The ISC Int'l Journal of Information Security

ISeCure

November 2022, Volume 14, Number 3 (pp. 21–31)

http://www.isecure-journal.org

PRESENTED AT THE ISCISC'2022 IN RASHT, IRAN.

Practical Differential Fault Analysis on CRAFT, a Lightweight Block Cipher [☆]

Hamed Ramzanipour [1], Navid Vafaei [1], and Nasour Bagheri [1,*]

[1] *Electrical Engineering Department, Shahid Rajaee Teacher Training University, Tehran, Iran.*

ARTICLE INFO.

Keywords:
Differential Fault Analysis, CRAFT, Implementation of Fault Attack, Glitch Frequency

Type:
Research Article

doi:
10.22042/isecure.2022.14.3.3

dor:
20.1001.1.20082045.2022.14.3.3.7

ABSTRACT

Differential fault analysis, a kind of active non-invasive attack, is an effective way of analyzing cryptographic primitives that have lately earned more attention. In this study, we apply this attack on CRAFT, a recently proposed lightweight tweakable block cipher, supported by simulation and experimental results. This cipher accepts a 64-bit Tweak, a 64-bit plaintext, and a 128-bit key to produce a 64-bit ciphertext. We assume that the target implementation of CRAFT does not use countermeasures in this paper. The considered fault model in the initial phase of this paper is a single-bit, but random nibble-injected fault, where we first present the fault injection attack as a simulation and then report on how to retrieve the round sub-keys. Next, we use frequency glitch as a fault injection technique in the experimental phase. This part aims to produce a single fault at a nibble in a specific round of the CRAFT. Following our statistical analysis and according to the simulation findings, we can reduce the key space to 30.28 and 24.37 bits, respectively, by using 4 and 5 faults. The experimental section also identifies the location of random faults injected by the hardware mechanism.

1 Introduction

One way of analyzing the security of a cryptographic system is to evaluate the implementation weaknesses [1]. Fault attacks are one approach to obtaining secret information by exploiting these weaknesses. This concept was first used by Boneh *et al.* against the RSA [2]. According to recent research, fault injection (FI) is one of the most efficient techniques for analyzing cryptographic systems. The FI attack is used against various cryptographic systems because it does not necessitate a high level of expertise or expensive analysis and equipment in some scenarios. In this approach, the attacker is assumed to have physical access to the target device containing the cryptographic system. While running the encryption process, s/he applies a deliberate manipulation in it and checks for variations in the presence of a fault. The variation may leak information about the secret part of the algorithm.

Power and clock glitches are common approaches in FI attacks. A glitch may cause a hardware fault by a sudden change in power or clock frequency. The glitch is an efficient, cost-effective, and widespread method

[*] Corresponding author.

[☆] The ISCISC'2022 program committee effort is highly acknowledged for reviewing this paper.

Email addresses: h.ramzanipour@sru.ac.ir, n.vafaei@sru.ac.ir, Nbagheri@sru.ac.ir

ISSN: 2008-2045 © 2022 ISC. All rights reserved.

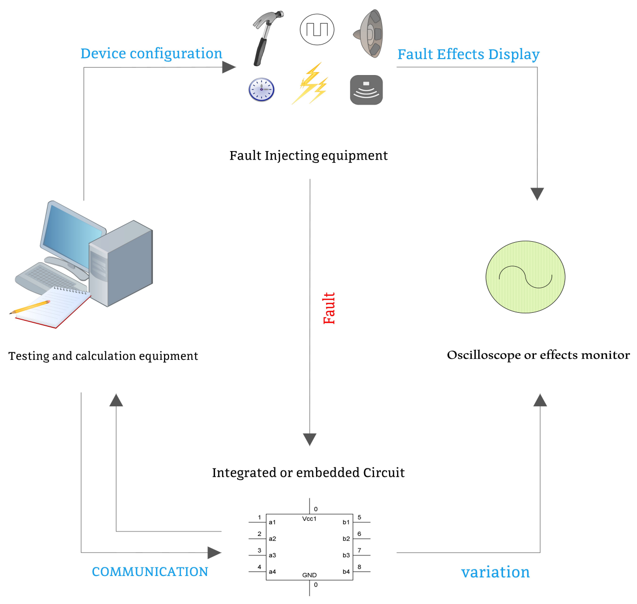

Device configuration
Fault Effects Display
Fault Injecting equipment
Fault
Testing and calculation equipment
Oscilloscope or effects monitor
Integrated or embedded Circuit
COMMUNICATION
variation

Figure 1. Fault analysis steps [4]

for FI attacks [3]. The FI is performed in a variety of ways, including electromagnetic field induction, laser radiation, etc. Figure 1 illustrates the steps of the fault attack. In this attack, a specific fault mechanism changes the victim circuit (a subset of embedded circuits). At the same time, the oscilloscope records the results of the modifications. The attacker evaluates the data using various analytical approaches and retrieves the secret parameters using test equipment. Fault attack has been extended to different methods for instance differential fault attack (DFA) [5], fault intensity attack (FIA) [6], statistical ineffective fault attack (SIFA) [7], statistical effective fault attack (SEFA) [8], persistent faults analysis (PFA) [9, 10] and statistical ineffective persistent faults analysis (SIPFA) [11]. However, one of the most frequently utilized cryptanalysis methods is DFA. This method, for the first time, was developed against symmetric cryptography by Biham and Shamir in 1997. The principle of DFA is combining faulty and non-faulty ciphertexts to reveal an intermediate state to retrieve the secret key. This method has been applied to a significant number of block cipher, like AES, SIMON, SHA-3, PRINCE, PRESENT, SKINNY [12–18]. CRAFT [19] is a tweakable block cipher that was established to prevent differential fault attacks. That is a 32 rounds block cipher with a 128-bit master key and 64-bit plaintext and Tweak. CRAFT contains characteristics including executing encryption and decryption in the same hardware area, as well as providing a fault detection mechanism with small overhead [20–22]. The comprehensive security of CRAFT is analyzed in [23]. Also, a single-tweak rectangle attack on 18 rounds of this cipher is proposed in [24].

1.1 Our Contribution

DFA is a strong tool for the attack on cryptographic algorithms. Although DFA requires fixing the Tweak in the tweakable block cipher, it is still a priceless tool for key recovery. In this work, we applied DFA against the unprotected version of CRAFT for the first time, to the best of our knowledge. We show that the unprotected CRAFT version is vulnerable to DFA, so the master key can be recovered with four faults. In addition, we validated our results with simulations. We also perform the practical fault attack on CRAFT on an AVR ATmega32A by using clock glitches. It is worth noting that using redundancies for detecting and correcting faults for the FI attack protection is considered for applications with no significant area limitations. The authors of CRAFT proposed a set of redundancies for the detection approach, all of which had area and execution time overheads and may not be suitable for area-constrained devices such as passive RFID tags. Therefore, security analysis of unprotected implementation is also required. It was the case for CRAFT's security analysis against other attacks. For instance, despite the fact that CRAFT was not designed to resist specific attacks such as SIFA and Related-key attacks [19], some researchers exploited these scenarios. For another instance, ElSheikh *et al.* [25] evaluated the CRAFT resistance to related-key attack in the *SPACE conference 2019*. As a consequence, by utilizing a simple key schedule of CRAFT and 2^{31} queries to the encryption oracle, they were able to retrieve the master key. Therefore, the security of cryptographic algorithms against various attacks is worth evaluating to determine its security boundaries accurately.

In the rest of this paper, we first introduce CRAFT and its design characteristics in Section 2. The statistical results and equations required for the analysis are discussed in Section 3, after a discussion of the differential fault attack and our strategy to apply this analytical technique against CRAFT. In Section 4, we show how a FI attack is performed in the hardware (CRAFT algorithm) Using low-cost equipment.

2 Specification of CRAFT

CRAFT is a 64-bit block cipher that uses a 128-bit master key and 64-bit Tweak. The master key was initially split into two 64-bit halves. CRAFT allocates 32 rounds to plaintext encryption. The 64-bit plaintext input creates the initial states X_1. The input State to i-th round, X_i, can be represented as a 4×4 matrix of nibbles as follows (brackets represent the nibble number):

Table 1. Round constants of CRAFT

Round i	$RC_i(a_i, b_i)$
0-15	11,84,42,25,96,c7,63,b1,54,a2,d5,e6,f7,73,31,14
16-31	82,45,26,97,c3,61,b4,52,a5,d6,e7,f3,71,34,12,85

$$X_i = \begin{bmatrix} X_i[0] & X_i[1] & X_i[2] & X_i[3] \\ X_i[4] & X_i[5] & X_i[6] & X_i[7] \\ X_i[8] & X_i[9] & X_i[10] & X_i[11] \\ X_i[12] & X_i[13] & X_i[14] & X_i[15] \end{bmatrix} \quad (1)$$

The encryption process of CRAFT, can be seen in Figure 2. Every round involve five operations: MixColumn (MC), RoundConstant (RC), Tweakeys (TK), PermuteNibbles (PN) and S-box (S).

$$R_i = MC \circ RC_i \circ TK_i \circ PN \circ S \quad (2)$$

Two operations, S-box and PermuteNibbles, were ignored in the last round (31-st round).

$$R_{31} = MC \circ RC_i \circ TK_i \quad (3)$$

The multiplication of 4×4 equations of the input (IM) and binary M (Equation 4) matrices generates the output of MC layer (Equation 5). Also, in the following matrices, j represents the column number.

$$M = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4)$$

$$\begin{bmatrix} X_i^j[0] \\ X_i^j[1] \\ X_i^j[2] \\ X_i^j[3] \end{bmatrix} = \begin{bmatrix} X_i^j[0] \oplus X_i^j[2] \oplus X_i^j[3] \\ X_i^j[1] \oplus X_i^j[3] \\ X_i^j[2] \\ X_i^j[3] \end{bmatrix} \quad (5)$$

In the RoundConstant sub-function, a four-bit $a_i^r = (a^3, a^2, a^1, a^0)$ and three-bit $b_i^r = (b^2, b^1, b^0)$ LFSR are used. These constant values are updated separately for each round, then XORed with $X_i[4]$ and $X_i[5]$, respectively. Table 1 shows the numbers.

By combining the two specified keys (k_0,k_1) with the Tweak, the four initial values Tweakeys TK_0, TK_1, TK_2, and TK_3 are produced. Each of these values is a separate 4×4 block that is used to add Tweakeys in different rounds.

$$TK_0 = k_0 \oplus T, TK_1 = k_1 \oplus T \quad (6)$$

Vector Q (Equation 8) is used to perform permutation for generating TK_2 and TK_3 in accordance to Equation 7.

$$TK_2 = k_0 \oplus Q(T), TK_3 = k_1 \oplus Q(T) \quad (7)$$

Table 2. Permutation of CRAFT

ρ	15	12	13	14	10	9	8	11	6	5	4	7	1	2	3	0

Table 3. S-box of CRAFT

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

Table 4. Notations

Symbol	Notation
$\mathcal{X}^*$	The faulty value of $\mathcal{X}$
Δ	Difference, e.g. $\Delta\mathcal{X} = \mathcal{X} \oplus \mathcal{X}^*$
M	A 4×4 binary matrix
MC	MixColumn sub-function
RC	RoundConstant sub-function
TK	Add round Tweakeys
PN	PermuteNibbles sub-function
$S(\cdot)$	Substitution box
$S^{-1}(\cdot)$	Reverse of $S(\cdot)$
ρ	Permutation mapping
Q	Permutation mapping of subkeys
X_i	Initial state at the beginning of the i-th round
Y_i	Data produced after RC_i
W_i	Data produced after TK_i
Z_i	Data produced after PN_i
k_0,k_1	Initial keys

$$Q = [12, 10, 15, 5, 14, 8, 9, 2, 11, 3, 7, 4, 6, 0, 1, 13] \quad (8)$$

Therefore, $TK_{i \bmod 4}$ is XORed with the intermediate values in round i (TK sequence is repeated every four rounds).

Next, the permutation table alters the placement of the nibbles (Table 2). And, at the end of each round, S-box converts every 4 bits of the inputs to a hexadecimal value at the outputs, according to Table 3.

Table 4 illustrates all of the symbols that are required in this paper.

3 Differential Fault Analysis of CRAFT

DFA is based on injecting a fault into the cryptographic algorithm and exploiting the differential characteristic using correct and faulty ciphertexts. In this section, we first explain the theory of the attack on the last round of CRAFT. Then the simulation results of the fault attack on CRAFT are presented.

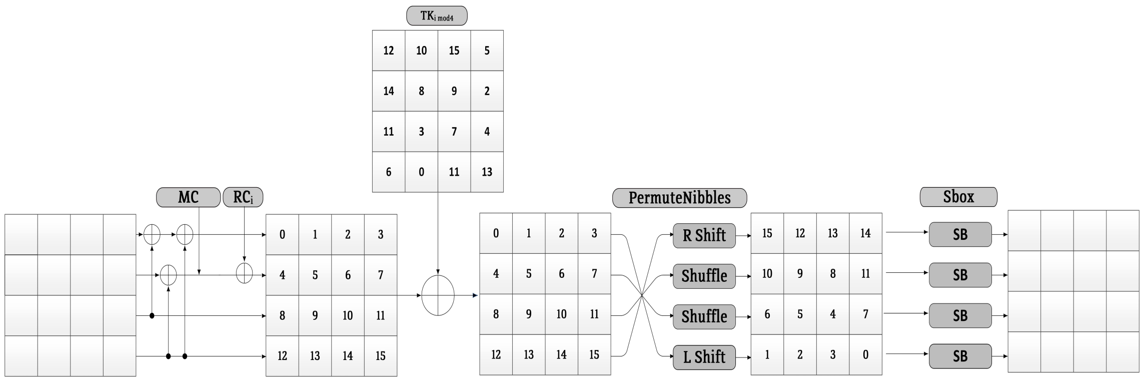


Figure 2. The round structure of CRAFT

3.1 DFA on CRAFT

Assume that a fault is injected in the first nibble at the beginning of the 27th round. The fault is propagated to the last rounds as illustrated in Figure 3. By injecting a fault into the internal state of CRAFT, the differential of correct and faulty ciphertexts are changed by passing through the S-box while remaining fixed during the linear sub-functions such as Tweakeys, PermuteNibbles, RoundConstant, and MixColumn. A differential nibble of the correct and faulty ciphertexts could be a non-zero, zero, or any possible difference. Non-zero differences, zero-differences, and any possible differences are represented by yellow, red, and white cells, respectively, which are depicted in Figure 3.

Suppose the difference between the correct and faulty value in the intermediate states with probability 1 is non-equal to zero. In that case, this concept is denoted as a non-zero difference. If we cannot guess the difference between the mentioned values, the difference in each subsection will be unpredictable, which is called any possible difference. Also, the zero differences are related to the values in which the FI has no effect [18, 24].

Since the S-box is a nonlinear sub-function, we are interested in exploiting differential values of the input and output of the S-box to recover the key. By injecting a fault in the first nibble at the beginning of the 27th round, a set of linear equations can be obtained from the differential input and output of the S-box in the last round, which is illustrated in Table 5. One of the equations is utilized for key recovery in the first evaluation, with the following assumptions:

(1) Plaintext is random, Tweak and (k_0, k_1) are fixed.
(2) The fault model is considered as a bit flip

All non-zero differences are related to a specific position of FI at the beginning of the 27th round, so we can use the output differential patterns to pinpoint the exact location of the fault. According to Figure 3, in the last round, three nibbles 9, 10, and 11 with non-zero differences can be obtained. Now, we are looking for the last round key (TK_3), and we will use the linear relations which are given in Table 5.

Table 5. Linear equations, by the FI into nibble 0 at 27th Round of the CRAFT

$\Delta Y_{28}[3] = \Delta Y_{28}[7] = \Delta Y_{28}[15]$
$\Delta Y_{29}[2] = \Delta Y_{29}[6] = \Delta Y_{29}[14]$
$\Delta Y_{29}[3] = \Delta Y_{29}[11]$
$\Delta Y_{30}[0] = \Delta Y_{30}[8]$
$\Delta Y_{30}[1] = \Delta Y_{30}[5] = \Delta Y_{30}[13]$
$\Delta Y_{30}[2] = \Delta Y_{30}[6] = \Delta Y_{30}[14]$
$\Delta Y_{31}[1] = \Delta Y_{31}[13] \oplus \Delta Y_{31}[9]$
$\Delta Y_{31}[5] = \Delta Y_{31}[13]$
$\Delta Y_{31}[7] = \Delta Y_{31}[15]$
$\Delta Y_{31}[4] = \Delta Y_{31}[12]$

As was mentioned before, the non-zero differences on both sides of the S-box are utilized to recover the last round key. Accordingly, one of the linear equations before the input of the S-box is Equation 9:

$$\Delta Y_{31}[4] = \Delta Y_{31}[12] \tag{9}$$

Due to the presence of a linear sub-function before the S-box, the following relationships are valid:

$$\begin{aligned} \Delta W_{31}[4] &= \Delta Z_{31}[10] \\ \Delta W_{31}[12] &= \Delta Z_{31}[1] \end{aligned} \tag{10}$$

Also, the following equation shows the difference between correct and faulty values in the output of the PermuteNibbles:

$$Z_{31}[10] \oplus Z^*_{31}[10] = Z_{31}[1] \oplus Z^*_{31}[1] \tag{11}$$

According to Figure 4, to recover the TK_3, the following equations should be verified:

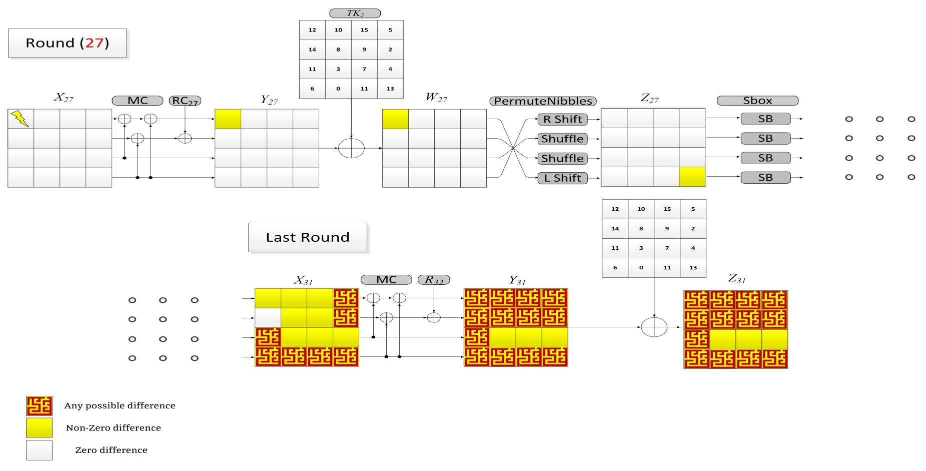


Figure 3. Fault propagation in CRAFT (assume the round counter begins from 0)

$$\begin{aligned} X_{32}[1] &= S(Z_{31}[1]) \Rightarrow Z_{31}[1] = S^{-1}(X_{32}[1]) \\ X_{32}[10] &= S(Z_{31}[10]) \Rightarrow Z_{31}[10] = S^{-1}(X_{32}[10]) \end{aligned} \quad (12)$$

$$\begin{aligned} X^*_{32}[1] &= S(Z^*_{31}[1]) \Rightarrow Z^*_{31}[1] = S^{-1}(X^*_{32}[1]) \\ X^*_{32}[10] &= S(Z^*_{31}[10]) \Rightarrow Z^*_{31}[10] = S^{-1}(X^*_{32}[10]) \end{aligned} \quad (13)$$

If we track the path of the fault propagation from ciphertexts to the output of the S-box (to backward) we would have the following equations:

$$\Delta X_{32}[1] = \Delta Y_{32}[1] \oplus \Delta Y_{32}[9] \oplus \Delta Y_{32}[13] \quad (14)$$

$$\Delta X_{32}[10] = \Delta Y_{32}[10] \quad (15)$$

$$\begin{aligned} &X_{32}[1] \oplus X^*_{32}[1] = \\ &Y_{32}[1] \oplus Y^*_{32}[1] \oplus Y_{32}[9] \oplus Y^*_{32}[9] \oplus Y_{32}[13] \oplus Y^*_{32}[13] \end{aligned} \quad (16)$$

$$X_{32}[10] \oplus X^*_{32}[10] = Y_{32}[10] \oplus Y^*_{32}[10] \quad (17)$$

To achieve $X_{32}[1]$ and $X_{32}[10]$, we need to guess four nibbles: $TK_3[1]$, $TK_3[9]$, $TK_3[13]$, and $TK_3[10]$. Now we need correct and faulty values of ciphertexts.

$$\begin{aligned} Z_{32}[1] &= Y_{32}[1] \oplus TK_3[1] \\ Z_{32}[9] &= Y_{32}[9] \oplus TK_3[9] \\ Z_{32}[13] &= Y_{32}[13] \oplus TK_3[13] \\ Z_{32}[10] &= Y_{32}[10] \oplus TK_3[10] \end{aligned} \quad (18)$$

$$\begin{aligned} Z^*_{32}[1] &= Y^*_{32}[1] \oplus TK_3[1] \\ Z^*_{32}[9] &= Y^*_{32}[9] \oplus TK_3[9] \\ Z^*_{32}[13] &= Y^*_{32}[13] \oplus TK_3[13] \\ Z^*_{32}[10] &= Y^*_{32}[10] \oplus TK_3[10] \end{aligned} \quad (19)$$

By using former equations, we construct new connections between the differential values at the input and output of the S-box at the last round:

$$\begin{aligned} &\Delta Z_{31}[10] = \Delta Z_{31}[1] \\ &Z_{31}[10] \oplus Z^*_{31}[10] = Z_{31}[1] \oplus Z^*_{31}[1] \\ &S^{-1}(X_{32}[10]) \oplus S^{-1}(X^*_{32}[10]) = S^{-1}(X_{32}[1]) \oplus S^{-1}(X^*_{32}[1]) \end{aligned} \quad (20)$$

$$\begin{aligned} &S^{-1}(Y_{32}[10]) \oplus S^{-1}(Y^*_{32}[10]) = \\ &S^{-1}(Y_{32}[1] \oplus Y_{32}[9] \oplus Y_{32}[13]) \oplus S^{-1}(Y^*_{32}[1] \oplus Y^*_{32}[9] \oplus Y^*_{32}[9]) \end{aligned} \quad (21)$$

Eventually, we guess the subkeys of the last round, in the following equation:

$$\begin{aligned} &S^{-1}(Z_{32}[10] \oplus TK_3[10]) \oplus S^{-1}(Z^*_{32}[10] \oplus TK_3[10]) = \\ &S^{-1}(Z_{32}[1] \oplus TK_3[1] \oplus Z_{32}[9] \oplus TK_3[9] \oplus Z_{32}[13] \oplus TK_3[13]) \\ &S^{-1}(Z^*_{32}[1] \oplus TK_3[1] \oplus Z^*_{32}[9] \oplus TK_3[9] \oplus Z^*_{32}[13] \oplus TK_3[13]) \end{aligned} \quad (22)$$

As can be resulted in Equation 22, we can guess 4 nibbles of subkey TK_3. Since each nibble has 2^4 possible states, there are 2^{16} candidates to retrieve these four nibbles of the last round key. Only those candidates are picked that yield the correct master key.

After evaluation of all candidates in Equation 22 equation, a limited number of 2^{16} possible candidates are reported as the surviving key candidates. The correct candidates are the terms used to describe these survival keys. Nibbles 1,9,10, and 13 of TK_3 subkey should be in the correct candidates With a probability of 1. The step-by-step instructions for recovering the last round key of CRAFT are set out in Algorithm 1.

3.2 Simulation Results of DFA on CRAFT

After describing DFA on CRAFT in the previous part, we now extend the attack for simulation. As mentioned before, the required nibbles of the TK_3 in the validation of the Equation 22 can be retrieved by using Algorithm 1. According to Table 5, other remained nibbles can also be recovered by using this algorithm due to FI into nibble 0 at the beginning of the 27th round.

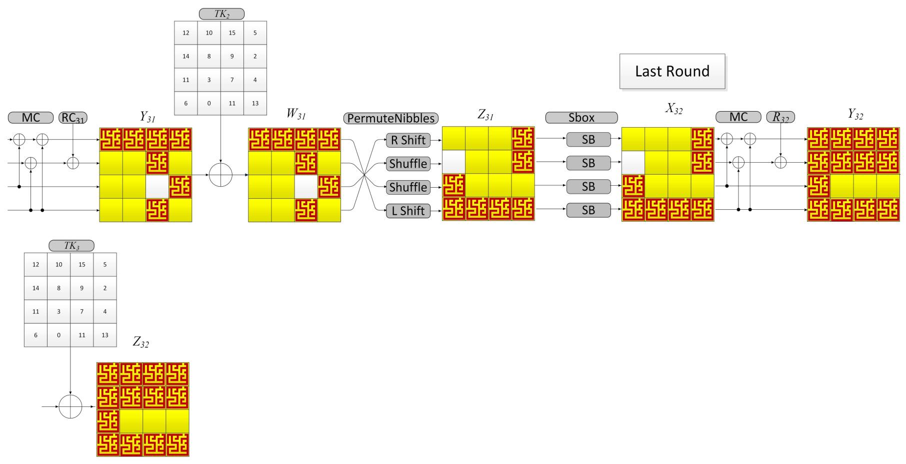


Figure 4. DFA on last Round of CRAFT (assume the round counter begins from 1)

Algorithm 1 DFA on Last Round of CRAFT

Require: P: Plaintext; TK: Tweak; K: Master Key; C: Correct Ciphertext; C^*: Faulty Ciphertext; Enc: Encryption; CC: Correct Candidates; $Tk_3[1]$: "1"; $Tk_3[9]$: "9"; $Tk_3[10]$: "10"; $Tk_3[13]$: "13".

Ensure: candidates for CC. (initialization)

```
 1: $Nibble \leftarrow 0$                              ▷ location of fault
 2: $f \leftarrow 0x1$                                 ▷ Value of fault
 3: $TK \leftarrow Fixed$
 4: $k_0, k_1 \leftarrow Fixed$
 5: while $True$ do
 6:     $P \xleftarrow{\$} \{0,1\}^{64}$
 7:     Correct $Enc(P) = C$
 8:     Faulty $Enc(P) = C^*$
 9:     if $round == 27$ then                         ▷ injection Fault
10:         $Nibble = Nibble \oplus f$
11:     end if
12:     $Z_{32} \leftarrow C$ 18
13:     $Z^*_{32} \leftarrow C^*$ 19
14: check 1:
15:     if 22 $== True$ then :
16:         save $(CC)$
17:     end if
18: check 2:
19:     if "1","9","10","13" $\in CC$ then
20:         print$(CC)$
21:     end if
22: end while
```

As another example, by considering the equation $\Delta Y_{31}[7] = \Delta Y_{31}[15]$, the extended equation will be Equation 23.

$$S^{-1}(Z_{32}[11] \oplus TK_3[11]) \oplus S^{-1}(Z^*_{32}[11] \oplus TK_3[11]) =$$
$$S^{-1}(Z_{32}[0] \oplus TK_3[0] \oplus Z_{32}[8] \oplus TK_3[8] \oplus Z_{32}[12] \oplus TK_3[12])$$
$$S^{-1}(Z^*_{32}[0] \oplus TK_3[0] \oplus Z^*_{32}[8] \oplus TK_3[8] \oplus Z^*_{32}[12] \oplus TK_3[12]) \tag{23}$$

As a result of the recent equation, nibbles 0, 8, 11, and 12 of TK_3 can be retrieved. Hence, By using other non-zero differences in Table 5, more information about TK_3 can be achieved. So far, we only used two linear equations (Equation 22,Equation 23) to extract the eight nibbles from the TK_3, while we can use the same strategy to recover more of the subkeys and reduce the computable space for the master key.

Table 6. By injecting fault to different nibbles of the 27th round CRAFT (single fault), the last round keys (TK_3) can be retrieved

Injection at	TK_3 nibbles that can be retrieved
nibble[0]	[0],[1],[2],[5],[8],[9],[10],[11],[12],[13],[14]
nibble[1]	[1],[2],[3],[6],[8],[9],[10],[11],[13],[14],[15]
nibble[2]	[0],[2],[3],[7],[8],[9],[10],[11],[12],[14],[15]
nibble[3]	[0],[1],[3],[4],[8],[9],[10],[11],[12],[13],[15]
nibble[4]	[0],[1],[4],[8],[9],[11],[12],[13],[14],[15]
nibble[5]	[2],[3],[7],[8],[9],[10],[11],[12],[13],[14],[15]
nibble[6]	[2],[3],[6],[9],[10],[11],[12],[13],[14],[15]
nibble[7]	[0],[1],[5],[7],[8],[9],[10],[11],[12],[13],[14],[15]
nibble[8]	[0],[1],[2],[5],[8],[9],[10],[11],[12],[13],[14]
nibble[9]	[1],[2],[3],[6],[8],[9],[10],[11],[13],[14],[15]
nibble[10]	[0],[2],[3],[7],[8],[9],[10],[11],[12],[14],[15]
nibble[11]	[0],[1],[3],[4],[8],[9],[10],[11],[12],[13],[15]
nibble[12]	[0],[8],[11],[12]
nibble[13]	[1],[3],[8],[9],[10],[11],[13],[15]
nibble[14]	[2],[9],[10],[14]
nibble[15]	[1],[9],[10],[13]

Since the location of the fault may change during every FI trial, different non-zero differences will be obtained for all of the nibbles in Table 6, which will be comparable to Table 5 information. Thus, at the beginning of the 27th round, more TK_3 nibbles can be retrieved depending on FI in various nibbles.

On average, 37.75 bits of the master key may be recovered if injecting a single fault during the 27th round of CRAFT. The Equation 24 shows this computation (according to Table 6).

$$((44 \times 9) + (40 \times 2) + 48 + (16 \times 3) + 32) \div 16 = 37.75 \quad (24)$$

We focus on the last results in Table 7. The last equation illustrates that by injecting a nibble fault into a random location of the 27th round, 37.75 bits of the last round key will be retrieved. Hence, it is a remaining computation at the order of $2^{26.25}$. For a personal computer, this is a feasible time complexity. Therefore, the key of the last round is retrieved at an acceptable time. Also, we can recover more bits of the master key by injecting the fault in separate nibbles of the 27th round (without iterating).

In Table 7, the symbols N, k_r, and k_{un}, respectively, refer to the number of faults that have been injected into the separate nibbles, the average number of bits that can be recovered from the last round, and the remaining bits of the master key (128 bits). Hence, the t_{k_r} is the time required to recover k_r bits, and $t_{k_{un}}$ is the remaining time to recover the master key. The $t_{k_{un}}$ exactly equal to the required time to do $2^{k_{un}}$ calls to CRAFT. The overall time complexity is represented in the paper by T_{total} (time to recover the entire master key) and obtained from the following equation:

$$T_{total} = t_{k_r} + t_{k_{un}} \quad (25)$$

We have been attacking TK_3 nibbles so far. Although the last round key is nearly recovered, there is still considerable time complexity to recover the master key (Table 7). To fully recover the master key, we need to consider TK_2 in our analysis.

If we use an exhaustive search to guess the remaining bits of the Tk_3 (for example, 26.25 from 64 bits), we can decrypt the last round and return to the penultimate round, then the Tk_2 can be retrieved by using the penultimate round linear equations and represent the simulation results on the average for it.

3.3 Recovering the Master Key with Minimum Time Complexity

The final step involve evaluating Tk_2 and TK_3. If we repeat the FI twice, 37.75 bits are recovered from TK_3 for the first time. Then, we guess the remaining bits that cost $2^{26.25}$. For a personal computer, this is an acceptable time complexity. The complete recovery of the last round key leads us to analyze the Tk_2 and recover 37.75 bits from the penultimate round key by the second fault injection. Hence, on average, 75.5 bits are obtained from the master key by two faults. Now, we can almost retrieve the large portion space of the master key by injecting two faults into the 27th round, as shown in Table 8.

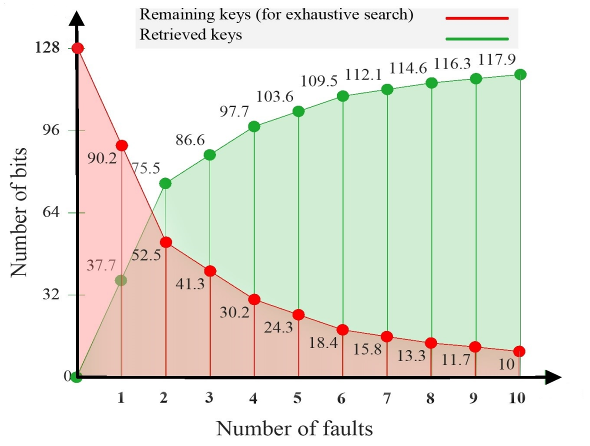


Figure 5. The number of bits that can be retrieved from the master key based on the number of injected faults (the number of keys that can be recovered is indicated with Green, while another number of keys that can not be recovered is indicated in Red (according to Table 8))

In conclusion, after four faults, on average, 30.28 bits of the master key remain, which may be estimated using an exhaustive search with little computational complexity and time. We utilized a processor with Corei3 @3.4GHz 4GB RAM for our computations. Figure 5 depicts the relation between increasing the number of faults and the number of recoverable bits of the master key.

The steps involved and outcomes of the above statistical simulations may be accessed at the following address:

`https://github.com/Hamed-Ramzanipour/hardwaresoftwaredfacraft/tree/main/Software`

4 Hardware Implementation of DFA

In this section, we aim to express the results of the practical differential fault attack by hardware implementation of the CRAFT and the fault mechanism.

4.1 Fault Injecting Structure

In the FI phase, the glitch frequency is used to disrupt the encryption and obtain the faulty ciphertexts. We employed two external clock sources to inject faults in this model.

Assume two separate circuits such that the first one (victim) provides the CRAFT hardware implementation and the second (glitch) operates as the FI mechanism at the same time. To inject a fault, the second circuit makes a sudden change in the external clock, which may disrupt the operating clock of the encryption circuit. Each sub-function of the encryption may not operate efficiently with a direct external

Table 7. The average number of bits retrieved from various single nibble FI (CRAFT encryption takes 0.0468 seconds)

N	1	2	3	4	5	6	7	8
$k_r(bit)$	37.75	48.86	54.77	57.34	58.96	60.10	60.95	61.60
$k_{un}(bit)$	90.25	79.14	73.23	70.66	69.04	67.9	67.05	66.40
$t_{k_r}(s)$	0.270	0.367	0.442	0.537	2.28	30.43	720.44	21600.4
$t_{k_{un}}(s)$	$2^{90.25} \times t_C$	$2^{79.14} \times t_C$	$2^{73.23} \times t_C$	$2^{70.66} \times t_C$	$2^{69.04} \times t_C$	$2^{67.9} \times t_C$	$2^{67.05} \times t_C$	$2^{66.40} \times t_C$
$T_{total}(s)$	$t_{k_r}+68 \times 10^{21}$	$t_{k_r}+31 \times 10^{18}$	$t_{k_r}+5 \times 10^{17}$	$t_{k_r}+8 \times 10^{16}$	$t_{k_r}+2 \times 10^{16}$	$t_{k_r}+1.2 \times 10^{16}$	$t_{k_r}+7 \times 10^{15}$	$t_{k_r}+7 \times 10^{15}$

Table 8. Analysis of the master key space (k_0, k_1 are 128-bits, and all parameters are associated with the last two rounds, Tk_2 and Tk_3)

N	2	3	4	5	6	7	8
$k_r(bit)$	75.5	86.61	97.72	103.63	109.54	112.11	114.68
$k_{un}(bit)$	52.5	41.39	30.28	24.37	18.46	15.89	13.32
$t_{k_r}(s)$	0.540	0.646	0.752	0.798	0.844	0.959	1.074
$t_{k_{un}}(s)$	$2^{52.5} \times t_C$	$2^{41.39} \times t_C$	$2^{30.28} \times t_C$	$2^{24.37} \times t_C$	$2^{18.46} \times t_C$	$2^{15.89} \times t_C$	$2^{13.32} \times t_C$
$T_{total}(s)$	$t_{k_r}+2 \times 10^{14}$	$t_{k_r}+1.3 \times 10^{11}$	$t_{k_r}+6.1 \times 10^{7}$	$t_{k_r}+1 \times 10^{6}$	$t_{k_r}+16 \times 10^{3}$	$t_{k_r}+2.8 \times 10^{3}$	479.66

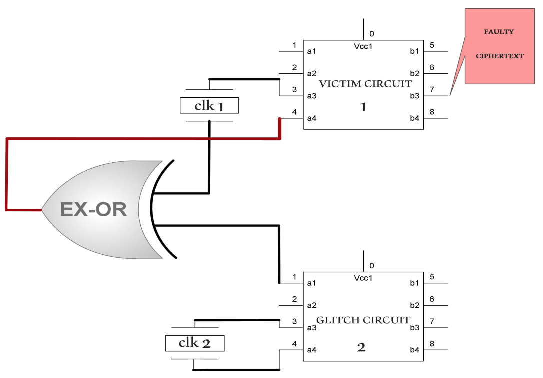


Figure 6. The Fault injecting structure

impact on the clock cycle, and provide us the non-zero differences, so we can do differential analysis.

The glitch is produced by connecting the fault trigger signal to the external clock of the victim circuit. Hence, one of the inputs of the external clock victim circuit is connected to the output of an XOR of the trigger signal and one of the crystal oscillator pins (belonging to the victim circuit). Figure 6 depicts this description.

To be more precise in injecting the fault, the operating clock frequency of circuit (2) is several times higher than the operating clock frequency of circuit (1). So, the glitch circuit has a higher speed frequency than the victim circuit, which may make a transient change during the execution of CRAFT. The appropriate bandwidth for the crystal oscillator of the fault circuit is determined by different criteria, such as the victim circuit frequency, noise, and fault accuracy.

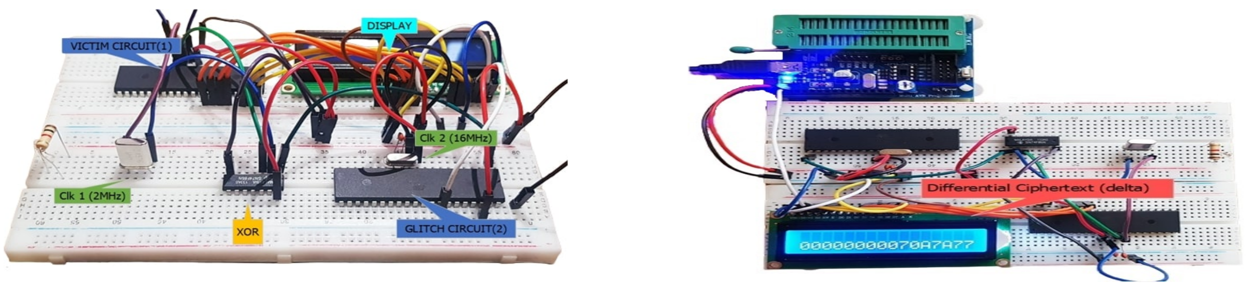
4.2 Experimental Results

We used AVR Microcontroller ATmega32A for experimental verification. Figure 7 depicts our hardware implementation of the FI mechanism. Two external crystal oscillators of 2 MHz and 16 MHz are used for the victim and glitch circuits, respectively, in the frequency sources.

In the previous subsection, the FI mechanism against the victim circuit was presented. Figure 7a depicts the CRAFT circuit components and also the glitch circuit. Trigger fault pulses are continually produced by the circuit (2). According to the FI mechanism, this glitch may disrupt the performance of any sub-function or skip part of it. Hence, the faulty outputs of the circuit (1) can be obtained, and we utilize these outputs to evaluate hardware implementation [26].

According to implementation studies [27, 28], faults that happened in the initial rounds had a higher impact on the resulting ciphertexts, which generated more than zero difference in the outputs. Hence We are looking for a significant zero difference between faulty and non-faulty ciphertexts (at least 8 nibbles out of 16), which may indicate that the fault is injected in the last rounds with high probability. For example, Figure 7b) shows a 40 bits zero-differences between the faulty and non-faulty ciphertext, which indicates that the fault occurred in one of the last rounds.

To identify the round of the injected fault, we employed a set of differential vectors generated by our implementation to evaluate the correct and faulty



(a) Hardware setting (b) The FI attack results

Figure 7. Implementation differential fault attack on hardware

Table 9. The number of FI by hardware implementations of fault circuit into the different rounds of CRFAT (assume the round counter begins from 0)

The *i*th round	The number of faults	Location of sub-function
28	3	Mixcolumn
29	4	Mixcolumn
30	18	Mixcolumn
31	76	Mixcolumn

ciphertexts in decryption mode by the correct key in the offline phase. The most probable round in the offline phase is regarded as a target round in the online phase, and other rounds could be considered noise.

So, after running the circuits for 10 minutes, we got around 200 different vectors from 1200 random faults, indicating that we should exclude 1000 injections due to AVR crashes. We store the number of valid output differential vectors during that time. The number of FI in each round is shown in Table 9. In the Table 9, 101 of 200 differential vectors had more than or equal to 8 zero differences. The offline verification in decryption mode confirmed that all faults are injected at the beginning of each round. As a result of using 1200 faults, at least 101 faults are injected at the beginning of different rounds in this experiment for two reasons: first, the higher speed of the fault circuit frequency, which, even with no control between glitch and the victim circuit executing the FI before the start of each round, and second, our assumption of selecting differential vectors with at least 8 zero differences.

According to results of Section 3.2, if 4 or 5 effective (beginning of the 27th round) faults are injected into CRAFT, the master key may be retrieved using this approach. However, by injecting a fault at the 31st round also key can be recovered by more injections.

The main goal of this part was to implement a fault circuit using low-cost equipment. We discovered the location of the faults during the analysis phase by combining hardware results and statistical analysis. Using a precise fault to modify only one bit (bit-flip) of a vector in the intermediate rounds of each algorithm requires more cost and time to boost the efficiency of DFA.

The hardware programming CRAFT and the FI circuit with the stated results are available at the following address:

`https://github.com/Hamed-Ramzanipour/hardwaresoftwaredfacraft/tree/main/Hardware`

5 Conclusion

CRAFT is a lightweight block cipher that has efficient features in encryption design. In this paper, we performed a differential fault attack against this algorithm, first in simulation and subsequently in hardware implementation. Our simulation showed that the recovery of the main key complexity could be reduced to 30.28 and 24.37 bits using 4 and 5 faults, respectively, which leaves little complexity for an exhaustive search. Experimental implementation of the FI attack was performed on CRAFT using low-cost equipment and finding the location of some of the single faults applied.

Although we were able to apply DFA to CRAFT effectively, there are still several topics that need to be investigated further. To begin with, the experiment is built on a very basic and almost uncontrollable circuit that needs several FI to create the required faults. It is proposed that a master-slave structure be used to shorten this time and improve FI accuracy. CRAFT's designers also suggested a fault detection mechanism to prevent differential fault attacks. However, we did not use it in our study. Hence, the evaluation of CRAFT security against DFA in the presence of countermeasures might be the second focus of future research.

References

[1] Raphael Spreitzer, Veelasha Moonsamy, Thomas Korak, and Stefan Mangard. Systematic Classi-

fication of Side-Channel Attacks: A Case Study for Mobile Devices. *IEEE Communications Surveys & Tutorials*, 20(1):465–488, 2017.

[2] Dan Boneh, Richard A DeMillo, and Richard J Lipton. On the Importance of Checking Cryptographic Protocols for Faults. In *International conference on the theory and applications of cryptographic techniques*, pages 37–51. Springer, 1997.

[3] Sho Endo, Takeshi Sugawara, Naofumi Homma, Takafumi Aoki, and Akashi Satoh. An on-chip glitchy-clock generator for testing fault injection attacks. *Journal of Cryptographic Engineering*, 1(4):265, 2011.

[4] Anubhab Baksi, Shivam Bhasin, Jakub Breier, Dirmanto Jap, and Dhiman Saha. Fault Attacks In Symmetric Key Cryptosystems. *IACR Cryptol. ePrint Arch.*, 2020:1267, 2020.

[5] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In Burton S. Kaliski Jr., editor, *Advances in Cryptology - CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 513–525. Springer, 1997.

[6] Nahid Farhady Ghalaty, Bilgiday Yuce, Mostafa Taha, and Patrick Schaumont. Differential Fault Intensity Analysis. In *2014 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pages 49–58. IEEE, 2014.

[7] 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.

[8] Navid Vafaei, Sara Zarei, Nasour Bagheri, Maria Eichlseder, Robert Primas, and Hadi Soleimany. Statistical Effective Fault Fttacks: The other Side of the Coin. *IEEE Transactions on Information Forensics and Security*, 2022.

[9] 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.

[10] Hadi Soleimany, Nasour Bagheri, Hosein Hadipour, Prasanna Ravi, Shivam Bhasin, and Sara Mansouri. Practical multiple persistent faults analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(1):367–390, 2022.

[11] Nasour Bagheri, Sadegh Sadeghi, Prasanna Ravi, Shivam Bhasin, and Hadi Soleimany. SIPFA: statistical ineffective persistent faults analysis on feistel ciphers. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(3):367–390, 2022.

[12] Sk Subidh Ali and Debdeep Mukhopadhyay. A Differential Fault Analysis on AES Key Schedule Using Single Fault. In *2011 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pages 35–42. IEEE, 2011.

[13] Junko Takahashi and Toshinori Fukunaga. Fault Analysis on SIMON Family of Lightweight Block Ciphers. In *International Conference on Information Security and Cryptology*, pages 175–189. Springer, 2014.

[14] Banashri Karmakar and Dhiman Saha. PRINCE under Differential Fault Attack: Now in 3D. In *Proceedings of the 4th ACM Workshop on Attacks and Solutions in Hardware Security*, pages 81–91, 2020.

[15] Nasour Bagheri, Reza Ebrahimpour, and Navid Ghaedi. New differential fault analysis on PRESENT. *EURASIP Journal on Advances in Signal Processing*, 2013(1):1–10, 2013.

[16] Navid Vafaei, Nasour Bagheri, Sayandeep Saha, and Debdeep Mukhopadhyay. Differential Fault Attack on SKINNY Block Cipher. In *International Conference on Security, Privacy, and Applied Cryptography Engineering*, pages 177–197. Springer, 2018.

[17] S Ehsan Hosiny Nezhad, Masoumeh Safkhani, and Nasour Bagheri. Relaxed Differential Fault Analysis of SHA-3. *The ISC International Journal of Information Security*, 11(2):129–143, 2019.

[18] Navid Vafaei, Sayandeep Saha, Nasour Bagheri, and Debdeep Mukhopadhyay. Fault Attack on SKINNY Cipher. *Journal of Hardware and Systems Security*, 4(4):277–296, 2020.

[19] Christof Beierle, Gregor Leander, Amir Moradi, and Shahram Rasoolzadeh. CRAFT: Lightweight Tweakable Block Cipher with Efficient Protection Against DFA Attacks. *IACR Transactions on Symmetric Cryptology*, 2019(1):5–45, 2019.

[20] Anita Aghaie, Amir Moradi, Shahram Rasoolzadeh, Aein Rezaei Shahmirzadi, Falk Schellenberg, and Tobias Schneider. Impeccable Circuits. *IEEE Transactions on Computers*, 69(3):361–376, 2019.

[21] Aein Rezaei Shahmirzadi, Shahram Rasoolzadeh, and Amir Moradi. Impeccable Circuits II. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020.

[22] Shahram Rasoolzadeh, Aein Rezaei Shahmirzadi, and Amir Moradi. Impeccable Circuits III. In *2021 IEEE International Test Conference (ITC)*, pages 163–169. IEEE, 2021.

[23] Hosein Hadipour, Sadegh Sadeghi, Majid M. Niknam, Ling Song, and Nasour Bagheri. Comprehensive Security Analysis of CRAFT. *IACR Trans. Symmetric Cryptol.*, 2019(4):290–317, 2019.

[24] Hosein Hadipour, Nasour Bagheri, and Ling

Song. Improved Rectangle Attacks on SKINNY and CRAFT. *IACR Transactions on Symmetric Cryptology*, pages 140–198, 2021.

[25] Muhammad ElSheikh and Amr M Youssef. Related-Key Differential Cryptanalysis of Full Round CRAFT. In *International Conference on Security, Privacy, and Applied Cryptography Engineering*, pages 50–66. Springer, 2019.

[26] Victor Arribas, Felix Wegener, Amir Moradi, and Svetla Nikova. Cryptographic Fault Diagnosis using verFI. In *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 229–240. IEEE, 2020.

[27] Xinjie Zhao, Shize Guo, Tao Wang, Fan Zhang, and Zhijie Shi. Fault-propagate pattern based DFA on PRESENT and PRINTcipher. *Wuhan University Journal of Natural Sciences*, 17(6):485–493, 2012.

[28] Xiaofei Guo, Debdeep Mukhopadhyay, Chenglu Jin, and Ramesh Karri. Security analysis of concurrent error detection against differential fault analysis. *Journal of Cryptographic Engineering*, 5(3):153–169, 2015.

Navid Vafaei has been pursuing a Ph.D. in the Department of Electronic Engineering at Shahid Rajaee Teacher Training University under the supervision of Dr. Nasour Bagheri. His research interests include formal methods and hardware security.

Hamed Ramzanipour received his B.Sc. degree in electrical engineering from the University of Science and Technology of Mazandaran, Iran, in 2019. He also received an M.Sc. degree in system telecommunication engineering from Shahid Rajaee University, Tehran, Iran, in 2021. His research interests are cryptography, hardware security, and side-channel analysis.

Nasour Bagheri received the M.Sc. and Ph.D. degrees in electrical engineering from the Iran University of Science and Technology (IUST), Tehran, Iran, in 2002 and 2010, respectively. He is currently an Associate Professor at the Electrical Engineering Department of Shahid Rajaee Teacher Training University, Tehran, and also the head of CPS^2 laboratory there. He is also a part-time researcher at the Institute for Research in Fundamental Sciences. He is the author of more than 100 articles on information security and cryptology. His research interests include cryptology, more precisely, designing and analysis of symmetric schemes, such as lightweight ciphers, e.g., block ciphers, hash functions, and authenticated encryption schemes, cryptographic protocols for the constrained environment, such as RFID tags and the IoT edge devices and hardware security, e.g., the security of symmetric schemes against side-channel attacks, such as fault injection and power analysis.