Article

Not peer-reviewed version

# General Extensions and Improvements of Algebraic Persistent Fault Analysis

Hanbing Li , Kexin Qiao * , Ye Xu , Changhai OU , An Wang

Posted Date: 7 April 2025

doi: 10.20944/preprints202504.0526.v1

Keywords: persistent fault; fault attack; S-box; algebraic representation; PRESENT; SKINNY; CRAFT

*Article*

# General Extensions and Improvements of Algebraic Persistent Fault Analysis

**Hanbing Li [1,2], Kexin Qiao [1,*], Ye Xu [2], Changhai Ou [3] and An Wang [1]**

[1] School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing 100081, China
[2] No.208 Research Institute of China Ordnance Industries, Beijing 102202, China
[3] School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China
* Correspondence: qiao.kexin@bit.edu.cn

**Abstract:** Algebraic Persistent Fault Analysis (APFA) combines algebraic analysis with persistent fault analysis, providing a novel approach for examining block cipher implementation security. Since its introduction, APFA has attracted considerable attention. Traditionally, APFA has assumed that fault injection occurs solely within the S-box during the encryption process. Yet, algorithms like PRESENT and AES also utilize S-boxes in the key scheduling phase, sharing the same S-box implementation as encryption. This presents a previously unaddressed challenge for APFA. In this work, we extend APFA's fault injection and analysis capabilities to encompass the key scheduling stage, validating our approach on PRESENT. Our experimental findings indicate that APFA continues to be a viable approach. However, due to faults arising during the key scheduling process, the number of feasible candidate keys does not converge. To address this challenge, we expanded the depth of our fault analysis without increasing the number of faulty ciphertexts, effectively narrowing the key search space to near-uniqueness. By employing a compact S-box modeling approach, we were able to construct more concise algebraic equations with solving efficiency improvements ranging from tens to hundreds of times for PRESENT, SKINNY and CRAFT block ciphers. The efficiency gains became even more pronounced as the depth of the fault leakage increased, demonstrating the robustness and scalability of our approach.

**Keywords:** persistent fault; fault attack; S-box; algebraic representation; PRESENT; SKINNY; CRAFT

## 1. Introduction

Information security has become a focal point, and cryptographic techniques are increasingly prevalent. Under resource constraints, lightweight block ciphers are widely used due to their simple architecture, high efficiency, and ease of implementation. Common lightweight block ciphers include PRESENT [1], GIFT, SKINNY [2], LED [3], and CRAFT [4], etc..

Fault Analysis (FA) is a specialized form of side-channel analysis that can be considered an active analysis method targeting physical cryptographic embedded devices [5]. It involves executing fault injection on a device to force it into an abnormal state and analyzing the collected faulty ciphertexts to recover keys using knowledge of the cryptographic algorithm. Fault injection can be achieved by altering power voltage [6], external clock frequency [7], temperature [8], or exposing circuits to lasers [9] during key scheduling or encryption. There is also the permanent fault model, where attackers employ more aggressive means to damage cryptographic devices, resulting in persistent faults [9].

Fault injection and its accompanying analysis techniques are among the most effective methods for analyzing cryptographic devices. The most renowned analysis technique in this field was the Differential Fault Analysis (DFA) that utilize the differences between the correct and faulty ciphertexts from fixed inputs to recover keys [10]. In 2010, Courtois et al. proposed Algebraic Fault Analysis (AFA) [11], combining fault analysis with algebraic analysis. It translates differential fault information into algebraic equations and integrates them with encryption algorithms. In 2018, Zhang et al. introduced a novel fault analysis method called Persistent Fault Analysis (PFA) [12]. Their fault model lies between transient and permanent faults. Faults are injected into algorithm constants stored in memory (ROM), such as an element within an S-box. Unless ROM is refreshed, the fault persists through all subsequent encryptions, affecting rounds accessing the specific faulty S-box element, and disappears once the fault-injected device is reset. PFA primarily conducts statistical analysis on the value distribution of ciphertext bytes and can be used as a ciphertext-only cryptanalysis method.

In 2022, Zhang et al. introduced APFA [13], merging the persistent fault model with algebraic fault analysis. Faults are injected into S-boxes stored in memory, causing a deviation in one element. Since S-box data is bijective, data passing through the S-box during encryption deviates after fault injection, significantly enhancing the solving success rate and efficiency of algebraic analysis. Fang et al. decomposed 4-bit S-boxes into two 2-bit S-boxes, reducing intermediate variable usage and effectively enhancing solving efficiency [14].

The equation systems constructed in APFA are in the satisfiability problem (SAT) form that can be solved by Off-the-shelf SAT solvers, such as CryptoMiniSAT [15]. The SAT inputs consist of clauses in conjunctive normal form (CNF), where clauses are connected by conjunction $\wedge$. Each clause comprises literals connected by disjunction $\vee$, with each literal being either a positive or negative variable. Additionally, CryptoMiniSAT supports XOR syntax inputs, offering convenience in equation transformation. Previous APFA research describes S-boxes by Tseitin transformation to decompose higher-degree terms into linear combinations. Conjunctive normal form is then used to depict the Boolean relationships of all valid input-output pairs [16]. This approach necessitates numerous intermediate variables to construct data relationships before and after S-boxes [13,14]. Studies have shown that CNF clauses generated by the Logic Friday tool exhibit significant advantages in SAT solvers, reducing clause redundancy compared to traditional Tseitin transformation [17].

**Motivations.** Prior APFA research overlooked the implications of faulty S-boxes during the key scheduling process. Our investigation reveals that prominent cryptographic standards like AES [18] and PRESENT [1] incorporate S-boxes into their key scheduling algorithms, leveraging the same S-box implementation as the encryption stage [19–23]. This previously unaddressed challenge necessitates a comprehensive extension of APFA to encompass the key scheduling phase. In this paper, we present our investigation and findings on extending algebraic persistent fault analysis to encompass the key scheduling stage of cryptographic algorithms.

### *Our Contributions*

We extend fault injection and analysis to the key scheduling stage, discovering that algebraic persistent fault analysis remains applicable, but find that it comes with two challenges. Firstly, it introduces uncertainty. Deviating from the definition of bijection of S-box, round keys can no longer be traced back to a unique master key, making it difficult to reduce key search space to a single solution. Secondly, the constructed algebraic equations are larger and more complex, demanding higher solving efficiency. Increasing the depth of fault analysis can reduce uncertainty and consequently shrink the key search space, thus leveraging the first problem. However, increased depth results in larger algebraic equations, necessitating improved solving efficiency to address the second problem. Our main contributions are as follows:

- Using the 80-bit version of PRESENT as an example, we extend APFA fault injection and analysis to the S-box in the key scheduling stage with compact S-box modeling. Experiments demonstrate that algebraic persistent fault analysis remains applicable when delving into the key scheduling stage. The key search space is difficult to reduce to a single solution; increasing fault analysis depth can shrink the key search space. Experiments show that when the fault analysis encompasses the entire key scheduling and encryption processes, the key search space can approach uniqueness.
- Since the descriptions of S-boxes account for significant computational resources consumed for solving the SAT problem in APFA, we propose a truth table-based optimization method for S-box modeling. By representing S-boxes with optimized truth tables and utilizing the Logic Friday tool, S-boxes are converted into CNF representation, reducing clause count and eliminating

the need for auxiliary variables. Compared to previous construction methods, our algebraic construction significantly enhances solving efficiency in APFA, especially in high-complexity scenarios. For PRESENT, SKINNY, and CRAFT encryption algorithms, the required number of variables and clauses to establish complete algebraic equations are nearly halved, with solving efficiency improved by tens or even hundreds of times. Besides, we discovered that if solving time is constrained, improved efficiency also translates to higher solving success rates.

To facilitate further exploration by researchers and developers, we have hosted the relevant code in the following link: https://github.com/Hanbing0734/APFA-Implementation-Improvement.git.

## 2. Background

### 2.1. Notation Table

Table 1 defines the notations used in this paper.

**Table 1.** Notations used in this paper.

| Notations | Definitions | Notations | Definitions |
|---|---|---|---|
| $n$ | The block size | $S^{'}$ | The faulty S-box |
| $w$ | The size of S-box | $l$ | The fault location in the faulty S-box |
| $r$ | The round index | $f$ | The fault value in the faulty S-box |
| $R$ | The total number of rounds | $U$ | $S[l]$ |
| $X^r$ | The $r$-th round data block | $V$ | $S[l] \oplus f$, i.e., $S^{'}[l]$ |
| $X_i^r$ | The $i$-th element of $X^r$ | $N_c$ | The number of ciphertexts |
| $K^r$ | The $r$-th round key | $N_r$ | The depth of fault leakages |
| $C$ | The ciphertext | $N_v$ | The total number of variables |
| $F$ | The round function | $N_e$ | The total number of equations |
| $S$ | The S-box | $N_k$ | Size of the key search space |

### 2.2. Lightweight Block Ciphers in SPN Structure

We use PRESENT [1], SKINNY [2], and CRAFT [4] as our target block ciphers. The Substitution-Permutation Network (SPN) is an iterative core structure used in block ciphers, achieving encryption strength through alternating nonlinear substitution (Substitution) and linear permutation (Permutation) operations over multiple rounds. A standard SPN structure consists of $R$ iterative rounds, each comprising three core operations:

**AddRoundKey (AK)**: The round key is XORed with the intermediate state, introducing key dependency.

**Substitution Layer (SB)**: The parallel S-boxes perform nonlinear transformations on fixed-length data blocks. Typically 4-bit S-box is used in lightweight block ciphers.

**Permutation Layer (PL)**: The substituted bits are deterministically rearranged to achieve diffusion across S-boxes. Common strategies include: (1) ShiftRows: Cyclic shifts by rows (e.g., SKINNY). (2) Bit Permutation: Fixed position mapping (e.g., PRESENT's 64-bit permutation matrix). (3) MixColumns: Linear transformation based on a finite field (e.g., CRAFT's $4 \times 4$ MDS matrix can be implemented using XOR operations).

The typical algorithm structure is $F^r = PL \circ SB \circ AK(X^r, K^r), 1 \leq r \leq R$. Table 2 compares the core parameters of PRESENT, SKINNY, and CRAFT. All three use 4-bit S-boxes, with the specific input-output mappings shown in Table 3. CRAFT's tweak key ($TK$) is used alongside the master key ($MK$) in key scheduling.

**Table 2.** Comparison of algorithm parameters.

| Property | PRESENT | SKINNY-64 | CRAFT |
|---|---|---|---|
| Block length (bit) | 64 | 64 | 64 |
| Master key length (bit) | 80/128 | 64/128/192 | 128 |
| Number of rounds | 31 | 32/36/40 | 32 |
| S-box width | 4-bit | 4-bit | 4-bit |
| Key scheduling S-box | Reuse encryption S-box | - | - |

**Table 3.** S-box definitions.

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

The key scheduling algorithm for PRESENT-80, with a key size of 80 bits, is shown in Algorithm 1.

**Algorithm 1** Key schedule of PRESENT-80

```
Input: 80-bit master key $MK = k_{79}k_{78}\dots k_0$
Output: 64-bit round keys $K^1, K^2, \dots, K^{32}$
1: $K \leftarrow MK$
2: for $r = 1$ to 31 do
3:     $K^r \leftarrow K[79:16]$
4:     $K \leftarrow K \lll 61$                               ▷ Cyclic left shift by 61 bits
5:     $K[79:76] \leftarrow S(K[79:76])$                      ▷ Substitute the top 4 bits
6:     $K[19:15] \leftarrow K[19:15] \oplus \mathrm{bin}_5(r)$   ▷ XOR with 5-bit round counter
7: end for
8: $K^{32} \leftarrow K[79:16]$
```

## 2.3. *Algebraic Persistent Fault Analysis*

The construction of APFA [13] algebraic equations should launch multiple encryptions using the same key, as is illustrated in Figure 1. APFA is a ciphertext-only cryptanalysis method, operating as follows:

1. An attacker injects a small fault into the S-box of an encryption device, affecting the output of one of its mappings and rendering the S-box's mapping unbalanced. The faulty S-box ($S'$) persists through multiple encryption rounds. The fault information in $S'$ includes the fault location ($l$) and fault value ($f$), where $S'[l] = S[l] \oplus f$.
2. The victim encrypts multiple plaintexts using a fixed key on the faulty encryption device.
3. The attacker collects ciphertexts containing fault information and establishes a system of algebraic equations based on the relationship between the key, encryption process, and fault information of the S-box.
4. SAT solving tools are used to solve the equations. If successful, the solution reveals the master key.

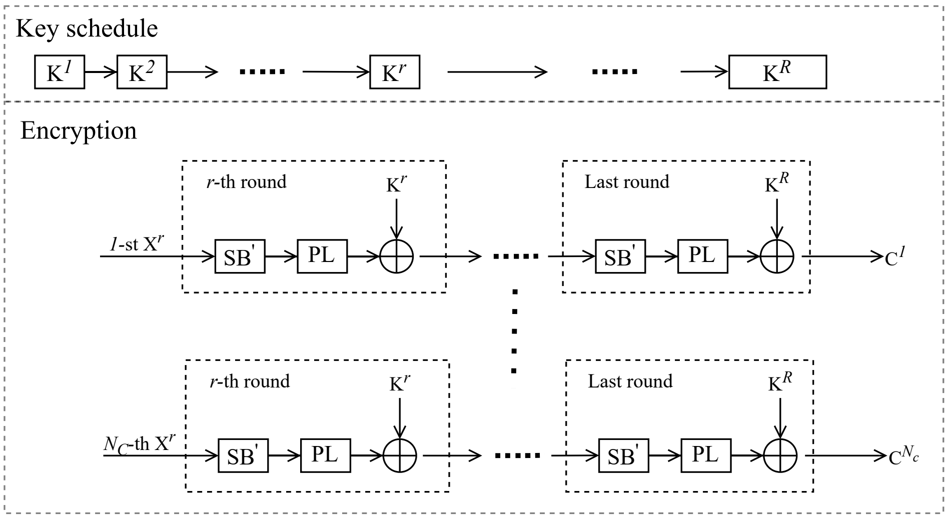

**Figure 1.** The relationship among the encryption process, key, S-box, and ciphertext.

For the faulty S-box $S^{'}$, assume the fault occurs at $S[l]$ (fault location $l$, i.e., the $l$-th byte of $S$), causing $S[l] = U$ to become $S^{'}[l] = V$. Consequently, for each round's $SB^{'}$, it holds that $S^{'}[X_i^r] \neq U$. This inequality is represented by CNF in Equation (1) and (2), where $d_{(j)}$ means the $j$-th bit of $d$. Since $d_{(j)}$s can not be 0 simultaneously, there must be at least one bit $d_i$ that is not zero, indicating that $S^{'}[X_i^r] \neq U$.

$$d_{(j)} \oplus S'[X_i^r]_{(j)} \oplus U_{(j)} = 0, \quad 0 \leq j < w \tag{1}$$

$$d_{(0)} \vee d_{(1)} \vee \cdots \vee d_{(w-1)} = 1 \tag{2}$$

## 3. Extending APFA to the Key Scheduling Phase

In this section, we explore the effects of extending the faulty S-box to the key scheduling phase. Our experiments and analyses confirm that APFA remains effective when fault injection and analysis are extended to the key scheduling process.

### *3.1. Implementation with Shared S-box*

When translating encryption algorithms from theory to practice, circuit implementations vary according to specific requirements. If throughput is prioritized, designs may consider executing the same operations in parallel, such as replicating S-boxes to allow parallel data processing through multiple S-boxes, thus enhancing efficiency. Conversely, to minimize the area of implementation circuits, reducing the number of implemented S-boxes is a viable approach. In circuit implementations of encryption devices, a single instance of an S-box may suffice, reducing circuit scale, with all related data passing through the S-box sequentially. Designs for AES [19,20] and PRESENT [21–23] reflect such considerations.

Practical circuits balance throughput and area based on requirements. For the PRESENT encryption algorithm's implementation, we assume a single S-box instance is used, with the key scheduling and encryption phases serially utilizing this S-box. A simplified representation of the data flow in its serial circuit construction is shown in Figure 2, where $r$ denotes the current round.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 April 2025 doi:10.20944/preprints202504.0526.v1

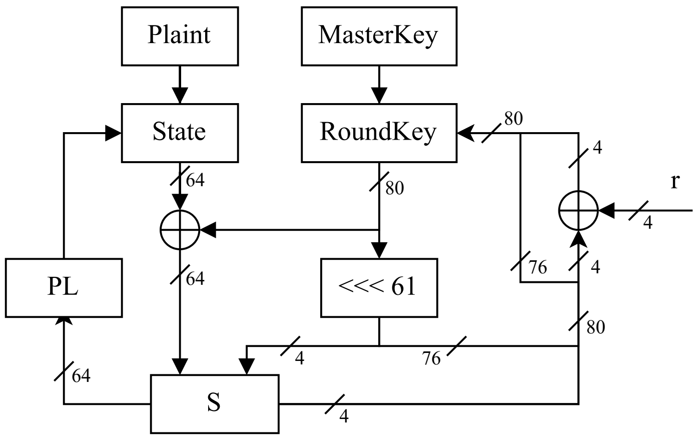

**Figure 2.** Data path of PRESENT encryption circuit (Unique S-box).

Focusing on the PRESENT encryption algorithm, where the key scheduling process requires the same S-box as used in encryption, we extend APFA fault injection to the key scheduling phase, assuming only one S-box instance in the circuit. Fault injection into the S-box affects both the encryption and key scheduling SB operations.

The use of a faulty S-box in the key scheduling phase significantly increases the scale and complexity of the algebraic system. We propose a new algebraic construction method for the S-box that greatly improves solving efficiency. Due to the structure of this paper, the introduction and effects of this method are discussed in Sections 4.2 and 5, respectively.

Our primary focus is on the feasibility of extending APFA to the key scheduling phase. Given the superior performance of our algebraic construction method, this section discusses extending APFA to the key scheduling phase using our proposed S-box modeling method exclusively.

## 3.2. Experimental Setup

We assume the key scheduling and encryption processes use the same faulty S-box instance. For the PRESENT encryption algorithm, using the faulty S-box $S'[0] = 0xd$ (Note that originally $S[0] = 0xc$). Compared to the initial APFA, extending to the key scheduling phase requires reflecting the S-box fault information during key scheduling. The complete process is detailed in Algorithm 2.

**Algorithm 2** APFA on 80-Bit PRESENT (Key schedule using faulty S-box)

```
Input: $\mathbb{C}, l, f, N_r$
Output: $MK$                                          ▷ Output: Recovered master key
 1: $U \leftarrow S[l]$                               ▷ Store original S-box value
 2: $S'[l] \leftarrow S[l] \oplus f$                  ▷ Inject fault at S-box position $l$
 3: for $r \leftarrow 0; r < R; r{+}{+}$ do
 4:     $genKeyScheduleConstraints(r, K^r)$           ▷ Normal key scheduling constraints
 5:     $genKeyScheduleFaultConstraints(r, K^r, U)$   ▷ Faulty S-box constraints
 6: end for
 7: for $C \in \mathbb{C}$ do
 8:     for $r \leftarrow R - N_r; r \leq R; r{+}{+}$ do   ▷ Analyze the last $N_r$ encryption rounds
 9:         $genSBoxConstraints(X^r)$                 ▷ S-box substitution constraints
10:         $genFaultSBoxConstraints(X^r, U)$         ▷ Faulty S-box constraints
11:         $genPermutationLayerConstraints(X^r)$     ▷ Permutation layer constraints
12:         $genAddRoundKeyConstraints(X^r)$
13:     end for
14:     $X^R \leftarrow C$                            ▷ Bind ciphertext to final state
15: end for
16: $MK \leftarrow RunAPFASolver()$                   ▷ Invoke solver to recover key
```

Here, "genKeyScheduleFaultConstraints" and "genFaultSBoxConstraints" represent constraints related to the faulty S-box during key scheduling and encryption, respectively, i.e., $S'[X] \neq U$.

Extending APFA to the key scheduling phase, the algebraic equation system includes the following constraints:

- Multiple encryptions use the same key.
- Constraints of the key scheduling process (using the faulty S-box). Since multiple encryptions use the same key, this constraint needs to be added only once.
- Constraints of the encryption process (using the faulty S-box).
- Additional constraints on the faulty S-box (the mapped value of the original S-box at the fault location $S[l]$ cannot appear in the data output through the faulty S-box, i.e., $S'[l] \neq S[l]$). This additional constraint must be added whenever data passes through a faulty S-box.

## *3.3. Results with Faulty S-box in Key Schedule*

In our experiments, we simulated fault injection using software and employed CryptoMiniSAT v5.11.22 as the SAT solver to resolve algebraic equations. For fairness, we configured the solver to execute in a single-threaded mode. The experiments were conducted on a PC equipped with 32 GB of memory and a 3.5 GHz Intel(R) Core(TM) i5-13600KF CPU, utilizing an Ubuntu 24.04.4 virtual machine. The virtual machine was allocated 16 GB of memory and 8 processors.

We measured the size of the master key search space after solving. As the depth of fault analysis increased from 5 to all encryption rounds, the average number of master key search space solutions obtained is shown in Figure 3. As the fault depth increases, the size of the master key search space decreases. When fault analysis covers the entire encryption and key scheduling process, the key search space approaches uniqueness.

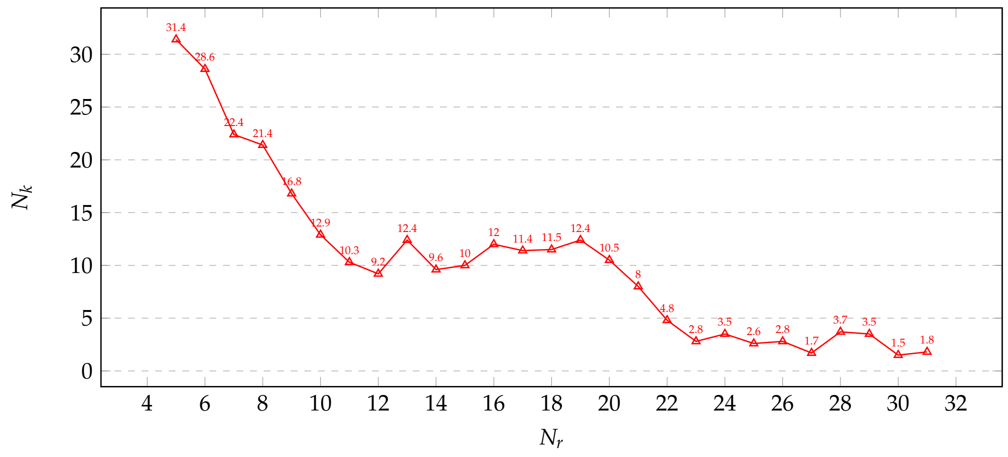

**Figure 3.** Reduction in master key search space with increasing fault analysis depth.

Since APFA is a ciphertext-only cryptanalysis method, it focuses solely on the last few rounds (equivalent to the fault analysis depth) of the encryption process. When constructing the system of algebraic equations, we include all key scheduling processes and the constraint equations corresponding to the last few rounds of encryption. Assuming a fault analysis depth of $N_r$, if no faults exist in the key scheduling process, the round key for the $(R - N_r + 1)$-th round, $K^{R-N_r+1}$, can be deduced uniquely and the unique master key ($K^1$) can be recovered accordingly.

However, if faults are injected into and the analysis is extended to the key scheduling process, uncertainty is also introduced into this process. Consequently, we are unable to make the search space for $K^{R-N_r+1}$ converge, which further complicates reducing the search space for the master key to a single solution.

Since the rules of key scheduling are deterministic, increasing the depth of fault analysis allows for a more in-depth examination, gradually eliminating the interference caused by faulty S-boxes, thereby reducing the search space of the master key. Ultimately, when $N_r = R$, meaning the fault analysis covers the entire key scheduling and encryption processes, the search space for the master key is reduced to approach uniqueness. However, it still cannot fully converge because, for the first S-box encountered in the key scheduling process, there may be multiple inputs corresponding to the desired S-box output, a problem that remains unresolved.

## 4. Algebraic Construction of the S-Box

The solving efficiency of APFA is influenced by the scale of the algebraic equation system. In constructing the algebraic system, addition operations can be directly represented using XOR, while permutation operations are realized through variable mapping. The core challenge lies in the algebraic representation of the S-box, as its complexity directly impacts the processing efficiency of the SAT solver [15]. This section introduces a novel method for S-box construction, with its innovation highlighted in Section 4.2.

### *4.1. Classical ANF-CNF Conversion Method*

The classical approach models the S-box by converting Algebraic Normal Form (ANF) to CNF. Taking the 4-bit S-box used in PRESENT as an example, assume $X_i = (x_0, x_1, \cdots, x_{w-1})$ and $Y_i = (y_0, y_1, \cdots, y_{w-1})$ are the S-box input and output, respectively. The ANF representation of the S-box is expressed in Equation (3). The AND operation is denoted by " $\cdot$ ", which may be omitted.

$$\begin{aligned} y_0 &= x_0 \oplus x_2 \oplus x_3 \oplus x_1x_2 \\ y_1 &= x_1 \oplus x_3 \oplus x_1x_3 \oplus x_2x_3 \oplus x_0x_1x_2 \oplus x_0x_1x_3 \oplus x_0x_2x_3 \\ y_2 &= 1 \oplus x_2 \oplus x_3 \oplus x_0x_1 \oplus x_0x_3 \oplus x_1x_3 \oplus x_0x_1x_3 \oplus x_0x_2x_3 \\ y_3 &= 1 \oplus x_0 \oplus x_1 \oplus x_3 \oplus x_1x_2 \oplus x_0x_1x_2 \oplus x_0x_1x_3 \oplus x_0x_2x_3 \end{aligned} \tag{3}$$

Each nonlinear term $x_0x_1 \ldots x_{w-1}$ requires the introduction of an intermediate variable $I_t$, establishing constraints as shown in Equation (4). Each term can be considered an independent CNF clause.

$$(x_0 \vee \bar{I}_t) \wedge (x_1 \vee \bar{I}_t) \wedge \ldots \wedge (x_{w-1} \vee \bar{I}_t) \wedge (\bar{x}_0 \vee \bar{x}_1 \vee \ldots \vee \bar{x}_{w-1} \vee I_t) = 1 \tag{4}$$

For an S-box with $n$ nonlinear terms, $O(n)$ intermediate variables and $O(n)$ CNF clauses are required. The original S-box in the PRESENT encryption algorithm requires 8 intermediate variables and 28 clauses for representation.

Assuming a 1-bit fault is injected into the S-box at fault location $l = 0$ with fault value $f = 0x0001$. Its algebraic representation requires 11 intermediate variables and 43 clauses.

While this method accurately models the S-box, the introduction of intermediate variables significantly increases the problem's dimensionality, leading to a substantial expansion of the SAT solving space.

### *4.2. Truth Table-Based Optimized S-Box Modeling Method*

We propose a direct modeling method based on truth table optimization. Let the S-box input be $X = (x_0, ..., x_{w-1})$ and output $Y = (y_0, ..., y_{w-1})$. Define a boolean function $P : \{0,1\}^w \times \{0,1\}^w \to \{0,1\}$, as shown in Equation (5).

$$P(X,Y) = \begin{cases} 1 & \text{if } Y = S(X) \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

For a given faulty S-box, convert it into a truth table of a Boolean function, where possible patterns have a value of 1 and impossible patterns have a value of 0. This truth table is input into the Logic Friday software. By invoking the Quine-McCluskey algorithm, a CNF description can be output [24]. The Espresso algorithm is used to simplify the sum-of-products representation [25].

For the PRESENT encryption algorithm, using the faulty S-box $S'[0] = 0xd$ (Note that originally $S[0] = 0xc$), the truth table input to the Logic Friday tool is shown in Table 4.

**Table 4.** Truth table of the boolean function constructed from a faulty S-box.

| $x_3$ | $x_2$ | $x_1$ | $x_0$ | $y_3$ | $y_2$ | $y_1$ | $y_0$ | $Prob$ |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |

After data transformation and reduction by the tool, the resulting CNF data is shown in Equation (6).

$$
\begin{aligned}
1 = {} & (\bar{x_3} \vee x_1 \vee y_1 \vee \bar{y_0}) \wedge (x_3 \vee x_1 \vee x_0 \vee y_0) \wedge (x_3 \vee x_2 \vee \bar{x_1} \vee y_1) \\
& \wedge (\bar{x_1} \vee x_0 \vee \bar{y_3} \vee y_1) \wedge (x_3 \vee x_1 \vee \bar{x_0} \vee y_3) \wedge (\bar{x_3} \vee \bar{x_2} \vee x_0 \vee \bar{y_1}) \\
& \wedge (\bar{x_2} \vee \bar{y_3} \vee \bar{y_1} \vee \bar{y_0}) \wedge (\bar{x_2} \vee \bar{y_2} \vee \bar{y_1} \vee y_0) \wedge (x_2 \vee y_2 \vee \bar{y_1} \vee y_0) \\
& \wedge (\bar{x_2} \vee \bar{x_1} \vee y_1 \vee y_0) \wedge (\bar{x_3} \vee x_2 \vee \bar{x_1} \vee y_3) \wedge (\bar{x_3} \vee \bar{x_1} \vee \bar{x_0} \vee y_2) \\
& \wedge (\bar{x_3} \vee \bar{x_1} \vee \bar{x_0} \vee \bar{y_0}) \wedge (x_3 \vee \bar{x_2} \vee y_3 \vee \bar{y_0}) \wedge (\bar{x_0} \vee y_3 \vee \bar{y_2} \vee y_0) \\
& \wedge (x_3 \vee y_3 \vee y_2 \vee \bar{y_1}) \wedge (\bar{x_3} \vee x_2 \vee x_1 \vee x_0 \vee \bar{y_2}) \wedge (x_2 \vee x_0 \vee \bar{y_3} \vee y_2) \\
& \wedge (x_0 \vee y_3 \vee \bar{y_2} \vee \bar{y_0}) \wedge (\bar{x_1} \vee \bar{y_3} \vee \bar{y_2} \vee y_0) \wedge (x_2 \vee \bar{x_0} \vee \bar{y_2} \vee \bar{y_1} \vee \bar{y_0}) \\
& \wedge (\bar{x_2} \vee \bar{x_0} \vee \bar{y_3} \vee y_2) \wedge (x_3 \vee \bar{y_2} \vee \bar{y_1} \vee \bar{y_0}) \wedge (\bar{y_3} \vee \bar{y_2} \vee y_1 \vee y_0) \\
& \wedge (x_0 \vee y_2 \vee y_1 \vee y_0) \wedge (x_2 \vee y_3 \vee y_2 \vee y_1) \wedge (\bar{x_2} \vee x_0 \vee \bar{y_3} \vee \bar{y_2}).
\end{aligned}
\tag{6}
$$

This algebraic construction method for the S-box does not require the introduction of additional auxiliary variables, and the results can be directly viewed as conforming to the format required by SAT solvers such as CryptoMiniSAT.

A comparison of the number of variables and clauses required for different S-box modeling methods is shown in Table 5. For the faulty S-box, we uniformly assume a 1-bit fault is injected at location $l = 0$ with fault value $f = 0x0001$.

## 5. APFA with Compact S-Box Modeling

We applied APFA to the PRESENT, SKINNY, and CRAFT encryption algorithms, utilizing our algebraic representation of the S-box and comparing it to the traditional method. For the faulty S-box used in the encryption process, we assumed a 1-bit fault was injected at fault location $l = 0$ with fault value $f = 0x0001$.

**Table 5.** Comparison of variable and clause requirements for different S-box construction methods.

| **Algorithm** | **S-box** | **Traditional method** | | **Proposed method** | |
|---|---|---|---|---|---|
| | | Intermediate variables | Clauses | Intermediate variables | Clauses |
| PRESENT | Original | 8 | 31 | - | 26 |
| | Faulty | 11 | 43 | - | 27 |
| SKINNY | Original | 8 | 30 | - | 22 |
| | Faulty | 8 | 33 | - | 23 |
| CRAFT | Original | 8 | 31 | - | 22 |
| | Faulty | 11 | 43 | - | 22 |

## 5.1. APFA on PRESENT

We assumed that the S-boxes used in the key scheduling and encryption phases of PRESENT reside in different memory units, with only the S-box in the encryption phase being faulty. The S-box used in the encryption phase has a single instance, through which all data is processed sequentially.

We implemented APFA on both the 80-bit and 128-bit versions of the PRESENT encryption algorithm. For the S-box, we employed two different algebraic construction methods.

Using our S-box algebraic construction method, the number of variables and clauses required to construct the complete algebraic expression is significantly reduced under different fault analysis depths and numbers of faulty ciphertexts. For the 80-bit version of PRESENT, we set parameters $N_r \in [5, 12]$, $N_c = 30$. The comparison with the original APFA is shown in Table 6.

**Table 6.** APFA on PRESENT-80 with compact S-box modeling.

| $N_r$ | $N_c$ | $N_v$ | | | $N_e$ | | | $time(s)$ | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | new | origin | ratio | new | origin | ratio | new | origin | ratio |
| 5 | 30 | 27164 | 48532 | 56.0% | 73846 | 104721 | 70.6% | 2.55 | 55.37 | 4.6% |
| 6 | 30 | 32924 | 59572 | 55.3% | 91126 | 129681 | 70.4% | 3.64 | 105.2 | 3.5% |
| 7 | 30 | 38684 | 70612 | 54.7% | 108406 | 154641 | 70.1% | 3.78 | 127.15 | 2.9% |
| 8 | 30 | 44444 | 81652 | 54.4% | 125686 | 179601 | 70.0% | 3.90 | 151.58 | 2.6% |
| 9 | 30 | 50204 | 92692 | 54.2% | 142966 | 204561 | 70.0% | 4.39 | 272.03 | 1.6% |
| 10 | 30 | 55964 | 103732 | 53.9% | 160246 | 229521 | 69.8% | 6.73 | 262.09 | 2.6% |
| 11 | 30 | 61724 | 114772 | 53.8% | 177526 | 254481 | 69.8% | 7.85 | 387.34 | 2.0% |
| 12 | 30 | 67484 | 125812 | 53.6% | 194806 | 279441 | 69.7% | 9.10 | 423.07 | 2.2% |

For the 128-bit version, we set parameters $N_r \in [5, 12]$, $N_c = 80$. The comparison with the original APFA is shown in Table 7.

**Table 7.** APFA on PRESENT-128 with compact S-box modeling.

| $N_r$ | $N_c$ | $N_v$ | | | $N_e$ | | | $time(s)$ | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | new | origin | ratio | new | origin | ratio | new | origin | ratio |
| 5 | 80 | 72136 | 128952 | 56.0% | 196252 | 278482 | 70.5% | 9.81 | 62.06 | 15.8% |
| 6 | 80 | 87496 | 158392 | 55.2% | 242332 | 345042 | 70.2% | 13.31 | 120.12 | 11.1% |
| 7 | 80 | 102856 | 187832 | 54.8% | 288412 | 411602 | 70.1% | 14.26 | 228.93 | 6.2% |
| 8 | 80 | 118216 | 217272 | 54.4% | 334492 | 478162 | 69.9% | 19.22 | 390.37 | 4.9% |
| 9 | 80 | 133576 | 246712 | 54.1% | 380572 | 544722 | 69.9% | 27.28 | 671.98 | 4.1% |
| 10 | 80 | 148936 | 276152 | 53.9% | 426652 | 611282 | 69.8% | 29.34 | 691.83 | 4.2% |
| 11 | 80 | 164296 | 305592 | 53.8% | 472732 | 677842 | 69.7% | 26.25 | 1051.61 | 2.5% |
| 12 | 80 | 179656 | 335032 | 53.6% | 518812 | 744402 | 69.7% | 26.41 | 1682.77 | 1.6% |

We consider these parameters representative. In experiments, both S-box construction methods with these parameters can solve successfully in a relatively short time, facilitating comparison.

Our algebraic representation significantly reduces the solving time for APFA, and as $N_r$ increases, the advantage becomes more pronounced due to the expanding scale of the algebraic system.

### *5.2. APFA on SKINNY-64*

We conducted APFA on three versions of the SKINNY-64 encryption algorithm. When the key length is 64 bits, both the original and our S-box construction methods successfully solve in a short time. For key lengths of 128 and 192 bits, due to our computational resource limitations, the original S-box construction method's solving efficiency becomes highly unstable, often failing to solve within 24 hours. We only perform comparisons on SKINNY-64-64.

Using our S-box algebraic construction method, for a key length of 64 bits, we set parameters $N_r \in [5, 12]$, $N_c = 20$. The comparison with the original APFA is shown in Table 8.

**Table 8.** APFA on SKINNY-64-64 with compact S-box modeling.

| $N_r$ | $N_c$ | $N_v$ | | | $N_e$ | | | $time(s)$ | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | new | origin | ratio | new | origin | ratio | new | origin | ratio |
| 5 | 20 | 23608 | 36408 | 64.8% | 55544 | 71544 | 77.7% | 1.70 | 27.83 | 6.1% |
| 6 | 20 | 27828 | 43188 | 64.4% | 66164 | 85364 | 77.5% | 3.87 | 47.86 | 8.1% |
| 7 | 20 | 32028 | 49948 | 64.1% | 76764 | 99164 | 77.4% | 6.03 | 101.35 | 5.9% |
| 8 | 20 | 36248 | 56728 | 63.9% | 87384 | 112984 | 77.3% | 7.69 | 186.41 | 4.1% |
| 9 | 20 | 40468 | 63508 | 63.7% | 98004 | 126804 | 77.3% | 8.03 | 293.41 | 2.7% |
| 10 | 20 | 44688 | 70288 | 63.6% | 108624 | 140624 | 77.3% | 10.76 | 623.49 | 1.7% |
| 11 | 20 | 48928 | 77088 | 63.5% | 119264 | 154464 | 77.2% | 11.55 | 1054.14 | 1.1% |
| 12 | 20 | 53168 | 83888 | 63.4% | 129904 | 168304 | 77.2% | 11.78 | 1277.21 | 0.9% |

Using our S-box algebraic construction method, we set the parameters as follows: for a key length of 128 bits, $N_r \in [7, 12]$ and $N_c = 120$; for a key length of 192 bits, $N_r \in [8, 12]$ and $N_c = 300$. The algebraic construction and the average solving time are detailed in Table 9. For a key length of 192 bits, when $N_r$ is set to 7, the key can always converge to a single-digit value, but it does not consistently converge to a unique value.

**Table 9.** APFA on SKINNY-64-128/192 with compact S-box modeling.

| Key lengths | $N_r$ | $N_c$ | $N_v$ | $N_e$ | $time(s)$ |
|---|---|---|---|---|---|
| 128 | 7 | 120 | 187320 | 455992 | 22.67 |
| | 8 | 120 | 212880 | 519952 | 30.52 |
| | 9 | 120 | 238320 | 583792 | 41.39 |
| | 10 | 120 | 263640 | 647512 | 67.19 |
| | 11 | 120 | 288840 | 711112 | 79.31 |
| | 12 | 120 | 314160 | 774832 | 111.38 |
| 192 | 8 | 300 | 530696 | 1298504 | 186.55 |
| | 9 | 300 | 594296 | 1458104 | 453.96 |
| | 10 | 300 | 657896 | 1617704 | 678.18 |
| | 11 | 300 | 721796 | 1777604 | 1015.61 |
| | 12 | 300 | 785696 | 1937504 | 1451.63 |

### *5.3. APFA on CRAFT*

CRAFT uses a 128-bit key and a 64-bit tweak to generate four 64-bit round keys, which are cycled through during encryption. The key information is obfuscated by the tweak key, thus multiple possible keys exist, with the correct key among them. Alternatively, we can recover the total 256-bit round key, which is determinable, but multiple combinations of master key and tweak key can produce this round key.

We do not consider finding a unique solution as the termination condition. In the experiment, we increase $N_c$ or $N_r$ as much as possible, ultimately reducing the key search space to 16, marking this as a successful solution.

For the CRAFT encryption algorithm, we set the parameters $N_r \in [5, 12]$ and $N_c = 80$. The comparison with the original APFA is shown in Table 10. When $N_r$ reaches 11, using the original construction of the S-box, we are unable to find a solution within 5 days.

**Table 10.** APFA on CRAFT with compact S-box modeling.

| $N_r$ | $N_c$ | $N_v$ | | | $N_e$ | | | $time(s)$ | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | new | origin | ratio | new | origin | ratio | new | origin | ratio |
| 5 | 80 | 86368 | 142688 | 60.6% | 183456 | 290976 | 63.0% | 29.45 | 557.76 | 5.3% |
| 6 | 80 | 104768 | 175168 | 59.8% | 226176 | 360576 | 62.7% | 52.68 | 1339.34 | 3.9% |
| 7 | 80 | 123088 | 207568 | 59.3% | 268816 | 430096 | 62.5% | 84.42 | 2349.01 | 3.6% |
| 8 | 80 | 141328 | 239888 | 59.0% | 311376 | 499536 | 62.3% | 189.31 | 10393.15 | 1.8% |
| 9 | 80 | 159488 | 272128 | 58.6% | 353856 | 568896 | 62.2% | 291.77 | 81903.38 | 0.4% |
| 10 | 80 | 177728 | 304448 | 58.4% | 396416 | 638336 | 62.1% | 132.69 | 124123.62 | 0.1% |
| 11 | 80 | 195888 | 336688 | 58.2% | 438896 | 707696 | 62.0% | 481.63 | / | / |
| 12 | 80 | 214128 | 369008 | 58.0% | 481456 | 777136 | 62.0% | 1026.81 | / | / |

## 5.4. Exploration with Limited Solving Time

In practice, due to limitations in computing resources, we often cannot invest unlimited time in automated analysis. Typically, a maximum time is set, and if unsolved within this timeframe, it is considered a failure.

We set the SAT solver's solving time to 3600s and conducted APFA on both versions of PRESENT. We gradually increased the fault analysis depth and recorded the number of ciphertexts required for successful solving within the time limit. The results are shown in Figures 4 and 5.

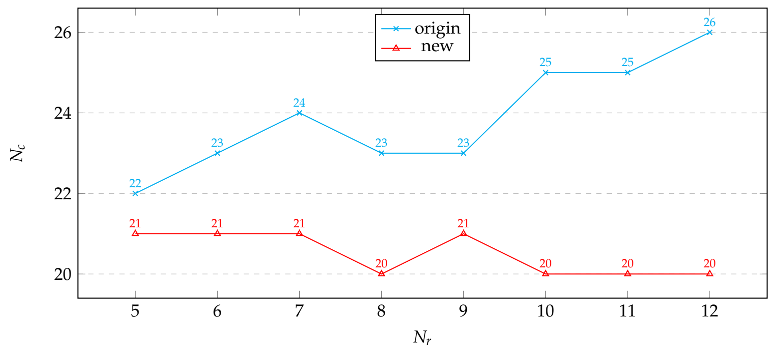

**Figure 4.** Number of ciphertexts required for solving PRESENT-80 (Max solution time = 3600s).

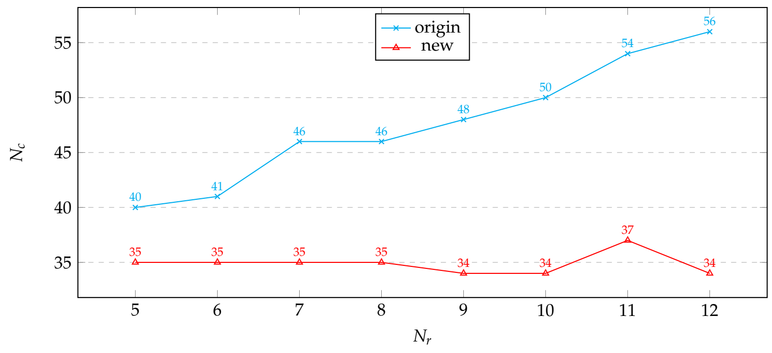

**Figure 5.** Number of ciphertexts required for solving PRESENT-128 (Max solution time = 3600s).

Under the time constraint, it can be observed that as the fault analysis depth increases, the number of ciphertexts required for solving using the original S-box construction method tends to rise. In contrast, using our model to construct the S-box, the number of ciphertexts required remains stable.

At the same fault analysis depth, based on the principle of Boolean function equivalence, the algebraic properties of different S-box construction methods are essentially the same, as both describe the S-box [26]. Theoretically, as the fault analysis depth increases, due to deeper analysis, the number of faulty ciphertexts required for successful solving using both S-box construction methods should decrease or remain stable, but should not rise.

With a constraint on solving time, aiming for a unique solution, an increase in faulty ciphertexts also means an increase in known conditions, which can improve solving efficiency and success rate. An increase in fault depth represents deeper exploration of faulty ciphertext information, which can also improve solving efficiency and success rate to some extent. However, both an increase in faulty ciphertexts and fault analysis depth lead to an increase in equation scale, requiring the solver to spend more time handling additional algebraic equations. These equations may not be necessary for successful solving, but the solver still needs to invest resources to verify them to ensure the results are not refuted by this information.

An increase in faulty ciphertexts has a linear impact on known conditions and equation scale. An increase in fault analysis depth has a linear impact on deep information exploration, but an exponential impact on equation scale. Each increase in fault analysis depth multiplies the complexity of the equation scale. Therefore, we believe there is a critical value for the impact of fault analysis depth on solving time. When the fault analysis depth is less than this value, an increase in depth means more efficient use of faulty ciphertexts, improving the success rate within the time limit, which is our main focus; when the depth exceeds this value, we focus more on the increase in solving time it brings.

Due to the relatively complex algebraic construction of the S-box in the original model, the problem scale becomes too large to solve within a limited time, necessitating an increase in ciphertext quantity to provide more "cost-effective" valid information. Using our S-box model, its algebraic construction is more conducive to the SAT solver extracting the key information contained within, and the critical value of fault analysis depth is much higher than that of the original S-box model, thus performing better in higher complexity scenarios.

## 6. Conclusions

In this paper, we validated the feasibility of extending APFA fault injection and analysis to the key scheduling phase. Injecting faults into the key scheduling process presents challenges for cryptanalysis, making it difficult to recover the correct key. In such cases, increasing the depth of fault analysis can reduce the search space for the master key. For the 80-bit version of the PRESENT encryption algorithm, using 30 faulty ciphertexts in our experiments, we observed that as the fault analysis depth increased, the key search space decreased. When fault analysis covered the entire encryption process, the key search space converged to near uniqueness.

We proposed an optimized algebraic representation for S-boxes in APFA, utilizing a truth table-based optimized S-box modeling method that eliminates the need for auxiliary variables and offers superior performance. We implemented this improvement across various SPN-based block ciphers, achieving significant enhancements in APFA solving efficiency, with improvements ranging from tens to hundreds of times, and better adaptability to high-complexity scenarios.

This work highlights the potential of our improved APFA method in enhancing the applicability and efficiency of fault-based cryptanalysis, particularly when fault injection extends to the key scheduling phase. Future research could further explore the optimization of this approach and its application to other cryptographic algorithms and fault models.

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

**Funding:** This research was funded by National Cryptologic Science Fund of China (No. 2025NCSF02011), National Natural Science Foundation of China (62102025), Beijing Natural Science Foundation (4222035), and Beijing Institute of Technology Research Fund Program for Young Scholars (XSQD-202024003).

**Data Availability Statement:** To facilitate further exploration by researchers and developers, we have hosted the relevant code in the following link: https://github.com/Hanbing0734/APFA-Implementation-Improvement.git.

**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.

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| APFA | Algebraic Persistent Fault Analysis |
| FA | Fault Analysis |
| DFAD | Differential Fault Analysis |
| AFA | Algebraic Fault Analysis |
| PFA | Persistent Fault Analysis |
| SAT | satisfiability problem |
| CNF | Conjunctive Normal Form |
| ANF | Algebraic Normal Form |

## 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 Proceedings of the Cryptographic Hardware and Embedded Systems-CHES 2007: 9th International Workshop, Vienna, Austria, September 10-13, 2007. Proceedings 9. Springer, 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 Proceedings of the Advances in Cryptology–CRYPTO 2016: 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II 36. Springer, 2016, pp. 123–153.
3. Guo, J.; Peyrin, T.; Poschmann, A.; Robshaw, M. The LED block cipher. In Proceedings of the Cryptographic Hardware and Embedded Systems–CHES 2011: 13th International Workshop, Nara, Japan, September 28–October 1, 2011. Proceedings 13. Springer, 2011, pp. 326–341.
4. Beierle, C.; Leander, G.; Moradi, A.; Rasoolzadeh, S. CRAFT: lightweight tweakable block cipher with efficient protection against DFA attacks. *IACR Transactions on Symmetric Cryptology* **2019**, *2019*.
5. Joye, M.; Tunstall, M.; et al. *Fault analysis in cryptography*; Vol. 147, Springer, 2012.
6. Barenghi, A.; Bertoni, G.M.; Breveglieri, L.; Pelosi, G. A fault induction technique based on voltage underfeeding with application to attacks against AES and RSA. *Journal of Systems and Software* **2013**, *86*, 1864–1878.
7. Aumüller, C.; Bier, P.; Fischer, W.; Hofreiter, P.; Seifert, J.P. Fault attacks on RSA with CRT: Concrete results and practical countermeasures. In Proceedings of the Cryptographic Hardware and Embedded Systems-CHES 2002: 4th International Workshop Redwood Shores, CA, USA, August 13–15, 2002 Revised Papers 4. Springer, 2003, pp. 260–275.
8. Hutter, M.; Schmidt, J.M. The temperature side channel and heating fault attacks. In Proceedings of the Smart Card Research and Advanced Applications: 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers 12. Springer, 2014, pp. 219–235.
9. Skorobogatov, S.P.; Anderson, R.J. Optical fault induction attacks. In Proceedings of the Cryptographic Hardware and Embedded Systems-CHES 2002: 4th International Workshop Redwood Shores, CA, USA, August 13–15, 2002 Revised Papers 4. Springer, 2003, pp. 2–12.

10. Biham, E.; Shamir, A. Differential fault analysis of secret key cryptosystems. In Proceedings of the Advances in Cryptology—CRYPTO'97: 17th Annual International Cryptology Conference Santa Barbara, California, USA August 17–21, 1997 Proceedings 17. Springer, 1997, pp. 513–525.
11. Courtois, N.T.; 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. Strategies Telecom and Multimedia, 2010.
12. Zhang, F.; Lou, X.; Zhao, X.; Bhasin, S.; He, W.; Ding, R.; Qureshi, S.; Ren, K. Persistent fault analysis on block ciphers. *IACR Transactions on Cryptographic Hardware and Embedded Systems* **2018**, pp. 150–172.
13. Zhang, F.; Feng, T.; Li, Z.; Ren, K.; Zhao, X. Free fault leakages for deep exploitation: algebraic persistent fault analysis on lightweight block ciphers. *IACR Transactions on Cryptographic Hardware and Embedded Systems* **2022**, pp. 289–311.
14. Fang, X.; Zhang, H.; Wang, D.; Yan, H.; Fan, F.; Shu, L. Algebraic persistent fault analysis of SKINNY_64 based on s_box decomposition. *Entropy* **2022**, *24*, 1508.
15. Soos, M.; Nohl, K.; Castelluccia, C. Extending SAT solvers to cryptographic problems. In Proceedings of the International Conference on Theory and Applications of Satisfiability Testing. Springer, 2009, pp. 244–257.
16. Knudsen, L.R.; Miolane, C.V. Counting equations in algebraic attacks on block ciphers. *International Journal of Information Security* **2010**, *9*, 127–135.
17. Hu, X.; Xu, S.; Tu, Y.; Feng, X.; Zhang, W. CNF characterization of sets over $\mathbb{Z}_{2^n}$ and its applications in cryptography. *Journal of systems science and complexity* **2024**, pp. 1–18.
18. Daemen, J.; Rijmen, V. *The design of Rijndael*; Vol. 2, Springer, 2002.
19. Banik, S.; Bogdanov, A.; Regazzoni, F. Atomic-AES: A compact implementation of the AES encryption/decryption core. In Proceedings of the Progress in Cryptology–INDOCRYPT 2016: 17th International Conference on Cryptology in India, Kolkata, India, December 11-14, 2016, Proceedings 17. Springer, 2016, pp. 173–190.
20. Moradi, A.; Poschmann, A.; Ling, S.; Paar, C.; Wang, H. Pushing the limits: A very compact and a threshold implementation of AES. In Proceedings of the Advances in Cryptology–EUROCRYPT 2011: 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings 30. Springer, 2011, pp. 69–88.
21. Hanley, N.; ONeill, M. Hardware comparison of the ISO/IEC 29192-2 block ciphers. In Proceedings of the 2012 IEEE computer society annual symposium on VLSI. IEEE, 2012, pp. 57–62.
22. Tay, J.; Wong, M.D.; Wong, M.; Zhang, C.; Hijazin, I. Compact FPGA implementation of PRESENT with boolean s-box. In Proceedings of the 2015 6th Asia Symposium on Quality Electronic Design (ASQED). IEEE, 2015, pp. 144–148.
23. Rolfes, C.; Poschmann, A.; Leander, G.; Paar, C. Ultra-lightweight implementations for smart devices–security for 1000 gate equivalents. In Proceedings of the Smart Card Research and Advanced Applications: 8th IFIP WG 8.8/11.2 International Conference, CARDIS 2008, London, UK, September 8-11, 2008. Proceedings 8. Springer, 2008, pp. 89–103.
24. Quine, W.V. The problem of simplifying truth functions. *The American mathematical monthly* **1952**, *59*, 521–531.
25. Brayton, R.K.; Hachtel, G.D.; McMullen, C.; Sangiovanni-Vincentelli, A. *Logic minimization algorithms for VLSI synthesis*; Vol. 2, Springer Science & Business Media, 1984.
26. Zelewski, S. *Komplexitätstheorie: als instrument zur klassifizierung und beurteilung von problemen des operations research*; Springer-Verlag, 2013.