Picking up the Fallen Mask: Breaking and Fixing the RS-Mask Countermeasure

Dilara Toprakhisar[1][0000−0003−4551−6775], Svetla Nikova[1][0000−0003−3133−9261], and Ventzislav Nikov[2]

[1] COSIC, KU Leuven, Leuven, Belgium
{dilara.toprakhisar,svetla.nikova}@esat.kuleuven.be
[2] NXP Semiconductors, Leuven, Belgium
venci.nikov@gmail.com

Abstract. Physical attacks pose a major challenge to the secure implementation of cryptographic algorithms. Although significant progress has been made in countering passive attacks such as side-channel analysis (SCA), protection against fault attacks is still less developed. One reason for this is the broader and more complex nature of fault attacks, which makes it difficult to create standardized fault evaluation methodologies for countermeasures like those used for SCA. This makes it easier to overlook potential vulnerabilities that attackers could exploit. RS-Mask, published at HOST 2020, is such a countermeasure that has been affected by the absence of a systematic analysis method. The fundamental concept behind the countermeasure is to maintain a uniform distribution of variables, regardless of whether they are faulty or correct. This property is particularly effective against Statistical Ineffective Fault Attacks (SIFA), which exploit the dependency between fault propagation and the secret data.

In this work, we present several fault scenarios involving single fault injections on the AES implementation protected with RS-Mask, where the fault propagation depends on the secret data. This happens because the random space mapping used in RS-Mask countermeasure retains a dependency on the secret data, as it is derived based on the S-box input. To address this, we propose a new countermeasure based on the core concept of RS-Mask, implementing a single mapping for all S-box inputs, involving an intrinsic duplication. Next, we evaluate the effectiveness of the new countermeasure against fault attacks by comparing the fault detection rate across all possible fault locations and values for every input. Additionally, we examine the output differences between faulty and correct outputs for each input. Our results show that the detection rate is uniform for each input, which ensures security against statistical attacks utilizing both effective and ineffective faults. Moreover, the output differences being uniform for each input ensures security against differential fault attacks.

Keywords: Fault Attacks · SIFA · AES.

1 Introduction

Cryptographic algorithms are designed to be secure against cryptanalytic attacks; however, their physical implementations in embedded devices remain vulnerable to physical attacks. These attacks exploit the physical characteristics of the cryptographic implementations and can be broadly classified into two categories: passive attacks, which involve observing the device's behavior (*e.g.*, power consumption [22], timing [21], and electromagnetic emanation [15]), and active attacks, which induce errors in the computation through physical manipulations (*e.g.*, clock/voltage glitching [2], electromagnetic interference [9], and laser injections [17]).

Side-channel analysis (SCA), being a passive attack technique, leverages the information leakage arising from physical characteristics of the implementations such as timing and power consumption. To protect against SCA, masking [5, 20, 27, 16] stands out as the most prominent countermeasure. Masking relies on splitting the secret input into a number of statistically independent shares, ensuring that even observing all but one share does not compromise the secret input. Notable approaches include, among others, Domain Oriented Masking (DOM) [16], which associates each share of the secret input with a distinct domain, and Threshold Implementations (TI) [24], which computes over coordinate functions operating on non-complete sets of secret input shares.

Differing from the passive observations exploited in SCA, fault attacks actively disrupt computations by injecting faults through physical fault injection mechanisms. Since Boneh *et al.* [3] first demonstrated fault attacks on RSA, numerous attack techniques have emerged to exploit injected faults in cryptographic implementations. These attacks exploit the data-dependent response to these injected faults. To protect against these attacks, redundancy is widely employed as a countermeasure, by employing temporal, spatial, or informational replication to detect whether a fault is injected or not. Upon a fault detection, redundancy-based techniques either suppress the output or infect it to prevent the adversary from extracting secret information. This approach has been further leveraged to enable error correction. Recent countermeasures combine redundancy with masking to provide simultaneous protection against both SCA and fault attacks, with masking also serving to enhance the protection against fault attacks. Such countermeasures were generally regarded as effective in protecting against fault attacks. However, Statistical Ineffective Fault Attacks (SIFA) [13], introduced in 2018, exploit data dependent fault propagation by collecting non-faulty ciphertexts, rendering redundancy vulnerable to such attacks. Building on this attack, Dobraunig *et al.* [12] utilized SIFA to overcome simple combination of masking with redundancy by faulting non-linear operations. Notable examples of such combined countermeasures include, but are not limited to, ParTI [31], CAPA [28], Private Circuits II [19], M&M [23], Impeccable Circuits [1, 32, 26], and Transform-and-Encode (TaE) [29], which were developed both before and after the introduction of SIFA. All these countermeasures employ redundancy in various forms (*e.g.,* linear codes, MAC tags) with different levels of error detection/correction granularity. Taking a different approach, Daemen *et al.* [8]

proposed the use of reversible operations, Toffoli gates, to ensure fault propagation independent of the secret data, with a single error check at the end of the encryption. Along the similar lines, the StaTI countermeasure [10] introduced the notion of "stability", a composable security property that ensures fault propagation independent of the secret data, enabling a single error check at the end of the encryption. Building on this concept, StaMAC [11] adopts the stability notion within MAC tags. Another countermeasure, Random Space Masking (RS-Mask) [25] that is published at HOST 2020, performs the S-box computations in a random space, eliminating any bias introduced by injected faults to the computations.

Despite these advancements in fault countermeasures, fault attacks still present a more complex and diverse attack surface compared to passive attacks, making the development of standardized fault evaluation methodologies challenging. The lack of systematic fault evaluation methodologies increases the risk of overlooking exploitable vulnerabilities in the countermeasures. Similarly, the RS-Mask countermeasure, while conceptually shows promise, particularly against statistical (ineffective) fault attacks, its fault evaluation is not comprehensive, leaving several attack scenarios unaddressed.

Contributions. In this work, we address existing gaps in the fault evaluation of RS-Mask and uncover previously overlooked vulnerabilities through specific fault scenarios involving single fault injections in the protected AES implementation. We demonstrate that RS-Mask's random space mapping retains a dependency on the secret data, therefore, failing to protect against statistical (ineffective) fault attacks, as well as differential fault attacks. These findings underscore the importance of thorough and systematic fault evaluations.

The identified vulnerabilities in RS-Mask stem from the use of distinct mappings derived for the S-box inputs (zero input vs. non-zero inputs), which are combined using an auxiliary variable. To address this issue, we propose an improved countermeasure that incorporates a single random space mapping for all S-box inputs. Our evaluation demonstrates that this approach achieves a uniform fault detection rate across all S-box inputs, ensuring robust protection against both statistical and differential fault attacks through intrinsic duplication. We additionally evaluate the side-channel security, and present hardware benchmark results.

Outline. In Section 2, we discuss the adversary and security models that are considered in this work, and provide an overview of the RS-Mask countermeasure. In Section 3, we discuss the fault security of the RS-Mask countermeasure, and outline the fault scenarios against the protected AES S-box implementation. Next, we describe our proposed countermeasure involving the random space masking specifically derived for the AES S-box in Section 4, and present its security and performance evaluation.

2 Preliminaries

In this section, we introduce the probing and the gate/register-faulting adversaries along with their respective security models assumed in this work. We then provide an overview of the RS-Mask countermeasure [25] and its hardware implementation details of the protected AES as described by the authors.

2.1 Adversary and Security Models

We consider an adversary with probing and faulting capabilities, assuming that these capabilities are not used in combination. The attack surface is considered as a Boolean circuit, a directed acyclic graph where the vertices correspond to Boolean gates, and the edges correspond to the wires. This circuit takes an input and calculates an output while having an internal state corresponding to the sharing of the secret data stored in the registers.

Probing capabilities. We assume the d-probing model introduced by Ishai *et al.* [20] to capture an attacker with probing capabilities. In this model, the adversary is restricted to observing at most d wires of the Boolean circuit, with the selected wires being specified before the circuit is executed. Additionally, we assume the glitch-extended robust probing model introduced by Faust *et al.* [14] to account for the physical effects of hardware glitches.

Faulting capabilities. We rely on the gate/register-faulting model [10] to capture an attacker with faulting capabilities. In this model, the adversary can replace an upper bounded number of gates or registers to output zero (through a reset fault), output one (through a set fault), or flip the output (through a bitflip fault). In this work, we consider a single gate/register-faulting adversary, where the adversary is capable of replacing one gate or register in the circuit throughout the whole computation, consistent with the single-fault attacks assumed in the RS-Mask countermeasure.

We adopt a more relaxed security model compared to the security guarantees associated with the gate/register-faulting model described in [10]. In our correctness model, we introduce a relaxation that allows the adversary to receive an incorrect output. However, this output remains *unexploitable* for all possible single fault injections. Specifically, the difference between the outputs of the faulty and correct circuits, given the same inputs, is uniformly distributed and independent of both the injected fault and the secret inputs. Moreover, the fault detection rate across all input values is also uniformly distributed. In our privacy model, we assume the same model described in [10]. Specifically, for every fixed injected fault, the probability of the abort signal is independent of the secret inputs of the circuit.

2.2 RS-Mask: Random Space Masking as an Integrated Countermeasure against Power and Fault Analysis

The RS-Mask countermeasure [25] ensures that intermediate variables maintain a uniform distribution, even in the presence of fault injection. This uniformity protects implementations against statistical fault attack techniques such as SIFA, which exploit the bias in the distribution of intermediate variables. The core idea of the countermeasure is to map the intermediate variables into a random space, perform operations within this random space, and then transform the cipher output back to the original space. By employing uniform random mapping, the operations performed on the intermediate variables remain independent of the secret data.

The linear operations of the cipher are inherently protected through masking, requiring no additional protective measures. The S-box computations, however, are performed in a random space where intermediate variables maintain a uniform distribution. In the RS-Mask paper, the random space masking is designed such that the S-box outputs the correct value XORed with a uniform random value, neutralizing any bias induced by fault injection. The faults injected to the intermediate variables in the random space are influenced by the random values, ensuring that their propagation remains independent of the secret data.

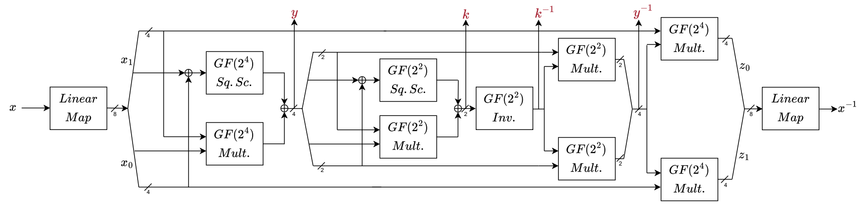


Fig. 1: Canright implementation of the $\mathbb{F}_{2^8}$ inverter in the AES S-box.

Derivation of RS mapping for AES. The mapping derived for the $\mathbb{F}_{2^8}$ inversion in the AES S-box ensures that, for a given input x, the output is $x^{-1} + r$, where r is a uniform random value. The mapping is specifically designed for the AES implementation based on the Canright construction [4], as depicted in Figure 1. In this construction, the input to the $\mathbb{F}_{2^8}$ inversion is denoted by $x = x_1 || x_0$, the input to the $\mathbb{F}_{2^4}$ inversion by y, the output of the $\mathbb{F}_{2^4}$ inversion by y^{-1}, and the output of the $\mathbb{F}_{2^8}$ inversion by $z = x^{-1} = z_1 || z_0$.

For inputs where $x \neq 0$, the following output is computed:

$$\begin{aligned} z'_1 &= z_1 + r_1 = (x_0 + y r_1) y^{-1} \\ z'_0 &= z_0 + r_0 = (x_1 + y r_0) y^{-1}\,, \end{aligned} \tag{1}$$

where $r = r_1 || r_0$ is the uniform random value masking the inversion output. To account for faults injected earlier in the inversion, the value y is derived along a separate data path from the input x, ensuring that y^{-1} does not correspond to the inverse of faulty y (or, conversely, that a faulty y^{-1} is not the inverse of correct y).

However, this approach does not apply to the case when $x = 0$, as both y and y^{-1} evaluate to zero. To handle the zero input case, the inverse of the random value r is XORed with the input x. This ensures that the output remains consistently masked with r, without relying on the calculations used for non-zero inputs (Equation 1):

$$z' = (0 + r^{-1})^{-1} = 0 + r\,, \tag{2}$$

thereby aligning the behavior with that of non-zero inputs. To derive a unified mapping for all inputs, an auxiliary variable $f \in \mathbb{F}_{2^4}$ is introduced, along with its bitwise negation $\bar{f}$:

$$f = \begin{cases} (1,1,1,1), & x \neq 0 \quad (\bar{f} = (0,0,0,0)) \\ (0,0,0,0), & x = 0 \quad (\bar{f} = (1,1,1,1))\,, \end{cases} \tag{3}$$

where $(1,1,1,1) = 15$ is the multiplicative unity in $\mathbb{F}_{2^4}$. Using this, the overall mapping is expressed as:

$$\begin{aligned} z'_1 &= z_1 + r_1 = (x_0 + r_0^{-1}\bar{f} + yr_1 f)y^{-1} \\ z'_0 &= z_0 + r_0 = (x_1 + r_1^{-1}\bar{f} + yr_0 f)y^{-1}\,, \end{aligned} \tag{4}$$

where $r^{-1} = r_1^{-1}||r_0^{-1}$.

Hardware implementation. The overall block diagram of the AES S-box implementation protected with RS-Mask countermeasure is presented in Figure 2. In this implementation, the AES state is split into three shares, where one of the shares is the RS share, the random value r. The round keys are split into two shares, with the RS share set to zero. After the linear map is applied to the S-box input x, the RS share is added to one of the two data shares. All non-linear components in the design are implemented using threshold implementations (TI) [24] with three shares as the masking technique.

As the random value r is independent of all other shares, its inverse r^{-1} is computed in an unshared form. To protect against the effects of hardware glitches, the RS-Mask paper specifies that the non-linear components of this inversion circuit are registered. In parallel to the computation of r^{-1}, the auxiliary variable $f \in \mathbb{F}_{2^4}$ is computed using the Kronecker delta function. To maintain throughput, the whole design is implemented as a 10-stage pipeline.

Although RS-Mask maps intermediate variables to a random space, the authors acknowledge that the distribution of the output differences resulting from the injected faults may still be non-uniform. This non-uniformity makes the implementation vulnerable against differential fault attacks (DFA). To address this vulnerability, the authors propose an infection mechanism as an additional protection layer against DFA. As described above, the variables x_0, x_1 and y^{-1}

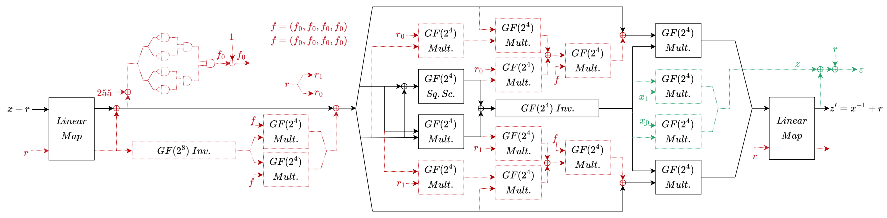


Fig. 2: AES S-box protected by RS-mask. Red components are the additional computations required for the RS mapping, green components are the additional components required for the infection mechanism.

are already available in the RS-Mask circuit. Using these, the output of the S-box inversion z can be recomputed with two additional $\mathbb{F}_{2^4}$ multipliers as shown in Figure 2.

If no error occurs during the inversion, the expression $\varepsilon = z + r + z'$ evaluates to zero (see Equation 4). Consequently, multiplying ε by a random value εr_1 (with r_1 being uniformly distributed) yields zero, and adding this to the S-box output has no effect. If an error occurs in the inversion circuit, then $\varepsilon \neq 0$, and the resulting product $\varepsilon r_1 \neq 0$ infects the S-box output.

However, this infection mechanism still does not protect against DFA, as an error in one byte propagates to the entire column with deterministic relations through the MixColumns operation. To mitigate this, the infection mechanism is extended by computing four random variables $\varepsilon_i = \varepsilon r_i$ for $i = 0, ..., 3$ using four uniformly random values r_i. ε_i is then added to the i-th byte of the respective column, randomizing all four bytes.

3 Security Analysis of RS-Mask

This section highlights the limitations of the AES S-box implementation protected using the RS-Mask countermeasure.

First of all, one notable consideration is the computation of r^{-1}, which includes additional register layers intended to mitigate the physical effects of hardware glitches. However, under the glitch-extended robust probing model introduced by Faust *et al.* [14], r, being a single share, can be directly provided to the simulator. As a result, these additional registers attempt to address an issue that is not relevant in this context, potentially introducing unnecessary implementation overhead.

Beyond the aforementioned overhead, another limitation of the RS-Mask countermeasure arises in its infection mechanism intended to protect against differential attacks. Specifically, the infection mechanism reuses the same S-box input x to recompute the expected output z, which is then compared with the actual output to generate the infection value εr_1. However, if a fault is injected to x, both the recomputed and actual outputs are affected similarly, causing ε, and thus the infection value, to evaluate to zero. As a result, the fault remains undetected and the state is uninfected. This allows DFA to still be successfully performed on the RS-Mask countermeasure by injecting a fault to the S-box input. We further explore this drawback in a fault scenario, which is elaborated upon later in this section.

Finally, in the rest of this section, we present several fault scenarios that demonstrate the vulnerability of the AES S-box implementation protected by RS-Mask countermeasure. Specifically, we identify and describe four groups of single fault locations that are vulnerable, as depicted in Figure 3.

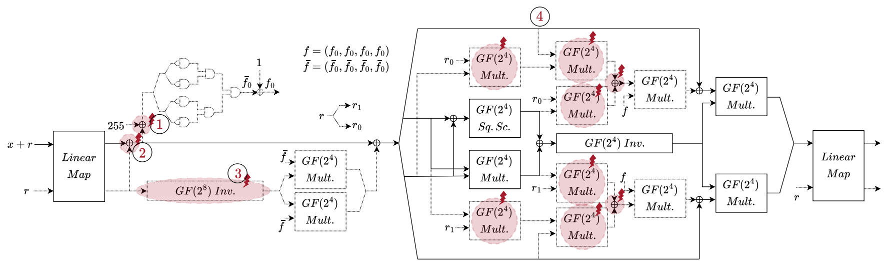


Fig. 3: Fault locations on AES S-box protected by RS-mask.

Fault location 1: Kronecker delta function. The Kronecker Delta function is evaluated at $x + 255$ to compute f. In one fault scenario, an attacker injects an additive fault Δ at the XOR operation that computes $x + 255$ (indicated as number 1 in Figure 3). As a result, the input to the Kronecker delta function becomes $x + 255 + \Delta$. The function then outputs $f = 0$ ($\bar{f} = 1$) if $x + \Delta = 0$ (*i.e.,* $x = \Delta$). Otherwise, if $x \neq \Delta$, the function outputs $f = 1$.

We analyze this fault scenario for three cases: $x = \Delta$, $x = 0$ and $x \neq \{0, \Delta\}$, where z' is the RS-Mask S-box output and z is the redundantly computed value used in the infection mechanism, as shown in Figure 2 (if no fault is injected $z' = z + r$).

First, $x = \Delta$ implies $f = 0$. Then, using Equation 4, we find:

$$\begin{aligned} z'_1 &= (\Delta_0 + r_0^{-1})y^{-1} \\ z'_0 &= (\Delta_1 + r_1^{-1})y^{-1}\,, \end{aligned}$$

where $z' = (\Delta + r^{-1})^{-1}$, and

$$\begin{aligned} z_1 &= \Delta_0 y^{-1} \\ z_0 &= \Delta_1 y^{-1}\,, \end{aligned}$$

where $z = \Delta^{-1}$. Then,

$$\begin{aligned} \varepsilon &= z + r + z' \\ &= \Delta^{-1} + r + (\Delta + r^{-1})^{-1}\,. \end{aligned}$$

If $r \neq 0$, then, $\varepsilon \neq 0$. Thus, the computation is infected.

Second, $x = 0$ implies $f = 1$ (and $y, y^{-1} = 0$). Then, using Equation 4, we find $z' = 0$ ($z'_1 = 0$, $z'_0 = 0$), and $z = 0$ ($z_1 = 0$, $z_0 = 0$). Then,

$$\varepsilon = z + r + z' = r\,.$$

If $r \neq 0$, then, $\varepsilon \neq 0$. Thus, the computation is infected.

Third, $x \neq \{0, \Delta\}$ implies $f = 1$. Then, using Equation 4, we find:

$$\begin{aligned} z'_1 &= (x_0 + y r_1) y^{-1} \\ z'_0 &= (x_1 + y r_0) y^{-1}\,, \end{aligned}$$

where $z' = x^{-1} + r$, and

$$\begin{aligned} z_1 &= x_0 y^{-1} \\ z_0 &= x_1 y^{-1}\,, \end{aligned}$$

where $z = x^{-1}$. Then,

$$\varepsilon = z + r + z' = 0\,.$$

Thus, the computation is not infected.

As a result, the computation is infected only when $x = \Delta$ or $x = 0$ (assuming $r \neq 0$). The attacker can then exploit this by performing the encryption twice: once with the fault injection and once without. If the outputs are different (indicating the encryption with a fault injected was infected), the attacker learns that $x = \Delta$ or $x = 0$. To distinguish between these two, the attacker can repeat the attack with a different Δ value. If the encryption with fault injection is again infected, it confirms that $x = 0$.

A potential fix involves replacing each AND gate in the Kronecker Delta function by a gate which outputs 1 only when both inputs are zero. Then, the Kronecker delta function can be computed directly on x, eliminating the need to first compute $x + 255$ as done in the original design.

Fault location 2: XOR of random r with the S-box input. The second fault scenario is that, an attacker injects an additive fault Δ at the XOR gate where the random value r is combined with the state after the linear map (indicated as number 2 in Figure 3). This alters the S-box input to $x + \Delta$. Consequently, the $\mathbb{F}_{2^8}$ inversion output becomes $(x + \Delta)^{-1} + r$. However, as the infection mechanism is using the same faulty input $x + \Delta$, it fails to detect the injected fault, making the implementation susceptible to DFA. Specifically, the infection mechanism of the RS-Mask design only accounts for faults occurring within the $\mathbb{F}_{2^8}$ inversion circuit, and does not cover the faults injected at earlier stages.

A potential fix could involve introducing a full redundancy. This would mean replicating the entire circuit and including an error check mechanism at the end of the computation. Such a design ensures that effective faults injected at any point in the circuit are detected to protect against DFA.

Fault location 3: Inversion of r. The third fault scenario is that an attacker injects a fault within the inversion circuit that computes r^{-1} (indicated as number 3 in Figure 3) causing the output of that circuit to be $r^{-1} + \varDelta$. We analyze this fault scenario in two cases, $x = 0$ and $x \neq 0$.

First, $x = 0$ implies $f = 0$. Then, using Equation 4, we find:

$$\begin{aligned} z'_1 &= (r^{-1} + \varDelta_0)y^{-1} \\ z'_0 &= (r^{-1} + \varDelta_1)y^{-1}\,, \end{aligned}$$

where $z' = (r^{-1} + \varDelta)^{-1}$, and $z = 0$ ($z_1 = 0$, $z_0 = 0$). Then,

$$\begin{aligned} \varepsilon &= z + r + z' \\ &= r + (r^{-1} + \varDelta)^{-1}\,. \end{aligned}$$

Since $\varDelta \neq 0$, $\varepsilon \neq 0$. Thus, the computation is infected.

Second, $x \neq 0$ implies $f = 1$. Then, using Equation 4, we find:

$$\begin{aligned} z'_1 &= (x_0 + yr_1)y^{-1} \\ z'_0 &= (x_1 + yr_0)y^{-1}\,, \end{aligned}$$

where $z' = x^{-1} + r$, and

$$\begin{aligned} z_1 &= x_0 y^{-1} \\ z_0 &= x_1 y^{-1}\,, \end{aligned}$$

where $z = x^{-1}$. Then,

$$\varepsilon = z + r + z' = 0\,.$$

Thus, the computation is not infected.

As a result, the computation is infected only when $x = 0$ (if $r \neq 0$). Similar to the first fault scenario, the attacker can then exploit this by performing the encryption twice: once with the fault injection and once without. The outputs of both encryptions being different implies that the first encryption was infected, which also implies $x = 0$.

A potential fix could involve precomputing all (r, r^{-1}) pairs before the encryption. Then, the probability of selecting the faulty pair becomes 1/256 considering a single fault injection. While this does not fully eliminate the vulnerability, it decreases the success probability of the attacker. A second potential fix could involve including r and r^{-1} in the infection mechanism, *i.e.,* infecting the computation by randomizing $(r^{-1})^{-1} + r$ which does not equal to zero if r^{-1} is faulty.

Fault location 4: Additional RS-Mask circuit inside the inversion. The fourth fault scenario is that an attacker injects an additive fault to one of the gates within the additional RS-Mask circuit, indicated as number 4 in Figure 3. Any fault injected to these gates causes a faulty value at the input of the $\mathbb{F}_{2^4}$ multiplication with f. Then, the output of this multiplication is subsequently multiplied by y^{-1}, the $\mathbb{F}_{2^4}$ inversion output. We analyze this fault scenario in two cases, $x = 0$ and $x \neq 0$.

First, $x = 0$ implies $f = 0$, thereby nullifying the effect of the injected fault, as the faulty value is multiplied by zero. Then, using Equation 4, we find $z' = r$, and $z = 0$. Then,

$$\varepsilon = z + r + z' = 0\,.$$

Thus, the computation is not infected.

Second, $x \neq 0$ implies $f = 1$. Then, using Equation 4, we find:

$$\begin{aligned} z'_1 &= (x_0 + (yr_1 + \Delta_1))y^{-1} \\ z'_0 &= (x_1 + (yr_0 + \Delta_0))y^{-1}\,, \end{aligned}$$

where $z' = x^{-1} + r + \Delta'$, and

$$\begin{aligned} z_1 &= x_0 y^{-1} \\ z_0 &= x_1 y^{-1}\,, \end{aligned}$$

where $z = x^{-1}$. Then,

$$\varepsilon = z + r + z' = \Delta' \neq 0\,.$$

Thus, the computation is infected.

As a result, the infection applies when $x \neq 0$, due to the injected fault being propagated to the output when $f = 1$. As in the previous fault scenarios, the attacker can repeat the encryption and compare the outputs; if no difference between the outputs is observed, they infer that $x = 0$.

One potential fix against this fault scenario is to multiply r_0 and r_1 by f before the RS-Mask $\mathbb{F}_{2^8}$ inversion circuit. This ensures that any fault injected to the gates indicated as number 4 propagates to the output, regardless of the value of x. This is because, within the RS-Mask inversion circuit, the input to the inversion logic is never zero, even when $x = 0$. Only the propagation of the faults injected to r_0 or r_1 depends on f, however then, these faults are equivalent to register faults injected to the RS-Mask random r. These would have the same effect as using a different r value.

In the original paper, the security of RS-Mask countermeasure is evaluated against statistical (ineffective) fault attacks, focusing on a single location of fault injection. However, as the overall security of the countermeasure cannot be ensured by evaluating only this single scenario, such an evaluation remains incomplete. The fault scenarios discussed in this section highlight a critical observation: any additional circuit introduced to protect against fault attacks inherently expands the attack surface, and may introduce new points of vulnerability that could be exploited. Consequently, for countermeasures without provable security guarantees, it is essential to comprehensively evaluate any additional circuit introduced against fault attacks to identify potential vulnerabilities. Similarly, the proposed fixes for the fault scenarios presented in this section do not inherently ensure security. These fixes may further expand the attack surface, and therefore, they must undergo a rigorous evaluation to assess their security against fault attacks.

In the following section, we describe a random space mapping for Canright's AES S-box, where we systemically evaluate each gate in the resulting circuit.

4 An AES-specific Countermeasure

In this section, we introduce an AES-specific countermeasure, based on the core idea of the RS-Mask countermeasure, protecting against single gate/register-faulting adversary. Unlike the RS mapping derived for the AES S-box implemented using the Canright construction, which is conditioned on the input value x, our proposed countermeasure derives a uniform mapping that is independent of the value of x.

Derivation of the random mapping for the AES S-box implemented using Canright construction. In Canright's AES S-box, as shown in Figure 1, two inversion operations are performed in $\mathbb{F}_{2^8}$ and $\mathbb{F}_{2^4}$. Each involves multiplying the input of the inversion (x, y) by the corresponding outputs of the inner inversions (y^{-1}, k^{-1}). These multiplications are vulnerable to SIFA, as faults present in y^{-1} and k^{-1} are not propagated to the S-box output when the input x is zero (which also leads to y being zero).

The variables (x, y, k, k^{-1}, y^{-1}, and x^{-1}) referenced in this section correspond to those shown in Figure 1.

Consider the following table, which summarizes the behavior of expressions derived from y and y^{-1} under different S-box input x conditions:

	y	y^{-1}	$yy^{-1}+1$	$y+yy^{-1}+1$	$y^{-1}+yy^{-1}+1$
$x=0$	0	0	1	1	1
$x\neq 0$	y	y^{-1}	0	y	y^{-1}

As observed in the above table, the following holds for all input values x:

$$(y+yy^{-1}+1)(y^{-1}+yy^{-1}+1)=1\,.$$

Similarly, the following expression also holds for all input values $x=x_1||x_0$:

$$x_0(yy^{-1}+1)=0$$
$$x_1(yy^{-1}+1)=0\,.$$

This behavior is in contrast to the original RS-Mask countermeasure, where the multiplication of y and y^{-1} introduces an input-dependent behavior.

We now describe the new mapping for the Canright's AES S-box as follows:

$$\begin{aligned} z'_1 &= z_1 + r_1 = (x_0 + (y + yy^{-1} + 1)r_1)(y^{-1} + yy^{-1} + 1) \\ z'_0 &= z_0 + r_0 = (x_1 + (y + yy^{-1} + 1)r_0)(y^{-1} + yy^{-1} + 1)\,, \end{aligned} \tag{5}$$

where $z' = z'_1||z'_0 = x^{-1} + r$.

Similar to the RS-Mask countermeasure, analyzing the distribution of the output differences (between the faulty and correct S-box outputs) under faults present in y and y^{-1} reveals a non-uniform pattern. This non-uniformity exposes the design to differential fault attacks.

Rather than relying on an infection mechanism, which, when implemented correctly (*e.g.,* via full duplication), requires substantial area overhead, we propose a lightweight tweak to the random mapping. Specifically, we introduce an additional term, $(yy^{-1}+1)r'y^{-1}$ to the S-box output. In the absence of faults, this term evaluates to zero, as shown in the table above. Here, r' is a uniform random value distinct from the one used in the main mapping described in Equation 4. The modified mapping is given as follows, where $r = r_1||r_0$ and $r' = r_3||r_2$:

$$\begin{aligned} z'_1 &= z_1 + r_1 = (x_1 + (y + yy^{-1} + 1)r_1)(y^{-1} + yy^{-1} + 1) + (yy^{-1} + 1)r_3y^{-1} \\ z'_0 &= z_0 + r_0 = (x_0 + (y + yy^{-1} + 1)r_0)(y^{-1} + yy^{-1} + 1) + (yy^{-1} + 1)r_2y^{-1}\,. \end{aligned} \tag{6}$$

It is clear that this mapping is correct if no faults occur during the computation of y and y^{-1}.

We note that it is essential to compute y using a redundant datapath similar to the RS-Mask countermeasure in order to account for the faults that are injected earlier in the circuit.

As previously noted, the inversion in Canright's AES S-box involves two multiplications that are susceptible to SIFA. To ensure both multiplications are secure against SIFA, the mapping introduced in Equation 5 is also applied to the $\mathbb{F}_{2^4}$ inversion. Furthermore, similar to the inversion in $\mathbb{F}_{2^8}$, k is also computed using a redundant datapath in order to account for the faults that are injected earlier in the circuit. We describe the mapping for the $\mathbb{F}_{2^4}$ inversion as follows:

$$\begin{aligned} y_1^{-1} &= (y_1 + (k + kk^{-1} + 1)r_5)(k^{-1} + kk^{-1} + 1) + r_5 \\ y_0^{-1} &= (y_0 + (k + kk^{-1} + 1)r_4)(k^{-1} + kk^{-1} + 1) + r_4\,, \end{aligned} \tag{7}$$

where $y^{-1} = y_1^{-1}||y_0^{-1}$.

Unlike the mapping derived for the inversion in $\mathbb{F}_{2^8}$ (Equation 4), the corresponding mapping for the inversion in $\mathbb{F}_{2^4}$ naturally yields uniform output differences, eliminating the need for an additional random term.

To protect against differential fault attacks, we introduce an intrinsic redundancy by duplicating the S-box input. The random space mapping components computed from the first input are then used in the calculation of the second S-box output using the duplicated S-box input as shown in Figure 4.

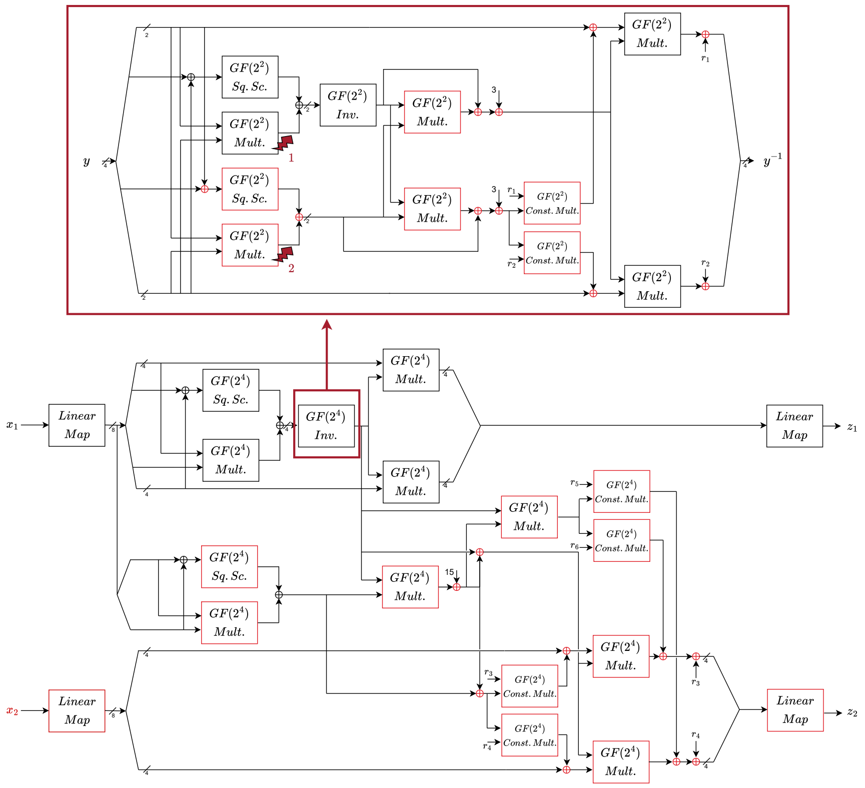


Fig. 4: $\mathbb{F}_{2^8}$ inversion of the AES S-box.

Hardware implementation. The overall block diagram of the AES S-box implementation protected using the proposed random space mapping is presented in Figure 4. Similar to RS-Mask countermeasure, the state is split into three shares, however we do not keep the random mask values as part of the shared state. Meaning, all linear and non-linear components in the design operate over three shares.

The upper circuit in Figure 4 depicts the mapping for the inversion in $\mathbb{F}_{2^4}$, and the lower circuit depicts the mapping for the whole AES S-box implemented using Canright's construction. The $\mathbb{F}_{2^4}$ inverter ($GF(2^4)$ *Inv.*) within the AES S-box uses the upper circuit. Components added beyond Canright's original AES S-box implementation are highlighted in red.

Our implementation builds on the $\mathbb{F}_{2^8}$ inversion circuit from [6] based on Consolidating Masking Schemes [27] with three shares to achieve second order security. In this construction, the inversion circuit consumes 162 bits of randomness. Building upon this foundation, we extend the circuit by integrating the additional components (shown as red in Figure 4) required for the random space mapping.

While extending the circuit, we adhere to the $d+1$-share multipliers from the original inversion circuit. Together with the 20 bits of randomness required by the random space mapping and additional multiplications, our implementation consumes 344 bits of randomness. The design is implemented over 8 pipeline stages.

4.1 Side-Channel Security

In this section, we discuss the SCA security of the AES S-box implementation using the proposed random mapping. To evaluate the SCA security of our design, we perform TVLA (Test Vector Leakage Assessment) [7]. We perform the assessment for the S-box implemented with the proposed random space mapping by supplying it with fixed versus random inputs. The S-box is placed on a Xilinx Spartan-6 FPGA located on a SAKURA-G evaluation board [30], whereas masks generation is located on a separate control FPGA. The devices are supplied with a stable 6.144 MHz clock and an oscilloscope samples power consumption traces at a rate of 1GS/sec. The results of the first and second-order tests are depicted in Figure 5. The results report no leakage regarding first and second-order probing security.

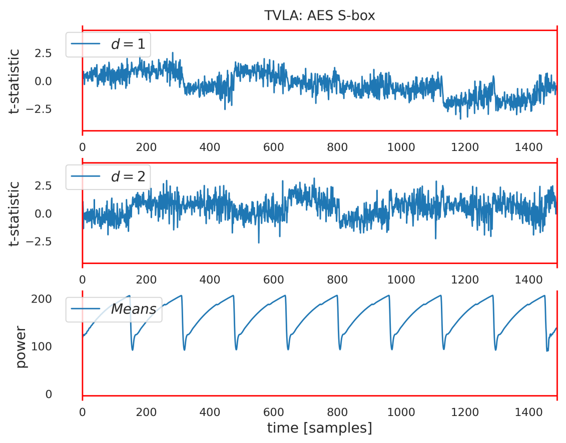


Fig. 5: TVLA result of the AES S-box implemented with the proposed random space mapping, first and second-order security, 100 million traces, and a sample trace.

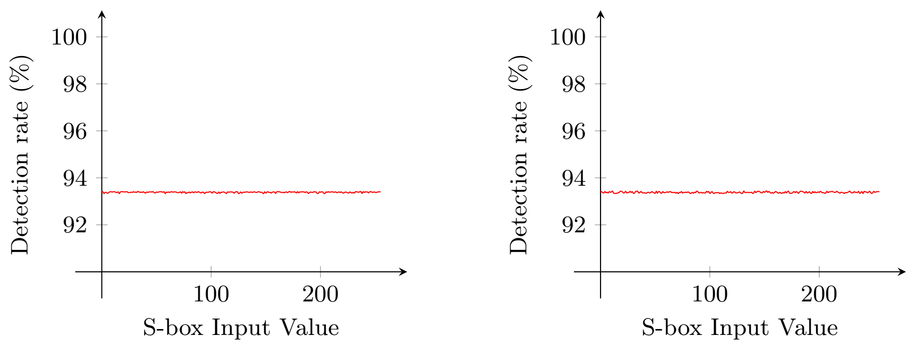
4.2 Fault Security

We now discuss the fault security of the AES S-box implementation using the proposed random space mapping.

Existing provable methods for fault security evaluation, typically based on error detection or correction, are not applicable to our design. Rather than preventing the propagation of injected faults through detection or correction mechanisms, our countermeasure ensures that the injected faults are deliberately propagated to infect the circuit's state. As a result, the infected output that is not detected by the error detection circuit at the end of the encryption becomes unexploitable.

To evaluate the fault security of our AES-specific countermeasure, we analyze the detection rate of the injected faults. A uniform detection rate across all input values indicates resistance against statistical fault attacks. We performed an attack simulation on the C implementation of the AES S-box design to evaluate the security against a single gate/register-faulting adversary. Faults were modeled as XOR additions applied to the intermediate variables. To simulate this, we extended the C code of the design by XORing a fault variable with selected critical intermediate values. Specifically, faults were injected to each component of the S-box, namely XOR gates and all field operations as shown in Figure 4). For each experiment, we marked the fault as detected if the outputs differed. For each input value, 500,000 experiments are conducted to account for wide range of fault values and random values introduced by the random mapping. The results demonstrated that the detection rate remains uniform across all input values for each fault location and fault value.

Figure 6a and 6b illustrate that the detection rate remains uniform across all S-box input values when a fault is injected to the output of the $\mathbb{F}_{2^2}$ multiplier (similar to the fault scenario used in the RS-Mask countermeasure). In our design, the multiplier is duplicated as part of the random mapping, corresponding to fault locations 1 and 2 in Figure 4).



(a) Fault location 1

(b) Fault location 2

Fig. 6: Distribution of the detection rate across all S-box input values when a fault is injected to the output of the duplicated $\mathbb{F}_{2^2}$ multipliers denoted with fault location 1 and 2 in Figure 4.

Figure 7 shows a uniform detection rate (%100 detection) across all S-box input values when simulating fault scenario 4, previously proposed against the RS-Mask countermeasure, by injecting a fault to the $\mathbb{F}_{2^4}$ constant multiplier as denoted with fault location 3 in Figure 4.

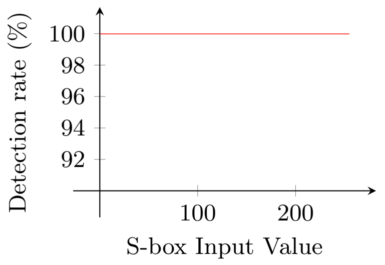


Fig. 7: Distribution of the detection rate across all S-box input values when a fault is injected to the output of the $\mathbb{F}_{2^4}$ constant multiplier (of the random mapping circuit) denoted with fault location 3 in Figure 4.

We repeated the evaluation, this time focusing on the distribution of output differences. As before, for each experiment, the resulting output differences were recorded. The results show that the output differences are uniformly distributed across all S-box input values for every fault location and fault value tested. Together, these two evaluations demonstrates the correctness and the privacy of the protected AES S-box with the proposed random space mapping.

4.3 Hardware Benchmarks

In this section, we evaluate the hardware performance of our protected AES S-box implementation. We use the Synopsys Design Compiler (version S-2021.06-SP3) together with the open source NANGATE 45nm library. The hierarchical structure is preserved during the synthesis, and undesired optimizations are avoided.

We compare our protected AES S-box implementation secure against SCA and fault attacks with the AES S-box implementation from [6] based on Consolidated Masking Scheme secure against only SCA which we implemented ourselves. Both designs provide second-order SCA security. Compared to the implementation from [6], the number of masked field multiplications is doubled (one of which is duplicated using same randomness) in our implementation. Together with the extra randomness required by the random space mapping, our design requires 110 additional random bits. Regarding the area costs, our design introduces an area overhead of approximately 2.85.

We report only the overhead factors relative to their corresponding SCA-only designs for the RS-Mask countermeasure [25], λ-detection M&M [18] and

StaMAC [11], rather than absolute area numbers. The selection of these countermeasures, in addition to RS-Mask, for comparison is motivated by the availability of area reports for their AES (S-box) implementations, as well as their protection against SCA and SIFA. While our design incurs a slightly higher area overhead compared to the original RS-Mask countermeasure, it offers stronger protection by additionally securing against SIFA and achieving second-order SCA security, features not offered by RS-Mask.

Table 1: Hardware cost comparison of various designs.

Design	SCA Order	S-box Stages	Randomness [bit]	Area (kGE)	Overhead
SCA only [6]	2	6	162	4.5**	
RS-Mask [25]	1	10			2*
λ-detection M&M [18]	2	6	903		3.49*
StaMAC [11]	1	9	56		3.83*
This paper	2	8	344	12.84**	2.85

* overhead for the whole AES implementation
** only inversion

5 Conclusion

In this paper, we addressed the gaps in the fault evaluation of RS-Mask and identified several vulnerabilities through which a single fault injection in the protected AES implementation compromise its security. The attack scenarios we present demonstrate that the random space mapping proposed for the AES S-box in the RS-Mask countermeasure retains dependency on the secret data, allowing attackers to exploit this dependency under specific fault scenarios. Although RS-Mask includes a formal security proof, it only guarantees that the random space mapping produces uniformly distributed values, and it does not address the fault security of the additional circuit required to implement this mapping. Our analysis shows that this overlooked aspect introduces exploitable weaknesses. These findings highlight the critical need for more comprehensive fault evaluations in the design of fault countermeasures, as increasing area to protect against fault attacks may introduce new attack surface to the attackers.

Besides these attack scenarios, we proposed an AES-specific countermeasure involving a single random space mapping for all S-box input values without an auxiliary variable. With this unified mapping, we aimed to mitigate the vulnerabilities we identified in the RS-Mask countermeasure. Our evaluation of the proposed countermeasure confirms the improved robustness against statistical and differential fault attacks.

Acknowledgments. This work was supported by CyberSecurity Research Flanders with reference number VR20192203.

References

1. Aghaie, A., Moradi, A., Rasoolzadeh, S., Shahmirzadi, A.R., Schellenberg, F., Schneider, T.: Impeccable circuits. IEEE Trans. Computers **69**(3), 361–376 (2020). https://doi.org/10.1109/TC.2019.2948617, https://doi.org/10.1109/TC.2019.2948617
2. Anderson, R.J., Kuhn, M.G.: Low cost attacks on tamper resistant devices. In: Christianson, B., Crispo, B., Lomas, T.M.A., Roe, M. (eds.) Security Protocols, 5th International Workshop, Paris, France, April 7-9, 1997, Proceedings. Lecture Notes in Computer Science, vol. 1361, pp. 125–136. Springer (1997). https://doi.org/10.1007/BFb0028165, https://doi.org/10.1007/BFb0028165
3. Boneh, D., DeMillo, R.A., Lipton, R.J.: On the importance of checking cryptographic protocols for faults (extended abstract). In: Fumy, W. (ed.) Advances in Cryptology - EUROCRYPT '97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding. Lecture Notes in Computer Science, vol. 1233, pp. 37–51. Springer (1997). https://doi.org/10.1007/3-540-69053-0_4, https://doi.org/10.1007/3-540-69053-0_4
4. Canright, D.: A very compact s-box for AES. In: Rao, J.R., Sunar, B. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3659, pp. 441–455. Springer (2005). https://doi.org/10.1007/11545262_32, https://doi.org/10.1007/11545262_32
5. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards sound approaches to counteract power-analysis attacks. In: Wiener, M.J. (ed.) Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings. Lecture Notes in Computer Science, vol. 1666, pp. 398–412. Springer (1999). https://doi.org/10.1007/3-540-48405-1_26, https://doi.org/10.1007/3-540-48405-1_26
6. Cnudde, T.D., Reparaz, O., Bilgin, B., Nikova, S., Nikov, V., Rijmen, V.: Masking AES with d+1 shares in hardware. In: Gierlichs, B., Poschmann, A.Y. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings. Lecture Notes in Computer Science, vol. 9813, pp. 194–212. Springer (2016). https://doi.org/10.1007/978-3-662-53140-2_10, https://doi.org/10.1007/978-3-662-53140-2_10
7. Cooper, J., DeMulder, E., Goodwill, G., Jaffe, J., Kenworthy, G., Rohatgi, P., et al.: Test vector leakage assessment (TVLA) methodology in practice. In: International Cryptographic Module Conference. vol. 20 (2013)
8. Daemen, J., Dobraunig, C., Eichlseder, M., Groß, H., Mendel, F., Primas, R.: Protecting against statistical ineffective fault attacks. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2020**(3), 508–543 (2020). https://doi.org/10.13154/TCHES.V2020.I3.508-543, https://doi.org/10.13154/tches.v2020.i3.508-543
9. Dehbaoui, A., Dutertre, J., Robisson, B., Tria, A.: Electromagnetic transient faults injection on a hardware and a software implementations of AES. In: Bertoni, G.,

Gierlichs, B. (eds.) 2012 Workshop on Fault Diagnosis and Tolerance in Cryptography, Leuven, Belgium, September 9, 2012. pp. 7–15. IEEE Computer Society (2012). https://doi.org/10.1109/FDTC.2012.15, https://doi.org/10.1109/FDTC.2012.15

10. Dhooghe, S., Ovchinnikov, A., Toprakhisar, D.: Stati: Protecting against fault attacks using stable threshold implementations. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2024**(1), 229–263 (2024). https://doi.org/10.46586/TCHES.V2024.I1.229-263, https://doi.org/10.46586/tches.v2024.i1.229-263
11. Dhooghe, S., Ovchinnikov, A., Toprakhisar, D.: Stamac: Fault protection via stable-mac tags. IACR Cryptol. ePrint Arch. p. 455 (2025), https://eprint.iacr.org/2025/455
12. Dobraunig, C., Eichlseder, M., Groß, H., Mangard, S., Mendel, F., Primas, R.: Statistical ineffective fault attacks on masked AES with fault countermeasures. In: Peyrin, T., Galbraith, S.D. (eds.) Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part II. Lecture Notes in Computer Science, vol. 11273, pp. 315–342. Springer (2018). https://doi.org/10.1007/978-3-030-03329-3_11, https://doi.org/10.1007/978-3-030-03329-3_11
13. Dobraunig, C., Eichlseder, M., Korak, T., Mangard, S., Mendel, F., Primas, R.: SIFA: exploiting ineffective fault inductions on symmetric cryptography. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2018**(3), 547–572 (2018). https://doi.org/10.13154/tches.v2018.i3.547-572, https://doi.org/10.13154/tches.v2018.i3.547-572
14. Faust, S., Grosso, V., Pozo, S.M.D., Paglialonga, C., Standaert, F.: Composable masking schemes in the presence of physical defaults & the robust probing model. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2018**(3), 89–120 (2018). https://doi.org/10.13154/TCHES.V2018.I3.89-120, https://doi.org/10.13154/tches.v2018.i3.89-120
15. Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic analysis: Concrete results. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings. Lecture Notes in Computer Science, vol. 2162, pp. 251–261. Springer (2001). https://doi.org/10.1007/3-540-44709-1_21, https://doi.org/10.1007/3-540-44709-1_21
16. Groß, H., Mangard, S., Korak, T.: Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order. In: Bilgin, B., Nikova, S., Rijmen, V. (eds.) Proceedings of the ACM Workshop on Theory of Implementation Security, TIS@CCS 2016 Vienna, Austria, October, 2016. p. 3. ACM (2016). https://doi.org/10.1145/2996366.2996426, https://doi.org/10.1145/2996366.2996426
17. Habing, D.H.: The use of lasers to simulate radiation-induced transients in semiconductor devices and circuits. IEEE Transactions on Nuclear Science **12**(5), 91–100 (1965)
18. Hirata, H., Miyahara, D., Arribas, V., Li, Y., Miura, N., Nikova, S., Sakiyama, K.: All you need is fault: Zero-value attacks on AES and a new λ-detection m&m. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2024**(1), 133–156 (2024)
19. Ishai, Y., Prabhakaran, M., Sahai, A., Wagner, D.A.: Private circuits II: keeping secrets in tamperable circuits. In: Vaudenay, S. (ed.) Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceed-

ings. Lecture Notes in Computer Science, vol. 4004, pp. 308–327. Springer (2006). https://doi.org/10.1007/11761679_19, https://doi.org/10.1007/11761679_19

20. Ishai, Y., Sahai, A., Wagner, D.A.: Private circuits: Securing hardware against probing attacks. In: Boneh, D. (ed.) Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings. Lecture Notes in Computer Science, vol. 2729, pp. 463–481. Springer (2003). https://doi.org/10.1007/978-3-540-45146-4_27, https://doi.org/10.1007/978-3-540-45146-4_27
21. Kocher, P.C.: Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In: Koblitz, N. (ed.) Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings. Lecture Notes in Computer Science, vol. 1109, pp. 104–113. Springer (1996). https://doi.org/10.1007/3-540-68697-5_9, https://doi.org/10.1007/3-540-68697-5_9
22. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M.J. (ed.) Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings. Lecture Notes in Computer Science, vol. 1666, pp. 388–397. Springer (1999). https://doi.org/10.1007/3-540-48405-1_25, https://doi.org/10.1007/3-540-48405-1_25
23. Meyer, L.D., Arribas, V., Nikova, S., Nikov, V., Rijmen, V.: M&m: Masks and macs against physical attacks. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2019**(1), 25–50 (2019). https://doi.org/10.13154/tches.v2019.i1.25-50, https://doi.org/10.13154/tches.v2019.i1.25-50
24. Nikova, S., Rechberger, C., Rijmen, V.: Threshold implementations against side-channel attacks and glitches. In: Ning, P., Qing, S., Li, N. (eds.) Information and Communications Security, 8th International Conference, ICICS 2006, Raleigh, NC, USA, December 4-7, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4307, pp. 529–545. Springer (2006). https://doi.org/10.1007/11935308_38, https://doi.org/10.1007/11935308_38
25. Ramezanpour, K., Ampadu, P., Diehl, W.: Rs-mask: Random space masking as an integrated countermeasure against power and fault analysis. In: 2020 IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2020, San Jose, CA, USA, December 7-11, 2020. pp. 176–187. IEEE (2020). https://doi.org/10.1109/HOST45689.2020.9300266, https://doi.org/10.1109/HOST45689.2020.9300266
26. Rasoolzadeh, S., Shahmirzadi, A.R., Moradi, A.: Impeccable circuits III. In: IEEE International Test Conference, ITC 2021, Anaheim, CA, USA, October 10-15, 2021. pp. 163–169. IEEE (2021). https://doi.org/10.1109/ITC50571.2021.00024, https://doi.org/10.1109/ITC50571.2021.00024
27. Reparaz, O., Bilgin, B., Nikova, S., Gierlichs, B., Verbauwhede, I.: Consolidating masking schemes. In: Gennaro, R., Robshaw, M. (eds.) Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9215, pp. 764–783. Springer (2015). https://doi.org/10.1007/978-3-662-47989-6_37, https://doi.org/10.1007/978-3-662-47989-6_37
28. Reparaz, O., Meyer, L.D., Bilgin, B., Arribas, V., Nikova, S., Nikov, V., Smart, N.P.: CAPA: the spirit of beaver against physical attacks. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings,

Part I. Lecture Notes in Computer Science, vol. 10991, pp. 121–151. Springer (2018). https://doi.org/10.1007/978-3-319-96884-1_5, https://doi.org/10.1007/978-3-319-96884-1_5

29. Saha, S., Jap, D., Roy, D.B., Chakraborti, A., Bhasin, S., Mukhopadhyay, D.: Transform-and-encode: A countermeasure framework for statistical ineffective fault attacks on block ciphers. IACR Cryptol. ePrint Arch. p. 545 (2019), https://eprint.iacr.org/2019/545
30. SAKURA: Side-channel Attack User Reference Architecture. http://satoh.cs.uec.ac.jp/SAKURA/index.html
31. Schneider, T., Moradi, A., Güneysu, T.: Parti - towards combined hardware countermeasures against side-channel and fault-injection attacks. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9815, pp. 302–332. Springer (2016). https://doi.org/10.1007/978-3-662-53008-5_11, https://doi.org/10.1007/978-3-662-53008-5_11
32. Shahmirzadi, A.R., Rasoolzadeh, S., Moradi, A.: Impeccable circuits II. In: 57th ACM/IEEE Design Automation Conference, DAC 2020, San Francisco, CA, USA, July 20-24, 2020. pp. 1–6. IEEE (2020). https://doi.org/10.1109/DAC18072.2020.9218615, https://doi.org/10.1109/DAC18072.2020.9218615