



Article

Differential Fault and Algebraic Equation Combined Analysis on PICO

Linxi Ding ¹, Hongxin Zhang ^{1,2,*}, Jun Xu ³, Xing Fang ⁴ and Yejing Wu ¹¹ School of Electronic Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China; dinglinxi@bupt.edu.cn (L.D.); yejingwu@bupt.edu.cn (Y.W.)² Beijing Key Laboratory of Work Safety Intelligent Monitoring, Beijing University of Posts and Telecommunications, Beijing 100876, China³ Beijing Institute of Spacecraft System Engineering, Beijing 100094, China; xujun_bupt@163.com⁴ Beijing Institute of Computer Technology and Applications, Beijing 100039, China; fancy_t@bupt.cn

* Correspondence: hongxinzhang@bupt.edu.cn

Abstract: In modern information technology, research on block cipher security is imperative. Concerning the ultra lightweight block cipher PICO, there has been only one study focused on recovering its complete master key, with a large search space of 2^{64} , and no fault analysis yet. This paper proposes a new fault analysis approach, combining differential fault and algebraic equation techniques. It achieved the recovery of PICO's entire master key with 40 faults in an average time of 0.57 h. S-box decomposition was utilized to optimize our approach, reducing the time by a remarkable 75.83% under the identical 40-fault condition. Furthermore, PICO's complete master key could be recovered with 28 faults in an average time of 0.78 h, indicating a significant 2^{37} reduction in its search space compared to the previous study. This marks the first fault analysis on PICO. Compared to conventional fault analysis methods DFA (differential fault analysis) and AFA (algebraic fault analysis), our approach outperforms in recovering PICO's entire master key, highlighting the cruciality of key expansion complexity in block cipher security. Therefore, our approach could serve to recover master keys of block ciphers with comparably complicated key expansions, and production of more secure block ciphers could result.

Keywords: PICO; block cipher; differential fault; algebraic equation; S-box decomposition**MSC:** 68P25

Citation: Ding, L.; Zhang, H.; Xu, J.; Fang, X.; Wu, Y. Differential Fault and Algebraic Equation Combined Analysis on PICO. *Mathematics* **2024**, *12*, 700. <https://doi.org/10.3390/math12050700>

Academic Editor: Jonathan Blackledge

Received: 30 January 2024

Revised: 19 February 2024

Accepted: 25 February 2024

Published: 28 February 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

With the continuous development and widespread adoption of modern information technology, information security has become a crucial topic. Block ciphers implemented on chips provide reliable means to ensure information security. As chips are becoming more compact, lightweight block ciphers have been widely used due to their efficient performance in environments with limited computational and storage resources, such as PRESENT [1], SKINNY [2], GIFT [3], SIMON [4], SPECK [4], LBlock [5], RECTANGLE [6], etc. In this case, it is essential to conduct security analysis on various block ciphers, with a particular focus on their performance in the face of key decryption.

PICO is an ultra lightweight block cipher presented by Bansod Gaurav et al. in 2016 that performs well on both software and hardware platforms. Compared to other block ciphers, PICO has compact key scheduling which leads to fewer gate counts and lower memory size. Along with its lower power consumption, this makes it well-suited for small-scale and energy-constrained applications, such as RFID tags, wireless sensors, and other IoT devices. Moreover, its strong S-box ensures robustness in the design and excellent avalanche effect, enabling it to resist attacks with higher probability. For detailed data comparisons between PICO and other block ciphers in terms of gate count, memory usage,

power consumption, etc., please refer to [7]. Given these strengths of PICO over other block ciphers, conducting an analysis on it holds substantial significance.

Studies focused on PICO are as follows. In 2016, designers [7] of PICO claimed that it can effectively resist linear attack, differential attack, biclique attack and algebraic attack, but there was no specific experimental data available to support these theoretical analyses. In 2017, refs. [8,9] constructed a 7-round multidimensional zero-correlation linear distinguisher for PICO and recovered 50 bits of subkeys against 10-round PICO. In 2019, ref. [10] studied the optimal differential trails of PICO and presented one for 21 rounds with probability 2^{-63} , suggesting its potential use for key recovery. In 2020, ref. [11] searched a 10-round integral distinguisher for PICO, utilized its 9-round integral distinguisher to perform 11-round key recovery through 16 rounds of filtering, and recovered a total of 128 bits for the 10th and 11th round subkeys with a data complexity of $2^{63.46}$. In 2021, ref. [12] used the same integral distinguisher as in ref. [11] to recover a total of 52 bits for the 11th round subkeys through 16 rounds of filtering with a data complexity of $2^{63.17}$. Ref. [13] also searched a 10-round integral distinguisher for PICO with a data complexity of 2^{63} . Ref. [14] found a 7-round impossible differential distinguisher, which is the longest impossible differential distinguisher found for PICO. In 2023, ref. [15] presented optimal differential trails with their probabilities for rounds one to 22 of PICO, as well as 21-round and 22-round differential distinguishers with their probabilities. It also constructed key recovery on the 26-round PICO by employing a 21-round differential distinguisher and on any round of PICO based on related-key distinguishers for any round with probability one. Ref. [16] identified four 21-round differential distinguishers with probability 2^{-63} for PICO, three of which differ from the one in ref. [10]. Additionally, three 20-round linear distinguishers with correlation 2^{-30} were uncovered for the first time, establishing them as the longest linear distinguishers for PICO. These findings indicated that PICO has strong resistance to both differential and linear attacks.

In a comprehensive review of the aforementioned research outcomes related to the key recovery of PICO, refs. [8,9,11,12] accomplished the recovery of its partial subkeys, without exploring the recovery of its master key. With ref. [15] being the only study addressing the recovery of PICO's master key, it obtained half of the entire 128-bit master key through exhaustive search, which demands significant computational resource. Moreover, none of the studies has provided a detailed fault analysis on PICO, which is an effective method for recovering the master key of a block cipher.

Conventional fault analysis methods, such as differential fault analysis (DFA) and algebraic fault analysis (AFA), are commonly employed. The concept of DFA was formally presented in 1997 [17] by Biham Eli and Shamir Adi, and has been utilized widely since then. It has been applied to block ciphers such as GIFT [18], PRESENT [18], SKINNY [19], LBlock [20], etc., successfully recovering their master keys. AFA was first proposed by Courtois NT et al. in 2010 [21] to break the key of DES and its general framework for lightweight block ciphers was introduced by Zhang Fan et al. in 2016 [22]. It has been successfully employed to recover the master keys of LBlock [22], PRESENT [22], SKINNY [23], etc. However, there is currently no research on the application of these two methods on PICO.

In this paper, we propose a new fault analysis approach targeting the characteristics of PICO's key expansion. By combining both differential fault and algebraic equation techniques, we successfully recovered the complete master key of PICO. Furthermore, through the application of S-box decomposition, we effectively enhanced solving efficiency and achieved notable advancements. In comparison to the previous study, the search space of PICO's master key was significantly reduced. To our knowledge, this represents the first fault analysis conducted on PICO. Through comparisons with conventional fault analysis methods DFA and AFA, our combined analysis approach excelled them with superior solving performance, thereby revealing the crucial impact of key expansion complexity on block cipher security.

The paper is organized as follows. Section 2 introduces PICO cipher. Section 3 analyzes the characteristics of PICO's key expansion, proposes our differential fault and algebraic equation combined analysis approach, and provides its application results on PICO. Section 4 applies S-box decomposition to enhance our combined analysis approach and compares its application results with the original ones. Section 5 compares our combined analysis approach with conventional fault analysis methods DFA and AFA in terms of their applications on PICO, thus highlighting our advantages. Section 6 summarizes our achievements and outlines potential avenues for future research.

2. Description of PICO Cipher

PICO is an ultra lightweight block cipher based on the SPN (substitution-permutation network) structure. It supports a 64-bit plaintext and a 128-bit master key, and produces a 64-bit ciphertext through 32 rounds of encryption and 1 round of whitening. Figure 1 shows the structure of PICO cipher. The encryption process of PICO is described in Algorithm 1, where P denotes the 64-bit plaintext, K denotes the 128-bit master key, C denotes the 64-bit ciphertext, K^j ($j = 0$ to 32) denotes a 64-bit subkey, p^i ($i = 0$ to 63) denotes a 1-bit binary digit, and symbol \parallel is used to concatenate bits with the left being the most significant bit and the right being the least significant bit.

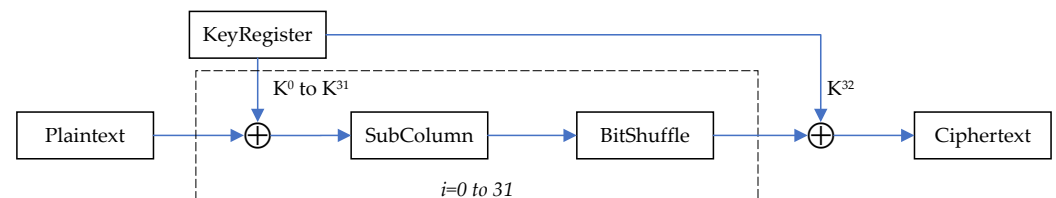


Figure 1. The structure of PICO.

Algorithm 1: Encryption of PICO

Input: P, K
Output: C

- 1 $P = p^{63} \parallel p^{62} \parallel \dots \parallel p^0$;
- 2 Generate subkeys K^0, K^1, \dots, K^{32} ;
- 3 $X^0 \leftarrow P$;
- 4 **for** $i = 0; i < 32; i++$ **do**
- 5 $S_{in}^i \leftarrow \text{AddRoundKey}(X^i, K^i)$;
- 6 $S_{out}^i \leftarrow \text{SubColumn}(S_{in}^i)$;
- 7 $P_{out}^i \leftarrow \text{BitShuffle}(S_{out}^i)$;
- 8 $X^{i+1} \leftarrow P_{out}^i$;
- 9 $C \leftarrow \text{AddRoundKey}(X^{32}, K^{32})$;
- 10 **return** C

- **Key Expansion**
 PICO's key expansion is motivated from that of SPECK cipher [4], according to its designers [7]. The 128-bit master key is used to generate 33 subkeys, each of size 64-bit. The exact process of key expansion is presented in Algorithm 2, where L_1^j and L_2^j ($j = 0$ to 31) are intermediate variables, k^i ($i = 0$ to 127) denotes a 1-bit binary digit, symbol \oplus represents bitwise exclusive-OR operation, $RCS(X, n)$ represents right circular shift of variable X by n bits, and $LCS(X, n)$ represents left circular shift of variable X by n bits.
- **AddRoundKey**
 Perform a bit-by-bit XOR operation between the 64-bit round input and its corresponding 64-bit subkey.

- *SubColumn*
Let $S_{in} = x^{63} \parallel x^{62} \parallel \dots \parallel x^0$ denote the 64-bit input of *SubColumn*, perform the substitution $(y^{48+i} \parallel y^{32+i} \parallel y^{16+i} \parallel y^i) \leftarrow \text{SubColumn}(x^{48+i} \parallel x^{32+i} \parallel x^{16+i} \parallel x^i)$ according to Table 1, where i ranges from 0 to 15, and obtain the 64-bit output of *SubColumn* as $S_{out} = y^{63} \parallel y^{62} \parallel \dots \parallel y^0$. This operation resembles the one in RECTANGLE cipher [6], as explained by PICO's designers [7].

Table 1. S-box of PICO.

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

- *BitShuffle*
Perform a bitwise permutation on the 64-bit output of *SubColumn* according to Table 2.

Table 2. P-box of PICO.

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

Algorithm 2: Key expansion of PICO

Input: K
Output: K^0, K^1, \dots, K^{32}
1 $K = k^{127} \parallel k^{126} \parallel \dots \parallel k^0$;
2 $K^0 = k^{63} \parallel k^{62} \parallel \dots \parallel k^0$;
3 $L_1^0 = k^{127} \parallel k^{126} \parallel \dots \parallel k^{64}$;
4 **for** $j = 0; j < 32; j++$ **do**
5 $L_2^j \leftarrow K^j \oplus \text{RCS}(L_1^j, 3) \oplus L_1^j$;
6 $K^{j+1} \leftarrow L_2^j \oplus \text{LCS}(K^j, 7) \oplus j$;
7 $L_1^{j+1} \leftarrow L_2^j$;
8 **return** K^0, K^1, \dots, K^{32}

3. Proposed Combined Analysis on PICO

This chapter unfolds as follows. Section 3.1 explores the characteristics of PICO's master key, emphasizing that two distinct master keys can generate an identical set of subkeys. Section 3.2 demonstrates the process of solving subkeys with differential faults. Section 3.3 analyzes the reverse key expansion of PICO and introduce algebraic equations. Section 3.4 illustrates the experimental results obtained from our combined analysis on PICO.

3.1. Characteristics of PICO's Master Key

Assuming all subkeys K^0, K^1, \dots, K^{32} are known, L_1^0 needs to be determined in order to solve for the master key K . The solution for L_1^0 is given in Equation (1)

$$L_1^0 \oplus \text{RCS}(L_1^0, 3) = K^0 \oplus K^1 \oplus \text{LCS}(K^0, 7). \quad (1)$$

Let $L_1^0 = k^{127} \parallel k^{126} \parallel \dots \parallel k^{64}$, where k^{127} denotes the most significant bit and k^{64} denotes the least significant bit, then Equation (1) can be expressed as Equation (2)

$$\begin{cases} k^{64+i} \oplus k^{67+i} = c^i, & i = 0 \text{ to } 60 \\ k^{64+i} \oplus k^{3+i} = c^i, & i = 61 \text{ to } 63, \end{cases} \quad (2)$$

where c^i ($i = 0$ to 63) is a known constant, and $k^{64+i} \in \{0, 1\}$. Therefore, the solutions to Equation (1) are either non-existent or consist of two solutions, with these two solutions being bitwise complements in a 64-bit binary form. Since the master key must exist, there is at least one solution, resulting in Equation (1) having two solutions.

In consequence, when all subkeys of PICO are determined, there exist two master keys capable of generating all correct subkeys, hence both of these master keys serving identical roles and bringing about equivalent effects in the encryption process. Based on this, in the subsequent solving steps in this paper, both of the master keys are considered correct, and obtaining either one is considered a successful solution.

3.2. Solving Subkeys with Differential Faults

The brief strategy for efficiently solving subkeys with differential faults involves several key steps. Initially, a fault is injected into the input of the *SubColumn* operation in the preceding round for the target subkey. Subsequently, the output of the *SubColumn* operation in that specific round is constrained based on the differential of each S-box output. Following this, faults are injected multiple times, and the results are intersected to further refine possible output values of the *SubColumn* operation, until that is exclusively determined. Ultimately, the desired subkey can be deduced through a systematic analysis of the round operations.

Firstly, we introduce the necessary tools. The inverse versions of Tables 1 and 2 serve as the lookup tables for the inverse S-box and P-box of PICO. Algorithm 3 details the computation process for establishing correspondences of the inputs, outputs and their differentials of PICO's S-box, resulting in associations between the differential outputs of the S-box and its potential outputs, as elaborated in Table 3.

Algorithm 3: Generation of correspondence relevant to differential values of PICO's S-box

```

1 for  $\Delta X = 0000$  to  $1111$  do
2   for  $X = 0000$  to  $1111$  do
3      $Y \leftarrow \text{SubColumn}(X)$ ;
4      $X^* \leftarrow X \oplus \Delta X$ ;
5      $Y^* \leftarrow \text{SubColumn}(X^*)$ ;
6      $\Delta Y \leftarrow Y \oplus Y^*$ ;

```

Table 3. Possible output of PICO's S-box by its differential output.

ΔY	Possible Y
3	0, 1, 2, 3, 8, 9, A, B, C, F
4	A, E
5	1, 4, 8, 9, C, D
6	3, 5
7	0, 1, 2, 5, 6, 7, 8, F
8	0, 8
9	4, 5, 6, 7, C, D, E, F
A	4, E
B	1, A
C	0, 7, B, C
D	2, 3, 6, B, E, F
E	3, 7, 9, D
F	2, 4, 5, 6, 9, A, B, D

Next, we delineate the process of solving subkeys with differential faults, where the subkeys are sequentially determined in reverse order, starting from K^{32} down to K^0 . In each round of *SubColumn* in PICO, a total of 16 S-boxes are utilized, and the output of each S-box needs to be determined according to Table 3. Only when the differential output ΔY of an S-box is non-zero can relevant information about that S-box be obtained. To leverage the substitution characteristics of the *SubColumn* operation, we adopt double-byte faults for the following reasons. When a fault is injected into the *SubColumn* input $X = x^{63} \parallel x^{62} \parallel \dots \parallel x^0$ in a particular round, for the four sets of bits $x^{63} \parallel x^{62} \parallel \dots \parallel x^{48}$, $x^{47} \parallel x^{46} \parallel \dots \parallel x^{32}$, $x^{31} \parallel x^{30} \parallel \dots \parallel x^{16}$, and $x^{15} \parallel x^{14} \parallel \dots \parallel x^0$, we note that each set of 16 bits distinctly impacts all 16 different S-boxes, and a double-byte fault can precisely accomplish a 0-1 flip for each bit among the 16 bits in one of these sets, which induces changes in the input to every S-box among the 16 different S-boxes. According to Table 1, changes in the input of an S-box result in definite changes to its output, signifying that the differential output ΔY of each S-box is non-zero, and therefore pertinent information for each S-box can be acquired. Moreover, up to four random and non-repeating fault injection in each round can cover all 64 bits, ensuring that every bit in each of the 4-bit S-box input undergoes a bit flip for all 16 S-boxes. The correspondence between S-box outputs and its potential differential values of the output can be derived from the reverse version of Table 3, and it indicates that to determine the output of an S-box, a maximum of four differential output is required. Hence, at most four random and non-repeating fault injections are sufficient to uniquely determine the correct output value of an S-box. In short, the application of double-byte faults ensures that valuable information about each S-box is obtained after each fault injection, and consequently, this minimizes the number of faults required to solve a subkey, with a maximum of four. Given the resemblance of the *SubColumn* operation in PICO to that in RECTANGLE, this double-byte differential fault model is applicable to block ciphers sharing comparable *SubColumn* properties.

Algorithm 4 outlines the detailed process of solving subkeys through the injection of differential faults, where X , X^* , and ΔX respectively denote the correct value, erroneous value, and their differential value before and after fault injection, *FaultInjection* represents injecting a double-byte fault that is randomly selected and non-repeating from previous instances and continuing encrypting to generate an erroneous ciphertext, and *TableLookup* represents the result set obtained from consulting Table 3.

3.3. Reverse Key Expansion and Representation of Algebraic Equations

Implementing the approach outlined in Section 3.2 allows for the extraction of one subkey with at most every four faults. This implies that a maximum of 132 faults can reveal all 33 subkeys for the entire rounds, and the master key can be obtained by referring to Section 3.1. However, this method requires a high number of faults. To address this issue, we propose an approach that combines the application of differential faults with algebraic equation solving. Our primary strategy comprises the following steps. At the outset, we utilize differential faults to deduce a subset of subkeys. Afterward, we restrict the potential values of the remaining subkeys and the master key according to reverse key expansion. Subsequently, we apply algebraic equations to achieve the reverse deduction. Finally, we filter the derived master keys based on the condition that the correct ciphertext can be generated through encryption when using the master key for the correct plaintext. This process continues until a qualifying master key is identified.

When a subset of subkeys is already solved, the procedure for deducing the rest of subkeys through reverse key expansion is presented in Algorithm 5. This reveals that when only one specific K^{j+1} ($0 \leq j \leq 31$) is known, there exist at most 2^{65} sets of possible values for K^j and K^{j-1} , and at most 2^{j-i+64} possible values for K^i ($0 \leq i \leq j-2$), and when both K^{j+1} and K^j are known, there are up to 2^{i-j+2} possible values for K^i ($0 \leq i \leq j-1$). Considering that subkeys are sequentially determined in reverse order, as explained in Section 3.2, when only K^{32} is known, there exist at most 2^{96} possible master keys, and when $K^{32} \dots K^j$ ($0 \leq j \leq 31$) are known, up to 2^{j+1} possible master keys can be obtained.

Algorithm 4: Solving for $K^{32}, K^{31} \dots K^0$ with Differential Faults

```

1  Generate random  $P, K$ , where  $K$  is unknown to the analyst;
2   $C \leftarrow \text{Enc}(P, K)$ ;
3  //solving for  $K^{32}$ 
4  for  $num = 0; num < 4; num++$  do
5      if  $|Set_{S_{out}^{31}}| > 1$  then
6           $C^* \leftarrow \text{FaultInjection}(\text{Enc}(P, K), \text{Fault}_{num}^{S_{in}^{31}})$ ;
7           $\Delta C \leftarrow C \oplus C^*$ ;
8           $\Delta P_{out}^{31} \leftarrow \Delta C$ ;
9           $\Delta S_{out}^{31} \leftarrow \text{BitShuffle}^{-1}(\Delta P_{out}^{31})$ ;
10          $Set_{S_{out}^{31}} \leftarrow Set_{S_{out}^{31}} \cap \text{TableLookup}(\Delta S_{out}^{31})$ ;
11  $S_{out}^{31} \in Set_{S_{out}^{31}}$ ;
12  $P_{out}^{31} \leftarrow \text{BitShuffle}(S_{out}^{31})$ ;
13  $K^{32} \leftarrow P_{out}^{31} \oplus C$ ;
14 //solving for  $K^j (j = 31 \text{ to } 1)$ 
15  $S_{in}^j \leftarrow \text{SubColumn}^{-1}(S_{out}^j)$ ;
16 for  $num = 0; num < 4; num++$  do
17     if  $|Set_{S_{out}^{j-1}}| > 1$  then
18          $C^* \leftarrow \text{FaultInjection}(\text{Enc}(P, K), \text{Fault}_{num}^{S_{in}^{j-1}})$ ;
19          $P_{out}^{31*} \leftarrow K^{32} \oplus C^*$ ;
20          $S_{out}^{31*} \leftarrow \text{BitShuffle}^{-1}(P_{out}^{31*})$ ;
21          $S_{in}^{31*} \leftarrow \text{SubColumn}^{-1}(S_{out}^{31*})$ ;
22         for  $i = 30; i > j - 1; i--$  do
23              $P_{out}^{i*} \leftarrow K^{i+1} \oplus S_{in}^{i+1*}$ ;
24              $S_{out}^{i*} \leftarrow \text{BitShuffle}^{-1}(P_{out}^{i*})$ ;
25              $S_{in}^{i*} \leftarrow \text{SubColumn}^{-1}(S_{out}^{i*})$ ;
26          $\Delta S_{in}^j \leftarrow S_{in}^j \oplus S_{in}^{j*}$ ;
27          $\Delta P_{out}^{j-1} \leftarrow \Delta S_{in}^j$ ;
28          $\Delta S_{out}^{j-1} \leftarrow \text{BitShuffle}^{-1}(\Delta P_{out}^{j-1})$ ;
29          $Set_{S_{out}^{j-1}} \leftarrow Set_{S_{out}^{j-1}} \cap \text{TableLookup}(\Delta S_{out}^{j-1})$ ;
30  $S_{out}^{j-1} \in Set_{S_{out}^{j-1}}$ ;
31  $P_{out}^{j-1} \leftarrow \text{BitShuffle}(S_{out}^{j-1})$ ;
32  $K^j \leftarrow P_{out}^{j-1} \oplus S_{in}^j$ ;
33 //solving for  $K^0$ 
34  $S_{in}^0 \leftarrow \text{SubColumn}^{-1}(S_{out}^0)$ ;
35  $K^0 \leftarrow P \oplus S_{in}^0$ ;
36 return  $K^{32}, K^{31} \dots K^0$ 

```

Algorithm 5: Solving for K^i ($0 \leq i \leq 31$) using reverse key expansion

```

1 //scenario 1: solving for  $K^i$  ( $i = j$  to 0) while  $K^{j+1}$  ( $0 \leq j \leq 31$ ) is solved
2  $K^j \oplus LCS(K^j, 7) \oplus L_1^j \oplus RCS(L_1^j, 3) = K^{j+1} \oplus j$ ;
3  $L_2^{j-1} = L_1^j$ ;
4  $K^{j-1} = RCS((K^j \oplus L_2^{j-1} \oplus (j-1)), 7)$ ;
5 for  $i = j-2; i \geq 0; i--$  do
6    $L_1^{i+1} \oplus RCS(L_1^{i+1}, 3) = K^{i+1} \oplus L_2^{i+1}$ ;
7    $L_2^i = L_1^{i+1}$ ;
8    $K^i = RCS((K^{i+1} \oplus L_2^i \oplus i), 7)$ ;
9 //scenario 2: solving for  $K^i$  ( $i = j-1$  to 0) while  $K^{j+1}$  and  $K^j$  ( $0 \leq j \leq 31$ ) are
  solved
10  $L_2^j = K^{j+1} \oplus LCS(K^j, 7) \oplus j$ ;
11 for  $i = j-1; i \geq 0; i--$  do
12    $L_1^{i+1} \oplus RCS(L_1^{i+1}, 3) = K^{i+1} \oplus L_2^{i+1}$ ;
13    $L_2^i = L_1^{i+1}$ ;
14    $K^i = RCS((K^{i+1} \oplus L_2^i \oplus i), 7)$ ;

```

Algebraic equations are introduced due to the following two reasons. Firstly, the reverse key expansion involves the solving of Equation (3)

$$L_1^{i+1} \oplus RCS(L_1^{i+1}, 3) = K^{i+1} \oplus L_2^{i+1}. \quad (3)$$

Moreover, the derived master key must undergo additional constraints, ensuring that its use for encrypting the correct plaintext produces the correct ciphertext. The construction of algebraic equations is detailed below, comprising representations for assignment, reverse key expansion, and forward encryption, and the solver we employ is CryptoMiniSat [24], with the specific version being CryptoMiniSat 5.8.0.

- **Equations for assignment**
Include the constant 1, subkeys $K^{32} \dots K^j$ ($0 \leq j \leq 32$) solved with differential faults, the correct plaintext P , and the correct ciphertext C , for a total of $(2241 - 64j)$ equations.
- **Equations for reverse key expansion**
For each round of reverse key expansion, three sets of equations are included, as indicated by Equations (4)–(6).

$$j^i = 0 \text{ or } j^i = 1, i = 0 \text{ to } 63 \quad (4)$$

$$\begin{cases} k^{j,i+57} \oplus k^{j+1,i} \oplus l_2^{j,i} \oplus j^i = 0, & i = 0 \text{ to } 6 \\ k^{j,i-7} \oplus k^{j+1,i} \oplus l_2^{j,i} \oplus j^i = 0, & i = 7 \text{ to } 63 \end{cases} \quad (5)$$

$$\begin{cases} l_1^{j,i} \oplus l_1^{j,i+3} \oplus k^{j,i} \oplus l_2^{j,i} = 0, & i = 0 \text{ to } 60 \\ l_1^{j,i} \oplus l_1^{j,i-61} \oplus k^{j,i} \oplus l_2^{j,i} = 0, & i = 61 \text{ to } 63 \end{cases} \quad (6)$$

In total there are 6144 equations for all 32 rounds.

- **Equations for forward encryption**
AddRoundKey: For the r -th ($r = 1$ to 33) round, let $x^{r,63} \parallel x^{r,62} \parallel \dots \parallel x^{r,0}$ denote the input of *AddRoundKey*, $k^{r,63} \parallel k^{r,62} \parallel \dots \parallel k^{r,0}$ denote the subkey, and $y^{r,63} \parallel y^{r,62} \parallel \dots \parallel y^{r,0}$ denote the output of *AddRoundKey*. Equation (7) signifies the *AddRoundKey* operation in each round.

$$x^{r,i} \oplus k^{r,i} \oplus y^{r,i} = 0, i = 0 \text{ to } 63 \quad (7)$$

In total there are 2112 equations for all 33 rounds.

SubColumn: The S-box needs to be expressed in the form of algebraic equations [25], and the computed representation of PICO's S-box is as depicted in Equation (8), where $x_3 \parallel x_2 \parallel x_1 \parallel x_0$ and $y_3 \parallel y_2 \parallel y_1 \parallel y_0$ respectively denote the input and output of an S-box, x_3 and y_3 denote the most significant bit, and x_0 and y_0 denote the least significant bit.

$$\begin{cases} y_0 = 1 + x_0 + x_1 + x_2 + x_3 + x_1x_3 \\ y_1 = x_0 + x_2 + x_3 + x_0x_1 + x_0x_2 + x_1x_2x_3 \\ y_2 = x_1 + x_2 + x_0x_3 + x_2x_3 \\ y_3 = x_3 + x_0x_1 + x_0x_2 + x_0x_3 + x_1x_2 + x_0x_1x_3 \end{cases} \quad (8)$$

For the i -th ($i = 0$ to 15) S-box in the r -th ($r = 1$ to 32) round, let $x_3^{r,i} \parallel x_2^{r,i} \parallel x_1^{r,i} \parallel x_0^{r,i}$ denote the input to the S-box, and $y_3^{r,i} \parallel y_2^{r,i} \parallel y_1^{r,i} \parallel y_0^{r,i}$ denote the output of the S-box. Equation (9) signifies the *SubColumn* operation in each round.

$$\begin{cases} 1 + x_0^{r,i} + x_1^{r,i} + x_2^{r,i} + x_3^{r,i} + x_1^{r,i}x_3^{r,i} + y_0^{r,i} = 0 \\ x_0^{r,i} + x_2^{r,i} + x_3^{r,i} + x_0^{r,i}x_1^{r,i} + x_0^{r,i}x_2^{r,i} + x_1^{r,i}x_2^{r,i}x_3^{r,i} + y_1^{r,i} = 0 \\ x_1^{r,i} + x_2^{r,i} + x_0^{r,i}x_3^{r,i} + x_2^{r,i}x_3^{r,i} + y_2^{r,i} = 0 \\ x_3^{r,i} + x_0^{r,i}x_1^{r,i} + x_0^{r,i}x_2^{r,i} + x_0^{r,i}x_3^{r,i} + x_1^{r,i}x_2^{r,i} + x_0^{r,i}x_1^{r,i}x_3^{r,i} + y_3^{r,i} = 0 \end{cases} \quad (9)$$

Meanwhile, for each S-box, there exist six quadratic variables $x_0^{r,i}x_1^{r,i}, x_0^{r,i}x_2^{r,i}, x_0^{r,i}x_3^{r,i}, x_1^{r,i}x_2^{r,i}, x_1^{r,i}x_3^{r,i}, x_2^{r,i}x_3^{r,i}$ and two cubic variables $x_0^{r,i}x_1^{r,i}x_3^{r,i}, x_1^{r,i}x_2^{r,i}x_3^{r,i}$, each requiring representation through additional equations due to the nature of CryptoMiniSat v5.8.0.

For a quadratic variable, let $m_{a,b}^{r,i} = x_a^{r,i}x_b^{r,i}$, and the representation is given in Equation (10).

$$\begin{cases} x_a^{r,i} \vee \neg m_{a,b}^{r,i} = 1 \\ x_b^{r,i} \vee \neg m_{a,b}^{r,i} = 1 \\ m_{a,b}^{r,i} \vee \neg x_a^{r,i} \vee \neg x_b^{r,i} = 1 \end{cases} \quad (10)$$

For a cubic variable, let $m_{a,b,c}^{r,i} = x_a^{r,i}x_b^{r,i}x_c^{r,i}$, and the representation is given in Equation (11).

$$\begin{cases} x_a^{r,i} \vee \neg m_{a,b,c}^{r,i} = 1 \\ x_b^{r,i} \vee \neg m_{a,b,c}^{r,i} = 1 \\ x_c^{r,i} \vee \neg m_{a,b,c}^{r,i} = 1 \\ m_{a,b,c}^{r,i} \vee \neg x_a^{r,i} \vee \neg x_b^{r,i} \vee \neg x_c^{r,i} = 1 \end{cases} \quad (11)$$

In total there are 15,360 equations for all 32 rounds.

BitShuffle: For the r -th ($r = 1$ to 32) round, let $x^{r,63} \parallel x^{r,62} \parallel \dots \parallel x^{r,0}$ denote the input of *BitShuffle*, and $y^{r,63} \parallel y^{r,62} \parallel \dots \parallel y^{r,0}$ denote the output of *BitShuffle*. Equation (12) signifies the *BitShuffle* operation in each round.

$$x^{r,i} \oplus y^{r,p(i)} = 0, i = 0 \text{ to } 63 \quad (12)$$

In total there are 2048 equations for all 32 rounds.

The complete set comprises $(27,905 - 64j)$ equations, where j ranges from 0 to 32, and we convert the aforementioned equations into CNF format as input for the solver.

3.4. Experiments about Our Combined Analysis on PICO

The comprehensive strategy of our combined analysis approach encompasses two main components—solving a subset of subkeys with differential faults, as detailed in Section 3.2, and establishing an algebraic equation set, as instructed by Section 3.3. In the case of PICO, double-byte faults are injected multiple times into the input of *SubColumn* in the preceding round for each of the target subkey, with the aim of minimizing the number of faults used, and the subset of subkeys can be acquired sequentially in reverse order. Within the algebraic equation set, equations representing the full-round reverse key expansion and assigning for the subset of subkeys derived from differential faults are utilized to solve potential full-round subkeys and their corresponding master keys. Additionally, equations representing the full-round forward encryption and assigning for the correct plaintext and ciphertext are employed to verify each set of potential full-round subkeys for the purpose of the correct master key, ensuring that the ciphertext obtained by encrypting the correct plaintext with the corresponding subkey set is accurate. Consequently, the complete master key of PICO can be obtained. Following this, we carried out experiments on our combined analysis on PICO.

In practical experiments, various methods can be employed to inject faults [26–34], including variations in the operational voltage, clock, or temperature of the circuit, electromagnetic pulse interference with circuit operation, utilization of laser, UV-ray, X-ray, or focused ion beam (FIB), etc. In addition to this, the rest of the key recovery processes are all accomplished through computer computations. Considering this case, this paper assumes that faults can be successfully injected and concentrated on computations within the computer. Our experiments run on a server with the Ubuntu 18.04.6 LTS operating system. It is equipped with two physical CPUs, each consisting of twelve cores and two threads per core. The solver version used is CryptoMiniSat 5.8.0 and multi-threading is implemented for parallel computation during the experiment.

Algorithm 6 outlines the experimental procedures of our combined analysis on PICO, where N denotes the number of experiments, n denotes the number of faults used, T_i ($i = 0$ to $N - 1$) denotes the solving time for the i -th experiment, and T_{ave} denotes the average solving time across N experiments. In our experiments, Python v3.9.2 is employed for simulating differential fault injection, solving a subset of subkeys, establishing algebraic equation sets, and generating cnf files. CryptoMiniSat v5.8.0 is utilized to solve CNF equations, with the process ceasing upon identification of a matching master key. As Python v3.9.2 can solve the subset of subkeys within a matter of seconds, the experimental results to be observed are the solving time by CryptoMiniSat v5.8.0, which covers the time required for solving reverse key expansion equations and filtering through encryption verification, and the number of faults used.

Algorithm 6: Experimental Procedures of our Combined Analysis on PICO

Input: N
Output: n, T_i, T_{ave}

```

1 for  $i = 0; i < N; i++$  do
2   Generate random  $P, K$ , where  $K$  is unknown to the analyst;
3    $C \leftarrow \text{Enc}(P, K)$ ;
4   Solving for  $K^{32} \cdots K^j$  ( $0 \leq j \leq 32$ ) with differential faults and record the
     number of faults used as  $n$ ;
5   Generate assignment equations for  $1, P, C, K^{32}, \dots, K^j$ ;
6   for  $\text{round} = 0; \text{round} < 33; \text{round}++$  do
7     Generate equations for reverse key expansion;
8     Generate equations for forward encryption;
9   Combine all the aforementioned equations into one set;
10   $T_i \leftarrow$  solving the equation set with CryptoMiniSat v5.8.0;
11  $T_{ave} \leftarrow \text{Sum}(T_i)/N$ ;
12 return  $n, T_i, T_{ave}$ 

```

For each scenario, we conducted 100 experiments, and the detailed results are presented in Figure 2 and Table 4. When utilizing 44 differential faults to derive subkeys $K^{32}, K^{31} \dots K^{22}$, the master key of PICO could be recovered in an average time of 884.77 s. Similarly, when utilizing 40 differential faults to derive subkeys $K^{32}, K^{31} \dots K^{23}$, the master key of PICO could be recovered in an average time of 2056.56 s. These demonstrate that our combined analysis approach is an effective method to obtain PICO's complete master key.

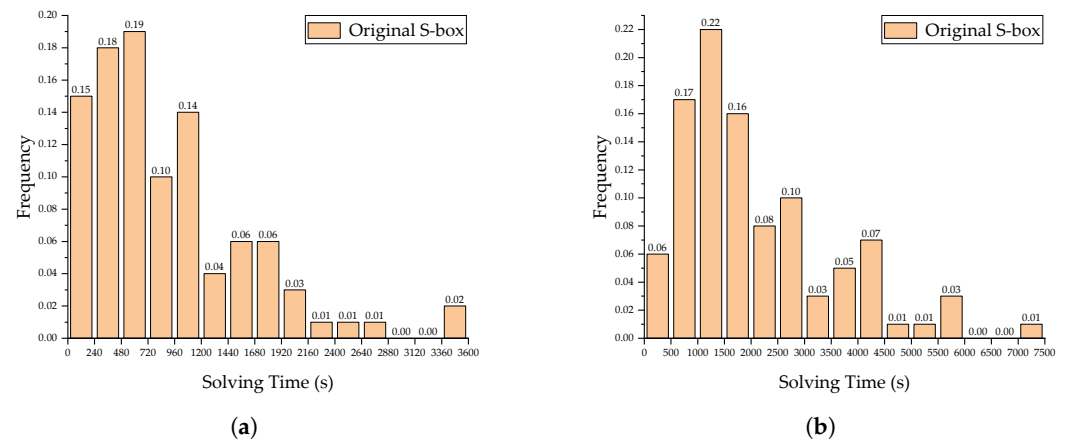


Figure 2. Distribution of solving time under varying numbers of faults when conducting combined analysis on PICO using the original S-box. (a) $n = 44$; (b) $n = 40$.

Table 4. Solving results under varying numbers of faults when conducting combined analysis on PICO using the original S-box.

Number of Faults	Solving Time (s)		Success Rate within 1 h
	Average	Median	
44	884.77	713.33	100%
40	2056.56	1691.03	83%

4. Optimization through S-Box Decomposition

Within our combined analysis approach proposed in Section 3, the two primary metrics for evaluating results are the number of applied faults and the time taken for successful solving. We aim to enhance the effectiveness of our approach further. As algebraic equations are introduced, and the nature of the solver CryptoMiniSat v5.8.0 necessitates the use of additional equations to represent higher-order variables, we implement S-box decomposition [35,36] to reduce the variety of higher-order variables, thereby decreasing the number of equations required to represent full-round *SubColumn*. Consequently, the solving time of the solver is shortened under the same quantity of faults, ultimately resulting in a reduction in the number of faults needed to recover the master key in practical applications.

The algebraic equation representation of PICO's S-box, as presented in Section 3, consists of six quadratic variables and two cubic variables. Our objective is to decompose the original 4×4 S-box into two separate 4×4 S-boxes, with two requirements. Firstly, higher-order variables in the representation of decomposed S-boxes should include only quadratic terms. Additionally, the output for identical inputs should remain consistent through either the original S-box or the decomposed S-boxes.

Let $X = (x_3, x_2, x_1, x_0)$ denote the input to the S-box, where x_3 denotes the most significant bit and x_0 denotes the least significant bit. Let Equation (13) represent the algebraic equations for the original S-box,

$$S(X) = (s_3(X), s_2(X), s_1(X), s_0(X)), \quad (13)$$

and Equations (14) and (15) represent the algebraic equations for the two decomposed S-boxes, respectively,

$$G(X) = (g_3(X), g_2(X), g_1(X), g_0(X)), \quad (14)$$

$$F(X) = (g_3(X), g_2(X), g_1(X), g_0(X)), \quad (15)$$

where $g_i(X)$ and $f_i(X)$ ($i = 0$ to 3) are both quadratic Boolean functions, and $G(X)$ and $F(X)$ are both quadratic vectorial Boolean functions, satisfying $S(X) = F(G(X))$ while $S, G, F : GF(2)^4 \rightarrow GF(2)^4$. Quadratic Boolean functions g_i ($i = 0$ to 3) are represented in the form of ANF equations, as indicated in Equation (16),

$$g_i(X) = a_{i,0} + a_{i,1}x_0 + a_{i,2}x_1 + a_{i,3}x_2 + a_{i,4}x_3 + a_{i,5}x_0x_1 + a_{i,6}x_0x_2 + a_{i,7}x_0x_3 + a_{i,8}x_1x_2 + a_{i,9}x_1x_3 + a_{i,10}x_2x_3, \quad (16)$$

where $a_{i,j}$ ($j = 0$ to 10) are the coefficients of g_i , with values of 0 or 1. The steps for implementing S-box decomposition are shown in Figure 3.

Following S-box decomposition using Python v3.9.2, we obtain 2,257,920 pairs of $G(X)$ and $F(X)$ satisfying $S(X) = F(G(X))$, with the highest-order variables in both $G(X)$ and $F(X)$ not exceeding quadratic. And after selection, we end up with 24 pairs of $G(X)$ and $F(X)$, where the minimum number of types of quadratic variables is five and the minimum weight sum is 244. These 24 pairs of $G(X)$ and $F(X)$ share similar forms, as presented in Table 5.

Table 5. Forms of the final 24 pairs of $G(X)$ and $F(X)$ after S-box decomposition.

Function	Quantity of Variables		
	Quadratic Terms	Linear Terms	Constant Terms
$g_i(X)$ ($i = 0$ to 3)	1	3	1
	2	2	0
	0	2	0
	0	1	0
$f_i(X)$ ($i = 0$ to 3)	0	2	0
	2	1	1
	0	1	0
	1	1	0

Therefore, we choose one set for subsequent experiments. The decomposed S-boxes we opted for is detailed in Table 6, and its algebraic equation representation is provided in Equation (17).

Table 6. Decomposed S-boxes of PICO.

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

$$\begin{aligned}
 G(x_3, x_2, x_1, x_0) &= (g_3, g_2, g_1, g_0) \\
 g_0 &= 1 + x_0 + x_2 + x_3 + x_1x_3 \\
 g_1 &= x_1 + x_2 + x_0x_3 + x_2x_3 \\
 g_2 &= x_1 + x_3 \\
 g_3 &= x_1 \\
 F(x_3, x_2, x_1, x_0) &= (f_3, f_2, f_1, f_0) \\
 f_0 &= x_0 + x_3 \\
 f_1 &= 1 + x_0 + x_0x_1 + x_1x_2 \\
 f_2 &= x_1 \\
 f_3 &= x_2 + x_0x_1
 \end{aligned} \quad (17)$$

An algebraic equation set can be established utilizing the decomposed S-boxes as per Section 3.3, and variations in parameters before and after decomposition are summarized in Table 7. It reveals a reduction of 3584 equations, which suggests a potential speedup.

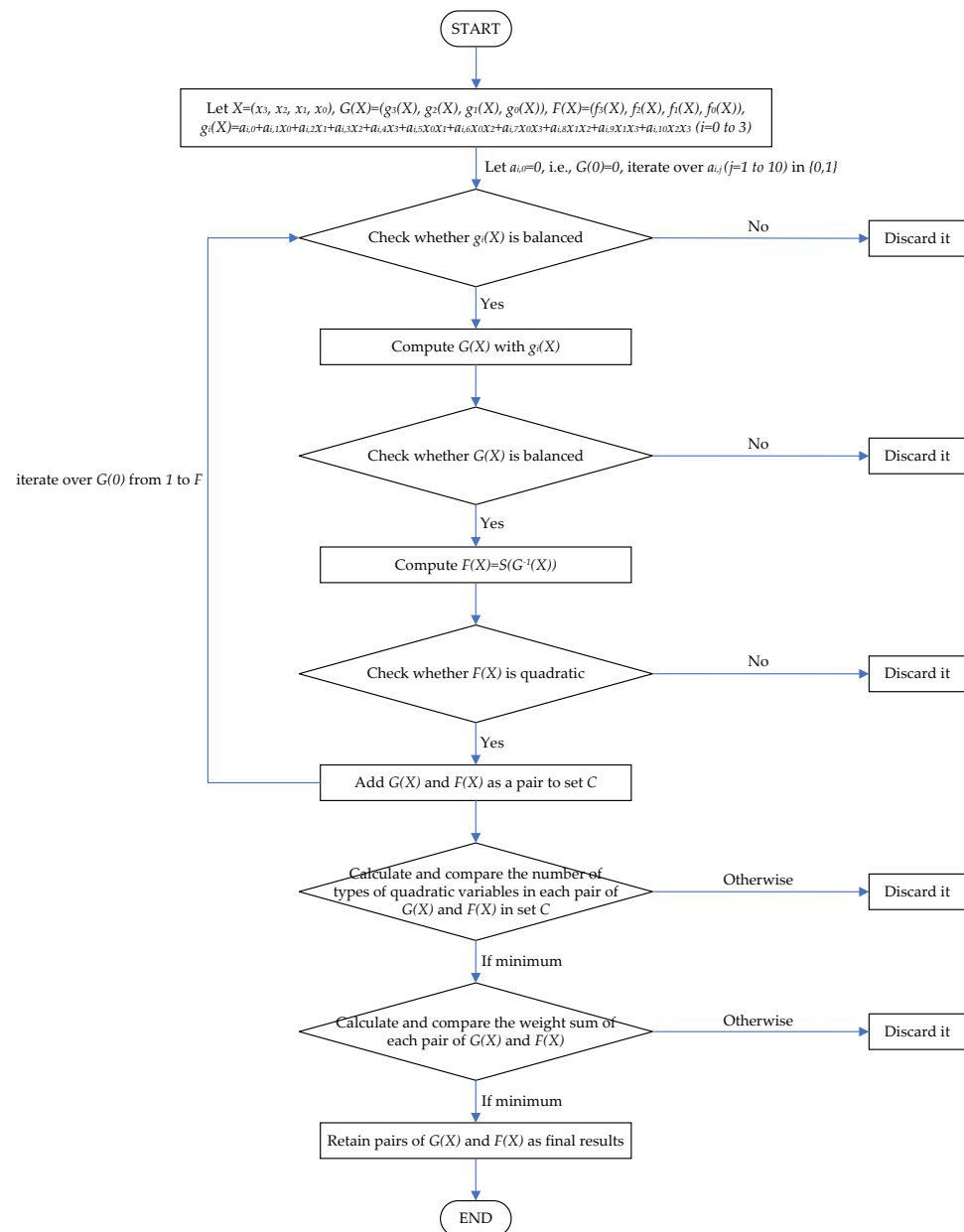


Figure 3. Flowchart for decomposing the S-box.

Table 7. Comparison of parameters before and after S-box decomposition.

	Quantity of Parameters				Equations in Total
	Types of Cubic Variables	Types of Quadratic Variables	Types of High-Order Variables in Total	Equations Representing Full-Round SubColumn	
Original S-box	2	6	8	15,360	$27,905-64j$ ($0 \leq j \leq 32$)
Decomposed S-boxes	0	$3+2$	5	11,776	$24,321-64j$ ($0 \leq j \leq 32$)

After establishing the equation set based on the decomposed S-boxes, experiments are performed following Algorithm 6, with 100 trials for each scenario. Comparisons of experimental results for 44 and 40 faults before and after S-box decomposition are displayed in Figure 4 and Table 8. Figure 4 demonstrates that, under consistent conditions, the time distribution for solving PICO's master key is significantly concentrated in shorter time intervals when using the decomposed S-boxes, in comparison with the period before S-box decomposition. Table 8 indicates that after S-box decomposition, for a fault count of 44, the average and median solving times for PICO's master key decreased by 64.89% and 64.13%, respectively. Similarly, for a fault count of 40, the average and median solving times decreased by 75.83% and 75.30%, respectively. Additionally, S-box decomposition enabled the success rate for solving within 1 h to reach 100% in both scenarios.

Table 8. Comparison of solving results under varying numbers of faults when conducting combined analysis on PICO using the original and decomposed S-boxes.

Number of Faults	Type of S-Box	Solving Time (s)		Success Rate within 1 h
		Average	Median	
44	original S-box	884.77	713.33	100%
	decomposed S-boxes	310.60	255.85	100%
40	original S-box	2056.56	1691.03	83%
	decomposed S-boxes	497.13	417.72	100%

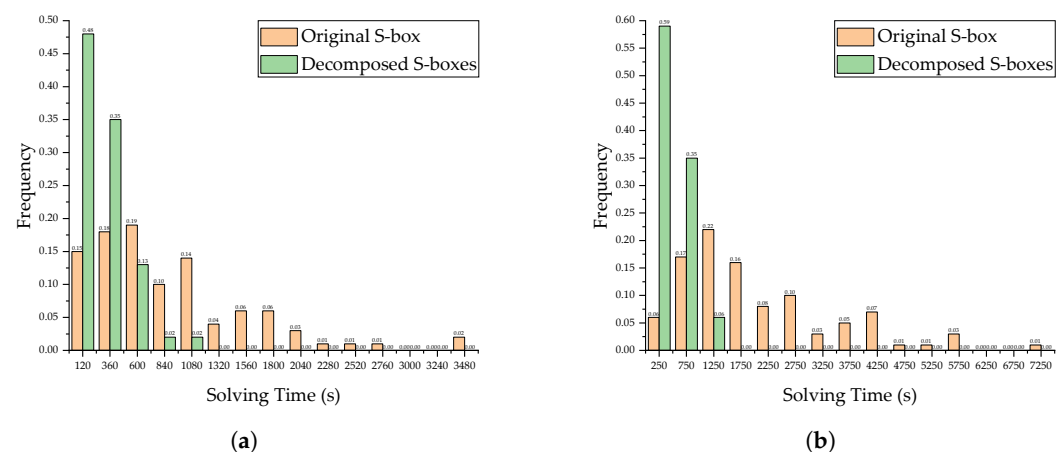


Figure 4. Comparison of solving time distribution under varying numbers of faults when conducting combined analysis on PICO using the original and decomposed S-boxes. (a) $n = 44$; (b) $n = 40$.

With the maximum solving time set to 1 h, we carried out more comparative experiments and the comparison of success rates within 1 h can be observed in Figure 5. The results suggest that with fault counts of 40 and 36, S-box decomposition led to a 100% success rate for solving within 1 h. Similarly, for fault counts of 32 and 28, S-box decomposition increased the success rate by 61% and 55%, respectively. All the aforementioned experimental outcomes confirm that S-box decomposition is a potent optimization for our combined analysis on PICO.

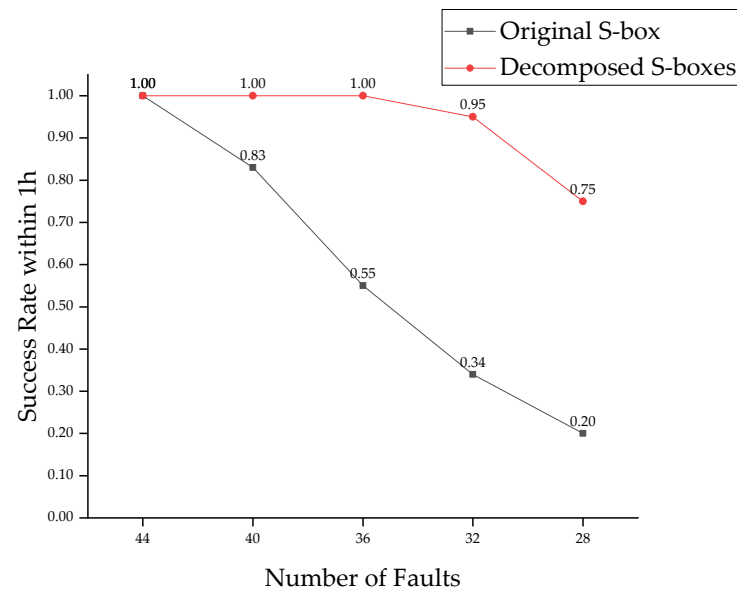


Figure 5. Comparison of success rate within 1 h under varying numbers of faults when conducting combined analysis on PICO using the original and decomposed S-boxes.

Figure 6 and Table 9 present detailed experimental results for fault counts of 32 and 28 when using the decomposed S-boxes. When utilizing 32 differential faults to derive subkeys $K^{32}, K^{31} \dots K^{25}$, the master key of PICO could be recovered in an average time of 1457.13 s. Similarly, when utilizing 28 differential faults to derive subkeys $K^{32}, K^{31} \dots K^{26}$, the master key of PICO could be recovered in an average time of 2821.62 s. Compared to Section 3, where 40 faults were required to recover PICO's master key in 0.57 h, S-box decomposition allowed the solution for only 28 faults in 0.78 h, which reduced the fault count by 30% while maintaining the average solving time within 1 h.

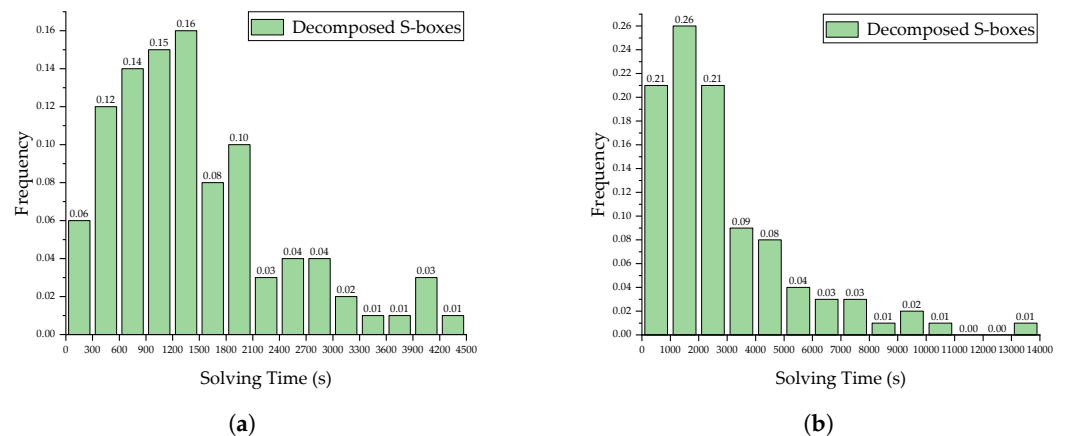


Figure 6. Distribution of solving time under varying numbers of faults when conducting combined analysis on PICO using the decomposed S-boxes. (a) $n = 32$; (b) $n = 28$.

Table 9. Solving results under varying numbers of faults when conducting combined analysis on PICO using the decomposed S-boxes.

Number of Faults	Solving Time (s)		Success Rate within 1 h
	Average	Median	
32	1457.13	1211.00	95%
28	2821.62	2194.28	75%

In contrast to ref. [15], the only study concerning master key recovery of PICO, the mere use of a 64-bit exhaustive search resulted in 2^{64} possible master key permutations, not accounting for the complexity of searching for the other 64 bits. However, in our combined analysis approach using the decomposed S-boxes with 28 faults, the search space for PICO's master key was reduced to 2^{27} , signifying a massive decrease of 2^{37} .

5. Comparison with DFA and AFA

Differential fault analysis (DFA) and algebraic fault analysis (AFA) are both conventional fault analysis methods for conducting key recovery. However, neither of these approaches has been applied in the study of PICO. In this section, we compare our combined analysis approach with these two methods and elucidate the reasons and benefits of opting for our approach.

5.1. Comparison with DFA

The fundamental concept of DFA is to extract valuable information about the key by examining the differences in ciphertext values before and after injecting faults, along with the encryption process from the site of fault injection to completion. According to Sections 3.2 and 3.3, for PICO cipher, if one utilizes differential faults to derive all subkeys, the correct master key can be obtained, but the number of faults used can be as high as 132. And if one opts to obtain a subset of subkeys with fewer differential faults and then deduce the master key through reverse key expansion, the process of reverse key expansion requires resolution of a significant number of equations, leading to non-unique solutions. Accordingly, the inclusion of accurate plaintext-ciphertext pairs and the entire encryption process is necessary to distinguish the correct master key. In this way, despite the reduction in faults, the solving process becomes much more intricate and complex.

Therefore, based on the characteristics of the solving process, we introduce an algebraic method, making the part of equation solving in the reverse key expansion process handled by the solver, with the subsequent filtering process achievable through algebraic equations.

5.2. Comparison with AFA

The core idea of AFA is to use algebraic equations to articulate the encryption process both before and after fault injection, and employ a solver to determine the master key. The established equations serve various roles. Multiple sets of equations that cover the encryption process from the site of fault injection to completion are to obtain the subkeys for these rounds. Additionally, equations representing key expansion are to acquire possible master keys. Furthermore, equations for full-round encryption are to ensure that the attained master key can encrypt the correct plaintext into the correct ciphertext.

Contrasting with our combined analysis, the distinguishing factor is that AFA utilizes algebraic equations to solve subkeys used after the round of fault injection, while our approach derives subkeys for each round in a reverse order using differential values before and after fault injection. In the context of AFA, there are two subjective factors that can affect the solving speed—the round of fault injection and the number of faults, each of which is discussed individually as follows. When considering the round where faults are injected, if the round is distant from the final round, there is a higher count of equations from the injection site to the end and a larger number of subkeys to be solved, resulting in an extended solving time. Conversely, if the round is close to the final round, the number of derived subkeys decreases, and the subsequent filtering process takes longer due to the increased count of possible master keys inferred from key expansion. The number of faults, on the other hand, affects the quantity of equations from the fault injection round to the end. These equations are designed for solving a specific number of subkeys. If the capability to solve is already established, increasing the number of faults indiscriminately does not enhance the solving speed. Hence, AFA is constrained by the objective experimental conditions. Faster solving speed is attainable only when a greater number of threads are concurrently engaged in the solution, provided that an appropriate selection of fault

injection round and fault number is made. Otherwise, there is no alternative means to expedite the process. However, our combined analysis approach can overcome this drawback. The increase in the number of faults allows for the extraction of more subkeys within seconds, leading to a reduction in the potential number of master keys and their filtering time, ultimately shortening the overall solution time.

We have also experimented with the application of AFA on PICO to validate our theory. All equations required are established through a forward process. The equations before fault injection, referring to Section 3.3, amount to a total of 25,664, with the forward and reverse representations of key expansion being identical. The assignment equations and those after fault injection are detailed below. Due to the challenge of obtaining the complete master key in a short time, we assign values to some bits of the master key and test the time required to solve the remaining portion.

- Equations for assignment
Include the constant 1, the correct plaintext P , the correct ciphertext C , v ($0 \leq v \leq 128$) bits assigned to the master key, and erroneous ciphertexts $C^{0*} \dots C^{n-1*}$ ($n \geq 1$) obtained from fault injections performed n times, for a total of $(64n + v + 129)$ equations.
- Equations for the differential values at the fault injection site
Assume the fault is injected in the input of *SubColumn* in the r -th ($1 \leq r \leq 32$) round. Let $x^{r,63} \parallel x^{r,62} \parallel \dots \parallel x^{r,0}$ denote the state before fault injection, $x^{r,63*} \parallel x^{r,62*} \parallel \dots \parallel x^{r,0*}$ denote the state after fault injection, and a fault known in width and specific bit positions be injected at $x^{r,q*} \parallel \dots \parallel x^{r,p*}$ ($0 \leq p \leq q \leq 63$). Equations are as shown in Equation (18).

$$\begin{cases} x^{r,i} \oplus x^{r,i*} = 0, & i = 0 \text{ to } p-1 \text{ and } i = q \text{ to } 63 \\ x^{r,i} \oplus x^{r,i*} = 1, & i = p \text{ to } q \end{cases} \quad (18)$$

In total there are $64n$ equations for n instances of fault injections.

- Equations for forward encryption from the round of fault injection to the end
This segment encompasses the encryption processes from the r -th round to the concluding round after each fault injection. Equation establishment refers to Section 3.3, resulting in a total of $(20, 128n - 608rn)$ equations for n instances of fault injections.

The complete set of AFA conducted on PICO comprises $(20, 256n + v + 25,793 - 608rn)$ equations, where n is no less than 1, v ranges from 0 to 128, and r ranges from 1 to 32.

Algorithm 7 outlines the experimental process of conducting AFA on PICO, where r ($1 \leq r \leq 32$) denotes the round of fault injection, n ($n \geq 1$) denotes the number of faults, w ($w = 1$ or 4 or 8 or 16) denotes the width of faults, v ($0 \leq v \leq 128$) denotes the number of bits assigned to the master key, N denotes the total number of experiments, N_{suc} denotes the number of experiments successfully solved within 1 h, and R_{1h} denotes the success rate within 1 h for N experiments. The solving results to be observed remain as the outputs from CryptoMiniSat v5.8.0. Additionally, each scenario in the following experiments consisted of 50 trials.

Taking single-bit faults as examples. The selection of the fault injection round is guided by Figure 7, with the aim of maximizing the diffusion and impact of the fault across several encryption rounds. Moreover, we perform fault injections in different S-boxes to ensure effective fault diffusion.

Experimental results for single-bit fault injection when conducting AFA on PICO are presented in Figure 8. Figure 8a depicts that even with assigning bits to part of PICO's master key, there is still no round of fault injection that can achieve a success rate of 100% within 1 h. Figure 8b shows that simply increasing the number of faults does not lead to an improvement in the success rate within 1 h. Figure 8c demonstrates a decline in the success rate within 1 h as the number of bits assigned to PICO's master key decreases, indicating the challenge of using AFA to recover the complete master key of PICO within 1 h. In addition, experiments were conducted under the conditions of $r = 29$, $n = 6$ and $v = 0$, with the objective of solving the entire master key of PICO. Across 15 experiments, the average

solving time for PICO's complete master key was 32.54 h, confirming the difficulty of obtaining it within 1 h using AFA. We also carried out experiments with nibble, single-byte, and double-byte faults at various fault injection rounds and fault counts, and achieving a success rate of 100% within 1 h all proved challenging when assigning 10 bits to PICO's master key. Considering the reasons, we believe that the high complexity of PICO's key expansion and the resulting multiplicity of equation solutions lead to excessively long equation solving time, causing difficulties in recovering its master key within a short time period using AFA.

Algorithm 7: Experimental Procedures of AFA on PICO

Input: r, n, w, v, N
Output: R_{1h}

```

1  $N_{suc} \leftarrow 0;$ 
2 for  $i = 0; i < N; i++$  do
3   Generate random  $P, K$ , where  $K$  is unknown to the analyst;
4    $C \leftarrow Enc(P, K);$ 
5   Generate assignment equations for  $1, P, C;$ 
6   for  $count = 0; count < v; count++$  do
7     Generate assignment equations for a randomly selected bit of  $K;$ 
8   for  $round = 0; round < 33; round++$  do
9     Generate equations for forward key expansion;
10    Generate equations for forward encryption;
11  for  $count = 0; count < n; count++$  do
12    Inject a fault of width  $w$  at a randomly chosen and known to the analyst
      position in the input of SubColumn in the  $r$ th round;
13    Generate assignment equations for  $C^{count*};$ 
14    Generate differential equations for the fault injection site;
15    for  $round = r; round < 33; round++$  do
16      Generate equations for forward encryption after the  $r$ th round ;
17  Combine all the aforementioned equations into one set;
18  Solve the equation set with CryptoMiniSat v5.8.0 while setting the maximum
      solving time to 1 h;
19  if solving succeeds then
20     $N_{suc} \leftarrow N_{suc} + 1;$ 
21  $R_{1h} \leftarrow N_{suc} / N;$ 
22 return  $R_{1h}$ 

```

However, our combined analysis approach adopts a strategy that involves first utilizing differential faults to solve partial subkeys of PICO and then establishing algebraic equations to recover its master key, with the former requiring no equation solving. This strategic approach narrows down the search space for equation solutions, thereby effectively addressing the issue of prolonged equation solving time. With the decomposed S-boxes, our combined analysis approach achieved successful recovery of PICO's entire master key in an average time of 0.78 h, which was significantly shorter than the 32.54 h required when using AFA. Furthermore, all these AFA experiments were conducted with known fault positions and still faced difficulties in obtaining solutions. In our approach, valuable information can be obtained as long as the fault positions are non-repetitive, eliminating the requirement for precise fault locations.

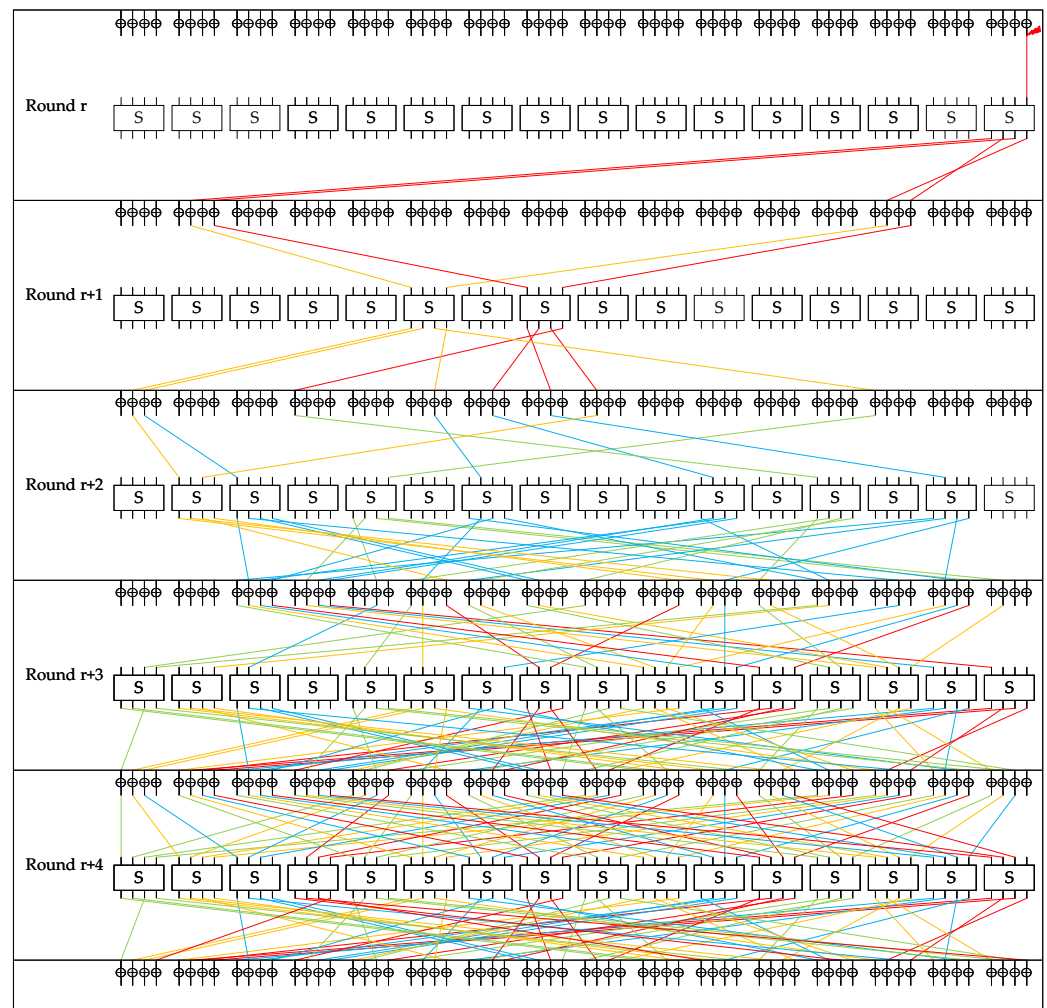


Figure 7. Diffusion of a single-bit fault in PICO.

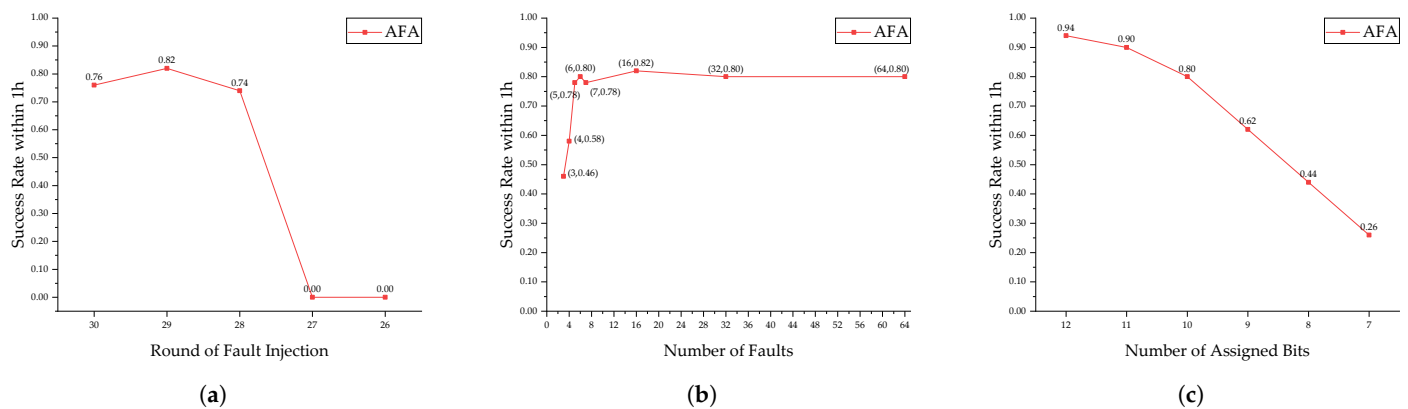


Figure 8. Success rate within 1 h under different conditions when conducting AFA (algebraic fault analysis) on PICO. (a) keeping $n = 16, v = 10$, and varying r as 30, 29, 28, 27, 26; (b) keeping $r = 29, v = 10$, and varying n as 3, 4, 5, 6, 7, 16, 32, 64; (c) keeping $r = 29, n = 6$, and varying v as 12, 11, 10, 9, 8, 7.

5.3. Summary

To summarize, the high complexity of PICO's key expansion results in challenges for conventional fault analysis methods DFA and AFA in their attempts to solve its entire master key with a reduced number of faults. It can be observed that, when using DFA, the reverse process of PICO's key expansion, specifically Equation (3), introduces the equation-solving

issue during the backward derivation of the master key. And the presence of multiple solutions to the equations results in an exponential increase in the number of derived master keys throughout the successive round-by-round backward deduction process, necessitating further filtering and verification. Likewise, when using AFA, the complexity arising from the equation representation of PICO's key expansion process poses challenges in solving its master key.

However, our combined analysis approach offers two significant solutions compared to DFA. As the number of faults is reduced, it effectively addresses the intricacy of equation solving in the reverse process of PICO's key expansion, and efficiently manages the filtering problem of identifying the correct master key. Our combined analysis approach also stands out from AFA with two notable advantages. It overcomes the limitation of AFA, where adjustments to its two main parameters—the round of fault injection and the number of faults—cannot shorten the solving time indiscriminately, and accomplishes the solution of PICO's entire master key within a significantly shorter time. Furthermore, it operates without the need of specific fault locations.

6. Conclusions and Future Work

In this paper, we propose an innovative fault analysis approach addressing the complicated key expansion of PICO cipher, which integrates techniques of both differential fault and algebraic equation. With our approach, we achieved the successful recovery of PICO's complete master key with 40 faults in an average time of 0.57 h. We further optimize our approach through S-box decomposition, attaining a significant 75.83% reduction in average solving time under the consistent 40-fault condition. This advancement enabled the resolution of PICO's complete master key with 28 faults in an average time of 0.78 h. Additionally, compared to the previous study, its search space was reduced from 2^{64} to 2^{27} , representing a significant decrease of 2^{37} . To our knowledge, this marks the first fault analysis conducted on PICO. Through contrast with DFA and AFA, it can be observed that the complex key expansion of PICO presents challenges for solving its master key, and our combined analysis approach outperforms these two conventional fault analysis methods in recovering PICO's entire master key. Accordingly, this provides valuable inspiration for future research.

As PICO's key expansion is inspired by that of SPECK, where both share similar characteristics, such as intermediate variables besides subkeys and circular shift operations, our combined analysis approach offers a strategy for recovering the master keys of other block ciphers with similarly intricate key expansions processes. For instance, if the reverse process of key expansion consists of complex equation representations, leading to a non-one-to-one correspondence between the parameters involved, thereby preventing the unique determination of the key value, or if the solution to the master key has multiple potential outcomes, and additional filtering is required to verify the solution. Moreover, S-box decomposition holds promise for enhancing solving efficiency, provided that after decomposing the S-box in the block cipher, there is a reduction in the number of types of high-order variables, leading to fewer equations within the equation set.

In addition, given that the high complexity in PICO's key expansion presents challenges for conventional fault analysis methods DFA and AFA in decrypting its master key, it can be deduced that complex key expansions play a crucial role in protecting block ciphers from key recovery attacks, where higher complexity corresponds to increased safety. Consequently, in the design of future block ciphers, the complexity of their key expansions can be enhanced to produce them with more secure structures, therefore enabling them to be more resistant to key decryption threats. This can involve introducing alternating iterative relationships among multiple intermediate variables, or implementing circular shift operations on variables to generate non-unique values during reverse key deduction. By doing so, block ciphers can be endowed with heightened security to effectively resist key decryption.

Author Contributions: Conceptualization, H.Z. and L.D.; methodology, L.D.; software, L.D.; validation, L.D. and Y.W.; formal analysis, L.D.; investigation, L.D. and X.F.; resources, H.Z.; data curation, L.D.; writing—original draft preparation, L.D.; writing—review and editing, L.D.; visualization, L.D.; supervision, H.Z. and J.X.; project administration, H.Z. and J.X.; funding acquisition, H.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (62071057).

Data Availability Statement: The data presented in this study are available on request from the corresponding author

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Bogdanov, A.; Knudsen, L.R.; Leander, G.; Paar, C.; Poschmann, A.; Robshaw, M.J.; Seurin, Y.; Vikkelsoe, C. PRESENT: An ultra-lightweight block cipher. In *Cryptographic Hardware and Embedded Systems—CHES 2007, Proceedings of the 9th International Workshop, Vienna, Austria, 10–13 September 2007*; Proceedings 9; Springer: Berlin/Heidelberg, Germany, 2007; pp. 450–466.
2. Beierle, C.; Jean, J.; Kölbl, S.; Leander, G.; Moradi, A.; Peyrin, T.; Sasaki, Y.; Sasdrich, P.; Sim, S.M. The SKINNY family of block ciphers and its low-latency variant MANTIS. In *Advances in Cryptology—CRYPTO 2016, Proceedings of the 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, 14–18 August 2016*; Proceedings, Part II 36; Springer: Berlin/Heidelberg, Germany, 2016; pp. 123–153.
3. Banik, S.; Pandey, S.K.; Peyrin, T.; Sasaki, Y.; Sim, S.M.; Todo, Y. GIFT: A small present: Towards reaching the limit of lightweight encryption. In *Cryptographic Hardware and Embedded Systems—CHES 2017, Proceedings of the 19th International Conference, Taipei, Taiwan, 25–28 September 2017*; Fischer, W., Homma, N., Eds.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2017; pp. 321–345, ISBN 978-3-319-66787-4.
4. Beaulieu, R.; Shors, D.; Smith, J.; Treatman-Clark, S.; Weeks, B.; Wingers, L. The SIMON and SPECK Families of Lightweight Block Ciphers. Cryptology ePrint Archive. Paper 2013/404. 2013. Available online: <https://eprint.iacr.org/2013/404> (accessed on 30 January 2024).
5. Wu, W.; Zhang, L. LBlock: A lightweight block cipher. In *Applied Cryptography and Network Security, Proceedings of the 9th International Conference, ACNS 2011, Nerja, Spain, 7–10 June 2011*; Proceedings 9; Springer: Berlin/Heidelberg, Germany, 2011; pp. 327–344.
6. Zhang, W.; Bao, Z.; Lin, D.; Rijmen, V.; Yang, B.; Verbauwhede, I. RECTANGLE: A Bit-Slice Lightweight Block Cipher Suitable for Multiple Platforms. Cryptology ePrint Archive. Paper 2014/084. 2014. Available online: <https://eprint.iacr.org/2014/084> (accessed on 30 January 2024).
7. Bansod, G.; Pisharoty, N.; Patil, A. PICO: An Ultra Lightweight and Low Power Encryption Design for Ubiquitous Computing. *Def. Sci. J.* **2016**, *66*, 259. [CrossRef]
8. Ma, C.; Liu, G.; Li, C. Zero-correlation Linear Cryptanalysis on PICO and RECTANGLE. *J. Cryptologic Res.* **2017**, *4*, 413–422.
9. Ma, C. Application of Mixed-Integer Linear Program in the Security Analysis of Block Cipher. Master's Thesis, National University of Defense Technology, Changsha, China, 2017.
10. Kumar, M.; Suresh, T.; Pal, S.K.; Panigrahi, A. Optimal differential trails in lightweight block ciphers ANU and PICO. *Cryptologia* **2020**, *44*, 68–78. [CrossRef]
11. Liu, Z.; Yuan, Z.; Zhao, C.; Zhu, L. Integral attack on PICO algorithm based on division property. *J. Comput. Appl.* **2020**, *40*, 2967.
12. Liu, Z. Security Analysis of Block Cipher Based on Automated Search. Master's Thesis, Xidian University, Xi'an, China, 2021.
13. Qiu, X. Research on Automatic Search Methods of New Distinguishers for Lightweight Block Ciphers. Master's Thesis, Guilin University of Electronic Technology, Guilin, China, 2021.
14. Zhao, C. Impossible Differential Cryptanalysis of Lightweight Block Ciphers. Master's Thesis, Xidian University, Xi'an, China, 2021.
15. Wang, C.; Zhang, Z.; Hu, L. Differential Cryptanalysis on Ultra Lightweight Block Cipher PICO. *J. Cryptologic Res.* **2023**, *10*, 685.
16. Shi, K.; Ren, J.; Chen, S. MILP-Based Search for Differential and Linear Distinguishers of PICO Algorithm. *J. Cryptologic Res.* **2023**, *10*, 910.
17. Biham, E.; Shamir, A. Differential fault analysis of secret key cryptosystems. In *Advances in Cryptology—CRYPTO'97, Proceedings of the 17th Annual International Cryptology Conference Santa Barbara, CA, USA, 17–21 August 1997*; Proceedings 17; Springer: Berlin/Heidelberg, Germany, 1997; pp. 513–525.
18. Luo, H.; Chen, W.; Ming, X.; Wu, Y. General differential fault attack on PRESENT and GIFT cipher with nibble. *IEEE Access* **2021**, *9*, 37697–37706. [CrossRef]
19. Vafaei, N.; Porkar, M.; Ramzanipour, H.; Bagheri, N. Practical Differential Fault Analysis on SKINNY. *ISecure* **2022**, *14*, 9–19.
20. Xiao, H.; Wang, L. Differential fault analysis on the key schedule of the LBlock algorithm. *IEEE Access* **2022**, *10*, 62402–62411. [CrossRef]

21. Courtois, N.; Jackson, K.; Ware, D. Fault-algebraic attacks on inner rounds of des. In Proceedings of the e-Smart'10 Proceedings: The Future of Digital Security Technologies, Sophia Antipolis, France, 22–24 September 2010.
22. Zhang, F.; Guo, S.; Zhao, X.; Wang, T.; Yang, J.; Standaert, F.X.; Gu, D. A framework for the analysis and evaluation of algebraic fault attacks on lightweight block ciphers. *IEEE Trans. Inf. Forensics Secur.* **2016**, *11*, 1039–1054. [\[CrossRef\]](#)
23. Zhu, L.; Gong, J.; Dong, L.; Zhang, C. Temperature-Triggered Hardware Trojan Based Algebraic Fault Analysis of SKINNY-64-64 Lightweight Block Cipher. *Comput. Mater. Contin.* **2023**, *75*, 5521–5537. [\[CrossRef\]](#)
24. Soos, M.; Nohl, K.; Castelluccia, C. Extending SAT solvers to cryptographic problems. In *Theory and Applications of Satisfiability Testing—SAT 2009, Proceedings of the International Conference on Theory and Applications of Satisfiability Testing, Swansea, UK, 30 June–3 July 2009*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 244–257.
25. Knudsen, L.R.; Miolane, C.V. Counting equations in algebraic attacks on block ciphers. *Int. J. Inf. Secur.* **2010**, *9*, 127–135. [\[CrossRef\]](#)
26. Shepherd, C.; Markantonakis, K.; van Heijningen, N.; Aboulkassimi, D.; Gaine, C.; Heckmann, T.; Naccache, D. Physical fault injection and side-channel attacks on mobile devices: A comprehensive analysis. *Comput. Secur.* **2021**, *111*, 102471. [\[CrossRef\]](#)
27. Baksi, A.; Bhasin, S.; Breier, J.; Jap, D.; Saha, D. A survey on fault attacks on symmetric key cryptosystems. *ACM Comput. Surv.* **2022**, *55*, 1–34. [\[CrossRef\]](#)
28. Breier, J.; Hou, X. How practical are fault injection attacks, really? *IEEE Access* **2022**, *10*, 113122–113130. [\[CrossRef\]](#)
29. Shuvo, A.M.; Zhang, T.; Farahmandi, F.; Tehranipoor, M. A Comprehensive Survey on Non-Invasive Fault Injection Attacks. Cryptology ePrint Archive. Paper 2023/1769. 2023. Available online: <https://eprint.iacr.org/2023/1769> (accessed on 30 January 2024).
30. Ruminot, N.; Estevez, C.; Montejo-Sánchez, S. A Novel Approach of a Low-Cost Voltage Fault Injection Method for Resource-Constrained IoT Devices: Design and Analysis. *Sensors* **2023**, *23*, 7180. [\[CrossRef\]](#) [\[PubMed\]](#)
31. Beckers, A.; Kinugawa, M.; Hayashi, Y.; Fujimoto, D.; Balasch, J.; Gierlichs, B.; Verbauwhede, I. Design considerations for em pulse fault injection. In *Smart Card Research and Advanced Applications, Proceedings of the 18th International Conference, CARDIS 2019, Prague, Czech Republic, 11–13 November 2019*; Revised Selected Papers 18; Springer: Cham, Switzerland, 2020; pp. 176–192.
32. Skorobogatov, S.P.; Anderson, R.J. Optical fault induction attacks. In *Cryptographic Hardware and Embedded Systems—CHES 2002, Proceedings of the 4th International Workshop, Redwood Shores, CA, USA, 13–15 August 2002*; Revised Papers 4; Springer: Berlin/Heidelberg, Germany, 2003; pp. 2–12.
33. Krček, M.; Ordas, T.; Picek, S. Short Paper: Diversity Methods for Laser Fault Injection to Improve Location Coverage. Cryptology ePrint Archive. Paper 2023/893. 2023. Available online: <https://eprint.iacr.org/2023/893> (accessed on 30 January 2024).
34. Anceau, S.; Bleuët, P.; Clédière, J.; Maingault, L.; Rainard, J.L.; Tucoulou, R. Nanofocused X-ray beam to reprogram secure circuits. In *Cryptographic Hardware and Embedded Systems—CHES 2017, Proceedings of the 19th International Conference, Taipei, Taiwan, 25–28 September 2017*; Springer: Cham, Switzerland, 2017; pp. 175–188.
35. Poschmann, A.; Moradi, A.; Khoo, K.; Lim, C.W.; Wang, H.; Ling, S. Side-channel resistant crypto for less than 2300 GE. *J. Cryptol.* **2011**, *24*, 322–345. [\[CrossRef\]](#)
36. Jati, A.; Gupta, N.; Chattopadhyay, A.; Sanadhya, S.K.; Chang, D. Threshold Implementations of GIFT: A Trade-Off Analysis. *IEEE Trans. Inf. Forensics Secur.* **2019**, *15*, 2110–2120. [\[CrossRef\]](#)

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.