applied sciences

MDPI

Article

A Low-Overhead Countermeasure against Differential Power Analysis for AES Block Cipher

Muhammad Asfand Hafeez [1], Mohammad Mazyad Hazzazi [2], Hassan Tariq [1,*], Amer Aljaedi [3], Asfa Javed [1] and Adel R. Alharbi [3]

[1] Department of Electrical Engineering, School of Engineering, University of Management and Technology, Lahore 5770, Pakistan; muhammadasfandh@gmail.com (M.A.H.); asfa.javed@umt.edu.pk (A.J.)
[2] Department of Mathematics, College of Science, King Khalid University, Abha 61413, Saudi Arabia; mmhazzazi@kku.edu.sa
[3] College of Computing and Information Technology, University of Tabuk, Tabuk 71491, Saudi Arabia; aaljaedi@ut.edu.sa (A.A.); aalharbi@ut.edu.sa (A.R.A.)
* Correspondence: hassantariq@umt.edu.pk; Tel.: +92-321-4833556

Abstract: This paper presents the employment of a DPA attack on the NIST (National Institute of Standards and Technology) standardized AES (advance encryption standard) protocol for key retrieval and prevention. Towards key retrieval, we applied the DPA attack on AES to obtain a 128-bit secret key by measuring the power traces of the computations involved in the algorithm. In resistance to the DPA attack, we proposed a countermeasure, or a new modified masking scheme, comprising (i) Boolean and (ii) multiplicative masking, for linear and non-linear operations of AES, respectively. Furthermore, we improved the complexity involved in Boolean masking by introducing Rebecca's approximation. Moreover, we provide a novel solution to tackle the zero mask problem in multiplicative masking. To evaluate the power traces, we propose our custom correlation technique, which results in a decrease in the calculation time. The synthesis results for original implementation (without countermeasure) and inclusion of countermeasure are given on a Zynq 7020 FPGA (Artix-7 device). It takes 424 FPGA slices when implemented without considering the countermeasure, whereas 714 slices are required to implement AES with the inclusion of the proposed countermeasure. Consequently, the implementation results provide the acceptability of this work for area-constrained applications that require prevention against DPA attacks.

Keywords: AES; block cipher; side-channel attacks; differential power analysis

Citation: Asfand Hafeez, M.; Mazyad Hazzazi, M.; Tariq, H.; Aljaedi, A.; Javed, A.; Alharbi, A.R. A Low-Overhead Countermeasure against Differential Power Analysis for AES Block Cipher. *Appl. Sci.* **2021**, *11*, 10314. https://doi.org/10.3390/app112110314

Academic Editor: Arcangelo Castiglione

Received: 21 September 2021
Accepted: 31 October 2021
Published: 3 November 2021

1. Introduction

Cryptography is one of the ways to secure unprotected data or information against unauthorized users on the unsecured internet. It comprises two types: (i) symmetric and (ii) asymmetric. The prior is beneficial to achieve low-area and power designs, whereas the former is more convenient to acquire high security [1,2]. Several applications are available in the literature, i.e., the internet of things, wireless sensor networks, radio frequency identification, and many more, that require low-area and power designs of cryptographic algorithms [3].

The security strength of symmetric algorithms depends on the structure of the cryptographic algorithm/protocol. For example, NIST (National Institute of Standards and Technology) defines the AES (advanced encryption standard) as a standard to prevent the current communications in terms of encryption and decryption from unwanted users [4]. It has a substitution–permutation structure. The encryption determines the transformation of plaintext to ciphertext, using a sequence of rules (called an algorithm/protocol). Decryption is the reverse of encryption. The plaintext is the original information, while a ciphertext is the transformed plaintext into an unreadable format. The term, i.e., substitution, replaces plaintext letters or strings of letters with letters, numbers, or symbols [3]. On the other

hand, permutation uses the plaintext message letters but rearranges them in a sequence of order.

There are two different approaches, i.e., software and hardware, to implement cryptographic algorithms. The latter is convenient to achieve higher throughputs while the prior is beneficial to achieve higher flexibility [1,3,5]. Concerning hardware prototyping, a side-channel attack (SCA) is one of the techniques to expose the security of the targeted crypto algorithm by analyzing its power consumption. The instantaneous power consumption of the cryptographic device depends on the data processing and operations. An SCA tries to extract secrets from a chip or a system by measuring and analyzing these operations and power consumption [6]. These type of attacks uses the fact that an intermediate value is processed during the execution of the targeted cryptographic algorithm and the acquired value is correlated with the power consumption of the device [7].

Many side-channel analysis techniques have proven successful in cracking an algorithmically resistant cryptographic operation and retrieving the secret key, posing a severe threat to modules that incorporate cryptographic systems. To achieve this, the number of attacks can be classified into three different types, i.e., (i) invasive, (ii) non-invasive, and (iii) semi-invasive [8]. Invasive attacks involve the alteration of the physical attributes of a chip in an irreversible way. The non-invasive attacks do not necessitate any initial device preparations and do not physically affect the device under evaluation. The attacker can either tap into the device's cables or connect it to a test circuit for analysis [9]. In the case of semi-invasive attacks, UV light, X-rays, and other forms of ionizing radiation, lasers, and electromagnetic fields could all be used to carry out the attack. Moreover, semi-invasive attacks are more difficult to implement than non-invasive ones, as they involve chip depackaging.

It is important to note that we have targeted a differential power analysis (DPA) attack, as this is non-invasive. Furthermore, the non-invasive attacks do not alter the device or its components. Moreover, attackers mostly exploit this method to retrieve information from cryptographic devices (as it is an inexpensive method). Based on this observation, we have also used a DPA method (in this paper) to retrieve the security key of the selected AES algorithm.

1.1. Existing Solutions with Respect to Side-Channel Attacks and Their Countermeasures

Solutions described only the SCA attacks. In [10], a wireless interceptive SCA is performed to break the AES-128 key. They analyzed the correlation electromagnetic analysis (CEMA) of the cryptographic device for the SCA attack and successfully revealed the encryption key in 20,000 electromagnetic traces.

The solutions, published in [11–13], perform a correlation power analysis (CPA) attack on AES by analyzing the power leakage of the device during the execution of the encryption process, and successfully retrieve the entire secret key. Generally, the CPA is an attack that allows attackers to perceive a secret encryption key that is stored on a victim device. The solutions described in [14–17] perform different orders of the DPA attack to find the encryption key. They use the CPA method to reduce the processing time for successful DPA attacks. In [18], a CEMA based attack is performed based on the correlation of energy traces and intermediate data in the AES implemented on the SAKURA-G FPGA (field-programmable gate array). Recently, a deep-learning SCA was performed in [19]. It recovers the secret key using a combination of different convolutional neural network (CNN) classifiers with different attack points, which reduce the number of power traces required to recover the key. In [20], a fault analysis attack on the FPGA implementation of AES is performed. Moreover, the temporal and spatial analyses of the rounds impacted by the fault injection process are analyzed.

A DPA attack using flash-based FPGA technology on AES is performed in [15]. For FPGAs, a lightweight AES architecture is presented in [21], which uses a partial separated dynamic differential logic (partial-SDDL) technique. In [22], a new method is presented that follows the sbox-reuse concept to secure the sbox of a block cipher. They first turned

the multi-sbox into 4 × 4 permutations, then utilized permutations with more than one algebraic degree to create a special reusable sbox, and then numbered the 4 × 4 permutation as input again.

Solutions provide the SCA attacks with their countermeasures. The power consumption of the target device depends on the data it processes and the operation it performs (it exposes the algorithm to SCA). To prevent this situation, several countermeasures have been developed/proposed in the literature to reduce the possibility of an attack on the cryptography algorithm.

A DPA attack on FPGA implementation of AES is given in [23]. Additionally, two different countermeasures, i.e., (i) register swapping, and (ii) random recharging, which increases the complexity of the attack, were provided. The solution published in [24] proposed a new countermeasure based on the phase-locked loop (PLL) technique. In this method, they placed a PLL in the power path of an AES, which suppresses the power characteristics of the encryption operation. In [25], a SPA attack on AES is performed to obtain the secret key. Moreover, a countermeasure based on adiabatic logic (implemented in the AddRoundKey block of AES) is provided. This breaks the dependency between the power consumption of the device and the secret key. Concerning the solution given in [26], the CPA and DPA attacks on AES are presented using Arduino devices. It is analyzed that both forms of the attack are feasible on the Arduino. The RTL (register transfer level) based countermeasures are designed in [27].

In [28], various on-chip voltage converters as a countermeasure against DPA attacks are described. Moreover, a comparison with the security strength of the proposed on-chip converter-reshuffling (CoRe) regulator to the security of conventional on-chip voltage regulators is provided. Similarly in [15], the authors check the vulnerability of AES against first order and second-order DPA attacks. They specify some principles to demonstrate the weak positions of AES which are vulnerable to DPA attack. In [29], researchers develop techniques at the logic level. The method employs logic gates with power consumption, which is independent of the data signals, and therefore, the technique removes the foundation for DPA.

Additional security issues with respect to commercial devices. Cryptography devices are extremely vulnerable when it comes to industrial applications. DPA attacks could lead toward monetary losses, data integrity and sensitive data leakage. These security threats have a great impact on the national or community level if we consider the broad spectrum of users. We present two different case studies that express the true need for the countermeasure and it leads toward the shortcoming as described in the forthcoming section. Case-study#1: In [30], P. Kocher first provided the hardware demonstration of DPA attacks on a smart card. He analyzed the power traces of the smart card and then revealed the key of encrypted data. This attack embarked on a new career of security threats, which led to different countermeasures. Case-study#2 is a recent example of the side-channel attack on AMD (advanced micro devices) processors. The side-channel attack provides access to a malicious application on the attacked system to exploit the hidden flaws inside the CPU (central processing unit). Their attack makes use of memory associated with apps (means applications) to obtain sensitive information, e.g., encryption keys, and passwords. They leverage power and timing measurements of prefetch instructions. Their analysis highlights the greater leakage of AMD processors as compared to a prefetch-based attack on Intel processors [31].

1.2. Limitations

Section 1.1 reveals that simple hiding and masking techniques are extensively used in the literature to mitigate the DPA attack on the cryptographic algorithm(s) [10–13]. Therefore, these techniques are not power efficient and increase the hardware resources (area). In other words, hardware-based countermeasures increased the implementation cost, which is not preferred in most cases (for area limited applications, i.e., radio-frequency identification, wireless sensor networks, and many more). Countermeasures based on hiding can easily

be exposed to higher-order DPA because of the dependency of higher-order on first order equations [15]. In many papers, the AddRoundKey block of AES is altered into a logic-based design [25,27,29]. It breaks the dependency of the power trace during the execution of the encryption process (which is adequate). On the other hand, it alters the characteristics of the AES algorithm, making it vulnerable to cryptanalysis attacks. Subsequently, there is a need for area-efficient designs in light of countermeasures to mitigate the DPA attacks over cryptographic algorithm(s).

1.3. Our Contributions

To apply the DPA attack, we have selected an open-source core of AES from Opencores [32]. Therefore, the contributions to this work are given as follows:

- **Setting to apply DPA attack.** To execute the DPA attack on modern low-power FPGA devices, a large number of power traces are required. This results in an increase in the correlation calculation time. Therefore, to reduce this calculation time, we proposed our custom correlation technique (details are given in Section 3.1).
- **Secret key identification using DPA attack.** We applied the DPA attack (our custom correlation technique) on our FPGA implementation to obtain the secret key by measuring the power traces of the computations involved in the AES algorithm.
- **Countermeasure to mitigate the DPA attacks.** To provide resistance against DPA attacks, we provided a countermeasure using Boolean and multiplicative masking for the linear (i.e., ShiftRows, MixColumns, and AddRoundkey) and non-linear functions (i.e., SubBytes) of the AES, respectively. The descriptions are given in Section 3.2.

We synthesized two different Verilog implemented designs, i.e., (i) without countermeasure (we termed this as DESIGN-I), and (ii) the inclusion of countermeasure (we named this as DESIGN-II), using the Vivado IDE tool. The implementation results for DESIGN-I and DESIGN-II are given on a Zynq 7020 FPGA device. Our DESIGN-I utilizes only the 424 FPGA slices. On the other hand, our DESIGN-II takes 714 FPGA slices. Subtraction of FPGA slices of DESIGN-I from DESIGN-II determines the FPGA slices (i.e., 290) required for the implementation of additional logic (i.e., our proposed countermeasure). With the utilization of these hardware resources, we have shown the extraction and prevention of a 128-bit secret key of AES. Consequently, our DESIGN-II provides the acceptability of this work for applications that require prevention against DPA attacks.

This paper is organized as follows: Section 2 provides the required mathematical background about the implementation of AES. Section 3 describes the proposed method to provide resistance against the DPA attack on FPGA. The implementation results are described in Section 4. Finally, this work is concluded in Section 6.

2. Preliminaries

The fundamental working of the AES is shown in Figure 1. It takes a 128-bit key (for a variant of AES-128) and plaintext as input and results in a 128-bit ciphertext as an output. We refer interested readers to [33] for the complete mathematical descriptions and formulations.

A variant of AES, named AES-128, encryption starts from the initial round followed with an additional ten rounds. Each round consists of the following four operations [34]:

- **SubBytes** splits the input data into bytes and then passes the input byte by byte through the substitution box (S-box). It is a non-linear substitution.
- **ShiftRows** determines each row of the 128-bit internal state of the cipher shifted by the fixed amount.
- **MixColumns** provides diffusion to the AES. It performs linear transformation, which makes AES secure against many attacks.
- **AddRoundKey** is responsible for performing a bitwise exclusive-OR (XOR) operation in each round.

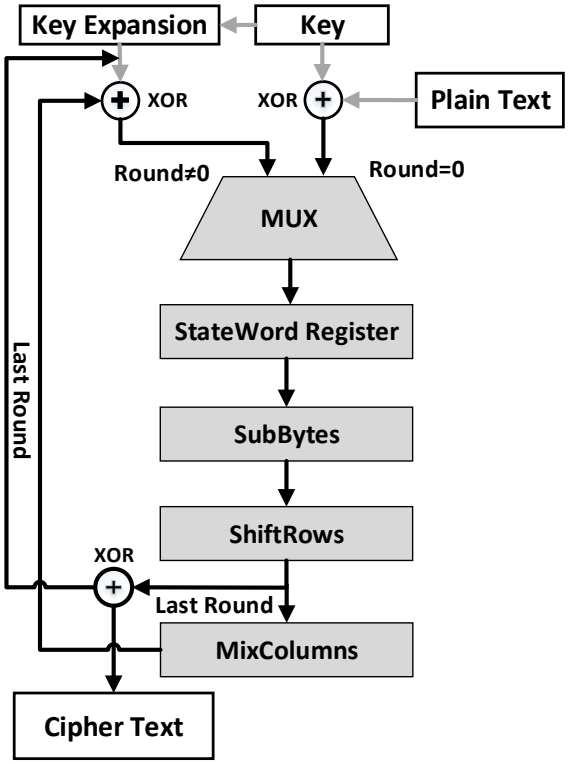

Figure 1. Structure of the AES algorithm.

In the last round, the MixColumns operation can be avoided. The ciphertext, at the end of the last round, presents the value of the StateWord Register [35]. It is important to note that this is the place where the DPA attack can be performed by inverting ShiftRows and the SubBytes operations on the selected byte. Despite these essential operations (SubBytes, ShiftRows, MixColumns, and AddRoundKey), a key expansion is responsible for generating a distinct key for each round during the execution of the AES protocol. AddRoundKey of AES completely transform the nature of the data, as it performs an XOR operation in each round. The attacker exploits the power consumption, as it solely depends on the processed data. The attacker records a large number of power tracks with the fact that the complexity of data processing is linear to the power consumption [36]. Moreover, the attacker already knows the working of the AES algorithm, and therefore, he/she targets the intermediate stage, e.g., AddRoundKey, which completely changes the nature of the data [37]. The attacker can also exploit the linearity property of ShiftRows and MixColumns. This vulnerability enables an attacker to build a hypothetical model to extract the secret key.

3. DPA Attack and Our Proposed Countermeasure

This section describes the perspectives to apply the DPA attack in Section 3.1. The mathematical model of our proposed countermeasure to mitigate the DPA attack on FPGA implementation of AES is presented in Section 3.2.

3.1. DPA Attack

The DPA attack is most popular attack based on power analysis. The interesting fact about DPA is that it does not require information about the attacked device. It requires a large number of power traces along with software computation to reveal the secret key of a cryptographic device. DPA exploits power consumption at a specified instant of time, as then it used power traces as a function of processed data. To apply the DPA attack, the initial step is to collect the power samples during the computation of the operations involved in the selected algorithm or protocol (AES in our case). The method to accumulate

the power traces is described later in Section 4.1. Therefore, the subsequent steps are described in the text given below.

Step 1: Selection of the intermediate values inside the algorithm. The first step of a DPA attack is to choose an intermediate result from the cryptographic algorithm running on a cryptographic device. Then, the intermediate results can be used to reveal the part of a secret key. The function $f(d, k)$ is used as the intermediate result. The value of d determines either the plain text or ciphered text, while k is a selected part of the key.

Step 2: Measuring the power consumption. In the second stage, we monitor the power traces of the cryptographic device during the execution of the operations associated with it. When performing encryption or decryption operations, the attacker must know the relevant data value d (this is required in the calculation of intermediate results, as described in step 1). For DPA attacks, it is crucial that the measured traces are aligned appropriately.

Step 3: Calculating intermediate values of hypothetical model. The next stage in the attack is to compute a hypothetical intermediate value for each conceivable k value. Using Equation (1), an attacker may simply compute the hypothetical intermediate values $f(d, k)$ for all encryption runs (we denote with D) and key hypotheses (we termed with K) on given data vector d and key hypothesis k.

$$v_{i,j} = f(d_i, k_j) \tag{1}$$

In Equation (1), i moves from $1, \ldots, D$ and j moves from $1, \ldots, K$. This means that the result stored in a matrix v is of size $D \times K$.

Step 4: Mapping intermediate values to estimate the power consumption. The DPA attack proceeds by mapping the hypothetical intermediate values v to a matrix H of hypothetical power consumption values. The power consumption of the device is simulated using the Hamming-distance and Hamming-weight model approaches for each hypothetical intermediate value $V_{i,j}$ in order to generate a hypothetical power consumption value $h_{i,j}$. The Hamming-distance and Hamming-weight models are the most widely utilized power models for mapping v to H.

Step 5: Comparing the hypothetical model with the measured power traces. The final step of DPA is to compare each key hypothesis with the recorded traces, using correlation. The result after comparison is stored to matrix R (size is $K \times T$ (this matrix contains the measured values of power traces from the attacked device)) after the comparison/correlation between matrix H and matrix T. This comparison is based on the Pearson correlation.

3.1.1. Pearson Correlation

As described in the previous section, the key guess is accomplished using the correlation coefficient between each sample and a predicted power leakage. To achieve this, the Pearson correlation coefficient method is one of the approach(es), as (it is assumed that) there is a linear dependence between the measured variables and the leakage. Therefore, the Pearson correlation coefficient between two random variables, i.e., X, and Y, can be calculated using Equation (2).

$$\rho_{X,Y} = \frac{\mathrm{cov}(X, Y)}{\sigma_X \sigma_Y} \tag{2}$$

In Equation (2), $cov(X, Y)$ is the covariance between two variables, i.e., X, and Y. The σ_X and σ_Y determines the standard deviations of the variables X and Y, respectively. The statistical sample (or the normalized measurement of a covariance such that the resultant value always lies between the range -1 and 1) of the Pearson correlation coefficient can be calculated using Equation (3).

$$\rho_{X,Y} = \frac{\mathrm{cov}(X, Y)}{\sqrt{\mathrm{E}\left[(X - \mu_X)^2\right]}\sqrt{\mathrm{E}\left[(Y - \mu_Y)^2\right]}} = \frac{\mathrm{E}[(X - \mu_X)(Y - \mu_Y)]}{\sqrt{\mathrm{E}\left[(X - \mu_X)^2\right]}\sqrt{\mathrm{E}\left[(Y - \mu_Y)^2\right]}} \tag{3}$$

In Equation (3), the values for μ_X and μ_Y determine the mean of variables X and Y, respectively. Furthermore, **E** shows the expectation or expected value. It is important to provide that the Pearson correlation is not much efficient (as it performs correlation in a sequence of rows one after the another) and is challenging to extract the correlation coefficient between two sets of stochastic variables (i.e., X, and Y), due to the presence of excessive noise. Additionally, this results in an increase in the correlation calculation time. Therefore, we have described our custom Pearson correlation function (in the succeeding Section 3.1.2) that optimizes the computation based on the statistical implementation. Moreover, it also reduces the calculation time.

3.1.2. Our Proposed Custom Correlation Algorithm

The sequence of instructions for our proposed custom correlation function is given in Algorithm 1. It takes the power traces (i.e., a) and a hypothetical model (i.e., b) as an input. It results in the correlation of a and b as an output (i.e., C). At the start, the algorithm first creates two variables for hypothetical values (a) and power traces (b) with their matrix size. Then, function **REPEAT_MEAN** removes mean values from the original matrices of hypothetical values and power traces (receptive process). Then, it multiplies the matrices as calculated in the previous function to obtain the dot product. Finally, the resultant of the product is divided with another product of the variance and square root to obtain the correlation.

Algorithm 1: Proposed custom correlation algorithm.

Input: a (*power traces*), b (*hopothetical model*)
Output: $C \leftarrow custom\_corr(a, b)$
1 $[X_r, X_c] \leftarrow$ SIZE(a)
2 $[Y_r, Y_c] \leftarrow$ SIZE(b)
3 $a \leftarrow a$ - REPEAT_MEAN(a, X_r)
4 $b \leftarrow b$ - REPEAT_MEAN(b, Y_r)
5 $C \leftarrow a' * b$
6 $C \leftarrow C /$ SQUARE_SUM(a, Y_c)
7 $C \leftarrow C /$ SQUARE_SUM(b, X_c)

The Algorithm 1 is much faster than the correlation function, provided in Equation (3). This is due to the correlation comparison performed in a matrix representation rather than correlating row by row (as used in Equation (3)). Our proposed custom correlation function is also faster than the ***corrcoef*** (this is a built-in function of MATLAB to correlate two vectors based on Pearson correlation) as it is based on the matrix comparison. On the other hand, the MATLAB function, i.e., ***corrcoef***, executes correlation similarly to Equation (3) (row by row). In terms of computation time, the proposed Algorithm 1 is 16.24 times (ratio of 9.26 to 0.57) faster as compared to ***corrcoef***. The time required to execute one byte for the correlation is 9.26 (for ***corrcoef***) and 0.57 (for Algorithm 1), respectively.

3.2. Our Proposed Countermeasure

To mitigate the DPA attacks, we have proposed our countermeasure using the masking technique. As mentioned earlier in Section 2, AES consists of four functions (i.e., ShiftRows, MixColumns, AddRoundkey and SubBytes). The ShiftRows, MixColumns, and AddRoundkey functions are linear, while a SubBytes function of AES is non-linear. The attack could be employed on both linear and non-linear functions of the AES. We propose a countermeasure for both linear and non-linear functions. Therefore, Boolean masking can be applied easily to the linear functions. The problem arises in the case of a non-linear function, where Boolean masking could not be applied. According to [23,29], the SubBytes function would be easy to mask through multiplicative masking. Consequently, we employed Boolean masking over the linear functions of AES (ShiftRows, MixColumns, and AddRoundkey), while multiplicative masking is applied to the non-linear function (SubBytes).

3.2.1. Multiplicative Masking

Original concept. In masking, the intention is to protect sensitive information while providing a functional alternative when real information is not needed. Generally, masking is the different representation of real information. It can distinguish using the splitting of a function (i.e., $f(x)$) into $n+1$ shares, as shown in Equation (4). The Boolean sharing is computed using a bit multiplication operation over the split shares.

$$x = ({q_0}^x, ..., {q_d}^x) \Longleftrightarrow x = \bigotimes_{i=0}^{d} {q_i}^x \tag{4}$$

In Equation (4), we use ${q_\mathrm{i}}^x$ to present Boolean shares of a function defined in terms of x. The ${q_0}^x \ldots {q_\mathrm{d}}^x$ are the $n+1$ Boolean shares of x, where n is the natural number.

Zero value problem. Initially, the significant security flaw of multiplicative masking was highlighted in [38]. It does not mask the value 0. The mean power consumption of a single share ${q_\mathrm{i}}^x$ reveals whether the underlying secret is zero or non-zero since $\mathrm{E}[{q_\mathrm{i}}^x | x = 0] = \mathrm{E}[{q_\mathrm{i}}^x | x \neq 0]$ for any share index i [39]. This means that for any number of shares, the original multiplicative masking scheme is vulnerable to first-order DPA.

Solution. Our proposed solution to avoid the zero value problem is illustrated in Figure 2. It realizes the fact that if we invert zero, we obtain zero, while if we invert one, we obtain one. To achieve this, we used only the Kronecker delta function (see $\delta(x)$ in Figure 2). The corresponding mathematical formulations are shown in Equations (5)–(7), respectively.

$$x^{-1} = x \text{ for } x \in \{0, 1\} \tag{5}$$

$$\delta(x) = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{if } x \neq 0 \end{cases} \tag{6}$$

$$x^{-1} = (x \oplus \delta(x))^{-1} \oplus \delta(x) \tag{7}$$

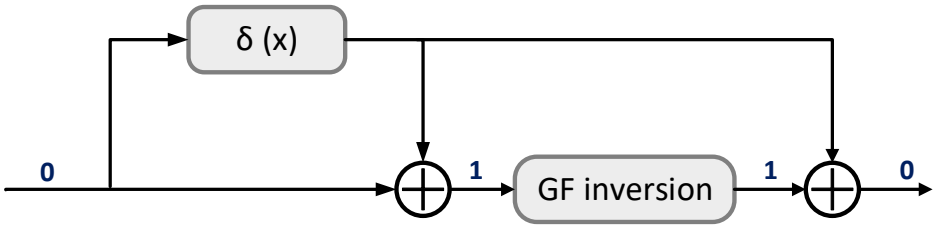

Figure 2. Our proposed solution to avoid the zero value problem in multiplicative masking.

For any bit value (0 or 1), Equation (5) determines the inverse of a Boolean share. The Kronecker delta function is provided in Equation (6). Finally, Equation (7) reflects the implementation of Figure 2. It is obtained with the substitution of Equations (5) and (6).

3.2.2. Boolean Masking

Original concept. In Boolean masking, the variable sharing splits into $d+1$ random shares (i.e., $s_0, s_1, s_2 \ldots s_d$). The resultant Boolean share is computed using the exclusive-OR operation over split random shares ($s_0 \oplus \cdots \oplus s_{d-1} \oplus s$). It shows that every share of s_i is statistically independent of s and uniformly distributed.

Problem implementing Boolean masking. When implementing circuits considering the concepts related to Boolean masking over application-specific integrated circuits (ASICs) or FPGA devices, the use of different wires with different length results in masking being less appropriate. The reason is that glitches may arise when especially the combinational logic for design implementation/modeling is used. The term "glitch" determines the unnecessary signal transitions without functionality.

Solution. To avoid glitches (useless transitions) in our design implementation, we used the Fourier expansion (in general, the Fourier expansion or series is a periodic function,

which is composed of harmonically related sinusoids, combined by a weighted summation. For complete mathematical descriptions and formulations, we refer readers to [40]) along with the REBECCA's approximation. The Fourier expansion of a Boolean function derives the correlation sets. Moreover, the Fourier expansion represents the Boolean functions as a polynomial over the real domain $\{1, \neg 1\}$, as shown in Equation (8). REBECCA considers glitches by demonstrating the stable and transient relationship of gates. To verify the resultant correlation as an output, it requires a correlation set as an input.

$$\begin{aligned} &\{\top, \bot\} \rightarrow \{-1, +1\} \\ &(x \oplus y) \rightarrow (xy) \\ &(x \wedge y) \rightarrow \left(\tfrac{1}{2} + \tfrac{1}{2}x + \tfrac{1}{2}y - \tfrac{1}{2}xy\right) \end{aligned} \tag{8}$$

4. Implementation Results

The experimental setup to implement our proposed DESIGN-I and DESIGN-II is provided in Section 4.1. The area and timing results are given in Section 4.2. The secret key identification and prevention against DPA attack are described in Section 4.3.

4.1. Experimental Setup

We have implemented two designs (DESIGN-I (without countermeasure) and DESIGN-II (inclusion of countermeasure)) in Verilog HDL on Digilent ZYBO development board. It contains a Zynq 7020 FPGA device for logic implementation. To apply the SCA attack on the selected board, certain alterations are required, i.e., (i) removal of the decoupling capacitors (C179, C180, C224 and C225), and (ii) replacement of a shunt resistance, i.e., R265, with a value 0.1 Ω [41]. This will sink the total current of the board through this resistance (0.1 Ω). The purpose of using a 0.1 Ω resistor is to reduce the static power, as Zynq SoC (with both FPGA logic and dual-core processor) has much larger static power consumption.

Implementation parameters. We have implemented a 128-bit variant of AES block cipher. It means our designs contain a 16-byte key, having values (0xA1, 0x78, 0x5B, 0x77, 0x2D, 0xCD, 0xD4, 0x1F, 0x9E, 0x55, 0xA3, 0x45, 0x7C, 0x8B, 0x26, 0xEC). All these values are in hexadecimal representation.

Capturing power traces. As shown in Figure 3, the probes of the oscilloscope are connected across the shunt resistance that we replace to 0.1 Ω. We captured the power traces as the AES performed encryption over the data (so the attack will have the highest probability of success). An external trigger signal is generated from FPGA to inform the oscilloscope when it will start to capture the data and vice versa. To set and reset the external trigger, we used pin 8 of port 2. The trigger is activated during the computation of the **AddRoundKey** and **SubBytes** functions. These two functions are executed 10 times for each plaintext value and the average value of 1000 power traces is captured by the oscilloscope. Note that the AddRoundKey function is simply the exclusive-OR of each byte of data with the key. So, it does not consume too much power. The SubBytes results in a leakage of information, as the cryptographic device uses too much power (because Sbox is looking for the bytes obtained after the computation of an exclusive-OR operation). This enables us to attack and perform such power analysis. After capturing the power traces, we developed a MATLAB script to perform the DPA attack, using Pearson correlation. Recall again that we also introduced a custom correlation algorithm to speed up the attack process (see Section 3.1.2).

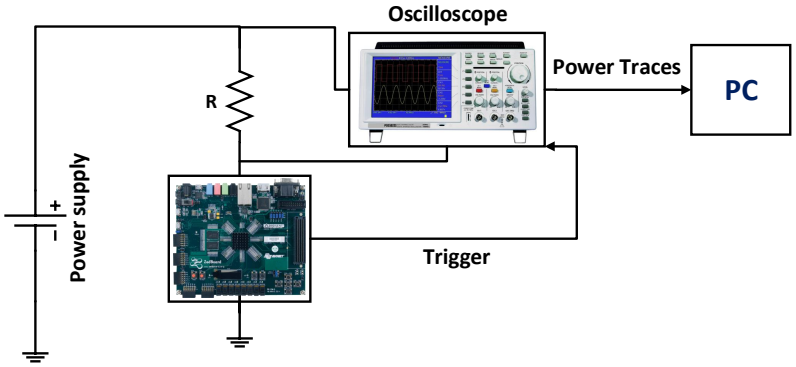

Figure 3. Hardware setup to collect power traces.

4.2. Area and Timing Results

The area and timing results (in terms of clock cycles) are given in Table 1. Column one provides the implemented design. The achieved clock frequency (freq. in MHz) is provided in column two. The area information in terms of slices, look-up-tables (LUTs) and flip-flops (FFs) are shown in columns three to five, respectively. The timing information in terms of required clock cycles to execute encryption (ENC) and decryption operations (DEC) is presented in columns six to seven, respectively.

Table 1. Area and clock cycles information on a Zynq 7020 SoC (Artix-7 device).

Designs	Freq.	Utilized Area			Clock Cycles	
		Slices	LUTs	FFs	ENC	DEC
DESIGN-I	250	424	1314	538	12	12
DESIGN-II	205 (−18%)	714 (+68%)	2110 (+60%)	1089 (+102%)	12	12

DESIGN-I is without the countermeasure. DESIGN-II is with the inclusion of countermeasure.

As shown in Table 1, DESIGN-I achieves 1.21 times higher clock frequency as compared to DESIGN-II. When considering the hardware utilization for comparison, DESIGN-II utilizes 1.68, 1.60 and 2.02 times higher slices, LUTs and FFs, respectively. This is due to the required additional logic for the implementation of our proposed countermeasure and Algorithm 1. Subtraction of FPGA slices of DESIGN-I from DESIGN-II determines the FPGA slices (i.e., 290) required for the implementation of additional logic. The percent increase in (when comparing DESIGN-II with DESIGN-I) slices, LUTs and FFs is also given in Table 1. The clock cycles required for the execution of encryption and decryption operations are identical in DESIGN-I and DESIGN-II (i.e., 12). Concerning the computational complexity (in terms of time required to perform one encryption and decryption operations) of our DESIGN-I, each encryption/decryption operation require 0.048 ms (ratio of 12 over 250). Similarly, the computational complexity of our DESIGN-II is 0.058 ms (ratio of 12 over 205). With the utilization of lower hardware resources, presented in Table 1, we believe our DESIGN-II is more beneficial for the applications that require prevention against DPA attacks.

It is important to note that the additional FPGA slices (290) to implement our proposed countermeasure is 68% of our DESIGN-I (where the achieved slices value is 424). It reveals that the proposed countermeasure is not suitable for the lightweight implementations of AES. However, it is preferred to use in applications that demand high-speed AES implementations. For example, in 2019, a high-speed area-efficient implementation of AES on Xilinx Artix-7 (XC7A100T-1CSG324C) FPGA is reported in [42]. Their architecture utilizes 15,850 slices. Our proposed countermeasure has more meaning in such an implementation because it has 54.6 times the hardware resources of [42].

4.3. Secret Key Identification and Prevention against DPA Attacks

Once after the collection of power traces (as described previously in Section 4.1), we used the MATLAB script along with Algorithm 1 (implemented in MATLAB) to predict the key-bit values. Moreover, for each power trace, the prediction was made on the highest correlations calculated and plotted for the average square of the total amplitude of the differentials on each keybyte separately. The predicted bit values or correlation plot for the first-byte of the 128-bit secret key (in decimal form) are shown in Figure 4. Furthermore, the corresponding correlated values for each key byte are given in Table 2. For a first-byte of a 128-bit secret key, the power traces of DPA attack with the inclusion of our proposed countermeasure is shown in Figure 5.

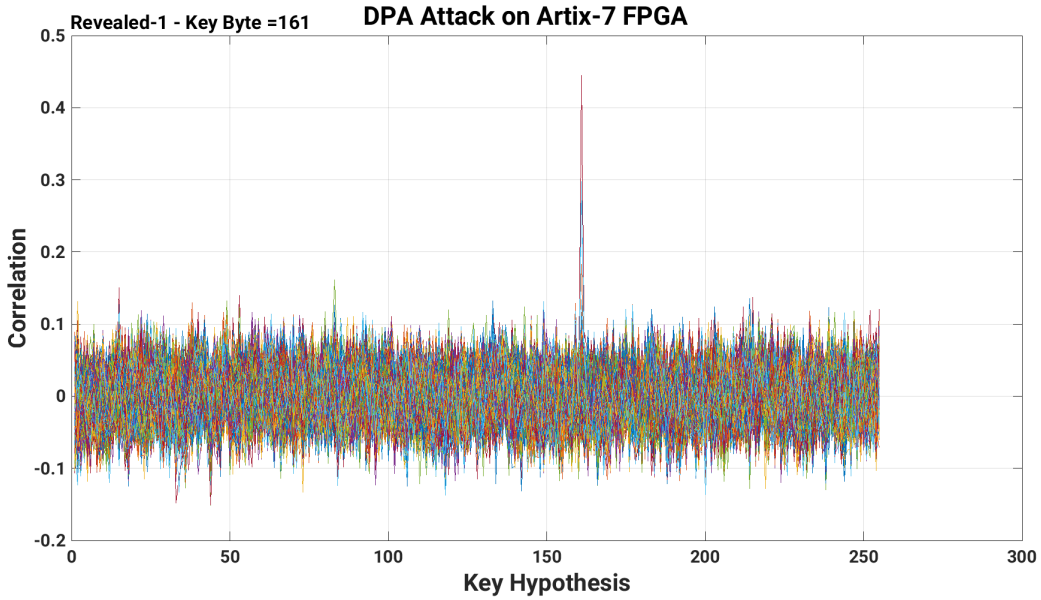

Figure 4. Retrieval of first-byte of a 128-bit secret key obtained after the execution of a DPA attack on DESIGN-I (AES without our proposed countermeasure).

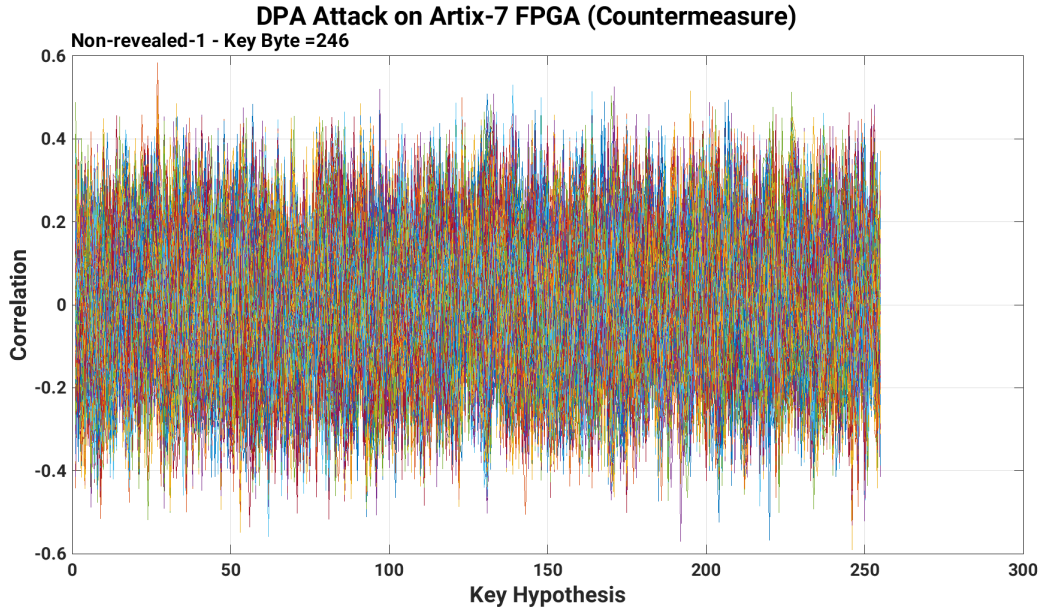

Figure 5. An erroneous retrieval of the first-byte of a 128-bit secret key obtained after the execution of a DPA attack on DESIGN-II (AES with inclusion of our proposed countermeasure).

Table 2. The achieved correlated values for 16 bytes of AES-128 key.

# of KeyBytes	Key Values (In Different Representations)		Correlation
	(In Decimal)	(In Hexadecimal)	
KeyByte 1	161	A1	0.445
KeyByte 2	120	78	0.493
KeyByte 3	91	5B	0.502
KeyByte 4	119	77	0.45
KeyByte 5	45	2D	0.525
KeyByte 6	205	CD	0.356
KeyByte 7	212	D4	0.481
KeyByte 8	31	1F	0.478
KeyByte 9	158	9E	0.505
KeyByte 10	85	55	0.293
KeyByte 11	163	A3	0.481
KeyByte 12	69	45	0.45
KeyByte 13	124	7C	0.513
KeyByte 14	139	8B	0.41
KeyByte 15	38	26	0.512
KeyByte 16	236	EC	0.447

The achieved values after correlation, presented in the last column (i.e., column four) of Table 2 reveals that we have successfully applied the DPA attack on the selected AES algorithm. Moreover, it passes the Pearson correlation test, as all these values (see last column of Table 1) are in the range -1 to 1. A larger peak in Figure 4 determines the identification of the first-byte (161 in a decimal) of a secret key. Including the first-byte, the identification of the remaining bytes of a 128-bit secret key is presented in Appendix A.

Figure 5 reveals that there are "ghost" peaks, which lead to wrong keys corresponding to the target Sbox. We also test the attack by increasing the power traces up to 5000, but still it leads to a wrong key guess. Including the first byte, the prevention against the remaining bytes of a 128-bit secret key is shown in Appendix A. As a result, we believe that our proposed countermeasure over AES resists the DPA attack.

5. Limitations of This Work

The state-of-the-art solutions guard the AES block cipher against DPA attacks. However, these solutions have a few limitations, such as area, security of linear and non-linear functions simultaneously, instantaneous power, etc. The techniques proposed in [36,37,43] tackle the DPA attack and its vulnerabilities. In contrast with our technique, we focused on area and security, simultaneously. The proposed security method protects both linear and non-linear functions of the AES algorithm. Moreover, the power leakage is approximately small as compared to the aforementioned solutions. Concerning our solution, our design could be implemented on FPGA and it could also be implemented as an embedded design, e.g., SoC (system on chip). This could be implemented on any SoC device, such as Zybo, Zedboard, Intel Arria, etc. This requires a C/C++ code, which will be executed on the processor to a special register inside the FPGA. The real-time control of the cipher/decipher is possible, which is controlled from a PC. However, this requires an external module that is integrated into the design for serializing the transmission, e.g., UART (universal asynchronous receiver–transmitter). These two limitations are user oriented, and we will address them in our future work for RFID (radio frequency identification)–based applications. Moreover, we mainly focused on the security side of the AES, which was also a primary goal of this work.

6. Conclusions

This paper presents the employment of a DPA attack on the NIST standardized AES algorithm for key retrieval and prevention. To retrieve the secret key, we have applied the DPA attack on AES to obtain a 128-bit secret key by measuring the power traces of

the computations involved in the algorithm. To provide resistance against the DPA attack, we have proposed a countermeasure based on the masking scheme: (i) Boolean and (ii) multiplicative. Moreover, we improved the complexity involved in the Boolean masking by introducing Rebecca's approximation. Additionally, we provided a novel solution to tackle the zero mask problem in multiplicative masking. We proposed our custom correlation technique, which resulted in a decrease in the calculation time. The synthesis results for DESIGN-I (an open-source core selected from GitHub) and DESIGN-II (inclusion of our countermeasure) are provided on a Zynq 7020 FPGA device. DESIGN-I takes 424 FPGA slices and achieves a clock frequency of 250 MHz. DESIGN-II requires 714 slices and achieves 205 MHz clock frequency. For both implementations (DESIGN-I and DESIGN-II), 12 clock cycles are required to execute one encryption and decryption operation, respectively. Therefore, the implementation results for our DESIGN-II provide the acceptability of this work for area-constrained applications that require prevention against DPA attacks.

Author Contributions: Conceptualization, H.T. and M.A.H.; methodology, M.M.H.; software, A.J.; validation, H.T. and A.R.A.; formal analysis, A.A.; investigation, A.A.; resources, M.M.H. and M.A.H.; data curation, H.T.; writing—original draft preparation, M.A.H. and H.T.; writing—review and editing, A.J. and M.M.H.; visualization, M.M.H.; supervision, A.R.A.; funding acquisition, A.A. All authors have read and agreed to the published version of the manuscript.

Funding: The author Mohammad Mazyad Hazzazi extends his gratitude to the Deanship of Scientific Research at King Khalid University for funding this work through a research group program under grant number R.G.P. 2/150/42.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Retrieval and Prevention of a 128-Bit Secret Key

The identification and prevention of an entire 128-bit secret key of AES is shown in Figures A1 and A2, respectively. There are 16 snippets in these figures (Figures A1 and A2). Each snippet determines one byte of a 128-bit key. The original selected values (also shown in a decimal representation in column two of Table 2) for the secret key are 161, 120, 91, 119, 45, 205, 212, 31, 158, 85, 163, 69, 124, 139, 38 and 236. Figure A1 reveals that the key byte values (shown in top on each snippet) are identical to the values provided in column two of Table 2. This means that we successfully obtain the 128-bit secret key of the selected AES algorithm. For the case of implemented countermeasure, the achieved values (given on top of each snippet in Figure A2) in decimal are 246, 129, 91, 3, 175, 113, 70, 137, 153, 71, 105, 103, 45, 223, 38 and 244. It can be seen that these values are distinct from the values reported in column two of Table 2. Subsequently, our proposed countermeasure provides resistance against DPA attacks (which was the persistence of this work).

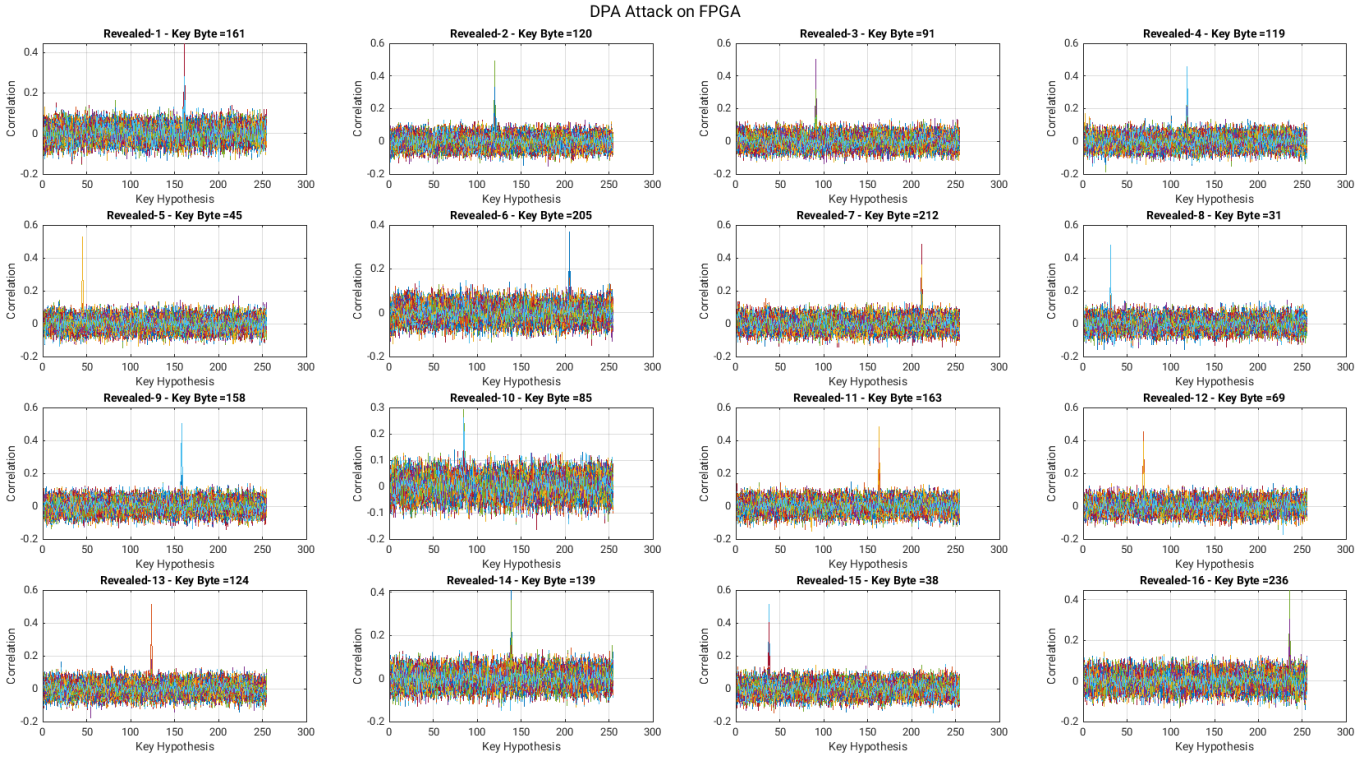

Figure A1. DPA attack on AES without the countermeasure.

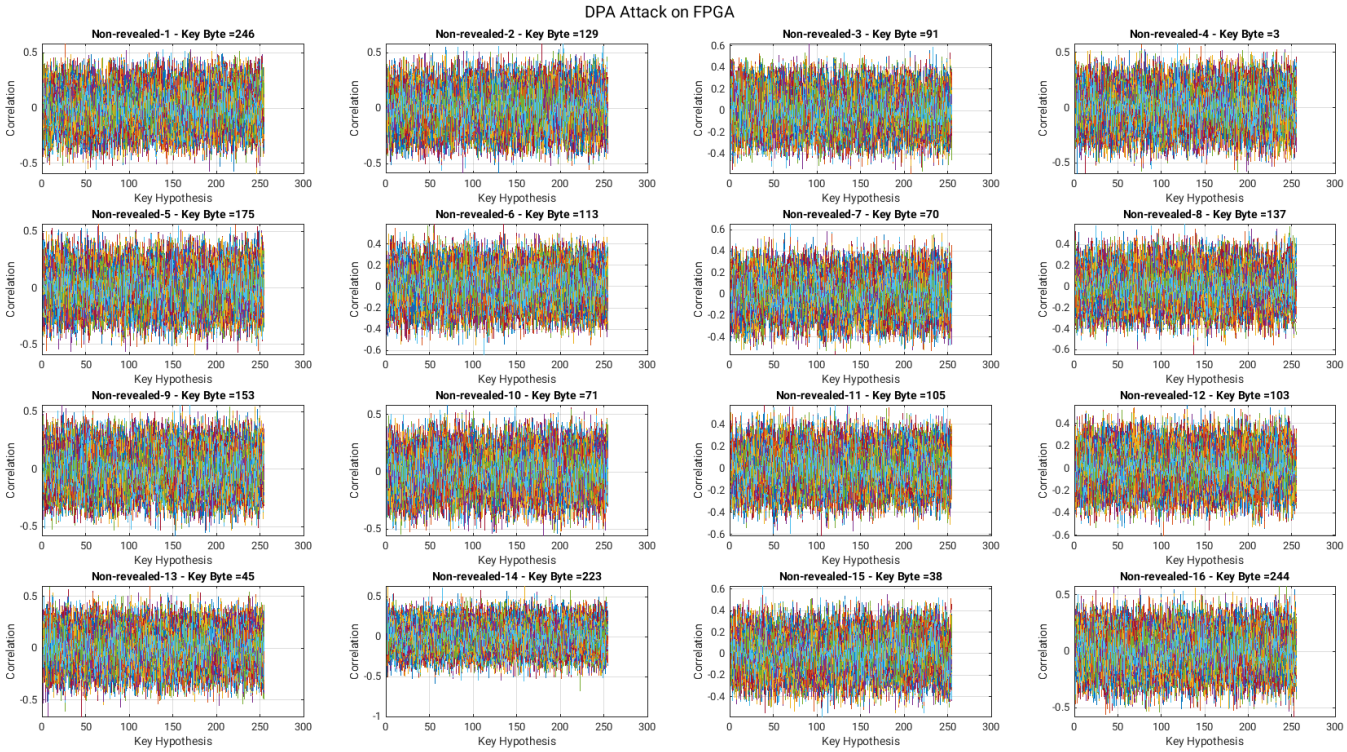

Figure A2. DPA attack on AES with the inclusion of our proposed countermeasure.

References

1. Imran, M.; Rashid, M.; Raza Jafri, A.; Najam-ul-Islam, M. ACryp-Proc: Flexible Asymmetric Crypto Processor for Point Multiplication. *IEEE Access* **2018**, *6*, 22778–22793. [CrossRef]
2. Coron, J.S. What is cryptography? *IEEE Secur. Priv.* **2006**, *4*, 70–73. [CrossRef]

3. Rashid, M.; Imran, M.; Jafri, A.R.; Al-Somani, T.F. Flexible Architectures for Cryptographic Algorithms—A Systematic Literature Review. *J. Circuits Syst. Comput.* **2019**, *28*, 1930003. [CrossRef]
4. Chandra, S.; Paira, S.; Alam, S.S.; Sanyal, G. A comparative survey of Symmetric and Asymmetric Key Cryptography. In Proceedings of the 2014 International Conference on Electronics, Communication and Computational Engineering (ICECCE), Hosur, India, 17–18 November 2014; pp. 83–93. [CrossRef]
5. Yilmaz, B.; ÖzdemİR, S. Performance comparison of cryptographic algorithms in internet of things. In Proceedings of the 2018 26th Signal Processing and Communications Applications Conference (SIU), Izmir, Turkey, 2–5 May 2018; pp. 1–4. [CrossRef]
6. Kumar, K.; Ramkumar, K.; Kaur, A.; Choudhary, S. A Survey on Hardware Implementation of Cryptographic Algorithms Using Field Programmable Gate Array. In Proceedings of the 2020 IEEE 9th International Conference on Communication Systems and Network Technologies (CSNT), Gwalior, India, 10–12 April 2020; pp. 189–194. [CrossRef]
7. Socha, P.; Brejník, J.; Bartik, M. Attacking AES implementations using correlation power analysis on ZYBO Zynq-7000 SoC board. In Proceedings of the 2018 7th Mediterranean Conference on Embedded Computing (MECO), Budva, Montenegro, 10–14 June 2018; pp. 1–4. [CrossRef]
8. Bhunia, S.; Tehranipoor, M. Chapter 10: Physical Attacks and Countermeasures. In *Hardware Security*; Bhunia, S., Tehranipoor, M., Eds.; Morgan Kaufmann: Cambridge, MA, USA, 2019; pp. 245–290. [CrossRef]
9. Mangard, S.; Oswald, E.; Popp, T. Revealing the Secrets of Smart Cards. In *Power Analysis Attacks*; Springer: Boston, MA, USA, 2007; p. XXIV-338. [CrossRef]
10. Pammu, A.A.; Chong, K.; Ho, W.; Gwee, B. Interceptive side channel attack on AES-128 wireless communications for IoT applications. In Proceedings of the 2016 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), Jeju, Korea, 25–28 October 2016; pp. 650–653. [CrossRef]
11. Zhang, X.; Chen, K.; Zhang, Y.; Gui, W.; Li, L. Correlation power analysis for AES encryption device. In Proceedings of the 4th National Conference on Electrical, Electronics and Computer Engineering, Xi'an, China, 12–13 December 2015; pp. 7–12. Available online: https://www.atlantis-press.com/article/25847048.pdf (accessed on 3 September 2021).
12. Zheng, Z.; Zou, X.; Liu, Z.; Chen, Y. Security Analysis and Optimization of AES S-Boxes Against CPA Attack in Wireless Sensor Network. In Proceedings of the 2007 International Conference on Wireless Communications, Networking and Mobile Computing, Shanghai, China, 21–25 September 2007; pp. 2608–2612. [CrossRef]
13. Benhadjyoussef, N.; Mestiri, H.; Machhout, M.; Tourki, R. Implementation of CPA analysis against AES design on FPGA. In Proceedings of the 2012 International Conference on Communications and Information Technology (ICCIT), Hammamet, Tunisia, 26–28 June 2012; pp. 124–128. [CrossRef]
14. Jaffe, J. A First-Order DPA Attack Against AES in Counter Mode with Unknown Initial Counter. In *Cryptographic Hardware and Embedded Systems—CHES 2007*; Paillier, P., Verbauwhede, I., Eds.; Springer: Berlin/Heidelberg, Germany, 2007; pp. 1–13. Available online: https://www.iacr.org/archive/ches2007/47270001/47270001.pdf (accessed on 19 September 2021).
15. Lu, J.; Pan, J.; den Hartog, J. Principles on the Security of AES against First and Second-Order Differential Power Analysis. In *Applied Cryptography and Network Security*; Zhou, J., Yung, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; pp. 168–185. Available online: https://link.springer.com/chapter/10.1007/978-3-642-13708-2_11 (accessed on 7 August 2021).
16. Kamoun, N.; Bossuet, L.; Ghazel, A. Experimental implementation of DPA attacks on AES design with Flash-based FPGA technology. In Proceedings of the 2009 6th International Multi-Conference on Systems, Signals and Devices, Djerba, Tunisia, 23–26 March 2009; pp. 1–4. [CrossRef]
17. Han, Y.; Zou, X.; Zhenglin, L.; Chen, Y. Efficient DPA Attacks on AES Hardware Implementations. *IJCNS* **2008**, *1*, 68–73. [CrossRef]
18. Bu, A.; Dai, W.; Lu, M.; Cai, H.; Shan, W. Correlation-Based Electromagnetic Analysis Attack Using Haar Wavelet Reconstruction with Low-Pass Filtering on an FPGA Implementaion of AES. In Proceedings of the 2018 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), New York, NY, USA, 1–3 August 2018; pp. 1897–1900. [CrossRef]
19. Wang, H.; Dubrova, E. Tandem Deep Learning Side-Channel Attack Against FPGA Implementation of AES. In Proceedings of the 2020 IEEE International Symposium on Smart Electronic Systems (iSES) (Formerly iNiS), Chennai, India, 14–16 December 2020; pp. 147–150. [CrossRef]
20. Khelil, F.; Hamdi, M.; Guilley, S.; Danger, J.L.; Selmane, N. Fault Analysis Attack on an FPGA AES Implementation. In Proceedings of the 2008 New Technologies, Mobility and Security, Tangier, Morocco, 5–7 November 2008; pp. 1–5. [CrossRef]
21. Kaps, J.P.; Velegalati, R. DPA Resistant AES on FPGA Using Partial DDL. In Proceedings of the 2010 18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines, Charlotte, NC, USA, 2–4 May 2010; pp. 273–280. [CrossRef]
22. Zhang, S.; Zhong, W. A New Type of Countermeasure against DPA in Multi-Sbox of Block Cipher. *Wirel. Commun. Mob. Comput.* **2018**, *2018*, 5945312. [CrossRef]
23. Babu, G.D.; Anandakumar, N.N.; Muralidharan, D. Countermeasures Against DPA Attacks on FPGA Implementation of AES. *J. Artif. Intell.* **2012**, *5*, 186–192. [CrossRef]
24. Attaran, A.; Mirhassani, M. An embedded low-overhead PLL-based countermeasure against DPA side channel attack. In Proceedings of the 2015 International Symposium on Signals, Circuits and Systems (ISSCS), Iasi, Romania, 9–10 July 2015; pp. 1–4. [CrossRef]

25. Mohana, P.; Srinivasan, R.; Kanchana Bhaaskaran, V.S. An energy recovery logic level countermeasure for power analysis attacks on AES. In Proceedings of the International Conference on Smart Structures And Systems (ICSSS'13), Chennai, India, 28–29 March 2013; pp. 164–170. [CrossRef]
26. Lo, O.; Buchanan, W.J.; Carson, D. Power analysis attacks on the AES-128 S-box using differential power analysis (DPA) and correlation power analysis (CPA). *J. Cyber Secur. Technol.* **2017**, *1*, 88–107. [CrossRef]
27. Kavi priya, S.; Arunmani, G. Advanced Logic Level Design Methodology for a Secure DPA Resistant FPGA. In Proceedings of the 2019 IEEE International Conference on Intelligent Techniques in Control, Optimization and Signal Processing (INCOS), Tamilnadu, India, 11–13 April 2019; pp. 1–3. [CrossRef]
28. Yu, W.; Köse, S. A Voltage Regulator-Assisted Lightweight AES Implementation Against DPA Attacks. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2016**, *63*, 1152–1163. [CrossRef]
29. Tiri, K.; Verbauwhede, I. Securing Encryption Algorithms against DPA at the Logic Level: Next Generation Smart Card Technology. In *Cryptographic Hardware and Embedded Systems—CHES 2003*; Walter, C.D., Koç, Ç.K., Paar, C., Eds.; Springer: Berlin/Heidelberg, Germany, 2003; pp. 125–136. Available online: https://link.springer.com/chapter/10.1007/978-3-540-45238-6_11 (accessed on 10 July 2021).
30. Discretix Technologies Ltd. Known Attacks against Smart Cards. 2021. Available online: http://www.infosecwriters.com/text_resources/pdf/Known_Attacks_Against_Smartcards.pdf (accessed on 19 October 2021).
31. Eduard Kovacs. Researchers Disclose New Side-Channel Attacks Affecting All AMD CPUs. 2021. Available online: https://www.securityweek.com/researchers-disclose-new-side-channel-attacks-affecting-all-amd-cpus?quicktabs_1=0 (accessed on 19 October 2021).
32. Zybo Z7. AES (Rijndael) IP Core. 2018. Available online: https://opencores.org/projects/aes_core (accessed on 3 May 2021).
33. Daemen, J.; Rijmen, V. *The Design of Rijndael: AES—The Advanced Encryption Standard*; Springer: Berlin/Heidelberg, Germany, 2002; p. XVII-238. Available online: https://www.springer.com/gp/book/9783540425809 (accessed on 24 July 2021).
34. Pitchaiah, M.; Daniel, D. Implementation of Advanced Encryption Standard Algorithm. *Int. J. Sci. Eng. Res.* **2012**, *3*, 14–17. Available online: https://www.ijser.org/researchpaper/Implementation-of-Advanced-Encryption-Standard-Algorithm.pdf (accessed on 21 July 2021).
35. Miškovský, V.; Kubátová, H.; Novotný, M. Influence of fault-tolerant design methods on differential power analysis resistance of AES cipher: Methodics and challenges. In Proceedings of the 2016 5th Mediterranean Conference on Embedded Computing (MECO), Bar, Montenegro, 12–16 June 2016; pp. 14–17. [CrossRef]
36. Ordu, L.; Ors, B. Power Analysis Resistant Hardware Implementations of AES. In Proceedings of the 2007 14th IEEE International Conference on Electronics, Circuits and Systems, Marrakech, Morocco, 11–14 December 2007; pp. 1408–1411. [CrossRef]
37. Ming, J.; Zhou, Y.; Li, H.; Zhang, Q. A secure and highly efficient first-order masking scheme for AES linear operations. *Cybersecurity* **2021**, *4*, 1408–1411. [CrossRef]
38. Trichina, E.; De Seta, D.; Germani, L. Simplified Adaptive Multiplicative Masking for AES. In *Cryptographic Hardware and Embedded Systems—CHES 2002*; Kaliski, B.S., Koç, Ç.K., Paar, C., Eds.; Springer: Berlin/Heidelberg, Germany, 2003; pp. 187–197. Available online: https://link.springer.com/content/pdf/10.1007%2F3-540-36400-5_15.pdf (accessed on 11 May 2021).
39. De Meyer, L.; Reparaz, O.; Bilgin, B. Multiplicative Masking for AES in Hardware. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2018**, *2018*, 431–468. [CrossRef]
40. O'Donnell, R. Analysis of Boolean Functions. 2014. Available online: https://www.cs.tau.ac.il/~amnon/Classes/2016-PRG/Analysis-Of-Boolean-Functions.pdf (accessed on 18 October 2021).
41. Zybo Z7. Zybo Z7: Zynq-7000 ARM/FPGA SoC Development Board. 2020. Available online: https://digilent.com/reference/_media/reference/programmable-logic/zybo-z7/zybo_z7_sch-public.pdf (accessed on 19 August 2021).
42. Mulani, A.O.; Mane, P.B. High-Speed Area-Efficient Implementation of AES Algorithm on Reconfigurable Platform, 2019. In *Computer and Network Security*; IntechOpen: London, UK, 2019. Available online: https://www.intechopen.com/chapters/67728 (accessed on 20 July 2020). [CrossRef]
43. Kamoun, N.; Bossuet, L.; Ghazel, A. SRAM-FPGA implementation of masked S-Box based DPA countermeasure for AES. In Proceedings of the 2008 3rd International Design and Test Workshop, Monastir, Tunisia, 20–22 December 2008; pp. 74–77. [CrossRef]