# The Security Evaluation of an Efficient Lightweight AES Accelerator †

Aljuffri, A.A.M.; Huang, R.; Muntenaar, L.V.M.; Gaydadjiev, G.; Ma, Kezheng ; Hamdioui, S.; Taouil, M.

**DOI**
10.3390/cryptography8020024

**Publication date**
2024

**Document Version**
Final published version

**Published in**
Cryptography

*cryptography*

MDPI

*Article*

# The Security Evaluation of an Efficient Lightweight AES Accelerator †

**Abdullah Aljuffri [1,2,*,‡], Ruoyu Huang [1,‡], Laura Muntenaar [1], Georgi Gaydadjiev [1], Kezheng Ma [3], Said Hamdioui [1] and Mottaqiallah Taouil [1]**

[1] Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, 2628 CD Delft, The Netherlands; r.huang-4@student.tudelft.nl (R.H.); lauramuntenaar@gmail.com (L.M.); g.n.gaydadjiev@tudelft.nl (G.G.); s.hamdioui@tudelft.nl (S.H.); m.taouil@tudelft.nl (M.T.)
[2] King Abdulziz City of Science and Technology, Riyadh 12354, Saudi Arabia
[3] Silicon Integrated, 5656 AE Eindhoven, The Netherlands; kezheng.ma@si-in.com
* Correspondence: a.a.m.aljuffri@tudelft.nl or aaljuffri@kacst.edu.sa; Tel.: +31-61-893-8166
† This paper is an extended version of our paper published in the 22nd IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom) in Exeter, UK, 1–3 November 2023.
‡ These authors contributed equally to this work.

**Abstract:** The Advanced Encryption Standard (AES) is widely recognized as a robust cryptographic algorithm utilized to protect data integrity and confidentiality. When it comes to lightweight implementations of the algorithm, the literature mainly emphasizes area and power optimization, often overlooking considerations related to performance and security. This paper evaluates two of our previously proposed lightweight AES implementations using both profiled and non-profiled attacks. One is an unprotected implementation, and the other one is a protected version using Domain-Oriented Masking (DOM). The findings of this study indicate that the inclusion of DOM in the design enhances its resistance to attacks at the cost of doubling the area.

**Keywords:** Advanced Encryption Standard; lightweight accelerator; IoT; side-channel attacks; domain-oriented masking



**Citation:** Aljuffri, A.; Huang, R.; Muntenaar, L.; Gaydadjiev, G.; Ma, K.; Hamdioui, S.; Taouil, M. The Security Evaluation of an Efficient Lightweight AES Accelerator. *Cryptography* **2024**, *8*, 24. https://doi.org/10.3390/cryptography8020024

Academic Editor: Jim Plusquellic

Received: 26 February 2024
Revised: 1 April 2024
Accepted: 13 May 2024
Published: 4 June 2024



## 1. Introduction

According to a recent report published by Cybersecurity Ventures [1], it is projected that the annual global cost of cybercrime will reach USD 10.5 trillion by the year 2025. This substantial amount signifies a notable escalation from the projected expense of USD 6 trillion in the year 2021. According to the report, there was an increase in the average cost of a data breach from USD 3.86 million in 2020 to USD 4.24 million in 2021. The prevalence of cyberattacks on IoT devices is steadily rising. In the year 2022, the global count of IoT cyberattacks exceeded 112 million [2], indicating a substantial 87% increase compared to the preceding year. It is anticipated that this pattern will persist in the forthcoming years as an increasing number of Internet of Things (IoT) devices are implemented. To protect their data, the Advanced Encryption Standard (AES) is used [3]. It is widely recognized and extensively employed. The AES is presently regarded as the preferred encryption technique in various domains, such as cloud computing [4] and healthcare [5]. Standard implementations of the AES require a notable amount of memory space and consume more power than lightweight ones [6]; it is therefore unfeasible to use them in IoT devices that are limited by the area and battery capacity, such as implantable devices [7]. As a result, implementations of the AES should make efficient use of both space and power while at the same time minimizing the impact on throughput and satisfying security requirements.

Several lightweight AES implementations have been proposed with the objective of minimizing both the area and power consumption. The objective of these accelerators is to enhance the efficiency of the conventional 128-bit data path by reducing it to 8 bits,

as demonstrated in various research studies, such as Lu et al. [8], Dhanuskodi et al. [9], Wamser and Sauer [10], and Banik et al. [11]. The aforementioned reduction leads to a decrease in the number of SBOXes from 16 to 1, thereby resulting in a more condensed hardware design and reduced power consumption. Nevertheless, the transition to 8-bit data paths has posed a considerable obstacle, particularly affecting the rate at which data can be processed. The encryption process of these designs necessitates a minimum of 160 cycles, thereby impacting the overall performance. In order to overcome this constraint, recent endeavors have undertaken the exploration of 32-bit designs, with the objective of achieving an improved equilibrium between performance and energy efficiency. This exploration has been documented in studies such as [12,13]. We recently published a new AES hardware accelerator [14] that enhances area, power usage, and latency. In terms of both area efficiency and power efficiency, our method outperforms state-of-the-art techniques, delivering 5.16 times higher performance per unit area and 2.68 times higher performance per unit power.

This paper presents a security evaluation of our recently published designs [14]. In the evaluation, we analyze the security of our non-protected and DOM-based protected lightweight AES implementations on both an FPGA and ASIC. The security evaluation consists of applying profiled and non-profiled attacks and performing conformance-style testing, which is a method to measure general leakage in the power traces. The contributions of this paper can be summarized as follows:

- The implementation of non-protected and DOM-based protected lightweight AES implementations using a Xilinx FPGA board and ASIC tools;
- A security evaluation of the implemented design using non-profiled attacks (i.e., correlation power analysis (CPA)) and profiled attacks (i.e., template-based analysis (TBA));
- A security evaluation using conformance-style techniques (i.e., Test Vector Leakage Assessment (TVLA) and Signal-to-Noise ratio (SNR)).

The remainder of this paper is organized as follows. Section 2 introduces the background on the AES, DOM, and analysis methods. Section 3 briefly recaps existing lightweight AES designs. Section 5 explains our proposed methodology. Security analysis results are provided in Section 6. Section 7 discusses the results of our work and outlines potential future directions. Finally, Section 8 concludes this paper.

## 2. Background

This section provides a brief overview of the relevant background topics. First, it explains the Advanced Encryption Standard algorithm. Thereafter, it provides a description of side-channel attacks. Finally, it explains the Domain-Oriented Masking (DOM) countermeasure.

### *2.1. The Advanced Encryption Standard (AES)*

The Advanced Encryption Standard (AES) is a symmetric cryptographic algorithm used to secure data by means of encryption and decryption. The algorithm has a data block size of 128 bits, with a key length of 128, 192, or 256 bits. The 128-bit data block is partitioned into 16 bytes, which are assigned to a $4 \times 4$ matrix known as the State array. Depending upon the length of the key, the algorithm performs a number of round operations, where the State matrix is updated: 128-bit keys require 10 rounds, 192-bit keys 12 rounds, and 256-bit keys 14 rounds. The steps performed in each round for both encryption and decryption are shown in Figure 1. The encryption process consists of four main modules in each round, namely, *SubBytes* , *ShiftRows*, *MixColumns*, and *AddRoundKey*. However, it is important to note that these modules are not applied in the first round (round 0) or the final round.

The *AddRoundKey* module consists of bit-wise XOR operations between the round key and the State array. The *SubBytes* module is the only nonlinear module inside the Advanced Encryption Standard (AES) and serves a critical function in providing security against linear cryptanalysis [15]. During the execution of the *SubBytes* module, the substitution of each byte in the State array is carried out by replacing it with another byte defined by

the SBOX. The entries of the SBOX are calculated using its multiplicative inverse inside the Galois Field GF($2^8$) and an affine transformation, as, for example, described in the work of Zhang and Parhi [16]. The *ShiftRows* module is responsible for performing rotate-left operations on the second, third, and fourth rows of the State array. This operation rotates the second row by one byte to the left, the third row by two bytes to the left, and the fourth row by three bytes to the left. The first row remains unaffected throughout this transformation. The *InvShiftRows* module is calculated by executing the inverse operation and hence contains rotate-right operations. The *MixColumns* and *InvMixColumns* modules are responsible for performing modular polynomial multiplications in the Galois Field GF($2^8$) on every column of the State array. Equation (1) shows the transformation for the *MixColumns* module and Equation (2) for the *InvMixColumns* module. The index $j$ denotes the column index of the State matrix, and hence, $0 \leq j \leq 3$.

$$\begin{bmatrix} b_{0,j} \\ b_{1,j} \\ b_{2,j} \\ b_{3,j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a_{0,j} \\ a_{1,j} \\ a_{2,j} \\ a_{3,j} \end{bmatrix} \tag{1}$$

$$\begin{bmatrix} a_{0,j} \\ a_{1,j} \\ a_{2,j} \\ a_{3,j} \end{bmatrix} = \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} b_{0,j} \\ b_{1,j} \\ b_{2,j} \\ b_{3,j} \end{bmatrix} \tag{2}$$

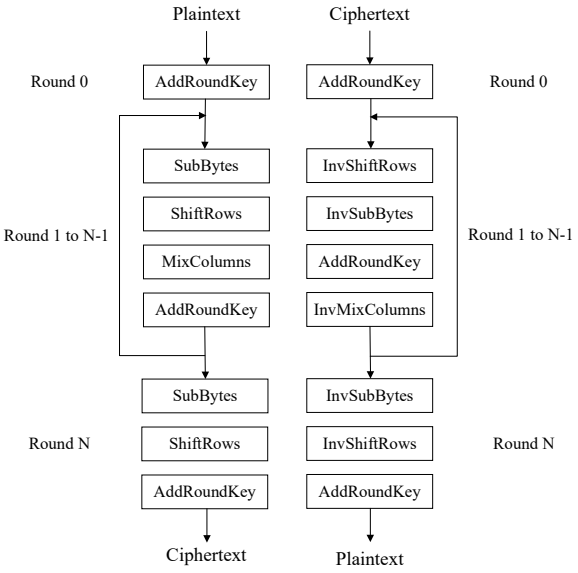



**Figure 1.** AES encryption and decryption flow diagram.

### *2.2. Side-Channel Attacks*

Side-channel attacks are a class of attacks that exploit system vulnerabilities through the analysis of unintentionally leaked physical information during the normal execution of a device [17]. The primary target of these attacks is the physical characteristics of a system, rather than the direct exploitation of vulnerabilities in the algorithm. The system's physical characteristics, such as power consumption, electromagnetic radiation, timing, and sound emissions, may provide valuable insights into its internal operations. Power consumption is a frequently exploited aspect due to its notably high success rate and simplicity. Power side-channel attacks target an intermediate operation, such as the SBOX operation, and perform a statistical analysis of the power consumption for different inputs. To correlate hypothetical assumptions of the key and the real power consumption, a leakage model is required [18]. Power attacks can be classified into two distinct categories: non-profiled attacks and profiled attacks. Next, an overview of each category is presented.

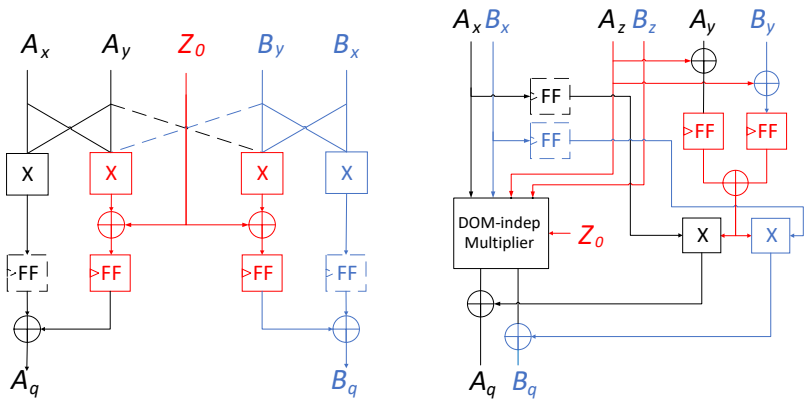
- Profiled power attacks: These attacks consist of two phases. In the first phase, the attacker creates a profile of the target system to understand how the system reacts to different activities by gathering power traces from the system under known circumstances (i.e., with known keys). With these data, the attacker builds a model of the behavior of the system. In the second phase, the attacker tries to extract sensitive data, such as cryptographic keys. The attacker realizes this by correlating side-channel information gathered during the actual attack on the target system with the model from the profile phase. Profiled attacks are generally more powerful, but they need access to a similar system to create the profile. Two well-known examples of such attacks are template-based attacks (TBAs) [19] and deep learning-based side-channel attacks (DL-SCAs) [20].
- Non-profiled power attacks: In contrast to profiled attacks, non-profiled attacks do not rely on prior knowledge or a profiling phase. Instead, the attacker directly observes the side-channel information from the target system and makes inferences based on these data. These attacks usually analyze how the system's behavior changes in response to different inputs. Non-profiled attacks are typically less accurate than profiled attacks but can be more practical in many real-world scenarios where the attacker is not able to create a profile in advance. Examples of such attacks are differential power attacks (DPAs) [21] and correlation power attacks (CPAs) [18].

### *2.3. Domain-Oriented Masking (DOM)*

In DOM [15] implementations, each variable is represented by $d + 1$ shares to protect the circuit from $d$th-order SCA attacks. The fundamental principle of the DOM approach is to maintain the independence of shares within each domain. The linear modules of the AES, such as *MixColumns*, *AddRoundKey*, and *ShiftRows*, do not share data between the domains, making it simple to maintain their independence. However, during the *SubBytes* operation, which is the only nonlinear module in the AES, data cross between the domains, making it necessary to use fresh random numbers to ensure their independence. Furthermore, registers are required in the *SubBytes* module to prevent glitches from propagating to other domains.

Hannes et al. [15] proposed two multipliers, named DOM-indep and DOM-dep. The DOM-indep multiplier requires that the inputs be shared independently, while the DOM-dep multiplier does not have such a restriction (i.e., the shares can be derived from a common source, and no additional cost of random numbers is required). The DOM-dep multiplier is based on the structure of the DOM-indep multiplier. Figure 2 shows the first-order DOM-indep and DOM-dep multipliers. $A_x$ and $B_x$ are shares of input $x$, while $A_y$ and $B_y$ are shares of input $y$, satisfying the relationships $x = A_x$ XOR $B_x$, and $y = A_y$ XOR $B_y$. $Z_0$ denotes a fresh random number used in the DOM-indep multiplier, while $A_z$ and $B_z$ are fresh random numbers used in the DOM-dep multiplier. The dotted-line registers shown in Figure 2 are optional and are only necessary for pipelining.

In the DOM-indep multiplier, Hannes et al. [15] categorize the product terms into two parts: inner-domain terms ($A_xB_x$ and $A_yB_y$) and cross-domain terms ($A_xB_y$ and $A_yB_x$). The inner-domain product terms do not reveal critical information since they are based on inputs from a single domain. In contrast, cross-domain calculations can only be performed on independent inputs to prevent the leakage of information for either $x$ or $y$. Hence, a fresh random value is used to ensure that the output of the DOM-indep multiplier remains statistically independent of other values. Additionally, flip-flops are employed to prevent glitches from propagating through this block. The first-order DOM-dep and DOM-indep multipliers can be extended to higher orders by using more shares and fresh random values without the need for redesigning the circuit.




(**a**) DOM-indep multiplier (**b**) DOM-dep multiplier

**Figure 2.** First-order DOM-indep and DOM-dep multipliers [15].

## 3. Related Work

Over the years, numerous advancements and innovations have been proposed to improve the performance and resource utilization of lightweight AES algorithms. These advancements have demonstrated notable improvements, each capitalizing on the merits and drawbacks of its predecessors. For example, in their work published in 2006, Hamalainen et al. [22] made a significant contribution by developing a lightweight *MixColumns* module that utilizes an 8-bit data path instead of the conventional 128-bit data path. The methodology employed by the researchers consisted of two SBOXes (instead of one to reduce latency) and implemented a parallel key expansion. Their implementation has a Gate Equivalent (GE) area of 3100 gates, with a latency of 160 cycles. In a subsequent study in 2011, Moradi et al. [23] made notable improvements using the Canright implementation technique [24], which computes Galois Field (GF) ($2^8$) operations using multiple operations in the GF ($2^4$), hence reducing the area by 23% to a GE of 2400. However, this technique increases the latency to 216 cycles. Further advancements have been made to improve the balance between the area and performance. For example, in 2014, Mathew et al. [25] introduced a novel method based on a single SBOX instance, resulting in a significant reduction in area to a GE of 1947. However, this enhancement led to a noticeable rise in the execution time, i.e., an increment to 336 cycles. Yu et al. [26], in 2019, synthesized the most effective characteristics from previous designs, resulting in a design with a GE of 1960 while still achieving a satisfactory execution time of 216 cycles. In 2020, Dhanuskodi et al. [9] proposed a methodology that effectively reduced state register activity to a single update per round. This approach resulted in an area with a GE of 886 and 160 cycles, thereby effectively optimizing resource utilization while maintaining a satisfactory performance. In a more recent publication in 2022, Davis and John [13] introduced an innovative approach to the ShiftRows technique, resulting in a significant reduction in the data path from 128 to 32 bits. In addition, the design optimized the MixColumns module and minimized the number of SBOX instances used for SubBytes. Furthermore, the design included optimizations that were applied to different stages of the AES algorithm, allowing for the sharing of hardware resources between the encryption and decryption modules. The implemented design achieved a significant reduction in area, i.e., a GE of 825, while simultaneously reducing the latency to a mere 61 cycles. Last year (i.e., 2023), we further introduced optimizations. For example, our proposed AES design [14] incorporates a key expansion bypass technique that verifies key alterations prior to each operation. If, during a sequence of encryption/decryption operations, the key remains unaltered, the procedure bypasses key expansion and proceeds directly to the encryption rounds. This technique greatly accelerates the decryption process by conserving eleven cycles. This technique comes with no area overhead, as all AES implementations typically require eleven 128-bit registers to store the keys during decryption. A second improvement was made to the MixColumns

operation. By selecting the columns of the State matrix as input to MixColumns, a number of MixColumns modules can be saved. The design additionally integrates resource sharing by allocating a single 128-bit register for state storage across all the modules. In addition, it presents optimizations across the encryption and decryption modules, thereby reducing the overall area. Three different designs with different data paths were proposed; they are designs with data path widths of 128, 64, and 32 with GEs of 846, 718, and 660 and latencies of 22, 42, and 82 cycles, respectively. In the same paper [14], we protected this lightweight design using DOM. Its purpose is to protect the implementation against side-channel analysis (SCA) attacks while maintaining a low area and performance overhead. The DOM is based on a five-stage SBOX, which is an enhanced variant of the eight-stage SBOX. The five-stage SBOX saves three cycles at the cost of slightly increasing the overall area. The DOM-independent and DOM-dependent multipliers were further optimized using simplified and shared multipliers, resulting in additional reductions in power consumption and area utilization. These simplifications guarantee that the design is efficient while theoretically ensuring security against SCA attacks. The aim of this paper is to evaluate the non-protected and protected implementations against side-channel attacks.

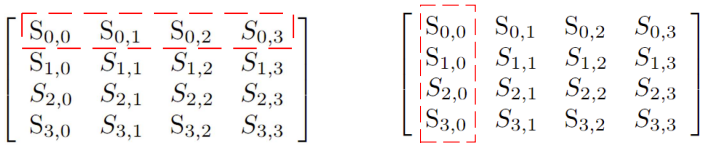
## 4. Lightweight DOM

This section presents the lightweight DOM implementation approach [14]. First, it describes the optimization techniques on the standard AES. Subsequently, it demonstrates how these optimization techniques have been integrated into DOM.

### *4.1. Lightweight AES Optimization Techniques*

The initial optimization method utilized for this lightweight version of the AES [14] is referred to as a key expansion bypass. This technique involves verifying the secret key before each execution. If there is a change in the key, the key expansion module is activated, and the keys are retained in the key registers. If no change is found, the procedure immediately proceeds to the round modules, namely, *AddRoundKey*, *SubBytes*, *MixColumns*, and *ShiftRows*. This strategy effectively cuts down the decryption time by eleven cycles. It is noteworthy that this optimization does not require extra registers for key storage and comparison, as both the standard and our enhanced AES designs use eleven 128-bit registers to store the key.

Furthermore, the efficiency of the MixColumns operation is enhanced by carefully choosing the input of the round function, hence enabling data sharing and minimizing computational complexities. In the context of creating a data-path design with a bit width smaller than 128 bits, it becomes imperative to choose the column numbers as the input for the round function. This decision is made with the objective of achieving a reduction of one cycle in each round and minimizing the number of MixColumns modules that are necessary. To demonstrate the impact of input selection on the round function, a 32-bit data-path architecture is used, as shown in Figure 3 as an illustrative example. The figure shows the selection of 32 bits in a sequential manner, both horizontally and vertically. It is seen that the *MixColumns* module executes a modular polynomial multiplication operation on every column of the State array. When the round function is applied with a row as the input, only a single row value is available after the SubBytes and ShiftRows operations. Consequently, it becomes infeasible to execute the MixColumns operation, as it necessitates a whole column. Hence, it is essential to allow for a waiting period of four cycles in order to effectively process all rows. This results in an extra cycle being introduced and a corresponding increase in the number of needed *MixColumns* modules from one to four. Alternatively, by opting to use the columns of the State array as the input for the round function, it becomes possible to execute the *MixColumns* operation more efficiently. This may be achieved by using a single *MixColumn* module, resulting in a reduction of one cycle every round.



(**a**) Row selection (**b**) Column selection

**Figure 3.** Input selection example in 32-bit design.

Lastly, in order to enhance the efficiency of the design space, the use of resource sharing is implemented. In the first stage, a constraint is imposed on the number of registers available for storing the state, in which just one 128-bit register is allocated and shared across all algorithm modules. Subsequently, both the encryption and decryption components are integrated into a single module in order to reduce the total area requirements. The employed shared-resource technique is shown in Figure 4, wherein the variable *cnt* denotes the round index, and *key[cnt]* represents the 128-bit key that is to be XORed with the State array during the current round. The boxes labeled as "shared" indicate the presence of a common functionality between the encryption and decryption procedures. A comprehensive elucidation of the common modules will be presented next.

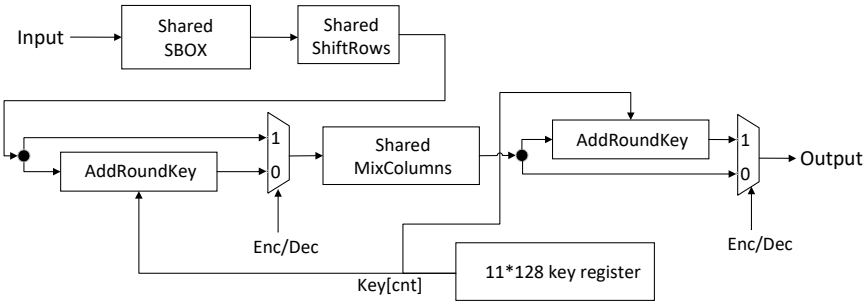



**Figure 4.** Proposed round function AES encryption and decryption.

- Shared SBOX: The proposed optimized shared SBOX is derived from the SBOX design proposed by [16,27–29]. The aforementioned articles used an SBOX shared by both the encryption and decryption modules in order to minimize the required space. According to the available information, the SBOX design discussed in the publication by Teng et al. [28,29] exhibits the smallest area among the known designs. In contrast to prior designs, the aforementioned approaches exhibited resource sharing across three distinct modules, namely, preprocess, postprocess, and scalar square. The preprocess module performs an isomorphic mapping and inverse affine transformation for the decryption process, whereas, for the encryption process, it only performs an isomorphic mapping. The postprocess module performs affine transformation and inverse isomorphic mapping for encryption and only inverse isomorphic mapping for decryption. The scalar-square module is used for the operations of squaring and multiplication with a constant value of $\lambda = \{1, 1, 0, 0\}$. These operations result in three XOR reductions. The suggested shared SBOX is shown in Figure 5. The optimization process includes simplifying the first multiplier, consolidating the computation of the latter two multipliers, and enhancing the inverter to improve its area efficiency.
- Shared ShiftRows: The ShiftRows and InvShiftRows functions execute distinct shift operations on every row of the State array. The State array, as shown in Figure 6, illustrates the modifications resulting from the application of the ShiftRows and InvShiftRows operations. The provided figure demonstrates that the outcomes of the ShiftRows and InvShiftRows operations for the first and third rows are identical, indicating that these rows may use the same shift procedures. However, it is necessary to use multiplexers for the remaining two rows when the shifts exhibit opposite directions.

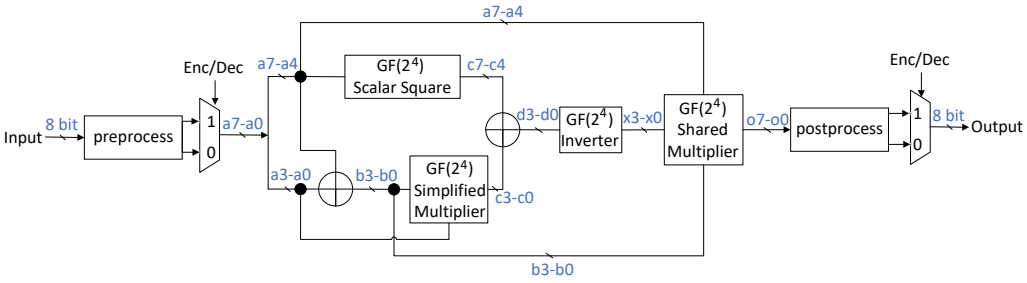



**Figure 5.** Proposed shared SBOX.

$$\begin{bmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,1} & S_{1,2} & S_{1,3} & S_{1,0} \\ S_{2,2} & S_{2,3} & S_{2,0} & S_{2,1} \\ S_{3,3} & S_{3,0} & S_{3,1} & S_{3,2} \end{bmatrix} \qquad \begin{bmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,3} & S_{1,0} & S_{1,1} & S_{1,2} \\ S_{2,2} & S_{2,3} & S_{2,0} & S_{2,1} \\ S_{3,1} & S_{3,2} & S_{3,3} & S_{3,0} \end{bmatrix}$$

(**a**) ShiftRows (**b**) InvShiftRows

**Figure 6.** Shift transformation of ShiftRows and InvShiftRows.

- Shared MixColumns: The optimization of the Shared *MixColumns* was performed according to the design provided in the publication by Zhang and Prouff [16]. This design effectively utilizes shared resources across the *MixColumns* and *InvMixColumns* operations. The design streamlines the "$X_2$" and "$X_4$" blocks, leading to a notable decrease in area. In order to accommodate our entire design, multiplexers were included. The architecture of the Shared MixColumns module is shown in Figure 7. In this figure, the outputs of MixColumns, InvMixColumns, and Combined MixColumns are denoted by the signals Out0, Out1, Out2, and Out3, represented in red, blue, and black, respectively. In our design, the need for separate *MixColumns* and *InvMixColumns* modules is eliminated, as we include a single block for both encryption and decryption. This is seen in Figure 4.

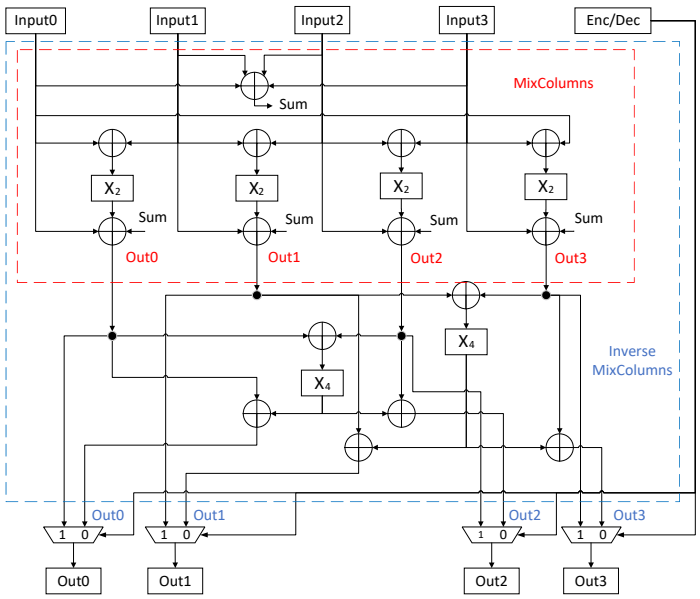



**Figure 7.** $GF(2^4)$ shared MixColumns.

### *4.2. Lightweight DOM Design*

DOM was proposed in [15] to protect AES implementations from SCA attacks. The authors introduced two types of SBOXes: a five-stage SBOX and an eight-stage SBOX. The

five-stage SBOX is an optimized version of the eight-stage SBOX and saves three cycles per round, with only a minor increase in the overall area. Hence, overall, it is 33 cycles (3 cycles × 11 rounds) faster, which is considered to be a significant improvement in performance. Therefore, we chose to start from the five-stage SBOX, further optimize it, and integrate it into our design. Note that first-order DOM can be easily scaled into higher-order DOM without redesigning components [15]. Hence, in our paper, we focus on the optimization of first-order DOM. As our proposed low-area design considers both encryption and decryption as a single unit for optimization reasons (see Figure 4), the lightweight DOM AES was implemented in the same manner. This is in contrast to the original design, where both encryption and decryption modules were treated separately [15]. Figure 8 shows the main part of our lightweight DOM design, where $A_{in}$ and $B_{in}$ are the input shares, and $A_{out}$ and $B_{out}$ the output shares.

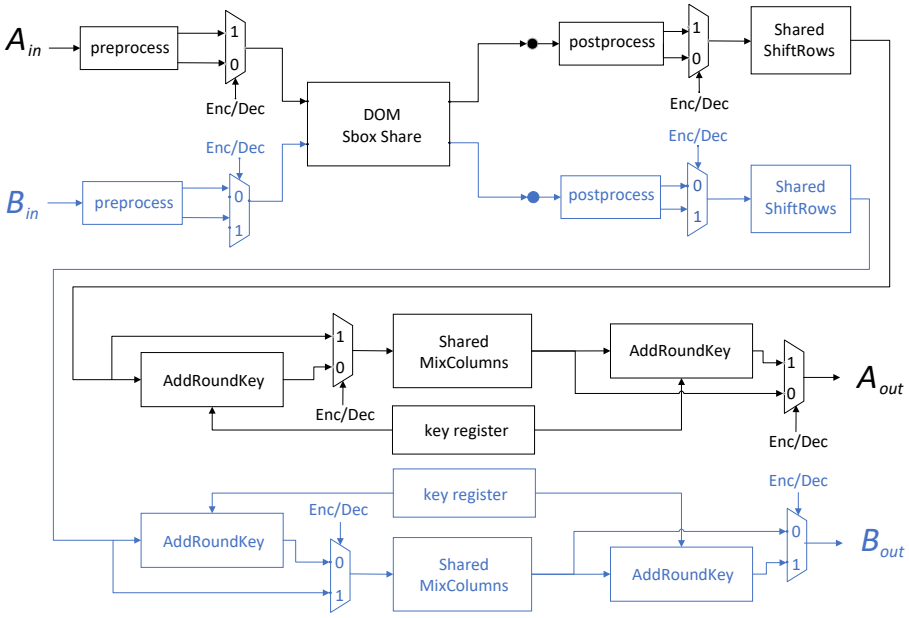



**Figure 8.** Proposed design for DOM encryption and decryption.

The lightweight DOM-SBOX design is based on the proposed lightweight AES SBOX in the previous subsection. In contrast to the original DOM-SBOX, our approach shares resources between encryption and decryption using the preprocess and postprocess functions. In addition, we optimized the DOM-indep and DOM-dep multipliers based on our simplified and shared multipliers to further reduce the area.

Figure 9 depicts our design of the optimized first-order five-stage lightweight DOM-SBOX shares, where $A_{sin}$ and $B_{sin}$ denote the input shares and $A_{sout}$ and $B_{sout}$ the output shares. Variables in red denote random values. The flip-flops in dotted boxes are optional registers that are only necessary in pipelining scenarios. For example, when the data path is less than 128 bits, the SBOX needs to be reused multiple times within one round, causing the input to change before the round is completed. In this case, the dotted flip-flops are necessary to ensure the design's functional correctness.

Figure 10 shows the design of the simplified DOM-indep and DOM-dep multipliers, which are similar to the other simplified multipliers shown in Figure 9. Compared to the original DOM multipliers, we replaced the normal multipliers with our simplified multipliers and shared multipliers, resulting in a reduction in power and area. The inputs of the first simplified DOM-indep are $A_{ai}$, $B_{ai}$, $A_{bi}$, and $B_{bi}$, while its outputs are $A_{qi}$ and $B_{qi}$ (see also Figure 9). $Z_0$ denotes a random number. Equation (3) shows the expression for the simplified DOM-indep multiplier, where "+" represents the XOR operation, $ai = A_{ai}+B_{ai}$, $bi = A_{bi}+B_{bi}$, and $qi = A_{qi}+B_{qi}$. In addition, $A_{ai}A_{bi}$, $A_{ai}B_{bi}$, $B_{ai}A_{bi}$, and $B_{ai}B_{bi}$ are calculated

using our proposed simplified multiplier. Flip-flops are employed to prevent glitches (and hence also power leakage) from propagating through this block.

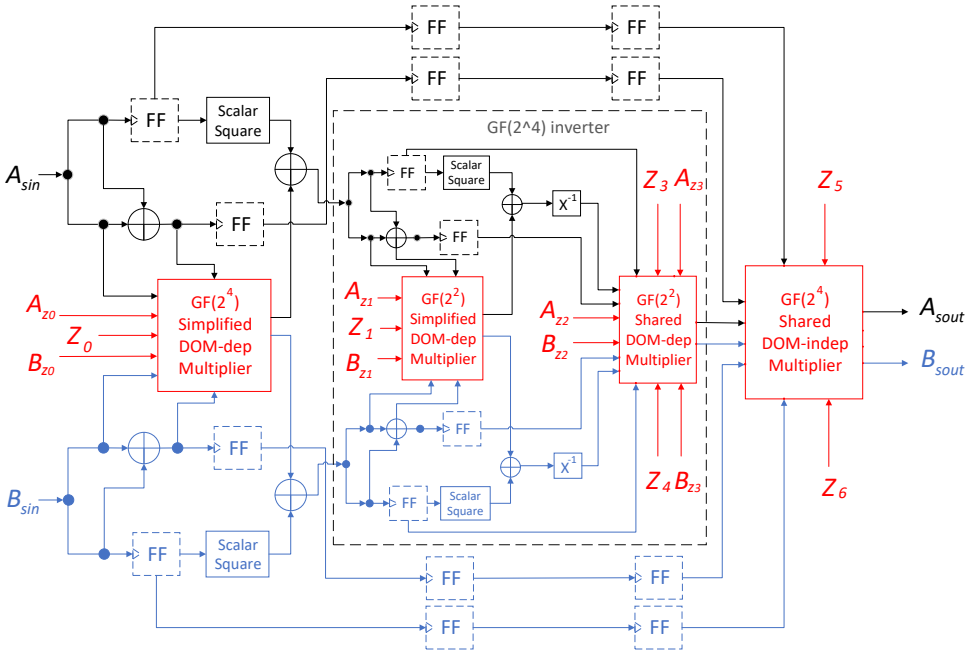
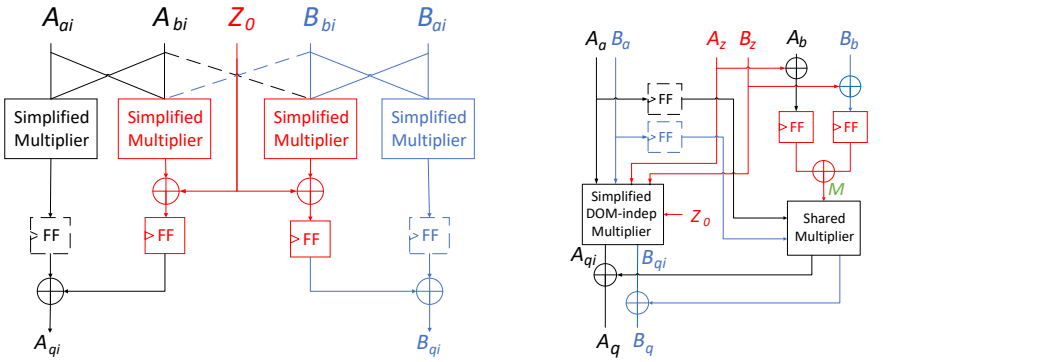



**Figure 9.** Structure of the first-order 5-pipeline-stage DOM SBOX shared module.




(**a**) Simplified DOM-indep (**b**) Simplified DOM-dep

**Figure 10.** Proposed first-order simplified DOM-indep and DOM-dep multipliers

$$\begin{aligned} ai * bi &= (A_{ai} + B_{ai})(A_{bi} + B_{bi}) \\ &= A_{ai}A_{bi} + A_{ai}B_{bi} + B_{ai}A_{bi} + B_{ai}B_{bi} \\ &= (A_{ai}A_{bi} + A_{ai}B_{bi} + Z_0) \\ &+ (B_{ai}A_{bi} + Z_0 + B_{ai}B_{bi}) \\ &= A_{qi} + B_{qi} = qi \end{aligned} \tag{3}$$

The simplified DOM-dep multiplier is designed based on the simplified DOM-indep multiplier. The inputs of the first simplified DOM-indep are $A_a$, $B_a$, $A_b$, and $B_b$, while its outputs are $A_q$ and $B_q$. $A_z$, $B_z$, and $Z_0$ are random numbers used to ensure the independence of shares. Equation (4) shows the expression of the DOM-dep multiplier, where $M = (A_b + B_b) + (A_z + B_z)$, and "+" represents the XOR operation, $a = A_a + B_a$,

$b = A_b + B_b$, $z = A_z + B_z$, and $q = A_q + B_q$. The shared multiplier is utilized for the calculation of $(A_a \cdot M + B_a \cdot M)$, leading to an area reduction.

$$\begin{aligned} a * b &= a * (b + z) + a * z \\ &= (A_a + B_a)(A_b + B_b + A_z + B_z) \\ &+ (A_a + B_a)(A_z + B_z) \\ &= (A_a + B_a)(A_b + B_b + A_z + B_z) \\ &+ DOM\_indep\{A_a, B_a, A_z, B_z\} \\ &= (A_a \cdot M + B_a \cdot M) + A_{qi} + B_{qi} \\ &= (A_a \cdot M + A_{qi}) + (B_a \cdot M + B_{qi}) \\ &= A_q + B_q = q \end{aligned} \tag{4}$$

Table 1 shows a comparison of the area between the proposed lightweight first-order DOM SBOX [14] and the original first-order SBOX proposed in [15]. Compared to their design, our eight-stage and five-stage first-order DOM SBOX designs achieve area reductions of 9.9% and 6.9%, respectively.

**Table 1.** Area comparison of DOM SBOX.

| Design | SBOX Type | Area ($\mu m^2$) | Area Ratio |
|---|---|---|---|
| [15] DOM SBOX | eight-stage | 19682 | 1 |
| | five-stage | 21196 | 1.077 |
| [14] Lightweight DOM SBOX | eight-stage | 17735 | 0.901 |
| | five-stage | 19741 | 1.003 |

## 5. Security Analysis Methodology

In this section, we describe the different techniques used to evaluate the security of our proposed designs. They are evaluation-style and conformance-style testing and are explained next.

### *5.1. Evaluation-Style Analysis Methodology*

In evaluation-style testing, the aim is to evaluate the security by applying existing and/or new attacks. We start with the correlation power analysis (CPA). CPA is a technique that relies on a theoretical model to exploit the relationship between the power consumption of a cryptographic device and the sensitive data being processed (i.e., the secret key). In the context of the AES, the attacker measures power traces of the AES algorithm for randomly selected plaintexts. Subsequently, the attacker statistically correlates the traces with the selected intermediate target of the algorithm (e.g., SBOX) using a leakage model (e.g., Hamming weight (HW), which is the count of the ones in the target output). CPA can be realized using Equation (5), where $N$, $T_{n,j}$, $H_{n,i}$, $\overline{T_j}$, and $\overline{H_i}$ represent the total number of traces, power consumption at sample $j$ of trace number $n$, Hamming value of trace $n$ based on guessed key $i$, the mean of traces at sample $j$, and the mean of Hamming weight values for guessed key $i$, respectively.

$$r_{i,j} = \frac{\sum_{n=1}^{N}(H_{n,i} - \overline{H_i}) \cdot (T_{d,j} - \overline{T_j})}{\sqrt{\sum_{n=1}^{N}(H_{n,i} - \overline{H_i})^2 \cdot \sum_{n=1}^{N}(T_{n,j} - \overline{T_j})^2}} \tag{5}$$

Over the past twenty years, there have been several notable advancements in improving CPA attacks. One significant development is the creation of a technique that allows for the gradual updating of the correlation coefficient matrix as more power traces are added [30]. This eliminates the necessity of reprocessing all the data whenever new traces are included. This enhancement has significantly improved the performance of CPA attacks.

The authors in [31] showed that the selection of particular plaintexts increases the signal-to-noise ratio (SNR) and hence has improved the accuracy of these attacks. These strategically selected plaintexts (i.e., a subset of the collected data) utilize a specific Hamming weight value of the intermediate target (i.e., 0, 1, 7, and 8) to enhance the precision and efficiency of CPA. The aforementioned method aims to extract a small subset of power traces with a high SNR to improve the attack. However, it blindly assumes that the traces with the Hamming weight 0, 1, 7, and 8 have the highest SNRs without verifying this. Dynamically verifying the Hamming weights with the largest SNR can have a significant impact on the success rate of attacks. The authors in [32] proposed a new SNR-centric power trace extractor called Shortest Distance First (SDF). SDF takes advantage of known plaintexts to estimate the SNR of each power trace. The extractor then selects power traces with the highest estimated SNRs.

$$\hat{\mathrm{SNR}}(\tau_j) = \frac{\sum_{i=1}^{N} \left(\bar{\mathcal{H}}(\tau) - \mu(\tau)\right)^2}{\sum_{i=1}^{N} \left(T_{i,j}(\tau) - \bar{\mathcal{H}}(\tau)\right)^2} \tag{6}$$

In our CPA attack, we employ recent techniques, with the aim of selecting the most powerful attack scenario. Moreover, in order to maximize the effectiveness, we implement an extreme attack scenario wherein only the targeted bytes of the plaintext exhibit varying Hamming weight values in a trace, while the remaining non-targeted bytes are maintained at a constant value. This scenario was used to improve the SNR, thereby enhancing the accuracy of the CPA attack. The proposed scenario enabled us to precisely separate the power fluctuations caused by the changing byte, thereby enhancing the clarity and ease of analysis when comparing them with other state-of-the-art models. The algorithm outlined in Algorithm 1 shows our advanced CPA attack. It involves measuring multiple traces for each subkey, where the subbyte under the attack of the plaintext $pt_i$ XORed with the subkey is random, while the rest of the input bytes remain constant. These traces are grouped into subsets $Trc_i$ to evaluate the subkeys independently. For each sample $j$, we evaluate the traces in subset $Trc_i$ and calculate the signal-to-noise ratio (SNR) according to [32] using Equation (6), where $\bar{\mathcal{H}}(\tau)$, $\mu(\tau)$, and $T_{i,j}(\tau)$ represent the Hamming weight, the mean, and the sample power value of trace $i$ at sample $j$. To expedite the process, we avoid their sorting method and instead apply a threshold to decide whether a trace should be included in the correlation analysis. Both the Hamming weight extraction [31] and fast computation approach [30] are disregarded due to their lack of impact on the attack's accuracy. By eliminating the non-targeted bytes in the attack, we mitigate the overall noise interference in the power measurements, resulting in more distinct and definitive outcomes. This strategy, in accordance with the most recent developments in CPA techniques, offers a strong framework for evaluating the security of our AES implementations. It effectively identifies the weaknesses of the implementations.

In a similar fashion, we modify these techniques to improve template-based attacks (TBAs), which are profiled side-channel attacks based on a multivariate distribution model. TBAs involve creating specific templates for each potential value of the leakage model of the target function, such as the Hamming weight of the SBOX function, by using the mean and covariance of the traces associated with that particular value. As a TBA is computationally intensive, the profiles are based on a subset of all sampled data points called Points of Interest (POIs). In our profiling phase, we employ the sum of squared pairwise T-differences (SOST) method to identify POIs [33]. During the attack phase, we evaluate the probability of each trace belonging to a certain Hamming weight using multivariate distribution models of the profile phase. By applying the same principles as applied to our CPA attack, we significantly diminish the noise in our TBA templates by focusing on a single targeted byte at a time while keeping the other bytes constant. This strategy improves the accuracy of the profile. The attack phase is further refined by leveraging the signal-to-noise ratio (SNR) to include the traces with the largest leakage in the attack.

$$p(t;(m,c)d_i,k_j) = \frac{\exp(-\frac{1}{2}\cdot(t-m)'\cdot C^{-1}\cdot(t-m))}{\sqrt{(2\cdot\pi)^T \det(c)}} \tag{7}$$

**Algorithm 1** Advanced CPA

**procedure** KEY_EXTRACT($Prediction_{set}$, $pt_{array}$)
  $P_k[0,255]$ = key probability
  $Prediction$ = the results of the trained model on the attack traces
  $pt$ = the plaintext used in the encryption process.
  **for** each subkey $i$ **do**
    Record $Trc_i$ power traces; where $pt_i$ is random and $pt_{(16-i)}$ is fixed
  **end for**
  **for** each subkey $i$ **do**
    $P_k[0,255] = 0$
    **for** $j$ in $Trc_i$ **do**
      $X_{0,255} = predict(Trc_{i,j})$
      **for** $k$ = 0 to 255 **do**
        Compute Equation (6)
        **if** $SNR \geq threshold$ **then**
          Compute Equation (5)
        **end if**
      **end for**
    **end for**
    $guess_{subkey} = max(P_k)$
  **end for**
**end procedure**

### *5.2. Conformance-Style Security Analysis*

In conformance-style testing, a design is evaluated to determine whether it generally leaks information without applying a real attack. The Test Vector Leakage Assessment (TVLA) [34] and the signal-to-noise ratio (SNR) [35] are two popular instances that perform such leakage analysis.

The *t*-test [34] developed by Welch is the foundation for the TVLA. This test determines whether two populations have distributions that are comparable to one another or not. Welch's *t*-test is used to determine whether a design leaks information about the secret key. The leakage is assessed by using two different sets of power traces: one with a fixed plaintext/ciphertext (referred to as fixed set), and the other with random plaintext/ciphertext (referred to as variable set). The value of the key is the same in both sets. Equation (8) shows Welch's equation, which is used in the execution of this test. In the equation, $X_1$, $S_1$, and $N_1$ represent the mean, variance, and total number of power traces in the fixed set, respectively, while $barX_2$, $S_2$, and $N_2$ represent the mean, the variance, and the total number of power traces in the variable set, respectively.

$$t = \frac{\bar{X_1} - \bar{X_2}}{\sqrt{\frac{S_1^2}{N_1} + \frac{S_2^2}{N_2}}} \tag{8}$$

The signal-to-noise ratio (SNR) plays a crucial role in power leakage analysis within integrated circuits, especially in the context of security concerns related to side-channel attacks. In power leakage analysis, the SNR represents the ratio between the strength of the signal carrying useful information (such as the key) and the background noise present in the power consumption measurements of the device. Higher SNR values indicate a stronger and clearer signal, making it easier to discern patterns or correlations between power consumption and the data being processed. However, a lower SNR implies that

the signal carrying the information is comparatively weaker and hence might be more challenging to be exploited by an attacker.

## 6. Experimental Results

This section presents the experimental setup, performed experiments, and the obtained results.

### *6.1. Setup*

Based on the comparative analysis of all proposed designs, it was observed that the 64-bit and 128-bit designs exhibit good results in terms of area and performance, respectively. Hence, both of these designs and their DOM-based implementations were selected for security examination. The chosen designs were implemented in both Application-Specific Integrated Circuit (ASIC) and Field-Programmable Gate Array (FPGA) scenarios. The gate-level implementation in ASIC utilized TSMC 40nm technology, while the Xilinx Artix 7 was used for FPGA emulation. The simulation tool Questasim [36], the synthesis tool Cadence Genus [37], and the power trace generation tool Synopsys Spyglass [38] were all used to validate the power traces in the ASIC scenario. For FPGA emulation, the Chipwhisperer CW305 equipment was used, as documented in a previous study [39]. The equipment comprised a Xilinx FPGA and an analog-to-digital conversion (ADC) module capable of sampling at a maximum rate of 105 MS/s. Ultimately, a comprehensive security analysis was carried out on both the unprotected and DOM-based protected AES design in order to assess the overall security of both systems. The attacks were executed using the Python programming language and attack libraries, including SCALib [40]. Table 2 details the area and maximum frequency for ASIC designs, whereas Table 3 provides the area and maximum frequency for FPGA designs.

**Table 2.** Comparative analysis of ASIC designs [14].

| Design | Data Path | Freq. (MHz) | Area ($\mu m^2$) | Cycle |
|---|---|---|---|---|
| Lightweight AES | 64-bit | 117.6 | 195,355 | 4211 |
| | 128-bit | 112.4 | 236,553 | 2211 |
| Lightweight DOM | 64-bit | 190.8 | 522,844 | 12251 |
| | 128-bit | 188.7 | 698,780 | 10251 |

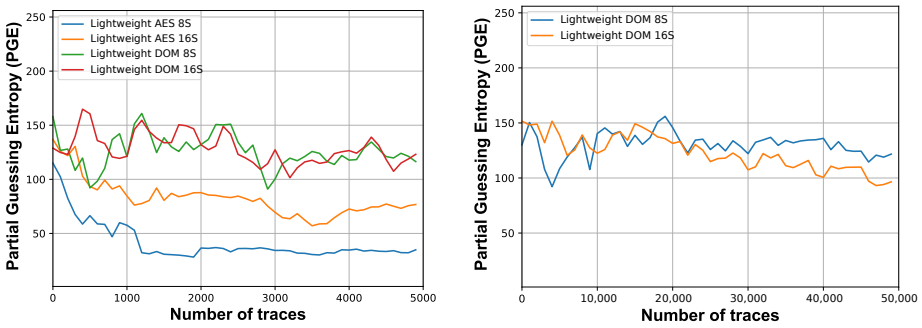
**Table 3.** Comparative analysis of FPGA designs [14].

| Design | Data Path | Freq. (MHz) | LUT | FF | DSP | BRAM |
|---|---|---|---|---|---|---|
| Lightweight AES | 64-bit | 95 | 2357 | 2713 | 0 | 0 |
| | 128-bit | 89.5 | 3223 | 2720 | 0 | 0 |
| Lightweight DOM | 64-bit | 89.25 | 5127 | 6372 | 0 | 0 |
| | 128-bit | 99 | 7515 | 7523 | 0 | 0 |

### *6.2. FPGA Security Analysis*

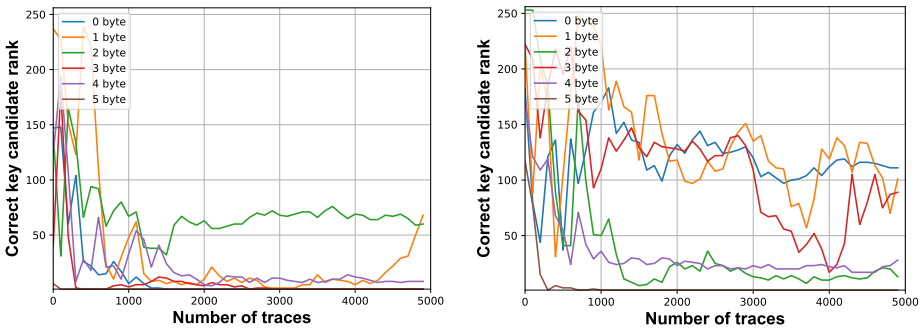
Figure 11a shows the results of the correlation power analysis (CPA) attack scenario for measured traces on the FPGA. For the unprotected implementations, a noticeable decline in the Partial Guessing Entropy (PGE) occurs after approximately 1000 power traces. This trend suggests that the accuracy of predicting the correct key values increases with the number of traces collected: the lower the PGE value, the more accurate the key prediction. Consequently, a detailed examination and ranking analysis of each subkey were conducted. The results are shown in Figure 12. The figure shows that the ranks of several subkeys were nearly zero, indicating correct guesses. This leads to the hypothesis that an adversary

equipped with more sophisticated tools and high-rate sampling devices could potentially determine the key accurately. On the other hand, for the implementations protected based on DOM, correctly identifying the key values proved to be challenging, as shown in Figure 11a. The PGE results consistently remained above 100, indicating a higher level of security. This observation is further substantiated when examining the individual subbyte rank analyses shown in Figure 13. These results demonstrate the effectiveness of the DOM-based design in protecting against CPA attacks. We conducted the same analysis using the template-based attack method described in Section 5.1. The rank analysis outcomes were comparable to the unprofiled CPA attacks as depicted in Figure 14. Therefore, we chose to proceed with the analysis using the unprofiled attack.



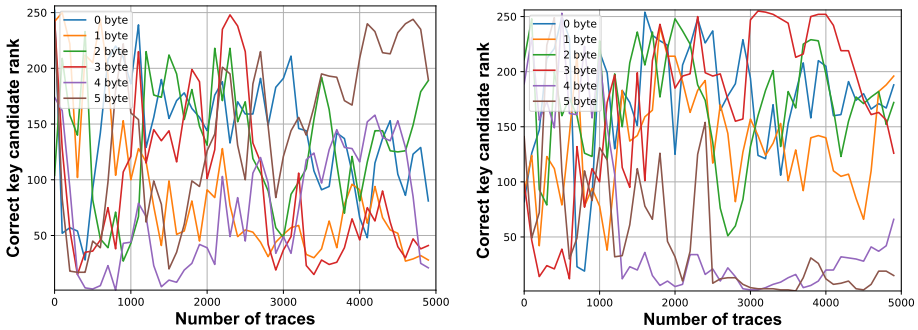

(**a**) FPGA analysis (**b**) ASIC analysis

**Figure 11.** Security analysis of unprotected and proposed designs.




(**a**) Lightweight AES 8SBOX (**b**) Lightweight 16SBOX

**Figure 12.** Rank analysis of the lightweight implementations for the FPGA.




(**a**) DOM 8SBOX (**b**) DOM 16SBOX

**Figure 13.** Rank analysis of the DOM-based lightweight implementations for the FPGA.

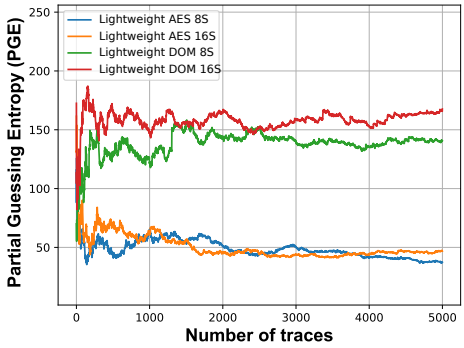


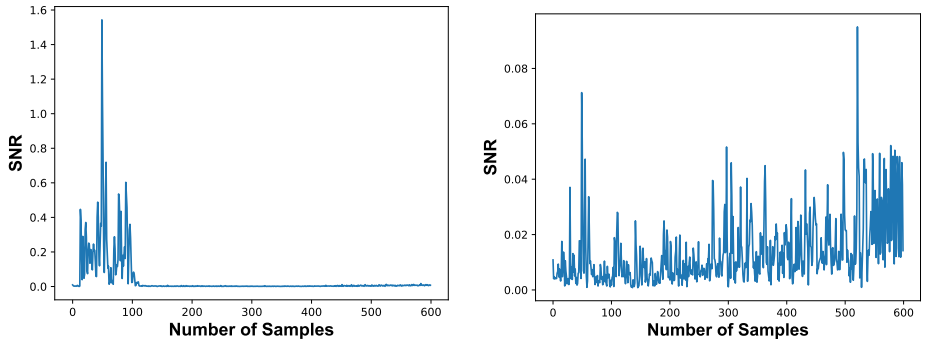

**Figure 14.** Template-based security analysis of unprotected and proposed designs.

### *6.3. ASIC Security Analysis*

The purpose of conducting the ASIC analysis was to evaluate the security in ideal conditions where there is no noise present in the power traces, as the experiment exclusively relies on simulations. Traces were obtained from the gate-level designs of both DOM-protected implementations, namely, the 16SBOX (i.e., 128-bit data path) and 8SBOX (i.e., 64-bit data path) models. We performed these analysis only for the protected design, as the unprotected implementation can be most likely broken due its vulnerabilities. As a result, we repeated the FPGA attack scenarios for the ASIC implementations. The results are depicted in Figure 11. The figure exhibits a strong resemblance to the outcomes observed for the attack performed on the FPGA, thereby validating the effectiveness of both designs in mitigating potential CPA attacks.

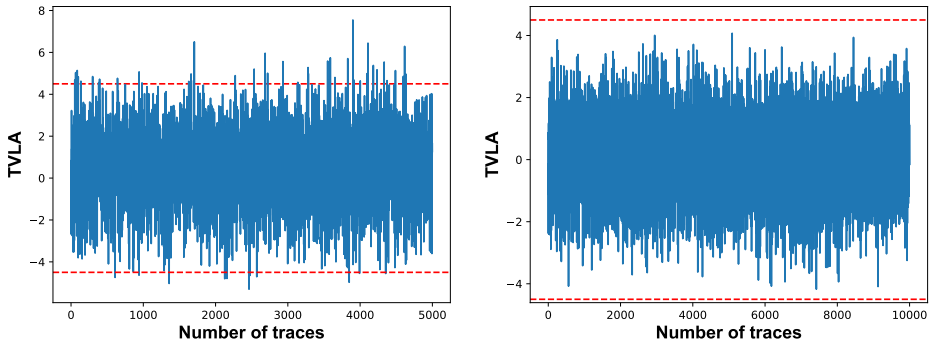
### *6.4. Conference Analysis Results*

To make sure that there is no leakage in general, we also applied the SNR and TVLA for the ASIC scenario. Note that both 16SBOX (i.e., 128-bit data path) and 8SBOX (i.e., 64-bit data path) showed similar results; for simplicity, the 8SBOX design was selected. For the unprotected designs (see Figures 15a and 16a), both the SNR and TVLA metrics demonstrate clear vulnerabilities by having high peak values. These SNR peaks of approximately 1.6 amplitude show a strong connection between power consumption and secret information (i.e., key values). This shows that there is a lot of leakage. In addition, the TVLA results, which reveal the difference between patterns in the data that are random and patterns that are not random, show clear deviations from randomness at these peak points, which exceeded 4.5 (above the red dotted line in Figure 16a). This confirms that these designs are probably vulnerable to side-channel attacks. In contrast, the protected designs exhibit much lower peak values in both SNR and TVLA analyses (see Figures 15b and 16b). The lower peaks signify that the correlation between side-channel leakage and secret data is substantially lower, indicating robustness against such attacks.




(**a**) Unprotected 8SBOX (**b**) Protected 8SBOX

**Figure 15.** SNR analysis.




(**a**) Unprotected 8SBOX (**b**) Protected 8SBOX

**Figure 16.** TVLA analysis.

## 7. Discussion

This work evaluated the security of our previously optimized design of the AES algorithm against side-channel analysis. Next, we discuss the limitations and future work based on the experimental results.

Practicality: One of the goals of the proposed lightweight AES scheme [14] is to provide a secure solution for Internet of Things (IoT) devices. However, the results showed that the non-protected design was susceptible to attacks that were carried out through side channels. Therefore, in cases where attackers can gain access to the power consumption or EM emission, versions protected by, for example, integrating DOM [15] must be used; unfortunately, the secured implementation doubled the area cost, which might be too expensive for lightweight applications such as IoT. Therefore, it is essential to explore other optimization possibilities and/or countermeasures.

High-order analysis: In our analysis, we utilized state-of-the-art first-order attacks, which included both a non-profiled side attack, such as CPA [31], and a profiled attack, such as TBA [19]. Although the attacks were sophisticated and exposed the vulnerability of the unsecured implementation, there is still a possibility that high-order attacks could be successful, such as collision attacks [41] or second-order attacks [42]. To ensure that the design is resilient against such attacks, they need to be evaluated as well.

Lightweight algorithms: In our comprehensive study, we undertook an in-depth comparison of our proposed design against other leading-edge implementations that similarly incorporate optimization techniques to enhance the AES algorithm. The National Institute of Standards and Technology (NIST) [43] has recently standardized a novel cryptographic algorithm named GIFT [44], specifically designed for lightweight applications. This development represents a significant advancement in the realm of cryptography, aiming to address the need for solutions on constrained devices. Note that similar optimization techniques [14] that we proposed could also be considered for such algorithms. Overall, it is important to understand where each algorithm stands in terms of computational efficiency, resource consumption, and resilience against side-channel and other attacks.

## 8. Conclusions

In our research, we conducted a thorough security assessment of an innovative AES accelerator, notable for its low power consumption and compact area while delivering high performance. Various methods were employed to improve the attack model, including the manipulation of a single byte, selection based on Hamming weight bias, and the exclusion of traces based on their signal-to-noise ratio (SNR) values. The designs under evaluation were evaluated using a Xilinx Artix7 FPGA chip and simulated on TSMC's 40nm technology. The findings indicate that our proposed version of DOM not only occupies a smaller area compared to conventional DOM but also upholds an equivalent degree of security (i.e., secure against first-order attacks).

**Author Contributions:** Conceptualization, A.A., R.H. and M.T.; methodology, A.A. and R.H.; software, A.A., R.H. and L.M.; validation, L.M.; writing—original draft preparation, A.A. and R.H.; writing—review and editing, M.T.; supervision, G.G., K.M., S.H. and M.T. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work is partially funded by the "Resilient Trust" project of the EU's Horizon Europe research and innovation programme under grant agreement No. 101112282

**Data Availability Statement:** The data presented in this study can be requested from the corresponding author as long as it is available.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Cybercrime to Cost The World $10.5 Trillion Annually by 2025. Available online: https://cybersecurityventures.com/hackerpocalypse-cybercrime-report-2016/ (accessed on 9 November 2023).
2. Global Annual Number of IoT Cyber Attacks 2018–2022. Available online: https://www.statista.com/statistics/1377569/worldwide-annual-internet-of-things-attacks/ (accessed on 25 November 2023).
3. Dworkin, M.; Barker, E.; Nechvatal, J.; Foti, J.; Bassham, L.; Roback, E.; Dray, J. *Advanced Encryption Standard (AES)*; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2001.
4. Sivakumar, P.; NandhaKumar, M.; Jayaraj, R.; Kumaran, A. Securing Data and Reducing the Time Traffic Using AES Encryption with Dual Cloud. In Proceedings of the 2019 IEEE International Conference on System, Computation, Automation and Networking (ICSCAN), Pondicherry, India, 29–30 March 2019; pp. 1–5. [CrossRef]
5. Khader, M.; Alian, M.; Hraiz, R.; Almajali, S. Simplified AES algorithm for healthcare applications on Internet of Thing. In Proceedings of the 2017 8th International Conference on Information Technology (ICIT), Amman, Jordan, 17–18 May 2017; pp. 543–547. [CrossRef]
6. Banik, S.; Pandey, S.K.; Peyrin, T.; Sasaki, Y.; Sim, S.M.; Todo, Y. GIFT: A Small Present—Towards Reaching the Limit of Lightweight Encryption. In Proceedings of the CHES, Taipei, Taiwan, 25–28 September 2017.
7. Kwarteng, E.; Cebe, M. A Survey on Security Issues in Modern Implantable Devices: Solutions and Future Issues. *Smart Health* **2022**, *25*, 100295. [CrossRef]
8. Lu, M.; Fan, A.; Xu, J.; Shan, W. A Compact, Lightweight and Low-Cost 8-Bit Datapath AES Circuit for IoT Applications in 28nm CMOS. In Proceedings of the 17th IEEE International Conference On Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering, TrustCom/BigDataSE 2018, New York, NY, USA, 1–3 August 2018; pp. 1464–1469. [CrossRef]
9. Dhanuskodi, S.N.; Allen, S.; Holcomb, D.E. Efficient Register Renaming Architectures for 8-bit AES Datapath at 0.55 pJ/bit in 16-nm FinFET. *IEEE Trans. Very Large Scale Integr. Syst.* **2020**, *28*, 1807–1820. [CrossRef]
10. Wamser, M.S.; Sigl, G. Pushing the limits further: Sub-atomic AES. In Proceedings of the 2017 IFIP/IEEE International Conference on Very Large Scale Integration, VLSI-SoC 2017, Abu Dhabi, United Arab Emirates, 23–25 October 2017; pp. 1–6. [CrossRef]
11. Banik, S.; Bogdanov, A.; Regazzoni, F. Atomic-AES: A Compact Implementation of the AES Encryption/Decryption Core. In Proceedings of the Progress in Cryptology—INDOCRYPT 2016—17th International Conference on Cryptology in India, Kolkata, India, 11–14 December 2016; pp. 173–190. [CrossRef]
12. Dao, M.H.; Hoang, V.P.; Dao, V.L.; Tran, X.T. An Energy Efficient AES Encryption Core for Hardware Security Implementation in IoT Systems. In Proceedings of the 2018 International Conference on Advanced Technologies for Communications (ATC), Ho Chi Minh City, Vietnam, 18–20 October 2018; pp. 301–304. [CrossRef]
13. Davis, C.; John, E. Shared Round Core Architecture: A Novel AES Implementation for Implantable Cardiac Devices. In Proceedings of the 65th IEEE International Midwest Symposium on Circuits and Systems, MWSCAS 2022, Fukuoka, Japan, 7–10 August 2022; pp. 1–4. [CrossRef]
14. Huang, R.; Aljuffri, A.; Hamdioui, S.; Ma, K.; Taouil, M. Securing an Efficient Lightweight AES Accelerator. In Proceedings of the 2023 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Exeter, UK, 1–3 November 2023.
15. Groß, H.; Mangard, S.; Korak, T. Domain-Oriented Masking: Compact Masked Hardware Implementations with Arbitrary Protection Order. *IACR Cryptol. ePrint Arch.* **2016**, 486. Available online: https://eprint.iacr.org/2016/486.pdf (accessed on 25 February 2024).
16. Zhang, X.; Parhi, K.K. High-speed VLSI architectures for the AES algorithm. *IEEE Trans. Very Large Scale Integr. Syst.* **2004**, *12*, 957–967. [CrossRef]
17. Zhou, Y.; Feng, D. Side-Channel Attacks: Ten Years After Its Publication and the Impacts on Cryptographic Module Security Testing. *IACR Cryptol. ePrint Arch.* **2005**, 388. Available online: https://eprint.iacr.org/2005/388.pdf (accessed on 25 February 2024).

18. Brier, E.; Clavier, C.; Olivier, F. Correlation Power Analysis with a Leakage Model. In Proceedings of the Cryptographic Hardware and Embedded Systems—CHES 2004, 6th International Workshop, Cambridge, MA, USA, 11–13 August 2004; Joye, M., Quisquater, J., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2004; Volume 3156, pp. 16–29. [CrossRef]
19. Chari, S.; Rao, J.R.; Rohatgi, P. Template Attacks. In Proceedings of the Cryptographic Hardware and Embedded Systems—CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, 13–15 August 2002; Lecture Notes in Computer Science; Revised Papers; Springer: Berlin/Heidelberg, Germany, 2002; Volume 2523, pp. 13–28. [CrossRef]
20. Aljuffri, A.; Reinbrecht, C.; Hamdioui, S.; Taouil, M. Impact of Data Pre-Processing Techniques on Deep Learning Based Power Attacks. In Proceedings of the 16th International Conference on Design & Technology of Integrated Systems in Nanoscale Era, DTIS 2021, Montpellier, France, 28–30 June 2021; pp. 1–6. [CrossRef]
21. Kocher, P.C.; Jaffe, J.; Jun, B. Differential Power Analysis. In Proceedings of the Advances in Cryptology—CRYPTO'99, 19th Annual International Cryptology Conference, Santa Barbara, CA, USA, 15–19 August 1999; Wiener, M.J., Ed.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1999; Volume 1666, pp. 388–397. [CrossRef]
22. Hämäläinen, P.; Alho, T.; Hännikäinen, M.; Hämäläinen, T.D. Design and Implementation of Low-Area and Low-Power AES Encryption Hardware Core. In Proceedings of the Ninth Euromicro Conference on Digital System Design: Architectures, Methods and Tools (DSD 2006), Dubrovnik, Croatia, 30 August–1 September 2006; pp. 577–583. [CrossRef]
23. Moradi, A.; Poschmann, A.; Ling, S.; Paar, C.; Wang, H. Pushing the Limits: A Very Compact and a Threshold Implementation of AES. In Proceedings of the Advances in Cryptology—EUROCRYPT 2011, 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, 15–19 May 2011; Lecture Notes in Computer Science; Paterson, K.G., Ed.; Springer: Berlin/Heidelberg, Germany, 2011; Volume 6632, pp. 69–88. [CrossRef]
24. Canright, D. A Very Compact S-Box for AES. In Proceedings of the Cryptographic Hardware and Embedded Systems—CHES 2005; Rao, J.R., Sunar, B., Eds.; Springer: Berlin/Heidelberg, Germany, 2005; pp. 441–455.
25. Mathew, S.; Sheikh, F.; Kounavis, M.E.; Gueron, S.; Agarwal, A.; Hsu, S.; Kaul, H.; Anders, M.A.; Krishnamurthy, R. 53 Gbps Native GF($2^4$)$^2$ Composite-Field AES-Encrypt/Decrypt Accelerator for Content-Protection in 45 nm High-Performance Microprocessors. *IEEE J. Solid State Circuits* **2011**, *46*, 767–776. [CrossRef]
26. Yu, J.; Aagaard, M. Benchmarking and Optimizing AES for Lightweight Cryptography on ASICs. 2019. Available online: https://csrc.nist.gov/CSRC/media/Events/lightweight-cryptography-workshop-2019/documents/papers/benchmarking-and-optimizing-aes-for-lwc-on-asics-lwc2019.pdf (accessed on 15 April 2023).
27. Satoh, A.; Morioka, S.; Takano, K.; Munetoh, S. A Compact Rijndael Hardware Architecture with S-Box Optimization. In Proceedings of the Advances in Cryptology—ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, 9–13 December 2001; Lecture Notes in Computer Science; Boyd, C., Ed.; Springer: Berlin/Heidelberg, Germany, 2001; Volume 2248, pp. 239–254. [CrossRef]
28. Ahmad, N.; Hasan, S.M.R. Low-power compact composite field AES S-Box/Inv S-Box design in 65 nm CMOS using Novel XOR Gate. *Integration* **2013**, *46*, 333–344. [CrossRef]
29. Teng, Y.; Chin, W.; Chang, D.; Chen, P.; Chen, P. VLSI Architecture of S-Box With High Area Efficiency Based on Composite Field Arithmetic. *IEEE Access* **2022**, *10*, 2721–2728. [CrossRef]
30. Dao, B.; Hoang, T.; Le, A.; Tsukamoto, A.; Suzaki, K.; Pham, C. Exploiting the Back-Gate Biasing Technique as a Countermeasure Against Power Analysis Attacks. *IEEE Access* **2021**, *9*, 24768–24786. [CrossRef]
31. Hu, W.; Wu, L.; Wang, A.; Xie, X.; Zhu, Z.; Luo, S. Adaptive Chosen-Plaintext Correlation Power Analysis. In Proceedings of the Tenth International Conference on Computational Intelligence and Security, CIS 2014, Kunming, China, 15–16 November 2014; pp. 494–498. [CrossRef]
32. Ou, C.; Lam, S.; Sun, D.; Zhou, X.; Qiao, K.; Wang, Q. SNR-Centric Power Trace Extractors for Side-Channel Attacks. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2021**, *40*, 620–632. [CrossRef]
33. Yang, M.; Meng, Q.; Wang, A.; Liu, X. Template Attacks Based on the Multivariate Joint Distribution. *IACR Cryptol. ePrint Arch.* **2020**, 1164. Available online: https://eprint.iacr.org/2020/1164 (accessed on 25 February 2024).
34. Test Vector Leakage Assessment (TVLA) Methodology in Practice. Available online: https://www.semanticscholar.org/paper/Test-Vector-Leakage-Assessment-(-TVLA-)-methodology-Becker-Cooper/60b993cb11fff28c9ea657b0e2882867b8f810e1 (accessed on 9 November 2023).
35. Mangard, S. Hardware Countermeasures against DPA ? A Statistical Analysis of Their Effectiveness. In Proceedings of the Topics in Cryptology—CT-RSA 2004, the Cryptographers' Track at the RSA Conference 2004, San Francisco, CA, USA, 23–27 February 2004; Lecture Notes in Computer Science; Okamoto, T., Ed.; Springer: Berlin/Heidelberg, Germany, 2004; Volume 2964, pp. 222–235. [CrossRef]
36. Siemens. Questa Advanced Simulato. Available online: https://eda.sw.siemens.com/en-US/ic/questa/simulation/advanced-simulator/ (accessed on 8 May 2021).
37. Cadence. Cadence Genus Sythnsis Solution. Available online: https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/synthesis/genus-synthesis-solution.html (accessed on 8 May 2021).
38. Synopsys. Synopsys SpyGlass Power. Available online: https://www.synopsys.com/verification/static-and-formal-verification/spyglass/spyglass-power.html (accessed on 8 May 2021).

39. Technology, N. CW305 Artix FPGA Target Board. Available online: http://store.newae.com/cw305-artix-fpga-target-board/ (accessed on 15 April 2023).
40. Bronchain, O. The Side-Channel Analysis Library (SCALib). 2021. Available online: https://github.com/simple-crypto/SCALib (accessed on 15 April 2023).
41. Schramm, K.; Leander, G.; Felke, P.; Paar, C. A Collision-Attack on AES: Combining Side Channel- and Differential-Attack. In Proceedings of the Cryptographic Hardware and Embedded Systems—CHES 2004, 6th International Workshop, Cambridge, MA, USA, 11–13 August 2004; Lecture Notes in Computer Science; Joye, M., Quisquater, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2004; Volume 3156, pp. 163–175. [CrossRef]
42. Okeya, K.; Sakurai, K. A Second-Order DPA Attack Breaks a Window-Method Based Countermeasure against Side Channel Attacks. In Proceedings of the Information Security, 5th International Conference, ISC 2002, Sao Paulo, Brazil, 30 September–2 October 2002; Lecture Notes in Computer Science; Chan, A.H., Gligor, V.D., Eds.; Springer: Berlin/Heidelberg, Germany, 2002; Volume 2433, pp. 389–401. [CrossRef]
43. National Institute of Standards and Technology. Lightweight Cryptography. 2020. Available online: https://csrc.nist.gov/projects/lightweight-cryptography (accessed on 15 April 2023).
44. Aljuffri, A.; Reinbrecht, C.; Hamdioui, S.; Taouil, M.; Sepúlveda, J. Balanced Dual-Mask Protection Scheme for GIFT Cipher Against Power Attacks. In Proceedings of the 2022 IEEE 40th VLSI Test Symposium (VTS), San Diego, CA, USA, 25–27 April 2022; pp. 1–6. [CrossRef]