Algebraic Persistent Fault Analysis of SKINNY_64 Based on S_Box Decomposition

Xing Fang 1, Hongxin Zhang 1,2,3,*, Danzhi Wang 1, Hao Yan 1, Fan Fan 4 and Lei Shu 4

1 School of Electronic Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China
2 Beijing Key Laboratory of Work Safety Intelligent Monitoring, Beijing University of Posts and Telecommunications, Beijing 100876, China
3 Beijing Key Laboratory of Work Safety Intelligent Monitoring, School of Electronic Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China
4 Beijing Microelectronics Technology Institute, Beijing 100076, China
* Correspondence: hongxinzhang@bupt.edu.cn

Abstract: Algebraic persistent fault analysis (APFA), which combines algebraic analysis with persistent fault attacks, brings new challenges to the security of lightweight block ciphers and has received widespread attention since its introduction. Threshold Implementation (TI) is one of the most widely used countermeasures for side channel attacks. Inspired by this method, the SKINNY block cipher adopts the S_box decomposition to reduce the number of variables in the set of algebraic equations and the number of Conjunctive Normal Form (CNF) equations in this paper, thus speeding up the algebraic persistent fault analysis and reducing the number of fault ciphertexts. In our study, we firstly establish algebraic equations for full-round faulty encryption, and then analyze the relationship between the number of fault ciphertexts required and the solving time in different scenarios (decomposed S_boxes and original S_box). By comparing the two sets of experimental results, the success rate and the efficiency of the attack are greatly improved by using S_box decomposition. In this paper, We can recover the master key in a minimum of 2000s using 11 pairs of plaintext and fault ciphertext, while the key recovery cannot be done in effective time using the original S_box expression equations. At the same time, we apply S_box decomposition to another kind of algebraic persistent fault analysis, and the experimental results show that using S_box decomposition can effectively reduce the solving time and solving success rate under the same conditions.

Keywords: S_box; decomposition; persistent fault; algebraic equation; key residual entropy

1. Introduction

With the development of Internet of Things (IoT) and chip technology, the security of information is becoming a key concern for people in practical production and life, and cryptography is gradually gaining popularity. In a resource-constrained environment, some lightweight block ciphers [1] have emerged in order to improve the security of information. They have the advantages of simple structure, high efficiency, and easy implementation. Common lightweight block ciphers include PRESENT, GIFT, SKINNY [2], LED, etc. These encryption ciphers are all substitution-permutation network (SPN) structures, where S_box substitution as a nonlinear operation plays a crucial role in the security of the ciphers. Therefore, it is important to conduct research on lightweight block ciphers and their S_boxes.

Fault attack, as a common side channel analysis method, has been receiving attention and research from scholars and experts since its inception. The main ways of fault attack [3] are voltage fault attack, electromagnetic fault attack, laser fault attack [4,5], temperature fault attack, etc. The purpose of these approaches is to alter the surroundings of the working cryptographic chip and change the encryption result by injecting an abnormal state. This concept was put forward by Boneh et al. on the RSA-CRT (Rivest-Shamir-Adleman China
Remainder Theorem) algorithm in 1996. Fault attacks can also be divided into three types: transient faults, persistent faults, and permanent faults. In 2018, Zhang Fan et al. proposed a persistent fault analysis method. Their research injects faults into the S_box lookup tables of AES, PRESENT, and other algorithms, which makes errors occur when accessing the lookup tables, thus generating fault ciphertexts. The master key can be recovered by analyzing this fault injection method and the fault ciphertexts.

As a masking method, Threshold implementation (TI) [6–8] is a kind of side-channel attack countermeasure against first-order Differential Power Analysis (DPA) [9]. This method was first proposed by Nikova et al. in 2006. In 2011, Poschmann et al. proposed an algebraic decomposition method that decomposes the cubic S_box into two quadratic S_boxes [10]. This method was originally proposed to reduce the mask component, reduce circuit complexity, and reduce additional hardware resources. By this method, the original S_box algebraic system of equations [11] can be equivalently replaced by a new system of two algebraic equations.

1.1. Related Works

The current common analysis methods for persistent fault [12,13] are mainly classical persistent fault analysis (PFA) [14,15], enhanced persistent fault analysis (EPFA) [16], and persistent fault-Based collision analysis (PFCA) [17]. For the SKINNY block cipher, classical persistent fault analysis cannot recovery the master key. EPFA requires 1500–1600 fault ciphertexts to recover the master key. When using the PFCA method, we need plaintext selection depending on the algorithm structure and a large number of fault ciphertexts. This method requires a more complex application scenario compared to other methods. In order to drastically reduce the number of fault ciphertexts required by the attack, we try to introduce algebraic analysis methods into the persistent fault analysis.

In our previous research, we performed an algebraic fault analysis [18] based on S_box decomposition for the SKINNY block cipher. By performing the S_box decomposition, we can represent the algebraic characteristics of the S_box using a smaller number of CNF equations and variables. The experimental results show that the experimental efficiency is substantially improved after S_box decomposition. Therefore, we try to introduce this method when performing algebraic persistent fault analysis to improve the speed and success rate of solving.

1.2. Our Contributions

In this paper, we propose algebraic persistent fault analysis methods for the SKINNY block cipher based on S_box decomposition. This is first work that combines S_box decomposition methodology and algebraic analysis to recover the master key of the SKINNY cipher under the condition of a persistent fault in S_box. Our main contributions are as follows.

- For the SKINNY block cipher, its S_box is the four-in-four-out type, and the output four-bit value can be represented by an algebraic equation of the four-bit input value. When there is a fault in the S_box lookup table, the original algebraic equation cannot represent the output result, and an algebraic representation using the changed set of algebraic equations is required. We give the distribution of the number of variables in the algebraic equations of S_box by traversing all possible single faults in the S_box;
- In this paper, we propose an algebraic persistent fault analysis method based known plaintexts(KP-APFA) with all rounds encryption. The attack is first attempted using the original faulty S_box algebraic expression. The experimental results show that the attack cannot complete key-recovery within the specified time;
- To achieve key-recovery of the SKINNY cipher, we introduce the S_box decomposition method and combine it with the KP-APFA method to analyze the SKINNY cipher, which can solve the key in 2000 s with at least 11 pairs of plaintext and faulty ciphertext. This reduces the number of fault samples by more than 100 times compared to the EPFA method;
• A constraint-based algebraic persistent fault analysis method was proposed by Zhang Fan et al. In this paper, the S\textsubscript{box} decomposition is combined with this method (referred to as SD-APFA), and the experimental results show that the solving speed and the success rate of solving in the specified time are improved, and the best case can improve the solving speed by more than 10 times. In addition, the relationship between key residual entropy, fault depth, and number of faults is further investigated in this paper.

The remainder of this article is organized as follows. We describe the algorithmic structure of SKINNY in Section 2. Afterward, the persistent attack on s\textsubscript{box} and the known-plaintext APFA method are presented in Section 3. We introduce the S\textsubscript{box} decomposition methodology of SKINNY and simulation experiments of KP-APFA based on S\textsubscript{box} decomposition in Section 4. In Section 5, we introduce the methodology of S\textsubscript{box} decomposition to APFA (SD-APFA) and provide the attack results of different methods in several scenarios. Meanwhile, we further investigate the relationship between key residual entropy, fault depth, and the number of faulty ciphertexts. We give the experimental setup and results in Section 6, followed by the Conclusions in Section 7.

2. Algorithmic Description of SKINNY

The SKINNY block cipher is a lightweight AES-like tunable block cipher with a novel SPN structure, proposed by Beierle et al. at CRYPTO 2016. SKINNY is a class of tunable block cipher with tunable key framework, which is divided into six different versions according to the tunable key size and block length. In this paper we choose the most common version SKINNY\textsubscript{64_64} as the research object.

Each encryption round of the SKINNY block cipher includes operations such as Subcells, Addconstants, AddroundTweakey, Shiftrows, and Mixcolumns. The single-round encryption process is shown in Figure 1. The number \( r \) of rounds to be performed during encryption depends on the block and tweakey sizes. For the SKINNY\textsubscript{64_64} version of the block cipher, its number of encryption rounds are 32.

![Figure 1. The structure of SKINNY\textsubscript{64}.](image)

• Subcells

Subcells are the only non-linear operation in the entire encryption process. The hexadecimal notation of this S\textsubscript{box} is given by the following Table 1.

| \( X \)   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
|----------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| \( S[X] \) | c | 6 | 9 | 0 | 1 | a | 2 | b | 3 | 8 | 5 | d | 4 | e | 7 | f |

The S\textsubscript{box} is a four-in and four-out type, and the four-bit values of output are related to the input four-bit values. Let the input of S\textsubscript{box} be \( x_3 \parallel x_2 \parallel x_1 \parallel x_0 \) and the output of S\textsubscript{box} be \( y_3 \parallel y_2 \parallel y_1 \parallel y_0 \), the algebraic relationship between them can be expressed by the following algebraic equations:
\[ Y(x_3, x_2, x_1, x_0) = (y_3, y_2, y_1, y_0) \]
\[ y_0 = x_1 + x_2 + x_3 + x_0 x_1 + x_0 x_2 + x_0 x_3 + x_1 x_3 + x_0 x_1 x_2 + x_1 x_2 x_3 \]
\[ y_1 = x_0 + x_3 + x_0 x_1 + x_1 x_2 + x_2 x_3 + x_1 x_2 x_3 \]  
(1)
\[ y_2 = 1 + x_1 + x_2 + x_3 + x_1 x_2 \]
\[ y_3 = 1 + x_0 + x_2 + x_3 + x_2 x_3 \]

Observation of Equation (1) reveals that the original S-box algebraic equation uses a total of eight quadratic and quadratic+ variables, which are \( x_0 x_1, x_0 x_2, x_0 x_3, x_1 x_2, x_1 x_3, x_2 x_3, x_0 x_1 x_2, x_1 x_2 x_3 \).

- **AddConstants**
  The constants of the SKINNY block cipher are generated through a 6-bit affine LFSR (Linear Feedback Shift Register), whose state is updated by following definition:
  \[
  (rc_5, rc_4, rc_3, rc_2, rc_1, rc_0) \leftarrow (rc_4, rc_3, rc_2, rc_1, rc_5 \oplus rc_4 \oplus 1) \]  
(2)

The initial value of these 6 bits is set to 0, which are updated before use in a given round. The bits from the LFSR are arranged into a \( 4 \times 4 \) array (only the first column of the state is affected by the LFSR bits):

\[
\begin{bmatrix}
  c_0 & 0 & 0 & 0 \\
  c_1 & 0 & 0 & 0 \\
  c_2 & 0 & 0 & 0 \\
  0 & 0 & 0 & 0 \\
\end{bmatrix}
\]  
(3)

with \( c_2 = 0 \times 2 \), \( (c_0, c_1) = (rc_3 \parallel rc_2 \parallel rc_1 \parallel rc_0, 0 \parallel 0 \parallel rc_5 \parallel rc_4) \).

- **AddRoundTweakey**
  The first and second rows of all tweakey arrays are extracted and bitwise exclusive-ored to the cipher internal state, respecting the array positioning. The specific subkey generation method can be found in Ref. [2].

- **ShiftRows**
  This operation can be represented as a permutation. A permutation \( P \) is applied on the cells positions of the cipher internal state cell array: for all \( 0 \leq i \leq 15 \), the operation can be showed as \( P = [0, 1, 2, 3, 7, 4, 5, 6, 10, 11, 8, 9, 13, 14, 15, 12] \).

- **MixColumns**
  This operation can mix each column by multiplication. The matrix \( M \) of the multiplication is shown as follows:
  \[
  \begin{bmatrix}
  1 & 0 & 1 & 1 \\
  1 & 0 & 0 & 0 \\
  0 & 1 & 1 & 0 \\
  1 & 0 & 1 & 0 \\
  \end{bmatrix}
  \]  
(4)

3. Persistent Fault Injection in S_Box
   The cryptographic world has never stopped attacking the SKINNY block cipher since this cipher’s inception. The SKINNY has good security properties and can bring security to information under resource-constrained conditions. Since Zhang Fan et al. proposed the persistent fault analysis method, persistent fault analysis on lightweight cryptographic algorithms has become one of the attack trends. Persistent fault analysis is the injection of faults into the S_box lookup table, which causes errors to occur when accessing specific S_box units. Selecting the appropriate fault injection point is one of the main focuses of the study.

Injecting a single fault in the S_box gives a total of \( 16 \times 15 = 240 \) possibilities. Our study traverses all possibilities and gives the changed set of S_box equations. By counting
the number of intermediate variables in the changed set of equations, the least number of cases is selected for building the equation.

Algorithm 1 is used to iterate through all individual persistent faults of the S_box and generate a system of algebraic fault equations for the S_box and calculate the number of higher-order variables in the system of equations. It is worth stating that the values of the following statistics are the number of variables that are not duplicated. A statistical table of the number of quadratic and quadratic+ variables in the original S_box for different persistent fault scenarios is given in Table 2.

Algorithm 1: Pseudocode for calculating the number of higher-order variables in a system of equations for an S_box

| Input: $S[X]$ | Output: $N[i][j]$ |
|---------------|------------------|
| $i = 0, j = 0$; | for $i = 0; i \leq 15; i + +$ do |
| for $j = 0; j \neq S[i]; i \leq 15; j + +$ do |
| Generate the equations of $S^*$; |
| Calculate the number of quadratic and quadratic+ variables in equations $N[i][j]$; |
| end |
| end |
| return $N[i][j]$; |

Table 2. Numbers of quadratic and the above variables under different faults in the original S_box.

|     | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | a  | b  | c  | d  | e  | f  |
|-----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| F[0]| 11 | 11 | 11 | 11 | 11 | 10 | 11 | 11 | 11 | 10 | 11 | 11 | —  | 8  | 11 | 11 |
| F[1]| 11 | 11 | 11 | 11 | 11 | 10 | —  | 8  | 11 | 11 | 11 | 11 | 11 | 11 | 11 | 11 |
| F[2]| 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 9  | —  | 7  | 10 | 10 | 10 | 10 | 9  |
| F[3]| —  | 9  | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| F[4]| 8  | —  | 9  | 10 | 10 | 10 | 10 | 10 | 9  | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| F[5]| 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| F[6]| 9  | 8  | —  | 8  | 8  | 9  | 9  | 9  | 9  | 9  | 9  | 9  | 9  | 9  | 9  | 9  |
| F[7]| 9  | 9  | 9  | 9  | 9  | 9  | 9  | 9  | 9  | 9  | 9  | 8  | —  | 9  | 9  | 9  |
| F[8]| 9  | 11 | 10 | —  | 11 | 11 | 11 | 11 | 11 | 11 | 10 | 11 | 11 | 11 | 11 | 11 |
| F[9]| 11 | 11 | 11 | 11 | 11 | 11 | 11 | 11 | 11 | 11 | 11 | 11 | 11 | 11 | 11 | 11 |
| F[a]| 10 | 10 | 10 | 10 | 10 | —  | 8  | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| F[b]| 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| F[c]| 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| F[d]| 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| F[e]| 9  | 9  | 9  | 9  | 8  | 9  | 9  | 9  | 9  | 9  | 9  | 9  | 9  | 9  | 9  | 9  |
| F[f]| 9  | 9  | 9  | 9  | 9  | 9  | 9  | 9  | 9  | 9  | 9  | 9  | 9  | 9  | 9  | 9  |

According to previous research experience, the more quadratic and above variables in the S_box equations, the slower the solving speed will be. Therefore, when injecting fault into the look-up table, in order to improve the solution efficiency, the number of intermediate variables in the modified equation sets should be as small as possible. When the persistent fault is injected into the S_box lookup table and makes $F[2] = 9$ into $F[2] = A$, the number of quadratic and quadratic+ variables is 7, and its corresponding equation expression becomes:
\[ Y(x_3, x_2, x_1, x_0) = (y_3, y_2, y_1, y_0) \]
\[
\begin{align*}
y_0 &= x_2 + x_3 + x_0 x_2 + x_0 x_3 + x_1 x_2 + x_0 x_1 x_3 + x_0 x_1 x_2 x_3 \\
y_1 &= x_0 + x_1 + x_3 + x_2 x_3 + x_0 x_1 x_2 + x_0 x_1 x_3 + x_0 x_1 x_2 x_3 \\
y_2 &= 1 + x_1 + x_2 + x_3 + x_1 x_2 \\
y_3 &= 1 + x_0 + x_2 + x_3 + x_2 x_3 
\end{align*}
\] (5)

Algorithm 2 is used to generate the faulty ciphertext. \( P \) and \( f \) are the inputs to the Algorithm 2, representing plaintext and fault, respectively. The output of Algorithm 2 is the fault ciphertext \( C^* \). The function of Algorithm 2 is equivalent to simulating a persistent fault injection experiment.

**Algorithm 2: The fault ciphertext generation of SKINNY_64**

| Input: | \( P, f \) |
|---|---|
| Output: | \( C^* \) |
| 1 | \( i = 0, r = 32; \) |
| 2 | \( S[i]^* = S[i] \oplus f; \) |
| 3 | **while** \( i \leq r - 1 \) **do** |
| 4 | \( X = \text{Subcells}^*(P); \) |
| 5 | \( X = \text{Addconstants}(X); \) |
| 6 | \( X = \text{AddRoundTweakey}(X); \) |
| 7 | \( X = \text{ShiftRows}(X); \) |
| 8 | \( P = \text{MixColumns}(X); \) |
| 9 | **end** |
| 10 | \( C^* = P; \) |
| 11 | **return** \( C^* \) |

In Algorithm 3, we give a pseudocode for algebraic persistent fault analysis based on known plaintext, referred to as KP-APFA. The input \( N \) in the algorithm represents the number of faults, i.e., the number of plaintext and fault ciphertext pairs. The output in the algorithm is the solving time. For the SKINNY block cipher, there is a constant algebraic relationship between the subkey and the master key for each round. We represent the faulty encryption process in the form of algebraic equations and convert all the useful information into the form of a system of CNF equations. Then, all the CNF equations are combined to perform the key solution in the \( T_{sol} = \text{RunAPFA}() \) means using CryptoMiniSAT to get the time of key-recovery.

We use the CryptoMiniSAT for key-recovery, which requires converting the algebraic equations into the form of CNF equations. For correct encryption, each round of S_box can be represented with 192 variables and 480 CNF equations, the round-constant is generated by a 6-bit affine LFSR, it can be represented with 6 variables and 6 CNF equations. Considering the operations of the Addconstants, Subkey, AddRoundTweakey, ShiftRows, and MixColumns, each round of them can be represented with 320 variables and 320 CNF equations.

Equation (5) gives the set of algebraic equations for the S_box after the persistent fault makes \( F[2] = 9 \) into \( F[2] = A \). For this faulty encryption, each round of fault S_box can be represented with 176 variables and 464 CNF equations. Considering the operations of the Addconstants, AddRoundTweakey, ShiftRows, and MixColumns, each round of them can be represented with 256 variables and 256 CNF equations.

Use the above method to build a set of equations for persistent algebraic fault analysis. Experiments were set up using 16, 18, 20, and 30 random plaintexts, respectively, and for the purpose of discussing the generality of the experiments, 50 samples were randomly generated for each set of experiments. The maximum solving time is set to 1 h in the experiment, and the solution is judged to fail if it exceeds 1 h. The results of the study
show that all samples cannot complete the recovery of the key within the specified time, therefore, the key-recovery cannot be completed by using the original S_box directly. We need to improve the expression of S_box algebraic equations.

Algorithm 3: The KP-APFA for SKINNY_64

|   |   |
|---|---|
| **Input:** | $N$ |
| **Output:** | $T_{sol}$ |
| 1 | $\text{Random}(P_i)$ | //Randomly generate $N$ plaintexts; |
| 2 | Generation equation set for $P_i$; |
| for $i = 0; i < N; i++$ do | |
| 4 | $C^*_i = \text{Algorithm 2}(P_i)$; |
| end |
| 6 | Generation equation set for $C^*_i$; |
| for $rc = 1; rc < 32; rc++$ do | |
| 8 | Generation equation set for sub-key |
| 9 | Generation equation set for above operations; |
| for $i = 0; i < N; i++$ do | |
| 11 | for $rc = 1; rc < 32; rc++$ do |
| 12 | $X = \text{Subcells}^*(P)$; |
| 13 | Generation equation set for Subcells with faulty S_box; |
| 14 | $X = \text{Addconstants}(X)$; |
| 15 | $X = \text{AddRoundTweakey}(X)$; |
| 16 | $X = \text{ShiftRows}(X)$; |
| 17 | $P = \text{MixColumns}(X)$; |
| 19 | Generation equation set for above operations; |
| 20 | end |
| 21 | $T_{sol} = \text{Run APFA}()$; |
| 22 | return $T_{sol}$ |

4. S_Box Decomposition of SKINNY

This section gives the general flow of algebraic persistent fault analysis based on S_box decomposition. Firstly, the S_box in the target algorithm is decomposed and a suitable decomposition scheme is selected. Then, all possible individual S_box persistent faults are traversed and a suitable fault injection scheme is selected. The key-recovery is finally done by simulating the fault injection to generate the faulty ciphertexts and build the cryptographic algebraic system of equations and other useful information using the CryptoMiniSAT. The specific flowchart is given in Figure 2.

![Figure 2. The flowchart of algebraic persistent fault analysis based on S_box decomposition.](image)

**Algorithm 2**: The KP-APFA for SKINNY_64

- **Input:** $N$
- **Output:** $T_{sol}$
- $\text{Random}(P_i)$ //Randomly generate $N$ plaintexts;
- Generation equation set for $P_i$;
- for $i = 0; i < N; i++$ do
  - $C^*_i = \text{Algorithm 2}(P_i)$;
- Generation equation set for $C^*_i$;
- for $rc = 1; rc < 32; rc++$ do
  - Generation equation set for sub-key
  - Generation equation set for above operations;
- for $i = 0; i < N; i++$ do
  - for $rc = 1; rc < 32; rc++$ do
    - $X = \text{Subcells}^*(P)$;
    - Generation equation set for Subcells with faulty S_box;
    - $X = \text{Addconstants}(X)$;
    - $X = \text{AddRoundTweakey}(X)$;
    - $X = \text{ShiftRows}(X)$;
    - $P = \text{MixColumns}(X)$;
  - Generation equation set for above operations;
- $T_{sol} = \text{Run APFA}()$;
- return $T_{sol}$
In our previous research on the SKINNY_64 block cipher for algebraic fault analysis, we found that the use of S-box decomposition can greatly improve the speed of the key-recovery. We can decompose the original cubic S-box into two quadratic S-boxes to reduce the number of CNF clauses in the set of algebraic equations and the number of quadratic and quadratic+ intermediate variables introduced due to the nonlinear operation S-box. A schematic diagram of the decomposition is given in Figure 3. In 2011, Poschmann et al. proposed a technique to decompose a cubic S-box function into two quadratic functions. This relation can be expressed by the following equations $S(X) = H(F(X))$, where $S, F, H : GF(2)^4 \rightarrow GF(2)^4$. Considering the input and output vectors $X = (x, y, z, w)$ and $F(X) = (f_0(X), f_1(X), f_2(X), f_3(X))$. Each $f_i$, as a quadratic Boolean function, can be represented in ANF as the following equation, where $a_i, a_{ij}$ are the binary coefficients of the Boolean function:

\[
f_i(x, y, z, w) = a_{0} + a_{1}x + a_{2}y + a_{3}z + a_{4}w + a_{12}xy + a_{13}xz + a_{14}xw + a_{23}yz + a_{24}yw + a_{34}zw
\]

This can be directly substituted for 15 nonzero values for $X$, as a quadratic Boolean function, can be represented in ANF as the following equation, where $a_{ij}$ are the binary coefficients of the Boolean function:

\[
f_i(x, y, z, w) = a_{1}x + a_{2}y + a_{3}z + a_{4}w + a_{12}xy + a_{13}xz + a_{14}xw + a_{23}yz + a_{24}yw + a_{34}zw
\]

As discussed in literature [10], in order to reduce the overall search space for the two decomposed functions of $F$ and $H$, the following two facts were used in this paper:

- Rewriting $S(X) = H(F(X))$ as $S(F^{-1}(X)) = H(X)$, one needs to search only for all possible quadratic functions for $G(X)$. This is then used to compute the other quadratic functions $F(X)$ as $S(F^{-1}(X))$.
- Rewriting $S(X) = H(F(X))$ as $S(X) = H'(F'(X))$ where $F'(X) = F(X) + F(0)$ and $H'(X) = H(X + F(0))$. We assume that $G(0) = 0$ and get the other decompositions directly by substituting 15 nonzero values for $G(0)$. Therefore, we only need to vary the 10 nonconstant coefficients in the ANF and the search space is reduced to $(2^{10})^4 = 2^{40}$.

With the above refinements, the actual quadratic Boolean equation used for the search is shown below:

\[
f_i(x, y, z, w) = a_{1}x + a_{2}y + a_{3}z + a_{4}w + a_{12}xy + a_{13}xz + a_{14}xw + a_{23}yz + a_{24}yw + a_{34}zw
\]

The following steps were implemented in order to compute the desired optimized quadratic Boolean functions for $F$ and $H$:

- We first iterate through all combinations of coefficients of function $f_i$. For each combination of coefficients we iterate through all possible values of $X$ and check whether the Boolean equation is balanced. If the combination is balanced we then add it to a set $B$ for all possible coefficients, otherwise it is discarded;
- The corresponding $F(X)$ equation is calculated iteratively for each balanced coefficient in the set $B$ in a group of four;
- Check whether the computed $F(X)$ is a quadratic permutation or not. If yes, we compute the $H(X)$, otherwise we discard it;
- Check whether the computed $H(X)$ is a quadratic function or not. If yes, add both the $F(X)$ and $H(X)$ equations to the set $P$ of all the possible decompositions, otherwise discard them. By doing the above, we can obtain the possible decomposition of 19,862 pairs;
- Considering 15 other possible constant terms, we finally get 317,777 possibilities for the S-box decomposition of the SKINNY block cipher using the above steps.
Figure 4 gives the flowchart of the decomposition of the cubic S_box into two quadratic S_boxes.

We followed the method provided in the literature [16] for censoring and finally obtained the optimal decomposition scheme. The selected $F(X)$ and $H(X)$, satisfying all the three TI requirements, namely Correctness, Non-Completeness, and Uniformity, are shown in Table 3.

Table 3. The decomposed S_boxes.

| X | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| F[X] | 0 | 1 | 6 | 7 | d | c | f | e | 5 | 4 | 3 | 2 | 9 | 8 | b | a |
| H[X] | c | 6 | d | 5 | 8 | 3 | 9 | 0 | e | 4 | f | 7 | a | 1 | b | 2 |

The ANFs of first S_box can be represented by Equation (8).

$$F(w, z, y, x) = (f_3, f_2, f_1, f_0)$$
$$f_0 = x + z + xw + w$$
$$f_1 = y$$
$$f_2 = y + yz + z + w$$
$$f_3 = z$$

(8)

The ANFs of second S_box can be represented by Equation (9).

$$H(w, z, y, x) = (h_3, h_2, h_1, h_0)$$
$$h_0 = y + yz$$
$$h_1 = x + xy + w$$
$$h_2 = 1 + z$$
$$h_3 = 1 + x$$

(9)
By looking at the set of Equations (8) and (9) we find that by decomposing the S_box, the number of quadratic and quadratic+ intermediate variables introduced is changed from the original 8 to 2 + 2. Next, we perform the persistent fault injection for the first S_box. The number of quadratic and quadratic+ intermediate variables is calculated by traversing all fault injection cases. The specific experimental results are shown in Table 4. According to the traversal results given in Table 4 we can find that when the fault is injected into S[F], the fault S_box has the least number of introduced intermediate quadratic and quadratic+ variables. Combining the original S_box and the decomposed S_box1 with the fault-injection intermediate variable traversal table, the S-box decomposition is used to minimize the number of quadratic and quadratic + variables and the number of CNF equation sets that can be used to express algebraic persistent fault analysis with minimal CNF equations. Next, an example is used to illustrate. Injecting fault in S[F] causes S[F] = A to become S[F] = E in the first S_box after decomposing, and for the whole encryption process, this change causes S[F] = F to become S[F] = B.

Table 4. Numbers of quadratic and the above variables under different faults in decomposed S_box1.

|   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| F[0] | — | 10 | 11 | 11 | 10 | 11 | 11 | 11 | 11 | 11 | 11 | 11 | 11 | 11 | 11 | 11 |
| F[1] | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 |
| F[2] | 8 | 8 | 7 | 8 | 8 | 8 | — | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 |
| F[3] | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
| F[4] | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 |
| F[5] | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | — | 6 | 6 | 6 | 6 |
| F[6] | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 4 | 5 | 5 | 5 | 5 |
| F[7] | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| F[8] | 8 | 8 | 8 | 8 | 7 | — | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 |
| F[9] | 6 | 6 | 6 | 6 | — | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
| F[a] | 6 | 6 | 6 | — | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
| F[b] | 4 | 4 | — | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| F[c] | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| F[d] | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| F[e] | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| F[f] | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |

Table 5 is the decomposed fault S_box1. The corresponding algebraic equation for the fault S_box is shown below.

\[
F^*(w, z, y, x) = (f_3, f_2, f_1, f_0)
\]

\[
f_0 = x + z + xw + w
\]

\[
f_1 = y
\]

\[
f_2 = y + yz + z + w + xyzw
\]

\[
f_3 = z
\]

Table 5. The decomposed fault S_box1.

| X | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| F*[X] | 0 | 1 | 6 | 7 | d | c | f | e | 5 | 4 | 3 | 2 | 9 | 8 | b | e |

For the original S_box, this persistent fault is equivalent to the following expression.
Table 6 is the faulty original S_box. The corresponding algebraic equation for the fault of the original S_box is shown in Equation (11).

\[
Y(x_3, x_2, x_1, x_0) = (y_3, y_2, y_1, y_0)
\]

\[
y_0 = x_1 + x_2 + x_3 + x_0x_1 + x_0x_2 + x_0x_3 + x_1x_3 + x_0x_1x_2 + x_0x_1x_3
\]

\[
y_1 = x_0 + x_3 + x_0x_1 + x_1x_2 + x_1x_3 + x_2x_3 + x_1x_2x_3
\]

\[
y_2 = 1 + x_1 + x_2 + x_3 + x_1x_2 + x_0x_1x_2x_3
\]

\[
y_3 = 1 + x_0 + x_2 + x_3 + x_2x_3
\]

We use the decomposing S_boxes algebraic equations and the original S_box algebraic equations for persistent algebraic fault analysis experiments, respectively.

**Table 6.** The faulty original S_box.

| X | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| \( S^* [X] \) | c | 6 | 9 | 0 | 1 | a | 2 | b | 3 | 8 | 5 | d | 4 | e | 7 | b |

First, we set the number of faulty plaintexts to 30, 20, 18, and 16, respectively. Fifty samples are randomly generated in each scenario, and the two methods are compared in experiments under the same fault conditions. The experiment sets the maximum solving time of the solver to 1 h, and the attack is judged to have failed after 1 h. Table 7 gives the average solving time and success rate of the two methods for different scenarios. The experimental results show that the key can be solved in effective time after using S_box decomposition, while all experiments cannot complete the key-recovery in effective time when using the original S_box expression. Figure 5 gives a histogram of the distribution of the solving time when using S_box decomposition in different scenarios, where the horizontal coordinates represent the solving time and the vertical coordinates represent the frequency. When the number of faulty ciphertexts is 30, the average solving time is 784.9 s. When the number of faulty ciphertexts is reduced to 20 and 18, the average solving time decreases. This may be due to the information redundancy caused by the larger number of algebraic equations when the number of faulty ciphertexts is large, and thus the solving time is longer than when the number of faulty ciphertexts is 20 and 18. As the number of faulty ciphertexts decreases further, the average solving time increases. When the number of faulty ciphertexts is 16, the average solving time reaches 450.6 s. From the histogram, we can find that 48% of the samples can be solved for the key within 400 s.

**Table 7.** The results of the algebraic persistent fault analysis on SKINNY 64 under different scenarios.

| N (Number) | \( T_{ave} \) (Seconds) of Original S_Box | \( T_{ave} \) (Seconds) of New S_Boxes | Success Rate of Original S_Box | Success Rate of New S_Boxes |
|-----------|----------------------------------------|-----------------------------------|------------------|------------------|
| 30        | -                                      | 784.9                             | 0%               | 100%             |
| 20        | -                                      | 496.2                             | 0%               | 100%             |
| 18        | -                                      | 409.6                             | 0%               | 100%             |
| 16        | -                                      | 450.6                             | 0%               | 100%             |

The Table 8 shows that when the setting time is 2 h and the number of ciphertexts is 12, the success rate is 54%. When the maximum solving time is relaxed to 10 h, the success rate of the solving for the same experimental samples reaches 92%, where the shortest solving time is 459.5 s. In our study, in order to explore the minimum number of ciphertexts that can be used to achieve key-recovery, the number of ciphertexts is further reduced to 11, 10 sets of samples are randomly selected, the maximum solving time is set to 10 h, and the attack is judged to failure after 10 h. The experimental results show that 50% of the samples can be solved within the specified time, and the shortest solving time is 1930.9 s. Figure 6 gives the histogram of the solving time distribution for the number of faulty ciphertexts...
of 14, 13, and 12. From Figure 4, we can find that as the number of faults decreases from 14 to 12, the number of samples that complete key-recovery within 450 s decreases from 64% to 10%, while the number of samples that do not complete key-recovery within 7200 s increases from 6% to 46%.

![Figure 5](image_url)

**Figure 5.** Distribution of solving time under different scenarios of new S_boxes. (a) N = 30; (b) N = 20; (c) N = 18; (d) N = 16.

**Table 8.** The results of the new S_boxes on SKINNY_64 under different scenarios.

| N (Number) | $T_{ave}$ (Seconds) of New S_Boxes | Success Rate of New S_Boxes |
|------------|-----------------------------------|-----------------------------|
| 14         | 833.0                             | 94%                         |
| 13         | 2739.2                            | 84%                         |
| 12         | 3040.6                            | 54%                         |

![Figure 6](image_url)

**Figure 6.** Histogram comparison of solving the time distribution under different scenarios of SKINNY_64.
5. Applying Another Persistent Algebraic Analysis of SKINNY

A method for persistent algebraic fault analysis [19] was proposed by Fan Zhang et al. in 2022, and the SKINNY block cipher was studied in the literature. Inspired by this literature, the relationship between the number of faulty ciphertexts, fault depth, and key residual entropy is analyzed on the basis of the original research in my paper. Our paper adopts the method of S_box decomposition instead of the original S_box, and analyzes the differences between the two methods in the same scenario. In the literature [19], Zhang Fan et al. proposed a constraint-based method to solve the problem of unsolvability due to information redundancy in full-round encryption. The method is briefly described in the following.

Suppose the original value of the fault S_box $S'$ to be $V$. In the $r$-th ($1 \leq r \leq 32$) round function, $X'$ is the input of the $r$-th ($1 \leq r \leq 32$) round, $Y' = MC(SR(K' \oplus (AC(SC'(X')))\))$. $Y'$ contains the information of both $SC'(X')$ and $K'$, which means that we can use $Y'$ to add new constraints to $K'$, as shown in Equation (12).

\[
\begin{align*}
SC'[X_r] \neq V\\
Y' = MC(SR(K' \oplus (AC(SC'(X')))\))
\end{align*}
\]

\[
MC^{-1}(SR^{-1}(Y')) = SC'(X') \oplus AC^{-1}(K')
\]

\[\Rightarrow \tilde{K}_i' \neq \tilde{Y}_i' \oplus V, 0 \leq i \leq 15 \quad (12)\]

In Equation (12), $X'_i$ and $Y'_i$ are the 4 bits of $X'$ and $MC^{-1}(SR^{-1}(Y'))$, respectively. $\tilde{K}'_i$ are the 4 bits of $AC^{-1}(K')$ and $V$ is a constant. Based on this inequality relation, we establish constraints so as to reduce the key search space. The reader can check the specific steps in the literature [19].

Algorithm 4 gives the pseudo-code for the SKINNY block cipher based on S_box decomposition for algebraic persistent fault analysis. The input of the algorithm $N$ is the number of faulty ciphertexts, $FD$ is the fault depth, and the output $T_{sol}$ is the solving time of the solver. First the algorithm randomly generates $N$ plaintexts and uses Algorithm 2 to get the set of faulty ciphertexts, then all faulty ciphertexts are transformed into CNF equations. Next, we express the relationship between each round of subkeys and the master key as an algebraic equation and transform it into the form of CNF equations. Based on the input fault depth, we perform an algebraic representation of the FD rounds encryption process. We use two decomposed S_boxes instead of the original S_boxes, and transform the algebraic equations for each of these two S_boxes. Similarly, we translate the information about Addconstants, AddRoundTweakey, ShiftRows, MixColumns and constraints into the form of algebraic equations. Finally, we associate all the sets of CNF equations and use $T_{sol} = Run SD - APFA()$ for key-recovery. It is worth stating that the solver is set to work for 1 h, and the attack is considered to fail if it is exceeded. Fifty samples are randomly generated for each set of experiments, and the average solving time and solving success rate are calculated in Table 9.

Table 9 gives the average solving time and the success rate of the solving within the specified time for the two methods for different scenarios. $N$ represents the number of faulty ciphertexts and $FD$ represents the depth of the fault. Under different scenarios, we randomly generated 50 samples and set the maximum solving time to 1 h. When $N = 30$ and $FD = 4$, the use of the new S_box algebraic representation can improve about 18 times over the original method. When the number of faulty ciphertexts is 20, 18, 17, 16, and 14, both methods are able to recover the keys within the specified time. At the same time, the solving speed is improved using the new S_box algebraic equations compared to the original method. When the number of faulty ciphertexts is 13 and 12, the solving success rate and average solving time using the new method are better than the original method. From the above experimental results, it is shown that the solving speed and solving success rate can be effectively improved by using the new S_box algebraic expression method.
Algorithm 4: The SD-APFA for SKINNY_64

Input: \(N, FD\)
Output: \(T_{sol}\)

1. \(Random(P_i)\) //Randomly generate \(N\) plaintexts;

2. for \(i = 0; i < N; i + +\) do
   3. \(C_i^* = Algorithm2(P_i)\);
   4. end

5. Generation equation set for \(C_i^*\);
6. for \(rc = 1; rc < 32; rc + +\) do
   7. Generation equation set for sub-key
   8. end

9. for \(i = 0; i < N; i + +\) do
   10. for \(rc = 32 - FD; rc < 32; rc + +\) do
       11. Generation equation set for decomposed S_box1;
       12. Generation equation set for decomposed S_box2;
       13. Generation equation set for Addconstants;
       14. Generation equation set for AddRoundTweakey;
       15. Generation equation set for ShiftRows;
       16. Generation equation set for MixColumns;
       17. Generation equation set for constraints;
   18. end
19. end
20. \(T_{sol} = Run\ SD - APFA();\)
21. return \(T_{sol}\)

Table 9. Comparison table of the two methods in different scenarios.

| \(N\) | \(FD\) | \(T_{ave}\) (Seconds) of Original S_Box | \(T_{ave}\) (Seconds) of New S_Boxes | Success Rate of Original S_Box | Success Rate of New S_Boxes |
|------|------|----------------------------------------|--------------------------------------|-----------------------------|-----------------------------|
| 30   | 6    | 16.78                                  | 1.97                                 | 100%                        | 100%                        |
| 30   | 4    | 14.24                                  | 0.78                                 | 100%                        | 100%                        |
| 20   | 8    | 29.82                                  | 18.05                                | 100%                        | 100%                        |
| 20   | 7    | 24.80                                  | 18.26                                | 100%                        | 100%                        |
| 20   | 6    | 19.28                                  | 18.05                                | 100%                        | 100%                        |
| 18   | 8    | 46.06                                  | 20.63                                | 100%                        | 100%                        |
| 18   | 7    | 36.36                                  | 20.64                                | 100%                        | 100%                        |
| 18   | 6    | 28.23                                  | 19.77                                | 100%                        | 100%                        |
| 17   | 8    | 62.13                                  | 21.61                                | 100%                        | 100%                        |
| 17   | 7    | 44.48                                  | 25.21                                | 100%                        | 100%                        |
| 17   | 6    | 30.43                                  | 21.27                                | 100%                        | 100%                        |
| 16   | 8    | 90.53                                  | 33.12                                | 100%                        | 100%                        |
| 16   | 7    | 59.36                                  | 23.23                                | 100%                        | 100%                        |
| 16   | 6    | 40.86                                  | 26.17                                | 100%                        | 100%                        |
| 14   | 12   | 59.36                                  | 23.23                                | 100%                        | 100%                        |
| 14   | 10   | 332.79                                 | 128.73                               | 100%                        | 100%                        |
| 14   | 8    | 183.90                                 | 83.42                                | 100%                        | 100%                        |
| 13   | 12   | 1112.10                                | 508.57                               | 96%                         | 100%                        |
| 13   | 10   | 656.31                                 | 338.52                               | 94%                         | 98%                         |
| 12   | 14   | 1972.54                                | 847.60                               | 44%                         | 68%                         |
| 12   | 12   | 1255.55                                | 760.94                               | 56%                         | 72%                         |
| 12   | 10   | 1334.56                                | 956.65                               | 76%                         | 86%                         |

When using the solver for solving the system of CNF equations, multiple solutions exist because many intermediate variables are introduced and these intermediate variables have relatively few constraints. As for the key, each faulty ciphertext can constrain the key, so the value of the key may be the same. We take the first solution solved by the solver as
the object of study and analyze the relationship between the solved key and the correct key. If they are the same, the key is determined to be unique, and if they are different, the key residual entropy is determined to be greater than 0. In other words, the value of the key is not unique.

We analyze and discuss the relationship between the number of faulty ciphertexts, fault depth, the key residual entropy, and the success rate of solving in a specified time under the new method. From Table 10, we find that when the number of faulty ciphertexts is 30 and the fault depth is 6, the first solution obtained by all 50 samples is equal to the correct key, and the key residual entropy is determined to be 0 in this experiment. Meanwhile, when the fault depth is reduced to 4, only 84% of the samples obtain the first solution equal to the correct key, while the remaining 16% obtain the first solution with a different value from the correct key. As the number of faulty ciphertexts decreases, there is such a relationship between the fault depth and the residual entropy of the key. When the fault depth is shallow, the key residual entropy cannot be 0, at the same time, the fault depth must be increased to satisfy that the key residual entropy is 0.

Table 10. Table of the experimental results of new S_boxes under different scenarios.

| N  | FD | Key Residual Entropy = 0 | Success Rate of New S_Boxes |
|----|----|--------------------------|-----------------------------|
| 30 | 6  | 100%                     | 100%                        |
| 30 | 4  | 84%                      | 100%                        |
| 20 | 8  | 100%                     | 100%                        |
| 20 | 7  | 100%                     | 100%                        |
| 20 | 6  | 96%                      | 100%                        |
| 18 | 8  | 100%                     | 100%                        |
| 18 | 7  | 100%                     | 100%                        |
| 18 | 6  | 96%                      | 100%                        |
| 17 | 8  | 100%                     | 100%                        |
| 17 | 7  | 98%                      | 100%                        |
| 17 | 6  | 86%                      | 100%                        |
| 16 | 9  | 100%                     | 100%                        |
| 16 | 8  | 96%                      | 100%                        |
| 16 | 6  | 76%                      | 100%                        |
| 14 | 12 | 100%                     | 100%                        |
| 14 | 10 | 98%                      | 100%                        |
| 14 | 8  | 88%                      | 100%                        |
| 13 | 12 | 100%                     | 100%                        |
| 13 | 10 | 98%                      | 98%                         |
| 12 | 14 | 68%                      | 68%                         |
| 12 | 12 | 72%                      | 72%                         |
| 12 | 10 | 76%                      | 86%                         |

Figure 7 gives the relationship between the number of faulty ciphertexts and the minimum fault depth when the key residual entropy is 0 in the SKINNY_64 block cipher. We can find that the minimum fault depth decreases gradually as the number of fault ciphertexts increases, in other words, there is an inverse relationship between the minimum fault depth and the number of fault ciphertexts. When the number of faulty ciphertexts is small, if the fault depth is shallow, there are fewer constraints on the key information, so there are more keys satisfying the algebraic equations and the first solution given by the solver is different from the correct key. When deepening the fault depth, the generated set of algebraic equations has more constraints on the key, which improves the possibility of the key being the unique solution.
6. Experimental Setup and Results

In this section, we will introduce our setup of the experiments and the comparison results with a variety of existing methods.

6.1. Experimental Setup

In our experiment, we simulate the fault injection experiment via software, and use the CryptoMiniSAT v5.8.0 as the solver to solve the algebraic equations using Ubuntu 18.04.5 on Windows. We implement the experiments on a PC that has 16 GB memory and Intel(R) Core(TM) i5-9500 CPU at 3 GHz. The operating system is a 64 bit Windows 10.

6.2. Results

Table 11 provides a variety of existing methods to persistent fault analysis of SKINNY_64. The relationship between precondition, fault depth, minimum number of faults, and key residual entropy for different methods is given in the table.

Table 11. Comparison with existing persistent-fault-analysis on SKINNY_64.

| Methods       | Precondition       | FD       | Minimum Number of Faults | Key Residual Entropy |
|---------------|--------------------|----------|--------------------------|----------------------|
| PFA           | Ciphertext only    | 1        | ≈100                     | 32                   |
| EPFA          | Ciphertext only    | 4        | 1500–1600                | 0                    |
| APFA          | Ciphertext only    | 6–14     | 10                       | 0                    |
| KP-APFA (this paper) | Known           | 32       | 11                       | 0                    |
| SD-APFA (this paper) | Plaintext/ciphertext | 6–14   | 10                       | 0                    |

From Table 11 we can see that by using the PFA method, the key cannot be fully recovered because the fault depth is 1. The residual entropy of the key is 32. The EPFA method uses multiple rounds of faulty information and can recover the entire key. A minimum of 1500–1600 faulty ciphertexts are required to complete key-recovery. In this paper, the proposed KP-APFA based on S_box decomposition can complete key-recovery.
using a minimum of 11 faults, and unlike other methods, plaintexts need to be provided. The SD-APFA method proposed in this paper based on APFA also requires a minimum of 10 fault ciphertexts, but we can find that it improved the solving speed and solving success rate in the same attack scenarios as in Figure 8 and Table 9.

Figure 8. Comparison of the solving time with the two methods in different scenarios.

7. Conclusions

In this paper, we combine S_box decomposition methods commonly found in the field of threshold implementation with algebraic analysis. A more suitable persistent fault injection scheme is found from the perspective of algebraic analysis. Under the condition of known plaintexts, we have conducted several sets of experiments on the KP-APFA method based on S_box decomposition, and the average solving time and solving success rate are given for a different number of faults. The key-recovery can be completed within 2000s using at least 11 faults. Meanwhile, this paper also combines the method of S_box decomposition with the APFA method, and the proposed SD-APFA method has been significantly improved in both solving speed and success rate. Meanwhile, we discuss the relationship between key residual entropy, number of faults, and fault depth in different methods. In summary, the simplification of the system of equations using the S_box decomposition technique to achieve the S_box substitution operation is beneficial to improve the solving speed of the CryptoMiniSAT, thus improving the attack efficiency of the attacker in persistent fault analysis. In future work, we will apply and generalize this approach on other lightweight block ciphers.

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

Funding: This work was supported by the National Natural Science Foundation of China (62071057), the Fundamental Research Funds for the Central Universities (2019XD17), Acronautical Science Foundation of China (2019ZG073001).

Institutional Review Board Statement: Not applicable.

Data Availability Statement: If necessary, the data can be requested from the author of the communication.
Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the study design, data collection, analysis and interpretation, manuscript writing, and decision to publish results.

References

1. Vaithiyanathan, M.P. A survey on lightweight ciphers for IoT devices. In Proceedings of the International Conference on Technological Advancements in Power and Energy (TAP Energy), Kollam, India, 21–23 December 2017; pp. 1–4.

2. Beierle, C.; Jean, J.; Kößl, 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, Part II; Springer: Berlin/Heidelberg, Germany, 2016; pp. 123–153.

3. Jovanovic, P.; Kreuzer, M.; Polian, I. A fault attack on the led block cipher. In International Workshop on Constructive Side-Channel Analysis and Secure Design; Springer: Berlin/Heidelberg, Germany, 2012; pp. 120–134.

4. Courbon, F.; Loubet-Moundi, P.; Fournier, J.J.; Tria, A. Adjusting laser injections for fully controlled faults. In International Workshop on Constructive Side-Channel Analysis and Secure Design; Springer: Berlin/Heidelberg, Germany, 2014; pp. 229–242.

5. Canivet, G.; Maistri, P.; Leveugle, R.; Clediere, J.; Valette, F.; Renaudin, M. Glitch and laser fault attacks onto a secure aes implementation on a sm-based fpga. J. Cryptol. 2011, 24, 247–268. [CrossRef]

6. Kutzner, S.; Nguyen, P.H.; Poschmann, A.; Wang, H. On 3-share Threshold Implementations for 4-bit S-boxes. In Proceedings of the International Workshop Constructive Side-Channel Analysis Secure Design, Paris, France, 6–8 March 2013; Springer: Berlin/Heidelberg, Germany, 2013; pp. 99–113.

7. Bilgin, B.; Nikova, S.; Nikov, V.; Rijmen, V.; Stütz, G. Threshold implementations of all 3 × 3 and 4 × 4 S-boxes. In Proceedings of the CHES, Leuven, Belgium, 9–12 September 2012; Springer: Berlin/Heidelberg, Germany, 2012; pp. 76–91.

8. Jati, A.; Gupta, N.; Chattopadhyay, A.; Sanadhya, S.K.; Chang, D. Threshold Implementations of GIFT: A Trade-Off Analysis. IEEE Trans. Inf. Forensics Secur. 2020, 15, 2110–2120. [CrossRef]

9. Amiel, F.; Clavier, C.; Tunstall, M. Fault analysis of dpa-resistant algorithms. In International Workshop on Fault Diagnosis and Tolerance in Cryptography; Springer: Berlin/Heidelberg, Germany, 2006; pp. 223–236.

10. 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]

11. Knudsen, L.R.; Miolane, C.V. Counting equations in algebraic attacks on block ciphers. Int. J. Inf. Secur. 2010, 9, 127–135. [CrossRef]

12. Zhang, F.; Lou, X.; Zhao, X.; Basins, S.; He, W.; Ding, R.; Qureshi, S.; Ren, K. Persistent fault analysis on block ciphers. IACR Trans. Cryptogr. Hardw. Embed. Syst. 2018, 150–172. [CrossRef]

13. Zhang, F.; Zhang, Y.; Jiang, H.; Zhu, X.; Basins, S.; Zhao, X.; Liu, Z.; Gu, D.; Ren, K. Persistent fault attack in practice. IACR Trans. Cryptogr. Hardw. Embed. Syst. 2020, 172–195. [CrossRef]

14. Carré, S.; Guilley, S.; Rioul, O. Persistent Fault Analysis with Few Encryptions. In Constructive Side-Channel Analysis and Secure Design, COSADE; Lecture Notes in Computer Science; Bertoni, G.M., Regazzoni, F., Eds.; Springer: Cham, Switzerland, 2021; Volume 12244.

15. Pan, J.; Zhang, F.; Ren, K.; Bhasin, S. One Fault is All it Needs: Breaking Higher-Order Masking with Persistent Fault Analysis. In Proceedings of the 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), Florence, Italy, 25–29 March 2019; pp. 1–6.

16. Joshi, P.; Mazumdar, B. ExtPFA: Extended Persistent Fault Analysis for Deeper Rounds of Bit Permutation Based Ciphers with a Case Study on GIFT. In Security, Privacy, and Applied Cryptography Engineering. SPACE; Lecture Notes in Computer Science; Batina, L., Picek, M., Mondal, M., Eds.; Springer: Cham, Switzerland, 2020; Volume 12586.

17. Zheng, S.; Liu, X.; Zang, S.; Deng, Y.; Huang, D.; Ou, C. A Persistent Fault-Based Collision Analysis Against the Advanced Encryption Standard. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 2021, 40, 1117–1129. [CrossRef]

18. 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]

19. Zhang, F.; Feng, T.; Li, Z.; Ren, K.; Zhao, X. Free Fault Leaksages for Deep Exploitation: Algebraic Persistent Fault Analysis on Lightweight Block Ciphers. IACR Trans. Cryptogr. Hardw. Embed. Syst. 2022, 2, 289–311. [CrossRef]