_3^{29} for given ciphertext. In a similar way with Step 1 in analysis of Piccolo-80, recover X_0^{29}, X_2^{29} by using the cube in Table 3 for F which takes X_0^{29}, X_2^{29}. We recover X_1^{30}, X_3^{30} since $X_0^{29} = Y_0^{29}$, $X_2^{29} = Y_2^{29}$, and $Y_0^{29} = (X_3^{30})^L \mid (X_1^{30})^R$, $Y_2^{29} = (X_1^{30})^L \mid (X_3^{30})^R$. Since $X_1^{30} \oplus F(X_0^{30}) \oplus rk_{58} = Y_1^{30}$and $X_3^{30} \oplus F(X_2^{30}) \oplus rk_{59} = Y_3^{30}$, we recover rk_{58}, rk_{59} (K_6, K_3).

Step 4. This step is similar to Step 3 in analysis of Piccolo-80. Given ciphertext C, we calculate X_0^{29}, X_2^{29}, Y_1^{29}, and Y_3^{29}. We want to recover X_1^{29}. Since $X_0^{28} = Y_0^{28} = (X_3^{29})^L \mid (X_1^{29})^R$, $X_2^{28} = Y_2^{28} = (X_1^{29})^L \mid (X_3^{29})^R$, if we recover right 8 bits of X_0^{28} and left 8 bits of X_2^{28}, then we recover X_1^{29}. To recover right 8 bits of X_0^{28}, inject fault into bits corresponding to cube index of last 8 equations in Table 3. For recovering left 8 bits of X_2^{28}, inject fault into bits corresponding to cube index of first 8 equations in Table 3. Then X_1^{29} is recovered. And then we recover rk_{56} (K_0), because $X_1^{29} \oplus F(X_0^{29}) \oplus rk_{56} = Y_1^{29}$.

Step 5. This step is similar to Step 3 in analysis of Piccolo-80. We want to recover X_3^{28}. Given ciphertext C, we calculate X_0^{28}, X_2^{28}, Y_1^{28}, and Y_3^{28}. Since $X_0^{27} = Y_0^{27} = (X_3^{28})^L \mid (X_1^{28})^R$, $X_2^{27} = Y_2^{27} = (X_1^{28})^L \mid (X_3^{28})^R$, we recover left 8 bits of X_0^{27} and right 8 bits of X_2^{27}. To recover left 8 bits of X_0^{27}, inject fault into bits corresponding to cube index of first 8 equations in Table 3. For recovering right 8 bits of X_2^{27}, inject fault into bits corresponding to cube index of last 8 equations in Table 3. Then we recover X_3^{28}. And we recover rk_{55} (K_1) since $X_3^{28} \oplus F(X_2^{28}) \oplus rk_{55} = Y_3^{28}$.

Use the above 5 steps to recover all the master keys used for Piccolo-128. This needs the assumption that we inject fault into at most 4 bits in the same time. Assume that we analyze only Steps 1, 2, 3, and 4 such as Piccolo-80. Even though we analyze only Steps 1, 2, 3, and 4, since at least 32 bits of 128-bit master key are recovered, we recover all the master keys with less operations than brute-force attack. Table 5 is showing encryption complexity needed for recovering the master key of each assumption.

4.4. Improved Attack. The attacks described in Sections 4.2 and 4.3 are valid under the assumption that an adversary is able to control the injecting time and bit positions to inject fault and the number of bits of fault injection. However it is difficult to control bit positions where faults are injected. Therefore, in this section, we explain the way of attack under the assumption that an adversary is not able to control bit positions to inject faults. That is, an adversary is able to inject i bits of random faults into Piccolo-80 and Piccolo-128. The attack of Section 4.4 is similar to the attack of Sections 4.2 and 4.3. But this attack adds the process that determines fault position before each Step of Sections 4.2 and 4.3.

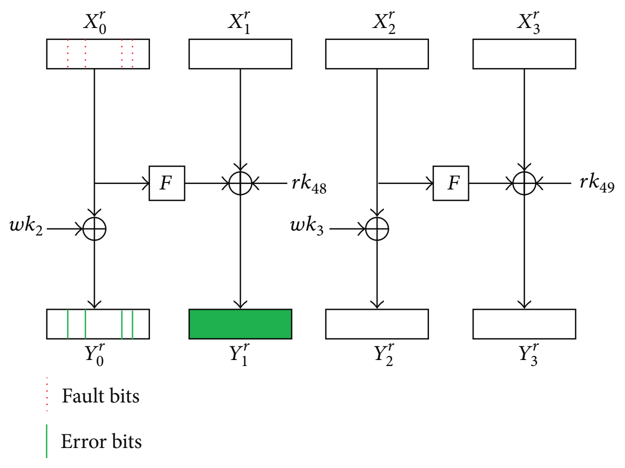

Figure 4: Determine position of fault injection for Case 1 ($i = 4$).

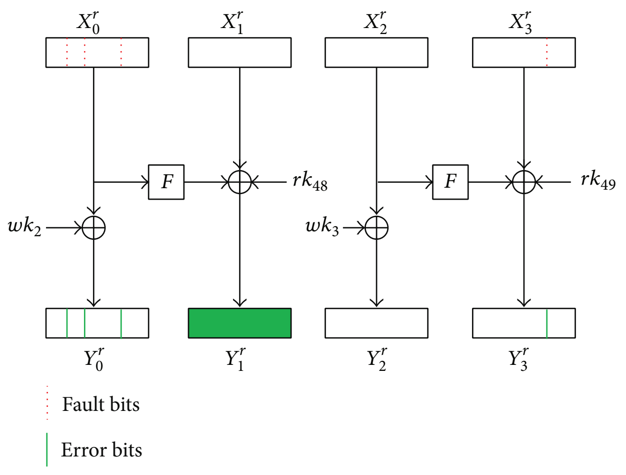

Figure 6: Determine position of fault injection for Case 3 ($i = 4$, $j = 1$).

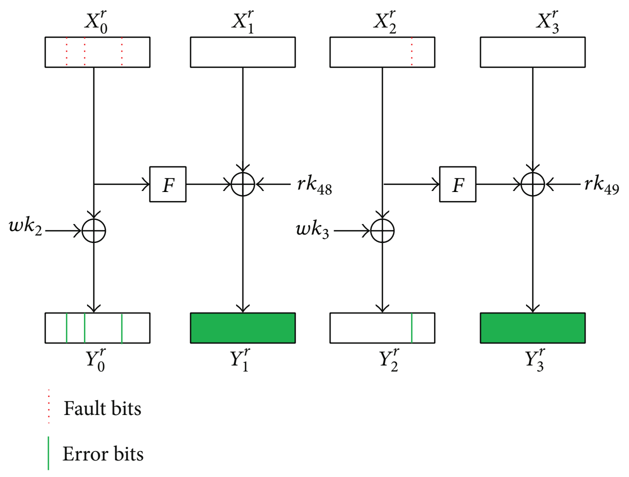

Figure 5: Determine position of fault injection for Case 2 ($i = 4$, $j = 1$).

Table 5: Attack complexity of Piccolo-128.

Assumption	Required fault	complexity
Step 1	$132 \approx 2^{7.04}$	2^{96}
Step 2	$264 \approx 2^{8.04}$	2^{64}
Step 3	$396 \approx 2^{8.63}$	2^{32}
Step 4	$479 \approx 2^{8.90}$	$2^{16.01}$
Step 5	$562 \approx 2^{9.13}$	$593.64 \approx 2^{9.21}$

Assume that we inject i bits of random fault into last round. In some cases, we determine fault position. Then, the following is how to determine position of fault injection for 3 cases. (Each case with 4 fault bits is described in Figures 4, 5, and 6).

Case 1. There are i bits error in $Y_0^r(Y_2^r)$ after fault injection. Since $X_0^r \bigoplus wk_2 = Y_0^r\ (X_2^r \bigoplus wk_3 = Y_2^r)$ and i fault bits, if $Y_0^r\ (Y_2^r)$ is different i bits from original ciphertext, then fault injection position in $X_0^r(X_2^r)$ must be same with changed i bits after fault injection in $Y_0^r\ (Y_2^r)$.

Case 2. There are j bits error in Y_0^r and $i - j$ bits error in Y_2^r after fault injection. In a similar way to Case 1, fault injection position in X_0^r must be same with changed j bits after fault injection in Y_0^r. And fault injection position in X_2^r must be same with changed i-j bits after fault injection in Y_2^r.

Case 3. There are $i - j$ bits error in $Y_0^r\ (Y_2^r)$ and j bits error in $Y_3^r(Y_1^r)$. And there is no error in $Y_2^r(Y_0^r)$. If there are j bits error in $Y_3^r(Y_1^r)$, then $X_2^r(X_0^r)$ or $X_3^r(X_1^r)$ is fault injected. Since there is no error in $Y_2^r(Y_0^r)$, fault injection position in $X_3^r(X_1^r)$ must be same with changed j bits after fault injection in $Y_3^r(Y_1^r)$. Furthermore, $X_0^r(X_2^r)$must be same with changed $i - j$ bits after fault injection in $Y_0^r\ (Y_2^r)$ since there are $i - j$ bits error in $Y_0^r\ (Y_2^r)$.

Therefore, we determine the position of fault injection for these 3 cases. Table 3 is the table of optimal cubes that are used for the attack in Sections 4.2 and 4.3. Therefore, there are many cubes except cubes in Table 3. Because there are too many cubes, we do not describe all cubes in this paper. For example, suppose that we get 16 ciphertexts for cube sum of $\{0, 1, 3, 4\}$. Then we calculate all of subcubes of $\{0, 1, 3, 4\}$. So, we use cubes in Table 6 and recover 1st, 2nd, 5th, and 7th bit of whitening key.

By the same method, we use sufficient cubes for recovering wk_2 and wk_3 of Piccolo-80 and Piccolo-128 and recover wk_2 and wk_3. Since we know wk_2 and wk_3, calculate intermediate values X_0^r, X_2^r for given ciphertext. Therefore, we inject faults into $(r - 1)$th round, determining position of fault injection by 3 cases that described this section. And we recover $(r - 1)$th round key. This process is the same with Step 2 in Section 4.2 and Step 2 in Section 4.3 except determining position of fault injection. By the same way, we recover all the master keys of Piccolo-80 and Piccolo-128 using Steps in Sections 4.2 and 4.3 with determining position of fault injection, respectively.

Table 6: Subcube of {0, 1, 3, 4}.

Cube index	Outbit (F_i)	Polyequation
0, 3	0	$x_2 + 1$
0, 4	8	$x_1 + x_5 + 1$
0, 1, 3, 4	2	x_5
0, 1, 3, 4	15	x_7

Table 7: Necessary positions of fault injections for Table 3.

Number of fault bits (Number of faults)	Necessary positions of fault injection
1 (11)	(0), (1), (4), (5), (6), (7), (8), (9), (11), (12), (13)
2 (27)	(0, 1), (0, 5), (0, 6), (0, 8), (0, 9), (0, 11), (0, 12), (1, 5), (1, 6), (4, 5), (4, 7), (4, 8), (4, 9), (4, 11), (5, 6), (5, 7), (5, 8), (5, 9), (5, 13), (6, 9), (6, 13), (7, 8), (8, 9), (8, 11), (8, 12), (9, 11), (11, 12)
3 (22)	(0, 1, 5), (0, 1, 6), (0, 5, 6), (0, 8, 9), (0, 8, 11), (0, 8, 12), (0, 9, 11), (0, 11, 12), (1, 5, 6), (4, 5, 7), (4, 5, 8), (4, 5, 9), (4, 7, 8), (4, 8, 9), (4, 8, 11), (4, 9, 11), (5, 6, 9), (5, 6, 13), (5, 7, 8), (5, 8, 9), (8, 9, 11), (8, 11, 12)
4 (6)	(0, 1, 5, 6), (0, 8, 9, 11), (0, 8, 11, 12), (4, 5, 7, 8), (4, 5, 8, 9), (4, 8, 9, 11)

The complexity of this attack depends on the complexity for finding ciphertext to recover round key. For Table 3, we need 66 ciphertexts. Table 7 is showing necessary positions of fault injection for cubes of Table 3.

For calculating complexity, we assume that an adversary always injects 4-bit random fault into the last three and five rounds for Piccolo-80 and Piccolo-128, respectively. Since an adversary injects exactly 4-bit fault, position of all fault bits has to match position that we want. This probability is $1/\binom{64}{4} \approx 2^{-19.277}$. That is, an adversary injects $2^{-19.277}$ times for each round and gets all ciphertexts that correspond to Table 3 for each round. Therefore, this attack on Piccolo-80 needs $3 \cdot 2^{19.277} \approx 2^{20.862}$ fault injections. Similarly, this attack on Piccolo-128 needs $5 \cdot 2^{19.277} \approx 2^{21.599}$ fault injections. The number of additional encryptions to recover the master key is negligible. Hence, our attack has the complexity of $2^{20.86}$ and $2^{21.60}$ encryptions with four bits of random fault injections for Piccolo-80 and Piccolo-128, respectively.

5. Conclusions

In this paper, we present the security weakness of Piccolo against fault analysis. Our attack fully exploits the structure of Piccolo, which is a Feistel network. We describe an attack for fault injection of target bit positions on Piccolo-80 and Piccolo-128. The master key of Piccolo-80 and Piccolo-128 is recovered by fault analysis by using cube attack with injecting faults $2^{8.44}$ and $2^{9.14}$, respectively. Our attack has the complexity of $2^{8.49}$ and $2^{9.21}$ encryptions for Piccolo-80 and Piccolo-128, respectively, which are practical complexities. And finally, an attack for *random* four bits fault injection for Piccolo-80 and Piccolo-128 is presented. This attack needs $2^{20.86}$ and $2^{21.60}$ encryptions with four bits of random fault injections for Piccolo-80 and Piccolo-128, respectively.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] D. Boneh, R. A. DeMillo, and R. J. Lipton, "On the importance of checking cryptographic protocols for faults," in *Advances in Cryptology—EUROCRYPT'97*, vol. 1233 of *Lecture Notes in Computer Science*, pp. 37–51, Springer, 1997.

[2] E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems," in *Advances in Cryptology—CRYPTO'97*, vol. 1294 of *Lecture Notes in Computer Science*, pp. 513–525, Springer, 1997.

[3] C. H. Kim and J.-J. Quisquater, "New differential fault analysis on AES key schedule: two faults are enough," in *Smart Card Research and Advanced Applications*, vol. 5189 of *Lecture Notes in Computer Science*, pp. 48–60, Springer, 2008.

[4] M. Tunstall, D. Mukhopadhyay, and S. Ali, "Differential fault analysis of the advanced encryption standard using a single fault," in *Information Security Theory and Practice. Security and Privacy of Mobile Devices in Wireless Communication*, vol. 6633 of *Lecture Notes in Computer Science*, pp. 224–233, Springer, 2011.

[5] W. Li, D. Gu, and J. Li, "Differential fault analysis on the ARIA algorithm," *Information Sciences*, vol. 178, no. 19, pp. 3727–3737, 2008.

[6] K. Jeong, Y. Lee, J. Sung, and S. Hong, "Differential fault analysis on block cipher SEED," *Mathematical and Computer Modelling*, vol. 55, no. 1-2, pp. 26–34, 2012.

[7] H. Chen, W. Wu, and D. Feng, "Differential fault analysis on CLEFIA," in *Information and Communications Security*, vol. 4861 of *Lecture Notes in Computer Science*, pp. 284–295, Springer, 2007.

[8] K. Jeong and C. Lee, "Differential fault analysis on block cipher LED-64," in *Future Information Technology, Application, and Service*, vol. 1, pp. 747–755, Springer, 2012.

[9] K. Jeong, "Security analysis of block cipher Piccolo suitable for wireless sensor networks," *Peer-to-Peer Networking and Applications*, 2013.

[10] S. Li, D. Gu, Z. Ma, and Z. Liu, "Fault analysis of the Piccolo block cipher," in *Proceedings of the 8th International Conference on Computational Intelligence and Security (CIS '12)*, pp. 482–486, 2012.

[11] H. Yoshikawa, M. Kaminaga, A. Shikoda, and T. Suzuki, "Round addition DFA on 80-bit Piccolo and TWINE," *IEICE Transactions on Information and Systems*, vol. E96-D, no. 9, pp. 2031–2035, 2013.

[12] F. Zhang, X. Zhao, S. Guo, T. Wang, and Z. Shi, "Improved algebraic fault analysis: a case study on Piccolo and applications to other lightweight block ciphers," in *Proceedings of the 4th International Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE '13)*, vol. 7864 of *Lecture Notes in Computer Science*, pp. 62–79, Springer, 2013.

[13] N. Bagheri, R. Ebrahimpour, and N. Ghaedi, "New differential fault analysis on PRESENT," *EURASIP Journal on Advances in Signal Processing*, vol. 2013, article 145, 10 pages, 2013.

[14] W. Y. Zhang, F. Liu, X. Liu, and S. Meng, "Differential fault analysis and meet-in-the-middle attack on the block cipher KATAN32," *Journal of Shanghai Jiaotong University (Science)*, vol. 18, no. 2, pp. 147–152, 2013.

[15] I. Dinur and A. Shamir, "Cube attacks on tweakable black boxp polynomials," in *Advances in Cryptology—EUROCRYPT 2009*, vol. 5479 of *Lecture Notes in Computer Science*, pp. 278–299, Springer, 2009.

[16] S. F. Abdul-Latip, M. R. Reyhanitabar, W. Susilo, and J. Seberry, "Fault analysis of the KATAN family of block ciphers," in *Information Security Practice and Experience*, vol. 7232 of *Lecture Notes in Computer Science*, pp. 319–336, Springer, 2012.

[17] I. Dinur and A. Shamir, "Applying cube attacks to stream ciphers in realistic scenarios," *Cryptography and Communications*, vol. 4, no. 3-4, pp. 217–232, 2012.

[18] K. Shibutani, T. Isobe, H. Hiwatari, A. Mitsuda, T. Akishita, and T. Shirai, "Piccolo: an ultra-lightweight blockcipher," in *Proceedings of the 13th International Workshop on Cryptographic Hardware and Embedded Systems (CHES '11)*, vol. 6917 of *Lecture Notes in Computer Science*, pp. 342–357, 2011.

[19] Y. Wang, W. Wu, and X. Yu, "Biclique cryptanalysis of reduced-round Piccolo block cipher," in *Information Security Practice and Experience*, vol. 7232 of *Lecture Notes in Computer Science*, pp. 337–352, Springer, 2012.

[20] J. Song, K. Lee, and H. Lee, "Biclique cryptanalysis on lightweight block cipher: HIGHT and Piccolo," *International Journal of Computer Mathematics*, vol. 90, no. 12, pp. 1–17, 2013.

[21] J.-P. Aumasson, I. Dinur, W. Meier, and A. Shamir, "Cube testers and key recovery attacks on reduced-round MD6 and trivium," in *Fast Software Encryption*, vol. 5665 of *Lecture Notes in Computer Science*, pp. 1–22, Springer, 2009.