The ISC Int'l Journal of Information Security
ISeCure

November 2022, Volume 14, Number 3 (pp. 9–19)
http://www.isecure-journal.org

Presented at the ISCISC'2022 in Rasht, Iran.

Practical Differential Fault Analysis on SKINNY [☆]

Navid Vafaei [1], Maryam Porkar [1], Hamed Ramzanipour [1], and Nasour Bagheri [1,2,*]
[1] *Electrical Engineering Department, Shahid Rajaee Teacher Training University, Tehran, Iran.*
[2] *School of Computer Science, Institute for Research in Fundamental Sciences, Tehran, Iran.*

ARTICLE INFO.

Keywords:
Differential Fault Analysis, SKINNY, Glitch Frequency, Nibble Fault Injection

Type:
Research Article

doi:
10.22042/isecure.2022.14.3.2

dor:
20.1001.1.20082045.2022.14.3.2.6

ABSTRACT

SKINNY is a lightweight tweakable block cipher that for the first time introduced in CRYPTO 2016. SKINNY is considered in two block sizes: 64 bits and 128 bits, as well as three TWEAK versions. In the beginning, this paper reflects our findings that improve the effectiveness of DFA analysis on SKINNY, then accomplishes the hardware implementation of this attack on SKINNY. Assuming that TWEAK is fixed, we first present the Enhanced DFA on SKINNY64-64 and SKINNY128-128. In order to retrieve the master key with the minimum number of faults, this approach depends on fault propagation in intermediate rounds. In our latest evaluations we can retrieve the master key with 2 and 3 faults in SKINNY64-64 and SKINNY128-128 respectively. This result should be compared with 3 and 4 faults for 64-bit and 128-bit versions respectively, in the models presented in the former work. Using the glitch model as well as a set of affordable hardware equipment, we injected faults into various rounds of the SKINNY algorithm in the implementation phase. More accurately, we can inject a single nibble fault into a particular round by determining the precise timing of the execution sub-function.

1 Introduction

Nowadays, fault attacks are threats to cryptographic implementations. Fault attacks are based on leakage that can be exploited by faulty encryption, which facilitates the recovery of the key for the attacker. Fault injection is a non-invasive attack that disrupts the normal sub-operation of an encryption system with transient changes using approaches such as electromagnetic field induction, laser radiation, voltage and frequency glitches, etc. [1] A frequency glitch is one of the most common fault attacks that may be performed with the lowest cost. The fault attack concept was first introduced by Boneh *et al.* [2] in September 1996. Biham and Shamir expanded the fault attack model into a differential of correct and faulty values called Differential Fault Analysis (DFA) [3]. In DFA, the attacker uses the combining (e.g., XOR operation) differential values of faulty and correct ciphertext to extract the secret key. Later, some more methods such as fault sensitivity attack [4], statistical fault attack [5], statistical ineffective fault attack [6], statistical effective fault attack [7] and persistent fault analysis [8–10] were also pre-

[*] Corresponding author.

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

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

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

sented. Despite the introduction of new types of fault analysis in recent years, DFA remains an invaluable tool in fault analysis due to the relaxed fault model assumptions.

SKINNY [11] is a lightweight tweakable block cipher that was proposed to compete with the NSA design SIMON. The tweakable block ciphers take advantage of one public input next to a secret key. To strengthen the security of each encryption, a (random) TWEAK is XORed with the master key. The SKINNY is available in two distinct block sizes: 64 and 128 bits. The key size (referred to as TWEAKEY in the SKINNY specification) will vary from 64 bits up to 512 bits. Three SKINNY variants are specified based on block size and key size: $n-n$, $n-2n$, and $n-3n$, where the first n indicates the block size and the other one, n, 2n, and 3n denote the TWEAKEY sizes.

According to the received wisdom, SKINNY provides significant security against linear and differential cryptanalysis. With this in mind, for the reduced rounds, some impossible differential attack was proposed in [12–16]. Moreover, others investigations on differential fault analysis of all SKINNY variants were reported in [17–19]. In the aforementioned fault attacks, the attacker exploits the differences between correct and faulty ciphertexts in the S-boxes' input and output for each round; all of these assessments assume that the TWEAKs have been fixed.

We present a fault attack on one variant of SKINNY $n-n$ that employs DFA in the deeper rounds. This method can reduce the fault injections compared to the previous attacks on SKINNY. Also, it covers other variants of SKINNY. Like the former attacks, the TWEAK is considered fixed and known to the attacker. In this paper, we show that the master key can be recovered with a high probability by injecting two and three faults at the beginning of the $(R-5)$ round in SKINNY64-64 and SKINNY128-128, respectively. We use low-cost equipment to inject frequency glitches into the SKINNY algorithm on a predefined nibble fault during a particular round in the hardware implementation section by determining the precise processing time of the operations. Our method of injecting faults is also applicable to the fault model in [17], since we know the timing of each round. Given the round's start time, it's more likely that the fault is injected into a desired nibble in the first row at the beginning of the required round. In our approach, the attacker should only inject fault in two nibbles at most, while in [17], it is required to change the target nibble more than twice. For example, SKINNY128-128 requires 22 faults on average to retrieve the key. Although some of these faults may occur many times in the same byte, retrieving the key still necessitates extensive adjustment. Because if an attacker wants to change the target byte, the time of the glitch or the laser's position should be modified. It was tough to adjust the fault position in the laser or the glitch time.

Table 1. Number of rounds of SKINNY for different input block(n) and TWEAKEY(z) sizes

n/z	1	2	3
64 bits	32 rounds	36 rounds	40 rounds
128 bits	40 rounds	48 rounds	56 rounds

The rest of this paper is organized as follows. In Section 2, we present the specification of the SKINNY cipher family. The previous DFA attacks on SKINNY are described in Section 2.2 along with the complexity analysis. The enhanced differential fault attack is introduced in Section 3. Hardware implementation of frequency glitch by injection of nibble fault in different SKINNY rounds is present in Section 4. Finally, we conclude in Section 5.

2 Preliminaries

In this section, we will take a quick look at the SKINNY specification; more detailed description can be found here [11]. SKINNY is a lightweight block cipher that comes in two block sizes: 64-bit and 128-bit. The input, state, and output of the SKINNY have been constructed as 4×4 array of cells in both variants, n=64 and n=128 (n is the block size). Each cell can be a nibble or a byte(s-bit), where the block size is 64 or 128 respectively ($s=\frac{n}{16}$). SKINNY based on the TWEAKEY construction. Three different TWEAK lengths by $t=n$-bit, $t=2n$-bit and $t=3n$-bit are employed. SKINNY can be shown based on the block size and TWEAK which is denoted as SKINNY$n-t$, hence, it takes a n-bit input and sorts it as follows: $\boldsymbol{m}=m_1||m_2||\cdots||m_{15}||m_{16}$, where m_i shows a s-bit cell. The input and internal state formation are as follows:

$$IS=\begin{bmatrix} m_1 & m_2 & m_3 & m_4 \\ m_5 & m_6 & m_7 & m_8 \\ m_9 & m_{10} & m_{11} & m_{12} \\ m_{13} & m_{14} & m_{15} & m_{16} \end{bmatrix} \tag{1}$$

In the next subsection, we provide the necessary details of different sub-operations of SKINNY.

2.1 Specification of Sub-Operations

One round of SKINNY consisting total 5 sub-operations – SubCells(SC), AddConstants(AC), AddRoundTweakey(ART), Shift Rows(SR), and MixColumns(MC). The total number of rounds depends on the block and the TWEAK sizes (see Table 1).

Table 2. The 4-bit S-box used in SKINNY-64 in hexadecimal form

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S_4[x]$	C	6	9	0	1	A	2	B	3	8	5	D	4	E	7	F

Table 3. The LFSR used in TK_2 and TK_3 in SKINNY

TK	s	LFSR
TK_2	4	$(x_4\|\|x_3\|\|x_2\|\|x_1) \rightarrow (x_3\|\|x_2\|\|x_1\|\|x_4 \oplus x_3)$
	8	$(x_8\|\|x_7\|\|\cdots\|\|x_1) \rightarrow (x_7\|\|x_6\|\|\cdots\|\|x_1\|\|x_8 \oplus x_6)$
TK_3	4	$(x_4\|\|x_3\|\|x_2\|\|x_1) \rightarrow (x_1 \oplus x_4\|\|x_4\|\|x_3\|\|x_2)$
	8	$(x_8\|\|x_7\|\|\cdots\|\|x_1) \rightarrow (x_7\|\|x_6\|\|\cdots\|\|x_1\|\|x_8 \oplus x_6)$

One round of SKINNY is depicted in Figure 1 [11].

`SubCells`(SC): A non-linear transformation on each cell is applied to the internal state (IS). The SubCells of SKINNY are designated based on the block-size. SubCells for 64-bit block size(a nibble for each cell) is depicted in Table 2. For more detail about SubCells for 128-bit block size refer to [11].

`AddConstants`(AC): This operation adds a constant value to internal state. The constant values are generated by Linear Feedback Shift Register (LFSR).

`AddRoundTWEAKEY`(ART): Two first rows of the AddRoundTweakey are bit-wise XORed to the IS. The TWEAKEY is a 4×4 array of s-bit cells, just like the IS. TWEAKs are updated using two functions shown in Figure 2. First, an operation changes the location of cells in each round by a permutation as follows:

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

Next, TK_2 and TK_3 (for different cell sizes) are updated by an LFSR, which is depicted in the Table 3.

`ShiftRows`(SR): The Shift Rows performs a rotation for the cells at the second, third and fourth row of IS.
`MixColumns`(MC): The IS multiplies with a 4×4 matrix $\mathbf{M}$.

$$\mathbf{M} = \begin{bmatrix} 1&0&1&1 \\ 1&0&0&0 \\ 0&1&1&0 \\ 1&0&1&0 \end{bmatrix} \qquad (3)$$

From the next section onwards, we shall describe the DFA attacks on SKINNY n-n and SKINNY n-$2n$.

2.2 Previous Differential Fault Analysis of SKINNY

In the DFA, fault injection occurs in the intermediate value. The differences between the correct and the obtained faulty ciphertexts and their connections with round key bits utilized to recover the master key. The DFA for all SKINNY variations was given access in [17, 19]. In mentioned works, the attack approach concentrated on the S-box's differential input and output values from the last round. The differential values changed during S-box executions; Then, some leakages can be exploited due to the non-linearity feature of the S-box. For more details, we can refer to the Section 2.3. The assumption of this attack is merely straightforward. The attacker can inject a random fault in a nibble or a byte at the beginning of a specific round, and also TWEAK fixed during the attack [1] . Later in [18] SKINNY was considered as one of the Substitution-Permutation networks to overview the DFA vulnerabilities in different block-ciphers. The attack used in [18] was the same as in the [17] paper. They illustrated the complexity of the attack for various numbers of correct and faulty pairs, as shown by the Table 4.

2.3 DFA Attack Model

At the attack's origin [17], it is assumed that an attacker can inject a random fault [2] at the beginning of the $(R-4)$ round. The fault propagates through MixColumns in each round, and the nonlinear operation of SKINNY, S-box changes the differential values in each round. The propagation of 5 rounds of SKINNY is depicted in Figure 3. However, the fault propagation of four rounds can be derived from the figure [3] .

Assume that a random fault is injected into the first cell at the beginning of the $(R-5)$ round, and the diffusion layer propagates the fault, particularly by the MixColumns. The differences between correct and faulty cells, which are called differential values, are altered by the S-box layer, and in the DFA attack, we are interested in the differential values where they are changed. Different differential values in the input and output of the S-box give the attacker leverage to recover the key. To be more specific, the solution of x in the equations $SC(x \oplus \Delta_{ip}) \oplus SC(x) = \Delta_o$ is required to recover the key, where Δ_{ip} and Δ_o is the differential input and output of S-box, respectively [4] .

[1] SKINNY models allow us to control the TWEAKs.
[2] We consider that a random fault is injected in a cell of SKINNY, so a random fault can be a nibble or a byte depending on the variant of SKINNY.
[3] The fault propagation is depicted from the $R-5$ round since this made the attack effective.
[4] x leads us to the key in this case.

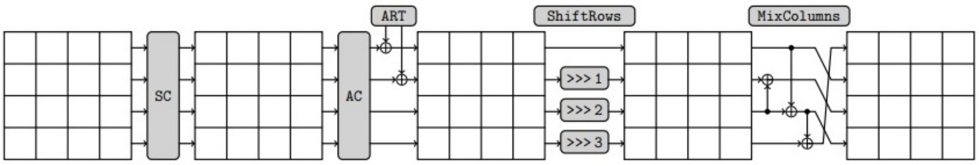


Figure 1. The SKINNY round function

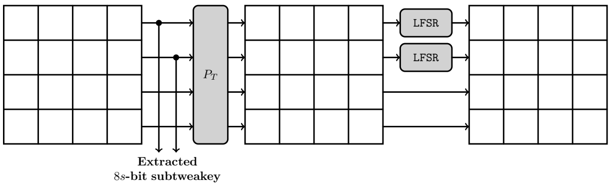


Figure 2. The TWEAKEY Schedule

Table 4. The comparison of the complexity in references (semi-random stands for faults that are injected in the first row of desired round)

Ref	Variant	Fault Model	Required Pair of Ciphertext	Remaining Key Candidate
[18]	64-64	Random Nibble	3	2^{34}
	128-128	Random Byte	4	$2^{11.56}$
[17]	64-64	Random Nibble	10.66	1
	128-128	Random Byte	22	1
This paper	64-64	Semi-Random Nibble	2	2^{6}
	128-128	Semi-Random Byte	3	$2^{30.8}$

To recover the x in the context of DFA, the attacker should first gain information about the Δip. The attacker can use the S-box differential distribution table to recover x. Since, for distinct input and output differential values of the S-box, the number of candidates is specified. The differential input and output table for SKINNY64-64 are depicted in the Table 5.

Consider the last round of SKINNY in Figure 3. Due to the linearity of MixColumns, Shift Rows, and AddRoundTweakey, it is straightforward to get the output differential values of the S-box in the last round. However, the S-box input differential values for the two upper rows are not obtained without the involved key, but equivalent differential values in the two lower rows are accessible for some cells in the upper rows. In other words, thanks to MixColumns characteristic, identical differential values are generated in certain cells. The cells with known input and output differential values of the S-box for the five last rounds are depicted in Figure 3 [5] . The differential input and output of the S-box may therefore be determined for specific cells. As a result, key candidates can be guessed for the cells with known input and output differential values of the S-box. The average number of key candidates for a cell with a known differential input and output of S-box can be extracted from the differential distribution table, which is depicted in Table 5 for SKINNY64-64. The candidates are $2^{1.42}$ and $2^{2.82}$ for SKINNY64-64 and SKINNY128-128 respectively. The most reasonable way to reduce the number of key candidates is to increase the number of cells with known S-box input and output differentials by injecting faults in deeper rounds or repeating injections to different cells. In [17], it is assumed that a fault attack is injected in the $(R-4)$. The differential value of S-box input for four cells on average may be derived for each fault in the last round. The number of cells depends on the position of the fault in the $(R-4)$ round. For example, if a fault is injected in the third row at the beginning of the $(R-4)$ round, the differential input value of the S-box for five cells can be retrieved. The attack can reduce the sixteen potential choices of key candidates for each cell to four possibilities at most in SKINNY64-64. Assume a fault is injected at the first nibble at the beginning of $(R-4)$, and the differential input of the S-box can be determined for four cells. Therefore, in SKINNY64-64, the remaining key of the last round key would be at most 2^{19}. By assuming that a fault attack can be repeated at the same round (not in the same nibble), differential values of S-box

[5] ΔX_7^R can be recovered due to the linear relationship between ΔX_{11}^R and ΔX_{15}^R.

input for other cells can also be retrieved. As a result, on average, the attacker would be able to retrieve the last round key by four pairs. By following the key recovery in the last round, the attacker can recover a portion of the penultimate round key using the previous faults. Although previous faults reduce the keyspace of the penultimate round, almost eight pairs are required on average to recover the master key. The procedure for the SKINNY128-128 is the same as the SKINNY64-64. However, more correct/faulty pairs are required. According to [17] to recover the master key of SKINNY64-64 and SKINNY128-128 the required number of faults is 10.66 and 21, respectively. The complexity of the attack is depicted in Table 4.

In the next section, we decrease the number of faults for both variants of SKINNY64-64 and SKINNY128-128 to recover the master key by brute force strategy. It is worth mentioning that a reduction in the number of fault injections is desirable because even in the random fault models, a fault might cause a crash on a system. By this approach, an improvement fault-based attack on AES was presented by Mukhopadhyay [20], which requires a brute-force search of 2^{32}.

3 Enhanced Differential Fault Attack

Algorithm 1 represents the main idea of the attack model and shows how to find candidates for four cells in the last and penultimate rounds. The algorithm starts with a differential fault attack in [17] by recovering key candidates for known differential input and output of S-box cells in the last round by using the relationship of cells in Section 2.3 to eliminate the incorrect keys. Therefore, key candidates have dropped for some cells but may not fully recover. Here, the algorithm refers to the remaining candidates as **True Candidate**. The attacker can employ the cell relationships in the deeper rounds to reduce the number of candidates by guessing a few undiscovered cell keys. As a result, additional "True" sets are determined, and the candidates are dropped by intersecting sets. The remaining candidates can finally use a brute force strategy to retrieve the master key.

3.1 Attack Model

Assume a fault is injected in the first row at the $(R-5)$ round. The first part is the same for both versions of SKINNY. The correct and faulty ciphertext is used to obtain the differential values in the input and output of the S-box in the last round. We partially decrypt the ciphertext towards the input of the S-box in the $(R)^{th}$ round and recover the possible keys. The Equation 4 shows the path from the ciphertext to ΔX_i^R.

Algorithm 1 Enhanced DFA on SKINNY

```
Require: $P$: Plainrtext; $C$: Ciphertext; $C^*$: Faulty ciphertext; $Enc$: Encryption; $TC(i)$: True Candidates for $TK(i)$; $TK(i)$: "i"; $n$: block size;
Ensure: candidates for $TC$.

Require: (initialization)
$\omega \leftarrow \frac{n}{64} + 1$
$Tweak$ ←Fixed
$MainKey$ ←Fixed
$Nibble/byte \leftarrow 0$                              ▷ fault location
$f$ ←Random number                                      ▷ Value of fault
for $0 \leq x < \omega$ do
    $plaintext$ ←Random number
    True $Enc(P) = C$
    Faulty $Enc(P) = C^*$
    if $round == (R-5)$ then                            ▷ injection Fault
        $Nibble = Nibble \oplus f$
    end if
    $\Delta X_i^R \leftarrow C\ ,\ C^*$                  ▷ (EQ:4)
check 1:DFA Regular
    $\Delta X_4^R = \Delta X_{12}^R, \Delta X_1^R = \Delta X_{13}^R$        ▷ $TC_9, TC_{12}$
check 2: Guess "11","13" and obtain $TC_2, TC_6$
    for $0 \leq$"11"$< 2^{2^\omega}$ do
        for $0 \leq$"13"$< 2^{2^\omega}$ do
            if $(\Delta X_0^{R-1} == \Delta X_4^{R-1} == \Delta X_{12}^{R-1})$ then :
                save $(TC_\omega(2), TC_\omega(6))$
            end if
        end for
    end for
end for
$\bigcap_{j=1}^{\omega} TC_\omega$
print$(TC(9), TC(12), TC(2), TC(6))$
```

$$\Delta X_i^R = SC^{-1}(Y_i^R) \oplus SC^{-1}(Y_i^{*R}) \tag{4}$$
$$= SC^{-1}(AC^{-1}(ART^{-1}(SR^{-1}(MC^{-1}(C_i^R))) \oplus TK_i^R))$$
$$\oplus SC^{-1}(AC^{-1}(ART^{-1}(SR^{-1}(MC^{-1}(C_i^{*R}))) \oplus TK_i^R))$$

Where X_i^R and Y_i^R are the i^{th} cell at (R) rounds of the input and output of the S-box, respectively. We can extract from the ciphertext the inverse of Shift Rows, MixColumns, and AddConstants in the last round. Also, differential values can pass through AddRoundTweakey. ΔX_4^R and ΔX_{12}^R are equal because of the MixColumns diffusion, which creates three same differentials in the first column in the final round. As a result, key candidates of $TK(12)$ can be recovered. Furthermore, $TK(9)$ can be recovered because ΔX_1^R and ΔX_{13}^R are equal. ΔX_6^R can be derived by $\Delta X_{10}^R \oplus \Delta X_{14}^R$ and also $\Delta X_7^R = \Delta X_{11}^R \oplus \Delta X_{15}^R$. Therefore, the key candidates of $TK(14)$ and $TK(15)$ can be retrieved.

Since the input and output differential values of the S-box for certain cells (four cells) in the last round are specified, the key candidates for the differential values are determined by the possible portion of DDT in Table 5. By considering $2^{1.42}$ and $2^{2.82}$ remaining

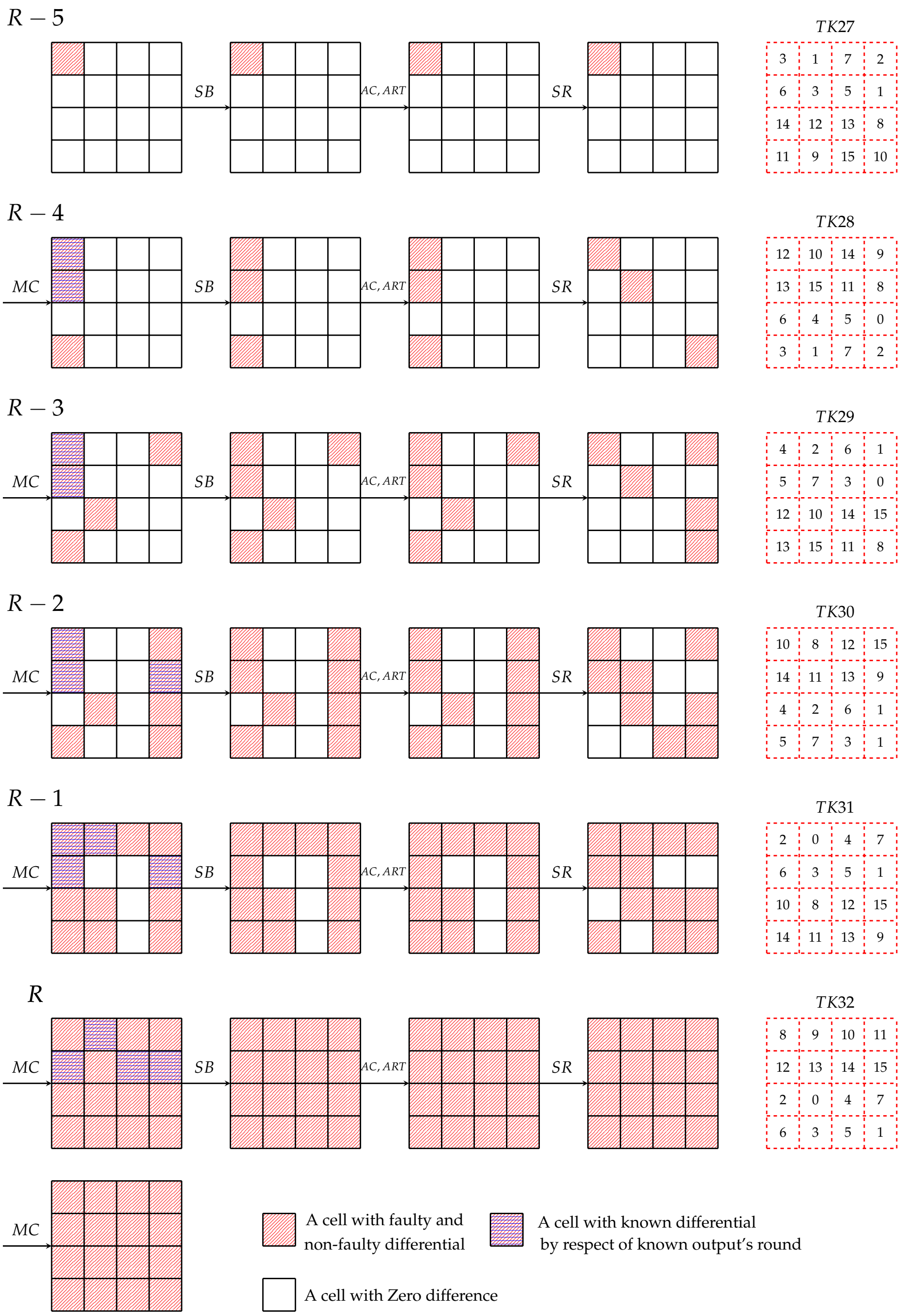


Figure 3. The fault propagation pattern in SKINNY with the fault injected at the 1st cell at the beginning of round $(R-5)$ (Each variable represents a non-zero fault differential, and each empty cell specifies a zero differential)

key candidates for each known input and output differential S-box's cell, the key candidates for these four cells drop down from 2^{16} and 2^{32} to $2^{5.6}$ and $2^{9.6}$ for SKINNY64-64 and SKINNY128-128 respectively. From here, we will explain the details of attack for SKINNY64-64 and SKINNY128-128 in Section 3.2 and Section 3.3, respectively.

3.2 SKINNY64-64

SKINNY is a column-based cipher. Hence, we analyze each column separately. By recovering part of the key in the last round, known differential values of the S-box's input in the penultimate round (and maybe in deeper rounds) should be targeted. The differential cells by known outputs in each round are specified in Figure 3.

Table 5. Differential input-output ($\Delta I/\Delta O$) of SubCells of SKINNY 64

$\Delta I/\Delta O$	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	4	4	4	4	0	0	0	0
2	0	4	0	4	0	4	4	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	2	2	2	2	2	4	2	2
4	0	0	4	0	0	0	2	2	0	0	0	4	2	2	0	0
5	0	0	4	0	0	0	2	2	0	0	4	0	4	2	0	0
6	0	2	0	2	2	0	0	2	2	0	2	0	0	2	2	0
7	0	2	0	2	2	0	0	2	0	2	0	2	2	0	0	2
8	0	0	0	0	4	4	0	0	0	0	0	0	2	2	2	2
9	0	0	0	0	4	4	0	0	0	0	0	0	2	2	2	2
A	0	0	0	0	0	4	4	0	2	2	2	2	0	0	0	0
B	0	4	0	4	0	0	0	0	0	0	0	0	2	2	2	2
C	0	0	4	0	0	0	2	2	4	0	0	0	0	0	2	2
D	0	0	4	0	0	0	2	2	0	4	0	0	0	0	2	2
E	0	2	0	2	2	0	0	2	0	2	0	2	0	2	2	0
F	0	2	0	2	2	0	0	2	2	0	2	0	2	0	0	2

✓In the penultimate round differential values of the S-box's input in the first column, ΔX_0^{R-1}, ΔX_4^{R-1} and ΔX_{12}^{R-1} are equal. On the other hand, the output differential value of the S-box is dependent on the last round key. $TK(12)$, $TK(13)$, and $TK(11)$ are the involved keys in the last round for the recovery of the first column of differential values of the S-box's input in the penultimate round. $TK(12)$ candidates are already recovered. As a result, ($TK(13)$, and $TK(11)$) should be guessed. The candidates for $TK(2)$ and $TK(6)$ can be obtained.

✓To extract differential values of ΔX_1^{R-1} and ΔX_4^{R-1}, $TK(13)$ and $TK(8)$ is required. $TK(13)$ is already guessed. Hence just $TK(8)$ should be guessed to recover the candidates for $TK(0)$.

✓The last differential values that should be obtained in the penultimate round are ΔX_7^{R-1}, ΔX_{11}^{R-1} and ΔX_{15}^{R-1}. $TK(13)$ and $TK(12)$ are already used in our calculation, we should guess $TK(10)$. Therefore, key candidates for $TK(1)$ can be derived.

Till now, we guess $\{TK(13), TK(11), TK(8), TK(10)\}$ to retrieve the key candidates for $\{TK(0), TK(1), TK(2), TK(6)\}$. Also, the key candidates of $\{TK(9), TK(12), TK(14), TK(15)\}$ are extracted. Consequently, the upper bound of the complexity would be $2^{1.42\times 8} \times 2^{32} = 2^{43.6}$.

Another fault reduces the overall complexity to 2^{32}. By the first injection in the first row at the beginning of the $(R-5)$, eight cells are activated in the last two rounds. In the worst-case scenario, an extra fault would reactivate eight previous cells, and the total complexity of the master key would be decreased to the threshold of practical search. In other words, 2^{32} remaining candidates can be guessed to exploit the differential in $(R-2)$. Three differential cells in $(R-2)$ decrease the complexity of the key candidates, as shown in Figure 3. It is possible to use differentials in the deeper rounds for the first or second fault propagation, but it is unnecessary. The brute force search of 2^{32} candidates takes less than a second with a modern CPU. However, we have employed the differentials in $(R-2)$ to reduce the keyspace.

✓To reduce the keyspace differential values of the $(R-2)$, ΔX_0^{R-2}, ΔX_4^{R-2} and ΔX_{12}^{R-2} is considered. To obtain the differential values, $\{TK(3), TK(6), TK(7)\}$ is required. Since $TK(6)$ is already extracted, we should guess $\{TK(3), TK(7)\}$. These differentials lead us to $TK(10)$ and $TK(7)$ in the $(R-2)$ round and eliminate wrong keys. The complexity would be $2^{32} \times 2^{2\times -2.6} = 2^{26.84}$.

✓In the next step ΔX_7^{R-2}, ΔX_{11}^{R-2} and ΔX_{15}^{R-2} is considered. Involved keys in the penultimate round which create these differentials are $\{TK(3), TK(6), TK(4)\}$. We guess $TK(4)$ and the complexity would be $2^{26.84} \times 2^{2^{-2.6}} = 2^{24.24}$. If we consider the worst-case scenario for the second fault, the complexity would be decreased to 2^{20} due to the same differential cells at $(R-2)$ round.

Now the total complexity is 2^{20}. The master key can be retrieved for SKINNY64-64 by brute force attack. However, differential cells in deeper rounds$(R-3)$ and $(R-4)$ can be employed to recover the $\{TK(13), TK(12), TK(4), TK(5)\}$ and also wrong candidates of keys is eliminated. After these two rounds, the total complexity would be $2^{6.2}$. So, the key is recovered by two faults in SKINNY64-64.

3.3 SKINNY128-128

As well as SKINNY64-64, the initial fault injection can retrieve key candidates for $\{TK(9), TK(12), TK(14), TK(15)\}$. Unlike the previous approach in Sec. 3.2, where we guess the key candidates for certain cells by two faults, injecting three faults in the first row to decrease the complexity is necessary. Key candidates resulting from a known input and output differential of S-box in the last round are reduced $2^{-5.2}$ on average (from 2^8 to $2^{2.8}$) by the first fault injection. If two faults are injected in one cell, $\{TK(9), TK(12), TK(14), TK(15)\}$ can be recovered completely. As a result, key candidates can be extracted uniquely by the next fault injection in the same cell. Here, for the sake of simplicity, suppose two faults in the first cell at the beginning of the

$(R-5)$. Therefore, by three faults, two in the first cell and one in another random cell in the first row, four key cells are recovered, and at least for two more cells, true candidates are determined. Thus, the key candidate decreased from 2^{32} to $2^{2.8\times 2} \times 2^{16} = 2^{21.6}$ for four other cells in the last round. Therefore, it is possible to guess the remaining key candidates. Then, we can decrypt the ciphertext partially to recover the differential values in the input of the S-box in the penultimate round.

✓To recover ΔX_0^{R-1}, ΔX_4^{R-1} and ΔX_{12}^{R-1} which can extract $TK(2)$, $TK(6)$, it is required to guess $TK(11)$, $TK(13)$. Total complexity would be $2^{64} \times 2^{21.6} \times 2^{2\times(-8)} = 2^{69.6}$.

✓In the next step to recover ΔX_1^{R-1} and ΔX_4^{R-1}, $TK(13)$ and $TK(8)$ is required. $TK(8)$ should be guessed to recover $TK(0)$. Total guessed key candidates is 2^{24} and the total complexity would be $2^{69.6} \times 2^{(-8)} = 2^{61.6}$.

✓ΔX_7^{R-1}, ΔX_{11}^{R-1} and ΔX_{15}^{R-1} are differential values which should be achieved to recover $TK(1)$. To reach this goal, we should guess $TK(10)$, which decreases the total complexity keys to $2^{61.6} \times 2^{(-8)} = 2^{51.6}$. The overall guessed key candidates would be 2^{32}.

✓We assume that three faults in the first row are injected into two cells, one in the first cell and the second one is random. The complexity of two faults in one cell is discussed. Suppose the third fault in the second cell of SKINNY, candidates of $TK(3)$, $TK(5)$ can be recovered, and the overall complexity would be $2^{51.6} \times 2^{2\times(-5.2)} = 2^{41.2}$.

It is necessary to employ differential values of the S-box's input in deeper rounds in this stage to reduce the remaining keyspace of the master key for a feasible brute force attack. In order to reduce the keyspace $(R-2)$, ΔX_0^{R-2}, ΔX_4^{R-2}, and ΔX_{12}^{R-2} should be retrieved which is requiring access of $TK(3), TK(6), TK(7)$. Because $TK(3)$, and $TK(6)$ have been extracted, we should guess $TK(7)$. The differential values would eliminate certain impossible keys. The complexity would be $2^{41.2} \times 2^{-5.2} = 2^{36}$. The total guessed values remained fixed since the former guessed values do not have impacts on $TK(3), TK(6)$ recovery. The total guess values would be $2^{32} + 2^{8} \approx 2^{32}$.

ΔX_7^{R-2}, ΔX_{11}^{R-2} and ΔX_{15}^{R-2} in the next step should be considered. Involved keys in the penultimate round which create these differential values are $\{TK(3), TK(6), TK(4)\}$. We guess $TK(4)$ and the complexity would be $2^{36} \times 2^{-5.2} = 2^{30.8}$. The total guess values would be $2^{32} + 2^{16} \approx 2^{32}$. Here, the total complexity is less than a practical brute force attack, and we can recover the master key. However, if we guess the total $2^{30.8}$ key candidates, we can use differential values of deeper rounds to decrease the candidates.

4 Hardware Implementation of Nibble Fault Injection

The aim of this section is to describe the results of injecting a fault into a specific nibble in the desired round of the SKINNY64-64 as a hardware implementation. We assume that the attacker has accessibility to the processing time for each S-box at different rounds. This presumption is fair and similar to [17].

4.1 Fault Model and Implementation Preliminaries

We have used ATMEGA32A(AVR) to implement the SKINNY encryption (target circuit) and SPARTAN6 XC6SLX9(FPGA) to inject a fault into the external crystal oscillator of the AVR by using the glitch frequency model. Figure 4 shows the target and glitch circuit details. The circuits are processing at two different clock speeds. The glitch signal and one of the crystal oscillator pins of the victim circuit are XORed to be used as the input frequency of the AVR. We employ the SN74F86 package for the XOR gate.

The precision of the fault injection at a specific location (nibble or byte of an arbitrary sub-function) is strongly associated with the difference between the clock frequency bandwidth of the target algorithm's processing and the glitch signal, in addition to the latency at the gate level and the additional noise. This implementation's target circuit and the FPGA board clock frequency bandwidths are 2.47 MHz and 50 MHz, respectively. Thus it is possible to inject the transient fault at high speed into the desired location since the target circuit is processing at a significantly slower clock rate. Hence, it is required to determine the processing time of the S-box's inputs to inject a fault into a specific nibble; we set a signal containing the precise start time of the S-box processing to apply the glitch signal. On the one hand, this control signal is an input for FPGA. On the other hand, the FPGA generates the glitch signal as one of the inputs of the XOR gate after receiving the start time of the first nibble processing (for each round) to inject a fault (glitch) into the target circuit encryption processing.

4.2 Experimental Results

We can improve the accuracy of fault injection to a single nibble by assessing the timing of each round's sub-functions. The hardware implementation of the fault injection mechanism is depicted in Figure 5. To

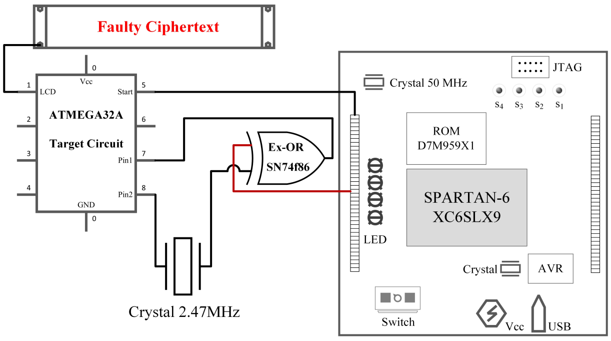


Figure 4. The Fault injecting setup

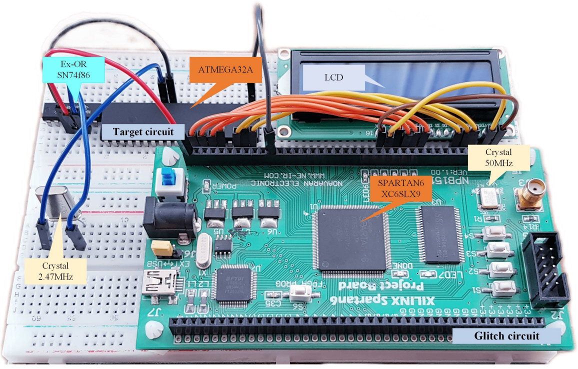


Figure 5. Glitch frequency setup

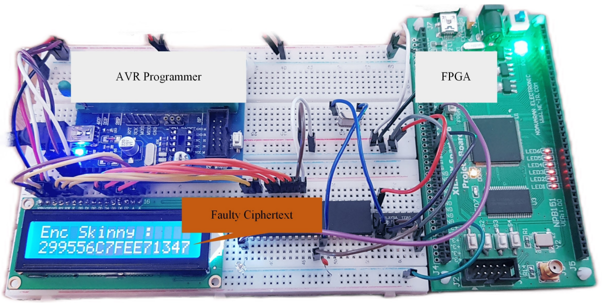


Figure 6. Faulty ciphertext by injection at 27th-round

achieve this, the FPGA receives a signal that indicates the S-box processing time at desired round. The effect of the glitch signal through the XOR gate may disturb normal processing. Then, each faulty and non-faulty ciphertext is compared in the DFA scenario. Figure 6 represents the faulty ciphertexts by injecting a fault into the third nibble at the beginning of the 27th round of SKINNY. Also, the nibble fault propagation pattern from the 27th round can be seen in Figure 3.

The main goal of this section was to inject a nibble fault into a specific round of the SKINNY algorithm with cheap equipment. To see the consequences of the implementation, the nibble fault injections into different rounds can be seen in Appendix A. Also, The Hardware programming for this implementation, including the glitch mechanism and target circuit, is available at the following address:
`https://github.com/Navidvafaei/Glitch-Attack.git`

5 Conclusion

SKINNY is a lightweight tweakable block cipher based on the recently proposed TWEAKEY framework[11]. In this paper, we enhanced the former differential fault attack on SKINNY[17] by recovering the master key with two and three faults in SKINNY64-64 and SKINNY128-128, respectively. Additionally, in the hardware implementation phase, we injected the nibble fault into any chosen round of the SKINNY algorithm using the glitch frequency model for validating the theoretical model. The fault was injected by specifying the S-box processing time at the beginning of each round. The next potential direction would be a fault attack in the presence of the countermeasure, which was proposed by Aghaie *et al.*[21].

References

[1] 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.

[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] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. *Advances in Cryptology—CRYPTO'97*, pages 513–525, 1997.

[4] Yang Li, Kazuo Sakiyama, Shigeto Gomisawa, Toshinori Fukunaga, Junko Takahashi, and Kazuo Ohta. Fault sensitivity analysis. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 320–334. Springer, 2010.

[5] Christoph Dobraunig, Maria Eichlseder, Thomas Korak, Victor Lomné, and Florian Mendel. Statistical fault attacks on nonce-based authenticated encryption schemes. In *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 22*, pages 369–395. Springer, 2016.

[6] Christoph Dobraunig, Maria Eichlseder, Thomas Korak, Stefan Mangard, Florian Mendel, and Robert Primas. Sifa: exploiting ineffective fault inductions on symmetric cryptography. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 547–572, 2018.

[7] Navid Vafaei, Sara Zarei, Nasour Bagheri, Maria

Eichlseder, Robert Primas, and Hadi Soleimany. Statistical effective fault attacks: The other side of the coin. *IEEE Transactions on Information Forensics and Security*, 2022.

[8] 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.

[9] 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.

[10] 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.

[11] 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 *Annual Cryptology Conference*, pages 123–153. Springer, 2016.

[12] Jeremy Jean, Amir Moradi, Thomas Peyrin, and Pascal Sasdrich. Bit-sliding: A generic technique for bit-serial implementations of spn-based primitives – applications to aes, present and skinny. Cryptology ePrint Archive, Report 2017/600, 2017.

[13] Mohamed Tolba, Ahmed Abdelkhalek, and Amr M Youssef. Impossible differential cryptanalysis of skinny. Technical report, Cryptology ePrint Archive, Report 2016/1115, 2016. http://eprint. iacr. org/2016/1115, 2016.

[14] Guozhen Liu, Mohona Ghosh, and Song Ling. Security analysis of skinny under related-tweakey settings. Technical report, Cryptology ePrint Archive, Report 2016/1108, 2016. http://eprint. iacr. org/2016/1108, 2016.

[15] Sadegh Sadeghi, Tahereh Mohammadi, and Nasour Bagheri. Cryptanalysis of reduced round SKINNY block cipher. *IACR Trans. Symmetric Cryptol.*, 2018(3):124–162, 2018.

[16] Ralph Ankele, Subhadeep Banik, Avik Chakraborti, Eik List, Florian Mendel, Siang Meng Sim, and Gaoli Wang. Related-key impossible-differential attack on reduced-round skinny. Technical report, Cryptology ePrint Archive, Report 2016/1127, 2016. http://eprint. iacr. org/2016/1127, 2017.

[17] 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.

[18] Mustafa Khairallah, Xiaolu Hou, Zakaria Najm, Jakub Breier, Shivam Bhasin, and Thomas Peyrin. SoK: On DFA Vulnerabilities of Substitution-Permutation Networks. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, pages 403–414, 2019.

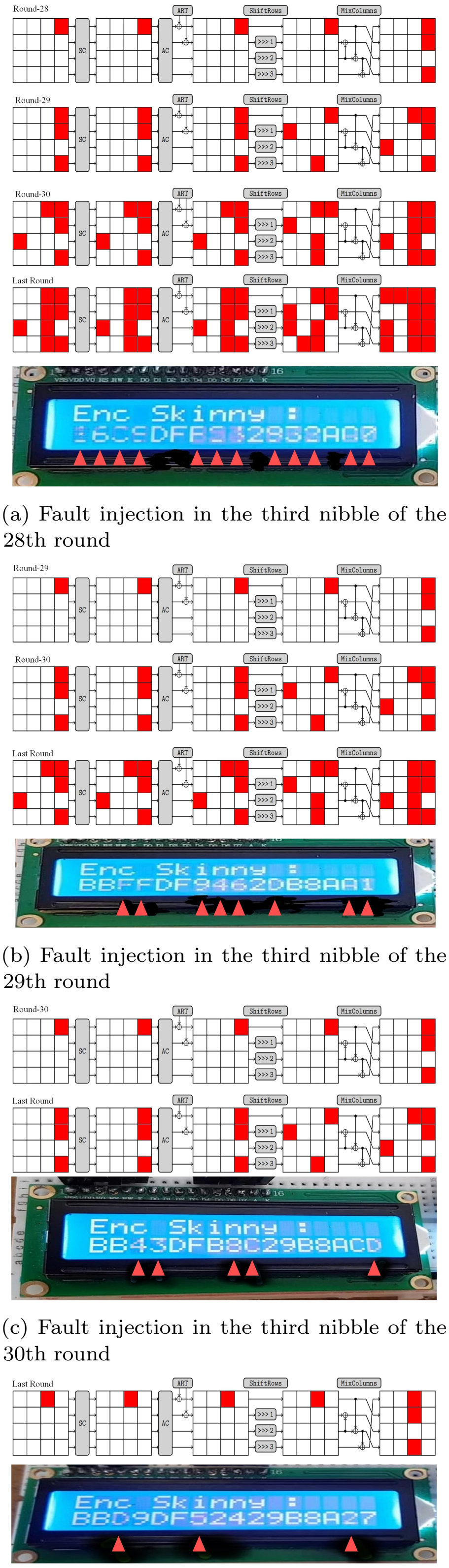
[19] Navid Vafaei, Sayandeep Saha, Nasour Bagheri, and Debdeep Mukhopadhyay. Fault attack on skinny cipher. *Journal of Hardware and Systems Security*, 4(4):277–296, 2020.

[20] Debdeep Mukhopadhyay. An improved fault based attack of the advanced encryption standard. In *International Conference on Cryptology in Africa*, pages 421–434. Springer, 2009.

[21] 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.

A The Nibble Fault Injection

In the SKINNY64-64 version the non-faulty ciphertexts is 0xbb39dfb2429b8ac7. The faulty ciphertexts are obtained by injecting a nibble fault at the beginning of the S-box, according to Figure A.1. It is possible to inject a fault into the desired nibble at various rounds, depending on the processing time for S-box's inputs. The fault propagation pattern displays the results with a red mark. (a), (b), (c), and (d) sub-figures are related to the nibble fault injection at the 28th, 29th, 30th and last round, respectively.



(a) Fault injection in the third nibble of the 28th round

(b) Fault injection in the third nibble of the 29th round

(c) Fault injection in the third nibble of the 30th round

(d) Fault injection in the second nibble of the last round

Figure A.1. Fault injection results in the different rounds of the SKINNY algorithm

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.

Maryam Porkar received her M.Sc. degree in system telecommunication engineering from Shahid Rajaee University, Tehran, Iran, in 2020. Her research interest is symmetric cryptography.

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.