A First-Order SCA Resistant AES without Fresh Randomness

Felix Wegener and Amir Moradi

Horst Görtz Institute for IT Security, Ruhr-Universität Bochum, Bochum, Germany
{firstname.lastname}@rub.de

Abstract. Since the advent of Differential Power Analysis (DPA) in the late 1990s protecting embedded devices against Side-Channel Analysis (SCA) attacks has been a major research effort. Even though many different first-order secure masking schemes are available today, when applied to the AES S-box they all require fresh random bits in every evaluation. As the quality criteria for generating random numbers on an embedded device are not well understood, an integrated Random Number Generator (RNG) can be the weak spot of any protected implementation and may invalidate an otherwise secure implementation. We present a new construction based on *Threshold Implementations* and *Changing of the Guards* to realize a first-order secure AES with zero per-round randomness. Hence, our design does not need a built-in RNG, thereby enhancing security and reducing the overhead.

1 Introduction

In 1999 Kocher *et al.* introduced the extraction of key information from the power consumption of a hardware device during cryptographic computations [22]. To break a symmetric cipher, key hypotheses are formed that categories power traces into groups and a statistical test is performed to estimate the likelihood of a correct guess. This process is known as Differential Power Analysis (DPA).
To prevent side-channel analysis (SCA) attacks extensive research in countermeasures has been undertaken [16, 18, 20, 28, 30]. A notable protection method is Threshold Implementation (TI) [25], as it grants provable security against first-order SCA attacks and can potentially be realized without additional randomness. TI demands three properties to ensure the security of an implementation: correctness, non-completeness and uniformity. While correctness simply preserves the validity of the computation, non-completeness ensures that every intermediate value is independent of secret values even in the presence of glitches. Uniformity implies that when the input is shared by the masks drawn from a uniform distribution, the output should be also represented by masks drawn from a uniform distribution. So far, all first-order protected AES implementations inject randomness in every round to achieve a uniform masking. Hence, they require to employ a random number generator on the embedded device in addition to the creation of the shared plaintext, which is considered to be done externally before being provided to the cryptographic core.

Related Works.

S-box Structure. In 2005 Canright suggested a tower field approach to realize the $\mathbb{GF}(2^8)$ inversion in the AES S-box by inversions and multiplications in smaller finite fields [8]. Due to its reduced size compared to a naive implementation most of the recent first-order secure AES implementations are based on this S-box design.

Threshold Implementation. Bilgin *et al.* suggested to mask all multipliers, squarers and inverters in Canright's construction separately by applying TI with two to four shares. This approach requires 16 random bits per S-box evaluation to recombine the different components in a uniform way and occupies an area of 2224 GE in the UMC 180 nm library [4].

Domain-Oriented Masking (DOM). Based on Canright's construction Groß *et al.* presented masked version of all multiplications in $\mathbb{GF}(2^2)$ with a DOM independent multiplier to achieve first-order security. The construction consumes 2600 GE in UMC $0.18\mu m$ and requires 18 bits of randomness per S-box call [17].

Masking with $d+1$ Shares. In [10] Cnudde *et al.* suggested to apply masking with $d+1$ shares [28] to operations in $\mathbb{GF}(2^2)$ to realize the AES S-box in 1872 GE in the NanGate 45 nm Open Cell Library with 54 bits of randomness per S-box call. In contrast, Ueno *et al.* kept the inversion in $\mathbb{GF}(2^4)$ as a single primitive and applied the same masking scheme, which requires 64 bits of randomness per S-box call and can be realized in only 1389 GE in TSMC 65 nm standard cells library [31].

Changing of the Guards. Recently, Daemen [11] showed that uniformity for a bijective S-box can be easily achieved on the entire S-box layer by injecting some additional randomness, called guards, into the first S-box. In order to satisfy the uniformity of the subsequent S-boxes, he introduced a mechanism to make use of the shares of the former S-box. More precisely, the uniformity is achieved by remasking, but instead of fresh randomness part of the shared cipher state is used.

Zero per-Round Randomness. One of the several designs presented by Ghoshal and De Cnudde in [14] is a four-share AES S-box with zero fresh randomness. They considered the S-box construction of Boyar and Peralta [6] and replaced the non-linear gates with their TI variants.

Our Contribution. Obviously, our target in this paper is the same as that of the four-share design of [14], i.e., no extra randomness per S-box. First, we demonstrate a first-order leakage of that construction both in theory and with a practical evaluation. Second, we present an alternative and novel design for the AES S-box which enables us to reuse the hardware modules hence shrinking the

area. Our approach, which is also a four-share TI design, is based on a bijective decomposition of the AES S-box allowing us to apply the *Changing of the Guards* method to achieve the uniformity. We would like to highlight that the application of this method is only possible in bijective constructions, and our design is the first in which the AES S-box is decomposed to bijective functions. Third, we present a trick in order to employ our four-share S-box in a two-share AES encryption design without additional randomness. In consequence, our practically-evaluated first-order secure AES implementation eliminates the necessity of generating any random bits in hardware as it only requires the externally shared plaintext and guards, but no fresh randomness. Further, our design has a comparable size to recent first-order secure implementations.

2 Preliminaries

In this section we introduce relevant definitions and our notation for the rest of the paper.

Adversary Model. To appropriately model the influence of glitches in a hardware circuit Ishai *et al.* suggested a d-probing model where the attacker may probe up to d wires corresponding to the intermediate values of the cipher [20]. As we focus on first-order security the attacker may probe only one wire. According to Duc *et al.* [12], it is commonly known that this corresponds to an attacker who only estimates the means (and no higher-order moments) on the recorded side-channel measurements.

Masking. As this paper exclusively deals with four-share implementations, we restrict our definitions to four shares for better readability. For a secret value $x \in \mathbb{F}_2^n$ we denote its Boolean sharing into four shares as $X = (a, b, c, d)$ with the property:

$$x = a \oplus b \oplus c \oplus d.$$

We write $\int(x)$ for the set of all possible Boolean sharings X of x.

Threshold Implementation. In 2006 Nikova *et al.* introduced TI as a provably secure masking scheme for hardware platforms [25]. Below, we give an introduction following our own notation.
Let $\mathsf{f} : \mathbb{F}_2^n \to \mathbb{F}_2^m$ be a Boolean function. We write that

$$\mathsf{F} : \mathbb{F}_2^{4n} \to \mathbb{F}_2^{4m}, \qquad \mathsf{F}(X) = \left(\mathsf{F}^A(X), \mathsf{F}^B(X), \mathsf{F}^C(X), \mathsf{F}^D(X)\right),$$

with $\mathsf{F}(.)$ being a first-order TI of $\mathsf{f}(.)$ if it fulfills two properties. First, it is *correct*, i.e., the summation of all output shares reveals the unshared result of the computation $\mathsf{f}(.)$ on x:

$$\mathsf{F}^A(X) \oplus \mathsf{F}^B(X) \oplus \mathsf{F}^C(X) \oplus \mathsf{F}^D(X) = \mathsf{f}(x).$$

Second, it must be *non-complete*, i.e., each component function $\mathsf{F}^{A/B/C/D}(.)$ computing one output share does not depend on all input shares. As an example,

$$\mathsf{F}^A(X) = \mathsf{F}^A(b,c,d), \qquad \mathsf{F}^B(X) = \mathsf{F}^B(a,c,d),$$
$$\mathsf{F}^C(X) = \mathsf{F}^C(a,b,d), \qquad \mathsf{F}^D(X) = \mathsf{F}^D(a,b,c).$$

The latter property ensures the security in the 1-probing model even in the presence of glitches as probing one output share yields at most three input shares. In the following, we are going to use a capital letter F to denote the TI of a function f. If it is clear which shared function we are referring to, we abbreviate a, b, c, d for its input shares and A, B, C, D for its output shares.

The central theorem underlying TI guarantees that given a sharing of x drawn equiprobably from all sharings $\int(x)$, the evaluation of the TI function $\mathsf{F}(.)$ does not cause first-order leakage. As $\mathsf{f}(.)$ is not the only non-linear function used in a cipher, and its output derives other non-linear TI functions (e.g., at next cipher rounds), we are interested in achieving the equiprobability of output sharings as well. This can either be achieved by injecting randomness or maintaining the equiprobability by a clever design of the TI functions $\mathsf{F}(.)$. This leads to a third property: a TI function $\mathsf{F}(.)$ is said to be uniform, if – given a uniform input – all sharings of the output occur with equal probability:

$$\forall x \in \mathbb{F}_2^n,\ \exists c_x,\ \forall X \in \int(x),\ \forall Y \in \int(\mathsf{f}(x));\ Pr(\mathsf{F}(X) = Y) = c_x.$$

Unfortunately, no uniform TI of the AES S-box is known. If uniformity is violated, the security proof of TI no longer holds.

Higher-Order Masking. In 2015 De Cnudde *et al.* extended the TI to higher-order of protection for AES [9] after Reparaz showed that introducing fresh randomness is always necessary to achieve multivariate higher-order security [27]. However, as our focus is zero fresh randomness we limit ourselves to first-order security.

Changing of the Guards. In [11], Daemen questioned the assumption that uniformity must be achieved in every S-box separately, and instead suggested a scheme to generate uniformity of the entire S-box layer at once. We illustrate his scheme for four shares. Let $(\mathsf{S}^A, \mathsf{S}^B, \mathsf{S}^C, \mathsf{S}^D)$ be a **non-uniform** TI of the bijective S-box $\mathsf{s}(.)$. Furthermore, let the entire non-linear layer consist of parallel executions of t times the same S-box where we denote the inputs to the i-th shared S-box as $(a_i, b_i, c_i, d_i), i \in \{1, \ldots, t\}$. Then *Changing of the Guards* with four shares is defined as

$$\begin{aligned}
A_i &= \mathsf{S}^A(b_i, c_i, d_i) \oplus c_{i-1} \oplus d_{i-1}, && i > 0 && \\
B_i &= \mathsf{S}^B(c_i, d_i, a_i) \oplus d_{i-1}, && i > 0, && B_0 = c_t \\
C_i &= \mathsf{S}^C(d_i, a_i, b_i) \oplus b_{i-1}, && i > 0, && C_0 = d_t \\
D_i &= \mathsf{S}^D(a_i, b_i, c_i) \oplus b_{i-1} \oplus c_{i-1}, && i > 0, && D_0 = b_t
\end{aligned}$$

As the only source for extra randomness, b_0, c_0, and d_0 need to be provided beforehand. All further guards are computed from the input of each previous S-box in the layer as illustrated in Figure 1, and B_0, C_0, and D_0 provide the guards for the next S-box layer. More precisely, the relabeling of b_2, c_2 and d_2 as D_0, B_0 and C_0 is only done to fulfill the formal non-completeness requirement that ouput share B does not depend on input share b and so forth. This sharing of the S-box layer inherits correctness and non-completeness from the TI of $\mathsf{s}(.)$ and in addition achieves uniformity due to the additional guards. A formal proof of uniformity for an arbitrary number of shares can be found in the original paper [11].

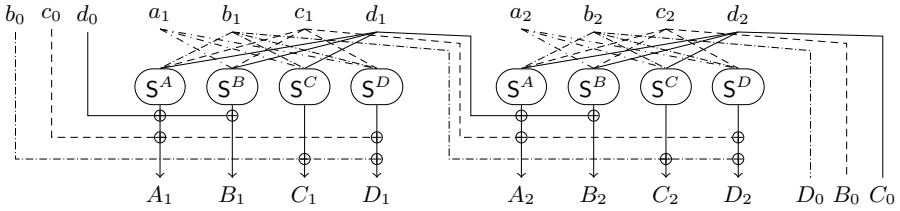


Figure 1: Illustration of *Changing of the Guards* with four shares and two S-boxes.

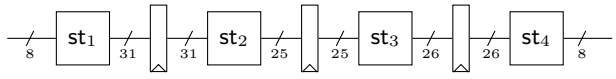
3 Insecurity of a Construction of [14]

We describe the theoretical background of a flaw in the "TI design with 4 shares and no randomness" presented in [14] and demonstrate significant leakage in a practical setting.

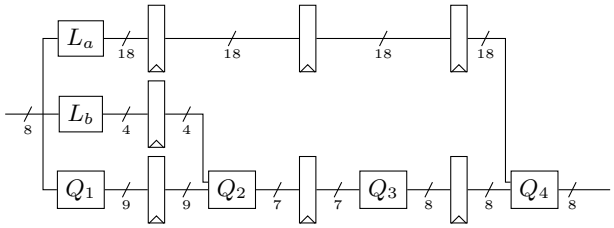
Construction. Based on the smallest known unprotected AES S-box representation by Boyar and Peralta [6], Ghoshal and De Cnudde suggested an allegedly first-order secure implementation by representing the circuit by only 2-bit XOR and AND gates, and dividing the circuit into four stages, each of which with algebraic degree of 2, i.e., quadratic (cp. Figure 2). They made use of a previously-known uniform four-share TI of the 2-input AND gate [25], and concluded that the uniformity of each separate gate is sufficient to achieve the uniformity of each quadratic stage, and consequently of the entire S-box.

We remind that it is not possible for any function $\mathsf{f} : \mathbb{F}_2^n \to \mathbb{F}_2^m,\ m > n$ to be shared uniformly without introducing fresh randomness. More specifically, a sharing of $\mathsf{f}(.)$ with d input and output shares has the form $\mathsf{F} : \mathbb{F}_2^{n \cdot d} \to \mathbb{F}_2^{m \cdot d}$. Since $m \cdot d > n \cdot d$, this function is not surjective and therefore output shares cannot be equiprobable. Hence, stages st_1 and st_3 cannot be shared uniformly as seen in Figure 2a. An investigation of the separate building blocks shows that the violation of uniformity does not stem from a lack of joint uniformity

of all components in a stage, but even the components Q_1 and Q_3 in Figure 2b cannot individually be shared uniformly, because of their higher output dimension. Furthermore, we verified computationally that the entire shared S-box (with 4 shares) is not a bijection over 4×8 bits, therefore it is not uniform. Hence, the requirement of equiprobable input sharing of TI is violated from stage two (st_2) on.



(a) High-level Structure



(b) Low-level Structure

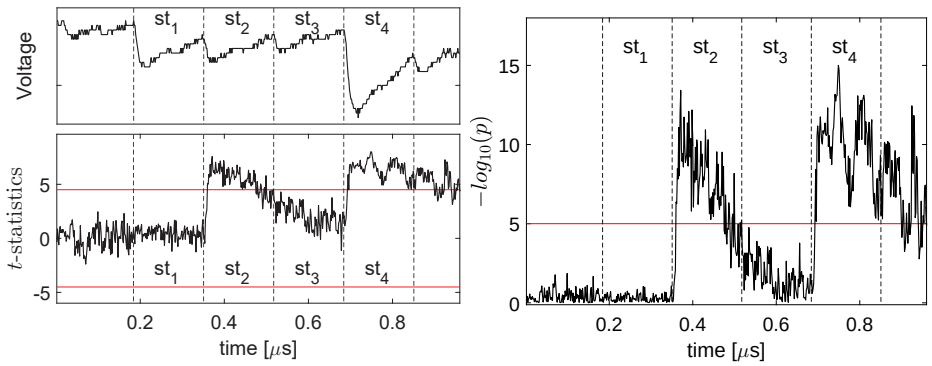
Figure 2: Illustration of the high-level and low-level structure of the S-box used by Ghoshal and De Cnudde [14]. L_a and L_b denote linear building blocks, while Q_1, Q_2, Q_3 and Q_4 indicate quadratic functions.

SCA Evaluation. The above-given justification makes indeed the practical results shown in [14] questionable. This may be due to the parallel use of many S-boxes which have increased the noise level and led to no detectable leakage. We re-evaluated the first-order leakage of this construction with only one instance of the S-box and observed leakage in stages two, three and four in accordance with our theoretical observations.

For all practical analyses we report in this work, we made use of the SAKURA-G board [1] (with Spartan-6 FPGA), and collected the power traces at a sampling rate of 625 MS/s using a digital oscilloscope with 350 MHz bandwidth by measuring the output of the AC amplifier embedded on the SAKURA-G. The target FPGA, on which the design is being run and measured, was operating at the clock frequency of 6 MHz. As the evaluation metric, we applied the common fixed-versus-random t-test [15] to examine the existence of detectable leakage. We further followed the procedure suggested in [29] and examined the uniformity of the random values used to initially mask the input. The result of such an evaluation on the aforementioned four-share S-box of [14] is shown in Figure 3.

We would like to highlight that in several related works, where the Welch's t-test is applied, the probability (to reject the null hypothesis) is not calculated. Instead, by ignoring the "degree of freedom" the threshold of 4.5 based on

the relation $p = 2F(-4.5, v > 1000) < 10^{-5}$ is taken, where $F(.)$ stands for the cumulative student's t distribution function and v the degree of freedom (see [29]). Since v can be different at various sample points, we estimated the probability (so-called p-value) by $p = 2F(-|t|, v)$, and similarly set the threshold to 10^{-5}.



(a) t-statistics **(b)** p-value

Figure 3: Evaluation of the four-share S-box (with no extra randomness) of [14] using 10 million traces.

4 Technique

In the following we illustrate how to decompose the AES S-box into two independently-shared stages and how to reuse the same hardware for several operations without introducing leakage or requiring fresh randomness.

Decomposition. The AES S-box consists of an inversion in $\mathbb{GF}(2^8)$ and a subsequent affine mapping $\mathsf{Aff}(.)$. As $(\mathbb{GF}(2^8), \cdot)$ forms a cyclic group with order 255, we can alternatively represent inversion as $x^{-1} = x^{254}$. Following the idea presented in [24], we show the below observation.

Observation 1 *Let* $\mathsf{f} : \mathbb{GF}(2^8) \to \mathbb{GF}(2^8)$ *be a monomial* $\mathsf{f}(x) = x^k$. *Then*

- *the algebraic degree of* $\mathsf{f}(.)$ *is defined by* $w(k)$, *and*
- $\mathsf{f}(.)$ *is not a bijection for* $w(k) = 2$,

where $w(k)$ *stands for the number of '1's in the binary representation of* k, *i.e., Hamming weight.*

As the inversion is a bijection, it cannot be decomposed into quadratic monomials. In other words,

$$\nexists\ k_1, \ldots, k_n; \forall i, w(k_i) = 2, x^{254} = \left(\left(\left(x^{k_1}\right)^{\cdot}\right)^{\cdot}\right)^{k_n}.$$

Hence, we focus on decomposition into cubic monomials

$$\mathsf{f}(x) = x^p,\ \mathsf{g}(x) = x^q,\ w(p) = w(q) = 3,$$
$$x^{254} = (x^p)^q = (x^q)^p = \mathsf{g} \circ \mathsf{f}(x) = \mathsf{f} \circ \mathsf{g}(x).$$

Such a search can be reduced to

$$\forall (p, q),\ w(p) = w(q) = 3;\ p \cdot q = 254 \mod 255.$$

By brute force it follows that all such (p, q) tuples are given as[1]

$$(13, 98), (26, 49), (52, 152), (104, 76), (208, 38), (161, 19), (67, 137), (134, 196).$$

Each tuple yields a decomposition of the AES S-box into two stages

$$(\mathsf{f}(.),\ \ \mathsf{Aff} \circ \mathsf{g}(.)).$$

As the size of a direct sharing [26] grows with the number of non-linear terms in the Algebraic Normal Form (ANF) of a function, we compared all monomials in the ANF of all above tuples. We determined that the ANFs for all tuples contain all monomials up to third order. Hence, the choice of the specific tuple is not crucial for area minimization.

As we use the UMC 180 nm standard library for all our ASIC syntheses, we verified $(p, q) = (26, 49)$ as the smallest choice by synthesizing the corresponding TI circuit of all above tuples (made by direct sharing).

4.1 S-box Construction

We construct a first-order TI of the AES S-box by separately applying direct sharing to the cubic components

$$\mathsf{f}(x) = x^{26},\ \mathsf{g}(x) = \mathsf{Aff}(x^{49}).$$

A naive construction that realizes the entire S-box in two cycles, would exhibit a prohibitively large area of more than 20 k GE. Hence, we follow two serialization methods, originally suggested in [23] for the PRESENT [5] S-box, to achieve an area reduction. We will refer to these methods as *serializing shares* and *serializing stages* and demonstrate the applicability to the AES S-box.

Serializing Shares. By applying direct sharing, all component functions become identical as long as we rotate the input shares in the following manner:

$$\mathsf{F}^A(X) = \mathsf{F}^*(b, c, d), \qquad \mathsf{F}^B(X) = \mathsf{F}^*(c, d, a),$$
$$\mathsf{F}^C(X) = \mathsf{F}^*(d, a, b), \qquad \mathsf{F}^D(X) = \mathsf{F}^*(a, b, c).$$

This allows us to compute all output shares with the same circuit, denoted as F^*, one after each other in a sequential manner (see Figure 4a).

[1] Obviously, for each (p, q) tuple, (q, p) is also a valid tuple.

Non-completeness. Special care must be taken in such a serialized architecture to not violate the non-completeness property. From a high-level point of view the multiplexer is a combinatorial circuit and should only depend on at most $d-1$ shares, which makes any such a serial design impossible. We overcome this challenge by

- making sure that each select signal is directly derived from a register,
- suggesting a special design that uses *Gray Code* for the select signals, and
- placing a register right after each multiplexer.

The choice of the Gray Code implies that only one select signal changes at each state transition. This guarantees the joint leakage of at most two shares at each clock cycle. As an example, considering Figure 4b, suppose that the select signals $(s_1, s_0) = 10$ (i.e., the b input is selected). At the next state, the select signals change to $(s_1, s_0) = 11$ leading to selecting the c input. In this state transition – due to supplying the select signals directly by registers – the combinatorial logic relevant to selecting the a and d inputs stay inactive. It holds for the other transitions $a \rightarrow b$, $c \rightarrow d$, and $d \rightarrow a$. Hence, with respect to Figure 4c, the input tuples (b, c, d), (c, d, a), (d, a, b), and (a, b, c) are sequentially selected.

Since the shared function $\mathsf{F}^*(.)$ non-linearly combines three input shares, without having a register at its input, the circuit would potentially exhibit first-order leakage between the input transitions. By means of such a register, we reset the input of the shared function $\mathsf{F}^*(.)$ to ZERO between each input transition. In other words, when the exemplary input tuple (b, c, d) is selected and the corresponding shared output is calculated, at the next clock cycle the input tuple (c, d, a) is selected and at the same time the input registers of $\mathsf{F}^*(.)$ are reset. At the next clock cycle, the selected tuple is not changed and is stored by the input registers thereby evaluating the corresponding output share. This implies 8 clock cycles for the entire computation of the 4 output shares.

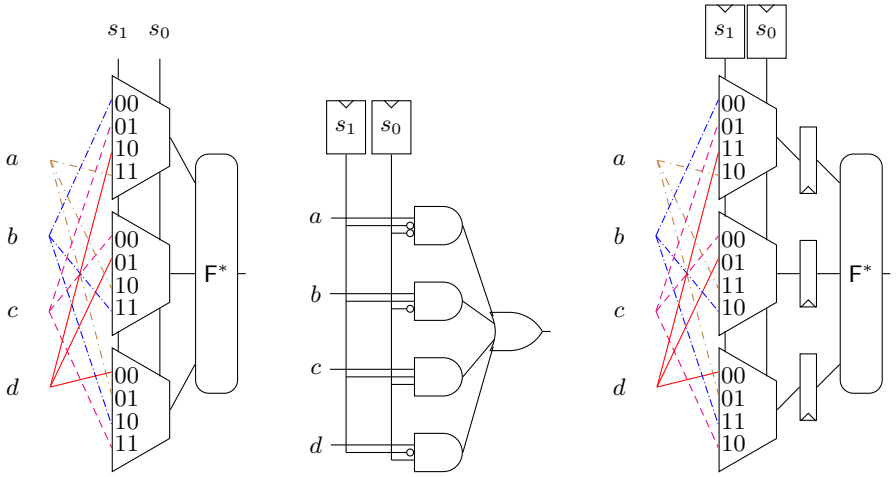
In contrast, both original work [23] and the recent suggestion by Gupta *et al.* [19], which applies serializing shares to GIFT by using the initial naive design seen in Figure 4a, violate the non-completeness.

Serializing Stages. As stated, both stages $\mathsf{f}(.)$ and $\mathsf{g}(.)$ are cubic functions. Considering their ANF, we can create a function $\mathsf{m}(.)$ that realizes all linear, quadratic and cubic terms of both $\mathsf{f}(.)$ and $\mathsf{g}(.)$. Afterwards, by means of two linear layers which combine (i.e., XOR) the corresponding terms, the specific functions $x \rightarrow x^{26}$ and $x \rightarrow \mathsf{Aff}(x^{49})$ can be realized.

More specifically, let $\mathsf{c}_1(.), \mathsf{c}_2(.)$ be two arbitrary cubic functions over the same number of bits. Each of them can be decomposed into a common non-linear layer $\mathsf{m}(.)$, followed by an individual linear layer $\mathsf{l}_1(.)$ (resp. $\mathsf{l}_2(.)$) as

$$\mathsf{c}_1(x) = \mathsf{l}_1 \circ \mathsf{m}(x), \;\; \mathsf{c}_2(x) = \mathsf{l}_2 \circ \mathsf{m}(x).$$

A serialization of stages can be achieved by connecting the output of $\mathsf{m}(.)$ to both $\mathsf{l}_1(.)$ and $\mathsf{l}_2(.)$ and attaching either a multiplexer or registers with different



(a) naive evaluation **(b)** non-complete MUX **(c)** non-complete share-serial evaluation

Figure 4: (a) A naive serialization of shares violates non-completeness. The select signals are not taken from registers and might glitch. The transition from input share $b \rightarrow c$ and $d \rightarrow a$ require both select signals to change. No register stage prevents the function $\mathsf{F}^*(.)$ to combine all shares, (b) Illustration of a multiplexer based on Gray Code with select signals are being directly supplied by registers, (c) Illustration of the entire non-complete share-serial evaluation of function $\mathsf{F}^*(.)$.

enable signals to their outputs. In our case, $\mathsf{l}_1(.)$ is the linear layer belonging to $x \rightarrow x^{26}$ and $\mathsf{l}_2(.)$ to $x \rightarrow \mathsf{Aff}(x^{49})$. Note that we represent the four-share version of these functions by $\mathsf{M}(.)$, $\mathsf{L}_1(.)$, and $\mathsf{L}_2(.)$.

Putting it Together. Following both serialization methods, we can now describe the S-box construction in detail (cp. Figure 5). At the start of the S-box evaluation the non-completeness register `ncR` has already been set to ZERO and the first multiplexer stage selects the lower inputs, i.e., (a, b, c, d). In the first cycle the non-completeness multiplexers `ncM` choose the shares (b, c, d) from the four-byte inputs (a, b, c, d) and the same values are written to the register gds for later use as guards. In cycle two, $\mathsf{L}_1 \circ \mathsf{M}$ which corresponds to $x \rightarrow x^{26}$ is evaluated, while only the register $\mathtt{A}_1$ is enabled to store the result, i.e., one output share of the application of $x \rightarrow x^{26}$ before being XORed with the guards gd_d. Subsequently, at the next clock cycle the `ncR` register is reset, and at the same time (c, d, a) is selected by the `ncM` multiplexers. Analogously, it takes two cycles to write the results for shares B, C, and D of $x \rightarrow x^{26}$ to registers $\mathtt{B}_1$, $\mathtt{C}_1$, and $\mathtt{D}_1$. After eight cycles, the secure evaluation of $x \rightarrow x^{26}$ is complete and the left-most multiplexer stage selects the upper input. Following the same procedure by resetting the `ncR` register and evaluating $\mathsf{L}_2 \circ \mathsf{M}$ the registers $\mathtt{A}_2, \ldots, \mathtt{D}_2$ are subsequently set to the value of the shares $x \rightarrow \mathsf{Aff}(x^{49})$ after achieving unifor-

mity by XORing with the guards stored in gds from the previous evaluation of $x \to x^{26}$. In total, after sixteen cycles the secure evaluation of a shared AES S-box can be read from $\mathtt{A}_2, \ldots, \mathtt{D}_2$ while a reordering of the values in registers $\mathtt{C}_1, \mathtt{D}_1, \mathtt{B}_1$ provides the guards for the next S-box evaluation (this corresponds to the definition given in [11]). We should highlight that all select signals controlling the multiplexers are derived by dedicated registers.

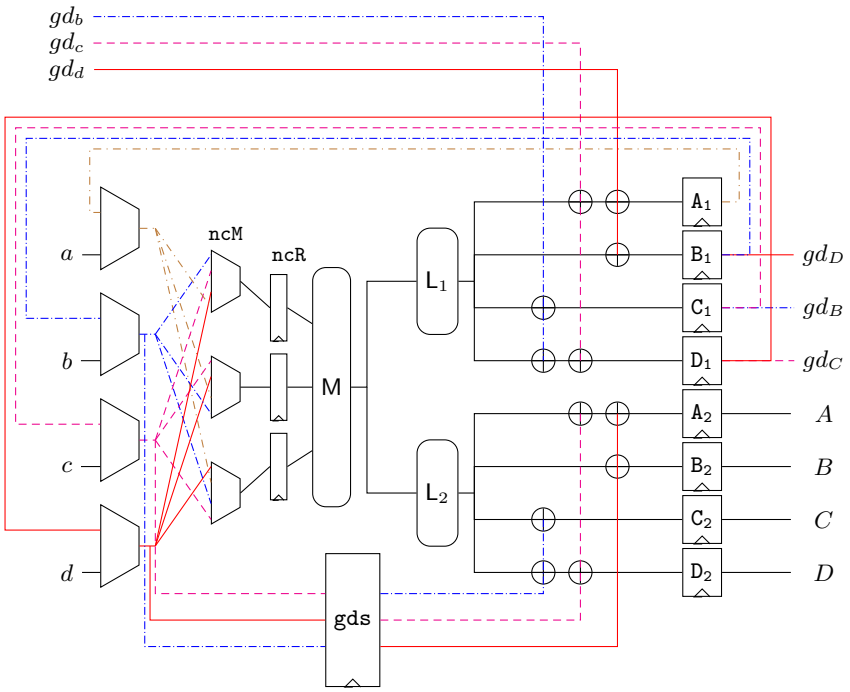


Figure 5: Fully-Serial Design of the AES S-box. In one clock cycle three of the four input shares $a, \ldots, d$ are selected by multiplexers to be fed into registers leading to $\mathsf{M}(.,.,.)$ which realizes all cubic, quadratic and linear terms. In the first eight cycles the output of $\mathsf{M}(.,.,.)$ is processed by $\mathsf{L}^1(.)$ to realize the first shared function and write the results to the upper registers $\mathtt{A}_1, \ldots, \mathtt{D}_1$. The registers feeding $\mathsf{M}(.,.,.)$ are reset after computation of each output share. In the successive eight cycles the same instance of $\mathsf{M}(.,.,.)$ is reused and its output is fed into $\mathsf{L}_2(.)$ to realize the second shared function. In sum, after sixteen clock cycles a uniform sharing of the AES Sbox is written to the output registers.

4.2 Full AES

Based on the 8-bit version of the encryption-only design [21], we constructed a two-share AES implementation (cp. Figure 6) with one S-box instance, a byte-wise shift register to hold the state and an unprotected key schedule.

Number of Shares. As our first-order S-box operates on four-shares, it would be natural to construct a four-share version of AES. However, sharing the entire AES state with four shares leads to a prohibitive area requirement. This motivated us to develop a heuristic that re-masks the state with itself to generate four shares from two shares. Our scheme is based on a trivial method to extend two shares (a_0, b_0) to four shares (a, b, c, d) by the introduction of two additional random bytes (r_1, r_2) via re-masking:

$$a = r_1, \quad b = r_1 \oplus a_0, \quad c = r_2, \quad d = r_2 \oplus b_0.$$

Extension. As we want to achieve zero per-round randomness, we chose a share of locally-independent state bytes as (r_1, r_2). To determine the independence of bytes, we remind that *MixColumns* is the only part of the AES round function that intermingles multiple bytes. Hence, it is the only cause of dependence between state bytes. Further, due to the diffusion properties of the AES, the combination of *ShiftRows* and *MixColumns* causes any difference to any single byte to propagate to all other bytes within only two rounds. In consequence, we can only aim at finding a heuristic to judge local independence of state bytes. We chose to look one round into the past and one round into the future to pick the state bytes that did not originate from the same *MixColumns* operation and are not used jointly in the next *MixColumns* operation. In consequence, we found that the output of the shift register (given to the S-box) can be masked with the byte at offset 2 and the byte at offset 9 (cp. Figure 6) counted against the direction of the shift register starting from zero. This can easily be verified by iterating through all 16 states of the shift register, e.g., in the default position, byte a_0 is masked with a_8 and b_0 is masked with b_6 corresponding to columns zero, two and one in the previous round and columns zero, two and three in the next round. We have chosen two different positions in the two shares of the state to avoid undesired unmasking of any byte in the state. Needless to say that the purely heuristic nature of remasking bytes with locally-independent bytes demands for an in-depth practical evaluation (cp. Section 5).

Reduction. We need to securely reduce the four-share S-box output (A, B, C, D) to two shares to write it back to the state register. As the S-box already contains a final register stage for each share $A_2, \ldots, D_2$ (cp. Figure 5) we can directly reduce the four shares to two shares by XORing them without the need of an additional register stage:

$$a_{15} \Leftarrow A \oplus B, \quad b_{15} \Leftarrow C \oplus D.$$

After power-up of the circuit, only for the first AES call a 24-bit random value should be provided for the guards, and it is the only randomness required for the circuit. During the operation of the S-box, the guard register is updated, and no extra randomness will be required. The next AES calls will make use of the last value stored in the guard register corresponding to the last AES call.

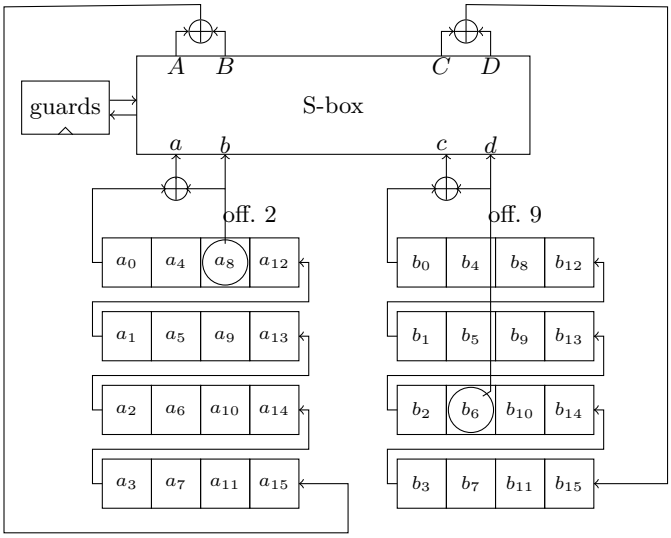


Figure 6: Two-share state of AES operates with a four-share S-box. The guards are initially provided. The linear layer is omitted for brevity.

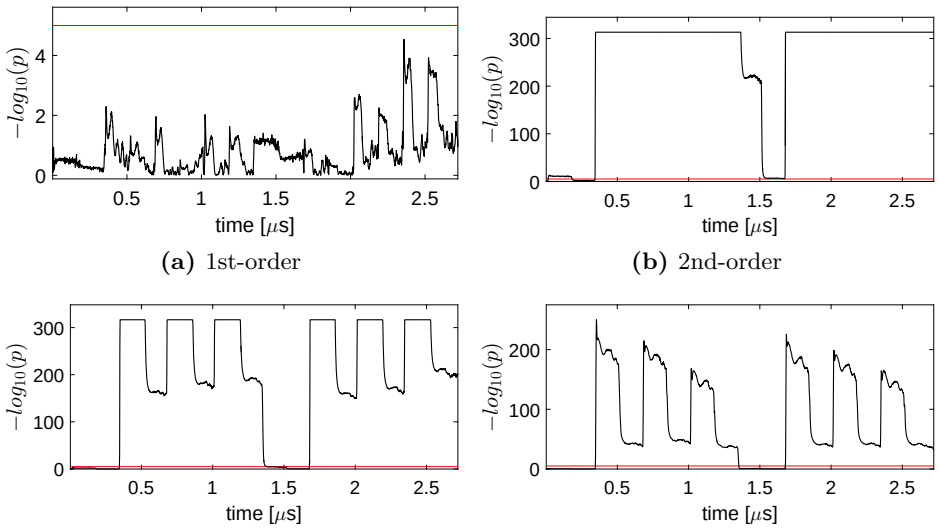
5 Practical Analysis

For the practical analyses we used the same setup of the analysis presented in Section 3. We first investigated a single S-box of our design in a similar way that we analyzed the Ghoshal and De Cnudde construction, with a difference that we provided fresh randomness for the *guards* at the start of each computation of the S-box. The corresponding results – using 100 million traces – are shown in Figure 7, indicating no first-order detectable leakage (p-value $< 10^{-5}$) and strong higher-order leakages.

As the next step, we implemented our two-share AES encryption design with no fresh randomness, and followed the same evaluation procedure. Since the traces are long compared to the previous experiment, we examined this design in two parts: 10 million traces covering the first encryption round, and 10 million traces for the last round. Figure 8 presents the results, where no first-order leakage is detectable.

6 Discussion

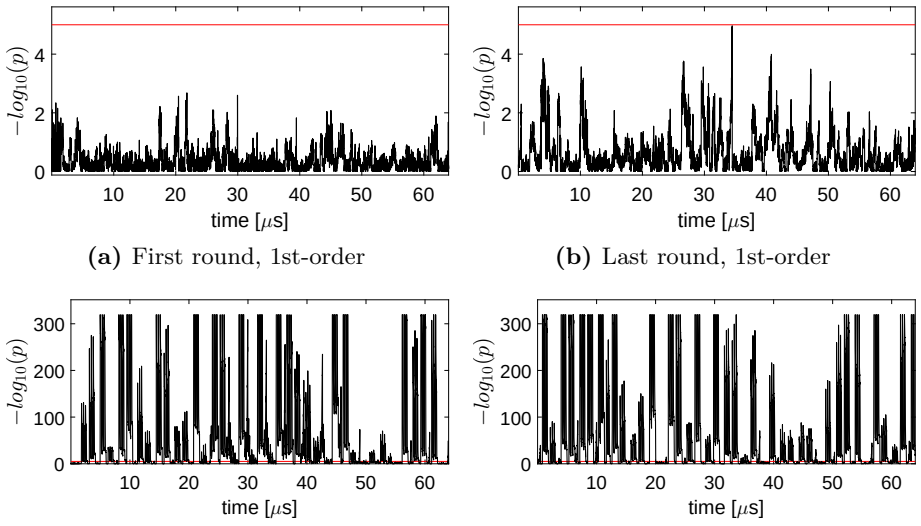
Comparison. We synthesized our design by the UMC 180 nm Standard cell library. Our S-box exhibits a two-fold to three-fold area increase compared to state-of-the-art first-order secure S-boxes. This in turn leads to an AES that occupies only several hundred to 1300 GE more than the status-quo (cp. Table 1) and does not need any fresh randomness. This comes at the cost of a greatly increased latency (eleven-fold) compared to the state-of-the-art implementations.



(a) 1st-order **(b)** 2nd-order

(c) 3rd-order **(d)** 4th-order

Figure 7: Evaluation of our four-share S-box, using 100 million traces.



(a) First round, 1st-order **(b)** Last round, 1st-order

(c) First round, 2nd-order **(d)** Last round, 2nd-order

Figure 8: Evaluation of our two-share AES encryption design with no fresh randomness, using 10 million traces.

Table 1: Comparison of state-of-the-art first-order secure AES encryption designs with our contribution regarding latency, S-box size, total size and randomness per S-box call.

Design	Latency (cycles)	S-box Size (kGE)	AES Size (kGE)	Rand./S-box (bits)
Bilgin *et al.* [4]	246	2.2	7.2	16
Gross *et al.* [18]	246	2.6	6.0	18
Cnudde *et al.* [10]	276	1.9	6.3	54
Ueno *et al.* [32]	219	1.4	6.3	64
This work	2804	4.2	7.6	0

Nevertheless, we believe the increase in area and latency to be an acceptable trade-off to be able to omit the internal generation of random bits completely. Note that all other implementations need to use either a true- or a pseudo random number generator (or a combination of them) internally, about whose area requirements we do not yet have a clear picture. In short, it is not possible to map the number of required random bits onto a meaningful area requirement of a corresponding RNG module. Hence, the reduction of required random bits is of paramount importance for any masking scheme. For example, there are several works in the area of masking (e.g., [2, 16, 18, 28]) which tried to reduce the required randomness.

Fixing the Construction of [14]. As *Changing of the Guards* is only applicable to bijections, we do not see any possibilities to apply it to the design of [14] or any other design based on the Boyar-Peralta implementation [6] of the AES S-box. Hence, the only suitable countermeasure to achieve uniformity in the construction of [14] appears to be the introduction of fresh randomness in each round.

Distinction from Recent Work. Recently, Gupta *et al.* [19] independently introduced a TI of the light-weight block cipher GIFT [3]. They used a masking method called *combined 3-shares* to decompose the cubic 4-bit S-box of GIFT into two quadratic bijections and apply direct-sharing individually. In contrast to our work, Gupta *et al.* do not discuss the importance of a special multiplexer design that prevents the combination of all three shares at a time. Furthermore, they do not incorporate a register stage after the multiplexers, which again violates non-completeness. Unfortunately, they do not detect this flaw as their leakage evaluation consists of failed attacks on a certain power model instead of a more elaborate leakage detection, e.g., Welsh's t-test. Further, Božilov *et al.* [7] decomposed the PRINCE S-box into two quadratic functions of the same equivalence class and demonstrated a three-share TI design with serialized stages.

7 Conclusion

We introduced the first zero per-round randomness construction of a first-order secure AES S-box which mitigates the fact that generation of randomness on embedded systems remains difficult. Further, we introduced a method to extend two shares to four shares for the evaluation of an S-box using uncorrelated state bytes and without introducing any fresh randomness. We should emphasize that we cannot yet provide any proof for the security of such a combination, which led to competitive size without any requirements on fresh or internal randomness. Instead, we could just practically confirm its first-order security using an FPGA prototype while we showed elementary flaws in an earlier design that claimed to achieve the same property.

Our S-box masking methodology is universally applicable to bijective S-boxes and is the first work that achieves non-completeness in a share-serially architecture of AES, which enables a new area vs. latency trade-off.

Acknowledgments

The work described in this paper has been supported in part by the German Federal Ministry of Education and Research BMBF (grant nr. 16KIS0666 SysKit_HW).

References

1. Side-channel AttacK User Reference Architecture. http://satoh.cs.uec.ac.jp/SAKURA/index.html.
2. Josep Balasch, Sebastian Faust, Benedikt Gierlichs, Clara Paglialonga, and François-Xavier Standaert. Consolidating inner product masking. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 724–754. Springer, 2017.
3. Subhadeep Banik, Sumit Kumar Pandey, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Yosuke Todo. GIFT: A small present - towards reaching the limit of lightweight encryption. In Fischer and Homma [13], pages 321–345.
4. Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. Trade-offs for threshold implementations illustrated on AES. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 34(7):1188–1200, 2015.
5. Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In *CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007.
6. Joan Boyar, Philip Matthews, and René Peralta. Logic minimization techniques with applications to cryptology. *J. Cryptology*, 26(2):280–312, 2013.
7. Dušan Božilov, Miroslav Knežević, and Ventzislav Nikov. Threshold implementations of prince: The cost of physical security. NIST Lightweight Cryptography Workshop, 2016. https://www.nist.gov/sites/default/files/documents/2016/10/17/bozilov-paper-lwc2016.pdf.

8. David Canright. A very compact s-box for AES. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 441–455. Springer, 2005.
9. Thomas De Cnudde, Begül Bilgin, Oscar Reparaz, Ventzislav Nikov, and Svetla Nikova. Higher-order threshold implementation of the AES s-box. In Naofumi Homma and Marcel Medwed, editors, *Smart Card Research and Advanced Applications - 14th International Conference, CARDIS 2015, Bochum, Germany, November 4-6, 2015. Revised Selected Papers*, volume 9514 of *Lecture Notes in Computer Science*, pages 259–272. Springer, 2015.
10. Thomas De Cnudde, Oscar Reparaz, Begül Bilgin, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. Masking AES with d+1 shares in hardware. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, volume 9813 of *Lecture Notes in Computer Science*, pages 194–212. Springer, 2016.
11. Joan Daemen. Changing of the guards: A simple and efficient method for achieving uniformity in threshold sharing. In Fischer and Homma [13], pages 137–153.
12. Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 423–440. Springer, 2014.
13. Wieland Fischer and Naofumi Homma, editors. *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, volume 10529 of *Lecture Notes in Computer Science*. Springer, 2017.
14. Ashrujit Ghoshal and Thomas De Cnudde. Several masked implementations of the boyar-peralta aes s-box. In *INDOCRYPT 2017*, 2017. to appear.
15. G. Goodwill, B. Jun, J. Jaffe, and P. Rohatgi. A testing methodology for side channel resistance validation. In *NIST non-invasive attack testing workshop*, 2011. http://csrc.nist.gov/news_events/non-invasive-attack-testing-workshop/papers/08_Goodwill.pdf.
16. Hannes Gross and Stefan Mangard. Reconciling d+1 masking in hardware and software. In Fischer and Homma [13], pages 115–136.
17. Hannes Gross, Stefan Mangard, and Thomas Korak. Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order. *IACR Cryptology ePrint Archive*, 2016:486, 2016.
18. Hannes Gross, Stefan Mangard, and Thomas Korak. An Efficient Side-Channel Protected AES Implementation with Arbitrary Protection Order. In *CT-RSA 2017*, volume 10159 of *Lecture Notes in Computer Science*, pages 95–112. Springer, 2017.
19. Naina Gupta, Arpan Jati, Anupam Chattopadhyay, Somitra Kumar Sanadhya, and Donghoon Chang. Threshold implementations of gift: A trade-off analysis. Cryptology ePrint Archive, Report 2017/1040, 2017. https://eprint.iacr.org/2017/1040.
20. Yuval Ishai, Amit Sahai, and David A. Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.

21. Jérémy Jean, Amir Moradi, Thomas Peyrin, and Pascal Sasdrich. Bit-sliding: A generic technique for bit-serial implementations of spn-based primitives - applications to aes, PRESENT and SKINNY. In Fischer and Homma [13], pages 687–707.
22. Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In *CRYPTO 1999*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.
23. Sebastian Kutzner, Phuong Ha Nguyen, Axel Poschmann, and Huaxiong Wang. On 3-share threshold implementations for 4-bit s-boxes. In Emmanuel Prouff, editor, *Constructive Side-Channel Analysis and Secure Design - 4th International Workshop, COSADE 2013, Paris, France, March 6-8, 2013, Revised Selected Papers*, volume 7864 of *Lecture Notes in Computer Science*, pages 99–113. Springer, 2013.
24. Amir Moradi. Advances in Side-Channel Security. Habilitation thesis, Ruhr-Universität Bochum, 2016.
25. Svetla Nikova, Christian Rechberger, and Vincent Rijmen. Threshold implementations against side-channel attacks and glitches. In Peng Ning, Sihan Qing, and Ninghui Li, editors, *Information and Communications Security, 8th International Conference, ICICS 2006, Raleigh, NC, USA, December 4-7, 2006, Proceedings*, volume 4307 of *Lecture Notes in Computer Science*, pages 529–545. Springer, 2006.
26. Svetla Nikova, Vincent Rijmen, and Martin Schläffer. Secure Hardware Implementation of Nonlinear Functions in the Presence of Glitches. *J. Cryptology*, 24(2):292–321, 2011.
27. Oscar Reparaz. A note on the security of higher-order threshold implementations. *IACR Cryptology ePrint Archive*, 2015:1, 2015.
28. Oscar Reparaz, Begül Bilgin, Svetla Nikova, Benedikt Gierlichs, and Ingrid Verbauwhede. Consolidating masking schemes. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 764–783. Springer, 2015.
29. Tobias Schneider and Amir Moradi. Leakage Assessment Methodology - A Clear Roadmap for Side-Channel Evaluations. In *CHES 2015*, volume 9293 of *Lecture Notes in Computer Science*, pages 495–513. Springer, 2015.
30. Elena Trichina. Combinational logic design for AES subbyte transformation on masked data. *IACR Cryptology ePrint Archive*, 2003:236, 2003.
31. Rei Ueno, Naofumi Homma, and Takafumi Aoki. A systematic design of tamper-resistant galois-field arithmetic circuits based on threshold implementation with (d + 1) input shares. In *47th IEEE International Symposium on Multiple-Valued Logic, ISMVL 2017, Novi Sad, Serbia, May 22-24, 2017*, pages 136–141. IEEE Computer Society, 2017.
32. Rei Ueno, Naofumi Homma, and Takafumi Aoki. Toward more efficient dpa-resistant AES hardware architecture based on threshold implementation. In Sylvain Guilley, editor, *Constructive Side-Channel Analysis and Secure Design - 8th International Workshop, COSADE 2017, Paris, France, April 13-14, 2017, Revised Selected Papers*, volume 10348 of *Lecture Notes in Computer Science*, pages 50–64. Springer, 2017.