Automatic Search of Meet-in-the-Middle Differential Fault Analysis on AES-like Ciphers

Anonymous Submission

Abstract. Fault analysis is a powerful technique to retrieve secret keys by exploiting side-channel information. Differential fault analysis (DFA) is one of the most powerful threats utilizing differential information between correct and faulty ciphertexts and can recover keys for symmetric-key cryptosystems efficiently. Since DFA usually targets the first or last few rounds of the block ciphers, some countermeasures against DFA only protect the first and last few rounds for efficiency. Therefore, to explore how many rounds DFA can affect is very important to make sure how many rounds to protect in practice. At CHES 2011, Derbez *et al.* proposed an improved DFA on `AES` based on MitM approach, which covers one more round than previous DFAs. To perform good (or optimal) MitM DFA on block ciphers, the good (or optimal) attack configurations should be identified, such as the location where the faults inject, the matching point with differential relationship, and the two independent computation paths where two independent subsets of the key are involved. In this paper, we formulate the essential ideas of the construction of the attack, and translate the problem of searching for the best MitM DFA into optimization problems under constraints in Mixed-Integer-Linear-Programming (MILP) models. With the models, we achieve more powerful and practical DFA attacks on `SKINNY`, `CRAFT`, `QARMA`, `PRINCE`, `PRINCEv2`, and `MIDORI` with faults injected in 1 to 9 earlier rounds than the best previous DFAs.

Keywords: Differential fault analysis · Meet-in-the-middle · Automatic search

1 Introduction

Fault analysis (FA), first introduced by Boneh *et al.* [BDL97] against RSA-CRT implementations, is one of the most powerful attacks on implementations of cryptographic primitives. It allows the attacker to get additional side-channel information and achieve the key recovery attacks in practical time. At CRYPTO 1997, Biham and Shamir [BS97] proposed the differential fault analysis (DFA) on DES block cipher. Since then, various fault attacks were introduced, such as (statistical) ineffective fault analysis [Cla07, DEK+18, DEG+18], collision fault analysis [BK06, Hem04], statistical fault attacks [FJLT13, DEK+16], fault intensity analysis [LSG+10], persistent fault attack [ZLZ+18], fault template attack [SBR+20], fault correlation analysis [SMC21], and other fault attacks [JSC+14, DPdC+15, BHJ+18, TLG+15, SHS16, HS13, GKPM18, MGV08, PAM19, RLK11]. For fault analysis on `AES`, the fault can be induced in the round counter [CT05, AK97] to reduce the number of rounds, or in the internal state during a round [PQ03, DLV03, BS03, Gir04] to exploit the confusion and diffusion characteristics of the fault, or in the key schedule [CY03, Gir04, TFY07, DV12]. There are several fault models in [RSG21]. A less common model is the random bit fault model [BS03], which allows the adversary to flip one particular bit. A more common model is the random byte fault model [PQ03].

Differential Fault Attack [BS97] exploits the difference between correct and faulty ciphertexts by injecting a fault in the internal state of the last several rounds. With the correct and faulty ciphertext pairs, the adversary can retrieve the secret key by a cryptographic analysis. The key point of DFA is that it allows the adversary to analyse a small number of rounds of a block cipher. DFA has been widely applied to the attacks on DES [Hem04, Riv09], AES [PQ03, DLV03, BS03, Gir04, MSS06, DFL11, YSW18, SHS16], PRESENT [SGSS14, GYS15, PBMB17] and others [FT09, TBM14]. The countermeasures against DFA include cipher or mode level (e.g. FRIET [SBD+20], CRAFT [BLMR19], DEFAULT [BBB+21], and others [MSGR10, MPR+11, DKM+15]) and implementation level ways [LRT12], etc. A widely used implementation level countermeasure against DFA is to perform the computation twice and check whether the same result is obtained [MSY06, ML08, JMR07, BBK+10].

Since DFA against block ciphers usually targets the last few rounds, one does not need to protect the whole cipher thus saving computation time [MSY06, CFGR10, BBK+10]. However, the number of rounds to protect must be chosen carefully in order to prevent security flaws. In order to determine how many rounds to protect, one has to know how many rounds DFA can work on. At CHES 2009, Rivain *et al.* [Riv09] studied against DES by introducing some faults at the end of round $R-7$, $R-6$, $R-5$ or $R-4$. For AES, most DFA techniques [PQ03, Gir04, Muk09] work by inducing faults at the end of round $R-3$, $R-2$ or $R-1$. Therefore, protecting the last and the first three rounds of AES against DFA is usually suggested [CFGR10]. At CHES 2011, Derbez, Fouque, and Leresteux [DFL11] introduced the Meet-in-the-Middle (MitM) and impossible differential fault analysis on AES by inducing faults at the end of round $R-4$ for the first time, that broke those implementations only protecting the last three rounds.

The Meet-in-the-Middle (MitM) approach is a time-memory trade-off cryptanalysis technique, which can be traced back to Diffie and Hellman's attack on DES [DH77]. The basic idea is to divide the key space into two independent subsets (also known as neutral sets), and then find matches from the two subsets. Let $E_K(\cdot)$ be a block cipher with block size n-bit, such that $C = E_K(P) = F_{K_2}(F_{K_1}(P))$, where $K = K_1 \| K_2$ has n bits, and K_1 and K_2 are neutral key materials of $n/2$ bits. For a given plaintext-ciphertext pair (P, C), a naive exhaust search attack needs a time complexity 2^n to find the key. However, the MitM attack computes independently $F_{K_1}(P)$ and $F^{-1}_{K_2}(C)$ with independent guesses of K_1 and K_2, and searches collision between $F_{K_1}(P)$ and $F^{-1}_{K_2}(C)$ to find the K with a time complexity about $2^{n/2}$. This approach has been widely applied in many cryptanalysis methods, like preimage attacks on hash functions [SA09] and key recovery attacks on block ciphers [BR10]. Derbez *et al.* [DFL11] for the first time combined the MitM approach with DFA by utilizing the pairwise matching in the differential internal state getting from correct and faulty ciphertexts with different subkey guessing. In the last decades, automatic search has boosted cryptanalysis, e.g. [BDF11, DF16]. Using automatic tools for solving problems like Mixed Integer Linear Programming (MILP), Satisfiability (SAT), Satisfiability Modulo Theories (SMT) problems, or Constrained Programming (CP), better cryptanalytic results have been achieved in topics including differential/linear attacks [MWGP11, SHW+14, KLT15], impossible differential attacks [ST17, CCF+21, SGL+17], cube or integral attacks based on division properties [XZBL16, TIHM17].

MILP is a method frequently used in business and economics to solve optimization problems. It deals with the problems of optimizing a linear objective function $f(x_1, x_2, \ldots, x_n)$ subject to linear inequalities involving variables x_i, $1 \le i \le n$. The first attempt to apply the MILP model to cryptanalysis is to determine the minimum number of differential active S-boxes for AES by Mouha *et al.* [MWGP11]. They assigned Boolean variable x_i for each S-box, where $x_i = 1$ means the i-th S-box is active. The variables are restricted by linear inequalities, which are derived from the differential propagation properties of each operation of AES. For example, given XOR operation $x_i \oplus x_j = x_k$, if $x_k = 1$ (active), there is at least one active S-box for the i-th and j-th S-box, i.e., $x_i + x_j \ge 1$. At EUROCRYPT 2021, Bao *et al.* [BDG+21] introduced the MILP-based automatic tools for

MitM preimage attacks on AES-like hashing, and many further improved tools for MitM have been proposed [DHS+21, BGST22, SS22]. At CRYPTO 2011, Bouillaguet, Derbez, and Fouque [BDF11] proposed an ad-hoc automatic tool to search DFAs on round-reduced AES. Their tool mainly used C++ programs to exhaust all possible DFA attacks on AES by exploiting its details and properties. Applying their tool to other block ciphers does not seem to be trivial, and many recent DFAs on other block ciphers are still done manually [CZS16, KAKS22, VSBM20]. In this paper, we manage to propose an MILP modelling method for the MitM DFAs, which is general for some popular block ciphers.

Our contributions. In this paper, we introduce an automatic search model based on MILP for the good or optimal MitM differential fault analysis, which is then successfully applied to several popular lightweight block ciphers. Under the fundamental assumptions of DFA, we can inject a random byte (nibble) fault to the internal state of the block cipher, and the location of the fault in the state could be controlled. We can obtain both correct and faulty ciphertexts. Under the assumptions, we generalize the MitM differential fault analysis, and translate the problems into MILP models by modelling the location and propagation of the differential faults, the matching point with differential relationship, the propagation of the two neutral sets, and the objective functions, etc.

Since many lightweight block ciphers adopt non-MDS layer (e.g. SKINNY [BJK+16]), we develop the matching rules for non-Maximum-Distance-Separable (non-MDS) operations, which include the modellings for the differential equations and the propagation of the two neutral sets. We develop the propagation rules of the two neutral sets, which will dominate the overall time complexity and should be balanced and minimized. When optimizing the computational complexity of the DFA attacks, the number of faults we need to inject is also one of the parameters to evaluate the strength of the attacks, which is one of the factors we considered in our objective function. To keep our DFA practical, we perform multiple MitM DFAs to recover the full key, while each MitM DFA only recovers a fraction of the key bits. Therefore, we develop the objective function to balance and minimize the time complexities of different MitM DFAs. Our tool is generic by writing down the linear inequalities on the matching part, the propagation of differential faults, the propagation of neutral sets, and the objective function. So, implementing the model for a different block cipher is just to replace the linear inequalities for each part.

As applications, we show better DFAs for SKINNY, CRAFT, QARMA, PRINCE, PRINCEv2, and MIDORI found by our tools in Sect. 4, where the positions of fault injections can be in 1 to 9 earlier rounds than the best previous attacks. We also employ our model in searching DFAs for AES, and only get improvements to secondary steps of AES-192 and AES-256. We show our attacks on AES in Sect. 4.6. We summarise our improved differential fault attacks in Table 1. The attacked "Round" means the number of non-linear layers between the position of the fault injection and the ciphertext. More rounds attacked means that there are more rounds need to be protected when implementing the cipher on devices. Columns with "Round (Previous)" and "Round (Ours)" list the best results of previous DFAs and our improved ones. The "Time" column and "Memory" column show the time complexities and memory complexities of our attacks, and the "# of faults" list the number of byte (nibble) faults need to inject in our attacks. Our model and source codes are publicly available at https://anonymous.4open.science/r/MITM-DFA-FDOC/.

Comparisons with previous MILP-based MitM attacks. In mathematic cryptanalysis, MILP-based MitM automatic models [BDG+21, DHS+21, SS22] have been proposed. In their models, they do not need to model the differential fault propagation. Therefore, their matching point is easier and only related to the propagations of two neutral sets, while our modelling for the matching point has to consider the propagations of both the neutral sets and the differential fault. Since our MitM DFA is for the practical attack, the sizes of two neutral sets are kept small, i.e., one MitM attack only recovers a small fraction of the full key. To recover the full key, we perform multiple MitM DFAs. Therefore, the overall time complexity is dominated by an MitM attack with the highest complexity. While previous models [BDG+21, DHS+21, SS22] only consider one MitM attack. Therefore, the objective function is different.

Table 1: Summary of DFA Results

Cipher	Round(Previous)	Round(Ours)	Time	Memory	# of faults
`SKINNY-128-128/256`	5 [VBSM18]	9	2^{24}	2^{24}	9
`SKINNY-64-64/128`	5 [VBSM18]	10	2^{24}	2^{24}	10
`CRAFT`	1 [RVB22]	10	2^{36}	2^{12}	28
`QARMA-64`	-	5	2^{16}	2^{16}	28
`QARMA-128`	-	4	2^{32}	2^{32}	10
`PRINCE/PRINCEv2`	4 [1][KAKS22]	4	2^{20}	2^{20}	10
`MIDORI-64`	3 [CZS16]	5	2^{16}	2^{16}	15
`MIDORI-128`	3 [CZS16]	5	2^{24}	2^{24}	9

[1] The ability in [KAKS22] is stronger, where they need to inject one-bit faults and know which bit is injected.

2 Preliminaries

2.1 Differential Fault Analysis using MitM

At CHES 2011, Derbez *et al.* [DFL11] proposed a differential fault analysis on `AES` using the MitM method shown in Figure 1. Note that $R = 10$ for `AES-128`. They induced a random byte fault injection at the end of round $R - 4$. For `AES-128`, according to the differential propagation rules, they can obtain several differential equations of the state at the beginning of Round 9 (i.e., X_9) between the correct and faulty states, i.e.,

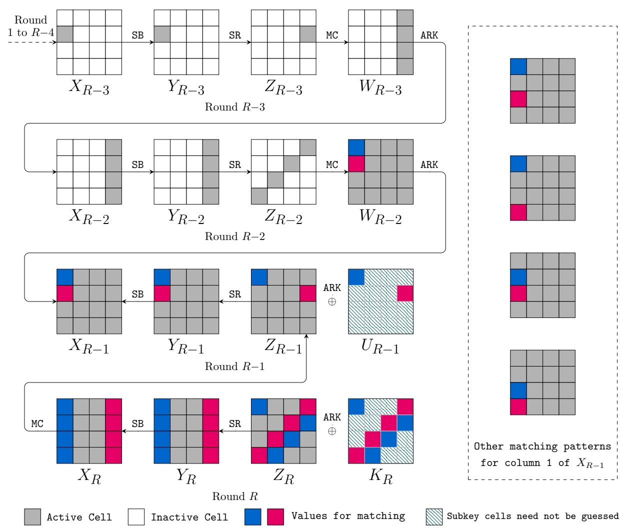

Figure 1: MitM differential fault analysis on `AES-128`.

$$X_9[0] \oplus \widetilde{X}_9[0] = I,\ X_9[1] \oplus \widetilde{X}_9[1] = I,\ X_9[2] \oplus \widetilde{X}_9[2] = 3I,\ X_9[3] \oplus \widetilde{X}_9[3] = 2I, \quad (1)$$

where $X_9[i]$ denotes the i-th byte in the correct state X_9, $\widetilde{X}_9[i]$ denotes the i-th byte in the faulty state $\widetilde{X}_9$, and $I \in \mathbb{F}_2^8$. These differential equations can provide a filter of subkey bits by constructing relationships, e.g.,

$$X_9[0] \oplus \widetilde{X}_9[0] = X_9[1] \oplus \widetilde{X}_9[1]. \tag{2}$$

Use C to denote the ciphertext state, the state X_9 can be represented as

$$X_9 = \texttt{SB}^{-1}(SR^{-1}(MC^{-1}(ARK^{-1}(SB^{-1}(SR^{-1}(ARK^{-1}(C))))))). \tag{3}$$

Denote K_0 as the whitening key, K_i $(i = 1, 2, ..., 10)$ as the subkey of Round i and $U_i = \texttt{MC}^{-1}(K_i)$. When computing $X_9[0]$ with Equ. (3), 5 key bytes, i.e., $K_{10}[0, 7, 10, 13]$ and $U_9[0]$, need to be guessed. Similarly, to compute $X_9[1]$, another 5 key bytes also need to be guessed, i.e., $K_{10}[3, 6, 9, 12]$ and $U_9[13]$. Since $X_9[0]$ and $X_9[1]$ are computed independently with two independent key subsets, we denote the two key subsets as *neutral key sets*. For a given pair of correct and faulty ciphertexts, the relationship in Equ. (2) has to be satisfied, which acts as a filter of 2^{-8} for the subkey space of 10 bytes involved in Equ. (2). In order to identify the one right 10-byte subkey, one has to collect 10 pairs of correct and faulty ciphertexts, which will provide a filter of $2^{-8\times 10} = 2^{-80}$. With the MitM approach, one guesses 5 subkey bytes ($K_{10}[0, 7, 10, 13]$, $U_9[0]$) to compute the 10-byte of $X_9[0] \oplus \widetilde{X}_9[0]$ of the 10 pairs for matching and store the guessed subkeys in a hash table indexed by the 10 matching bytes. Then, for each guess of $K_{10}[3, 6, 9, 12]$ and $U_9[13]$, one computes $X_9[1] \oplus \widetilde{X}_9[1]$ for the 10 pairs and checks against the hash table to find a match. One expects 1 match, which should be the correct 10-byte subkey. The time and memory costs are both 2^{40}.

The key point of the MitM DFA is to exploit differential equations, e.g. Equ. (2), whose expressions on both sides depend on different subsets of keys. At CRYPTO 2011, Bouillaguet, Derbez, and Fouque introduced an ad-hoc and dedicated automatic tool to search MitM DFAs on `AES` [BDF11]. In this paper, we try to build generic automatic models based on MILP for MitM DFAs to realize the optimal fault injection and key recovery phase.

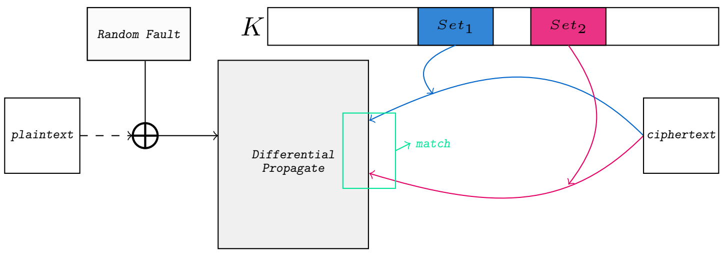

Figure 2: A high-level overview of the DFA based on MitM.

Generalizations and Notations of MitM DFA. Use Round 1 to Round R to denote the iterated round function of block ciphers, and assume we can inject random cell (byte or nibble) faults at a specified internal state to obtain correct and faulty ciphertexts. Denote the round we inject faults as Round R_s, and denote the round R_m as the matching point if there exist differential equations that can act as a filter of subkeys, e.g., $\Delta S[i] = \Delta S[j]$, where $\Delta S[i]$ is only determined by key cells in the neutral set Set_1 and $\Delta S[j]$ is determined by neutral set Set_2. Thereafter, a MitM DFA is performed to recover the subkey. Figure 2 gives a high-level framework.

2.2 Description of SKINNY

We will take the MILP-based automatic model of MitM DFA attack on SKINNY as an example. Therefore, we briefly describe SKINNY here. SKINNY is a family of lightweight block ciphers designed by Beierle *et al.* [BJK+16] based on the TWEAKEY framework [JNP14]. The block size is $n = 64$ or 128. There are versions with tweakey size $t = n$-bit TK_1, $2n$-bit (TK_1,TK_2), or $3n$-bit (TK_1,TK_2,TK_3). The r-th ($1 \leq r \leq R$) round function shown in Figure 3 consists of five operations: SubCells (SC), AddConstants (AC), AddRoundTweakey (ART), ShiftRows (SR) and MixColumns (MC).

For AddRoundTweakey (ART), the first and second rows of subtweakey STK are xored to the internal state. Initially, $STK = TK_1$ for SKINNY-n-n, $STK = TK_1 \oplus TK_2$ when $t = 2n$, and $STK = TK_1 \oplus TK_2 \oplus TK_3$ for $t = 3n$ version. The STK for each round is obtained by tweakey update function, which consists of two parts: firstly, a permutation $P_T = [9, 15, 8, 13, 10, 14, 12, 11, 0, 1, 2, 3, 4, 5, 6, 7]$ is applied to the tweakey arrays. Then, use different LFSRs to update the the first and second rows of TK_2 and TK_3 if $t = 2n$ or $t = 3n$.

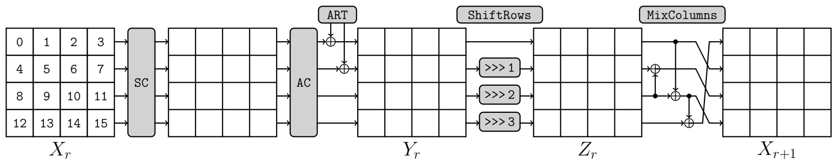

Figure 3: Round function of SKINNY.

3 Programming the MitM-DFA with MILP

3.1 Formulate the MitM Differential Fault Analysis

In this section, we show the specific modelling schemes for SKINNY. To facilitate the visualization of our analysis, each cell can take one of the five colors (White, Gray, Red, Blue, and Orange) according to certain rules, and a valid coloring scheme in our model corresponds to a configuration of the MitM differential fault analysis. The semantics of the colors are listed as follows:

- White(W): Inactive in forward differential propagation.
- Gray(G): Active in forward differential propagation.
- Blue(B): Known cells by guessing the key cells in Set_1.
- Red(R): Known cells by guessing the key cells in Set_2.
- Orange(O): Cells determined by $Set_1 \cap Set_2$.

For SKINNY with R rounds, we set the round of fault injection as R_s, and use R_m to denote the round of the matching point. For each internal state A in the rounds R_s to $R_m - 1$, we introduce the Binary variable a_i^A to denote if the i-th cell of state A is active, i.e., $a_i^A = 1$ means active cell with Gray color; $a_i^A = 0$ means inactive cell with White.

For the states in the rounds R_m to R, we introduce two additional Binary variables b_i^A and c_i^A, such that $(a_i^A, b_i^A, c_i^A) = (1, 1, 0)$ represents Blue, $(a_i^A, b_i^A, c_i^A) = (1, 0, 1)$ represents Red, $(a_i^A, b_i^A, c_i^A) = (1, 1, 1)$ represents Orange, $(a_i^A, b_i^A, c_i^A) = (1, 0, 0)$ represents Gray and $(a_i^A, b_i^A, c_i^A) = (0, 0, 0)$ represents White. For rounds from R_s to R, the propagation of Gray and White cells depends on the differential propagation. For rounds from R to R_m, the propagation of R, B and O rely on the backward computation rules.

3.2 Programming the MILP Model

In this section, we show how to build constraints for each component and how to solve the model for specific block ciphers. Use SKINNY as an example, we show the details below.

3.2.1 Constraints for the states from Round R_s to Round $R_m - 1$

According to the differential property of SKINNY, we model the constraints of active cells before and after SC, SR and MC operations. Use X_r, Y_r and Z_r to denote the state before SC, SR and MC operations in round r, respectively. Let STK_r denote the tweakey state used in round r. Since ARK and AC operations don't affect the propagation of the differential, we only list the the rules for SC, SR and MC operations below.

- SC: Because we only consider the cell-level truncated differential propagation, for each S-box, the output difference is nonzero if and only if the input difference is nonzero. The constraint for SC is $a_i^{Y_r} - a_i^{X_r} = 0$, $\forall\, 0 \le i \le 15$. We add the constraint $\sum_i a_i^{X_{R_s}} = 1$ to make sure that random fault is injected to one cell at the starting point.
- SR: For the SR operation, a cell permutation P is applied, i.e., $a_i^{Z_r} - a_{P[i]}^{Y_r} = 0$, $\forall\, 0 \le i \le 15$, where $P = [0, 1, 2, 3, 7, 4, 5, 6, 10, 11, 8, 9, 13, 14, 15, 12]$.
- MC: For each column j of MC, we build constraints according to the following rules, where $0 \le j \le 3$:

  $$a_j^{X_{r+1}} = a_j^{Z_r} \vee a_{8+j}^{Z_r} \vee a_{12+j}^{Z_r},\ a_{4+j}^{X_{r+1}} = a_j^{Z_r},\ a_{8+j}^{X_{r+1}} = a_{4+j}^{Z_r} \vee a_{8+j}^{Z_r},\ a_{12+j}^{X_{r+1}} = a_j^{Z_r} \vee a_{8+j}^{Z_r}.$$

  Following the method by [SHW+14], we convert the above equations to inequalities.

For the matching point $X_{R_m} = \mathtt{MC}(Z_{R_m-1})$, the possible matching rules are given as follows (also in Figure 4):

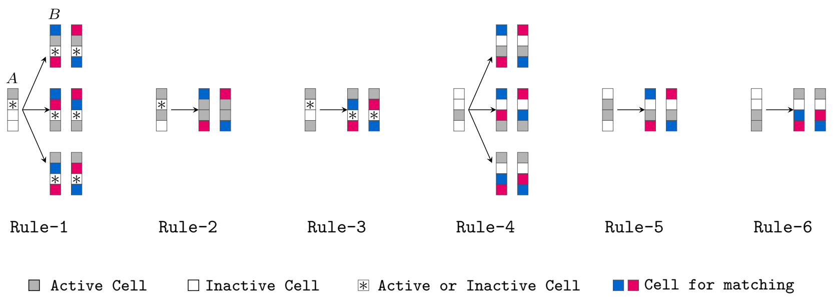

Figure 4: Rule for matching state of SKINNY.

- Rule-1: With input truncated differential form 1*00, the 0-th cell, the 1-th cell, and the 3-th cell of the output difference are possible for matching.

  For example, for $\mathtt{MC}(A) = B$ in Figure 4, since $\Delta A[2] = \Delta A[3] = 0$, $\Delta B[0] = \Delta B[1] = \Delta B[3]$ holds according to the property of MC of SKINNY. Therefore, if $\Delta B[0]$ can be computed by key subset marked by Blue and $\Delta B[3]$ can be computed by another key subset marked by Red, then a MitM attack can be performed with these two neutral key subsets and the relation $\Delta B[0] = \Delta B[3]$, i.e., a matching point exists. We restrict that only the three cells $B[0, 1, 3]$ can be marked by Blue or Red. In other words, if this column has both Blue and Red cells, there exists one matching equation.
- Rule-2: With input truncated differential form 1*10, the 0-th cell and the 3-th cell of the output difference are possible for matching.

- `Rule 3:` With input truncated differential form `1*01`, the 1-th cell and the 3-th cell of the output difference are possible for matching.
- `Rule-4:` With input truncated differential form `0010`, the 0-th cell, the 2-th cell and the 3-th cell of the output difference are possible for matching.
- `Rule-5:` With input truncated differential form `0110`, the 0-th cell and the 3-th cell of the output difference are possible for matching.
- `Rule-6:` With input truncated differential form `0011`, the 2-th cell and the 3-th cell of the output difference are possible for matching.

We introduce Binary variables $match_i (0 \leq i \leq 3)$ for each column. If there exists at least one `Blue` cell and one `Red` cell in the i-th column, then $match_i = 1$, otherwise $match_i = 0$ or the model is infeasible. This can be done with the constraints

$$match_i \geq b_{4j+i}^{X_m},\ match_i \geq c_{4j+i}^{X_m},\ match_i \leq \sum_j b_{4j+i}^{X_m},\ match_i \leq \sum_j c_{4j+i}^{X_m}, \quad \text{for } 0 \leq j \leq 3.$$

We call $\textsc{DoM} = \sum_i match_i$ the degree of match, and add constraint $\textsc{DoM} = 1$ to make sure that one differential equation is utilized to act as a filter in each injection phase.

3.2.2 Constraints for the states from Round R to Round R_m

In this section, we show the constraints for the known values by guessing two neutral key subsets in DFA on `SKINNY`. Use b_i^A and c_i^A to denote the value of the i-th cell in the state A, which is known by guessing subkey cells in Set_1 and Set_2 respectively. The propagation rules of b_i^A and c_i^A are determined by operations in the decryption direction. We list the rules for different operations for b_i^A, and the constraints for c_i^A are the same.

- `SC`: If the output of `SC` is known, the input is known, i.e., $b_i^{X_r} - b_i^{Y_r} = 0,\ \forall\ 0 \leq i \leq 15$.
- `ARK`: For `SKINNY`, the first and second rows of subtweakey array are XORed to the internal state. If a cell is XORed with the subtweakey state, we can deduce the value of the cell by guessing the corresponding cell in the subtweakey state, else we need not guess any subtweakey cell. Thus we have the following constraints for `ARK`:

$$b_i^{STK_r} - b_i^{Y_r} = 0\ (0 \leq i \leq 7),\ b_i^{STK_r} = 0\ (8 \leq i \leq 15).$$

- `SR`: For `SR` operation, a cell permutation P is applied on the internal state, *i.e.*, $b_{P[i]}^{Y_r} - b_i^{Z_r} = 0,\ \forall\ 0 \leq i \leq 15$.
- `MC`: For each column j, the known cells in the input column deduced from the output column follow the rules below, where $0 \leq j \leq 3$:

$$b_j^{Z_r} = b_{4+j}^{X_{r+1}},\ b_{4+j}^{Z_r} = b_{4+j}^{X_{r+1}} \wedge b_{8+j}^{X_{r+1}} \wedge b_{12+j}^{X_{r+1}},\ b_{8+j}^{Z_r} = b_{4+j}^{X_{r+1}} \wedge b_{12+j}^{X_{r+1}},\ b_{12+j}^{Z_r} = b_j^{X_{r+1}} \wedge b_{12+j}^{X_{r+1}}.$$

  Following the method by [SHW+14], we convert the equations to inequalities.

According to these principles, we build constraints for internal states from round R to R_m.

3.2.3 Objective Function and Solving Process

To properly evaluate the performance of our attacks, we take into account two factors: the number of faults required to inject and the time complexity of recovering the key cells for each injection. Since multiple fault injections may be necessary to fully recover the key, the solving phase is a multi-step process.

For SKINNY-n-n, the key schedule is linear, where a cell-wise permutation P_T is applied in each round to update the tweakey matrix. Denote the master tweakey state as tk, where tk_i represents the i-th cell of the state. To prevent repeatedly guessing one byte of the tweakey across multiple rounds, we introduce KB_i and KR_i $(0 \le i \le 15)$ to indicate whether each tweakey cell belongs to Set_1 or Set_2. We constrain the variables with

$$KB_i = \bigvee_{r=R_m}^{R} b^{STK_r}_{P_T^{(r-1)}[i]}\ , \text{ and } KR_i = \bigvee_{r=R_m}^{R} c^{STK_r}_{P_T^{(r-1)}[i]}.$$

We introduce the auxiliary variables KG_i $(0 \le i \le 15)$ to represent whether tk_i has been guessed or not, which are initialized as $KG_i = 0,\ \forall\ 0 \le i \le 15$. After each fault injection and key recovery phase, we update the state with $KG_i = 1$ if tk_i has been obtained in the key recovery phase, i.e., we update the model according to the following equation after each solving phase:

$$KG_i = KG_i \vee KB_i \vee KR_i,\ \forall\ 0 \le i \le 15.$$

Use $\overline{V}$ to denote the "NOT" operation for any variable V. In each solving phase, $n_1 = \sum_{i=0}^{15}(KB_i \wedge \overline{KG_i})$ and $n_2 = \sum_{i=0}^{15}(KR_i \wedge \overline{KG_i})$ denote the number of cells that we need to guess in each tweakey subset (Set_1 or Set_2), and $n_3 = \sum_{i=0}^{15}(\overline{KG_i} \wedge \overline{KB_i} \wedge \overline{KR_i})$ denotes the number of cells in tweakey state that we have not guessed. For each recovery phase, the time complexity to recover the full tweakey state is about $(2^{c \cdot n_1} + 2^{c \cdot n_2}) \times 2^{c \cdot n_3}$. So we set the objective function as

$$\text{Minimize}\left(\sum_{i=0}^{15}\left(\overline{KG_i} \wedge \overline{KB_i} \wedge \overline{KR_i}\right) + \text{Max}\left(\sum_{i=0}^{15}\left(KB_i \wedge \overline{KG_i}\right), \sum_{i=0}^{15}\left(KR_i \wedge \overline{KG_i}\right)\right)\right)$$

to search for the attack with the optimal time complexity. To ensure our attack can run in a practical time, We add constraints for KB_i and KR_i. For example, we add constraints

$$\text{Max}\ \{\sum_{i=0}^{15} KB_i, \sum_{i=0}^{15} KR_i\} \le 5$$

to make sure the time complexity of each filter phase is bounded by 2^{40}.

To perform the key recovery attack with fewer fault injections, we expect to exhaust the matching degree that can be got from each injection. Thus, we set the cell with a fault injection at the starting point to always have a difference, and repeatedly solve the model until there are no differential equations that can be used for matching. That is, if we get $a_i^{X_{R_s}} = 1$ after the first solving phase, we add constraint $a_i^{X_{R_s}} = 1$ to the model and run the optimizing phase again, until the model is infeasible. Then we remove the constraint $a_i^{X_{R_s}} = 1$. The results of each optimizing phase will be output.

4 Applications

In this section, we show some applications of our automatic search model. We applied our algorithm to SKINNY, CRAFT, QARMA, PRINCE , PRINCEv2, and MIDORI block ciphers. All the results are shown in Table 1.

4.1 Differential fault analysis on SKINNY-n-n and SKINNY-n-2n

4.1.1 DFA on SKINNY-n-n

By utilizing our automatic search model, we are able to develop better DFAs on SKINNY-n-n, with fault injections at the beginning of round $R-8$. Denote the i-th cell in the master tweakey state as tk_i. In this section, we show the attack for SKINNY-128-128 by steps as an example.

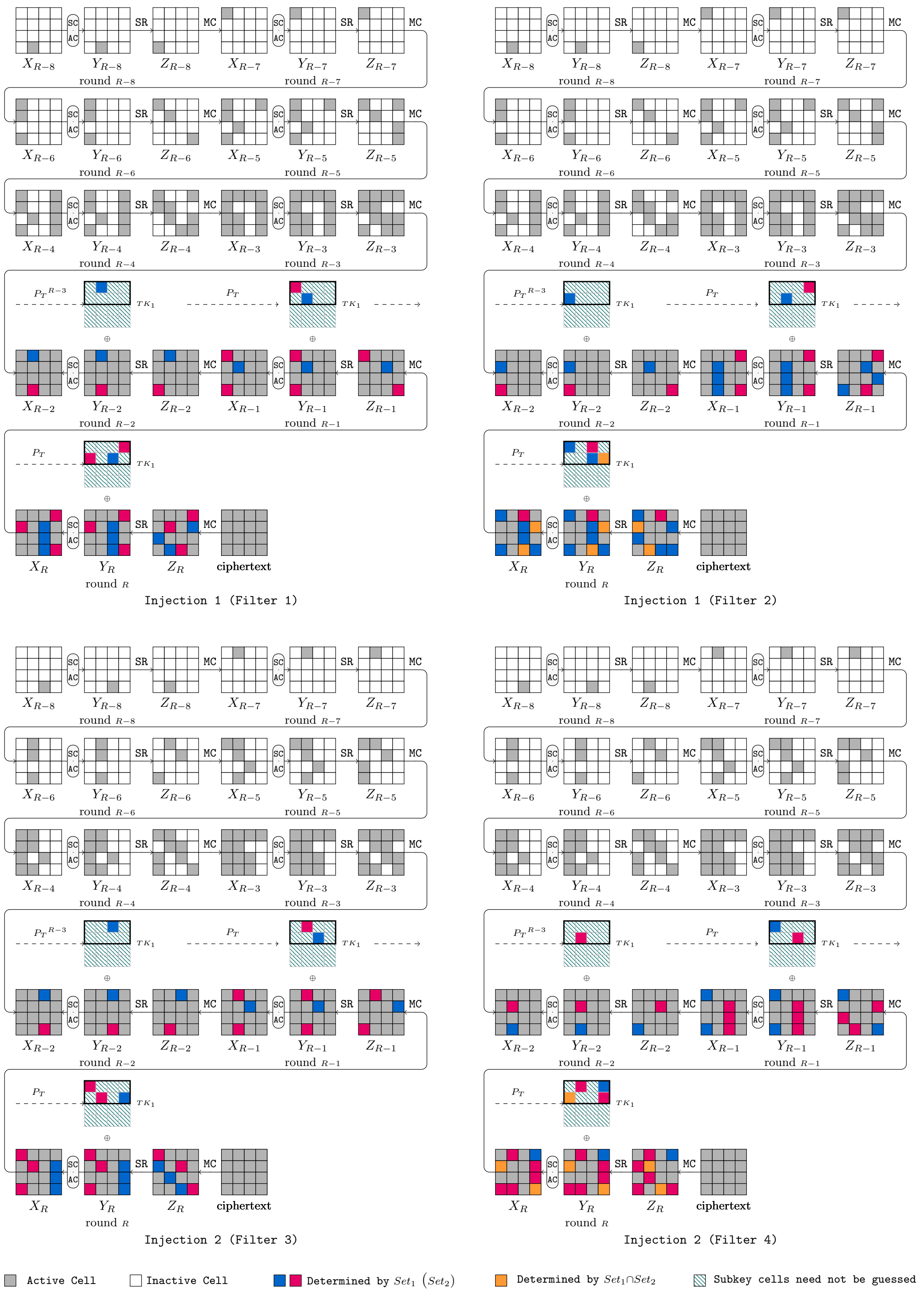
SC AC
SR
MC
X_{R-8} Y_{R-8} Z_{R-8} X_{R-7} Y_{R-7} Z_{R-7}
round $R-8$ round $R-7$
X_{R-6} Y_{R-6} Z_{R-6} X_{R-5} Y_{R-5} Z_{R-5}
round $R-6$ round $R-5$
X_{R-4} Y_{R-4} Z_{R-4} X_{R-3} Y_{R-3} Z_{R-3}
round $R-4$ round $R-3$
${P_T}^{R-3}$ P_T TK_1 $\oplus$
X_{R-2} Y_{R-2} Z_{R-2} X_{R-1} Y_{R-1} Z_{R-1}
round $R-2$ round $R-1$
X_R Y_R Z_R ciphertext
round R
Injection 1 (Filter 1)
Injection 1 (Filter 2)
Injection 2 (Filter 3)
Injection 2 (Filter 4)
Active Cell
Inactive Cell
Determined by Set_1 (Set_2)
Determined by $Set_1 \cap Set_2$
Subkey cells need not be guessed

Figure 5: DFA on SKINNY-128-128.

- **Injection 1:** As shown in Figure 5, firstly we inject random byte faults at $X_{R-8}[13]$, and query for the correct and faulty ciphertext pairs. According to the matching rules we build for `SKINNY`, we get two differential equations that correspond to filter phase `Filter 1` and `Filter 2`, *i.e.* `Filter 1` uses the differential equation $\Delta X_{R-2}[1] = \Delta X_{R-2}[13]$ to filter the wrong key bytes and `Filter 2` uses $\Delta X_{R-2}[4] = \Delta X_{R-2}[12]$ to do that.

  `Filter 1`: We consider the filter process of

$$\Delta X_{R-2}[1] = \Delta X_{R-2}[13]. \tag{4}$$

  Use $\widetilde{X}_{R-2}$ to denote the faulty state of X_{R-2} with fault injections. The differential state $\Delta X_{R-2}[1] = X_{R-2}[1] \oplus \widetilde{X}_{R-2}[1]$ can be calculated with the correct and faulty ciphertext pairs by guessing the tweakey cells $\{tk_2, tk_9, tk_{13}\}$, and the differential state $\Delta X_{R-2}[13]$ can be calculated by guessing $\{tk_3, tk_{10}, tk_{15}\}$. Equ. (4) provides a filter of 2^{-8}. Therefore, we require 6 pairs of correct and faulty ciphertexts with differences at $X_{R-8}[13]$ to recover the tweakey bytes $\{tk_2, tk_3, tk_9, tk_{10}, tk_{13}, tk_{15}\}$.

  Such that we choose 6 plaintexts, and inject random byte faults at $X_{R-8}[13]$ to get the corresponding ciphertexts, and guess the tweakey bytes to compute the differential state $\Delta X_{R-2}[1]$ and $\Delta X_{R-2}[13]$. After the filter of Equ. (4), there is about 1 possible value of $\{tk_2, tk_3, tk_9, tk_{10}, tk_{13}, tk_{15}\}$ remaining. We use hash tables to reduce the complexity of this process. Specifically, we calculate the differential state $\Delta X_{R-2}[1]$ for 6 pairs of correct and faulty ciphertexts, and store the corresponding tweakey bytes $\{tk_2, tk_9, tk_{13}\}$ indexed by the value $\Delta X^j_{R-2}[1] (1 \le j \le 6)$ of the 6 pairs. Then we calculate $\Delta X_{R-2}[13]$ for all 6 pairs of ciphertexts with all possible hypotheses of $\{tk_3, tk_{10}, tk_{15}\}$. If there is an intersection with the index, the 6-byte key is potentially correct. We expect to retrieve one correct value of the 6 tweakey bytes $\{tk_2, tk_3, tk_9, tk_{10}, tk_{13}, tk_{15}\}$ due to the 6 pairs match of $\Delta X_{R-2}[1] = \Delta X_{R-2}[13]$. The time complexity of this step is about 2^{24} and the memory cost is also about 2^{24}.

  `Filter 2`: We use the differential equation $\Delta X_{R-2}[4] = \Delta X_{R-2}[12]$ to filter the tweakey bytes, similar to the process in `Filter 1`. With already recovered $\{tk_2, tk_9, tk_{13}\}$ in the previous phase, we guess tk_{12} to compute the value of differential state $\Delta X_{R-2}[4]$, and guess $\{tk_4, tk_{11}, tk_{12}\}$ to compute the value of differential state $\Delta X_{R-2}[12]$. We build a hash table indexed by $\Delta X_{R-2}[4]$ for the 6 ciphertext pairs obtained in `Filter 1` to filter the key bytes, where about 1 candidate remaining. The time complexity of this step is about 2^{24} and the memory cost is about 2^8.

- **Injection 2:** In this stage, we inject random byte faults at $X_{R-8}[14]$ and filter the key bytes. This is a two-step process involving `Filter 3` and `Filter 4`.

  `Filter 3`: We treat the differential equation $\Delta X_{R-2}[2] = \Delta X_{R-2}[14]$ by guessing $\{tk_0, tk_{12}, tk_{15}\}$ to compute $\Delta X_{R-2}[2]$, and guessing $\{tk_5, tk_8, tk_{13}\}$ to compute $\Delta X_{R-2}[14]$. Note that we have retrieved $\{tk_{12}, tk_{13}, tk_{15}\}$ in **Injection 1**. Such that we can build hash tables to filter $\{tk_0, tk_5, tk_8\}$ with 3 pairs of correct and faulty ciphertexts with fault injections at $X_{R-8}[14]$. This step costs about 2^{16} time complexity and 2^8 memory.

  `Filter 4`: We compute $\Delta X_{R-2}[13]$ with known key bytes $\{tk_3, tk_{10}, tk_{15}\}$, and compute $\Delta X_{R-2}[5]$ using known key bytes $\{tk_0, tk_{10}, tk_{12}, tk_{15}\}$ and guessed tk_{14}. Then we can filter the value of tk_{14} by $\Delta X_{R-2}[13] = \Delta X_{R-2}[5]$ using the 3 ciphertext pairs in `Filter 3`. This costs 2^8 time complexity. So far, we have retrieved the value of $\{tk_0, tk_2, tk_3, tk_4, tk_5, tk_8, tk_9, tk_{10}, tk_{11}, tk_{12}, tk_{13}, tk_{14}, tk_{15}\}$.

- **Exhaustively Search:** For the remaining 3 bytes of tweakey that we have not guessed, we test all possible values and check the plaintext and ciphertext. This step costs a time complexity of about 2^{24}.

The overall time and memory complexities of our attack are about 2^{24}, and we need to inject 9 random bytes faults to recover the full key.

For SKINYY-64-64, the process is similar, which only alters the positions of tweakey cells guessed because of the difference in encryption round. And we can inject faults at the beginning of round $R-9$ to get a more powerful differential fault analysis.

Similar to the attack on SKINNY-128-128, we inject random faults at $X_{R-9}[13]$ and get the differential equation $\Delta X_{R-3}[1] = \Delta X_{R-3}[13]$. Then we guess the tweakey nibbles in Set_1 and Set_2 to get the values of the left and the right side of the equation respectively, where $Set_1 = \{tk_2, tk_5, tk_9, tk_{11}, tk_{12}, tk_{15}\}$, $Set_2 = \{tk_6, tk_7, tk_{10}, tk_{13}, tk_{15}\}$. We can utilize 10 pairs of correct and faulty ciphertexts with differences in $X_{R-9}[13]$ to reduce the number of possible values of the 10 tweakey nibbles. We choose 10 random plaintexts and inject random faults at $X_{R-9}[13]$. This phase costs about 2^{24} time and 2^{20} memory. And for the remaining 6 nibbles $\{tk_0, tk_1, tk_3, tk_4, tk_8, tk_{14}\}$, we exhaustively search and test with the correct plaintext and ciphertext, which costs 2^{24} time for encryption.

4.1.2 Simulation results

We perform the simulation experiments according to our methods in Sect. 4.1.1. We randomly choose keys and repeat the attack process for 1000 times. In each experiment, we vary the number of injected faults in each filter step and calculate the average number of key candidates after filtering. Our results are summarized in Figure 6, where the x-axis represents the number of fault injections in each filter step, and the y-axis represents the average number of remaining key candidates across 1000 simulations. For example, the red triangle point with the x-axis coordinate of 3 represents the number of remaining keys after filtering by 3 pairs of ciphertexts in `Filter 2`, which is $2^{58.2}$. In our experiments, with all 9 fault injections, the number of remaining key candidates after `Filter 1-4` is always 2^{24}, which validates that our attack is effective. All simulations are performed on a PC using C code, and each key recovery operation takes a few minutes. Our codes are available at https://anonymous.4open.science/r/MITM-DFA-FD0C/.

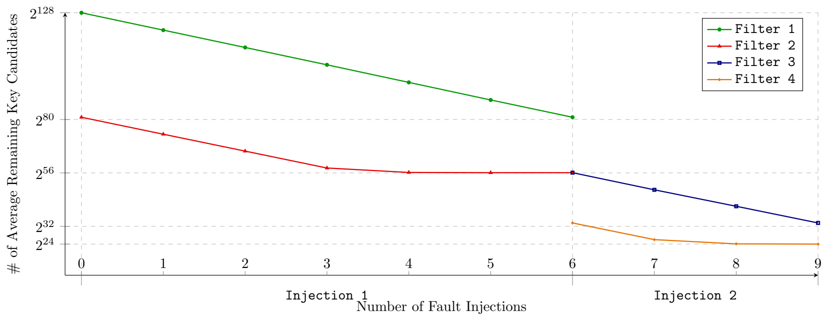

Figure 6: Simulation results

4.1.3 DFA on SKINNY-n-2n

The attack in Sect. 4.1.1 targets at the SKINNY-n-n, where only TK_1 is used. SKINNY also supports versions with $t = 2n$ and $t = 3n$. In the TWEAKEY framework, users can choose what part of the tweakey serves as input key material or tweak material, and TK_1 is recommended for processing the public tweak material. In this section, we consider the case of SKINNY-n-2n, where TK_1 is used for tweak and TK_2 is used for key material.

According to our injection model in Figure 5, we can extract $\mathcal{L}_1^{R-3}(TK_1) \oplus \mathcal{L}_2^{R-3}(TK_2)[1]$, $\mathcal{L}_1^{R-2}(TK_1) \oplus \mathcal{L}_2^{R-2}(TK_2)[0,6]$ and $\mathcal{L}_1^{R-1}(TK_1) \oplus \mathcal{L}_2^{R-1}(TK_2)[4,5,7]$ in `Filter 1`, where $\mathcal{L}_1$ and $\mathcal{L}_2$ denote the linear operations on TK_1 and TK_2. $\mathcal{L}_1$ only represents the permutation P_T whereas $\mathcal{L}_2$ denotes the combination of P_T and LFSR on TK_2. Since the tweak TK_1 can be seen as public, we can compute $\mathcal{L}_2^{R-3}(TK_2)[1]$, $\mathcal{L}_2^{R-2}(TK_2)[0,6]$ and $\mathcal{L}_2^{R-1}(TK_2)[4,5,7]$ straightforwardly from `Filter 1`. Then we can determine the corresponding bytes in TK_2 by inverting the LFSR and the permutation P_T round by round. The same for other filter phases.

In the actual situation, the security does not rely on the privacy of tweak material, and it is also unrealistic to keep privacy in practice. So we can retrieve the key material for `SKINNY-n-2n` as same as for `SKINNY-n-n`. It indicates that the tweak material does not provide any extra protection against fault attack if keeping fixed.

4.2 Differential fault attack on `CRAFT`

4.2.1 Description of `CRAFT`

`CRAFT` [BLMR19] is a lightweight tweakable block cipher with a 64-bit block size, a 128-bit key K, and a 64-bit tweak T. The cipher's internal state can be represented as a 4×4 square array of nibbles or as a 16-nibble vector by concatenating the rows of the square array. Use the index $4 \times i + j$ to denote the nibble at row i and column j of the 4×4 array, where $0 \leq i \leq 3,\ 0 \leq j \leq 3$.

For `CRAFT`, each round function applies five involutory round operations: `SubSbox(SB)`, `MixColumn(MC)`, `PermuteNibbles(PN)`, `AddConstant(ARC)` and `AddTweakey(ATK)`. The last round omits `SubSbox(SB)` and `PermuteNibbles(PN)`. The r-th round function R_r is defined as $R_r = \texttt{SB} \circ \texttt{PN} \circ \texttt{ATK} \circ \texttt{ARC} \circ \texttt{MC}$ $(1 \leq r \leq 31)$, and the last round $R_{32} = \texttt{ATK} \circ \texttt{ARC} \circ \texttt{MC}$. The involutory binary matrix M used in `MixColumn(MC)` is non-MDS, where

$$M = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

In the key schedule, four 64-bit tweakeys TK_0, TK_1, TK_2, TK_3 are derived from $K = (K_0||K_1)$ and T, as $TK_0 = K_0 \oplus T$, $TK_1 = K_1 \oplus T$, $TK_2 = K_0 \oplus Q(T)$, $TK_3 = K_1 \oplus Q(T)$, where Q is a permutation and $TK_{(i-1) \bmod 4}$ is applied in the i-th round.

When guessing the key cells, we can choose the nibbles in equivalent subkeys, *i.e.* we can adjust the operations in round function as $R_i = \texttt{SB} \circ \texttt{PN} \circ \texttt{ARC} \circ \texttt{MC} \circ \texttt{ATK}'$, the key used in $\texttt{ATK}'$ is the equivalent subkey $\texttt{MC}^{-1}(TK_i)$.

4.2.2 DFA on `CRAFT`

`CRAFT` [BLMR19] is designed to be efficiently protected in its implementations against differential fault analysis. However, we assume that the target implementation of `CRAFT` does not use such countermeasures, as studied in [RVB22]. In round r, denote the state at the beginning of round r as X_r, the state after `MC` as Y_r, and the internal state after `PN` as Z_r. Further, $A[i]$ denotes the i-th nibble of the state A, where $0 \leq i \leq 15$.

- **Injection 1:** As shown in Figure 7, we inject random nibble faults at $Z_{R-10}[7]$, the differential equation $\Delta Y_{R-3}[5] = \Delta Y_{R-3}[13]$ can be derived. And we guess the subkey nibbles $\{\texttt{MC}^{-1}(K_0)[3,4,13],\ \texttt{MC}^{-1}(K_1)[0,1,2,7,10,14]\}$ to compute $\Delta Y_{R-3}[13]$ by partial decryption. Similarly, guess $\{\texttt{MC}^{-1}(K_0)[5],\ \texttt{MC}^{-1}(K_1)[2,9]\}$ to obtain $\Delta Y_{R-3}[5]$.

  Because the differential equation provides a filter of one nibble, we can retrieve the subkey nibbles with 11 pairs of correct and faulty ciphertexts. We choose 11 plaintexts

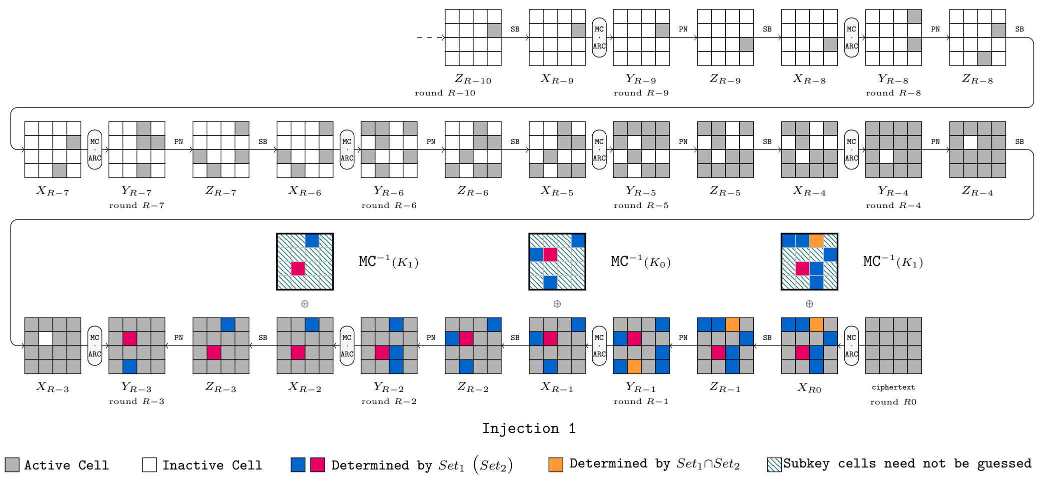

Figure 7: DFA on `CRAFT`.

and obtain the ciphertexts, then inject random faults at $Z_{R-10}[7]$ to each and query 11 faulty ciphertexts. Filtering the subkey nibbles with differential equation $\Delta Y_{R-3}[5] = \Delta Y_{R-3}[13]$, there is about 1 value of the key nibbles remaining. We need to inject 11 random nibble faults. The time complexity is about 2^{36} and the memory cost is about 2^{12}.

- **Injection 2:** In this step, we inject random nibble faults at $Z_{R-10}[5]$, and get the differential equation $\Delta Y_{R-3}[7] = \Delta Y_{R-3}[15]$. We can filter the subkey nibbles $\{\mathtt{MC}^{-1}(K_0)[7], \mathtt{MC}^{-1}(K_1)[11]\}$ and $\{\mathtt{MC}^{-1}(K_0)[1,6,15], \mathtt{MC}^{-1}(K_1)[3,5,8,12]\}$ with 9 pairs of correct and faulty ciphertexts. The time complexity is about 2^{28} and the memory complexity is about 2^8.
- **Injection 3:** Then we inject random nibble faults at $Z_{R-10}[6]$, filter the subkey nibbles $\{\mathtt{MC}^{-1}(K_0)[2,12], \mathtt{MC}^{-1}(K_1)[4,13]\}$ by $\Delta Y_{R-3}[4] = \Delta Y_{R-3}[12]$ with 4 pairs of correct and faulty ciphertexts. The time cost is 2^{16}.
- **Injection 4:** Finally we inject random nibble faults at $Z_{R-10}[4]$, filter the subkey nibbles $\{\mathtt{MC}^{-1}(K_0)[0,14], \mathtt{MC}^{-1}(K_1)[6,15]\}$ by $\Delta Y_{R-3}[6] = \Delta Y_{R-3}[14]$ with 4 pairs of correct and faulty ciphertexts. And then exhaustively search the remaining four subkey nibbles, the time cost is 2^{16}.

The overall time complexity of the attack is about 2^{36}, and the memory cost is about 2^{12}. The number of faults we need to inject in all phases is $11 + 9 + 4 + 4 = 28$.

4.3 Differential fault attack on `QARMA`

4.3.1 Description of `QARMA`

`QARMA` [Ava17] is a family of lightweight tweakable block cipher. It supports two kinds of block sizes $n = 64$ and $n = 128$, denoted by `QARMA-64` and `QARMA-128`. Each version applies a master key of $2n$ bits. `QARMA` uses an Even-Mansour scheme with a keyed pseudo-reflector. The first r rounds of the cipher use the forward round function $\mathcal{R}(IS, tk)$, which is composed of four operations in order: `AddRoundTweakey`, `ShuffleCells`(τ), `MixColumns`(M) and `SubCells`(S). The $(r+1)$-th round function omits `ShuffleCells` and `MixColumns` operations. The last r rounds use the backward round function $\overline{\mathcal{R}}(IS, tk)$, which is the inverse of $\mathcal{R}(IS, tk)$. The round function used in the first and last rounds

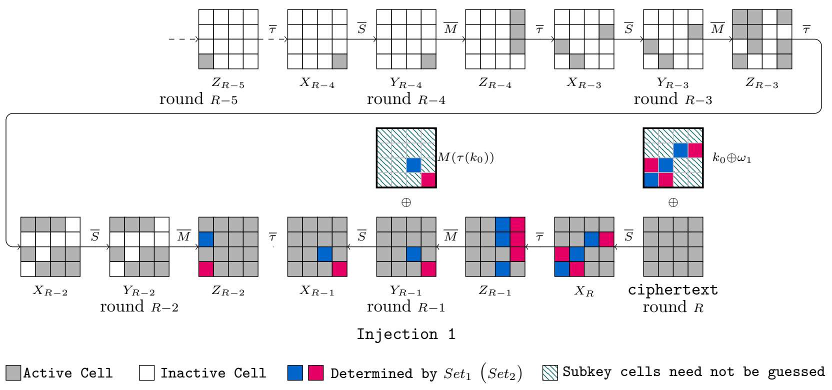

Figure 8: DFA on QARMA-64.

omits ShuffleCells and MixColumns. The matrix M used in MixColumns is defined as

$$M_4 = circ(0, \rho, \rho^2, \rho) \text{ and } M_8 = circ(0, \rho, \rho^4, \rho^5),$$

where M_4 is uesd for QARMA-64 and M_8 is used for QARMA-128. ρ can be seen as a simple circular left rotation of the bits. The $2n$-bit K is partitioned as $\omega_0||k_0$. In encryption, $k_1 = k_0$ is used for each round and whitening keys ω_0 and $\omega_1 = o(\omega_0)$ are added at the beginning and the end, respectively. In the key-recovery phase, we will guess the equivalent key $M(\tau(k_0))$.

4.3.2 DFA on QARMA-64

Because the matrix M used in MixColumns is different in each version, QARMA-64 and QARMA-128 have different differential matching rules. In this section, we show how to achieve the key recovery attack on QARMA-64. Part of the attack is shown in Figure 8.

- **Injection 1:** Inject random nibble faults at $X_{R-4}[15]$ ($Z_{R-5}[13]$), filter the subkey nibbles $\{M(\tau(k_0))[10],\ k_0 \oplus \omega_1[6, 9, 12]\}$ and $\{M(\tau(k_0))[15],\ k_0 \oplus \omega_1[7, 8, 13]\}$ by the differential equation $\Delta Z_{R-2}[4] = \Delta Z_{R-2}[12]$ with 8 pairs of correct and faulty ciphertexts.
- **Injection 2:** Inject random nibble faults at $X_{R-4}[12]$ ($Z_{R-5}[6]$), filter the subkey nibbles $\{M(\tau(k_0))[5],\ k_0 \oplus \omega_1[4, 11, 14]\}$ and $\{M(\tau(k_0))[0], k_0 \oplus \omega_1[5, 10, 15]\}$ by $\Delta Z_{R-2}[8] = \Delta Z_{R-2}[0]$ with 8 pairs of correct and faulty ciphertexts.
- **Injection 3:** Inject random nibble faults at $X_{R-4}[0]$ ($Z_{R-5}[0]$), filter the subkey nibbles $\{M(\tau(k_0))[6],\ k_0 \oplus \omega_1[3]\}$ and $\{M(\tau(k_0))[3], k_0 \oplus \omega_1[18]\}$ by $\Delta Z_{R-2}[2] = \Delta Z_{R-2}[10]$ with 4 pairs of correct and faulty ciphertexts.
- **Injection 4:** Inject random nibble faults at $X_{R-4}[6]$ ($Z_{R-5}[2]$), filter the subkey nibbles $\{M(\tau(k_0))[1],\ k_0 \oplus \omega_1[1]\}$ and $\{M(\tau(k_0))[4], k_0 \oplus \omega_1[0]\}$ by $\Delta Z_{R-2}[5] = \Delta Z_{R-2}[13]$ with 4 pairs of correct and faulty ciphertexts.
- **Injection 5** and **Injection 6** respectively inject random faults at $X_{R-4}[3]$ and $X_{R-4}[4]$. **Injection 5** filter $M(\tau(k_0))[9, 12]$ by $\Delta Z_{R-2}[6] = \Delta Z_{R-2}[14]$, and **Injection 6** filter $M(\tau(k_0))[11, 14]$ by $\Delta Z_{R-2}[1] = \Delta Z_{R-2}[9]$. Each filter phase needs 2 pairs of correct and faulty ciphertexts.

After the six injection phases, we exhaustively search for the remaining 4 subkey nibbles $M(\tau(k_0))[2, 7, 8, 13]$ and retrieve subkey k_0 by linear transformation $\overline{\tau} \circ \overline{M}$. Then we compute ω_1 by k_0 and $k_0 \oplus \omega_1$. The overall time complexity of our attack is about 2^{16} and the memory complexity is about 2^{16}. It requires the injection of 28 nibble faults.

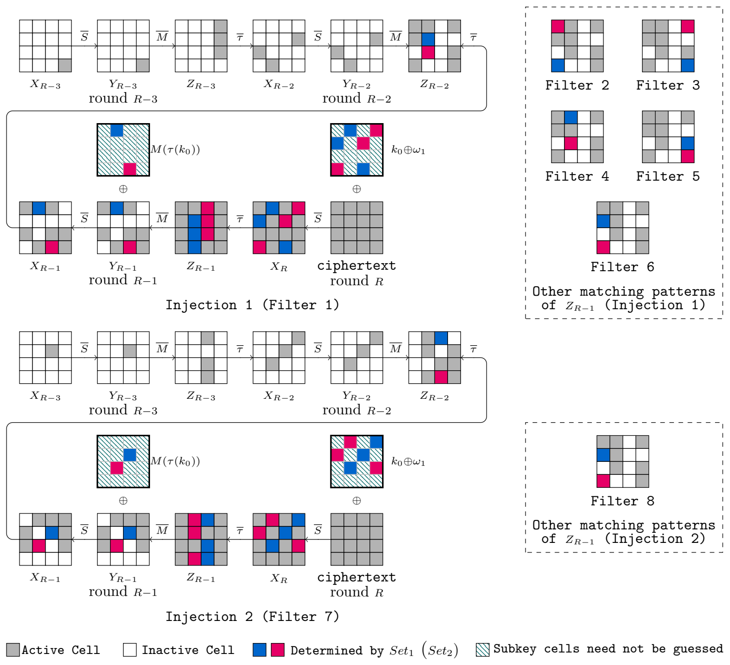

Figure 9: DFA on QARMA-128.

4.3.3 DFA on QARMA-128

In this section, we show the process of our fault attack on QARMA-128 with 2^{32} time complexity and 2^{32} memory complexity.

- **Injection 1:** We inject random faults at $X_{R-3}[15]$ and get following equation system:

$$\begin{cases} \Delta Z_{R-2}[5] = \rho^3 \cdot \Delta Z_{R-2}[9], & \rho \cdot \Delta Z_{R-2}[0] = \Delta Z_{R-2}[12], \\ \rho^3 \cdot \Delta Z_{R-2}[3] = \Delta Z_{R-2}[15], & \rho^4 \cdot \Delta Z_{R-2}[1] = \Delta Z_{R-2}[9], \\ \Delta Z_{R-2}[11] = \rho \cdot \Delta Z_{R-2}[15], & \rho^4 \cdot \Delta Z_{R-2}[4] = \Delta Z_{R-2}[12]. \end{cases} \tag{5}$$

  Each equation provides a one-byte filter for involved subkey bytes. For example, we choose 8 plaintexts and inject random faults to get the correct and faulty ciphertexts. We can then guess the subkey bytes $\{M(\tau(k_0))[1], k_0 \oplus \omega_1[1,4,14]\}$ to compute $\Delta Z_{R-2}[5]$ and guess $\{M(\tau(k_0))[14], k_0 \oplus \omega_1[3,6,12]\}$ to compute $\Delta Z_{R-2}[14]$ for the 8 pairs. By filtering with $\rho^3 \cdot \Delta Z_{R-2}[5] = \Delta Z_{R-2}[9]$, we can retrieve the value of the 8 subkey bytes.
  Similarly, we can retrieve the subkey bytes $M(\tau(k_0))[0,2,8,10,11,13,15]$ and $k_0 \oplus \omega_1[0,2,5,7,8,9,10,11,13,15]$ with the remaining equations. We can use the 8 pairs of correct and faulty ciphertexts obtained in the first step to accomplish this.

- **Injection 2:** Then we inject random faults at $X_{R-3}[6]$ and get 2 differential equations for a further filter of subkey bytes,

$$\rho^3 \cdot \Delta Z_{R-2}[2] = \Delta Z_{R-2}[14],\ \Delta Z_{R-2}[7] = \rho^4 \cdot \Delta Z_{R-2}[15]. \tag{6}$$

  Only at most one unknown subkey byte is involved in one hand of each equation. So we choose 2 plaintexts and inject faults to get the correct and faulty ciphertexts. We expect to get the right value of $M(\tau(k_0))[6,7,9]$.

Then we exhaustively search for the remaining 4 subkey bytes $M(\tau(k_0))[3,4,5,12]$ to retrieve the full key. The overall time complexity of this attack is about 2^{32} and the memory cost is 2^{32}. The number of faults we need to inject in the attack is 10.

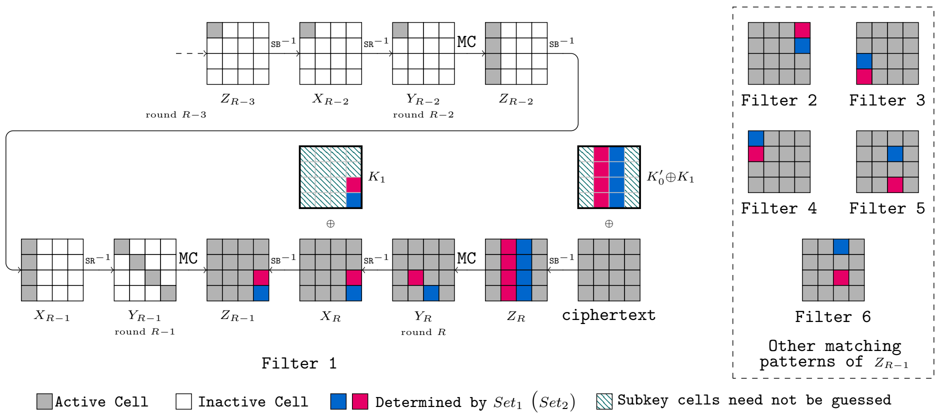

Figure 10: DFA on PRINCE.

4.4 Differential fault attack on PRINCE and PRINCEv2

4.4.1 Description of PRINCE

PRINCE [BCG+12] is a lightweight block cipher that follows FX-construction with a 64-bit block size and a 128-bit key size. The 128-bit key $K = K_0||K_1$ is split into two 64-bit parts. K_0 is used as a whitening key and K_1 is used as round keys for the core of the structure, named as PRINCEcore. The round function R of PRINCEcore applies an S-box layer SB, followed by a linear layer consisting of a MixColumns operation MC and a ShiftRows SR. SR is the same as AES, and the matrices in MC-layer are built from the following four matrices:

$$m_1 = \begin{pmatrix} 0&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1 \end{pmatrix}, m_2 = \begin{pmatrix} 1&0&0&0\\0&0&0&0\\0&0&1&0\\0&0&0&1 \end{pmatrix}, m_3 = \begin{pmatrix} 1&0&0&0\\0&1&0&0\\0&0&0&0\\0&0&0&1 \end{pmatrix}, m_4 = \begin{pmatrix} 1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&0 \end{pmatrix}.$$

In MC-layer, bit-wise matrix M_0 is multiplied with the first and the last columns of the internal state, and M_1 is multiplied with the second and the third columns.

$$M_0 = \begin{pmatrix} m_1&m_2&m_3&m_4\\m_2&m_3&m_4&m_1\\m_3&m_4&m_1&m_2\\m_4&m_1&m_2&m_3 \end{pmatrix}, M_1 = \begin{pmatrix} m_2&m_3&m_4&m_1\\m_3&m_4&m_1&m_2\\m_4&m_1&m_2&m_3\\m_1&m_2&m_3&m_4 \end{pmatrix}.$$

The last five rounds apply the inverse of the forward round function R^{-1}, and the middle layer applies R'. The round function R, R' and R^{-1} are shown as:

$$R = \texttt{SR} \circ \texttt{MC} \circ \texttt{SB}, \; R' = \texttt{SB}^{-1} \circ \texttt{MC} \circ \texttt{SB}, \; \text{ and } \; R^{-1} = \texttt{SB}^{-1} \circ \texttt{MC} \circ \texttt{SR}^{-1}.$$

PRINCEv2 [BEK+20] modifies the middle layer and key schedule of PRINCE. The 128-bit key is departed into $K = (K_0||K_1)$ and alternately used in each function. The subkeys used in the last two rounds are $K_0 \oplus \alpha$ and $K_1 \oplus \beta$, where α and β are constants. The process for retrieving the full key is similar to that for PRINCE.

4.4.2 DFA on PRINCE

We can recover the full key of PRINCE with random faults injected at $Z_{R-3}[0]$. As shown in Figure 10, we can build the following equation system according to differential characters:

$$\begin{cases} m_1 \cdot \Delta Z_{R-1}[3] = m_4 \cdot \Delta Z_{R-1}[7], & m_2 \cdot \Delta Z_{R-1}[3] = m_4 \cdot \Delta Z_{R-1}[11], \\ m_3 \cdot \Delta Z_{R-1}[3] = m_4 \cdot \Delta Z_{R-1}[15], & m_2 \cdot \Delta Z_{R-1}[7] = m_1 \cdot \Delta Z_{R-1}[11], \\ m_3 \cdot \Delta Z_{R-1}[7] = m_1 \cdot \Delta Z_{R-1}[15], & m_3 \cdot \Delta Z_{R-1}[11] = m_2 \cdot \Delta Z_{R-1}[15]. \end{cases} \quad (7)$$

We guess $\{K_1[11],\ K'_0 \oplus K_1[1,5,9,13]\}$ to compute $\Delta Z_{R-1}[11]$ and filter the values of $\{K_1[15],\ K'_0 \oplus K_1[2,6,10,14]\}$ by $m_3 \cdot \Delta Z_{R-1}[11] = m_2 \cdot \Delta Z_{R-1}[15]$. This equation provides a 2-bit filter for each correct and faulty ciphertext pair. With 10 pairs of correct and faulty ciphertext pairs, we get 20-bit information on these 10 subkey nibbles.

Similarly, we can get 20-bit information of $\{K_1[3,7],\ K'_0 \oplus K_1[0,3,4,7,8,11,12,15]\}$ by the filter of $m_1 \cdot \Delta Z_{R-1}[3] = m_4 \cdot \Delta Z_{R-1}[7]$. We can obtain further filters by other equations in Equ. (7). For example, $m_3 \cdot \Delta Z_{R-1}[7] = m_1 \cdot \Delta Z_{R-1}[15]$ provides one more bit filter for $\Delta Z_{R-1}[15]$ and $\Delta Z_{R-1}[7]$.

Then we can build equation systems for other columns of ΔZ_{R-1}, like

$$m_4 \cdot \Delta Z_{R-1}[8] = m_3 \cdot \Delta Z_{R-1}[12],\ m_2 \cdot \Delta Z_{R-1}[0] = m_1 \cdot \Delta Z_{R-1}[4] \tag{8}$$

to filter the subkey nibbles. We expect to get the correct value of $K'_0 \oplus K_1$ and at least 48-bit information of K_1. We exhaustively search for the remaining possible values of K_1, which costs about 2^{16} time, and compute K'_0 to retrieve the full key. The overall time and memory complexities are both about 2^{20}, with 10 fault injections.

4.5 Differential fault attack on MIDORI

4.5.1 Description of MIDORI

MIDORI [BBI+15] is a family of low-energy block cipher, which includes two versions with block size $n = 64$ and $n = 128$, i.e., MIDORI-64 and MIDORI-128. Both versions accept a 128-bit keys K. For MIDORI-64, K is denoted as two 64-bit parts $K = K_0 \| K_1$. The whitening key is $WK = K_0 \oplus K_1$ and the round key is $RK_r = K_{(r-1) \bmod 2} \oplus \alpha_r$ for $1 \leq r \leq 15$. For MIDORI-128, $WK = K$ and $RK_r = K \oplus \beta_r$ for $1 \leq r \leq 19$, where α_r and β_r are constants. The round function of MIDORI consists of SubSbox(SB), ShuffleCell(SC), MixColumn(MC), and KeyAdd(AK). The ShuffleCell(SC) applies a cellwise permutation P to the internal state, where $P = [0,11,6,13,10,1,12,7,5,14,3,8,15,4,9,2]$. In MixColumn(MC) operation, each column of the state array is multiplied by the matrix M cellwise, where $M = circ(0,1,1,1)$. The last round only applies SubSbox(SB) operation and adds the whitening key WK to the internal state.

4.5.2 DFA on MIDORI

DFA on MIDORI-64. We firstly inject random faults at $X_{R-4}[2]$ as shown in Figure 11, and get three differential equations

$$\Delta X_{R-1}[5] = \Delta X_{R-1}[13],\ \Delta X_{R-1}[0] = \Delta X_{R-1}[4],\ \Delta X_{R-1}[7] = \Delta X_{R-1}[11]. \tag{9}$$

Each equation represents a filter step (Filter 1-3). Filter 1 guess $\{\texttt{MC}^{-1}(K_0)[8],\ K_0 \oplus K_1[0,4,12]\}$ to compute $\Delta X_{R-1}[5]$, and guess $\{\texttt{MC}^{-1}(K_0)[3],\ K_0 \oplus K_1[7,11,15]\}$ to compute $\Delta X_{R-1}[13]$, then filter the subkey nibbles by $\Delta X_{R-1}[5] = \Delta X_{R-1}[13]$. We expect to get one candidate of $\{\texttt{MC}^{-1}(K_0)[0,1,3,7,8,13],\ K_0 \oplus K_1[0,1,3,4,5,7,8,9,11,12,13,15]\}$ after Filter 1-3 with 8 pairs of correct and faulty ciphertexts.

Secondly, we inject faults at $X_{R-4}[9]$ and filter the subkey nibbles with equations

$$\Delta X_{R-1}[3] = \Delta X_{R-1}[15],\ \Delta X_{R-1}[8] = \Delta X_{R-1}[12]. \tag{10}$$

We expect to get one right value of $\{\texttt{MC}^{-1}(K_0)[6,10,11,12], K_0 \oplus K_1[2,6,10,14]\}$ with 5 pairs of correct and faulty ciphertexts (Filter 4-5).

Then we inject faults at $X_{R-4}[4]$ and filter $\texttt{MC}^{-1}(K_0)[2,5,9,14]$ with the equations $\Delta X_{R-1}[1] = \Delta X_{R-1}[9]$ and $\Delta X_{R-1}[6] = \Delta X_{R-1}[14]$. We expect to get the right value with 2 pairs of correct and faulty ciphertexts.

Finally, we exhaustively search for the values of $\texttt{MC}^{-1}(K_0)[4,15]$ and compute K_0 and K_1 to recover the full key. The overall time complexity is about 2^{16} and the memory complexity is 2^{16}, and the number of faults we need to inject is 15.

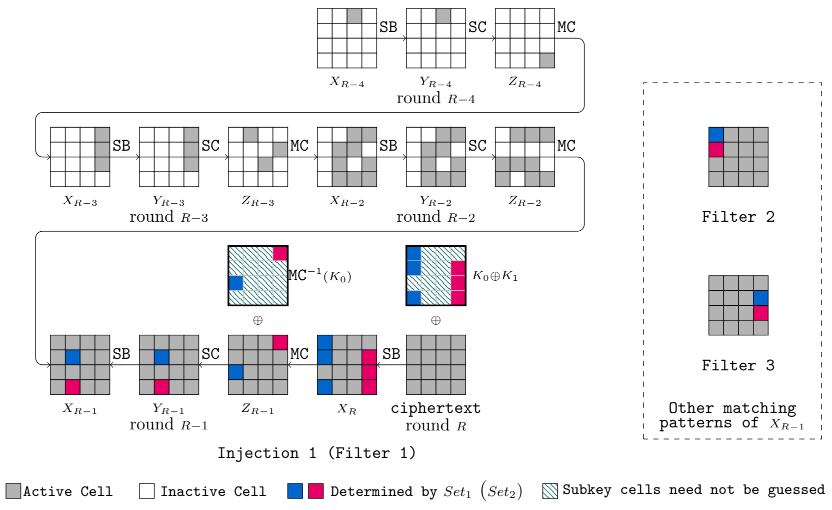

Figure 11: DFA on `MIDORI`.

DFA on `MIDORI-128`. For `MIDORI-128`, the key used in the last two rounds is the same except for the constant addition. We can recover $K[0,1,3,4,5,7,8,9,11,12,13,15]$ according to the filter of Equ. (9) with faults injected to $X_{R-4}[2]$. To compute the value of each side in Equ. (9), we need to guess 3 bytes of K, for example, $K[0,1,12]$ for $\Delta X_{R-1}[5]$. This phase can be done with 6 pairs of correct and faulty ciphertexts. Then, we inject faults at $X_{R-4}[9]$ and filter the subkey bytes according to Equ. (10). We expect to get the right value of the full key with a filter of 3 pairs. Thus, the overall time complexity of the attack on `MIDORI-128` is about 2^{24}, and the memory cost is also 2^{24}, with 9 fault injections.

4.6 Extended Attack on `AES-192`

Derbez *et al.* simply extended the DFA attack on `AES-128` to `AES-192` in [DFL11]. The DFA attack on `AES-128` is stated in Sect. 2.1. Figure 1 represents the last four rounds of `AES-192`. The extended DFA attack on `AES-192` is briefly given as follows.

Step I. They first recover the key cells of K_{12} for `AES-192` using the same MitM algorithm as `AES-128`. Substitute R with 12, and denote differential state $X_{11} \oplus \tilde{X}_{11}$ as ΔX_{11}. Filter the values of $\{K_{12}[1,4,11,14], U_{11}[7]\}$, $\{K_{12}[2,5,8,15], U_{11}[10]\}$, $\{K_{12}[0,7,10,13], U_{11}[0]\}$ and $\{K_{12}[3,6,9,12], U_{11}[13]\}$ according to the equations between $\Delta X_{11}[0]$, $\Delta X_{11}[1]$, $\Delta X_{11}[2]$ and $\Delta X_{11}[3]$. The time of this step is the same as the DFA on `AES-128`, which is 3×2^{40}.

Step II. After recovering the 128-bit subkey K_{12}, in order to recover the full 192-bit key, they peel off the last round of `AES-192` to perform a 3-round (Round 9-11) differential fault attacks following the idea of Piret and Quisquater [PQ03] with the same correct and faulty pairs. Therefore, an additional time complexity is required to implement Piret and Quisquater's attack, which is estimated as 2^{40} "Piret and Quisquater resolution" by Derbez *et al.* [DFL11].

4.6.1 Our Improved Attack

We introduce a new MitM DFA on `AES-192`, which uses the same Step I as Derbez *et al.* [DFL11] to recover K_{12}. But we do not use Piret and Quisquater's attack (Step II) to recover the remaining key bits.

A new Step II: After recovering K_{12}, we also peal off the last round. In order to recover the full 192-bit key, we need to recover the last two columns of K_{11}, which is equivalent to recovering $U_{11}[8-15]$. Among those bytes, $U_{11}[10,13]$ are already recovered in Step I.

After peel off Round 12, we can derive $\Delta X_{11}[8] = 3 \cdot \Delta X_{11}[11]$, which is equivalent to

$$(SB^{-1}(A \oplus U_{11}[8]) \oplus SB^{-1}(\tilde{A} \oplus U_{11}[8])) = 3 \cdot SB^{-1}(B \oplus U_{11}[15]) \oplus SB^{-1}(\tilde{B} \oplus U_{11}[15]), \quad (11)$$

where A and B are known values at this stage and only depend on the correct ciphertext and K_{12}, similar to $\tilde{A}$ and $\tilde{B}$. Then, we use Eq. (11) as a filter to build a local MitM attack, where two correct-faulty ciphertext pairs will provide a filter of 2^{-16}. Therefore, only one candidate of $U_{11}[8,15]$ will remain. The time complexity of local MitM is 2×2^8 with about 2^8 memory.

Similarly, we can recovery $U_{11}[11,14]$ and $U_{11}[9,12]$ with matching equations like Eq. (11). The last two columns of K_{11} are recovered. The total time complexity of our new Step II is 6×2^8 with about 2^8 memory, which is smaller than Derbez *et al.*'s Step II.

Two correct-faulty ciphertext pairs needed in our new Step II can reuse the pairs from Step I, and no additional fault injections are needed here. This process can be also applied to `AES-256` to improve Step II.

5 Discussion

In order to achieve lightweight, many lightweight ciphers adopt sparser diffusion layers. For example, `MIDORI`, `PRINCE` and `QARMA` adopt the so-called 4×4 almost MDS layer. Note that the branch numbers (the smallest nonzero sum of active inputs and outputs of the matrix) of MDS and almost MDS matrices are 5 and 4. Therefore, `MIDORI` needs 3 rounds to achieve full diffusion (any input bit nonlinearly affects all the state bits after 3 rounds), while `AES` needs only 2 rounds. For `SKINNY` and `CRAFT`, the diffusion layers are more lightweight than `MIDORI` etc., so full diffusions are achieved after 6 rounds and 7 rounds, respectively.

The main idea behind MitM DFA is to explore the potentially low diffusion of the target ciphers. With the lower diffusion layers, the number of key bits involved in the two neutral key sets (e.g., Set_1 and Set_2 in Figure 2) are smaller. It actually dominates the time complexity of the MitM attack, because the keys in Set_1 (and Set_2) must be enumerated to compute the differential equation (e.g., Eq. (2)). Besides, with a lower diffusion layer, a differential fault injected will propagate more slowly, so that a deterministic differential relation (e.g., Eq. (2)) exists with higher probability after more rounds than a cipher with stronger diffusion, which will act as the matching to filter the wrong key guessing. These are the reasons behind the attacks summarized in Table 1, where for `SKINNY` and `CRAFT`, the MitM DFA can achieve 9 and 10 rounds, but for `MIDORI`, `PRINCE` and `QARMA`, the attack can only work on 4 and 5 rounds.

5.1 Countermeasures

The double-check mechanism is a common countermeasure against fault injection attacks [MSY06, ML08, JMR07, BBK+10]. The crucial operations of encryption devices, which are vulnerable to fault analysis, should run twice. If the results of the two executions match each other, the results are credible. Indeed, this mechanism is always accompanied by a loss of efficiency. Our improved DFAs in Sect. 4 provide new insight into how many rounds of encryption devices need to be protected. For example, we suggest protecting at least the last 9 rounds of `SKINNY-128-128`, with the double-check mechanism, the running procedure of `SKINNY-128-128` is shown below:

1. Compute $State_1 = Round_{1-31}(Message, Key)$, $State_2 = State_1$.

2. Compute $Res_1 = Round_{32-40}(State_1, Key)$, and $Res_2 = Round_{32-40}(State_2, Key)$.
3. After a random delay, check whether $Res_1 = Res_2$. If yes, output Res_1 as ciphertext, else discard the result.

This fault detection scheme requires about 49/40 computational sources of `SKINNY-128-128`, and provides more reliable protection than parity check schemes.

6 Conclusion

In this paper, we present the MILP-based automatic tools for MitM DFAs and apply to `SKINNY`, `CRAFT`, `QARMA`, `PRINCE` and `MIDORI`. We make full use of the differential matching rules and reach better key-recovery attacks for these block ciphers in practical time. We achieve fault attacks with faults injected in earlier rounds, which imply that more rounds of the encryption devices of these ciphers should be protected against DFA.

References

[AK97] Ross J. Anderson and Markus G. Kuhn. Low cost attacks on tamper resistant devices. In Bruce Christianson, Bruno Crispo, T. Mark A. Lomas, and Michael Roe, editors, *Security Protocols, 5th International Workshop, Paris, France, April 7-9, 1997, Proceedings*, volume 1361 of *Lecture Notes in Computer Science*, pages 125–136. Springer, 1997.

[Ava17] Roberto Avanzi. The QARMA block cipher family. almost MDS matrices over rings with zero divisors, nearly symmetric even-mansour constructions with non-involutory central rounds, and search heuristics for low-latency s-boxes. *IACR Trans. Symmetric Cryptol.*, 2017(1):4–44, 2017.

[BBB$^+$21] Anubhab Baksi, Shivam Bhasin, Jakub Breier, Mustafa Khairallah, Thomas Peyrin, Sumanta Sarkar, and Siang Meng Sim. DEFAULT: cipher level resistance against differential fault attack. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part II*, volume 13091 of *Lecture Notes in Computer Science*, pages 124–156. Springer, 2021.

[BBI$^+$15] Subhadeep Banik, Andrey Bogdanov, Takanori Isobe, Kyoji Shibutani, Harunaga Hiwatari, Toru Akishita, and Francesco Regazzoni. Midori: A block cipher for low energy. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, volume 9453 of *Lecture Notes in Computer Science*, pages 411–436. Springer, 2015.

[BBK$^+$10] Alessandro Barenghi, Luca Breveglieri, Israel Koren, Gerardo Pelosi, and Francesco Regazzoni. Countermeasures against fault attacks on software implemented AES: effectiveness and cost. In *Proceedings of the 5th Workshop on Embedded Systems Security, WESS 2010, Scottsdale, AZ, USA, October 24, 2010*, page 7. ACM, 2010.

[BCG$^+$12] Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knezevic, Lars R. Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar, Christian Rechberger, Peter Rombouts, Søren S. Thomsen, and Tolga Yalçin.

PRINCE - A low-latency block cipher for pervasive computing applications - extended abstract. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, volume 7658 of *Lecture Notes in Computer Science*, pages 208–225. Springer, 2012.

[BDF11] Charles Bouillaguet, Patrick Derbez, and Pierre-Alain Fouque. Automatic search of attacks on round-reduced AES and applications. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*, pages 169–187. Springer, 2011.

[BDG+21] Zhenzhen Bao, Xiaoyang Dong, Jian Guo, Zheng Li, Danping Shi, Siwei Sun, and Xiaoyun Wang. Automatic search of meet-in-the-middle preimage attacks on aes-like hashing. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part I*, volume 12696 of *Lecture Notes in Computer Science*, pages 771–804. Springer, 2021.

[BDL97] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults (extended abstract). In Walter Fumy, editor, *Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, volume 1233 of *Lecture Notes in Computer Science*, pages 37–51. Springer, 1997.

[BEK+20] Dusan Bozilov, Maria Eichlseder, Miroslav Knezevic, Baptiste Lambin, Gregor Leander, Thorben Moos, Ventzislav Nikov, Shahram Rasoolzadeh, Yosuke Todo, and Friedrich Wiemer. Princev2 - more security for (almost) no overhead. In Orr Dunkelman, Michael J. Jacobson Jr., and Colin O'Flynn, editors, *Selected Areas in Cryptography - SAC 2020 - 27th International Conference, Halifax, NS, Canada (Virtual Event), October 21-23, 2020, Revised Selected Papers*, volume 12804 of *Lecture Notes in Computer Science*, pages 483–511. Springer, 2020.

[BGST22] Zhenzhen Bao, Jian Guo, Danping Shi, and Yi Tu. Superposition meet-in-the-middle attacks: Updates on fundamental security of aes-like hashing. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part I*, volume 13507 of *Lecture Notes in Computer Science*, pages 64–93. Springer, 2022.

[BHJ+18] Jakub Breier, Xiaolu Hou, Dirmanto Jap, Lei Ma, Shivam Bhasin, and Yang Liu. Practical fault attack on deep neural networks. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 2204–2206. ACM, 2018.

[BJK+16] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The

SKINNY family of block ciphers and its low-latency variant MANTIS. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 123–153. Springer, 2016.

[BK06] Johannes Blömer and Volker Krummel. Fault based collision attacks on AES. In Luca Breveglieri, Israel Koren, David Naccache, and Jean-Pierre Seifert, editors, *Fault Diagnosis and Tolerance in Cryptography, Third International Workshop, FDTC 2006, Yokohama, Japan, October 10, 2006, Proceedings*, volume 4236 of *Lecture Notes in Computer Science*, pages 106–120. Springer, 2006.

[BLMR19] Christof Beierle, Gregor Leander, Amir Moradi, and Shahram Rasoolzadeh. CRAFT: lightweight tweakable block cipher with efficient protection against DFA attacks. *IACR Trans. Symmetric Cryptol.*, 2019(1):5–45, 2019.

[BR10] Andrey Bogdanov and Christian Rechberger. A 3-subset meet-in-the-middle attack: Cryptanalysis of the lightweight block cipher KTANTAN. In Alex Biryukov, Guang Gong, and Douglas R. Stinson, editors, *Selected Areas in Cryptography - 17th International Workshop, SAC 2010, Waterloo, Ontario, Canada, August 12-13, 2010, Revised Selected Papers*, volume 6544 of *Lecture Notes in Computer Science*, pages 229–240. Springer, 2010.

[BS97] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In Burton S. Kaliski Jr., editor, *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*, volume 1294 of *Lecture Notes in Computer Science*, pages 513–525. Springer, 1997.

[BS03] Johannes Blömer and Jean-Pierre Seifert. Fault based cryptanalysis of the advanced encryption standard (AES). In Rebecca N. Wright, editor, *Financial Cryptography, 7th International Conference, FC 2003, Guadeloupe, French West Indies, January 27-30, 2003, Revised Papers*, volume 2742 of *Lecture Notes in Computer Science*, pages 162–181. Springer, 2003.

[CCF$^+$21] Tingting Cui, Shiyao Chen, Kai Fu, Meiqin Wang, and Keting Jia. New automatic tool for finding impossible differentials and zero-correlation linear approximations. *Sci. China Inf. Sci.*, 64(2), 2021.

[CFGR10] Christophe Clavier, Benoit Feix, Georges Gagnerot, and Mylène Roussellet. Passive and active combined attacks on AES combining fault attacks and side channel analysis. In Luca Breveglieri, Marc Joye, Israel Koren, David Naccache, and Ingrid Verbauwhede, editors, *2010 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2010, Santa Barbara, California, USA, 21 August 2010*, pages 10–19. IEEE Computer Society, 2010.

[Cla07] Christophe Clavier. Secret external encodings do not prevent transient fault analysis. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 181–194. Springer, 2007.

[CT05] Hamid Choukri and Michael Tunstall. Round reduction using faults. In *Proceedings of the Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2005*, volume 5, pages 13–24, 2005.

[CY03] Chien-Ning Chen and Sung-Ming Yen. Differential fault analysis on AES key schedule and some coutnermeasures. In Reihaneh Safavi-Naini and Jennifer Seberry, editors, *Information Security and Privacy, 8th Australasian Conference, ACISP 2003, Wollongong, Australia, July 9-11, 2003, Proceedings*, volume 2727 of *Lecture Notes in Computer Science*, pages 118–129. Springer, 2003.

[CZS16] Wei Cheng, Yongbin Zhou, and Laurent Sauvage. Differential fault analysis on midori. In Kwok-Yan Lam, Chi-Hung Chi, and Sihan Qing, editors, *Information and Communications Security - 18th International Conference, ICICS 2016, Singapore, November 29 - December 2, 2016, Proceedings*, volume 9977 of *Lecture Notes in Computer Science*, pages 307–317. Springer, 2016.

[DEG$^+$18] Christoph Dobraunig, Maria Eichlseder, Hannes Groß, Stefan Mangard, Florian Mendel, and Robert Primas. Statistical ineffective fault attacks on masked AES with fault countermeasures. In Thomas Peyrin and Steven D. Galbraith, editors, *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part II*, volume 11273 of *Lecture Notes in Computer Science*, pages 315–342. Springer, 2018.

[DEK$^+$16] Christoph Dobraunig, Maria Eichlseder, Thomas Korak, Victor Lomné, and Florian Mendel. Statistical fault attacks on nonce-based authenticated encryption schemes. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 369–395, 2016.

[DEK$^+$18] Christoph Dobraunig, Maria Eichlseder, Thomas Korak, Stefan Mangard, Florian Mendel, and Robert Primas. SIFA: exploiting ineffective fault inductions on symmetric cryptography. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(3):547–572, 2018.

[DF16] Patrick Derbez and Pierre-Alain Fouque. Automatic search of meet-in-the-middle and impossible differential attacks. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 157–184. Springer, 2016.

[DFL11] Patrick Derbez, Pierre-Alain Fouque, and Delphine Leresteux. Meet-in-the-middle and impossible differential fault analysis on AES. In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, volume 6917 of *Lecture Notes in Computer Science*, pages 274–291. Springer, 2011.

[DH77] Whitfield Diffie and Martin E. Hellman. Special feature exhaustive cryptanalysis of the NBS data encryption standard. *Computer*, 10(6):74–84, 1977.

[DHS$^+$21] Xiaoyang Dong, Jialiang Hua, Siwei Sun, Zheng Li, Xiaoyun Wang, and Lei Hu. Meet-in-the-middle attacks revisited: Key-recovery, collision, and preimage attacks. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology -*

CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part III, volume 12827 of *Lecture Notes in Computer Science*, pages 278–308. Springer, 2021.

[DKM$^+$15] Christoph Dobraunig, François Koeune, Stefan Mangard, Florian Mendel, and François-Xavier Standaert. Towards fresh and hybrid re-keying schemes with beyond birthday security. In Naofumi Homma and Marcel Medwed, editors, *Smart Card Research and Advanced Applications - 14th International Conference, CARDIS 2015, Bochum, Germany, November 4-6, 2015. Revised Selected Papers*, volume 9514 of *Lecture Notes in Computer Science*, pages 225–241. Springer, 2015.

[DLV03] Pierre Dusart, Gilles Letourneux, and Olivier Vivolo. Differential fault analysis on A.E.S. In Jianying Zhou, Moti Yung, and Yongfei Han, editors, *Applied Cryptography and Network Security, First International Conference, ACNS 2003. Kunming, China, October 16-19, 2003, Proceedings*, volume 2846 of *Lecture Notes in Computer Science*, pages 293–306. Springer, 2003.

[DPdC$^+$15] Louis Dureuil, Marie-Laure Potet, Philippe de Choudens, Cécile Dumas, and Jessy Clédière. From code review to fault injection attacks: Filling the gap using fault model inference. In Naofumi Homma and Marcel Medwed, editors, *Smart Card Research and Advanced Applications - 14th International Conference, CARDIS 2015, Bochum, Germany, November 4-6, 2015. Revised Selected Papers*, volume 9514 of *Lecture Notes in Computer Science*, pages 107–124. Springer, 2015.

[DV12] François Dassance and Alexandre Venelli. Combined fault and side-channel attacks on the AES key schedule. In Guido Bertoni and Benedikt Gierlichs, editors, *2012 Workshop on Fault Diagnosis and Tolerance in Cryptography, Leuven, Belgium, September 9, 2012*, pages 63–71. IEEE Computer Society, 2012.

[FJLT13] Thomas Fuhr, Éliane Jaulmes, Victor Lomné, and Adrian Thillard. Fault attacks on AES with faulty ciphertexts only. In Wieland Fischer and Jörn-Marc Schmidt, editors, *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography, Los Alamitos, CA, USA, August 20, 2013*, pages 108–118. IEEE Computer Society, 2013.

[FT09] Toshinori Fukunaga and Junko Takahashi. Practical fault attack on a cryptographic LSI with ISO/IEC 18033-3 block ciphers. In Luca Breveglieri, Israel Koren, David Naccache, Elisabeth Oswald, and Jean-Pierre Seifert, editors, *Sixth International Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2009, Lausanne, Switzerland, 6 September 2009*, pages 84–92. IEEE Computer Society, 2009.

[Gir04] Christophe Giraud. DFA on AES. In Hans Dobbertin, Vincent Rijmen, and Aleksandra Sowa, editors, *Advanced Encryption Standard - AES, 4th International Conference, AES 2004, Bonn, Germany, May 10-12, 2004, Revised Selected and Invited Papers*, volume 3373 of *Lecture Notes in Computer Science*, pages 27–41. Springer, 2004.

[GKPM18] Aymeric Genêt, Matthias J. Kannwischer, Hervé Pelletier, and Andrew McLauchlan. Practical fault injection attacks on SPHINCS. *IACR Cryptol. ePrint Arch.*, page 674, 2018.

[GYS15] Nahid Farhady Ghalaty, Bilgiday Yuce, and Patrick Schaumont. Differential fault intensity analysis on PRESENT and LED block ciphers. In Stefan Mangard and Axel Y. Poschmann, editors, *Constructive Side-Channel Analysis and Secure Design - 6th International Workshop, COSADE 2015, Berlin, Germany, April 13-14, 2015. Revised Selected Papers*, volume 9064 of *Lecture Notes in Computer Science*, pages 174–188. Springer, 2015.

[Hem04] Ludger Hemme. A differential fault attack against early rounds of (triple-)des. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, volume 3156 of *Lecture Notes in Computer Science*, pages 254–267. Springer, 2004.

[HS13] Michael Hutter and Jörn-Marc Schmidt. The temperature side channel and heating fault attacks. In Aurélien Francillon and Pankaj Rohatgi, editors, *Smart Card Research and Advanced Applications - 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers*, volume 8419 of *Lecture Notes in Computer Science*, pages 219–235. Springer, 2013.

[JMR07] Marc Joye, Pascal Manet, and Jean-Baptiste Rigaud. Strengthening hardware AES implementations against fault attacks. *IET Inf. Secur.*, 1(3):106–110, 2007.

[JNP14] Jérémy Jean, Ivica Nikolic, and Thomas Peyrin. Tweaks and keys for block ciphers: The TWEAKEY framework. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II*, volume 8874 of *Lecture Notes in Computer Science*, pages 274–288. Springer, 2014.

[JSC$^+$14] Anju P. Johnson, Sayandeep Saha, Rajat Subhra Chakraborty, Debdeep Mukhopadhyay, and Sezer Gören. Fault attack on AES via hardware trojan insertion by dynamic partial reconfiguration of FPGA over ethernet. In Ting Yu and Shengqi Yang, editors, *Proceedings of the 9th Workshop on Embedded Systems Security, WESS '14, New Delhi, India, October 17, 2014*, pages 1:1–1:8. ACM, 2014.

[KAKS22] Anup Kumar Kundu, Aikata, Banashri Karmakar, and Dhiman Saha. Fault analysis of the PRINCE family of lightweight ciphers. *J. Cryptogr. Eng.*, 12(4):475–494, 2022.

[KLT15] Stefan Kölbl, Gregor Leander, and Tyge Tiessen. Observations on the SIMON block cipher family. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 161–185. Springer, 2015.

[LRT12] Victor Lomné, Thomas Roche, and Adrian Thillard. On the need of randomness in fault attack countermeasures - application to AES. In Guido Bertoni and Benedikt Gierlichs, editors, *2012 Workshop on Fault Diagnosis and Tolerance in Cryptography, Leuven, Belgium, September 9, 2012*, pages 85–94. IEEE Computer Society, 2012.

[LSG$^+$10] Yang Li, Kazuo Sakiyama, Shigeto Gomisawa, Toshinori Fukunaga, Junko Takahashi, and Kazuo Ohta. Fault sensitivity analysis. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*, volume 6225 of *Lecture Notes in Computer Science*, pages 320–334. Springer, 2010.

[MGV08] Nele Mentens, Benedikt Gierlichs, and Ingrid Verbauwhede. Power and fault analysis resistance in hardware through dynamic reconfiguration. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*, volume 5154 of *Lecture Notes in Computer Science*, pages 346–362. Springer, 2008.

[ML08] Paolo Maistri and Régis Leveugle. Double-data-rate computation as a countermeasure against fault analysis. *IEEE Trans. Computers*, 57(11):1528–1539, 2008.

[MPR$^+$11] Marcel Medwed, Christophe Petit, Francesco Regazzoni, Mathieu Renauld, and François-Xavier Standaert. Fresh re-keying II: securing multiple parties against side-channel and fault attacks. In Emmanuel Prouff, editor, *Smart Card Research and Advanced Applications - 10th IFIP WG 8.8/11.2 International Conference, CARDIS 2011, Leuven, Belgium, September 14-16, 2011, Revised Selected Papers*, volume 7079 of *Lecture Notes in Computer Science*, pages 115–132. Springer, 2011.

[MSGR10] Marcel Medwed, François-Xavier Standaert, Johann Großschädl, and Francesco Regazzoni. Fresh re-keying: Security against side-channel and fault attacks for low-cost devices. In Daniel J. Bernstein and Tanja Lange, editors, *Progress in Cryptology - AFRICACRYPT 2010, Third International Conference on Cryptology in Africa, Stellenbosch, South Africa, May 3-6, 2010. Proceedings*, volume 6055 of *Lecture Notes in Computer Science*, pages 279–296. Springer, 2010.

[MSS06] Amir Moradi, Mohammad T. Manzuri Shalmani, and Mahmoud Salmasizadeh. A generalized method of differential fault attack against AES cryptosystem. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, volume 4249 of *Lecture Notes in Computer Science*, pages 91–100. Springer, 2006.

[MSY06] Tal Malkin, François-Xavier Standaert, and Moti Yung. A comparative cost/security analysis of fault attack countermeasures. In Luca Breveglieri, Israel Koren, David Naccache, and Jean-Pierre Seifert, editors, *Fault Diagnosis and Tolerance in Cryptography, Third International Workshop, FDTC 2006, Yokohama, Japan, October 10, 2006, Proceedings*, volume 4236 of *Lecture Notes in Computer Science*, pages 159–172. Springer, 2006.

[Muk09] Debdeep Mukhopadhyay. An improved fault based attack of the advanced encryption standard. In Bart Preneel, editor, *Progress in Cryptology - AFRICACRYPT 2009, Second International Conference on Cryptology in Africa, Gammarth, Tunisia, June 21-25, 2009. Proceedings*, volume 5580 of *Lecture Notes in Computer Science*, pages 421–434. Springer, 2009.

[MWGP11] Nicky Mouha, Qingju Wang, Dawu Gu, and Bart Preneel. Differential and linear cryptanalysis using mixed-integer linear programming. In Chuankun Wu, Moti Yung, and Dongdai Lin, editors, *Information Security and Cryptology - 7th International Conference, Inscrypt 2011, Beijing, China, November 30 - December 3, 2011. Revised Selected Papers*, volume 7537 of *Lecture Notes in Computer Science*, pages 57–76. Springer, 2011.

[PAM19] Antoon Purnal, Victor Arribas, and Lauren De Meyer. Trade-offs in protecting keccak against combined side-channel and fault attacks. In Ilia Polian and Marc Stöttinger, editors, *Constructive Side-Channel Analysis and Secure Design - 10th International Workshop, COSADE 2019, Darmstadt, Germany, April 3-5, 2019, Proceedings*, volume 11421 of *Lecture Notes in Computer Science*, pages 285–302. Springer, 2019.

[PBMB17] Sikhar Patranabis, Jakub Breier, Debdeep Mukhopadhyay, and Shivam Bhasin. One plus one is more than two: A practical combination of power and fault analysis attacks on PRESENT and present-like block ciphers. In *2017 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2017, Taipei, Taiwan, September 25, 2017*, pages 25–32. IEEE Computer Society, 2017.

[PQ03] Gilles Piret and Jean-Jacques Quisquater. A differential fault attack technique against SPN structures, with application to the AES and KHAZAD. In Colin D. Walter, Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2003, 5th International Workshop, Cologne, Germany, September 8-10, 2003, Proceedings*, volume 2779 of *Lecture Notes in Computer Science*, pages 77–88. Springer, 2003.

[Riv09] Matthieu Rivain. Differential fault analysis on DES middle rounds. In Christophe Clavier and Kris Gaj, editors, *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*, volume 5747 of *Lecture Notes in Computer Science*, pages 457–469. Springer, 2009.

[RLK11] Thomas Roche, Victor Lomné, and Karim Khalfallah. Combined fault and side-channel attack on protected implementations of AES. In Emmanuel Prouff, editor, *Smart Card Research and Advanced Applications - 10th IFIP WG 8.8/11.2 International Conference, CARDIS 2011, Leuven, Belgium, September 14-16, 2011, Revised Selected Papers*, volume 7079 of *Lecture Notes in Computer Science*, pages 65–83. Springer, 2011.

[RSG21] Jan Richter-Brockmann, Pascal Sasdrich, and Tim Güneysu. Revisiting fault adversary models - hardware faults in theory and practice. *IACR Cryptol. ePrint Arch.*, page 296, 2021.

[RVB22] Hamed Ramzanipour, Navid Vafaei, and Nasour Bagheri. Practical differential fault analysis on craft, a lightweight block cipher. *The ISC International Journal of Information Security*, 14(3):21–31, 2022.

[SA09] Yu Sasaki and Kazumaro Aoki. Finding preimages in full MD5 faster than exhaustive search. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, volume 5479 of *Lecture Notes in Computer Science*, pages 134–152. Springer, 2009.

[SBD$^+$20] Thierry Simon, Lejla Batina, Joan Daemen, Vincent Grosso, Pedro Maat Costa Massolino, Kostas Papagiannopoulos, Francesco Regazzoni, and Niels Samwel.

Friet: An authenticated encryption scheme with built-in fault detection. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 581–611. Springer, 2020.

[SBR+20] Sayandeep Saha, Arnab Bag, Debapriya Basu Roy, Sikhar Patranabis, and Debdeep Mukhopadhyay. Fault template attacks on block ciphers exploiting fault propagation. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 612–643. Springer, 2020.

[SGL+17] Siwei Sun, David Gérault, Pascal Lafourcade, Qianqian Yang, Yosuke Todo, Kexin Qiao, and Lei Hu. Analysis of aes, skinny, and others with constraint programming. *IACR Trans. Symmetric Cryptol.*, 2017(1):281–306, 2017.

[SGSS14] Fabrizio De Santis, Oscar M. Guillen, Ermin Sakic, and Georg Sigl. Ciphertext-only fault attacks on PRESENT. In Thomas Eisenbarth and Erdinç Öztürk, editors, *Lightweight Cryptography for Security and Privacy - Third International Workshop, LightSec 2014, Istanbul, Turkey, September 1-2, 2014, Revised Selected Papers*, volume 8898 of *Lecture Notes in Computer Science*, pages 85–108. Springer, 2014.

[SHS16] Bodo Selmke, Johann Heyszl, and Georg Sigl. Attack on a DFA protected AES by simultaneous laser fault injections. In *2016 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2016, Santa Barbara, CA, USA, August 16, 2016*, pages 36–46. IEEE Computer Society, 2016.

[SHW+14] Siwei Sun, Lei Hu, Peng Wang, Kexin Qiao, Xiaoshuang Ma, and Ling Song. Automatic security evaluation and (related-key) differential characteristic search: Application to simon, present, lblock, DES(L) and other bit-oriented block ciphers. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 158–178. Springer, 2014.

[SMC21] Albert Spruyt, Alyssa Milburn, and Lukasz Chmielewski. Fault injection as an oscilloscope: Fault correlation analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(1):192–216, 2021.

[SS22] André Schrottenloher and Marc Stevens. Simplified MITM modeling for permutations: New (quantum) attacks. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part III*, volume 13509 of *Lecture Notes in Computer Science*, pages 717–747. Springer, 2022.

[ST17] Yu Sasaki and Yosuke Todo. New impossible differential search tool from design and cryptanalysis aspects - revealing structural properties of several ciphers. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference* *on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part III*, volume 10212 of *Lecture Notes in Computer Science*, pages 185–215, 2017.

[TBM14] Harshal Tupsamudre, Shikha Bisht, and Debdeep Mukhopadhyay. Differential fault analysis on the families of SIMON and SPECK ciphers. In Assia Tria and Dooho Choi, editors, *2014 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2014, Busan, South Korea, September 23, 2014*, pages 40–48. IEEE Computer Society, 2014.

[TFY07] Junko Takahashi, Toshinori Fukunaga, and Kimihiro Yamakoshi. DFA mechanism on the AES key schedule. In Luca Breveglieri, Shay Gueron, Israel Koren, David Naccache, and Jean-Pierre Seifert, editors, *Fourth International Workshop on Fault Diagnosis and Tolerance in Cryptography, 2007, FDTC 2007: Vienna, Austria, 10 September 2007*, pages 62–74. IEEE Computer Society, 2007.

[TIHM17] Yosuke Todo, Takanori Isobe, Yonglin Hao, and Willi Meier. Cube attacks on non-blackbox polynomials based on division property. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III*, volume 10403 of *Lecture Notes in Computer Science*, pages 250–279. Springer, 2017.

[TLG$^+$15] Shahin Tajik, Heiko Lohrke, Fatemeh Ganji, Jean-Pierre Seifert, and Christian Boit. Laser fault attack on physically unclonable functions. In Naofumi Homma and Victor Lomné, editors, *2015 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2015, Saint Malo, France, September 13, 2015*, pages 85–96. IEEE Computer Society, 2015.

[VBSM18] Navid Vafaei, Nasour Bagheri, Sayandeep Saha, and Debdeep Mukhopadhyay. Differential fault attack on SKINNY block cipher. In Anupam Chattopadhyay, Chester Rebeiro, and Yuval Yarom, editors, *Security, Privacy, and Applied Cryptography Engineering - 8th International Conference, SPACE 2018, Kanpur, India, December 15-19, 2018, Proceedings*, volume 11348 of *Lecture Notes in Computer Science*, pages 177–197. Springer, 2018.

[VSBM20] Navid Vafaei, Sayandeep Saha, Nasour Bagheri, and Debdeep Mukhopadhyay. Fault attack on SKINNY cipher. *J. Hardw. Syst. Secur.*, 4(4):277–296, 2020.

[XZBL16] Zejun Xiang, Wentao Zhang, Zhenzhen Bao, and Dongdai Lin. Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 648–678, 2016.

[YSW18] Bilgiday Yuce, Patrick Schaumont, and Marc Witteman. Fault attacks on secure embedded software: Threats, design, and evaluation. *J. Hardw. Syst. Secur.*, 2(2):111–130, 2018.

[ZLZ$^+$18] Fan Zhang, Xiaoxuan Lou, Xinjie Zhao, Shivam Bhasin, Wei He, Ruyi Ding, Samiya Qureshi, and Kui Ren. Persistent fault analysis on block ciphers. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(3):150–172, 2018.