SIFA on Nonce-based Authenticated Encryption: When Does It Fail? Application to ASCON

Viet Sang Nguyen, Vincent Grosso, Pierre-Louis Cayrel
Université Jean Monnet Saint-Etienne, CNRS, Institut d Optique Graduate School,
Laboratoire Hubert Curien UMR 5516,
F-42023 Saint-Etienne, France
{*viet.sang.nguyen, vincent.grosso, pierre.louis.cayrel*}*@univ-st-etienne.fr*

***Abstract*—In nonce-based authenticated encryption schemes, fault attacks such as differential fault analysis are not applicable to due to the uniqueness of the nonce. In this context, Dobraunig *et al.* showed that Statistical Ineffective Fault Attacks (SIFA) remain applicable and powerful. The authors proposed a SIFA-based attack strategy targeting the initialization in nonce-based authenticated encryption schemes and demonstrated its practicality using a common fault method: instruction skip.**

In this work, we provide a more in-depth analysis of this attack strategy, with a focus on instruction skip as the fault method. First, we model common instruction skip scenarios in practice and formalize the probability that a fault is ineffective. Our analysis reveals that this probability depends on the instruction type and the device architecture. Notably, we show that obtaining a sufficient number of ineffective faults for SIFA is likely impractical when skipping a `XOR` instruction on 32-bit or 64-bit systems, where register data tends to be uniformly distributed. Second, we prove that, in certain authenticated encryption implementations, the intermediate value targeted by the attack unexpectedly remains unbiased under ineffective faults, making SIFA inapplicable. As a case study, we demonstrate this behavior in an 8-bit ASCON implementation.

***Keywords*-Fault Attack; SIFA; Instruction skip; Authenticated Encryption; ASCON;**

I. INTRODUCTION

Since the seminal works of Boneh *et al.* [1] and Biham and Shamir [2], fault attacks have emerged as a significant threat to cryptographic implementations. This threat is particularly serious for embedded devices, to which attackers often have physical access. Recently, NIST selected ASCON as the new standard for lightweight authenticated encryption, targeting such constrained environments. As a consequence, ASCON is expected to be widely deployed in real-world devices. This highlights the importance of investigating fault attacks against this authenticated encryption scheme.

In the literature, Differential Fault Analysis (DFA)-like attacks [2] have been shown to be ineffective against authenticated encryption schemes due to the use of a unique nonce in each encryption [3]. To address this limitation, Dobraunig *et al.* [4] proposed the use of Statistical Ineffective Fault Attacks (SIFA) [5], [6]. The authors demonstrated that their attack strategy applies to a wide range of authenticated encryption schemes where the nonce is mixed with the key during the initialization phase. The core idea of the attack is to cause a biased distribution in a bit before the S-box computation of the 2nd initialization round by *ineffective faults*, and then recover the key through a statistical analysis of this bias. The authors validated their approach by mounting attacks on 8-bit implementations of two authenticated encryption schemes based on the Keccak-f permutation [7], namely, Keyak [8] and Ketje [9]. To induce faults, they employed a common method, instruction skip, realized by clock glitching. The authors conjectured that their attack strategy can be adopted to other authenticated encryption schemes such as ASCON.

In [4] and many other demonstrations of SIFA [5], [6], instruction skip is a common used method to cause a bias in the intermediate value under ineffective faults. This skip typically targets an instruction involved in the S-box computation. However, it is often unclear which specific instruction was skipped in those demonstrations. This may lead one to think that blindly skipping any instruction in the S-box computation is sufficient to introduce a bias in the intermediate value under ineffective faults. In this work, we show that this is not the case in certain nonce-based authenticated encryption schemes.

Our contributions. This paper provides a more in-depth analysis of the attack strategy introduced by Dobraunig *et al.* [4], with a focus on instruction skip as the fault method. First, we model common instruction skip scenarios in practice and formalize the probability of a fault being ineffective. Our analysis shows that when an attacker skips an instruction, the probability of causing an ineffective fault depends on both the type of instruction and the device architecture. In particular, we show that skipping a `XOR` instruction on 32-bit or 64-bit systems is very unlikely to result in an ineffective fault, making SIFA becomes impractical.

Second, we show that in certain authenticated encryption implementations, the intermediate value targeted by the attack strategy in [4] remains unbiased under ineffective faults, making SIFA inapplicable in these cases. We model

such implementations and provide a formal proof for the uniformity of the targeted intermediate value. To support our findings, we present a case study on an 8-bit ASCON implementation, where the intermediate value remains uniformly distributed under ineffective faults.

Outline. This paper is organized as follows. Section II provides the background on ASCON and SIFA. Section III details the application of the attack strategy in [4] to ASCON. Section IV models common instruction skip scenarios and analyzes the probability that an ineffective fault occurs. Section V presents the authenticated encryption implementations where the intermediate value remains uniform under ineffective faults and a case study on ASCON. Finally, Section VI concludes our work.

II. BACKGROUND

In this section, we begin by describing the ASCON authenticated encryption scheme. We then provide a brief introduction to SIFA, followed by a recap of its application to nonce-based authenticated encryption as proposed by Dobraunig *et al.* [4].

A. ASCON

ASCON [10] is a suite of Authenticated Encryption with Associated Data (AEAD) and hashing algorithms based on the duplex sponge construction [11]. This paper focuses on the authenticated encryption ASCON-AEAD128, referred to as ASCON hereafter. Figure 1 illustrates the encryption process. Its inputs include a 128-bit key K, a 128-bit nonce N, a constant initialization vector $\texttt{IV}$, associated data $A_1, \ldots, A_s$, each of r bits, and plaintexts $P_1, \ldots, P_t$, each of r bits. It produces as output a tag T of 128 bits and ciphertexts $C_1, \ldots, C_t$, each of r bits. The tag T is used during the decryption to verify the authenticity of the ciphertexts and associated data.

The permutations p^a and p^b, consisting of a and b rounds, form the core of the ASCON construction. The version standardized by NIST has $a = 12$, $b = 8$ and $r = 128$. Each round is composed of three steps operating on a 320-bit state: (1) addition of constants, (2) substitution layer (S-box), and (3) linear diffusion layer. The three steps are depicted in Figure 2. The 320-bit state is divided into five 64-bit words, which can be stored in one or more smaller-sized registers. This design facilitates the transition from the mathematical description to practical and efficient implementations.[1]

Let $x_0, \ldots, x_4$ represent the five 64-bit words of the round input. In the first step, a round constant is added to the least significant byte of x_2. Since the constant addition step is not important in this work, we simplify the notation by continuing to denote the output of the first step as $x_0, \ldots, x_4$.

The second step involves a non-linear transformation applied to on five bits, with one bit taken from each word of the first step's output. Let $y_0, \ldots, y_4$ represent the output state of the S-box, and let $\mathbf{1}$ (in bold) denote a word filled with 64 bit 1s. The algebraic normal form (ANF) of the S-box, with all operations performed on the full 64-bit words (in bitsliced form) can be expressed as:

$$\begin{aligned}
y_0 &= x_4x_1 \oplus x_3 \oplus x_2x_1 \oplus x_2 \oplus x_1x_0 \oplus x_1 \oplus x_0,\\
y_1 &= x_4 \oplus x_3x_2 \oplus x_3x_1 \oplus x_3 \oplus x_2x_1 \oplus x_2 \oplus x_1 \oplus x_0,\\
y_2 &= x_4x_3 \oplus x_4 \oplus x_2 \oplus x_1 \oplus \mathbf{1},\\
y_3 &= x_4x_0 \oplus x_4 \oplus x_3x_0 \oplus x_3 \oplus x_2 \oplus x_1 \oplus x_0,\\
y_4 &= x_4x_1 \oplus x_4 \oplus x_3 \oplus x_1x_0 \oplus x_1.
\end{aligned} \tag{1}$$

The third step, linear diffusion layer, applies a rotation to each word at the S-box output twice. The rotated words are then XOR-ed with the original one. Let $z_0, \ldots, z_4$ denote the output of the linear diffusion layer. The linear functions applied to each word are defined as:

$$\begin{aligned}
z_0 &= y_0 \oplus (y_0 \ggg 19) \oplus (y_0 \ggg 28),\\
z_1 &= y_1 \oplus (y_1 \ggg 61) \oplus (y_1 \ggg 39),\\
z_2 &= y_2 \oplus (y_2 \ggg \ \ 1) \oplus (y_2 \ggg \ \ 6),\\
z_3 &= y_3 \oplus (y_3 \ggg 10) \oplus (y_3 \ggg 17),\\
z_4 &= y_4 \oplus (y_4 \ggg \ \ 7) \oplus (y_4 \ggg 41).
\end{aligned} \tag{2}$$

At the beginning of the initialization (Figure 1), the 64-bit initialization vector $\texttt{IV}$ is stored in x_0. The two 64-bit halves of the key, $(k_0, k_1) = K$, are stored in x_1 and x_2. The two 64-bit halves of the nonce, $(n_0, n_1) = N$, are stored in x_3 and x_4. The S-box computation during the first round of the initialization phase can thus be written as follows:

$$\begin{aligned}
y_0 &= n_1k_0 \oplus n_0 \oplus k_1k_0 \oplus k_1 \oplus k_0\texttt{IV} \oplus k_0 \oplus \texttt{IV},\\
y_1 &= n_1 \oplus n_0k_1 \oplus n_0k_0 \oplus n_0 \oplus k_1k_0 \oplus k_1 \oplus k_0 \oplus \texttt{IV},\\
y_2 &= n_1n_0 \oplus n_1 \oplus k_1 \oplus k_0 \oplus \mathbf{1},\\
y_3 &= n_1\texttt{IV} \oplus n_1 \oplus n_0\texttt{IV} \oplus n_0 \oplus k_1 \oplus k_0 \oplus \texttt{IV},\\
y_4 &= n_1k_0 \oplus n_1 \oplus n_0 \oplus k_0\texttt{IV} \oplus k_0.
\end{aligned} \tag{3}$$

B. Statistical Ineffective Fault Attacks (SIFA)

Proposed by Dobraunig *et al.* [5], SIFA combines the ideas of Ineffective Fault Attacks (IFA) [12] and Statistical Fault Attacks (SFA) [13]. It exploits the distribution caused by ineffective faults, *i.e.*, those that do not affect the outcome of a computation. As the fault effect depends on the *intermediate value* at the chosen *attack point*, the distribution of this value is often biased. Relying on solely ineffective faults make SIFA applicable even in the presence of popular detection/infection countermeasures [5]. In this case, the countermeasures can be seen as a *filter* for the ineffective faults.

The principle of SIFA is as follows. First, the attacker chooses an intermediate value as the attack point such that its calculation depends on parts of the key. Second, he forces the values at the attack point to follow a non-uniform distribution by fault inductions. In practice, these

[1]Implementations for 8-bit, 32-bit, 64-bit architectures can be found at https://github.com/ascon/ascon-c

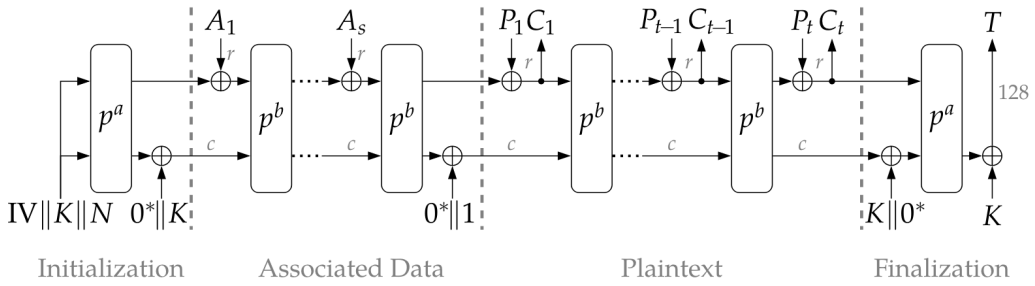
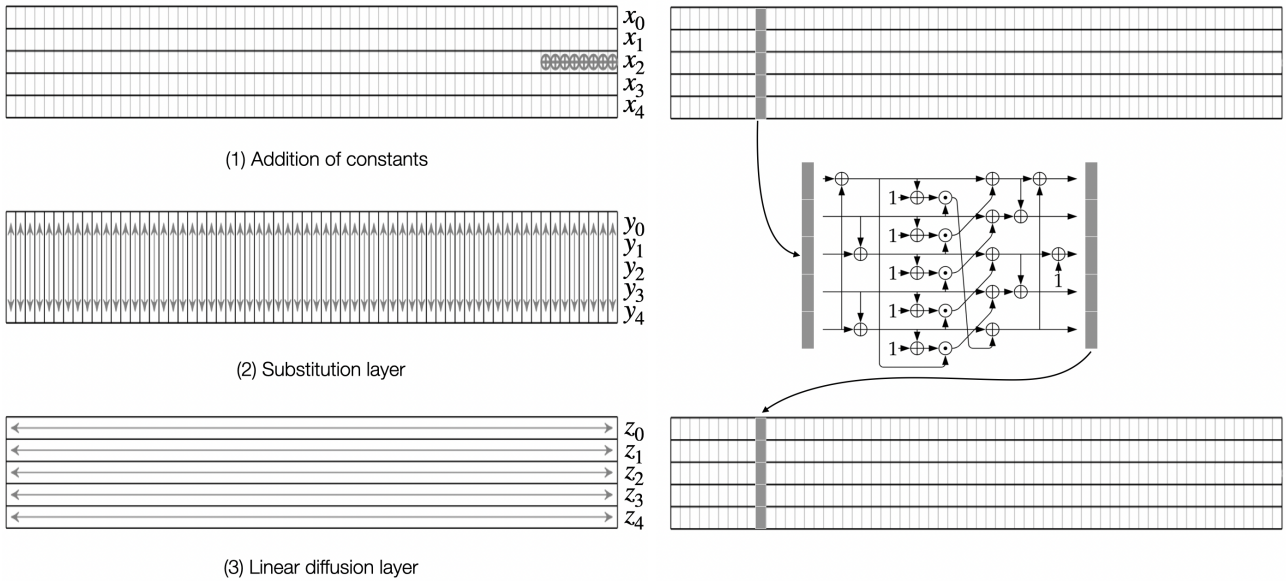


Figure 1: Encryption in ASCON [10].



(a) Three steps of a round

(b) An S-box computation.

Figure 2: Each step in a round [10].

faults can be achieved with an instruction skip using clock glitches [5], [6] or using laser [14]. Third, he collects a set of plaintexts/ciphertexts corresponding to the ineffective faults. Finally, key recovery is performed using this set. For each key guess, the attacker computes the intermediate value using the collected plaintexts/ciphertexts. This value should follow an (unknown) non-uniform distribution when the key is correctly guessed. In contrast, the distribution of the calculated values is expected to be close to uniform for any wrong key guesses. Hence, the key guess corresponding to the calculated distribution that is "furthest" from uniformity can be identified as the correct key.

C. SIFA on Nonce-based Authenticated Encryption

As this work investigate more thoroughly the extension of SIFA to nonce-based authenticated encryption schemes by Dobraunig *et al.* [4], we now recap the principle of their attack strategy. The authors focus on the initialization phase of the authenticated decryption, where the nonce N and the key K are processed. The verification of the tag T in the decryption is considered as the filter for ineffective faults. The state before the application of the S-box in the 2nd round of the initialization is chosen as the attack point. The distribution of one or multiple bits (intermediate value) at this state is assumed to be non-uniform for the filtered computations. A sufficient number of nonces corresponding to ineffective faults is collected for key recovery.

In the demonstration on Keyak and Ketje, the authors recovered the key by exploiting the biased distribution of a single bit before the application of the S-box in the 2nd round. They also detailed the analysis of the involved key bits in the calculation of the targeted bit. Their experiment was conducted on a 8-bit Xmega 128D4 microprocessor. An instruction in the S-box computation of the first round was skipped by clock glitches. This leads to the bias in the distribution of the targeted bit.

III. SIFA ON ASCON

In [4], the authors conjectured that their attack strategy can also be adopted to other schemes and ASCON is one of them. In this section, we detail the application of this attack strategy to ASCON before presenting our analysis in the next sections.

Following [4], a bit at the input of the S-box in the 2nd round of the initialization is chosen as the attack point. Note that, in ASCON, this is the same as choosing the first round output as the attack point because the step of constant addition does not change the distribution of the targeted bit. Thus, the attacker can choose one of the following as the intermediate value: $z_0^j, z_1^j, z_2^j, z_3^j, z_4^j$, where the superscript $j \in [0, 63]$ denotes the j-th bit in a word (the rightmost bit is at index 0). The computation of each of these bits involves three S-box applications as the effect of the linear diffusion layer.

We provide hereafter a detailed analysis for a bit of the first word, z_0^j. A similar analysis is applicable for a bit of other words, $z_1^j, z_2^j, z_3^j, z_4^j$. Figure 3 depicts the involved bits in the computation $z_0^j = y_0^j \oplus y_0^{j+19} \oplus y_0^{j+28}$, where y_0 is computed as in Equation 3 (the additions in the superscripts are implicitly taken modulo 64). Generally, the computation of z_0^j involves the following bits:

- 6 bits of the key, including 3 bits from the first key half $(k_0^j, k_0^{j+19}, k_0^{j+28})$ and 3 bits from the second key half $(k_1^j, k_1^{j+19}, k_1^{j+28})$,
- 6 bits of the nonce, including 3 bits from the first nonce half $(n_0^j, n_0^{j+19}, n_0^{j+28})$ and 3 bits from second nonce half $(n_1^j, n_1^{j+19}, n_1^{j+28})$,
- 3 constant bits of the initialization vector `IV`.

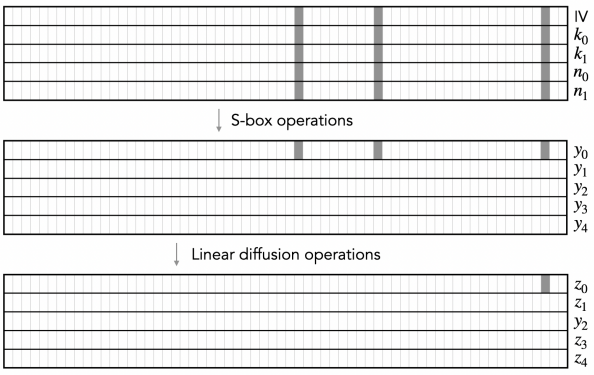


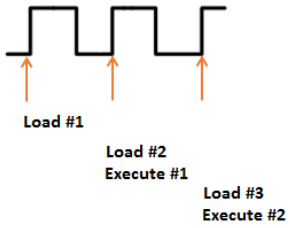
Figure 3: Involved bits in the computation of z_0^j.

We observe that the key bits only have linear influence (XOR operations) on the computations of z_2^j and z_3^j (see Equation 3 and Equation 2). As a consequence, the intermediate value (z_2^j or z_3^j) will have the same distribution for all the key candidates in the key recovery (not considering its sign). For this reason, the correct key guess cannot be distinguished if one of these bits is chosen as the attack point.

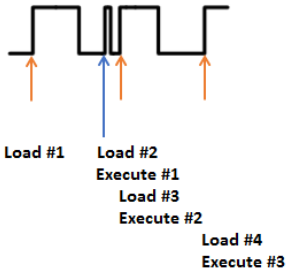
Meanwhile, the key bits influence the computations of z_0^j, z_1^j and z_4^j in a non-linear manner (there are AND operations between the nonce and the key). Therefore, the correct key and the wrong key guesses can be distinguished if the attacker chooses one of these bits as the attack point. We will thus focus on the computations of z_0^j, z_1^j and z_4^j in our analyzes in the next sections.

IV. INEFFECTIVE FAULTS WITH INSTRUCTION SKIP

In the context of SIFA, a commonly used method to induce a fault in practice is to skip an instruction. This skipped instruction is usually involved in the S-box computation, as demonstrated in prior works [5], [6], [4]. Figure 4 provide an illustrative explanation of how a clock glitch can cause an instruction to be skipped in a pipelined system. The glitch inserts a fast clock cycle between two ordinary clock cycles. As a consequence, Execution #1 is skipped since the system does not have enough time to perform the instruction.



(a) Normal clock.



(b) Glitched clock.

Figure 4: Example of instruction skip by a clock glitch in a pipelined system. Figure taken from https://github.com/newaetech/chipwhisperer-jupyter

In this section, we analyze the feasibility of obtaining an ineffective fault via an instruction skip in a w-bit architecture, where $w \in \{8, 32, 64\}$. More specifically, we consider the skipped instruction to be one of the following operations: `XOR`, `AND` and `NOT`, which are core components of authenticated encryption schemes such as ASCON and those based on Keccak permutation. We then validate our analysis with a case study on ASCON.

A. Main idea

In the following, we analyze the effect of skipping each of XOR, AND and NOT instructions. For each case, we assess the practicality of SIFA by computing the probability that a fault is ineffective.

1) Skipping a XOR*:* We first generalize the skipped instruction into two scenarios: one where the source registers are different from the destination register, and another where one of the source registers is also the destination register. These scenarios commonly appear in ARM architectures. Additionally, the latter scenario captures the 2-operand instruction format found in architectures such as AVR. The skipped instruction is highlighted in red as follows:

$\texttt{OP}_0$	$\texttt{R}_2$	–	–	$\texttt{OP}_0$	$\texttt{R}_2$	–	–
	...				...		
XOR	$\texttt{R}_2$	$\texttt{R}_1$	$\texttt{R}_0$	XOR	$\texttt{R}_2$	$\texttt{R}_2$	$\texttt{R}_0$
	...				...		
$\texttt{OP}_2$	–	$\texttt{R}_2$	–	$\texttt{OP}_2$	–	$\texttt{R}_2$	–
Scenario 1 : $\texttt{R}_2 = \texttt{R}_1 \oplus \texttt{R}_0$				Scenario 2 : $\texttt{R}_2 = \texttt{R}_2 \oplus \texttt{R}_0$			

In the above illustration, $\texttt{OP}_0$, and $\texttt{OP}_2$ represent operations, $\texttt{R}_1$ and $\texttt{R}_2$ denote w-bit registers, and $\texttt{R}_0$ represents either a w-bit register or an immediate value. Registers or immediates that are not relevant to our analysis are denoted by "–" for improved readability.

We focus on the value of the destination register $\texttt{R}_2$ in $\texttt{OP}_0$ and XOR. Let v_2 denote the value of $\texttt{R}_2$ after the execution of $\texttt{OP}_0$. Under normal conditions (i.e., without fault injection), this register is subsequently updated by the execution of XOR, resulting in a new value v_2'. However, if XOR is skipped due a to a fault, the value of $\texttt{R}_2$ remains unchanged at v_2. Consequently, the instruction $\texttt{OP}_2$ operates on a faulty value of $\texttt{R}_2$.

Let v_0 and v_1 denote the value of $\texttt{R}_0$ and $\texttt{R}_1$ before the execution of XOR. We have

$$v_2' = v_1 \oplus v_0$$

in the first scenario and

$$v_2' = v_2 \oplus v_0$$

in the second scenario. We make the following assumptions:

(a) v_0, v_1 and v_2 are uniformly distributed and independent. This typically occurs when these values originate from, or are derived from, random inputs (*i.e.*, nonces), which is usually the case in practical SIFA scenarios. As a result, v_2' is uniformly distributed and independent of v_2.

(b) When a fault occurs (*i.e.*, the XOR instruction is skipped), it is considered *ineffective* if $v_2 = v_2'$. Otherwise, the fault is referred to as *effective*.

As v_2 and v_2' are independent and uniformly distributed, the probability of an ineffective fault is:

$$\Pr[v_2 = v_2'] = 2^{-w},$$

where $w \in \{8, 16, 32\}$ denotes register bit-width of the hardware architecture. For 32-bit and 64-bit architectures, $\Pr[v_2 = v_2'] \approx 0$, meaning that it is likely impractical to obtain an ineffective fault. In contrast, $\Pr[v_2 = v_2'] = 2^{-8}$ for 8-bit architecture, implying that an ineffective fault can be expected once every 2^8 executions with random inputs. This shows that performing SIFA by skipping a XOR instruction is likely infeasible for 32-bit and 64-bit architectures, as the probability of achieving an ineffective fault is too low to be practical.

2) Skipping an AND*:* Similar to before, we first generalize the skipped instruction into two scenarios. In the case of a XOR instruction, the analysis is the same for both scenarios. However, for the AND instruction considered here, the analysis differs between the two.

$\texttt{OP}_0$	$\texttt{R}_2$	–	–	$\texttt{OP}_0$	$\texttt{R}_2$	–	–
	...				...		
AND	$\texttt{R}_2$	$\texttt{R}_1$	$\texttt{R}_0$	AND	$\texttt{R}_2$	$\texttt{R}_2$	$\texttt{R}_0$
	...				...		
$\texttt{OP}_2$	–	$\texttt{R}_2$	–	$\texttt{OP}_2$	–	$\texttt{R}_2$	–
Scenario 1 : $\texttt{R}_2 = \texttt{R}_1 \wedge \texttt{R}_0$				Scenario 2 : $\texttt{R}_2 = \texttt{R}_2 \wedge \texttt{R}_0$			

Using the same notations as before, let v_0, v_1 and v_2 denote the values of registers $\texttt{R}_0$, $\texttt{R}_1$, and $\texttt{R}_2$, respectively, prior to the execution of AND. Let v_2' represent the value of $\texttt{R}_2$ after the execution of AND under normal conditions (*i.e.*, without fault injection). If the AND instruction is skipped, the value of $\texttt{R}_2$ remains unchanged at v_2. We make the following assumptions:

(a) v_0, v_1 and v_2 are independent and uniformly distributed.

(b) When a fault occurs (*i.e.*, the AND instruction is skipped), it is considered *ineffective* if $v_2 = v_2'$. Otherwise, the fault is referred to as *effective*.

Scenario 1. We have $v_2' = v_1 \wedge v_0$. For simplicity, we consider 1-bit values for v_0, v_1, v_2 and v_2'. Table I presents the truth table of the AND operation. Since v_2 is uniformly distributed, it takes the value 0 with probability 0.5 and the value 1 with probability 0.5. When $v_2 = 0$, the probability that the fault is ineffective equals the probability that $v_2' = 0$, which is 0.75. Conversely, when $v_2 = 1$, the probability that the fault is ineffective equals the probability that $v_2' = 1$, which is 0.25. Therefore, the total probability of an ineffective fault is

$$\Pr[v_2 = v_2'] = 0.5 \times 0.75 + 0.5 \times 0.25 = 0.5.$$

When considering a w-bit architecture, the probability of an ineffective fault becomes

$$\Pr[v_2 = v_2'] = 2^{-w}.$$

This result is analogous to skipping a XOR instruction. For 32-bit and 64-bit architectures, the probability of achieving an ineffective fault is too low to be practical.

v_0	v_1	v_2'
0	0	0
0	1	0
1	0	0
1	1	1

Table I: Truth table of AND operation for scenario 1.

Scenario 2. We have $v_2' = v_2 \wedge v_0$. We consider the 1-bit truth table in Table II for this equation. It can be seen that the probability of a fault being ineffective is $\Pr[v_2 = v_2'] = 0.75$. When considering a w-bit architecture, this probability becomes

$$\Pr[v_2 = v_2'] = 0.75^w.$$

For $w = 32$ and $w = 64$, the values of $\Pr[v_2 = v_2']$ are approximately 10^{-4} and 10^{-8}, respectively. This means that it is more practical for a fault to be ineffective. Therefore, skipping an instruction of this type makes SIFA more applicable.

v_0	v_2	v_2'	$v_2 = v_2'$
0	0	0	✓
0	1	0	
1	0	0	✓
1	1	1	✓

Table II: Truth table of AND operation for scenario 2.

3) Skipping a NOT*:* As before, we first generalize the skipped instruction into two scenarios. We use the same notation for register values as above. We make the following assumptions:

(a) v_0 and v_2 are independent and uniformly distributed.
(b) When a fault occurs (*i.e.*, the NOT instruction is skipped), it is considered *ineffective* if $v_2 = v_2'$. Otherwise, the fault is referred to as *effective*.

```
OP0  R2  –  –          OP0  R2  –  –
     ...                    ...
NOT  R2  R0            NOT  R2  R2
     ...                    ...
OP2  –   R2 –          OP2  –   R2 –
Scenario 1 : R2 = ¬R0  Scenario 2 : R2 = ¬R2
```

Scenario 1. We have $v_2' = \neg v_0$. Since v_0 is uniformly distributed, v_2' is also uniformly distributed. Therefore, the probability of a fault being ineffective is

$$\Pr[v_2 = v_2'] = 2^{-w}.$$

This probability is similar to that of skipping a XOR instruction, meaning that it is likely impractical to obtain an ineffective fault on 32-bit and 64-bit architectures.

Scenario 2. We have $v_2' = \neg v_2$. In this case, an ineffective fault never occurs,

$$\Pr[v_2 = v_2'] = 0.$$

Therefore, if a NOT instruction corresponding to this scenario is skipped, the application of SIFA becomes infeasible.

B. Application to ASCON

To validate the above analysis, we simulate an instruction skip on the 8-bit ASCON implementation.[2] We use two instances of the encryption implementation, one as the reference and the other modified to skip an instruction. Specifically, an 8-bit instruction involved in the first round computation during the initialization is skipped. This instruction corresponds to one of the scenarios discussed above. Both instances are executed with the same random nonce (while the associated data and plaintext are set to empty strings, as they are not relevant to the attack). If the outputs produced by both instances are identical, the fault is considered ineffective.

Skipped Instruction	# Ineffective Faults	Empirical Probability	Theoretical Probability
XOR S1	81	0.0040	0.0039
XOR S2	79	0.0040	0.0039
AND S1	80	0.0040	0.0039
AND S2	2011	0.1006	0.1001
NOT S1	74	0.0037	0.0039
NOT S2	0	0	0

Table III: Empirical results from the instruction skip simulation. The encryption instances were executed 20000 times with random nonces.

Table III presents the results of our simulation. As expected, the empirical probabilities are close to the theoretical ones. This confirms the soundness of our analysis.

C. Discussion

The key point from our analysis is that, although SIFA is a powerful attack, its applicability depends on both how the fault is introduced and on the architecture of the target device. If an attacker blindly skips an instruction in the hope of causing an ineffective fault, the chances of success can be varied. Especially when skipping a XOR instruction in 32-bit and 64-bit architectures, obtaining an ineffective fault becomes highly impractical. In such cases, other fault injection techniques may be more efficient. For example, an attacker could use laser fault injection to force one or several bits in a register to be stuck at 0. In this case, only a few bits are affected, rather than the entire 32-bit or 64-bit register, increasing the probability of a fault being ineffective.

[2]The 8-bit implementation can be found at: https://github.com/ascon/ascon-c/tree/main/crypto_aead/asconaead128/bi8

In our analysis, we model instruction skip scenarios in authenticated encryption schemes like ASCON. However, those scenarios may not cover all practical cases. For example, in the second scenario involving a skipped NOT instruction, where $v_2' = \neg v_2$, we argue (under stated assumptions) that an ineffective fault never occurs. Now, suppose that v_2' is used in a subsequent operation, such as $u_c = v_2' \wedge v_0 = \neg v_2 \wedge v_0$. If the NOT is skipped, $\mathtt{R}_2$ retains the value v_2, and the operation becomes $u_f = v_2 \wedge v_0$. In this case, an ineffective fault occurs with the probability of 0.5, as shown in Table IV.

v_0	v_2	u_c	u_f	$u_c = u_f$
0	0	0	0	✓
0	1	0	0	✓
1	0	1	0	
1	1	0	1	

Table IV: Truth table of u_c and u_f.

V. Uniformity of intermediate value

In authenticated encryption schemes such as ASCON and those based on Keccak permutation, each round typically consists of two types of operations: one for diffusion and another for non-linearity (S-box). In the attack strategy proposed by Dobraunig *et al.* [4], a single bit of the state just before the S-box application in the 2nd round of initialization is chosen as the intermediate value. This bit generally results from a diffusion process at the end of the 1st round or at the beginning of the 2nd round. In other words, the intermediate value is the XOR of several S-box output bits from the 1st round.

Our key observation is as follows: if these S-box output bits are uniformly distributed and independently computed, and if at least one of the S-box computations is unaffected by the fault (*e.g.*, instruction skip), then the distribution of the intermediate value remains uniform. Consequently, SIFA is not applicable in this scenario.

In this section, we first formalize this property and provide a general proof. Then, we demonstrate its application in the context of ASCON.

A. Main idea

Let v be the intermediate value, and let $y^0, y^1, \ldots, y^{\ell-1}$ denote ℓ output bits from ℓ S-box computations involved in the XOR computation of v. We write:

$$v = y^0 \oplus y^1 \oplus \ldots \oplus y^{\ell-1}. \tag{4}$$

We denote $y^j = f(x_0^j, x_1^j, \ldots, x_{m-1}^j)$ the computation of the S-box output bit y^j, where $j \in [0, \ell-1]$ and $x_0^j, x_1^j, \ldots, x_{m-1}^j$ are m S-box input bits. We make the following assumptions:

(a) The S-box computations producing $y^0, y^1, \ldots, y^{\ell-1}$ are independent, which also implies that their input tuples $(x_0^0, x_1^0, \ldots, x_{m-1}^0)$, $(x_0^1, x_1^1, \ldots, x_{m-1}^1)$, …, $(x_0^{\ell-1}, x_1^{\ell-1}, \ldots, x_{m-1}^{\ell-1})$ are independent. This independence is usually the case when the S-box is implemented in bitsliced form, as in ASCON.
(b) Each y^j, for $0 \le j \le \ell-1$, is uniformly distributed.
(c) At least one of the S-box computations is unaffected by the fault (*e.g.*, instruction skip). Without loss of generality, we assume that the computation of y^0 is unaffected.

Theorem 1. *In the case of ineffective faults, the filtered values of v follow a uniform distribution.*

As a consequence of Theorem 1, SIFA is not applicable because the distribution of the intermediate value v remains unbiased. We now proceed to prove this theorem. If there is no filter for ineffective faults, the proof becomes trivial since the uniformity of y^0 (by assumption) directly implies the uniformity of the v. However, SIFA relies exclusively on ineffective faults, meaning that only (filtered) correct values of v are considered. This filtering makes the proof more complicated.

Here is the proof sketch. First, we define a Boolean variable ξ that indicates the effectiveness of the fault: $\xi = 1$ if the fault is effective, and $\xi = 0$ otherwise. Our goal is to show that the distribution of v remains uniform given that the fault is ineffective. Formally, we want to prove the following conditional probability:

$$\Pr[v = \mathtt{a} | \xi = 0] = 0.5, \tag{5}$$

where $\mathtt{a} \in \{0, 1\}$. Next, we demonstrate that ξ is independent of y^0. Since y^0 is uniformly distributed by assumption, this independence implies that its uniformity is preserved even when conditioning on $\xi = 0$. Finally, the uniformity of y^0 leads directly to the uniformity of v.

Lemma 1. *The variable ξ can be expressed as a Boolean function of the input bits to the S-box producing $y^1, \ldots, y^{\ell-1}$. These input bits are denoted by x_i^j, for $0 \le i \le m-1$ and $1 \le j \le \ell-1$.*

Proof of Lemma 1: We construct a truth table for $y^1, \ldots, y^{\ell-1}$, considering all possible values of the inputs x_i^j, for $0 \le i \le m-1$ and $1 \le j \le \ell-1$. We use two copies of this table: the first serves as a reference, while in the second, we re-evaluate the values of $y^1, \ldots, y^{\ell-1}$, taking the fault effect into account. By comparing the two tables, we identify input values where the fault is ineffective. Example 1 provides a demonstration of this process.

We then construct the truth table for ξ, considering all possible values of x_i^j, for $0 \le i \le m-1$ and $1 \le j \le \ell-1$. If an input results in an ineffective fault, we set $\xi = 0$; otherwise, $\xi = 1$. From this truth table, the Boolean function representing ξ can be derived. ■

Example 1. We consider the function $y^1 = x_0^1 \oplus x_1^1 x_2^1$ ($\ell = 2$, $m = 3$). Let registers registers $\mathtt{R_0}$, $\mathtt{R_1}$, and $\mathtt{R_2}$ store the values x_0^1, x_1^1, and x_2^1, respectively. The value y^1 is computed using the following instructions:

```
AND  R1  R1  R2
XOR  R0  R0  R1
```

If the AND instruction is skipped, the computation becomes $\tilde{y}^1 = x_0^1 \oplus x_1^1$. The fault is consider ineffective if $y^1 = \tilde{y}^1$, as shown in Table V.

x_0^1	x_1^1	x_2^1	y^1	$\tilde{y}^1$	$y^1 = \tilde{y}^1$	ξ
0	0	0	0	0	✓	0
0	0	1	0	0	✓	0
0	1	0	0	1		1
0	1	1	1	1	✓	0
1	0	0	1	1	✓	0
1	0	1	1	1	✓	0
1	1	0	1	0		1
1	1	1	0	0	✓	0

Table V: Example of truth table construction for ξ.

Corollary 1. *The variable ξ is independent of the S-box output bit y^0.*

It is evident that ξ is independent of y^0, since the Boolean function representing ξ does not depend on the input bits of y^0, and the computation of y^0 is independent of those of $y^1, \ldots, y^{\ell-1}$.

Proof of Theorem 1: Let $\mathtt{a} \in \{0,1\}$. By assumption, $\Pr[y^0 = \mathtt{a}] = 1/2$. Since ξ and y^0 are independent (Corollary 1), it follows that

$$\Pr[y^0 = \mathtt{a} | \xi = 0] = \Pr[y^0 = \mathtt{a}] = 0.5.$$

This means that y^0 remains uniformly distributed even when conditioned on the fault being ineffective. Consequently, the uniformity of v directly follows from the uniformity of y^0. ■

B. Application to ASCON

In the context of SIFA on ASCON, following the attack strategy proposed in [4], a bit in the first round output of the initialization is chosen as the intermediate value (also corresponding to a bit in the state before the application of the S-box in the 2nd round). As discussed in Section III, we focus only on the bits z_0^j, z_1^j and z_4^j, as the key influences these bits in a non-linear manner. In this section, we show that the intermediate value (z_0^j, z_1^j or z_4^j) has uniform distribution even when conditioned on ineffective faults. As a consequence, SIFA is not applicable since the distribution of the intermediate value is unbiased. Our analysis considers an instruction skip as the fault in an 8-bit implementation.

To prove the uniformity of the intermediate value, we show that the 8-bit implementation of ASCON satisfies the three assumptions for Theorem 1. We present here a detailed analysis for z_0^j, a similar analysis can be straightforwardly applied to z_1^j and z_4^j. Recall that the output bits y_0^j, y_0^{j+19} and y_0^{j+28}, each resulting from separate S-box computations, are involved in the XOR computation of z_0^j, as presented in Section III. Thus, Equation 4 becomes

$$z_0^j = y_0^j \oplus y_0^{j+19} \oplus y_0^{j+28}.$$

First, it is straightforward to observe that the S-box computations producing y_0^j, y_0^{j+19} and y_0^{j+28} are independent, due to the bitsliced design of ASCON. Hence, the first assumption is satisfied.

Second, we show that each of y_0^j, y_0^{j+19} and y_0^{j+28} is uniformly distributed under fault-free conditions. As the S-box computation is identical for these three bits, it suffices to examine the distribution of one of them, say y_0^j. Recall that the computation of y_0^j is given by:

$$y_0^j = n_1^j k_0^j \oplus n_0^j \oplus k_1^j k_0^j \oplus k_1^j \oplus k_0^j \mathtt{IV}^j \oplus k_0^j \oplus \mathtt{IV}^j$$

Note that in a SIFA attack, the key is fixed on the device. Therefore, we must verify that, for each fixed key (k_0^j, k_1^j) and each constant bit of the initialization vector ($\mathtt{IV}^j$), the distribution of y_0^j over all possible values of the nonce bits (n_0^j, n_1^j) is uniform. When $\mathtt{IV}^j = 0$, the evaluation of y_0^j simplifies to:

$$y_0^j = \begin{cases} n_0^j & \text{when } (k_0^j, k_1^j) = (0,0), \\ n_0^j \oplus 1 & \text{when } (k_0^j, k_1^j) = (0,1), \\ n_1^j \oplus n_0^j \oplus 1 & \text{when } (k_0^j, k_1^j) = (1,0), \\ n_1^j \oplus n_0^j \oplus 1 & \text{when } (k_0^j, k_1^j) = (1,1), \end{cases}$$

and when $\mathtt{IV}^j = 1$, it becomes:

$$y_0^j = \begin{cases} n_0^j \oplus 1 & \text{when } (k_0^j, k_1^j) = (0,0), \\ n_0^j & \text{when } (k_0^j, k_1^j) = (0,1), \\ n_1^j \oplus n_0^j & \text{when } (k_0^j, k_1^j) = (1,0), \\ n_1^j \oplus n_0^j \oplus 1 & \text{when } (k_0^j, k_1^j) = (1,1), \end{cases}$$

Since n_0^j and n_1^j are uniformly distributed, it follows that y_0^j is also uniformly distributed. Hence, the second assumption is satisfied.

Third, we show that at least one of the computations of y_0^j, y_0^{j+19} and y_0^{j+28} is unaffected by an instruction skip (*i.e.*, the fault). Recall that we exclusively focus on the 8-bit implementation. To establish this, we demonstrate that the bitsliced S-box computation never processes the data involved in the computations of y_0^j, y_0^{j+19} and y_0^{j+28} within a single instruction. In other words, the bits at the three indices $(j, j+19, j+28)$ of a word never appear in the same 8-bit block (byte/register) in the 8-bit implementation. We verify this for two practical bit arrangements used in ASCON, with

and without the interleaving technique, as shown in Table VI and Table VII. Indeed, for every $j \in [0, 63]$, the three indices never fall within the same 8-bit block. Therefore, skipping an instruction does not affect all three computations of y_0^j, y_0^{j+19} and y_0^{j+28}. Thus, the third assumption is satisfied.

Byte 7	63	62	61	60	59	58	57	56
Byte 6	55	54	53	52	51	50	49	48
Byte 5	47	46	45	44	43	42	41	40
Byte 4	39	38	37	36	35	34	33	32
Byte 3	31	30	29	28	27	26	25	24
Byte 2	23	22	21	20	19	18	17	16
Byte 1	15	14	13	12	11	10	9	8
Byte 0	7	6	5	4	3	2	1	0

Table VI: Arrangement of 64-bit word into bytes. Positions of indices 0, 19 and 28 are highlighted in red.

Byte 7	63	55	47	39	31	23	15	7
Byte 6	62	54	46	38	30	22	14	6
Byte 5	61	53	45	37	29	21	13	5
Byte 4	60	52	44	36	28	20	12	4
Byte 3	59	51	43	35	27	19	11	3
Byte 2	58	50	42	34	26	18	10	2
Byte 1	57	49	41	33	25	17	9	1
Byte 0	56	48	40	32	24	16	8	0

Table VII: Arrangement of 64-bit word into bytes using interleaving technique. Positions of indices 0, 19 and 28 are highlighted in red.

To empirically validate our analysis, we compute the probabilities of the intermediate value z_0^j taking values 0 and 1, based on the ineffective faults simulated in the previous section. The results are shown in Table VIII. As observed, the empirical probabilities have slight deviations from uniformity. We expect these probabilities to converge more closely to uniform as more ineffective faults are collected.

Skipped Instruction	# Ineff. Faults	Emp. Prob. (Pr[0] / Pr[1])	Theo. Prob. (Pr[0] / Pr[1])
XOR S1	81	0.54 / 0.46	0.5 / 0.5
XOR S2	79	0.44 / 0.56	0.5 / 0.5
AND S1	80	0.53 / 0.47	0.5 / 0.5
AND S2	2011	0.52 / 0.48	0.5 / 0.5
NOT S1	74	0.42 / 0.58	0.5 / 0.5
NOT S2	0	- / -	- / -

Table VIII: Simulation for uniformity of the intermediate value. The encryption instances were executed 20000 times with random nonces.

VI. Conclusion

In this work, we provide a more in-depth analysis of the SIFA-based attack strategy on authenticated encryption schemes proposed by Dobraunig *et al.* [4], with a particular focus on instruction skip as the fault method. We show that the probability of achieving an ineffective fault depends on the type of instruction and the device architecture. Notably, skipping a XOR instruction in a 32-bit and 64-bit architectures is unlikely to result in an ineffective fault. Furthermore, we formally show that the intermediate value at the attack point can unexpectedly remain unbiased under ineffective faults. Both findings highlight the cases where SIFA becomes inapplicable.

A limitation of this work is that the modeling of instruction skip scenarios in Section IV may not be exhaustive. Practical implementations might use other instructions, such as OR or BIC in ARM. We leave the exploration of these cases for future work.

Our findings also open up a research question: *Can we leverage insights from the uniformity of the intermediate value, as analyzed in Section V, to design a generic implementation method that is resistant to SIFA?*

References

[1] D. Boneh, R. A. DeMillo, and R. J. Lipton, "On the importance of checking cryptographic protocols for faults (extended abstract)," in *EUROCRYPT'97*, ser. LNCS, W. Fumy, Ed., vol. 1233. Springer, Berlin, Heidelberg, May 1997, pp. 37–51.

[2] E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems," in *CRYPTO'97*, ser. LNCS, B. S. Kaliski, Jr., Ed., vol. 1294. Springer, Berlin, Heidelberg, Aug. 1997, pp. 513–525.

[3] D. Saha and D. R. Chowdhury, "EnCounter: On breaking the nonce barrier in differential fault analysis with a case-study on PAEQ," in *CHES 2016*, ser. LNCS, B. Gierlichs and A. Y. Poschmann, Eds., vol. 9813. Springer, Berlin, Heidelberg, Aug. 2016, pp. 581–601.

[4] C. Dobraunig, S. Mangard, F. Mendel, and R. Primas, "Fault attacks on nonce-based authenticated encryption: Application to Keyak and Ketje," in *SAC 2018*, ser. LNCS, C. Cid and M. J. Jacobson, Jr., Eds., vol. 11349. Springer, Cham, Aug. 2019, pp. 257–277.

[5] C. Dobraunig, M. Eichlseder, T. Korak, S. Mangard, F. Mendel, and R. Primas, "SIFA: Exploiting ineffective fault inductions on symmetric cryptography," *IACR TCHES*, vol. 2018, no. 3, pp. 547–572, 2018. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/7286

[6] C. Dobraunig, M. Eichlseder, H. Groß, S. Mangard, F. Mendel, and R. Primas, "Statistical ineffective fault attacks on masked AES with fault countermeasures," in *ASIACRYPT 2018, Part II*, ser. LNCS, T. Peyrin and S. Galbraith, Eds., vol. 11273. Springer, Cham, Dec. 2018, pp. 315–342.

[7] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche, "The Keccak SHA-3 submission," https://keccak.team/files/Keccak-submission-3.pdf, 2011.

[8] G. Bertoni, J. Daemen, M. Peeters, G. V. Assche, and R. V. Keer, "CAESAR submission: Keyak v2," https://keccak.team/files/Keyakv2-doc2.2.pdf, 2016.

[9] ——, "CAESAR submission: Ketje v2," https://keccak.team/files/Ketjev2-doc2.0.pdf, 2016.

[10] C. Dobraunig, M. Eichlseder, F. Mendel, and M. Schläffer, "Ascon v1.2: Lightweight authenticated encryption and hashing," *Journal of Cryptology*, vol. 34, no. 3, p. 33, Jul. 2021.

[11] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, "Duplexing the sponge: Single-pass authenticated encryption and other applications," in *SAC 2011*, ser. LNCS, A. Miri and S. Vaudenay, Eds., vol. 7118. Springer, Berlin, Heidelberg, Aug. 2012, pp. 320–337.

[12] C. Clavier, "Secret external encodings do not prevent transient fault analysis," in *CHES 2007*, ser. LNCS, P. Paillier and I. Verbauwhede, Eds., vol. 4727. Springer, Berlin, Heidelberg, Sep. 2007, pp. 181–194.

[13] T. Fuhr, É. Jaulmes, V. Lomné, and A. Thillard, "Fault attacks on AES with faulty ciphertexts only," in *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography, Los Alamitos, CA, USA, August 20, 2013*, W. Fischer and J. Schmidt, Eds. IEEE Computer Society, 2013, pp. 108–118. [Online]. Available: https://doi.org/10.1109/FDTC.2013.18

[14] C. Dobraunig, M. Eichlseder, T. Korak, V. Lomné, and F. Mendel, "Statistical fault attacks on nonce-based authenticated encryption schemes," in *ASIACRYPT 2016, Part I*, ser. LNCS, J. H. Cheon and T. Takagi, Eds., vol. 10031. Springer, Berlin, Heidelberg, Dec. 2016, pp. 369–395.