computers

Article

Comparing the Cost of Protecting Selected Lightweight Block Ciphers against Differential Power Analysis in Low-Cost FPGAs †

William Diehl *, Abubakr Abdulgadir, Jens-Peter Kaps and Kris Gaj

Department of Electrical and Computer Engineering, George Mason University, Fairfax, VA 22030, USA; aabdulga@gmu.edu (A.A.); jkaps@gmu.edu (J.-P.K.); kgaj@gmu.edu (K.G.)

* Correspondence: wdiehl@gmu.edu; Tel.: +1-703-993-1561

† This paper is an extended version of our paper published in the International Conference on Field-Programmable Technologies (FPT 2017).

Received: 1 March 2018; Accepted: 9 April 2018; Published: 23 April 2018

Abstract: Lightweight block ciphers are an important topic in the Internet of Things (IoT) since they provide moderate security while requiring fewer resources than the Advanced Encryption Standard (AES). Ongoing cryptographic contests and standardization efforts evaluate lightweight block ciphers on their resistance to power analysis side channel attack (SCA), and the ability to apply countermeasures. While some ciphers have been individually evaluated, a large-scale comparison of resistance to side channel attack and the formulation of absolute and relative costs of implementing countermeasures is difficult, since researchers typically use varied architectures, optimization strategies, technologies, and evaluation techniques. In this research, we leverage the Test Vector Leakage Assessment (TVLA) methodology and the FOBOS SCA framework to compare FPGA implementations of AES, SIMON, SPECK, PRESENT, LED, and TWINE, using a choice of architecture targeted to optimize throughput-to-area (TP/A) ratio and suitable for introducing countermeasures to Differential Power Analysis (DPA). We then apply an equivalent level of protection to the above ciphers using 3-share threshold implementations (TI) and verify the improved resistance to DPA. We find that SIMON has the highest absolute TP/A ratio of protected versions, as well as the lowest relative cost of protection in terms of TP/A ratio. Additionally, PRESENT uses the least energy per bit (E/bit) of all protected implementations, while AES has the lowest relative cost of protection in terms of increased E/bit.

Keywords: block cipher; encryption; field programmable gate array; side channel attack; countermeasure; lightweight; TVLA; *t*-test; FOBOS

1. Introduction

Cryptographic services, such as confidentiality, integrity, and authentication, are required in many of the billions of small devices constituting the "Internet of Things" (IoT). Such devices could include cyber-physical sensors and actuators, wireless sensors, biometric devices, driverless cars, and so forth. These devices are often heavily constrained by size, weight, and power (SWaP). They are also often located apart from secure data facilities and are thus more vulnerable to physical compromise.

Existing standards for cryptographic block ciphers such as DES (Data Encryption Standard), 3DES (Triple-DES), and AES (Advanced Encryption Standard), are primarily intended for information-intensive applications and are optimized for throughput and for use in high-speed communication protocols. However, with the migration of applications away from mainframe servers and personal computers to embedded and wireless remote devices in the IoT, there is a growing emphasis on providing solutions that are less power and resource-intensive at the cost of somewhat relaxed security margins. Many such solutions can be realized using lightweight cryptographic algorithms.

Adversaries can attempt to recover sensitive variables (such as a secret key) through "side-channel attacks" (SCA). While cryptanalytic attacks on well-constructed cryptographic algorithms are generally infeasible using current computing capabilities, real ciphers must still exist on physical devices and are vulnerable to information leakage. Differential Power Analysis (DPA) is one SCA technique that can be used to target cryptographic implementations (including lightweight block ciphers) to recover sensitive information [1–3].

Several current cryptographic contests and standards development projects have targeted improvements in lightweight cryptography. One example is the Competition for Authenticated Encryption: Security, Applicability, and Robustness (CAESAR), with the expected selection of a final portfolio in 2018 [4]. Beginning with Round 3, the CAESAR committee specified use cases for which candidates would be optimized and ultimately selected [5]. One of these use cases is lightweight applications (resource constrained environments), for which desired characteristics include "natural ability to protect against side-channel attacks" [5].

A second example is the National Institute of Standards and Technology's (NIST) Lightweight Cryptography Project, which will develop new recommendations using an open call for proposals to standardize and evaluate algorithms based on several characteristics, including side-channel resistance [6,7].

In this research, we investigate encryption-only variants of six secret-key block ciphers: AES, SIMON, SPECK, PRESENT, LED, and TWINE. The block cipher implementations in this research are either designed by the authors, using register-transfer level (RTL) methodology or adopted from publicly-available source codes as annotated below. A brief introduction to these six block ciphers is as follows:

1. AES

AES (Advanced Encryption Standard) is the U.S. federal standard and a de-facto worldwide standard for symmetric block ciphers as defined in Reference [8]. AES-128 uses a 128-bit key, encrypts (or decrypts) 128-bit blocks of plaintext (or ciphertext), and consists of 10 rounds. There are four transformations which occur on a state defined as a 4×4 matrix of bytes.

The SubBytes transformation introduces non-linearity through a one-to-one byte substitution. SubBytes is often rendered as 16 8×8 S-Boxes which can be implemented in look-up tables. The ShiftRows transformation performs a permutation on the state word, where the ith row is rotated left by i bytes for rows 0 through 3. The MixColumns transformation is equivalent to the multiplication of each column of the state by a 4×4 matrix of constants in GF(2^8) (Galois Field). The MixColumns transformation is skipped in the final round. In the AddRoundKey transformation, a 128-bit round key is added to the state by a simple bitwise XOR operation.

2. SIMON

SIMON $2n/nm$ is defined in Reference [9], where n is the n-bit word size and m is the number of keywords. SIMON 96/96 is implemented in this study, which consists of a 96-bit data block (where operations are conducted on a 48-bit status word) and a 96-bit key.

SIMON uses a Feistel structure with simple round features designed for lightweight operations. SIMON 96/96 has 52 rounds. Each round consists of bitwise ANDs, left rotations by j bits, and XORs. There are no S-Boxes in SIMON; non-linearity is achieved by the bitwise AND.

3. SPECK

SPECK is defined in Reference [9]. It is also a Feistel structure and is similar to SIMON, except that the algorithm uses both left and right rotations, and uses addition modulo 2^n in place of AND. Also, there are only 28 rounds in SPECK 96/96.

4. PRESENT

PRESENT is a 64-bit block cipher with an 80-bit key and is defined in Reference [10]. PRESENT-80 requires 31 rounds, each of which consists of the addition of the round key, 16 4×4 S-Boxes, and a bitwise permutation layer.

5. LED

LED is a 64-bit block cipher with an 80-bit key and is defined in Reference [11]. In LED, the 64-bit plaintext is split into 16 nibbles. LED is similar to AES in that each round consists of four transformations: SubCells, ShiftRows, MixColumnsSerial, and AddConstant. SubCells applies a four-bit S-Box to each of 16 nibbles, and MixColumnsSerial computes a 4×4 matrix multiplication in GF(2^4). There are 48 rounds. Every four rounds are grouped together to form a step, in which the key material is supplied.

6. TWINE

TWINE, defined in Reference [12], uses a generalized Feistel structure (GFS) consisting of 16 nibbles in one 64-bit block. It uses eight four-bit S-Boxes, eight bitwise permutations (computed pair-wise in four-bit nibbles), and XORs. TWINE uses an 80-bit key and requires 36 rounds to complete a block encryption.

All of the above ciphers, except for LED, require some form of key scheduling, where operations are performed on the initial secret key to increase the cipher's security. A specific round key is produced for each round. The round keys can either be produced outside of the cipher and downloaded at run-time, produced by the cipher at initialization and stored in memory, or produced during each round, or "on-the-fly."

The major characteristics of block cipher variants implemented in this research are shown in Table 1. The internal structure of a regular cipher round for all investigated ciphers is shown in Figures 1 and 2. The reader is referred to References [8–12] for the detailed specifications of all implemented ciphers, including any additional information on pre-processing (computations performed before the first round), post-processing (computations executed after the last round), special rounds (for example, a reduced-functionality last round), and key scheduling (that is, calculation of round keys).

Five of these ciphers are used as primitives for the authenticated ciphers that were evaluated during CAESAR Round 3 (which concluded in March 2018), including CLOC-AES, CLOC-TWINE, AES-JAMBU, SIMON-JAMBU, SILC-AES, SILC-PRESENT, and SILC-LED [13,14].

Table 1. The principal features of block cipher variants implemented in this research.

Cipher	Block Size	Key Size	Rounds	Type	Basic Operations
AES-128	128	128	10	SPN [1]	SubBytes, ShiftRows, MixColumns, AddRoundKey
SIMON 96/96	96	96	52	Feistel	AND, rotation, XOR
SPECK 96/96	96	96	28	Feistel, ARX [2]	ADD, rotation, XOR
PRESENT-80	64	80	31	SPN	S-Box, permutation, XOR
LED-80	64	80	48	SPN	AddConstant, SubCells, ShiftRows, MixColumnsSerial, XOR
TWINE-80	64	80	36	SPN, Feistel	S-Box, permutation, XOR

[1] SPN is "Substitution Permutation Network"; [2] ARX is "Addition, Rotation, XOR".

In this work, we support the goals of the CAESAR competition and NIST standardization effort by evaluating the resistance of these six secret-key block ciphers to DPA using the Test Vector Leakage Assessment (TVLA) methodology [15–17], and the Flexible Open-source workBench fOr Side-channel analysis (FOBOS) [18]. Resistance to DPA is evaluated on the low-cost Spartan-3E Field Programmable Gate Array (FPGA), incorporated in the Digilent Nexys-2 Starter Board.

We then apply an equivalent level of protection against 1st order DPA for all six ciphers using threshold implementations (TI) [19] and the verify improved resistance to DPA using FOBOS. TI are an algorithmic countermeasure against power-analysis side-channel attack. TI are based on secret sharing and multi-party communications, where the communications of a single party cannot be exploited to learn the secret content [20,21]. They improve upon traditional Boolean masking in that they provide security in the presence of glitches. Although Boolean masking provides mathematically-secure protection against DPA, it can fail in CMOS (Complementary Metal Oxide Semiconductor) technology, since the power change that occurs in a CMOS gate during a transition due to a glitch is relatively large compared to the normal operation of a device. Measuring the toggle rate of CMOS glitches has been used to successfully attack a masked version of AES [22].

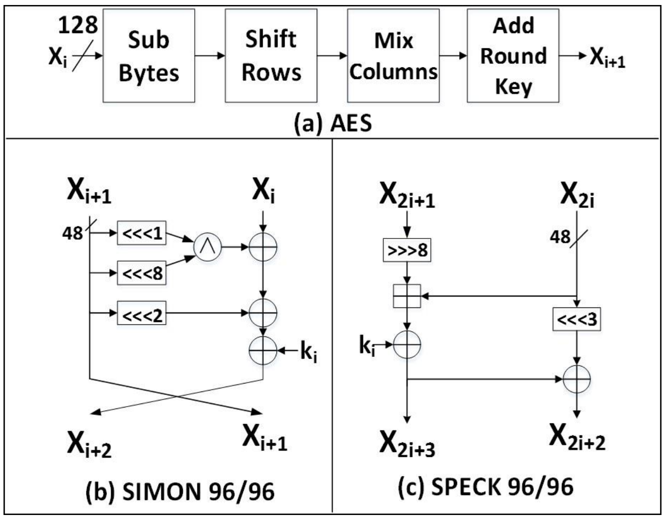


Figure 1. (**a**) AES, (**b**) SIMON 96/96, and (**c**) SPECK 96/96. Notation: >>>n, <<<n are rotations by n positions to the right or left, respectively; $\wedge$ is bitwise AND; $\oplus$ is bitwise XOR; $\boxplus$ is addition modulo 2^{48}.

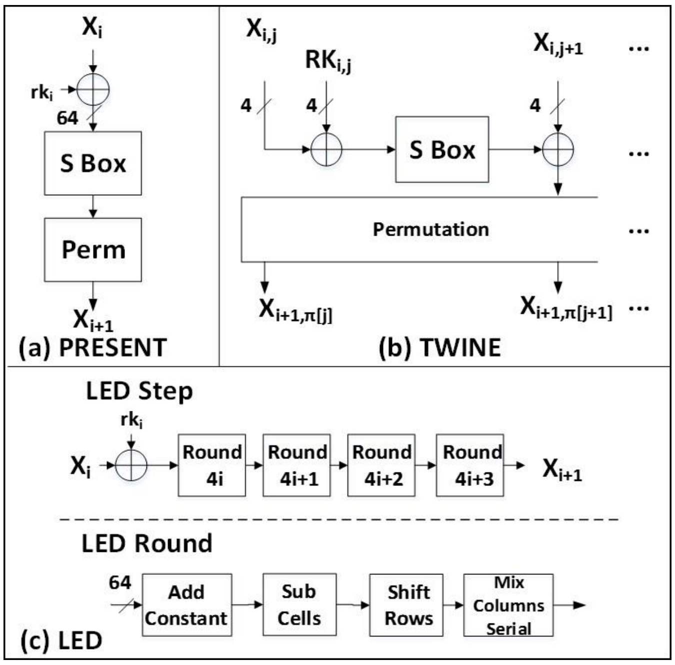


Figure 2. (**a**) PRESENT, (**b**) TWINE, and (**c**) LED. In (**b**) TWINE, only 1 of 8 pairwise nibble operations are shown (that is, $j = 0, 1, \ldots, 7$).

A threshold implementation must have the following three properties, outlined in Reference [19], to be provably secure against power analysis in the presence of glitches:

1. Non-completeness. Every function is independent of at least one share of each of the input variables. Defined formally, if $z = F(x, y)$, and x and y are divided into n shares, then

$$z_1 = f_1(x_2, x_3, \ldots, x_n, y_2, y_3, \ldots, y_n), \tag{1}$$

$$z_2 = f_2(x_1, x_3, \ldots, x_n, y_1, y_3, \ldots, y_n), \tag{2}$$

$$\vdots$$

$$z_n = f_n(x_1, x_2, \ldots, x_{n-1}, y_1, y_2, \ldots, y_{n-1}). \tag{3}$$

   In other words, If z_i does not depend on x_i and y_i, it cannot leak information about x_i or y_i.
2. Correctness. The sum of the output shares gives the desired output. Formally

$$z = \bigoplus_{i=1}^{n} z_i = N(x, y). \tag{4}$$

3. Uniformity. A realization of sharing $z = F(x, y)$ is uniform if, for all distributions of the inputs x and y, the output distribution preserves the input distribution. In other words, if the input function is a permutation, the output function should also be a permutation.

A non-linear function of algebraic degree 2, such as $z = xy$ (for example, a two-input `AND` gate), can be shared using three TI shares since $d + 1$ shares are required to share a function of degree d. However, as discussed in References [23,24], achieving TI which are both non-complete and uniform is not trivial. This can be realized by supplying fresh random bits (for example, "resharing" or "remasking" randomness), however, this requires the resourcing of sufficient randomness, which must either be imported into the device or generated internally at run-time. Thus, the decision to use 3-share TI which require an increased number of random bits, or 4-share TI with more required resources but no additional randomness, is an engineering design tradeoff.

After applying TI and verifying the improved DPA resistance, we evaluate the protected implementations in terms of the area, throughput, and throughput-to-area (TP/A) ratio on two FPGAs—the Xilinx Virtex-7 and the Spartan-3E—and measure the actual power and energy usage during cipher operations on the Spartan-3E FPGA. Finally, we rank the unprotected and protected cipher implementations based on the absolute and relative costs of protection against 1st order DPA, where the relative costs are computed as ratios of protected versus unprotected implementations of the above metrics.

AES, PRESENT, LED, SIMON, and SPECK have been previously protected against differential power analysis using threshold implementations and the subsequent resistance has been evaluated in ASIC (Application Specific Integrated Circuit) or FPGA [23–29]. However, these evaluations are typically made by individual research groups, which implement only the targeted cipher and do not conduct direct measurements of other ciphers.

In contrast, the authors of Reference [30] perform a direct comparison of several of the above block ciphers, including AES, PRESENT, SIMON, and SPECK, and determine the area overhead for an equivalent level of DPA resistance. While similar in methodology, the study in Reference [30] references acceptable results of 3-share TI protection schemes conducted in other studies, whereas we verify the improved DPA resistance for our exact choice of implementation.

As discussed in Reference [30], it is more desirable to perform a direct comparison of all ciphers, that is, implemented by the same hardware designers and evaluated on the same test bench, to eliminate differences in implementer style or choice of hardware. Our primary contribution is to facilitate a relevant comparison by implementing six block ciphers, protecting each cipher with an identical level of protection to DPA, evaluating the unprotected and protected versions of all ciphers in an identical analysis suite, and comparing the ciphers in terms of area, throughput, throughput-to-area (TP/A) ratio, power, and energy. To our knowledge, we present the first documented, verified, and benchmarked results of a 3-share TI-protected implementation of TWINE.

Additionally, the protected versions of ciphers in this research employ anti-optimization constraints, in order to prevent synthesis tools from including optimizations that could nullify algorithmic countermeasures against DPA. These anti-optimization tools exert a cost in terms of the increased area and reduced performance. Designers of protected ciphers in [23–27] use anti-optimization techniques, but do not discuss the resulting cost. By examining our six protected cipher implementations on two FPGAs, we are able to quantify and characterize the expected degradations in the area, throughput, and TP/A ratio on both target devices.

Finally, the TVLA leakage detection methodology discussed in References [15–17] is designed to provide a less-comprehensive, but far-less time-consuming evaluation of side channel leakage. We validate this methodology by providing a large-scale comparison of multiple ciphers which would be enormously difficult using traditional methods of DPA evaluation.

2. Results

2.1. Implementations of Ciphers with 3-Share TI Protection against DPA

In order to protect cipher implementations against 1st order DPA and simultaneously provide a fair basis for comparison, we implement a maximum of a 3-share threshold implementation (TI) (that is, no greater than 3 shares) on each of the subject ciphers. The specific techniques used for 3-share TI implementations are discussed below.

2.1.1. AES

We start with an implementation of an S-Box using combinational logic, as described in References [31,32]. As the AES S-Box has a maximum algebraic degree of 7, a direct sharing would require a minimum of 8 shares. An 8-share TI is not feasible, even if such a sharing could be discovered that meets all the TI properties (none have been discovered to date). However, using the method of Tower Fields, where inversions in GF(2^8) are represented as operations in GF(2^4), which are, in turn, represented in GF(2^2), field multiplications and inversions in low-degree non-linear representations become feasible. In fact, each inversion in GF(2^8) requires three multiplications in GF(2^4), and one inversion in GF(2^4). In turn, each multiplication in GF(2^4) requires one multiplication and one multiplication and scaling in GF(2^2), and each inversion requires three multiplications and one inversion in GF(2^2). Using a preliminary conversion to a normal basis as outlined in Reference [31], inversions in GF(2^2) (like squares) are linear transformations. We choose not to produce a full-width, basic iterative architecture TI-protected version of AES for the following reasons:

1. Each 8-bit S-Box using Tower Fields requires nine GF(2^2) regular multiplications and three GF(2^2) scaled multiplications, which is enormously costly when implementing multiple S-Boxes;
2. The Tower Fields approach results in multiple cascaded non-linear sharings which could cause long glitch-dependent circuit paths.

Non-linear multiplications do not satisfy TI Property 3 (Uniformity) in that they are not permutations. Therefore, they require mask refreshing during or after every TI-shared calculation. The total fresh randomness required either increases the I/O requirements, or increases the area, if generated on-chip. Therefore, it is better to distribute this requirement over multiple clock cycles.

Therefore, we leverage approaches in References [23,24] to develop a hybrid 8-bit and 32-bit datapath in a pipelined approach. We follow the method of Reference [23] and instantiate only one complete 8-bit S-Box, which is separated into five stages (shown in Figure 3). However, we adopt a method described in Reference [24] to employ a hybrid 2-/3-share TI approach, where linear calculations (such as round key addition, column multiplications, basis conversions, affine transformations, and so forth) are conducted on only two shares to save resources.

Our resulting protected design has a 5-stage pipeline, where one S-Box operation commences with every clock cycle. A 128-bit round completes every 16 cycles, with one additional cycle occupied by a programmed stall. Therefore, a 128-bit block encryption executes in 17 (cycles/round) $\times$ 10 ($AES - 128$ rounds) $+$ 5 (priming cycles) $=$ 175 clock cycles. The design uses 16 bits of fresh randomness for resharing from two to three shares, and two fresh remasking bits per GF(2^2) multiplier and multiplier-scalar instance, resulting in a total of 40 random bits required for each S-Box. The three shares are recombined into two shares at the end of the non-linear chain to reduce the resources required for affine transformation, change of basis, 32-bit column multiplications, and round key addition.

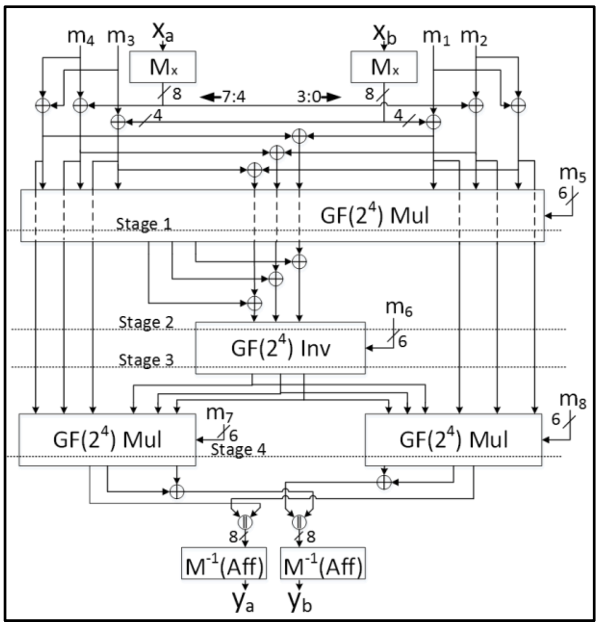


Figure 3. The 2-/3-share hybrid Threshold Implementation (TI)-protected 5-stage pipelined 8-bit AES S-Box. The dashed lines indicate that the signals do not affect the blocks in which they occur. Every signal that crosses a dotted line is registered (registers not shown).

2.1.2. SIMON

We adopt the SIMON 96/96 full-width implementation with a basic iterative architecture available at Reference [14] and modify it as necessary for our test methodology. A 3-share threshold implementation (TI) of SIMON is easily achieved using the methodology described in Reference [19,27]. SIMON uses only a single 2-input 48-bit AND to achieve non-linearity. Therefore, a 3-share TI of this quadratic equation is achieved without requiring any cascading or composite functions.

$$\begin{aligned} x'_1 = y_2 \bigoplus (x_2 \lll 2) \bigoplus (x_2 \lll 1) \wedge (x_2 \lll 8) \bigoplus (x_2 \lll 1) \wedge \\ (x_3 \lll 8) \bigoplus (x_3 \lll 1) \wedge (x_2 \lll 8) \bigoplus k_2, \end{aligned} \tag{5}$$

$$\begin{aligned} x'_2 = y_3 \bigoplus (x_3 \lll 2) \bigoplus (x_3 \lll 1) \wedge (x_3 \lll 8) \bigoplus (x_3 \lll 1) \\ \wedge (x_1 \lll 8) \bigoplus (x_1 \lll 1) \wedge (x_3 \lll 8) \bigoplus k_3, \end{aligned} \tag{6}$$

$$\begin{aligned} x'_3 = y_1 \bigoplus (x_1 \lll 2) \bigoplus (x_1 \lll 1) \wedge (x_1 \lll 8) \bigoplus (x_1 \lll 1) \\ \wedge (x_2 \lll 8) \bigoplus (x_2 \lll 1) \wedge (x_1 \lll 8) \bigoplus k_1, \end{aligned} \tag{7}$$

$$y_1 = x_1,\ y_2 = x_2,\ y_3 = x_3. \tag{8}$$

The TI properties 1 (non-completeness) and 2 (correctness) are satisfied, as each share lacks at least one of the component shares in its calculation. The non-linear nature of the round function suggests that the shares are not a permutation, and therefore, do not automatically satisfy Property 3

(Uniformity). However, uniformity is satisfied in this case by considering the key shares, included in each TI-share calculation, as a source of randomness [27]. Therefore, no mask refreshing is required in SIMON 3-share TI, which leads to a very efficient TI-protected implementation.

2.1.3. SPECK

SPECK is an ARX (Addition, Rotation, XOR) cipher with non-linearity provided by addition modulo 2^{48}. Masking additions against DPA are possible using formulas such as $x = x' + r_x \ mod \ 2^n$, where $x\prime$ is a masked variable and r_x is an arithmetic mask. However, SPECK operations contain components that require Boolean masking (for example, rotations and XORs) in addition to arithmetic masking. Techniques that employ both Boolean and arithmetic masking are abundant in the literature and have been investigated over the last 15 years in order to provide masking for AES competition candidates with modulo additions (for example, RC6, TWOFISH, MARS, IDEA), as well as eSTREAM candidates and the SHA-1 (Secure Hash Algorithm) hash function. Such techniques require Boolean-to-arithmetic and arithmetic-to-Boolean masking conversions and are discussed in References [33–37]. More recently, the authors in Reference [38] proposed new conversion algorithms based on Reference [33] that can be made secure for any order of masking. To convert masks of a size of k bits securely against attacks of order n, the proposed algorithms have a time complexity of $O\left(n2^k\right)$.

While it is possible to apply the above conversion techniques to SPECK, the resulting protected design is likely to be resource-intensive and highly complex. Accordingly, we have chosen an alternative approach using only Boolean masking. The authors in Reference [39] describe a technique to achieve a 3-share threshold implementation (TI) for a 32-bit adder using the Kogge-Stone adder. First published in Reference [40], the Kogge-Stone adder produces recursive carry "generate" and "propagate" trees. In a preprocessing stage, the ith generation and propagation bits are computed as $g_i = a_i \wedge b_i$ and $p_i = a_i \bigoplus b_i$. In subsequent stages, the generation bits are computed as $g_{i,j} = g_i \bigoplus (g_j \wedge p_i)$, and propagation bits are computed as $p_{i,j} = p_i \wedge p_j$. Finally, the summation bits s_0 to s_k are produced using cascaded XOR gates of the full-adder as $s_i = a_i \bigoplus b_i \bigoplus c_i$, where $c_{i\in\{2\ldots n\}} = g_{i-1:0}$, $c_1 = g_0$, and $c_0 = 0$. The total number of stages required is $n = [\log_2 k] + 1$, where k is the number of adder bits (for example, 48 bits required for SPECK). Therefore, $n = 7$ for SPECK, where the first stage (Stage 0) is a preprocessing stage.

In the Kogge-Stone adder, the largest `AND` gate is two-input. Therefore, the maximum degree d of non-linearity is 2 and the adder can be shared using $d + 1 = 3$ shares. We adopt the TI method as outlined in Reference [39], which requires $k = 48$ bits (denoted m_i) for mask refreshing in the preprocessing stage in order to satisfy the TI uniformity property. However, we also provide mask refresh bits for stages 1 through 6 in order to satisfy the TI uniformity property. The number of mask refresh bits required decreases logarithmically for each stage. The total number of bits required is $k + \sum_{i=1}^{[\log_2 k]} k - 2^{i-1}$, or 273 bits for one complete 48-bit addition.

It is infeasible to provide 273 random bits in one clock cycle. Additionally, the performance of seven levels of cascaded non-linear TI operations in one clock cycle risks leaking information through glitch dependencies. Therefore, we adopt a multi-cycle architecture executing in eight clock cycles per round, in which the carry chain results are registered after every non-linear computation on the shared components. Two hundred and seventy-three bits amortized over eight clock cycles results in a requirement of 34 bits per clock cycle—a large requirement, but at least feasible.

The Kogge-Stone modulo 2^{48} adder, as implemented in SPECK, is shown in Figure 4. Registers are placed at the output of each stage (including preprocessing Stage 0) and at the end of the round.

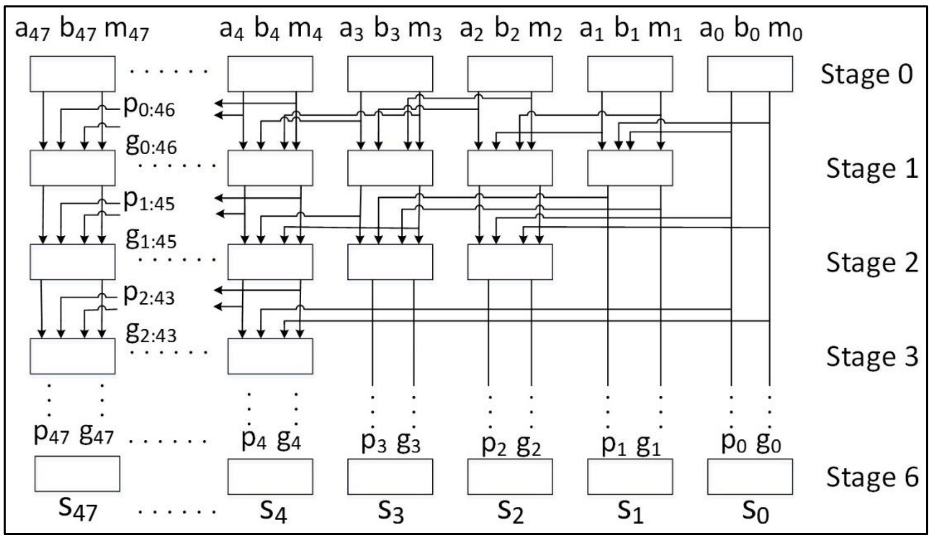


Figure 4. A 48-bit Kogge-Stone Adder (KSA) used in the 3-share TI protected version of SPECK. All bus widths are single bit unless indicated.

2.1.4. PRESENT

A 3-share TI-protected version of PRESENT is efficiently achieved using the strategies described in References [25,26]. PRESENT uses a 4-bit S-Box of cubic degree. Therefore, a direct sharing using a minimum of four shares is possible. However, a 3-share version is achieved by defining composite functions F and G such that $F(x)\cdot G(x) = S(x)$, and where F and G are quadratic. Such a composition is $S(x) = A(G(G(Bx \bigoplus c) \bigoplus d))$, where

$$A = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix}, \tag{9}$$

$$B = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}, \tag{10}$$

$c = 1$, and $d = 5$. We utilize the innovation described in Reference [26] where one reusable function G is defined for all 3-share S-Box computations. However, in keeping with our strategy of full-width implementations using a basic iterative architecture, we instantiate a full six instances of the function G, vice the single instance described in Reference [26]. Using the following strategy, for shares a, b, and:

$$G_{s\in\{a,b,c\}}(x_1, y_1, z_1, w_1, x_2, y_2, z_2, w_2) = g_{s\in\{a,b,c\}} \tag{11}$$

$$g_{s3=}x_1 \bigoplus y_1 \wedge z_1 \bigoplus y_1 \wedge z_2 \bigoplus y_2 \wedge z_1 \bigoplus y_1 \wedge w_1 \bigoplus y_1 \wedge w_2 \bigoplus y_2 \wedge w_1, \tag{12}$$

$$g_{s2} = w_1 \bigoplus x_1 \wedge y_1 \bigoplus x_1 \wedge y_2 \bigoplus x_2 y_1, \tag{13}$$

$$g_{s1} = y_1, \text{ and} \tag{14}$$

$$g_{s0} = z_1 \bigoplus y_1 \wedge w_1 \bigoplus y_1 \wedge w_2 \bigoplus y_2 \wedge w_1 \tag{15}$$

As discussed in Reference [26], the uniformity property is satisfied for the shared functions G, since the output is a permutation on the input. Therefore, no additional randomness is required.

2.1.5. LED

A 3-share TI implementation of LED is achieved using the methodology described above for PRESENT since LED uses the PRESENT S-Box. The only additional consideration for LED is that the PRESENT permutation is essentially "no-cost" in hardware, whereas linear transformations conducted in LED (for example, MixColumnsSerial) are costly. Therefore, there is a tradeoff to consider in using a hybrid 2-/3-share structure, as documented, for AES in order to reduce the number of matrix multiplier instances. However, this would require the addition of random bits for resharing, whereas LED would otherwise require no random bits. Therefore, we maintain a strict 3-share TI-protected LED and accept the cost of instantiating three matrix multipliers for our full-width basic iterative architecture.

2.1.6. TWINE

TWINE uses a 4-bit S-Box based on a cubic function (that is, $d = 3$) and is designed using the same strategy as the AES S-Box, that is, a field inversion followed by an affine transformation. The S-Box is defined as $S(x) = A(x \bigoplus b)^{-1} mod\ p$, where a^{-1} denotes the inverse of a in GF(2^4), where the zero element is mapped to itself. The irreducible polynomial is $z^4 + z + 1,\ b = 1$, and

$$A = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \tag{16}$$

To achieve a 3-share TI, we employ a strategy previously used for AES in Reference [41]. According to Fermat's Little Theorem (FLT), $a^p \equiv a\ mod\ p$, and $a^{p-2} \equiv a^{-1}\ mod\ p$. In this case, we can compute $x^{14} \equiv x^{-1}$ in GF(2^4). This conveniently decomposes into two cascaded multipliers of quadratic order, which enables our 3-share TI. The FLT inverter also uses three squares per share, but the squares are nearly free (for example, two XOR gates) and are linear operators.

In contrast to PRESENT and LED, the cascaded multipliers on GF(2^4) are not permutations—they do not satisfy the TI uniformity property. Refreshed masking is required at each of the two levels to ensure this property. Uniformity is achieved with one random bit per 4-bit multiplier, for a total of two bits per S-Box and 16 bits per clock cycle in a basic iterative architecture. Monte Carlo simulations demonstrated that the single-bit mask refreshing achieves an average probability of 0.499 of an output '1' ($P_{i\in\{0,1,2,3\}} = 1$) for the four output bits (given equally likely '0' or '1' at input bits), with a minimum single bit probability of 0.498. The 3-share FLT inverter as applied in TWINE is shown in Figure 5.

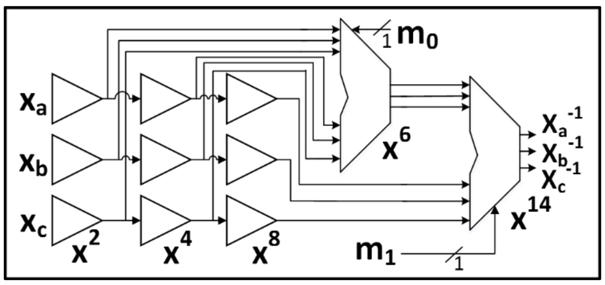


Figure 5. The 3-share TI GF(2^4) (Galois Field) inverter used in TWINE. x^2, x^4, x^8 (produced by squares), and x^6 (where $x^6 = x^2 \cdot x^4$) are intermediate products; x^{14} (where $x^{14} = x^6 \cdot x^8$) is the final product; All bus widths are 4 bits, except for the random bits m_0 and m_1, which are single bits.

2.2. Side-Channel Resistance of Unprotected Versions

We first investigate the information leakage for the unprotected cipher implementations, using TVLA (further discussed in "Materials and Methods), on the Spartan-3E FPGA. The *t*-test graphical results for the unprotected versions of AES (Figure 6a,b), SIMON (Figure 7a), SPECK (Figure 8a), PRESENT (Figure 9a), LED (Figure 10a), and TWINE (Figure 11a) are shown below, where the unprotected time-domain (samples) are on the horizontal axis, and the t-values are on the vertical axis; $t = \pm 4.5$ are shown by the horizontal lines. All unprotected ciphers fail the *t*-test since the *t*-correlation values of $|t| > 4.5$ appear at multiple sample values in each case. Although a *t*-test failure does not prove the vulnerability to DPA, it indicates an information leakage, in that the *t*-test is able to distinguish between two (or several) populations of test vectors.

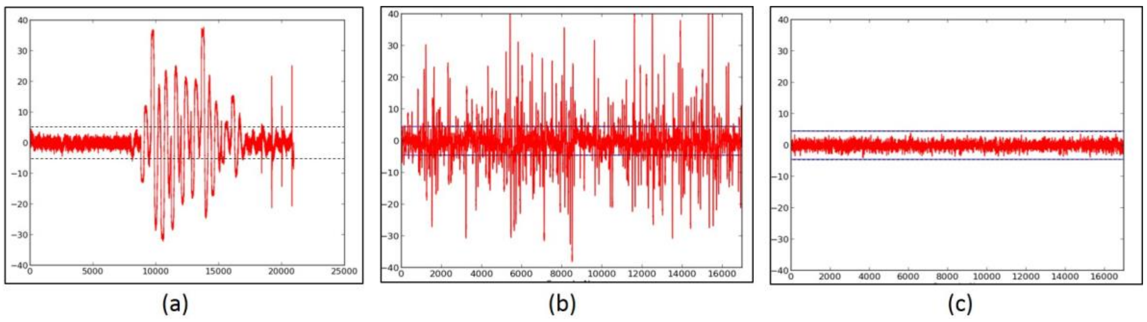


Figure 6. (**a**) The Fixed-versus-Random *t*-test of unprotected AES with a full-width (that is, 128 bit) datapath, basic iterative architecture; (**b**) the 5-stage pipelined 8-bit datapath unprotected AES; (**c**) the 5-stage 8-bit datapath AES protected with hybrid 2-/3-share TI. In this and subsequent figures, the time domain (samples) are on the x-axis; the *t*-values are on the y-axis; and $t = \pm 4.5$ are shown by the horizontal lines.

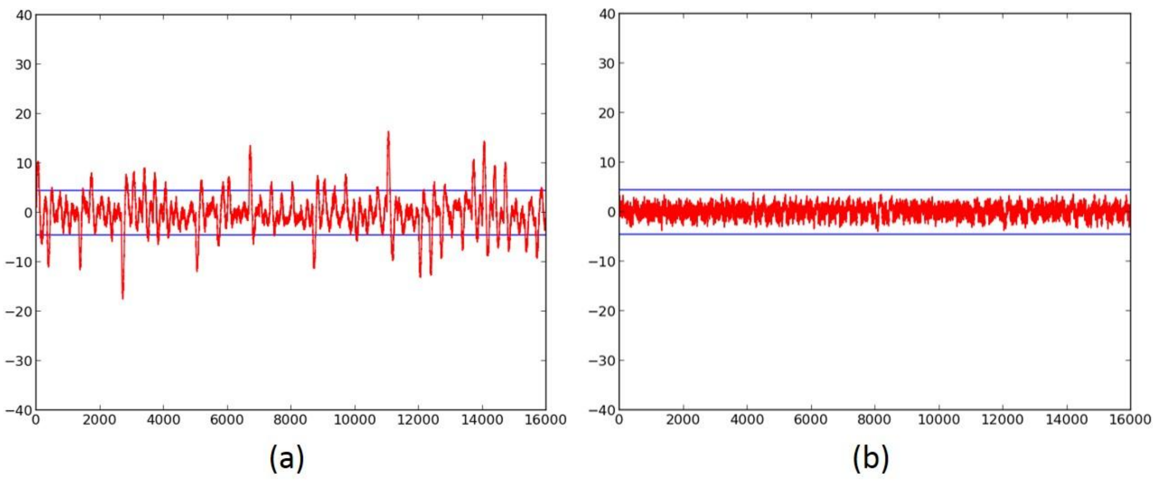


Figure 7. (**a**) The Fixed-versus-Random *t*-test of unprotected SIMON with a full-width (that is, 48 bits) datapath, basic iterative architecture; (**b**) the 3-share TI-protected SIMON with a full-width datapath, basic-iterative architecture.

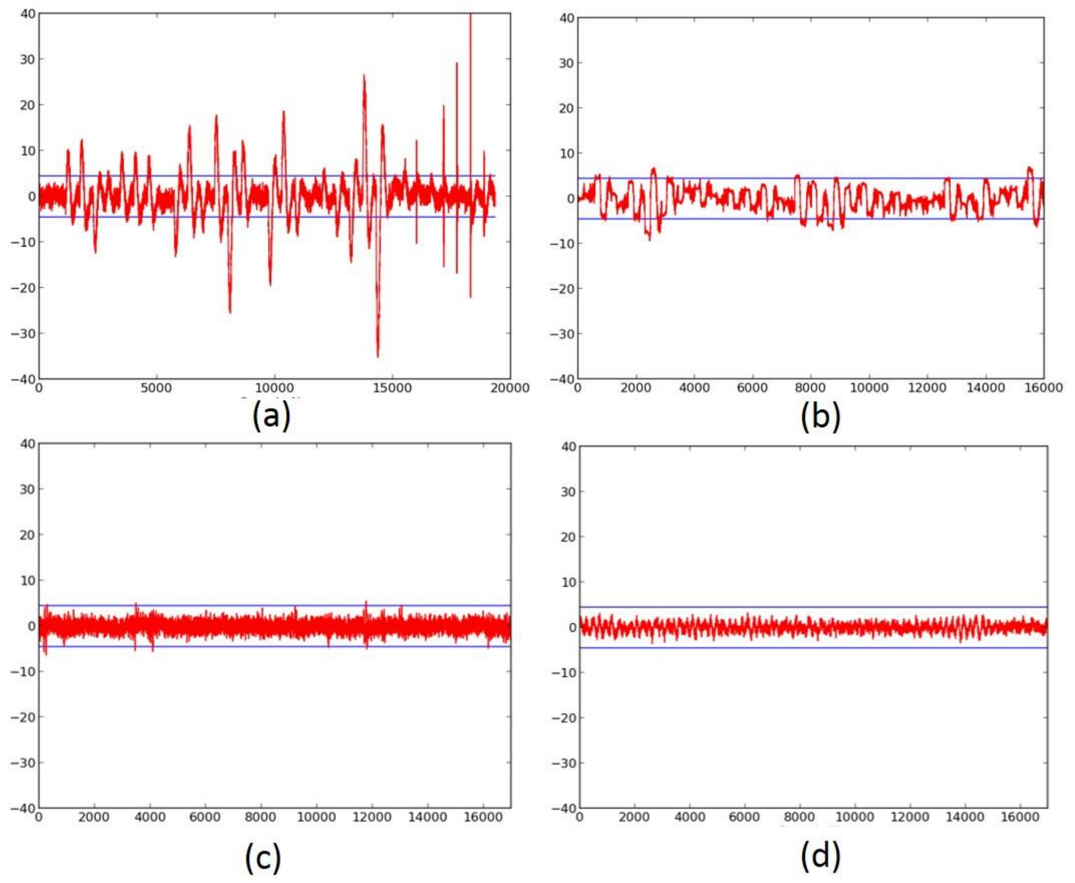


Figure 8. (**a**) The Fixed-versus-Random *t*-test of unprotected SPECK with a full-width (that is, 48 bits) datapath, basic iterative architecture; (**b**) the SPECK with 3-share TI protection and a full-width datapath, basic-iterative architecture; (**c**) the SPECK with 3-share TI protection and a full-width datapath, 8-cycle-per-round architecture, with insufficient refresh randomness; (**d**) the SPECK with 3-share TI protection and a full-width datapath, 8-cycle-per-round architecture with sufficient randomness.

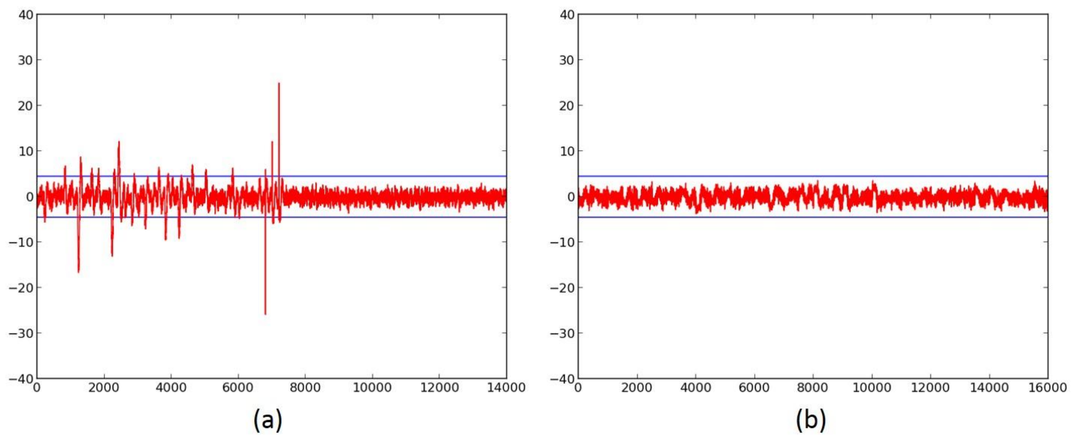
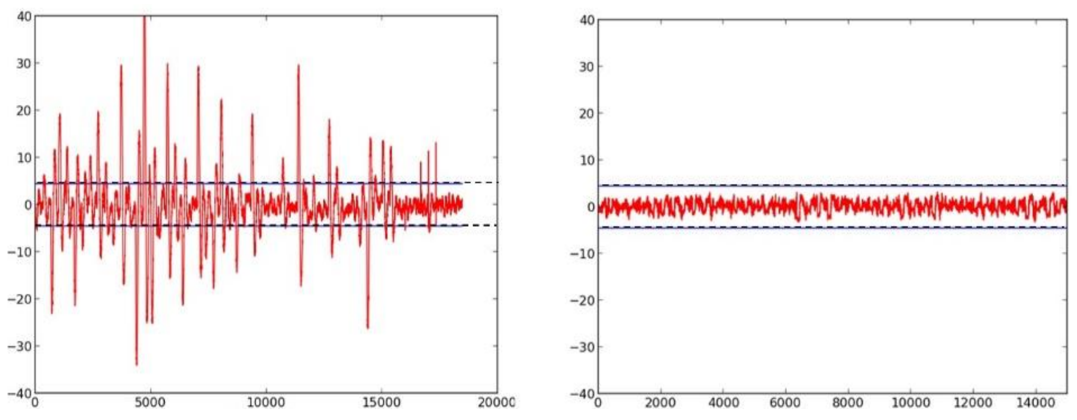


Figure 9. (**a**) The Fixed-versus-Random *t*-test of unprotected PRESENT with a full-width (that is, 64 bits) datapath, basic iterative architecture; (**b**) the 3-share TI-protected PRESENT with a full-width datapath, basic-iterative architecture.



(**a**) (**b**)

Figure 10. (**a**) The Fixed-versus-Random *t*-test of unprotected LED with a full-width (that is, 64 bits) datapath, basic iterative architecture; (**b**) the 3-share TI-protected LED with a full-width datapath, basic-iterative architecture.

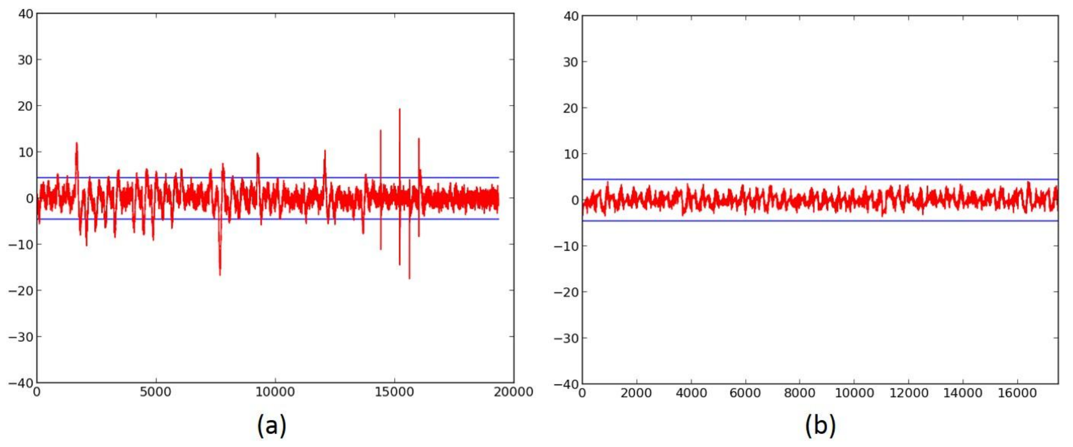


Figure 11. (**a**) The Fixed-versus-Random *t*-test of unprotected TWINE with a full-width (that is, 64 bits), basic iterative architecture; (**b**) the 3-share TI-protected TWINE with a full-width datapath, basic-iterative architecture.

2.3. Failed Attempts at Protection

An important trial of the *t*-test leakage detection methodology is to verify its ability to catch failed protection, that is, ciphers that leak information even after countermeasures are applied. Examples of ciphers that failed *t*-tests after the implementation of countermeasures are shown in Figure 8b,c. These tests are on the 3-share TI-protected version of SPECK, using a full-width basic iterative architecture (Figure 8b), and an 8-cycle-per-round multi-cycle architecture (Figure 8c), respectively. The deficiency in these cipher implementations was the failure to provide refreshed masking for adder stages 1–6. Although the failure in Figure 8b is explainable due to the accumulation of long glitch chains, the failure in Figure 8c is likely due to correlations of outputs appearing due to the violation of TI Property 3 (Uniformity).

2.4. Successful 3-Share TI Protected Ciphers

The ciphers, protected against 1st order DPA using 3-share threshold implementations (TI) as described above, were retested using the TVLA methodology using 2000 fixed-versus-random traces. Randomness for initial masking is externally generated in the software. SIMON, PRESENT, LED, and TWINE achieved satisfactory *t*-tests using full-width datapaths, with basic iterative architectures.

Their results are shown in Figure 7b, and in Figures 9b, 10b and 11b, respectively. We do not achieve full-width basic iterative architecture protected versions of AES and SPECK. The results for the AES 5-stage pipelined version and the SPECK 8-cycle-per-round multi-cycle version are shown in Figures 6c and 8d, respectively.

2.5. Benchmarking of Results

Table 2 shows the results of benchmarking of the unprotected version of the ciphers in this study. The results are generated using Xilinx 14.7 for the Virtex-7 and the Spartan-3E. The frequency is shown in MHz; the throughput (TP) is shown in Mbps, and the throughput-to-area (TP/A) ratio is shown as Mbps/LUT (Look-up Table). The rankings are based on results in the Virtex-7 (in order to frame the results in the context of an advanced FPGA) and are in the order of the lowest area (in LUTs), the highest throughput, and the highest TP/A ratio.

Table 2. The results of the implementation of the unprotected ciphers on Virtex-7 and Spartan-3E Field Programmable Gate Arrays (FPGA).

	Device	**AES**	**AES**	**SIMON**	**SPECK**	**SPECK**	**PRESENT**	**LED**	**TWINE**
Architecture		Full	Pipl [1]	Full	Full	MC [2]	Full	Full	Full
Area (LUT)	Virtex-7	2620	697	435	385	649	381	602	302
	Spartan-3E	2845	1182	565	634	1021	595	727	296
Area (Slice)	Virtex-7	991	253	146	130	202	133	211	122
	Spartan-3E	1691	806	403	462	664	408	486	229
Frequency (MHz)	Virtex-7	229	326	624	363	374	537	309	552
	Spartan-3E	73	128	176	111	116	177	116	200
TP (Mbps)	Virtex-7	2937	238	1152	1245	160	1108	411	982
	Spartan-3E	934	94	329	380	49.7	366	134	355
TP/A (Mbps/LUT)	Virtex-7	1.12	0.34	2.65	3.24	0.25	2.91	0.68	3.25
	Spartan-3E	0.33	0.08	0.57	0.61	0.05	0.62	0.19	1.2
Rank									
Area	Virtex-7	8	7	4	3	6	2	5	1
TP	Virtex-7	1	7	3	2	8	4	6	5
TP/A	Virtex-7	5	7	4	2	8	3	6	1

[1] "Pipl" is "pipelined." [2]"MC" is "multi-cycle".

Table 3 shows the results of the benchmarking of the protected versions of the ciphers that successfully passed the *t*-test and did not show signs of leakage. Table 4 shows the relative cost of protection, computed as the ratio of the protected-to-unprotected area (that is, the growth factor), and ratios of unprotected-to-protected TP and TP/A (that is, the reduction factor) cipher implementations. The rankings are based on the benchmarking results in the Virtex-7 and are in order of the lowest reduction factors for TP/A ratio. Additionally, the TP, area (in LUTs), and TP/A ratios for unprotected and protected cipher versions, benchmarked in the Virtex-7, are shown in Figure 12.

Table 3. The results of the implementation of the protected ciphers on Virtex-7 and Spartan-3E FPGAs.

	Device	**AES**	**SIMON**	**SPECK**	**PRESENT**	**LED**	**TWINE**
Architecture		Pipl [1]	Full	MC [2]	Full	Full	Full
Area (LUT)	Virtex-7	1791	1520	3328	1317	1691	2573
	Spartan-3E	2387	2151	4792	1707	2175	2946
Area (Slice)	Virtex-7	902	434	1714	429	928	1256
	Spartan-3E	1736	1404	3958	1221	1290	1777
Frequency (MHz)	Virtex-7	106	456	334	189	145	207
	Spartan-3E	86	176	108	70	55	67

Table 3. *Cont.*

	Device	AES	SIMON	SPECK	PRESENT	LED	TWINE
TP (Mbps)	Virtex-7	77	841	143	390	193	367
	Spartan-3E	63	326	46	143	73	118
TP/A (Mbps/LUT)	Virtex-7	0.043	0.553	0.043	0.296	0.114	0.143
	Spartan-3E	0.026	0.151	0.010	0.084	0.033	0.040
Random bits		40	0	34	0	0	16
Rank							
Area	Virtex-7	4	2	6	1	3	5
TP	Virtex-7	6	1	5	2	4	3
TP/A	Virtex-7	5	1	6	2	4	3

[1] "Pipl" is "pipelined." [2]"MC" is "multi-cycle".

Table 4. The growth or Reduction Factor in terms of area (LUT), Throughput (TP), and the Throughput-to-area (TP/A) ratios of Protected-versus-Unprotected Versions on Virtex-7 (V7) and Spartan-3E (S3E) FPGAs.

Cipher	LUT (Growth Factor)		TP (Reduction Factor)		TP/A (Reduction Factor)		Ranking
	V7	S3E	V7	S3E	V7	S3E	
AES	2.57	2.03	3.13	1.49	7.94	3.03	4
SIMON	3.49	3.81	1.37	1.00	4.76	3.85	1
SPECK	5.13	4.69	1.12	1.08	5.81	5.00	2
PRESENT	3.46	2.87	2.86	2.56	9.80	7.41	5
LED	2.81	2.99	2.13	1.83	5.95	5.75	3
TWINE	8.52	9.95	2.70	3.03	22.7	30.3	6

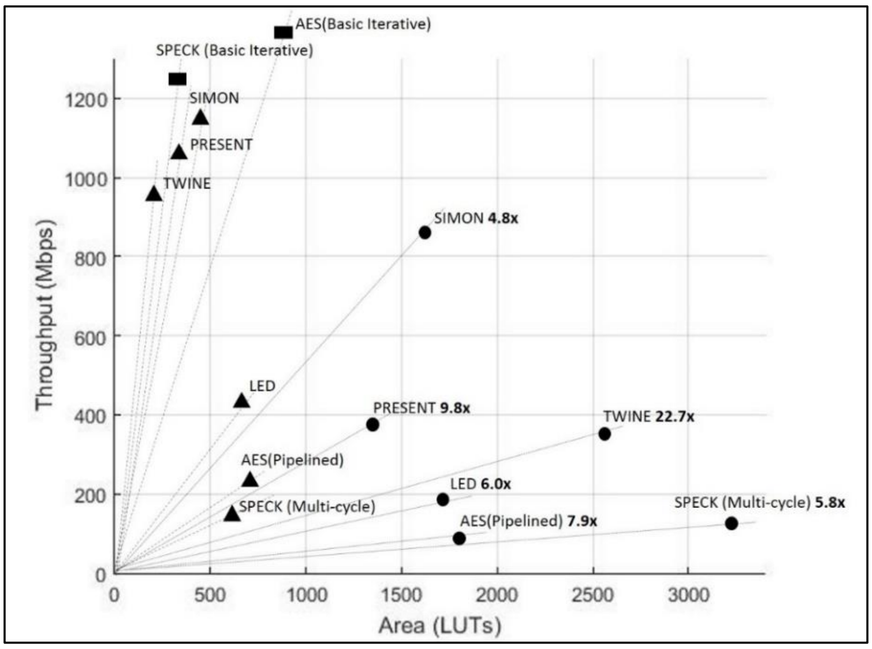


Figure 12. The throughput (Mbps) and area (LUTs) of unprotected and protected ciphers in the Virtex-7 Field Programmable Gate Array (FPGA). The unprotected implementations with corresponding protected versions are shown with triangles. The corresponding protected versions are shown with circles. Versions with rectangles have no corresponding protected version. The slopes of the lines connecting cipher implementations with the origin are equivalent to the throughput-to-area (TP/A) ratio. The reduction factor in the TP/A ratio for the protected versus unprotected versions is shown next to the protected cipher implementations (circles).

Table 5 shows the average power (consisting of static and dynamic power, in mW) and energy per bit (E/bit) (nJ/bit) for the unprotected and protected cipher implementations as measured by FOBOS across a 1 Ω shunt resistor on the Spartan-3E FPGA at 5 MHz. Our power measurements include only the power consumed by the 1.2 V (Vcc_{INT}) line; our measurements do not include the power consumed by the 2.5 V Vcc_{AUX} or the 3.3 V Vcc_{O} lines. The Xilinx XPower Analyzer (XPA) simulations show that about 30% of the static power and about 70% of the dynamic power is consumed by Vcc_{INT} in our tested designs. However, the relative differences in the dynamic power usage between the separate designs, and between the unprotected and protected implementations, are preserved by the FOBOS power measurements. The growth factor for both the power and energy is shown as the ratio of the protected-to-unprotected cipher implementations. The power and E/bit comparisons for the unprotected and protected implementations are shown in Figures 13 and 14, respectively.

Table 5. The average power and E/bit of the unprotected and protected implementations on Spartan-3E FPGA @ 5 MHz.

	AES	SIMON	SPECK	PRESENT	LED	TWINE
			Unprotected			
Average Power (mW)	15.0	13.4	14.0	12.9	15.2	12.9
E/bit (nJ/bit)	4.10	1.45	6.56	1.25	2.28	1.45
			Protected			
Average Power (mW)	19.2	17.7	19.4	18.4	30.0	27.9
E/bit (nJ/bit)	5.25	1.92	9.22	1.78	4.50	3.14
Protected-to-Unprotected Ratio	1.3	1.3	1.4	1.4	2.0	2.2

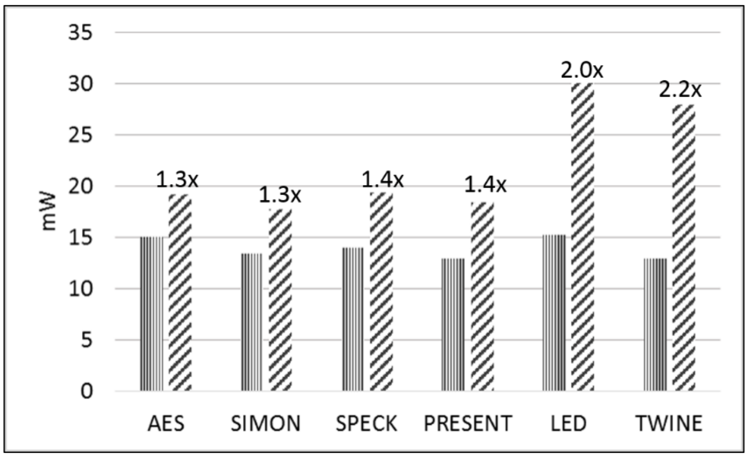


Figure 13. The average power (mW) of the unprotected (vertical lines) and protected (diagonal lines) implementation in the Spartan-3E FPGA at a fixed frequency of 5 MHz, with the growth factor of the protected version depicted above the protected implementations.

In all comparisons (both absolute and relative), the ratios between the protected and unprotected cipher implementations are based on the ciphers with analogous architectures. In other words, the comparisons of protected AES and SPECK (for which full-width protected versions with basic iterative architecture were not achieved) are based on an unprotected 8-bit 5-stage pipelined AES and an 8-cycle unprotected full-width datapath SPECK, respectively.

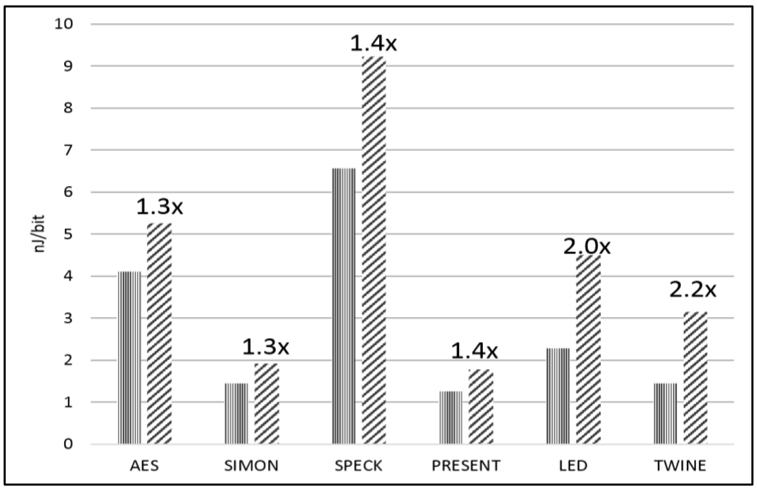


Figure 14. The energy per bit (nJ/bit) of unprotected (vertical lines) and protected (diagonal lines) implementations in the Spartan-3E with a fixed frequency of 5 MHz, with the growth factor of the protected version depicted above the protected implementations.

2.6. Cost of Anti-Optimization Constraints

KEEP constraints prevent nets from being absorbed into adjacent or higher-echelon logic blocks. The KEEP HIERARCHY prevents Xilinx XST (Xilinx Synthesis Technology) from attempting to "flatten" a hierarchy of netlists, which could result in the optimized results for resource use and the reduction of the critical path. The Xilinx "keep" constraints are not designed to provide designs that preserve algorithmic countermeasures (such as Threshold Implementations), but rather, are designed to produce faster implementation results by permitting modules and blocks to be optimized separately [42]. However, the KEEP constraints can have the effect of ensuring the separation of signal paths that are intended by the designer to be logically separate, in order to reduce the possibility of correlation through SCA. We follow the recommendations of References [23–27] and apply KEEP constraints (KEEP SIGNAL and KEEP HIERARCHY) for all protected versions in this research.

The KEEP constraints, however, impose a degradation in terms of the area, throughput, and throughput-to-area (TP/A) ratio. Although the effects of these anti-optimization constraints are dependent on the algorithm, device, and synthesis software, we are nevertheless able to state the general costs based on our analysis of a large number of ciphers.

The relative growth (or reduction) factors of area (in terms of LUTs), throughput (Mbps), and throughput-to-area (TP/A) ratio (in terms of Mbps/LUT) for the Virtex-7 and Spartan-3E FPGAs, when comparing the protected versions with no anti-optimization constraints to the protected versions with anti-optimization constraints applied, are shown in Table 6. The results show that, in our implementations, the use of anti-optimization constraints causes a 22% increase in LUTs, a 4% reduction in frequency, and a 21% reduction in TP/A ratios in the Virtex-7, on average; and a 5% increase in LUTs, a 16% reduction in frequency, and a 20% reduction in TP/A ratios in the Spartan-3E, on average.

The examination of the synthesis results of the selected ciphers shows that the proportion of basic elements (BEL) also changes with the use of anti-optimization constraints. For example, in the SIMON protected implementation, the number of LUT2 BELs grows from 75 with anti-optimization constraints, to 438 when using KEEP constraints (Virtex-7), and from 54 to 395 (Spartan-3E). The corresponding growth in LUT2 BELs for the protected SPECK implementation is 182 to 606 (Virtex-7), and 106 to 615 (Spartan-3E), respectively. This shows that the synthesizer is forced to apportion the logic distribution to a higher number of smaller BELs, which results in the increased number of physical LUTs (6-input and 4-input for Virtex-7 and Spartan-3E, respectively), and a more complex routing arrangement, which results in the reduction of the maximum frequency. The proportion of LUT

BELs in the protected versions of SIMON and SPECK, with and without anti-optimization constraints, are shown in Figures 15 and 16, respectively.

Table 6. The growth or reduction in area, throughput, and throughput-to-area ratio when comparing the protected versions of cipher without, and with, anti-optimization constraints applied.

	Area	TP	TP/A	Area	TP	TP/A
		Virtex-7			Spartan-3E	
AES	1.129	0.926	0.82	1.082	0.838	0.755
SIMON	1.123	0.866	0.721	1.251	0.986	0.788
SPECK	1.286	0.972	0.756	1.071	0.935	0.873
PRESENT	1.254	1.415	1.126	1.047	0.647	0.678
LED	1.028	0.742	0.722	0.889	0.806	0.907
TWINE	1.501	0.843	0.562	0.994	0.803	0.807
Average	1.220	0.961	0.785	1.056	0.836	0.801

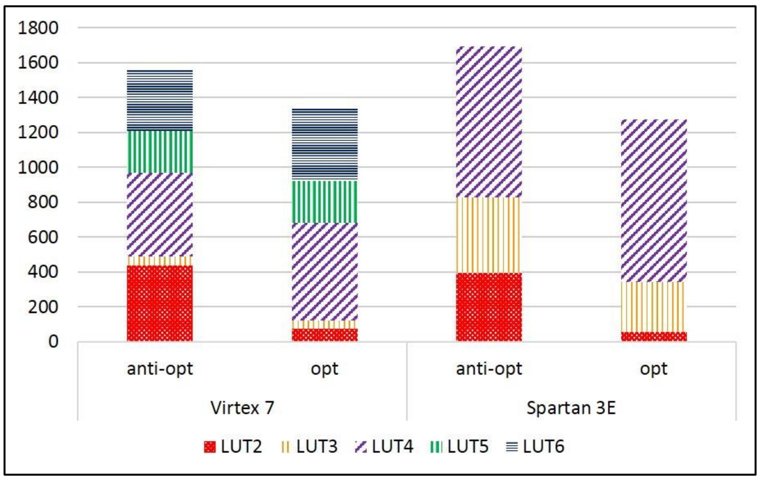


Figure 15. Total and proportional number of LUT2, LUT3, LUT4 for Virtex-7 and Spartan-3E, and LUT5 and LUT6 for Virtex-7 in the SIMON protected implementation with (annotated "anti-opt") and without (annotated "opt") the KEEP SIGNAL and HIERARCHY (that is, anti-optimization) constraints.

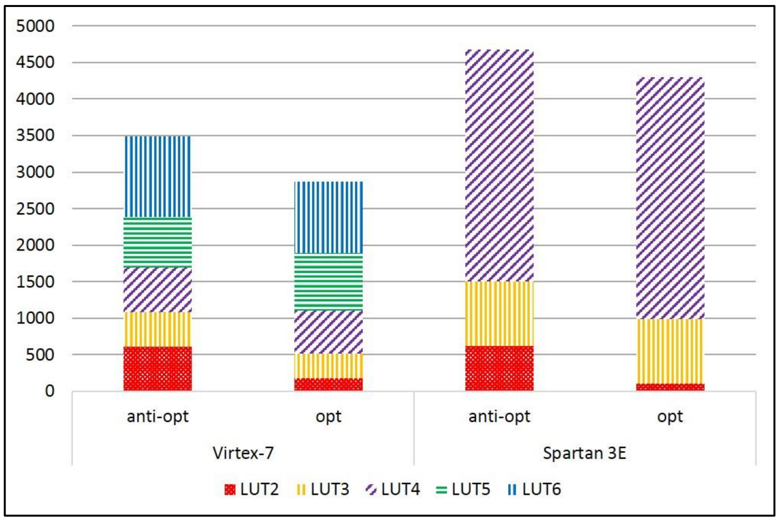


Figure 16. The total and proportional number of LUT2, LUT3, and LUT4 for Virtex-7 and Spartan-3E, and LUT5 and LUT6 for Virtex-7 in SPECK protected implementation with (annotated "anti-opt") and without (annotated "opt") the KEEP SIGNAL and HIERARCHY (that is, anti-optimization) constraints.

3. Discussion

Analysis of results in this research is based on the comparative costs of protection (including absolute and relative costs) and the comparison of these results with previous work.

In terms of absolute costs of protection, SIMON has the highest throughput-to-area (TP/A) ratio of ciphers protected against 1st order DPA in this research, followed by PRESENT, TWINE, LED, AES, and SPECK. In terms of area, PRESENT is the smallest, followed by SIMON, LED, AES, TWINE, and SPECK; in terms of throughput, SIMON is the highest, followed by PRESENT, TWINE, LED, SPECK, and AES. It is important to consider the TP/A ratio as a key performance metric, since:

1. Changes in the cipher protection schemes affect frequency as well as area;
2. The factors of throughput and area are needed to normalize the ciphers that have different block sizes, different architectures, and varying clock cycles per block.

SIMON is particularly well-suited for the threshold implementations since its only non-linearity is the equivalent of a two-input 48-bit `AND` gate. Therefore, the cascading of the non-linearity and random mask refreshing bits are not required.

PRESENT has the lowest area and second best TP/A ratio. Additionally, PRESENT has the lowest E/bit and second-lowest average power. The 3-share TI S-Box implementation used in References [25,26] require only two cascaded levels of functions and did not leak information in our *t*-test, even in a full-width implementation with no random refresh bits. The linear permutation layer of PRESENT is essentially "no cost" in hardware, which saves area in the 3-share threshold implementations. LED uses the same S-Box as PRESENT, and thus, has a relatively low masking cost. However, it has a higher area and significantly lower throughput than PRESENT, due to a higher number of rounds required (48 versus 31 for 80-bit keys). Additionally, the linear transformations in LED are more costly than PRESENT. In fact, each instantiation of MixColumnsSerial takes 140 LUTs on the Virtex-7 and must be instantiated three times for a 3-share TI. A recommendation would be to attempt a hybrid 2-/3- sharing, like that used in AES, at the cost of random resharing bits required at runtime.

TWINE comes in third for the throughput and TP/A ratios. While the S-Box non-linear GF(2^4) inverter strategy employed is simple, it requires only two cascaded non-linear layers and uses 16 bits of randomness per clock cycle for a validated 3-share TI, it has a relatively large growth in area, especially compared to an optimal unprotected TWINE using 4-bit LUT S-Boxes. A recommendation would be to consider a protected S-Box using the techniques discussed in References [31,43].

AES trails most lightweight ciphers in terms of relative growth due to protection. This is a function of its 8-bit S-Box, which has a high algebraic degree and requires four levels of cascaded non-linear functions (fewer levels are possible but require more complex non-linear functions).

Unfortunately, our protected version of SPECK finishes last or nearly last in all categories. This is due to the large cost of masking an adder defined in purely Boolean logic. A recommendation would be to investigate the TI-constructions based on arithmetic masking techniques and alternative adders with higher propagation delay (such as a multi-stage TI-protected carry propagate adder), but lower gate count.

In terms of relative costs of protection, SIMON has the lowest cost in terms of the reduction factor of TP/A ratio on the Virtex-7 FPGA, followed by the SPECK and LED. However, the reduction factor of SPECK is based on a comparison of the multi-cycle protected SPECK to an analogous multi-cycle unprotected SPECK, which is an inherently suboptimal implementation. TWINE and PRESENT have the highest relative costs of protection in terms of TP/A ratios. This is because the 4-bit S-Boxes in unprotected TWINE and PRESENT are very efficiently implemented using LUTs in FPGAs, whereas the combinational logic used for protected S-Boxes is much more costly. The relative cost of protection for ASIC implementations would likely be much less.

In terms of power and E/bit, AES and SIMON have the lowest relative costs of protection and LED and TWINE have the highest relative costs. The low relative cost of AES is explained by the use of a pipelined 8-bit architecture and the instantiation of only one protected S-Box, meaning that there is little additional consumption of power (and minimal change in throughput) for the protected version. However, the low relative cost of protection of SIMON is significant, since both unprotected and protected SIMON implementations are full-width using basic iterative architecture. This supports our conclusion that SIMON is very efficiently protected against DPA.

The protected versions of LED and TWINE have more than double the power and energy consumption of their corresponding unprotected versions. This is due to the differences between the combinational and LUT implementations of S-Boxes, and (in the case of LED) the growth in resources due to the instantiation of three matrix multipliers.

One can also attempt to compare the results in this research with similar results from previous work. Techniques from previous TI-implementations have been considered for the adaptation to ciphers in this research. However, the direct comparison with previous results is still difficult, since the authors adopt different technologies and different optimization strategies, for example, serial, low-area, and so forth

Table 7 shows the previously reported results of individual implementations in ASIC or FPGA. The growth factor is shown as the ratio of protected to unprotected implementations, in terms of Gate Equivalents (GE) or slices. In some cases, the authors produce only a protected version and compare it to a previously published unprotected version. All ASIC implementations are compiled at a fixed frequency of 100 KHz; therefore, the frequency, throughput, and throughput-to-area ratio comparisons are not relevant.

Our 5-stage pipelined AES design has less area growth (that is, 2.56 times more LUTs when comparing protected 8-bit to the unprotected 8-bit pipelined AES), takes fewer clock cycles (175 versus 266 or 256), and uses fewer random bits (40 versus 48 or 44). However, in this research, we mask only the status word and not the secret key, which accounts for some of the above savings.

Regarding PRESENT, although we employ a similar S-Box TI-protection strategy (that is, the 3-share six-function technique used in Reference [25]), we build a full-width basic iterative architecture, in contrast to their serial architectures. Since we invoke 16 S- Boxes per clock cycle, this accounts for an area growth of our protected version of nearly twice that of References [25,26]. A direct comparison of growth in TP/A ratio is not possible since the ASIC versions are implemented at a fixed clock frequency which is not representative of the best performance achievable.

Regarding SIMON and SPECK, a closer comparison with previous results is possible, since the authors of References [27,28,44] use the Spartan-3E FPGA (note that the authors of Reference [27] compare their protected results to the unprotected results of Reference [44]). However, the goal of these studies is to achieve a low area using strictly serial implementations, whereas our goal is for an optimal TP/A ratio. This explains why their ratios of growth, in terms of area, for protected versus unprotected, are 2.7 (SIMON) and 2.3 (SPECK), which are less than our relative costs of 3.8 (SIMON) and 7.6 (SPECK) on the Spartan-3E.

Table 7. The previous results of Threshold Implementation (TI)-protected ciphers addressed in this work.

	AES	**AES**	**PRESENT**	**PRESENT**	**SIMON**	**SPECK**	**LED**
Reference	[24]	[23]	[25]	[26]	[27,44]	[28]	[29]
Width (bits)	8	8	4	4	1	1	64
Architecture	5-pipl [1]	3-pipl	Serial	Serial	Serial	Serial	2-pipl
Technology	180 [2]	180	180	180	Spartan-3E	Spartan-3E	180
Unprotected (UnPr)	2400	2400	1111	1111	36	43	-
Protected (Pr)	10,793	8171	2282	2105	96	99	20,212
Ratio (Pr/UnPr)	4.50	3.40	2.05	1.89	2.67	2.30	-
Shares	3	2-/3-	3	3	3	3	3
Random bits	48	44	0	0	0	0	0
Cycles	266	256	547	2996	4835	2048	96

[1] "Pipl" is "pipelined". [2] "180" is "180 nm ASIC standard libraries".

The implementation of LED in Reference [29] is similar to our full-width 64-bit datapath with 3-share TI-protection. However, it contains additional features to present fault attacks and uses a 128-bit key, and is thus, not directly comparable.

A more direct comparison is possible with the results of Reference [30] since the authors examine similar block ciphers and use a similar methodology. The authors use 3-share threshold implementations (TI) from the literature, implement both the unprotected and protected designs using various ASIC technologies (180 nm, 90 nm, and 45 nm) and compare the area overheads at a commonly assumed maximum frequency. Although their choice of block cipher parameters (that is, block and key size) and their choice of architectures differ from ours, there are similarities in growth ratios from unprotected to protected versions.

The results are compared and contrasted in Table 8. The results show that the absolute ordering of ciphers in terms of area is nearly identical, where in both cases, PRESENT and SIMON are the smallest and second smallest, respectively. The reversal of rankings of AES and SPECK is explained by the fact that, while our AES implementation is similar to Reference [30] (that is, 8-bit multi-cycle implementation), our SPECK implementation is a full-width, multi-cycle implementation while the work in Reference [30] uses a serialized low-area implementation, similar to Reference [28].

Table 8. The comparisons of Implementations of Reference [30] with this work.

	This Work				[30]			
Cipher	**Block/Key**	**Cycles/Block**	**Virtex-7 (LUT)**	**Virtex-7 (Ratio)**	**Block/Key**	**Cycles/Block**	**ASIC (GE)**	**ASIC (Ratio)**
AES	128/128	175	1791	2.57	128/128	266	6340	2.62
SIMON	96/96	52	1520	3.49	128/128	2912	5686	4.61
SPECK	96/96	224	3328	5.13	128/128	2048	5940	2.94
PRESENT	64/80	31	1317	3.46	64/80	547	5236	3.23

The ciphers with the two lowest growth ratios of protected versus unprotected versions in both studies are AES and PRESENT. The difference in growth ratios is explained by our different choices of SIMON and SPECK architecture, where we use full-width implementations with few clock cycles, while implementations in Reference [30] resemble SIMON in Reference [27] and SPECK in Reference [28], respectively.

A direct comparison of the absolute and relative growth in TP/A ratios is not possible since the implementations at Reference [30] use ASIC technologies at a fixed frequency of 100 KHz.

In summary, most results show that PRESENT has both a low area and low cost of protection against 1st order DPA, regardless of hardware implementation, while serialized, lower-area implementations have a lower cost of protection in terms of area growth than parallelized high-speed, full-width implementations.

Further to the above results, it would be desirable to determine to what extent our DPA protection methods actually improve resistance to power analysis side channel attacks. While the TVLA methodology detects information leakage, it is not sufficient to measure incrementally improving resistance provided by the addition of countermeasures. To quantify the improvement of resistance, one should employ methods such as Correlation Power Analysis (CPA), as discussed in Reference [3]. The countermeasure techniques employed in this research are likewise not evaluated for other types of side channel attacks (for example, electromagnetic, acoustic, or optical attacks), nor is any claim of additional resistance provided against fault-injection attacks.

Finally, it is valuable to consider the relative merits of our chosen protection method—threshold implementations (TI)—with other potential countermeasures. Both hardware and software implementations of cryptographic algorithms are vulnerable to power analysis side channel attack, such as Differential Power Analysis (DPA) [2,3,45]. Our research examines a limited subset of RTL hardware implementations of block ciphers, which can additionally all be classified as symmetric (that is, secret key) and constant time (that is, the execution times of these implementations do not vary depending on sensitive data). Over the last two decades, a number of approaches to countermeasures have been attempted for these classifications of ciphers, including *algorithmic* and *non-algorithmic*.

In general, only algorithmic countermeasures (which include Boolean masking and threshold implementations (TI) as applied in this research) are provably secure against DPA. While costs vary with the exact implementation style and degree d of DPA protection, the hardware costs using Boolean masking or TI are generally on the order $O(d^2)$ [19,46] and are measured in increasing number of gates (ASIC) or LUTs/slices (FPGA). The increased area and resource costs are usually accompanied by a slight drop in Throughput (TP) due to an increase in logic levels and routing congestion. Conversely, the use of multi-cycle architectures (with s stages) to meet TI properties, randomness requirements, and glitch resistance, leads to a linear decrease in TP, of the order $O(s)$. TI are the only DPA countermeasure evaluated in this research.

Examples of non-algorithmic countermeasures include hiding, which is accomplished through Dual-rail Pre-charge Logic (DPL), Wave Dynamic Differential Logic (WDDL), and Separated Dynamic and Differential Logic (SDDL) [47–49]. The essence of these techniques is to hide the power consumption caused by processing sensitive data through always computing the complementary function and enforcing transitions on every clock cycle through pre-charge waves. The costs of these techniques vary from an area factor increase of five [48] to about two [49], but they generally increase linearly with number of gates g, that is, $O(g)$. While this is less costly than the best-known costs of TI, there are no provably-secure non-algorithmic countermeasures. This is because hiding techniques require perfect symmetric routing and matching of gate input capacitances between true and complementary logic to be completely effective. This cannot be fully achieved on ASICs, and especially not on FPGAs.

In another example of a non-algorithmic countermeasure, an equivalent or ideally complementary computation is performed on d instances of the same hardware, to achieve dth-order protection against DPA. An example of such a scheme is Multiprocessor Balancing for AES (MUTE-AES) [50]. As in DPL, WDDL, and SDDL, the cost of protection is generally of order $O(d)$. However, while simpler to implement than the above techniques, multiprocessor balancing schemes are likewise not provably secure against DPA.

Table 9 summarizes the above observations on the protection costs for each countermeasure. For our RTL hardware implementations, the "protection cost" (for dth-order protection against DPA) is primarily reflected in the increased resources (LUTs/slices) and power (mW).

Table 9. The summary of costs of various countermeasures against DPA.

Type of Countermeasure	Cost	Example	Reference
Algorithmic	$O(d^2)$	Boolean Masking	[46,51]
Algorithmic	$O(d^2)$	Threshold Implementation	[19], This work
Non-algorithmic	$O(g)$	Dual-rail Pre-charge Logic (DPL)	[47]
Non-algorithmic	$O(g)$	Wave Dynamic Differential Logic (WDDL)	[48]
Non-algorithmic	$O(g)$	Separated Dynamic and Differential Logic (SDDL)	[49]
Non-algorithmic	$O(d)$	Multiprocessor Balancing for AES (MUTE-AES)	[50]

4. Materials and Methods

Our methodology for this research is as follows:

1. We develop implementations for the six ciphers using register transfer level (RTL) methodology in VHDL. Unprotected cipher implementations are either developed entirely by the authors of this research or adopted from publicly-available source codes as annotated above. In order to maximize the throughput-to-area (TP/A) ratio, we use a full-width datapath (that is, the maximum internal datapath corresponding to the full block size in terms of bits), basic iterative architecture (that is, one round in one clock cycle) when possible.

2. We evaluate the DPA resistance of the unprotected ciphers using the FOBOS framework (see below description) [18] and the Test Vector Leakage Assessment (TVLA) (that is, the Welch's *t*-test) leakage detection methodology [15–17]. Leakage is evaluated using a non-specific "fixed-versus-random" *t*-test consisting of 2000 high-fidelity (that is, 16,000–20,000 samples per block encryption) traces, on a custom-modified Spartan-3E FPGA clocked externally at 500 KHz to minimize inductive and capacitive leakage attenuation.
3. We modify the victim ciphers to include a maximum of three shares of TI protection (3-share TI) and try to minimize the additional required randomness for refreshing and resharing masks.
4. We verify the improved DPA resistance of the protected ciphers on FOBOS using the methodology described above. Per the recommendations of Reference [17], we verify the results of the fixed-versus-random *t*-test with at least two sets of fixed data.
5. Although the DPA resistances are verified in only one FPGA (Spartan-3E), we implement (that is, gather Place & Route (PAR) results) all versions on two FPGAs, the Spartan-3E (xc3s500e vq100 -5) (that is, a low-cost device used in the FOBOS architecture) and in the Virtex-7 (xc7vx300t ffg1157 -3) (that is, an expensive high-end FPGA). Implementations use the Xilinx 14.7 ISE® design suite. We prevent Block RAM (BRAM) and Digital Signal Processor DSP instantiation during benchmarking in order to ensure a fair comparison between ciphers. Ciphers are compared in terms of area (Look-up Tables, or "LUTs"), throughput (Mbps), and throughput-to-area (TP/A) ratio.
6. We measure the actual power (mW) for each version on the Spartan-3E FPGA at a fixed frequency of 5 MHz and compute the energy per bit (E/bit) (nJ/bit) by measuring an amplified voltage (using the TI INA225 amplifier) across a 1 Ω shunt resistor coupled to the FOBOS test bench.

Differential Power Analysis (DPA) is used to recover sensitive variables, such as all or a portion of a secret key, by statistically comparing the differences between observed power measurements (for example, collected in "power traces"), and the presumed contents of a sensitive intermediate variable, according to a hypothetical power model [2,3]. However, the authors of References [15,16] recognized that traditional DPA are time- and resource-intensive, in that the attacker must have access to the underlying architecture and conduct expert analysis (often through trial-and-error) to develop an accurate power model.

In cases where we desire to show that a cryptographic implementation is leaking information, or determine whether or not our power-analysis countermeasures are effective, we can employ an expedited leakage assessment methodology called the Test Vector Leakage Assessment (TVLA) methodology, which uses the Welch's *t*-test [15]. As described in References [15–17], the Welch's *t*-test determines whether two distributions are different from one another. In contrast to attack-based testing, the *t*-test finds leakages of information without mounting an attack, does not rely on the knowledge of the underlying architecture, and can quickly reveal when information leaks and when a countermeasure has failed. However, it does not provide information about the difficulty of mounting an attack and cannot be used to recover sensitive intermediate values, such as the secret key.

In the Welch's *t*-test, a figure of merit t is calculated as

$$t = (\mu_0 - \mu_1)/\sqrt{{s_0}^2/n_0 + {s_1}^2/n_1}, \tag{37}$$

where μ_0 and μ_1 are the means of distributions Q_0 and Q_1, s_0 and s_1 are standard deviations, and n_0 and n_1 are the cardinalities of the distributions or the number of samples. If, during our *t*-test, we encounter points in the time domain (that is, "samples") where $|t| > 4.5$, we conclude that "we can distinguish between Q_0 and Q_1," that is, "the device is leaking information."

One method of evaluating leakage on a device, before and after the application of countermeasures, is the "non-specific *t*-test." In one type of non-specific *t*-test, called a "fixed-versus-random" *t*-test, we preselect some "fixed" sensitive data D (for example, a message). Then we randomly interleave the feeding of D, or the random data, to the victim cipher. The power traces collected from the fixed data or random data are used to populate the Q_0 and Q_1 distributions (respectively), upon which the *t*-test is conducted [17].

Our power leakage analysis is performed on the Flexible Open-source Board fOr Side-channel analysis (FOBOS). FOBOS is a free and open tool which provides a single "acquisition to analysis" solution to measure the resistance to power analysis side-channel attack (SCA) and the evaluation of the effectiveness of countermeasures. In this research, we leverage open-source, low-cost hardware, specifically, the Digilent Nexys-2 and Xilinx Spartan-3E FPGA Starter Board, as shown in Figure 17.

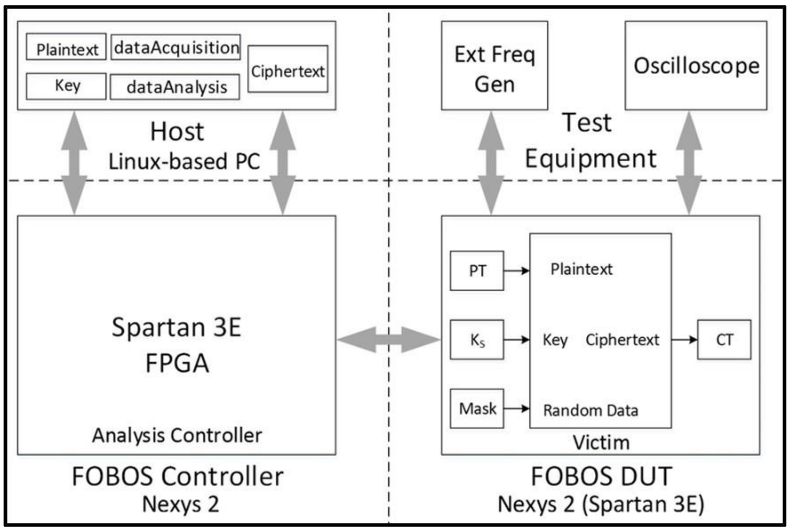


Figure 17. The Flexible Open-source workBench fOr Side-channel analysis (FOBOS) side-channel analysis framework, consisting of Nexys-2 with Spartan-3E FPGA.

A complete description of FOBOS capabilities is available at Reference [18]. We start with the baseline FOBOS software suite available at Reference [18] and modify the analysis toolset to perform non-specific *t*-tests as described above. The procedure for conducting non-specific *t*-tests using the above architecture is summarized as follows:

1. "Fixed-versus-random" test vectors are pre-generated using Python scripts. Random data (generated in software) is substituted for random instances of plaintext. Plaintext and key files are stored in a working directory in the host PC.
2. The victim cipher is instantiated inside a thin wrapper on the FOBOS Device-Under-Test (DUT) board which consists of registers to pre-store plaintext (*PT*), key (K_S), and random data (*Mask*) prior to a block encryption.
3. The user launches the dataAcquisition.py program from the host PC. The acquisition program sends the next block of plaintext and random mask data (used for initial masking of sensitive data) at the start of each trace. The secret key K_S is fixed in this research.
4. The Analysis Controller (located on the FOBOS Controller board) forwards data to the victim cipher and sends a trigger to the oscilloscope (Agilent Technologies DSO6054A) for each trace. Data from the current probe (Tektronix CT2) is transmitted by the oscilloscope to the host PC and stored in a .npy format to allow for post-run analysis [52]. Ciphertext from each trace is stored in the Ciphertext file on the host PC. Although ciphertext is not used for side-channel analysis in this research, it is valuable to examine the ciphertext output to ensure the correct operation of the device.
5. At the completion of all traces, the user conditions the .npy file containing all traces, and "splits" the file into data sets Q_0 and Q_1, in accordance with a "fixed-versus-random" choice metadata file created during test-vector generation.

6. The user runs the *t*-test utility on data sets Q_0 and Q_1, and notes the results.

We adopt several assumptions and simplifications:

1. Any required round keys are computed "on-the-fly;"
2. Only the encryption case is implemented, as the use of the encryption mode is often sufficient to implement both encryption and decryption in an authenticated cipher based on a given block cipher;
3. Only round functions are masked; key scheduling is not masked (with the exception of SIMON, where key sharing is required to achieve uniformity and is relatively of low cost). As discussed in Reference [26], a relevant comparison of ciphers is achieved without key masking;
4. Randomness is simulated by ingesting a large number of random bits (for example, 256 bits for AES) and reusing them after rotations by prime numbers (such as 43 or 61 bits) since an integrated PRNG would require significant additional resources. This assumption of randomness does not affect our tests for a short number of total clock cycles (that is, 30–250 cycles) but is not secure for long-term cipher operation.

5. Conclusions

In this research, we performed a comparison of six secret-key block ciphers—AES, SIMON, SPECK, PRESENT, LED, and TWINE—in terms of cost of protection against differential power analysis (DPA). We tested resistance to 1st order DPA of the unprotected implementations of the above ciphers using the Test Vector Leakage Assessment (TVLA) (that is, Welch's *t*-test) methodology in the FOBOS framework. The results show that unprotected implementations of all of the above ciphers failed the *t*-test and are plausibly vulnerable to DPA.

We then leveraged available published and theoretical techniques to produce 3-share threshold implementation (TI)-protected versions of the above ciphers. We verified improved resistance of the protected ciphers to DPA using the TVLA methodology and the FOBOS framework.

We then compared the unprotected and protected implementations in terms of throughput, area, throughput-to-area (TP/A) ratio, power, and energy per bit (E/bit) to determine absolute and relative costs of protection. Given an identical level of protection (that is, 3-share threshold implementation) against DPA, SIMON has the highest TP/A ratio, followed by PRESENT, TWINE, LED, AES, and SPECK. However, PRESENT uses the least E/bit of the above-protected cipher implementations. Additionally, SIMON has the lowest relative cost of protection in terms of reduction factor in TP/A ratio, and AES has the lowest relative growth in power and E/bit (with SIMON a close-second), when comparing unprotected to protected versions.

All of the protected cipher implementations used anti-optimization constraints to ensure the logical separation of shared signals and the separation of non-linear modules to reduce potential leakage. An analysis of all ciphers in this research showed that anti-optimization techniques result in a 22% increase in LUTs, a 4% reduction in frequency, and a 21% reduction in TP/A ratios in the Virtex-7, on average; and a 5% increase in LUTs, a 16% reduction in frequency, and a 20% reduction in TP/A ratios in the Spartan-3E FPGAs, on average.

In comparison to previous individually-reported results, our AES results are approximately on par with previous 3-share TI-protected ciphers, although the use of different assumptions, architectures, goals, and technologies make a direct comparison difficult. Our protected implementations of SIMON, SPECK, and PRESENT experience roughly twice the growth of the previously reported results due to our selection of full-width basic iterative architecture vice the area-optimized serialized approach adopted in previous research.

The difficulty in making comparisons with previous TI-protected results shows the value of our direct comparison of unprotected and protected implementations of six ciphers. Our research validates one of the advantages of the TVLA methodology in providing a comparative analysis of a large number of ciphers, which would be very time-consuming using historical examples of differential power analysis and attack-based key-recovery techniques alone.

Acknowledgments: This research is based on work supported by the National Science Foundation under Grant No. 1314540.

Author Contributions: William Diehl conceived and designed the experiments and research methodology; William Diehl and Abubakr Abdulgadir performed the experiments and recorded results outlined in “Results”; William Diehl and Kris Gaj analyzed the results, and formulated the findings in “Discussions” and “Conclusions”; Abubakr Abdulgadir and Jens-Peter Kaps contributed to the equipment and test methodology discussed in “Materials and Methods”; William Diehl wrote the paper.

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

References

1. Diehl, W.; Abdulgadir, A.; Kaps, J.-P.; Gaj, K. Comparing the cost of protecting selected lightweight block ciphers against differential power analysis in low-cost FPGAs. In Proceedings of the International Conference on Field Programmable Technologies (FPT 2017), Melbourne, Australia, 11–13 December 2017; pp. 128–135.
2. Kocher, P.C.; Jaffe, J.; Jun, B. Differential power analysis. In Proceedings of the CRYPTO ’99—19th International Conference on Cryptology, Santa Barbara, CA, USA, 15–19 August 1999.
3. Kocher, P.; Jaffe, J.; Jun, B.; Rohatgi, P. Introduction to Differential Power Analysis. *J. Cryptogr. Eng.* **2011**, *1*, 5–27. [CrossRef]
4. CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness. Available online: http://competitions.cr.yp.to/caesar.html (accessed on 31 March 2018).
5. Bernstein, D. Cryptographic Competitions. Google Groups, 16 July 2016. Available online: https://groups.google.com/forum/#!forum/crypto-competitions (accessed on 27 February 2018).
6. National Institute of Standards and Technology (NIST). Lightweight Cryptography. Available online: https://www.nist.gov/programs-projects/lightweight-cryptography (accessed on 31 March 2018).
7. McKay, K.; Bassham, L.; Turan, M.; Mouha, N. *Report on Lightweight Cryptography (NISTIR 8114)*; National Institute of Standards and Technology (NIST): Gaithersburg, MD, USA, 2017. Available online: http://nvlpubs.nist.gov/nistpubs/ir/2017/NIST.IR.8114.pdf (accessed on 27 February 2018).
8. Federal Information Processing Standards Publication 197, Advanced Encryption Standard (AES). Available online: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf (accessed on 16 April 2018).
9. Beaulieu, R.; Treatman-Clark, S.; Shors, D.; Weeks, B.; Smith, J.; Wingers, L. The SIMON and SPECK lightweight block ciphers. In Proceedings of the 2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, 8–12 June 2015; pp. 1–6.
10. Bogdanov, A.; Knudsen, L.; Leander, G.; Paar, C.; Poschmann, A.; Robshaw, M.; Seurin, Y.; Vikkelsoe, C.; Paillier, P.; Verbauwhede, I. PRESENT: An ultra-lightweight block cipher. In Proceedings of the Cryptographic Hardware and Embedded Systems—CHES 2007: 9th International Workshop, Vienna, Austria, 10–13 September 2007; pp. 450–466.
11. Guo, J.; Peyrin, T.; Poschmann, A.; Robshaw, M. The LED block cipher. In Proceedings of the Cryptographic Hardware and Embedded Systems (CHES 2011): 13th International Workshop, Nara, Japan, 28 September–1 October 2011; pp. 326–341.
12. Suzaki, T.; Minematsu, K.; Morioka, S.; Kobayashi, E. TWINE: A Lightweight Block Cipher for Multiple Platforms. *SAC* **2012**, *7707*, 339–354.
13. Iwata, T.; Minematsu, K.; Guo, J.; Morioka, S.; Kobayashi, E. CLOC and SILC v3. September 2016. Available online: https://competitions.cr.yp.to/round3/clocsilcv3.pdf (accessed on 31 March 2018).
14. Wu, H.; Huang, T. JAMBU Lightweight Authenticated Encryption Mode. 2016. Available online: http://www3.ntu.edu.sg/home/wuhj/research/caesar/caesar.html (accessed on 27 February 2018).
15. Cooper, J.; DeMulder, E.; Goodwill, G.; Jaffe, J.; Kenworthy, G.; Rohatgi, P. Test Vector Leakage Assessment (TVLA) methodology in practice. In Proceedings of the International Cryptographic Module Conference, Gaithersburg, MD, USA, 24–26 September 2013.
16. Goodwill, G.; Jun, B.; Jaffe, J.; Rohatgi, P. A testing methodology for side channel resistance validation. In Proceedings of the NIST Non-Invasive Attack Testing Workshop, Nara, Japan, 25–27 September 2011.
17. Schneider, T.; Moradi, A. Leakage Assessment Methodology. *J. Cryptogr. Eng.* **2016**, *6*, 85–89. [CrossRef]

18. Cryptographic Engineering Research Group. Flexible Open-Source WorkBench fOr Side-Channel Analysis (FOBOS). 25 October 2016. Available online: https://cryptography.gmu.edu/fobos/ (accessed on 27 February 2018).
19. Nikova, S.; Rechberger, C.; Rijmen, V. Threshold implementations against side-channel attacks and glitches. In Proceedings of the 8th International Conference on Information and Communications Security, Raleigh, NC, USA, 4–7 December 2006; Volume 4307, pp. 529–545.
20. Shamir, A. How to Share a Secret. *Commun. ACM* **1979**, *22*, 612–613. [CrossRef]
21. Yao, A. Protocols for secure computation. In Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (sfcs 1982), Chicago, IL, USA, 3–5 November 1982; pp. 160–164.
22. Mangard, S.; Pramstaller, N.; Oswald, E. Successfully attacking masked AES hardware implementations. In *International Workshop on Cryptographic Hardware and Embedded Systems*; Springer: Heidelberg/Berlin, Germany, 2005; Volume 3659, pp. 157–171.
23. Bilgin, B.; Gierlichs, B.; Nikova, S.; Nikov, V.; Rijmen, V. A more efficient AES threshold implementation. In Proceedings of the 7th International Conference on Cryptology in Africa (AFRICACRYPT 2014), Marrakesh, Morocco, 28–30 May 2014; pp. 267–284.
24. 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 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2011), Tallinn, Estonia, 15–19 May 2011; pp. 69–98.
25. Poschmann, A.; Moradi, A.; Khoo, K.; Lim, C.; Wang, H.; Ling, S. Side-Channel Resistant Crypto for Less than 2300 GE. *J. Cryptol.* **2011**, *24*, 322–345. [CrossRef]
26. Kutzner, S.; Nguyen, P.; Poschmann, A.; Wang, H. On 3-share threshold implementations for 4-Bit S-boxes. In Proceedings of the Constructive Side-Channel Analysis and Secure Design: 4th International Workshop, COSADE 2013, Paris, France, 6–8 March 2013; pp. 99–113.
27. Shahverdi, A.; Taha, M.; Eisenbarth, T. Lightweight Side Channel Resistance: Threshold Implementations of Simon. *IEEE Trans. Comput.* **2017**, *66*, 661–671. [CrossRef]
28. Chen, C.; Inci, M.S.; Taha, M.; Eisenbarth, T. SpecTre: A tiny side-channel resistant speck core for FPGAs. In Proceedings of the Smart Card Research and Advanced Applications: 15th International Conference, CARDIS 2016, Cannes, France, 7–9 November 2016; pp. 73–88.
29. Schneider, T.; Moradi, A.; Güneysu, T. ParTI—Towards combined hardware countermeasures against side-channel and fault-injection attacks. In Proceedings of the 36th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO 2016), Santa Barbara, CA, USA, 14–18 August 2016; pp. 302–332.
30. Sadhukhan, R.; Patranabis, S.; Ghoshal, A.; Mukhopadhyay, D.; Saraswat, V.; Ghosh, S. An Evaluation of Lightweight Block Ciphers for Resource-Constrained Applications: Area, Performance, and Security. *J. Hardw. Syst. Secur.* **2017**, *1*, 203–218. [CrossRef]
31. Canright, D.; Batina, L. A Very Compact 'Perfectly Masked' S-box for AES. In Proceedings of the 6th International Conference on Applied Cryptography and Network Security, ANCS 2008, New York, NY USA, 3–6 June 2008; Volume 5037, pp. 446–459.
32. Gaj, K.; Chodowiec, P. FPGA and ASIC implementations of AES. In *Cryptographic Engineering*; Koç, Ç.K., Ed.; Springer Science & Business Media: New York, NY, USA, 2009; pp. 235–294.
33. Goubin, L. A sound method for switching between Boolean and arithmetic masking. In Proceedings of the Cryptographic Hardware and Embedded Systems—CHES 2001, Third International Workshop, Paris, France, 14–16 May 2001; Volume 2162, pp. 3–15.
34. Messerges, T. Securing the AES finalists against power analysis attacks. In *Fast Software Encryption*; Springer: Heidelberg/Berlin, Germany, 2002; Volume 1978, pp. 150–164.
35. Coron, J.S.; Tchulkine, A. A New algorithm for switching from arithmetic to Boolean masking. In Proceedings of the Cryptographic Hardware and Embedded Systems—CHES 2003: 5th International Workshop, Cologne, Germany, 8–10 September 2003; pp. 89–97.
36. Debraize, B. Efficient and provably secure methods for switching from arithmetic to Boolean masking. In Proceedings of the 14th International Conference on Cryptographic Hardware and Embedded Systems—CHES 2012, Leuven, Belgium, 9–12 September 2012; Volume 7428 of the Series Lecture Notes in Computer Science; pp. 107–121.

37. Golic, J. Techniques for random masking in Hardware. In *IEEE Transactions on Circuits and Systems I: Regular Papers*; IEEE: Piscataway, NJ, USA, 2007; Volume 54, pp. 291–300.
38. Coron, J.; Großschädl, J.; Vadnala, P. Secure conversion between Boolean and arithmetic masking of any order. In Proceedings of the Cryptographic Hardware and Embedded Systems—CHES 2014, 16th International Workshop, Busan, Korea, 23–26 September 2014.
39. Schneider, T.; Moradi, A.; Güneysu, T. Arithmetic addition over Boolean Masking. In Proceedings of the Applied Cryptography and Network Security: 13th International Conference, ACNS 2015, New York, NY, USA, 2–5 June 2015; pp. 559–578.
40. Kogge, P.; Stone, H. A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations. *IEEE Trans. Comput.* **1973**, *100*, 786–793. [CrossRef]
41. Rivain, M.; Prouff, E. Provably secure higher-order masking of AES. In Proceedings of the Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, 17–20 August 2010; Volume 6225, pp. 413–427.
42. Xilinx. UG625 Constraints Guide (v. 14.5). Available online: https://www.xilinx.com/support/documentation/sw_manuals/xilinx14_7/cgd.pdf (accessed on 16 April 2018).
43. Bilgin, B.; Nikova, S.; Nikov, V.; Rijmen, V. Threshold implementation of all 3 × 3 and 4 × 4 S-boxes. In Proceedings of the Cryptographic Hardware and Embedded Systems CHES 2012: 14th International Workshop, Leuven, Belgium, 9–12 September 2012; pp. 76–91.
44. Aysu, A.; Gulcan, E.; Schaumont, P. SIMON Says: Break Area Records of Block Ciphers on FPGAs. *IEEE Embed. Syst. Lett.* **2014**, *6*, 37–40. [CrossRef]
45. Ambrose, J.; Ignjatovic, A.; Parameswaran, S. *Power Analysis Side Channel Attacks: The Processor Design-Level Context*; VDM Publishing: Saarbrücken, Germany, 2010.
46. Ishai, Y.; Sahai, A.; Wagner, D. Private Circuits: Securing Hardware against Probing Attacks. In *Advances in Cryptology, CRYPTO 2003*; Volume 2729 of Lecture Notes in Computer Science; Springer: Heidelberg/Berlin, Germany, 2003; pp. 463–481.
47. Tiri, K.; Verbauwhede, I. A logic level design methodology for a secure DPA Resistant ASIC or FPGA implementation. In Proceedings of the Design, Automation and Test in Europe Conference and Exhibition, Paris, France, 16–20 February 2004; pp. 246–251.
48. Yu, P.; Schaumont, P. Secure FPGA circuits using controlled placement and routing. In Proceedings of the 5th IEEE/ACM International Conference on Hardware/Software Codesign and System Synthesis, Salzburg, Austria, 30 September–3 October 2007; pp. 45–50.
49. Velegalati, R.; Kaps, J.P. DPA resistance for light-weight implementations of cryptographic algorithms on FPGAs. In Proceedings of the 2009 International Conference on Field Programmable Logic and Applications, Prague, Czech Republic, 31 August–2 September 2009; pp. 385–390.
50. Ambrose, J.A.; Parameswaran, S.; Ignjatovic, A. MUTE-AES: A multiprocessor architecture to prevent power analysis-based side channel attack of the AES algorithm. In Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design, San Jose, CA, USA, 10–13 November 2008; pp. 678–684.
51. Diehl, W.; Gaj, K. Implementation of a Boolean masking scheme for the SCREAM cipher. In Proceedings of the 2016 Euromicro Conference on Digital System Design (DSD), Limassol, Cyprus, 31 August–2 September 2016; pp. 723–726.
52. Kern, R. A Simple File Format for NumPy Arrays. 20 December 2007. Available online: https://docs.scipy.org/doc/numpy-dev/neps/npy-format.html (accessed on 27 February 2018).