One More Motivation to Use Evaluation Tools This Time for Hardware Multiplicative Masking of AES

Hemin Rahimi, Amir Moradi
Technische Universität Darmstadt
Darmstadt, Germany
{firstname.lastname}@tu-darmstadt.de

***Abstract*—Safeguarding cryptographic implementations against the increasing threat of Side-Channel Analysis (SCA) attacks is essential. Masking, a countermeasure that randomizes intermediate values, is a cornerstone of such defenses. In particular, SCA-secure implementation of AES, the most-widely used encryption standard, can employ Boolean masking as well as multiplicative masking due to its underlying Galois field operations. However, multiplicative masking is susceptible to vulnerabilities, including the zero-value problem, which has been identified right after the introduction of multiplicative masking. At CHES 2018, De Meyer et al. proposed a hardware-based approach to manage these challenges and implemented multiplicative masking for AES, incorporating a Kronecker delta function and randomness optimization.**

In this work, we evaluate their design using the PROLEAD evaluation tool under the glitch- and transition-extended probing model. Our findings reveal a critical vulnerability in their first-order implementation of the Kronecker delta function, stemming from the employed randomness optimization. This leakage compromises the security of their presented masked AES Sbox.

After pinpointing the source of such a leakage, we propose an alternative randomness optimization to address this issue, and demonstrate its effectiveness through rigorous evaluations by means of PROLEAD.

***Index Terms*—Side Channel Analysis, Implementation Security, Multiplicative Masking, AES**

I. Introduction

Protecting confidential information has always been crucial. Today, safeguarding data and securing both software and hardware implementation of cryptographic primitives are more critical than ever to prevent sensitive information from being exposed by Side-Channel Analysis (SCA) attacks [1]. Numerous countermeasures have been devised to address these attacks. Masking, introduced by Chari et al. in 1999 [2], is among the most widely used and recognized techniques. It is highly regarded due to its strong theoretical basis and the possibility to prove its security under specific adversarial models.

Boolean masking is straightforward to apply on linear functions because the XOR operation with a mask is preserved through the function, allowing the masked computation to maintain its structure and security without complicating the underlying process. Therefore, most of the relevant state-of-the-art research focuses on applying Boolean masking to non-linear Boolean functions, such as the Sboxes in symmetric block ciphers. However, utilizing Boolean masking for the non-linear AES Sbox is inefficient because the Sbox involves complex operations, like inversion in a Galois field, which does not preserve the linearity required for straightforward masking. This results in the need for additional, complex countermeasures to maintain security, leading to increased computational overhead and implementation complexity.

An alternative approach is to apply multiplicative masking to the AES Sbox. The application of this type of masking to the Galois field inversion of the AES Sbox effectively addresses the previously mentioned challenges. However, this requires conversion between Boolean and multiplicative masking at the beginning and end of the inversion. Further, it has been shown in [3] that multiplicative masking can be easily broken due to its inability to mask zero values, as the multiplication of zero with any mask leads to zero. This issue has been addressed by a series of work [4]–[6] showing that zero being the Sbox input can be mapped to 1 and turned back to zero after the inversion avoiding the zero-value issue. This approach requires tracking zero values, and more importantly, the entire tracking and mappings should be masked as well. Note that this technique has been applied in (embedded) software applications, resulting in secure and efficient masked software implementations of AES.

Masking in hardware, however, involves many additional challenges. Mangard et al. [7] were the first to report a critical weakness in masked hardware implementation related to unintended transitions at the output of gates, known as glitches. Due to this vulnerability, the vast majority of proposed masked hardware designs have become insecure, as they overlook the impact of glitches in their constructions. By considering the effect of glitches, Nikova et al. [8] proposed the first methodology, named Threshold Implementation (TI), successfully maintaining first-order SCA security which forces a high number of shares depending on the algebraic degree of the target function. Notable examples of implementations based on this approach include [9] and [10]. However, its application on the AES Sbox is not straightforward and leads to a significant area overhead. Subsequently, Domain-Oriented Masking (DOM) was introduced by Gross et al. [11], offering the same level of security as TI with the minimum number of shares and hence lower area and randomness overhead.

To construct a masked hardware implementation of the AES Sbox, De Meyer et al. [12] took a different approach compared to prior similar designs such as those shown in [10], [11], [13]. More specifically, their research builds upon works done by

Golić and Tymen [3] and Genelle et al. [5], which introduced adaptive masking technique involving switching between different masking schemes and managing the zero problem. The core idea of the work by Genelle et al. is to track and map zero values, and De Meyer et al. extended this concept to hardware by applying the well-known DOM scheme to a Kronecker delta function. This approach reduces the area overhead and the need for fresh masks compared to the state-of-the-art. Additionally, they proposed a custom optimization to reduce randomness further by strategically recycling fresh masks according to a specific procedure.

In this work, we conduct a comprehensive security evaluation of the masked AES Sbox and its corresponding modules as detailed in [12], employing the PROLEAD evaluation tool [14]. In addition to the security evaluations, we carefully analyzed the components to identify potential vulnerabilities and their root causes. Following this procedure, we have identified first-order leakage in the first-order design of [12] originating from the randomness optimization technique employed in the masked Kronecker delta function aiming at reducing the number of fresh masks. Finally, we present another optimization technique to address these weaknesses maintaining first-order security under glitch- and transition-extended probing models.

The rest of the paper is structured as follows. The next section delves into the relevant concepts providing the necessary background to follow the paper. The core of the paper comprises in Section III and Section IV, where in addition to the security evaluations and analyses, our proposed optimization technique is presented and assessed. Finally, Section V summarizes our key findings and highlights potential future directions.

II. Background

Notations. In equations, square brackets [.] serve as a register. $\oplus$ and $\otimes$ are used to indicate addition and multiplication in $\mathbb{F}_n = \mathrm{GF}(n)$. AND operations have been presented by "&" and in some cases, this notation has been omitted for brevity. Also, $\mathbb{F}_n^* = \mathbb{F}_n/\{0\}$, and by subscripts x_i we refer to the i-th element (bit) of a vector X, and by superscript x^j to the j-th share of x.

A. Masking

Masking is a countermeasure technique employed to protect sensitive information against SCA attacks, such as power analysis. This technique randomizes intermediate values within a circuit, obscuring the correlation between power consumption and data that is expected to be processed, i.e., intermediates of a cipher. Two primary masking techniques are employed in the AES Sbox implementations: *Boolean* and *multiplicative*. As follows, we present an exploration of these schemes and their applications within the context of AES.

a) Boolean Masking: This method involves splitting sensitive data $X \in \mathbb{F}_n$ into $d > 1$ shares $\langle X^1, ..., X^d \rangle \in (\mathbb{F}_n)^d$. One option to share X is to select $d-1$ shares $\langle X^1, \ldots, X^{d-1} \rangle$ uniformly at random (each over $\mathbb{F}_n$), and set the last share X^d as the XOR result of the original data X with all other shares (random values). In short, we should be able to write

$$X = \bigoplus_{i=1}^{d} X^i. \tag{1}$$

Boolean masking is particularly effective for linear operations $L(X \oplus Y) = L(X) \oplus L(Y)$, as the target function can be directly applied to the shares, i.e., $L(\bigoplus_{i=1}^d X^i) = \bigoplus_{i=1}^d L(X^i)$. For instance, linear operations of AES such as `AddRoundKey`, `MixColumns`, and `ShiftRows` can utilize Boolean masking by instantiating d instances of each operation, while each of which processes a share individually. Despite its suitability for affine operations, it cannot easily handle non-linear operations, such as the AES Sbox. Since the AES Sbox is an inversion in $\mathrm{GF}(2^8)$ followed by an affine transformation $A(.)$

$$S(X) = A\left(X^{-1}\right), \tag{2}$$

one option to deal with this difficulty is to use a mixture of Boolean and multiplicative masking and switch between them [15].

b) Multiplicative Masking: When dealing with the AES Sbox with $S(X \oplus Y) \neq S(X) \oplus S(Y)$, multiplicative masking can be used rather than Boolean masking. Similar to Boolean masking, it involves d shares denoted as $\langle X^1, \ldots, X^d \rangle$. A subset of $d-1$ shares is randomly selected while omitting $\{0\}$, i.e., from $\mathbb{F}_n^*$. The remaining share is computed as the product of the secret value $X \in \mathbb{F}_n$ and all other shares, i.e., the randomly selected values. This construction must adhere to the following equation.

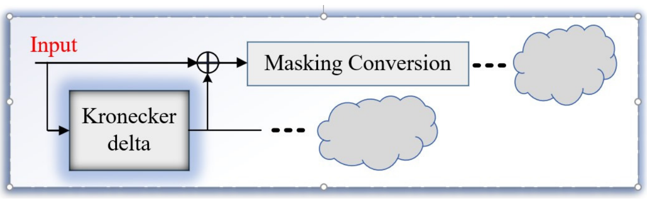
$$X = \left(\bigotimes_{i=1}^{d-1} \left(X^i\right)^{-1} \right) \otimes X^d \tag{3}$$

However, this scheme has a critical flaw known as the *zero-value problem*, first identified in [3]. More specifically, it can only effectively mask $X \in \mathbb{F}_n^*$.

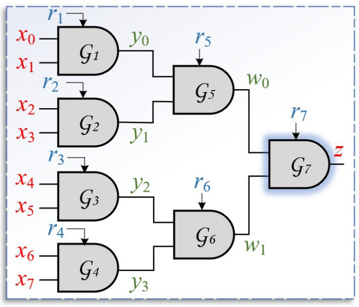
B. Zero-Value Problem in Multiplicative Masking

The zero-value problem occurs when the Sbox input is zero, leaving it unchanged and unmasked in multiplicative masking, compromising security and making the system vulnerable to first-order Differential Power Analysis (DPA) attacks. To solve this, a technique has been introduced in [5], which converts zero inputs to non-zero values (ideally to 1) and tracks them to revert after the Sbox inversion. This conversion and tracking must also be securely masked to prevent leakage. De Meyer et al. [12] applied the same concept in hardware, using the Kronecker delta function to map zero inputs to non-zero values before applying multiplicative masking. The underlying structure is depicted in Fig. 1.

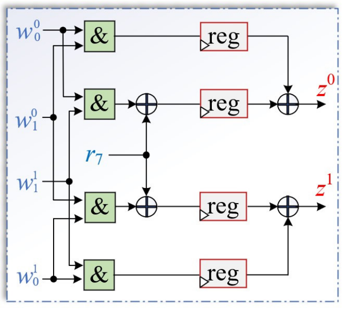
The AES Sbox has been designed based on switching between different types of masking. However, as illustrated in Fig. 1a, detecting and mapping zero inputs (using the Kronecker delta) should be applied before the masking conversion. The output of the Kronecker delta function is "1" (in a Boolean masked form) only when the input is zero. In any other case, its output is "0" (naturally in a Boolean masked form).



(a) Location of the Kronecker delta function in the AES Sbox.



(b) Structure of the Kronecker delta function.



(c) Structure of a masked DOM-AND gate ($\mathcal{G}_7$).

Fig. 1: Overview of the Kronecker delta function within the AES Sbox.

More precisely, the output of the Kronecker delta function is determined by the following equation.

$$z = \overline{x}_0 \;\&\; \overline{x}_1 \;\&\; \ldots \;\&\; \overline{x_7}, \tag{4}$$

with $\overline{x}_0$ being the logical invert of the first input bit x_0 (respectively for the other input bits $x_1, \ldots, x_7$). Note that the concept of mapping zero elements to 1 is directly inspired by the property of the finite field where both zero and 1 are their own inverses. Therefore, we can write

$$(z \oplus X)^{-1} \oplus z = X^{-1}.$$

The Kronecker delta function's first-order design, shown in Fig. 1b, consists of a $\log_2(n)$-level 2-input AND-tree, where the AND gates are based on the DOM-indep multiplier [11]. Each DOM-AND gate (depicted in Fig. 1c) requires a single-bit random input, called fresh mask bit. As illustrated in Fig. 1b the Kronecker structure demands a total of 7 fresh mask bits. Accordingly, to decrease such a demand, the authors of [12] proposed custom optimizations to re-use the fresh masks, i.e., to give the same fresh mask bit to multiple DOM-AND modules. These optimizations are based on the simplified equation of the DOM-indep multiplier as follows.

$$b_z^i = b_x^i\, b_y^i \oplus [b_x^i\, b_y^{i\oplus 1} \oplus r], \text{ for } i \in \{0, 1\},$$

where, $z = x\,y$ and their first-order Boolean mask representations are $z = b_z^0 \oplus b_z^1$, $x = b_x^0 \oplus b_x^1$, and $y = b_y^0 \oplus b_y^1$. Using the associative and commutative properties of XOR, we can write

$$b_z^i = b_x^i\, y \oplus r. \tag{5}$$

Based on the simplified expression in Equation (5), any randomness previously employed to mask y is eliminated from the output share z. Building upon these insights, the authors of [12] proposed the following optimization for their first-order implementation.

$$\begin{aligned} &r_1 = r_3 \xleftarrow{\$} \mathrm{GF}(2), \quad && r_2 = r_4 \xleftarrow{\$} \mathrm{GF}(2), \quad && r_5 \xleftarrow{\$} \mathrm{GF}(2) \\ &r_6 = [r_5 \oplus r_2], && r_7 = r_1 && \end{aligned} \tag{6}$$

This means that the randomness cost is reduced from 7 to 3 bits. Due to the security proof of the Kronecker delta function, the authors opted for a manual or *pen-and-paper* proof instead of relying on any software to exhaustively examine the security of their construction.

Through their manual proof, aligned with the concept of Strong Non-Interference (SNI) [16] and one-time pad transformation [17], the authors demonstrated that their first-order Kronecker structure is 1-SNI, thereby establishing the provable security of their implementation.

C. Sbox Structure

The structure of the AES Sbox (first order) is illustrated in Fig. 2. Following the zero-mapping process (Kronecker delta function), a sequence of operations is applied: 1) Boolean to multiplicative masking conversion, 2) local inversion based on the work presented in [18], and 3) subsequent multiplicative to Boolean masking conversion. More specifically, ignoring the Kronecker delta function, the initial stage involves converting the given input X from Boolean masking with $X = B^0 \oplus B^1$ to multiplicative masking with $X = \left(P^0\right)^{-1} \otimes P^1$. One option to convert $\langle B^0, B^1 \rangle$ to $\langle P^0, P^1 \rangle$ is to write the multiplicative shares as follows.

$$P^0 = R, \qquad\quad P^1 = [B^0 \otimes R] \oplus [B^1 \otimes R]$$

Note that the fresh mask R is randomly selected from the set of non-zero values, i.e., $R \xleftarrow{\$} \mathbb{F}_{256}^*$.

When X is zero, resulting in $B^0 = B^1$, the corresponding P^1 becomes also zero, which results in $\left(P^1\right)^{-1}$ to be also zero. To

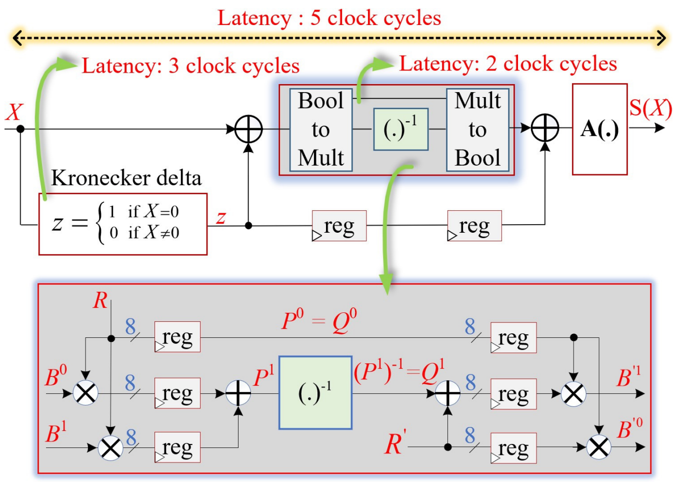


Fig. 2: The entire structure of the masked AES Sbox of [12].

mitigate this potential leakage, input X is preprocessed using the Kronecker delta function before the masking conversion process, whose functionality has been explained previously.

Assuming that $\langle Q^0, Q^1\rangle$ are multiplicative shares of the inversion's output (see Fig. 2), the conversion from multiplicative masking to Boolean masking is done as follows, while $R' \xleftarrow{\$} \mathbb{F}_{256}$.

$$B'^0 = R' \otimes Q^0, \qquad B'^1 = [R' \oplus Q^1] \otimes Q^0$$

Note that, as shown in Fig. 2, some registers are added to this construction to form a pipeline circuit that can process one input per clock cycle.

Finally, an affine transformation (naturally in Boolean masked form) is performed and the final Sbox output is generated. The overall latency of the first-order design is five clock cycles, with three cycles dedicated to the Kronecker and two cycles to the masking conversions. Notably, the affine transformation is fully combinational.

D. Verification Tools

The authors of [12] made a considerable stride in achieving SCA security by presenting the first masked hardware implementation of the AES encryption function using multiplicative masking. One of the critical phases in implementation security is its evaluation and leakage assessment [19], which necessitates formal evaluation and verification tools. Unfortunately, a high percentage of the existing formal verification tools struggle to handle large circuits and implementations [20]–[23]. These limitations pose significant obstacles for researchers attempting to effectively validate their secure designs. For instance, the authors of [12] employed VerMI [24] to verify their masked Sbox implementation. However, when applying their custom optimizations to the Kronecker delta function, they could not reuse VerMI as it mainly examines the non-completeness property of masked designs. Further, the only tool available at that time was maskVerif [20] which in some cases reports the insecurity of secure designs, i.e., false negatives (see [25]).

Given the limitations of the existing verification tools, Müller and Moradi introduced PROLEAD [14], a state-of-the-art evaluation tool specifically designed for masked hardware implementations. The key features of PROLEAD include:

- It operates merely on the gate-level netlist of the given circuit, eliminating the need for a power model.
- It enables the analysis of complete masked cipher implementations, not only small circuits/gadgets.
- It detects vulnerabilities arising from combined glitch and transition effects, as well as higher-order multivariate leakages.
- It avoids false negatives as well as false positives unlike some other tools with the same goal.

As a result, PROLEAD is making major strides in overcoming the limitations of former evaluation tools and filling these gaps. However, it should be noted that it is not a formal verification tool. In other words, if it reports the security of a given design, it cannot be considered as a proof. However, if it detects leakage with high confidence, the insecurity of the circuit can be concluded. Finally, PROLEAD generates a report identifying vulnerable intermediates and categorizing them by leakage susceptibility.

III. Evaluation and Systematic Analysis

In order to ensure a robust framework for analysis, we carefully implemented the complete AES Sbox design of [12] along with its associated modules in Verilog-HDL. To evaluate the first-order security of the design, we made use of the open-source Yosys tool [26] for the synthesis while utilizing the NanGate 45 nm standard cell library to generate the corresponding gate-level netlist. To ensure the integrity of the design, we maintained a hierarchical structure throughout the synthesis process as instructed and suggested by the original authors. For the PROLEAD's evaluations, we have set the tool to conduct a fixed versus random test under the glitch-extended probing model. To ensure a comprehensive evaluation, a total of 4 million simulations were conducted, allowing for a robust statistical analysis of the results.

When excluding the Kronecker delta function and selecting a non-zero input as the fixed value of the test, the design passes the PROLEAD's security assessments. This confirms the correctness and security of the masking conversions, inversion, and affine transformation of the design (see Fig. 2). However, by including the Kronecker delta function and selecting zero as the fixed input, the design failed to pass the PROLEAD's security evaluation. The assessment revealed critical vulnerabilities that compromise security of the implementation. According to PROLEAD's detailed report, these vulnerabilities were directly associated with the Kronecker delta function, which played a pivotal role in the design's failure. The report specifically identified certain intermediate values within the design as leakage points, visually marked with red stars in the gate $\mathcal{G}_7$ for clarity, as shown in Fig. 3.

We should highlight that in these analyses, we applied the randomness optimization technique proposed by the original authors (see Equation (6)). By avoiding such an optimization, i.e., providing 7 individual and independent fresh mask bits per clock cycle for the Kronecker delta function $r_0, \ldots, r_7$, the

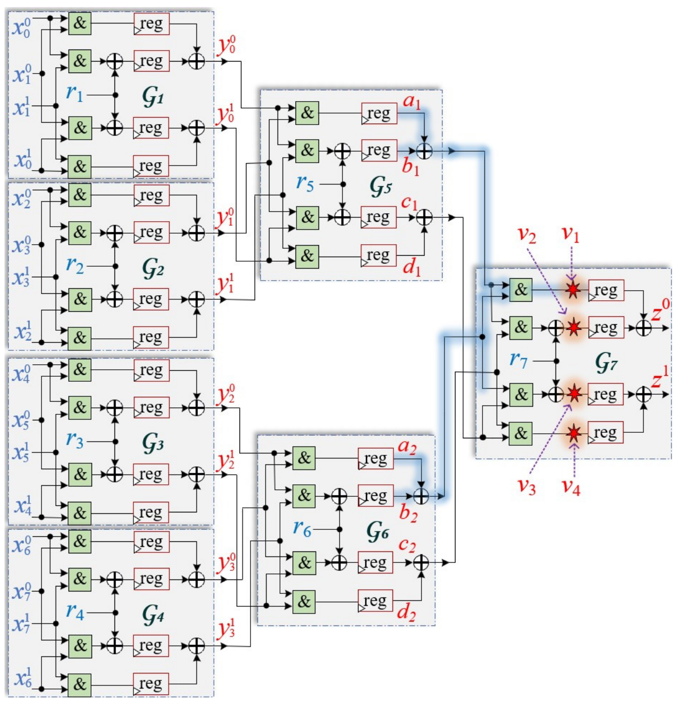


Fig. 3: Kronecker delta function architecture, based on DOM-AND gate.

design passes all PROLEAD's security evaluations. This clearly shows that the applied randomness optimization is the Achilles' heel of the design.

To determine the root cause of these leakages, we need to precisely analyze the interdependencies between inputs, random masks, and the identified leaking probes, while temporarily disregarding the proposed optimization. A preliminary examination of Fig. 3 reveals the following key equations, which shed light on how these factors contribute to the observed vulnerabilities, where the simplifications are based on Equation (5).

$$
\begin{aligned}
y_0^i &= [x_0^i\,x_1^i] \oplus [x_0^i\,x_1^{i\oplus 1} \oplus r_1] = x_0^i\,x_1 \oplus r_1,\\
y_1^i &= [x_2^i\,x_3^i] \oplus [x_2^i\,x_3^{i\oplus 1} \oplus r_2] = x_2^i\,x_3 \oplus r_2,\\
y_2^i &= [x_4^i\,x_5^i] \oplus [x_4^i\,x_5^{i\oplus 1} \oplus r_3] = x_4^i\,x_5 \oplus r_3,\\
y_3^i &= [x_6^i\,x_7^i] \oplus [x_6^i\,x_7^{i\oplus 1} \oplus r_4] = x_6^i\,x_7 \oplus r_4,\\
w_0^i &= [y_0^i\,y_1^i] \oplus [y_0^i\,y_1^{i\oplus 1} \oplus r_5] = y_0^i\,y_1 \oplus r_5,\\
w_1^i &= [y_2^i\,y_3^i] \oplus [y_2^i\,y_3^{i\oplus 1} \oplus r_6] = y_2^i\,y_3 \oplus r_6,\\
z_0^i &= w_0^i\,w_1 \oplus r_7, \quad i \in \{0,1\} \qquad (7)
\end{aligned}
$$

PROLEAD reports indicate that the leakage observed at each node $v_1, \ldots, v_4$ stems from inner domain terms, particularly a_1, a_2, d_1, and d_2, as depicted in Fig. 3. To investigate the source of this leakage more precisely, exemplary consider v_1 and its associated inner domain extensions a_1 and a_2, which are highlighted in Fig. 3.Analyzing these extensions will provide critical insights into the pathways through which the leakage occurs, thereby enhancing our understanding of the underlying vulnerabilities within the design.

Consider a scenario in which a single randomness optimization is implemented, specifically by setting $r_1 = r_3$ (see Equation (6)). This choice simplifies the circuit's behavior and facilitates a more straightforward analysis of its implications. Under this condition, we can derive the following relationships and equations, which highlight how this optimization affects the overall functionality and security of the design. More precisely, we place a probe on v_1 and focus on some of its glitch-extended probes a_1 and a_2 while

$$a_1 = [y_0^0\,y_1^0], \qquad a_2 = [y_2^0\,y_3^0],$$

and

$$
\begin{aligned}
y_0^0 &= x_0^0\,x_1 \oplus r_1, & y_1^0 &= x_2^0\,x_3 \oplus r_2,\\
y_2^0 &= x_4^0\,x_5 \oplus r_1, & y_3^0 &= x_6^0\,x_7 \oplus r_4.
\end{aligned}
$$

Now, let us consider a situation in which a_1 and a_2 both capture the same value $a_1 = a_2 = 1$. This implies that $y_0^0 = y_1^0 = y_2^0 = y_3^0 = 1$. Focusing on $y_0^0 = y_2^0 = 1$, we can write

$$
\begin{aligned}
y_0^0 = y_2^0 &\implies x_0^0\,x_1 \oplus r_1 = x_4^0\,x_5 \oplus r_1\\
&\implies x_0^0\,x_1 = x_4^0\,x_5. \qquad (8)
\end{aligned}
$$

Here, x_0^0 and x_4^0 denote random shares that contribute to the masking process. However, the vulnerability arises when unmasked values $x_1 = x_5 = 0$, which implies this correctness of the equality in Equation (8). In other words, considering the concept behind simulatability [27], the observation achieved by the probe $v_1 \to \{a_1, b_1, a_2, b_2\}$ is not independent of the circuit's unmasked values x_1 and x_5. This phenomenon occurs due to the removal of fresh masks r_1 from both sides of the equation, which eliminates the blinding effect that is crucial for maintaining security.

This analysis can be extended to the other leaky probes v_2, v_3, and v_4. Consequently, re-using randomness within the Kronecker delta function compromises the design's security. It is essential to emphasize that these findings are based on applying only one randomness optimization of Equation (6) in the first AND layer of the Kronecker delta function. Considering other optimizations such as $r_2 = r_4$ could further exacerbate the vulnerabilities identified. For instance, under the previously discussed conditions (i.e., observing $v_1 = 1$), we can derive that $y_1^0 = y_3^0$, which allows us to express the following.

$$
\begin{aligned}
y_1^0 = y_3^0 &\implies x_2^0\,x_3 \oplus r_2 = x_6^0\,x_7 \oplus r_2\\
&\implies x_2^0\,x_3 = x_6^0\,x_7,
\end{aligned}
$$

which implies the same conclusion that the observation made by the probing set $\{a_1, b_1, a_2, b_2\}$ is different when $x_3 = x_7 = 0$.

IV. Our Proposed Optimization

The analysis detailed in Section III illustrates that the randomness optimizations proposed in [12] unintentionally introduce notable vulnerabilities within the design, particularly under the glitch-extended probing model. Although the authors intended to mitigate the zero problem associated with multiplicative masking through the use of the Kronecker delta function, their proposed solution inadvertently creates additional security risks. Specifically, the implementation of these optimizations can undermine the effectiveness of the masking technique, resulting in potential information leakage that compromises the overall security of the circuit.

Our analysis, supported by PROLEAD, emphasizes that the independent selection of fresh mask bits for the first layer of DOM-AND gates in the Kronecker delta function is crucial. Ensuring that each fresh mask bit is chosen independently enhances the security of the design by avoiding any glitch-extended probe placed on the circuit from observing any data-dependent information, i.e., first-order security. Accordingly, fresh mask bits $r_1, \ldots, r_4$ should be defined as follows.

$$r_1 \xleftarrow{\$} \mathrm{GF}(2), \qquad r_2 \xleftarrow{\$} \mathrm{GF}(2),$$
$$r_3 \xleftarrow{\$} \mathrm{GF}(2), \qquad r_4 \xleftarrow{\$} \mathrm{GF}(2)$$

The second and third layers of the DOM-AND gates exhibit distinct characteristics that set them apart from the first layer. Our comprehensive evaluations indicate that, in terms of vulnerability, these layers are less critical compared to the first layer, which serves as the primary line of defense against potential leakage. This reduced vulnerability is largely attributed to the behavior of the masking mechanism; specifically the masking of the second input of the DOM-AND gate disappears at its output (see Equation (5)). As a result, the influence of fresh masks in this context is less significant, allowing for more flexible configurations.

The primary constraint pertains to the fresh masks assigned to $\mathcal{G}_5$ and $\mathcal{G}_6$, which must not be identical. Otherwise, this can introduce vulnerabilities, undermining the security of the cryptographic implementation. To illustrate this, let's examine the scenario where if $r_5 = r_6$. According to the analysis presented in Section III, the outputs of $\mathcal{G}_5$ and $\mathcal{G}_6$ can be expressed by the following equations.

$$w_0^i = (x_0^i\, x_1 \oplus r_1)\, x_2\, x_3 \oplus r_5,$$
$$w_1^i = (x_4^i\, x_5 \oplus r_3)\, x_6\, x_7 \oplus r_5, \quad i \in \{0, 1\}$$

When the exemplary probe placed on v_1 observes 1, it can be concluded that $w_0^0 = w_1^0 = 1$ leading to

$$(x_0^0\, x_1 \oplus r_1)\, x_2\, x_3 \oplus r_5 = (x_4^0\, x_5 \oplus r_3)\, x_6\, x_7 \oplus r_5$$
$$\Longrightarrow (x_0^0\, x_1 \oplus r_1)\, x_2\, x_3 = (x_4^0\, x_5 \oplus r_3)\, x_6\, x_7$$

Hence, similar to the former analysis, this indicates that if any of x_2 or x_3 (on the left side of the equation) is zero, and any of x_6 or x_7 (on the right side) is zero, the distribution observed by v_1 is not uniform.

Therefore, as a potential solution, the fresh masks assigned to the DOM-AND gates $\mathcal{G}_5$, $\mathcal{G}_6$, and $\mathcal{G}_7$ can be specified as follows while maintaining the security under the glitch-extended probing model.

$$r_5 = r_4, \qquad r_6 = r_2, \qquad r_7 = r_3 \tag{9}$$

In order to rigorously evaluate this optimization, we did not follow the one-time pad transformation approach applied in [12] since we believe that it cannot appropriately examine the security of such optimizations as its result does not fit to security evaluation under the glitch-extended probing model. Therefore, we conducted comprehensive analysis by PROLEAD confirming that the Kronecker delta function with the fresh mask optimization given in Equation (9) effectively mitigates vulnerabilities and ensures first-order security.

As mentioned earlier, our analysis and optimizations were based on the adversarial model considered in [12], i.e., glitch-extended probing model. However, when we expand the evaluation to consider transitions, which is a natural way to evaluate masked hardware designs, the results change quite a bit, revealing new insights and challenges. In short, none of the optimizations discussed above can maintain security under glitch- and transition-extended probing models. We should highlight that under such a model, a probe placed on a signal does not only propagate backward through the combinational circuit contributing to the probed signal but also to two consecutive inputs of such a combinational circuit. By means of trial and error, we found four solutions given below, which indeed do not play a significant role in reducing the demand for fresh mask bits.

$$r_1 \xleftarrow{\$} \mathrm{GF}(2), \qquad r_2 \xleftarrow{\$} \mathrm{GF}(2), \quad r_3 \xleftarrow{\$} \mathrm{GF}(2),$$
$$r_4 \xleftarrow{\$} \mathrm{GF}(2), \qquad r_5 \xleftarrow{\$} \mathrm{GF}(2), \quad r_6 \xleftarrow{\$} \mathrm{GF}(2),$$
$$r_7 = r_i,\ \forall i \in \{1, 2, 3, 4\}$$

Naturally, under the glitch- and transition-extended probing models, PROLEAD did not detect any first-order vulnerability in the Kronecker delta function when any of these fresh randomness optimizations is applied.

We should further refer to a second-order implementation of the masked AES Sbox presented in [12] following the same concept. The authors have also provided an optimization technique to reduce the number of fresh masks from 21 to 13 bits. We have constructed this design as well and performed similar analysis. None of our analyses by PROLEAD (considering both glitches and transitions) up to second order and using at least 100 million simulations revealed any vulnerability. Our designs and evaluation results can be found in https://github.com/ChairImpSec/MultiplicativeMaskingAES.

V. Conclusions

This research presented a comprehensive security evaluation of the multiplicative masking scheme applied to an AES hardware design. Focusing on the AES Sbox, we identified a critical vulnerability in the first-order implementation stemming from the randomness optimization of the Kronecker delta function proposed in [12]. This vulnerability, which could compromise the security of the underlying cryptographic implementation, is addressed through an alternative optimization, ensuring that the design meets first-order security requirements although the gain is minimal.

Our findings emphasize the crucial role of evaluation and formal verification tools in assessing the security of masked implementations particularly when customized optimizations are in place. Although we have employed PROLEAD for our entire evaluations, we should admit that it is not the only tool capable to detecting this flaw. We predict that applying other tools like SILVER [25] would lead to similar results.

References

[1] P. C. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, ser. Lecture Notes in Computer Science, M. J. Wiener, Ed., vol. 1666. Springer, 1999, pp. 388–397.

[2] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi, "Towards sound approaches to counteract power-analysis attacks," in *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, ser. Lecture Notes in Computer Science, M. J. Wiener, Ed., vol. 1666. Springer, 1999, pp. 398–412.

[3] J. D. Golic and C. Tymen, "Multiplicative masking and power analysis of AES," in *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, ser. Lecture Notes in Computer Science, B. S. K. Jr., Ç. K. Koç, and C. Paar, Eds., vol. 2523. Springer, 2002, pp. 198–212.

[4] I. Damgård and M. Keller, "Secure multiparty AES," in *Financial Cryptography and Data Security, 14th International Conference, FC 2010, Tenerife, Canary Islands, Spain, January 25-28, 2010, Revised Selected Papers*, ser. Lecture Notes in Computer Science, R. Sion, Ed., vol. 6052. Springer, 2010, pp. 367–374.

[5] L. Genelle, E. Prouff, and M. Quisquater, "Secure multiplicative masking of power functions," in *Applied Cryptography and Network Security, 8th International Conference, ACNS 2010, Beijing, China, June 22-25, 2010. Proceedings*, ser. Lecture Notes in Computer Science, J. Zhou and M. Yung, Eds., vol. 6123, 2010, pp. 200–217.

[6] ——, "Montgomery's trick and fast implementation of masked AES," in *Progress in Cryptology - AFRICACRYPT 2011 - 4th International Conference on Cryptology in Africa, Dakar, Senegal, July 5-7, 2011. Proceedings*, ser. Lecture Notes in Computer Science, A. Nitaj and D. Pointcheval, Eds., vol. 6737. Springer, 2011, pp. 153–169.

[7] S. Mangard, N. Pramstaller, and E. Oswald, "Successfully attacking masked AES hardware implementations," in *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, ser. Lecture Notes in Computer Science, J. R. Rao and B. Sunar, Eds., vol. 3659. Springer, 2005, pp. 157–171.

[8] S. Nikova, V. Rijmen, and M. Schläffer, "Secure hardware implementation of nonlinear functions in the presence of glitches," *J. Cryptol.*, vol. 24, no. 2, pp. 292–321, 2011.

[9] B. Bilgin, B. Gierlichs, S. Nikova, V. Nikov, and V. Rijmen, "A more efficient AES threshold implementation," in *Progress in Cryptology - AFRICACRYPT 2014 - 7th International Conference on Cryptology in Africa, Marrakesh, Morocco, May 28-30, 2014. Proceedings*, ser. Lecture Notes in Computer Science, D. Pointcheval and D. Vergnaud, Eds., vol. 8469. Springer, 2014, pp. 267–284.

[10] A. Moradi, A. Poschmann, S. Ling, C. Paar, and H. Wang, "Pushing the limits: A very compact and a threshold implementation of AES," in *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, ser. Lecture Notes in Computer Science, K. G. Paterson, Ed., vol. 6632. Springer, 2011, pp. 69–88.

[11] H. Groß, S. Mangard, and T. Korak, "Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order," in *Proceedings of the ACM Workshop on Theory of Implementation Security, TIS@CCS 2016 Vienna, Austria, October, 2016*, B. Bilgin, S. Nikova, and V. Rijmen, Eds. ACM, 2016, p. 3.

[12] L. D. Meyer, O. Reparaz, and B. Bilgin, "Multiplicative masking for AES in hardware," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2018, no. 3, pp. 431–468, 2018.

[13] T. D. Cnudde, O. Reparaz, B. Bilgin, S. Nikova, V. Nikov, and V. Rijmen, "Masking AES with d+1 shares in hardware," in *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, ser. Lecture Notes in Computer Science, B. Gierlichs and A. Y. Poschmann, Eds., vol. 9813. Springer, 2016, pp. 194–212.

[14] N. Müller and A. Moradi, "PROLEAD A probing-based hardware leakage detection tool," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2022, no. 4, pp. 311–348, 2022.

[15] M. Akkar and C. Giraud, "An implementation of DES and aes, secure against some attacks," in *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, ser. Lecture Notes in Computer Science, Ç. K. Koç, D. Naccache, and C. Paar, Eds., vol. 2162. Springer, 2001, pp. 309–318.

[16] G. Barthe, S. Belaïd, F. Dupressoir, P. Fouque, B. Grégoire, P. Strub, and R. Zucchini, "Strong non-interference and type-directed higher-order masking," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, Eds. ACM, 2016, pp. 116–129.

[17] J. Coron, "Formal verification of side-channel countermeasures via elementary circuit transformations," in *Applied Cryptography and Network Security - 16th International Conference, ACNS 2018, Leuven, Belgium, July 2-4, 2018, Proceedings*, ser. Lecture Notes in Computer Science, B. Preneel and F. Vercauteren, Eds., vol. 10892. Springer, 2018, pp. 65–82.

[18] J. Boyar, P. Matthews, and R. Peralta, "Logic minimization techniques with applications to cryptology," *J. Cryptol.*, vol. 26, no. 2, pp. 280–312, 2013.

[19] T. Schneider and A. Moradi, "Leakage assessment methodology - extended version," *J. Cryptogr. Eng.*, vol. 6, no. 2, pp. 85–99, 2016.

[20] G. Barthe, S. Belaïd, P. Fouque, and B. Grégoire, "maskverif: a formal tool for analyzing software and hardware masked implementations," *IACR Cryptol. ePrint Arch.*, p. 562, 2018. [Online]. Available: https://eprint.iacr.org/2018/562

[21] S. Belaïd, D. Mercadier, M. Rivain, and A. R. Taleb, "Ironmask: Versatile verification of masking security," in *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022*. IEEE, 2022, pp. 142–160.

[22] N. Bordes and P. Karpman, "Fast verification of masking schemes in characteristic two," in *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part II*, ser. Lecture Notes in Computer Science, A. Canteaut and F. Standaert, Eds., vol. 12697. Springer, 2021, pp. 283–312.

[23] J. S. Coron, "Checkmasks: Formal verification of side-channel countermeasures," 2017. [Online]. Available: https://github.com/coron/checkmasks

[24] V. Arribas, S. Nikova, and V. Rijmen, "Vermi: Verification tool for masked implementations," in *25th IEEE International Conference on Electronics, Circuits and Systems, ICECS 2018, Bordeaux, France, December 9-12, 2018*. IEEE, 2018, pp. 381–384.

[25] D. Knichel, P. Sasdrich, and A. Moradi, "SILVER - statistical independence and leakage verification," in *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part I*, ser. Lecture Notes in Computer Science, S. Moriai and H. Wang, Eds., vol. 12491. Springer, 2020, pp. 787–816.

[26] C. Wolf, "Yosys open synthesis suite." [Online]. Available: https://yosyshq.net/yosys/

[27] Y. Ishai, A. Sahai, and D. A. Wagner, "Private circuits: Securing hardware against probing attacks," in *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, ser. Lecture Notes in Computer Science, D. Boneh, Ed., vol. 2729. Springer, 2003, pp. 463–481.