Hindawi
IET Information Security
Volume 2024, Article ID 8851878, 17 pages
https://doi.org/10.1049/2024/8851878

Research Article

MS-LW-TI: Primitive-Based First-Order Threshold Implementation for 4 × 4 S-boxes

Botao Liu and Ming Tang

Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, Hubei, China

Correspondence should be addressed to Ming Tang; m.tang@126.com

Received 26 July 2023; Revised 15 March 2024; Accepted 8 April 2024; Published 11 May 2024

Academic Editor: Tianyu Ye

Threshold implementation (TI) is a lightweight countermeasure against side-channel attacks when glitches happen. As to masking schemes, an S-box is the key part to protection. In this paper, we propose a general first-order lightweight TI scheme for 4 × 4 S-boxes and name it as MiniSat-lightweight-threshold implementation (MS-LW-TI). First, we use MiniSat to optimally decompose an S-box into the least number of three different logic gate operations, AND, OR, and XOR. Among these operations, we define two primitives and the extension of two primitives for TI design. Furthermore, we prove that the primitives and their extensions strictly comply with the security properties. Finally, we implement MS-LW-TI on Xilinx Spartan-6 Field Programmable Gate Array (FPGA) to show that the S-boxes of PRESENT, GIFT, and PICCOLO consume only 17, 15, and 13 look-up-tables (LUTs), 16, 9, and 16 flip-flops (FFs), 6, 5, and 6 slices, respectively. Compared with the existing lightweight TI design, our TI for PRESENT S-box has a 22%, 38%, and 25% reduction of LUTs, FFs, and slices to the design by Shahmirzadi and Moradi at IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES) 2021, and our TI for GIFT S-box has a 6%, 25%, and 28% reduction of LUTs, FFs, and slices to the design by Jati et al., which is the smallest.

1. Introduction

All along, the constrained devices of the Internet of Things (IoT) have been in demand, for example, RFID tags, smart cards, and sensors in wireless networks with low computing power and implementation area. Many lightweight block ciphers such as PRESENT [1], GIFT [2], PICCOLO [3], and LED [4] have been proposed to protect the sensitive data of constrained devices.

The physical security of cryptography has made remarkable attention since Kocher et al. [5] proposed side-channel attacks (SCA) in 1999. Unprotected hardware implementations of lightweight block ciphers are vulnerable to SCA [6–8]. Masking uses secret sharing to eliminate the correlation between sensitive data and power consumption during operation and is considered as the theoretical secure resistance [9]. However, masking has to face potential leakage in hardware implementation when glitches happen. In 2006, Nikova et al. [10] proposed threshold implementation (TI) to solve the weakness from glitches. TI has become well-known in hardware masking schemes.

1.1. Related Works. Lightweight block ciphers are used in resource-constrained environments and have strict requirements for resources, especially hardware resources [11]. Therefore, the topic of lightweight and side-channel resistant implementation is a pressing issue for these ciphers [12].

TI is based on secret sharing and multiparty computation methods, and it must satisfy three important properties: correctness, noncompleteness, and uniformity. For the linear components of lightweight cryptographic algorithms, it is easy to satisfy the three properties at the same time. However, the nonlinear components (such as the S-box) are challenging to meet and constructing the TI of an S-box takes a lot of resources. For example, an S-box needs to be added fresh randomness [13] or requires a larger number of shares [14]. Thus, the lightweight TI of these ciphers reduces the number of shares without adding random numbers. According to the TI setting, an S-box with algebraic degree t should

be split into $td+1$ shares to achieve security against dth order attacks. An S-box with a high algebraic degree can be decomposed into multiple S-boxes with a lower algebraic degree. Then, these S-boxes can be designed for pipeline implementation. Generally, the Boolean equation of a 4×4 S-box is represented by algebraic normal form (ANF), and its algebraic degree is 3. Poschmann et al. [12] have decomposed the S-box with algebraic degree 3 into two S-box with algebraic degree 2 to construct a three shares TI of PRESENT S-box. Afterward, Kutzner et al. [15] decomposed the PRESENT S-box of degree 3 into combining two quadratic functions and some linear functions to construct a lightweight three shares TI of PRESENT S-box. Since LED uses the S-box of PRESENT, Yao et al. [16] adopted the decomposed S-box of Poschmann et al. [12] directly. Jati et al. [17] decomposed the S-box of GIFT into two S-boxes, which are selected by the LIGHTER tool with estimating gate equivalents, and they realized a three shares TI of GIFT S-box. The three shares TI of GIFT S-box was used by the work of Satheesh and Shanmugam [18] and the work of Caforio et al. [19], respectively. Meanwhile, Reparaz et al. [20] and Gross et al. [21] showed how to use only $d+1$ shares when dth order security. Chen et al. [22] adopted the decomposed S-box of Kutzner et al. [15] to realize two shares TI of PRESENT S-box. Subsequently, Shahmirzadi and Moradi [23] proposed lightweight two shares TI of S-boxes of lightweight block ciphers.

The idea of using low algebraic degree S-boxes instead of one high algebraic degree S-box, in essence, reduces redundant logic units. However, there are different decomposition methods for an S-box, and the decomposition methods of the above schemes still have redundant logic units in the ANF equation of S-boxes. These redundant logic units lead to a large number of AND and XOR gate operations for TI. The cost of these gates is very expensive for lightweight implementation.

1.2. Our Contributions. To further solve the problem of lightweight and side-channel resistant implementation, we propose a general, efficient, and low-resource TI scheme named MS-LW-TI.

S-boxes contain the least AND, OR, and XOR gates after MiniSat optimization. Based on these logic gates, we constructed two primitives for TI, which are $t=x\oplus y$ and $t=x\times y\oplus z$. For 4×4 S-boxes, we can use the input variables as z of the primitive. For 8×8 S-boxes, we have to add fresh randomness or merge AND logic gates. We build the first-order MS-LW-TI based on two primitives and their extensions. MS-LW-TI can guarantee the first-order glitch-extended probing secure.

We implement MS-LW-TI on the S-boxes of PRESENT, GIFT, and PICCOLO, which consume only 17, 15, and 13 look-up-tables (LUTs); 16, 9, and 16 flip-flops (FFs); 6, 5, and 6 slices; and 3, 3, and 2 clock cycles, respectively. Compared with the existing lightweight TI design, our two-shares scheme for PRESENT S-box has a 22%, 38%, and 25% reduction of LUTs, FFs, and slices to the design by Shahmirzadi and Moradi at TCHES 2021 [23], and our three-shares scheme for GIFT S-box has a 6%, 25%, and 28% reduction of LUTs, FFs, and slices to the design by Jati et al. [17], which is the smallest one presently.

1.3. Organization. In Section 2, we introduce the security properties of TI, and some classic two and three shares TI of S-boxes of lightweight block ciphers. In Section 3, we analyze two primitives and the extensions of the first-order MS-LW-TI. In Section 4, we present the implementation and security analysis of the first-order MS-LW-TI. Section 5 concludes the paper.

2. Preliminaries

2.1. Threshold Implementations (TIs). As we know, glitches can occur in the combinational circuit of cryptographic algorithms. In this instance, Faust et al. [24] proposed the glitch-extended probing model. It involves placing a probe on an output port of a circuit, propagating it backward, and extending it to multiple probes at the input ports of the combinational circuit that drives the probed port. For the dth order security of a nonlinear function of S-box, there are at least $(d+1)$ input shares of TIs. After dividing the binary variable $x\in\mathbb{F}_2$ into $(d+1)$ shares $\tilde{x}=\{x_0,\ldots,x_d\}$, the coordinate Boolean function f is split into $(d+1)$ shared functions $\tilde{f}=\{f_0,\ldots,f_d\}$. TI has to satisfy three important properties, namely correctness, noncompleteness, and uniformity [10, 13]. Given a variable x, its share is represented as follows:

$$x=x_0\oplus x_1\oplus\ldots\oplus x_d. \tag{1}$$

For Boolean functions f, its share is represented as follows:

$$f=f_0\oplus f_1\oplus\ldots\oplus f_d. \tag{2}$$

Property 1 (Correctness). *The shared functions $\{f_0,\ldots,f_d\}$ are said to be correct if the output share $\{y_0,\ldots,y_d\}$ represents the output y of the original Boolean function f, i.e.:*

$$y=f(x)=y_0\oplus y_1\oplus\ldots\oplus y_d=f_0(\widetilde{x}_0)\oplus f_1(\widetilde{x}_1)\oplus\ldots\oplus f_d(\widetilde{x}_d), \tag{3}$$

where the reduced sharing elements are defined by the following equation:

$$\widetilde{x}_0=\{x_1,\ldots,x_d\},\widetilde{x}_1=\{x_0,x_2,\ldots,x_d\},\widetilde{x}_d=\{x_0,x_1,\ldots,x_{d-1}\}. \tag{4}$$

Property 2 (Noncompleteness). *Every shared function is independent of at least one share of the input variable x. The shared functions in Equation (3) are noncomplete.*

Property 3 (Uniformity). *If the function f is invertible, then uniformity is satisfied by invertible realizations. And the probabilistic distributions of the shared and original inputs are*

denoted by $\tilde{P}_I(\tilde{x})$ and $P_I(x)$, respectively. The input share is said to be uniform if and only if, for any input x, its shares occur with the same probability:

$$\tilde{P}_I(\tilde{x}) = \frac{P_I(x)}{\alpha} = \frac{P_I(x_0 \oplus x_1 \oplus \ldots \oplus x_d)}{\alpha}, \quad (5)$$

where α is a constant related to the number of shares and the value of variables. And given uniform input shares, the probabilistic distributions of the shared and original outputs are denoted by $\tilde{P}_I(\tilde{y})$ and $P_I(y)$, respectively. The output share is said to be uniform if and only if, for any output y, its shares occur with the same probability:

$$\tilde{P}_I(\tilde{y}) = \frac{P_I(y)}{\alpha} = \frac{P_I(y_0 \oplus y_1 \oplus \ldots \oplus y_d)}{\alpha}. \quad (6)$$

2.2. State-of-the-Art Two and Three Shares Threshold Implementations. TI with $(d+1)$ or $(td+1)$ input shares to resist dth order attacks have been shown in [12, 15–19, 22, 23]. Considering the boundaries of the first-order TI when the t is 2 and the d is 1, there are two or three input shares. Some classic TI of S-boxes of lightweight block ciphers are described below.

Poschmann et al. [12] proposed a three shares first-order TI of PRESENT, in which the cubic S-box is decomposed into two quadratic S-boxes G and F represented as $S(x) = F(G(x))$. The ANF equations of G and F have nine AND gates and 19 XOR gates. The ANF equations of three shares TI of S-box are shown in [12], which have 81 AND gates and 105 XOR gates. And the ANF equations of G and F are as follows:

$$\begin{aligned}
&G(x_3, x_2, x_1, x_0) = (g_3, g_2, g_1, g_0),\\
&g_0 = 1 \oplus x_0 \oplus x_3 \times x_2 \oplus x_3 \times x_1 \oplus x_2 \times x_1,\\
&g_1 = 1 \oplus x_3 \oplus x_1 \oplus x_2 \times x_0 \oplus x_1 \times x_0,\\
&g_2 = 1 \oplus x_2 \oplus x_1, g_3 = x_2 \oplus x_1 \oplus x_0,\\
&F(x_3, x_2, x_1, x_0) = (f_3, f_2, f_1, f_0),\\
&f_0 = x_1 \oplus x_2 \times x_0, f_1 = x_2 \oplus x_1 \oplus x_3 \times x_0,\\
&f_2 = x_3 \oplus x_1 \times x_0, f_3 = x_2 \oplus x_1 \oplus x_0 \oplus x_3 \times x_0.
\end{aligned} \quad (7)$$

Kutzner et al. [15] also proposed a three shares first-order TI of PRESENT, in which the S-box can be decomposed as $S(x) = A(G(G(B(x))))$, and the ANF equations of A, G, and B are as follows:

$$\begin{aligned}
&B(x_3, x_2, x_1, x_0) = (b_3, b_2, b_1, b_0),\\
&b_0 = x_3 \oplus x_2, b_1 = x_2 \oplus x_1, b_2 = x_1, b_3 = 1 \oplus x_2 \oplus x_0,\\
&G(x_3, x_2, x_1, x_0) = (g_3, g_2, g_1, g_0),\\
&g_0 = x_1 \oplus x_2 \times x_0, g_1 = x_2, g_2 = x_0 \oplus x_3 \times x_2,\\
&g_3 = x_3 \oplus x_2 \times x_1 \oplus x_2 \times x_0,\\
&A(x_3, x_2, x_1, x_0) = (a_3, a_2, a_1, a_0),\\
&a_0 = 1 \oplus x_3 \oplus x_1 \oplus x_0, a_1 = x_1, a_2 = 1 \oplus x_2, a_3 = x_3 \oplus x_1.
\end{aligned} \quad (8)$$

The ANF equations for one B, two G, and one A have eight AND gates and 17 XOR gates. The ANF equations of three shares TI of S-box are shown in [15], which have 72 AND gates and 93 XOR gates. The ANF equations of two shares TI of S-box are shown in [22], which have 32 AND gates and 47 XOR gates.

Jati et al. [17] proposed a first-order TI of GIFT using a similar technique of Poschmann et al. [12] and the ANF equations of G and F are as follows:

$$\begin{aligned}
&G(x_3, x_2, x_1, x_0) = (g_3, g_2, g_1, g_0),\\
&g_0 = x_0 \oplus x_1 \oplus x_1 \times x_0 \oplus x_2 \oplus x_3, g_1 = x_1 \oplus x_2 \times x_0,\\
&g_2 = 1 \oplus x_2, g_3 = x_0 \oplus x_1 \oplus x_2 \times x_1,\\
&F(x_3, x_2, x_1, x_0) = (f_3, f_2, f_1, f_0),\\
&f_0 = 1 \oplus x_0, f_1 = x_0 \oplus x_1,\\
&f_2 = 1 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_3 \times x_0, f_3 = x_1 \times x_0 \oplus x_3.
\end{aligned} \quad (9)$$

These equations have five AND gates and 15 XOR gates. And the ANF equations of three shares TI of G and F are shown in [17–19], which have 45 AND gates and 69 XOR gates.

Shahmirzadi and Moradi proposed a lightweight two shares first-order TI of PRESENT, in which the S-box of degree 3 is used directly. And the ANF equations of the S-box are as follows:

$$\begin{aligned}
&S(x_3, x_2, x_1, x_0) = (y_3, y_2, y_1, y_0),\\
&y_0 = x_3 \oplus x_1 \oplus x_0 \oplus x_2 \times x_1,\\
&y_1 = x_2 \oplus x_0 \oplus x_2 \times x_0 \oplus x_1 \times x_0 \oplus x_3,\\
&\times x_2 \times x_1 \oplus x_3 \times x_2 \times x_0 \oplus x_3 \times x_1 \times x_0,\\
&y_2 = 1 \oplus x_1 \oplus x_0 \oplus x_3 \times x_2 \oplus x_3 \times x_0,\\
&\oplus x_2 \times x_0 \oplus x_3 \times x_2 \times x_0 \oplus x_3 \times x_1 \times x_0,\\
&y_3 = 1 \oplus x_3 \oplus x_2 \oplus x_0 \oplus x_2 \times x_1 \oplus x_3,\\
&\times x_2 \times x_1 \oplus x_3 \times x_2 \times x_0 \oplus x_3 \times x_1 \times x_0.
\end{aligned} \quad (10)$$

These equations have 23 AND gates and 23 XOR gates. And the ANF equations of two shares TI of the S-box are shown in [23], which have 200 AND gates and 164 XOR gates.

3. First-Order TI Design for Primitives

The two and three input shares of the first-order TI have their features. The implementation of three input shares TI can require computational resources to calculate multiple shares. At the same time, the two input shares may require more storage resources to register variables. For example, for a TI of AND gate, the two input shares require four registers, while the three input shares only require three registers. In this paper, we will discuss both two and three input shares of the first-order MS-LW-TI.

3.1. Decomposition of 4×4 S-Boxes. The lightweight block cipher PRESENT has become a standard algorithm in International Organization for Standardization (ISO-29192). And the GIFT is an improved version of the PRESENT. The

PICCOLO is also a well-known algorithm. Thus, we choose these algorithms as our research objects. To ensure the high security of cryptographic algorithms, the 4×4 S-boxes have a high algebraic degree for these algorithms, and their algebraic degree is 3. However, the higher algebraic degree requires more area in hardware implementation. To optimize implementation, we adopt MiniSat to reduce AND gates, OR gates, and XOR gates of an S-box based on the scheme by Stoffelen [25] at Fast Software Encryption (FSE) 2016.

The algebraic degree of 4×4 S-box in GIFT is 3. MiniSat (cryptominisat-5.8.0) decomposes the S-box into multiple logic gate functions, which include three AND gates, one OR gate, and 10 XOR gates. The logic gate functions of the S-box are as follows:

$$\begin{aligned} &S(x_3, x_2, x_1, x_0) = (y_3, y_2, y_1, y_0), \\ &t_0 = x_1 \times x_3, t_1 = x_2 \parallel x_3 \oplus 1, t_2 = t_1 \oplus x_1 \oplus 1, \\ &t_3 = t_0 \oplus x_2, y_3 = x_0 \oplus t_2 \oplus 1, y_2 = y_3 \oplus t_3 \oplus 1, \\ &t_6 = t_3 \times x_0, y_0 = x_3 \oplus t_6, t_8 = y_0 \times y_2, y_1 = t_2 \oplus t_8, \end{aligned} \tag{11}$$

where x_0, x_1, x_2, and x_3 are input variables, y_0, y_1, y_2, and y_3 are output variables, and t_0, t_1, t_2, t_3, t_6, and t_8 are intermediate variables. The symbols $\times$, $\parallel$, and $\oplus$ are AND, OR, and XOR gates, respectively. We know that the OR gate can be represented by the AND gate, and the NOT gate can be represented by the XOR gate and a constant 1. So $t_1 = x_2 \parallel x_3 \oplus 1$ can be rewritten as $t_1 = (x_2 \oplus 1) \times (x_3 \oplus 1)$.

The algebraic degree of the PRESENT S-box is also 3. Courtois et al. [26] presented the optimal logic gate functions of the S-box by MiniSat, these logic functions include two AND gates, two OR gates, and 10 XOR gates. For the logic functions of the S-box, see Appendix A.

The PICCOLO S-box is a simple nonlinear function, its algebraic degree is 3. The optimal logic gate functions of the PICCOLO S-box include four OR gates and nine XOR gates. For the logic gate functions of the PICCOLO S-box, see Appendix B.

The S-box optimizations are also applied by Bilgin et al. [27] and Cassiers et al. [28] to reduce AND depth and gate complexity, which make a low-latency hardware circuit. For the number of logic gates, their S-box optimizations are not the least. But our S-box optimization obtains the minimum number of logic gates. Our scheme is different from theirs. For example, we use the number of logic gates of a PRESENT S-box to discuss the difference. For better understanding, we make AND gate represent OR gate and XOR gate represent NOT gate. The scheme of Bilgin et al. [27] needs four AND gates and 24 XOR gates. And the scheme of Cassiers et al. [28] needs four AND gates and 20 XOR gates. Our scheme only needs four AND gates and 16 XOR gates. So our scheme is the least. The AND depth of schemes by Bilgin et al. [27] and Cassiers et al. [28] is 2, our scheme is 3, which is the largest for AND depth, but the number of logic gates is the least, which is suitable for lightweight block cipher to reduce implementation area.

3.2. TI of Primitives

3.2.1. The Basic Idea of Primitives. An S-box is represented as two logic functions after MinSat: two-input XOR gate $t = x \oplus y$ and two-input AND gate $t = x \times y$, where x and y are input variables and t is the output variable because the OR gate can be represented by AND gate. The $t = x \times y$ is a noninvertible function without satisfying uniformity. We construct an invertible function to satisfy uniformity by adding XOR a new number z. The invertible function is expressed as $t = x \times y \oplus z$. For GIFT, PRESENT, and PICCOLO 4×4 S-boxes, there are four $t = x \times y$ functions, which means they need four one-bit z to construct these invertible functions. The four one-bit numbers are the four input variables of S-boxes. For AES 8×8 S-box, there are 32 functions, which means they need 32 one-bit numbers, while the eight input variables are not enough, and they have to add 24 one-bit fresh random numbers. If we do not add the fresh random numbers in the scheme, we can consider merging the 32 functions, and a like way is described in detail in [23]. We will not discuss it in this paper.

3.2.2. The Establishment of Primitives. We design the first-order TI of two and three shares corresponding to $(d+1)$ and $(2d+1)$, respectively.

Linear logic gate function: $t = x \oplus y$.

Nonlinear logic gate function: $t = x \times y \oplus z$.

Thus, the two logic gate functions $t = x \oplus y$ and $t = x \times y \oplus z$ are defined as two basic primitives of MS-LW-TI.

Two basic primitives of two shares are described as follows: The x_0 and x_1 are shares of x, and the y_0 and y_1 are shares of y. z is the third variable, whose shares are z_0 and z_1:

$$x = x_0 \oplus x_1, y = y_0 \oplus y_1, z = z_0 \oplus z_1. \tag{12}$$

(1) Linear Transformation: $t = x \oplus y$. The first-order TI with two shares is as follows:

$$t_0 = x_1 \oplus y_1, t_1 = x_0 \oplus y_0. \tag{13}$$

(2) Nonlinear Transformation: $t = x \times y \oplus z$. The first-order TI with two shares is as follows:

$$\begin{aligned} &x_0 \times y_0 \oplus z_0 \longrightarrow t'_0, \\ &x_0 \times y_1 \longrightarrow t'_1, \\ &x_1 \times y_0 \longrightarrow t'_2, \\ &x_1 \times y_1 \oplus z_1 \longrightarrow t'_3, \\ &t_0 = t'_0 \oplus t'_1, t_1 = t'_2 \oplus t'_3, \end{aligned} \tag{14}$$

where t'_0, t'_1, t'_2 and t'_3 are register variables, t_0 and t_1 are output shares. The operates of $t'_0 \oplus t'_1 = t_0$ and $t'_2 \oplus t'_3 = t_1$ are defined as compression layers.

Two basic primitives of three shares are described as follows: The x_0, x_1, and x_2 are shares of x, and the y_0, y_1, and y_2 are shares of y. z is the third variable, whose shares are z_0, z_1, and z_2.

$$x = x_0 \oplus x_1 \oplus x_2, y = y_0 \oplus y_1 \oplus y_2, z = z_0 \oplus z_1 \oplus z_2. \tag{15}$$

(3) Linear Transformation: $t = x \oplus y$. The first-order TI with three shares is as follows:

$$t_0 = x_1 \oplus y_1, t_1 = x_2 \oplus y_2, t_2 = x_0 \oplus y_0. \tag{16}$$

(4) Nonlinear Transformation: $t = x \times y \oplus z$. The first-order TI with three shares is as follows:

$$\begin{aligned} t_0 &= x_1 \times y_1 \oplus x_1 \times y_2 \oplus x_2 \times y_1 \oplus z_1, \\ t_1 &= x_2 \times y_2 \oplus x_2 \times y_0 \oplus x_0 \times y_2 \oplus z_2, \\ t_2 &= x_0 \times y_0 \oplus x_0 \times y_1 \oplus x_1 \times y_0 \oplus z_0. \end{aligned} \tag{17}$$

where t_0, t_1, and t_2 are output shares.

Based on the two basic primitives, we can extend to logical gate functions such as $t = x \times y \oplus x \oplus y \oplus z$, $t = x \oplus y \oplus 1$, and $t = x \times y \oplus x \oplus y \oplus z \oplus 1$.

3.2.3. Secure Properties of Primitives. We prove the security properties of the two primitives for $(d+1)$ and $(2d+1)$ boundaries of the first-order TI.

Proposition 1. *The primitive $t = x \oplus y$ has correctness, noncompleteness, and uniformity of TI.*

Proof. (1) Two shares of the scheme: In Equation (13), the original output is equal to the XOR of output shares, i.e.:

$$t = t_0 \oplus t_1 = x_0 \oplus x_1 \oplus y_0 \oplus y_1 = x \oplus y. \tag{18}$$

It satisfies the correctness.

In Equation (13), t_0 is independent of x_1 and y_1, and t_1 is independent of x_0 and y_0. It satisfies the noncompleteness.

The probabilistic distributions of input and output shares are denoted as $\Pr(\bar{x}, \bar{y})$ and $\Pr(\bar{t})$, respectively. The probability of input shares is described as follows:

$$\Pr(\bar{x}, \bar{y}) = Q^{1-S} \Pr(x = x_0 \oplus x_1, y = y_0 \oplus y_1), \tag{19}$$

where the number of different (x, y) values Q is 4 and the number of share S is 2.

$$\begin{aligned} &\Pr(\bar{x}, \bar{y}) = \frac{1}{4 \times 4}, \Pr(x = x_0 \oplus x_1, y = y_0 \oplus y_1) = \frac{1}{4}, \\ &\Pr(\bar{x}, \bar{y}) = Q^{1-S} \Pr(x = x_0 \oplus x_1, y = y_0 \oplus y_1) \\ &\longrightarrow \frac{1}{4 \times 4} = 4^{1-2} \times \left(\frac{1}{4}\right). \end{aligned} \tag{20}$$

According to Equation (6), it satisfies the uniformity.

Based on the uniformity of inputs, the probability of shared outputs is described as follows:

$$\Pr(\bar{t}) = Q^{1-S} \Pr(t = t_0 \oplus t_1), \tag{21}$$

where the number of different t values Q is 2, the number of share S is 2.

$$\begin{aligned} &\Pr(\bar{t}) = \frac{1}{2 \times 2}, \Pr(t = t_0 \oplus t_1) = \frac{1}{2}, \\ &\Pr(\bar{t}) = Q^{1-S} \Pr(t = t_0 \oplus t_1) \longrightarrow \frac{1}{2 \times 2} = 2^{1-2} \times \left(\frac{1}{2}\right). \end{aligned} \tag{22}$$

According to Equation (6), the output of primitive satisfies the uniformity. Under the condition of uniformity of shared inputs, the shared outputs are uniform, which meets the uniformity.

(2) Three shares of the scheme: The proof process of Equation (16) is the same as Equation (13), similarly, $t = x \oplus y$ satisfies correctness, noncompleteness, and uniformity of TI with three shares. □

Proposition 2. *The primitive $t = x \times y \oplus z$ also has correctness, noncompleteness, and uniformity of TI.*

Proof. (1) Two shares of the scheme: In Equation (14), the original output is equal to the XOR of output shares, i.e.:

$$\begin{aligned} t = t_0 \oplus t_1 = (x_0 \times y_0 \oplus z_0) \oplus (x_0 \times y_1) \oplus \\ (x_1 \times y_0) \oplus (x_1 \times y_1 \oplus z_1) = x \times y \oplus z. \end{aligned} \tag{23}$$

It satisfies the correctness. In Equation (14), there are no two shares of an input variable that appears in registers t_0', t_1', t_2', and t_3' at the same time. Thus, it satisfies the noncompleteness.

The primitive $t = x \times y \oplus z$ is an invertible function, and the probabilistic distribution shared inputs and outputs are denoted by $\Pr(\bar{x}, \bar{y}, \bar{z})$ and $\Pr(\bar{t})$, respectively. The uniformity of shared inputs is described as follows:

$$\begin{aligned} &\Pr(\bar{x}, \bar{y}, \bar{z}) = Q^{1-S} \Pr(x = x_0 \oplus x_1, y = y_0 \oplus y_1, z = z_0 \oplus z_1) \\ &\longrightarrow \frac{1}{8 \times 8} = 8^{1-2} \times \left(\frac{1}{8}\right), \end{aligned} \tag{24}$$

where the number of different (x, y, z) values Q is 8 and the number of share S is 2. According to Equation (6), the input shares satisfy the uniformity.

Under the uniformity of inputs, the uniformity of shared outputs is described as follows:

$$\Pr(\bar{t}) = Q^{1-S} \Pr(t = t_0 \oplus t_1) \longrightarrow \frac{1}{2 \times 2} = 2^{1-2} \times \left(\frac{1}{2}\right), \tag{25}$$

where the number of different t values Q is 2 and the number of share S is 2. According to Equation (6), the output of primitive satisfies the uniformity.

(2) Three shares of the scheme: The proof process of Equation (17) is the same as Equation (14), similarly, $t = x \times y \oplus z$ satisfies correctness, noncompleteness, and uniformity of TI with three shares. □

Based on primitives $t = x \oplus y$ and $t = x \times y \oplus z$, we can extend to logic gate functions that also satisfy the correctness, noncompleteness, and uniformity.

Property 1: The function $t = x \oplus y \oplus 1$ has correctness, noncompleteness, and uniformity.

Proof. According to Proposition 1, the primitive $x \oplus y$ satisfies three secure properties of TI. If we regard f as a new variable to replace $x \oplus y$, then $t = x \oplus y \oplus 1$ can be transformed into $t = f \oplus 1$, which is equivalent to the inverse of f. Thus, the function $t = x \oplus y \oplus 1$ has correctness, noncompleteness, and uniformity. □

Property 2: The function $t = x \times y \oplus x \oplus y \oplus z$ has correctness, noncompleteness, and uniformity.

Proof. According to Proposition 2, the primitive $x \times y \oplus z$ satisfies three secure properties of TI. Meanwhile, according to Proposition 1, the primitive $x \oplus y$ satisfies three secure properties of TI. If we regard f as a new variable to replace $x \times y \oplus z$ and g to replace $x \oplus y$, then $t = x \times y \oplus x \oplus y \oplus z$ is transformed into $t = f \oplus g$, which meets Proposition 1. Thus, the function $t = x \times y \oplus x \oplus y \oplus z$ satisfies correctness, noncompleteness, and uniformity. □

Property 3: The function $t = x \times y \oplus x \oplus y \oplus z \oplus 1$ has correctness, noncompleteness, and uniformity.

The proof process of Property 3 is the same as Property 1 and Property 2.

4. MS-LW-TI of 4 × 4 S-Boxes

4.1. Design of MS-LW-TI. We take the S-box of GIFT as an example to introduce the specific construction and implementation of the MS-LW-TI. An S-box is solved by MiniSat to obtain two-input logic gates, $t = x \oplus y$ and $t = x \times y$. After the two logic gates are constructed, the following five logic gate functions are summarized: $t = x \oplus y$, $t = x \times y \oplus z$, $t = x \times y \oplus x \oplus y \oplus z$, $t = x \oplus y \oplus 1$, and $t = x \times y \oplus x \oplus y \oplus z \oplus 1$. We describe the implementation of the five logic gate functions:

(1) $t = x \oplus y$ and $t = x \times y \oplus z$.

The logic gate functions $t = x \oplus y$ and $t = x \times y \oplus z$ are the two basic primitives, which have been introduced in Section 3.2.

(2) $t = x \times y \oplus x \oplus y \oplus z$.

Two shares of the scheme: x_0, x_1, y_0, y_1, z_0, and z_1 are x, y, and z input shares, respectively:

$$\begin{aligned} &x_0 \times y_0 \oplus x_0 \oplus y_0 \oplus z_0 \longrightarrow t_0', \\ &x_0 \times y_1 \longrightarrow t_1', \\ &x_1 \times y_0 \longrightarrow t_2', \\ &x_1 \times y_1 \oplus x_1 \oplus y_1 \oplus z_1 \longrightarrow t_3', \\ &t_0 = t_0' \oplus t_1', t_1 = t_2' \oplus t_3', \end{aligned} \tag{26}$$

where t_0 and t_1 are output shares.

Three shares of the scheme: $x_0, x_1, x_2, y_0, y_1, y_2, z_0, z_1$, and z_2 are x, y, and z input shares, respectively:

$$\begin{aligned} t_0 &= x_1 \times y_1 \oplus x_1 \times y_2 \oplus x_2 \times y_1 \oplus x_1 \oplus y_1 \oplus z_1, \\ t_1 &= x_2 \times y_2 \oplus x_2 \times y_0 \oplus x_0 \times y_2 \oplus x_2 \oplus y_2 \oplus z_2, \\ t_2 &= x_0 \times y_0 \oplus x_0 \times y_1 \oplus x_1 \times y_0 \oplus x_0 \oplus y_0 \oplus z_0, \end{aligned} \tag{27}$$

where t_0, t_1, and t_2 are output shares.

(3) $t = x \oplus y \oplus 1$.

Two shares of the scheme: x_0, x_1, y_0, and y_1 are x and y input shares, respectively:

$$t_0 = x_1 \oplus y_1 \oplus 1, t_1 = x_0 \oplus y_0. \tag{28}$$

Three shares of the scheme: x_0, x_1, x_2, y_0, y_1, and y_2 are x and y input shares, respectively:

$$t_0 = x_1 \oplus y_1 \oplus 1, t_1 = x_2 \oplus y_2, t_2 = x_0 \oplus y_0. \tag{29}$$

(4) $t = x \times y \oplus x \oplus y \oplus z \oplus 1$.

According to the design of $t = x \times y \oplus x \oplus y \oplus z$ and $t = x \oplus y \oplus 1$, we can implement $t = x \times y \oplus x \oplus y \oplus z \oplus 1$.

After MiniSat, the S-box of GIFT is decomposed as 14 logic gates. And we construct the primitives and extensions from the 14 logic gates. These primitives and extensions have a cascaded and parallel relationship with each other. To disallow the propagation of glitches in cascaded logic functions that include AND logic gate, we need to insert several registers among these functions. For example, $t = x \times y \oplus z$, $t = x \times y \oplus x \oplus y \oplus z$, and $t = x \times y \oplus x \oplus y \oplus z \oplus 1$. The implementation process of all the logic functions of the GIFT S-box is shown below.

The x_0, x_1, x_2, and x_3 are input variables, the y_0, y_1, y_2, and y_3 are output variables, and the t_2 and t_3 are intermediate variables:

$$\begin{aligned} &S(x_3, x_2, x_1, x_0) = (y_3, y_2, y_1, y_0), \\ &t_2 = x_2 \times x_3 \oplus x_2 \oplus x_3 \oplus x_1, t_3 = x_1 \times x_3 \oplus x_2, \\ &y_3 = x_0 \oplus t_2 \oplus 1, y_2 = y_3 \oplus t_3 \oplus 1, \\ &y_0 = x_0 \times t_3 \oplus x_3, y_1 = y_0 \times y_2 \oplus t_2. \end{aligned} \tag{30}$$

For the GIFT S-box, the design architectures of two shares and three shares of MS-LW-TI are shown in Figures 1 and 2, respectively. In Figure 1, the compress functions compress logic functions to $(d+1)$ output shares by means of XORs. And the ANF equations of two shares and three shares of the MS-LW-TI are listed in Appendices C and D, respectively.

At the same time, the implementation process of the logic gate functions of PRESENT S-box and PICCOLO S-box are listed in Appendices E and F, respectively.

4.2. Security Analysis of MS-LW-TI. Theoretically, we have proved that the five logic gate functions satisfy the basic properties of TI. These functions can be considered local

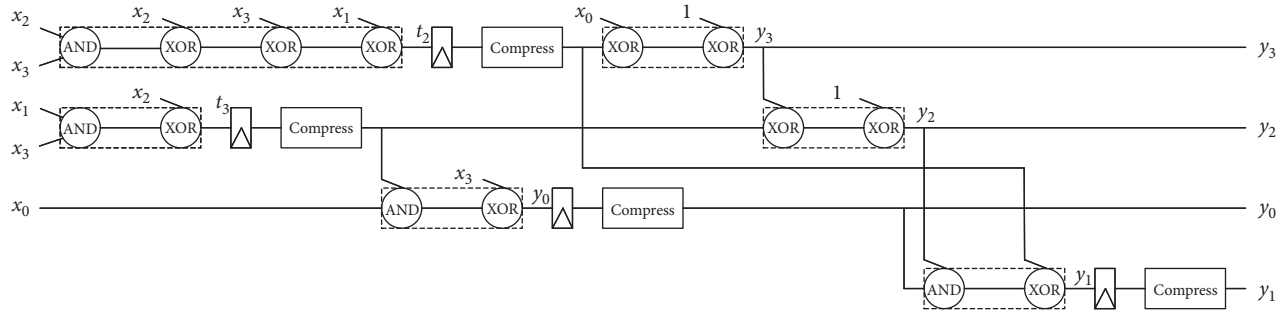

FIGURE 1: MS-LW-TI of GIFT S-box with two shares.

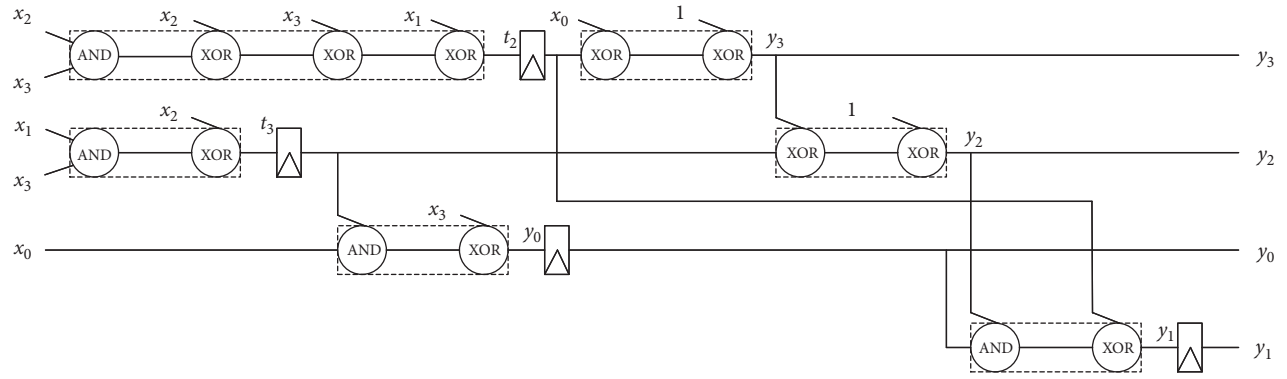

FIGURE 2: MS-LW-TI of GIFT S-box with three shares.

security, but the security of the overall S-box composed of these functions is unknown in implementation. Because there is a cascaded and parallel relationship between these functions, which may lead to the S-box failing to meet the basic properties of TI. The security definitions of cascaded and parallel functions are described by Bilgin et al. [29] and Dhooghe et al. [30]. It shows that let $y_1 = f(x)$ and $y_2 = g(x)$ be two functions with the same uniform input x. If $y_1 = f(x)$ and $y_2 = g(x)$ are parallel, each satisfies the basic properties of TI. Then TI of $y_1 = f(x)$ and $y_2 = g(x)$ cannot lead to information leakage. If the outputs of $y_1 = f(x)$ and $y_2 = g(x)$ are the inputs of $k = h(y_1, y_2)$, which are cascaded. In addition to satisfying the properties of TI, the joint distribution of the y_1 and y_2 is uniform, then TI of $k = h(y_1, y_2)$ is safe.

Based on the above definition, we have verified the security of MS-LW-TI. Here, we selected the GIFT S-box to discuss, PRESENT S-box, and PICCOLO S-box can also be verified in the same way. We will not describe them in detail.

First, for the correctness of the scheme, we can verify the output shares according to the input shares of the shared S-box, and logic gate functions satisfy the correctness of MS-LW-TI, then the output results of the shared S-box are equal to the original S-box. Second, we verify the noncompleteness of MS-LW-TI. The x_0, x_1, x_2, and x_3 are input variables of the S-box, and the shares of each input variable are operated separately and do not appear at the same time in the shared function circuits. So, placing a probe on each shared function circuit does not reveal any information.

Finally, we verify the important uniformity of MS-LW-TI. The first-order probing secure implementation of $t = x \times y \oplus z$ can be easily achieved with z to replace the fresh mask bit [23]. For the GIFT S-box, two shares of the scheme, the variables $x_{(1,0)}$, $x_{(1,1)}$, $x_{(2,0)}$, $x_{(2,1)}$, $x_{(3,0)}$, $x_{(3,1)}$, $t_{(2,0)}$, and $t_{(2,1)}$ can act as the fresh mask bit in the logic functions t_2, t_3, y_0, and y_1, respectively. Meanwhile, three shares of the scheme, the variables $x_{(1,0)}$, $x_{(1,1)}$, $x_{(1,2)}$, $x_{(2,0)}$, $x_{(2,1)}$, $x_{(2,2)}$, $x_{(3,0)}$, $x_{(3,1)}$, $x_{(3,2)}$, $t_{(2,0)}$, $t_{(2,1)}$, and $t_{(2,2)}$ can act as the fresh mask bit in the logic functions t_2, t_3, y_0, and y_1, respectively.

Two shares of the scheme: Among these functions, t_2, t_3, y_0, and y_1 containing AND logic gates, sequential circuit implementation is performed to block glitches using registers. Thus, we just need to consider the uniformity of input share and output share for each function. All possible input values of shares $x_{(0,0)}$, $x_{(0,1)}$, $x_{(1,0)}$, $x_{(1,1)}$, $x_{(2,0)}$, $x_{(2,1)}$, $x_{(3,0)}$, and $x_{(3,1)}$ for the whole S-box are $2^8 = 256$. $x_{(1,0)}$, $x_{(1,1)}$, $x_{(2,0)}$, $x_{(2,1)}$, $x_{(3,0)}$, and $x_{(3,1)}$ are the input shares of t_2 and t_3 functions, $t_{(2,0)}$ and $t_{(2,1)}$ are the output shares of t_2, and $t_{(3,0)}$ and $t_{(3,1)}$ are the output shares of t_3. When the output shares of one function are used as the input shares of another function, $t_{(3,0)}$, $t_{(3,1)}$, $x_{(0,0)}$, $x_{(0,1)}$, $x_{(3,0)}$, and $x_{(3,1)}$ are the input shares of y_0, and $y_{(0,0)}$ and $y_{(0,1)}$ are the output shares of y_0. $t_{(2,0)}$, $t_{(2,1)}$, $y_{(0,0)}$, $y_{(0,1)}$, $y_{(2,0)}$, and $y_{(2,1)}$ are the input shares of y_1, and $y_{(1,0)}$ and $y_{(1,1)}$ are the output shares of y_1. The input and output share distributions of these functions are shown in Table 1. Looking at the distributions of the shares in Table 1, the input and output shares of t_2, t_3, y_0, and y_1 satisfy uniformity.

Table 1: Number of times of input shares and output shares for two shares of the functions.

(Input shares) $(x_{(3,1)}, x_{(3,0)}, x_{(2,1)}, x_{(2,0)}, x_{(1,1)}, x_{(1,0)})$ $(t_{(3,1)}, t_{(3,0)}, x_{(3,1)}, x_{(3,0)}, x_{(0,1)}, x_{(0,0)})$ $(t_{(2,1)}, t_{(2,0)}, y_{(2,1)}, y_{(2,0)}, y_{(0,1)}, y_{(0,0)})$	(Numbers)	(Input shares) $(x_{(3,1)}, x_{(3,0)}, x_{(2,1)}, x_{(2,0)}, x_{(1,1)}, x_{(1,0)})$ $(t_{(3,1)}, t_{(3,0)}, x_{(3,1)}, x_{(3,0)}, x_{(0,1)}, x_{(0,0)})$ $(t_{(2,1)}, t_{(2,0)}, y_{(2,1)}, y_{(2,0)}, y_{(0,1)}, y_{(0,0)})$	(Numbers)
(0,0,0,0,0,0)	4	(0,0,0,0,0,1)	4
(1,1,0,0,0,0)	4	(1,1,0,0,0,1)	4
(0,0,1,1,0,0)	4	(0,0,1,1,0,1)	4
(1,1,1,1,0,0)	4	(1,1,1,1,0,1)	4
(0,0,0,0,1,1)	4	(0,0,0,0,1,0)	4
(1,1,0,0,1,1)	4	(1,1,0,0,1,0)	4
(0,0,1,1,1,1)	4	(0,0,1,1,1,0)	4
(1,1,1,1,1,1)	4	(1,1,1,1,1,0)	4
(0,1,0,0,0,0)	4	(0,1,0,0,0,1)	4
(1,0,0,0,0,0)	4	(1,0,0,0,0,1)	4
(0,1,1,1,0,0)	4	(0,1,1,1,0,1)	4
(1,0,1,1,0,0)	4	(1,0,1,1,0,1)	4
(0,1,0,0,1,1)	4	(0,1,0,0,1,0)	4
(1,0,0,0,1,1)	4	(1,0,0,0,1,0)	4
(0,1,1,1,1,1)	4	(0,1,1,1,1,0)	4
(1,0,1,1,1,1)	4	(1,0,1,1,1,0)	4
(0,0,0,1,0,0)	4	(0,0,0,1,0,1)	4
(1,1,0,1,0,0)	4	(1,1,0,1,0,1)	4
(0,0,1,0,0,0)	4	(0,0,1,0,0,1)	4
(1,1,1,0,0,0)	4	(1,1,1,0,0,1)	4
(0,0,0,1,1,1)	4	(0,0,0,1,1,0)	4
(1,1,0,1,1,1)	4	(1,1,0,1,1,0)	4
(0,0,1,0,1,1)	4	(0,0,1,0,1,0)	4
(1,1,1,0,1,1)	4	(1,1,1,0,1,0)	4
(0,1,0,1,0,0)	4	(0,1,0,1,0,1)	4
(1,0,0,1,0,0)	4	(1,0,0,1,0,1)	4
(0,1,1,0,0,0)	4	(0,1,1,0,0,1)	4
(1,0,1,0,0,0)	4	(1,0,1,0,0,1)	4
(0,1,0,1,1,1)	4	(0,1,0,1,1,0)	4
(1,0,0,1,1,1)	4	(1,0,0,1,1,0)	4
(0,1,1,0,1,1)	4	(0,1,1,0,1,0)	4
(1,0,1,0,1,1)	4	(1,0,1,0,1,0)	4
(Output shares) $(t_{(2,1)}, t_{(2,0)})$ $(t_{(3,1)}, t_{(3,0)})$ $(y_{(0,1)}, y_{(0,0)})$ $(y_{(1,1)}, y_{(1,0)})$	(Numbers)	(Output shares) $(t_{(2,1)}, t_{(2,0)})$ $(t_{(3,1)}, t_{(3,0)})$ $(y_{(0,1)}, y_{(0,0)})$ $(y_{(1,1)}, y_{(1,0)})$	(Numbers)
(0,0)	64	(0,1)	64
(1,1)	64	(1,0)	64

We also need to consider glitches for the combinational circuit implementation of the compress and y_2 functions. Among compress functions, the joint probability distributions of the output data $(t'_{(2,0)}, t'_{(2,1)})$ are equal, the number of occurrences of (0,0), (0,1), (1,0), and (1,1) are 96, 32, 96, and 32, respectively, and the joint probability distributions of the output data $(t'_{(3,0)}, t'_{(3,1)})$, $(y'_{(0,0)}, y'_{(0,1)})$, and $(y'_{(1,0)}, y'_{(1,1)})$ have the same situation. Meanwhile, the joint probability distributions of the output data $(t'_{(2,2)}, t'_{(2,3)})$ are also equal, the number of occurrences of (0,0), (0,1), (1,0), and (1,1) are 96, 96, 32, and 32, respectively, and the joint probability distributions of the output data $(t'_{(3,2)}, t'_{(3,3)})$, $(y'_{(0,2)}, y'_{(0,3)})$, and $(y'_{(1,2)}, y'_{(1,3)})$ also have the same situation. Thus, it is shown that the output of the compress function is independent of the input data and satisfies the first-order glitch-extended probing secure.

For y_2 function, the joint probability distributions of the output data $(t'_{(2,0)}, t'_{(2,1)}, t'_{(3,0)}, t'_{(3,1)})$ are equal, the number of

Table 2: Number of times that output shares occur for all possible two input shares.

Input shares	$(y_{0,0}, y_{0,1})$				$(y_{1,0}, y_{1,1})$				$(y_{2,0}, y_{2,1})$				$(y_{3,0}, y_{3,1})$			
(x_0, x_1, x_2, x_3)	(0,0)	(0,1)	(1,0)	(1,1)	(0,0)	(0,1)	(1,0)	(1,1)	(0,0)	(0,1)	(1,0)	(1,1)	(0,0)	(0,1)	(1,0)	(1,1)
(0,0,0,0)	8	0	0	8	8	0	0	8	8	0	0	8	0	8	8	0
(0,0,0,1)	0	8	8	0	8	0	0	8	0	8	8	0	8	0	0	8
(0,0,1,0)	8	0	0	8	0	8	8	0	8	0	0	8	8	0	0	8
(0,0,1,1)	0	8	8	0	0	8	8	0	8	0	0	8	8	0	0	8
(0,1,0,0)	8	0	0	8	0	8	8	0	0	8	8	0	8	0	0	8
(0,1,0,1)	0	8	8	0	0	8	8	0	0	8	8	0	0	8	8	0
(0,1,1,0)	8	0	0	8	8	0	0	8	0	8	8	0	0	8	8	0
(0,1,1,1)	0	8	8	0	8	0	0	8	8	0	0	8	0	8	8	0
(1,0,0,0)	8	0	0	8	8	0	0	8	0	8	8	0	8	0	0	8
(1,0,0,1)	0	8	8	0	0	8	8	0	8	0	0	8	0	8	8	0
(1,0,1,0)	0	8	8	0	8	0	0	8	0	8	8	0	0	8	8	0
(1,0,1,1)	8	0	0	8	0	8	8	0	0	8	8	0	0	8	8	0
(1,1,0,0)	8	0	0	8	0	8	8	0	8	0	0	8	0	8	8	0
(1,1,0,1)	8	0	0	8	8	0	0	8	8	0	0	8	8	0	0	8
(1,1,1,0)	0	8	8	0	8	0	0	8	8	0	0	8	8	0	0	8
(1,1,1,1)	0	8	8	0	0	8	8	0	0	8	8	0	8	0	0	8

occurrences of (0,0,0,0), (0,0,0,1), (0,0,1,0), (0,0,1,1), (0,1,0,0), (0,1,0,1), (0,1,1,0), (0,1,1,1), (1,0,0,0), (1,0,0,1), (1,0,1,0), (1,0,1,1), (1,1,0,0), (1,1,0,1), (1,1,1,0), and (1,1,1,1) are 48, 0, 32, 16, 0, 16, 0, 16, 48, 16, 32, 0, 0, 0, 32, and 0, respectively. The joint probability distributions of the output data $(t'_{(2,2)}, t'_{(2,3)}, t'_{(3,2)}, t'_{(3,3)})$ are also equal, the number of occurrences of (0,0,0,0), (0,0,0,1), (0,0,1,0), (0,0,1,1), (0,1,0,0), (0,1,0,1), (0,1,1,0), (0,1,1,1), (1,0,0,0), (1,0,0,1), (1,0,1,0), (1,0,1,1), (1,1,0,0), (1,1,0,1), (1,1,1,0), and (1,1,1,1) are 48, 32, 0, 16, 48, 32, 16, 0, 0, 0, 16, 16, 0, 32, 0, and 0 respectively. Thus, it is shown that the output of the y_2 function is also independent of the input data and satisfies the first-order glitch-extended probing secure.

Three shares of the scheme: This scheme is like two shares of the scheme where we need to consider the uniformity of input share and output share for t_2, t_3, y_0, and y_1 in a sequential circuit. All possible input values of shares $x_{(0,0)}$, $x_{(0,1)}$, $x_{(0,2)}$, $x_{(1,0)}$, $x_{(1,1)}$, $x_{(1,2)}$, $x_{(2,0)}$, $x_{(2,1)}$, $x_{(2,2)}$, $x_{(3,0)}$, $x_{(3,1)}$, and $x_{(3,2)}$ for whole S-box are $2^{12} = 4{,}096$. $x_{(1,0)}$, $x_{(1,1)}$, $x_{(1,2)}$, $x_{(2,0)}$, $x_{(2,1)}$, $x_{(2,2)}$, $x_{(3,0)}$, $x_{(3,1)}$, and $x_{(3,2)}$ are the input shares of t_2 function whose distribution contains 512 different cases, each occurring eight times. And $t_{(2,0)}$, $t_{(2,1)}$, and $t_{(2,2)}$ are the output shares of t_2 function whose distribution contains eight different cases, each of which occurs 512 times. Similarly, the t_3, y_0, and y_1 have the same input and output distributions as the t_2. Thus, the input and output shares of t_2, t_3, y_0, and y_1 also satisfy uniformity. In combinational circuit, for the y_2 function, the joint probability distributions of the output shares $(t_{(2,0)}, t_{(2,1)}, t_{(2,2)}, t_{(3,0)}, t_{(3,1)}, t_{(3,2)})$ are 64 different cases, each occurring 64 times. Therefore, the joint probability distributions of the output shares are equal, the y_2 function also satisfies the first-order glitch-extended probing secure.

We also verify that the output shares uniformity of the whole S-box for all possible input shares is based on the definition in [29]. Uniform sharing of a function (circuit): The sharing $\tilde{f}$ is uniform if and only if:

$\forall x \in \mathcal{F}^m$, $\forall y \in \mathcal{F}^n$ with $f(x) = y$, $\forall \tilde{y} \in \text{sharing}(y)$:

$|\{\tilde{x} \in \text{sharing}(x) | \tilde{f}(\tilde{x}) = \tilde{y}\}| = \frac{|\mathcal{F}|^{m\times(S_x-1)}}{|\mathcal{F}|^{n\times(S_y-1)}}$,

where the m and n are the numbers of input variables and output variables, respectively. Meanwhile, the S_x and the S_y are the number of input shares and output shares, respectively.

For two shares of S-box of GIFT, $m = 4$, $n = 1$, $S_x = 2$, and $S_y = 2$, $|\{\tilde{x} \in \text{sharing}(x) | \tilde{f}(\tilde{x}) = \tilde{y}\}| = 8$. For three shares of S-box of GIFT, we also verify the uniformity of each output. The $m = 4$, $n = 1$, $S_x = 3$, and $S_y = 3$, then $|\{\tilde{x} \in \text{sharing}(x) | \tilde{f}(\tilde{x}) = \tilde{y}\}| = 64$. The distribution of output shares of the S-box shows in Tables 2 and 3 for all possible input shares. From the data in Tables 2 and 3, the distribution of output shares of shared S-box is uniform. Thus, MS-LW-T satisfies the first-order glitch-extended probing secure.

4.3. Hardware-Resources Consideration. Logic circuits are composed of logic gates, and the number of logic gates determines the size of a logic circuit [12]. In Section 2.2, it is mentioned that the implementation of an S-box can have different logic functions, and different logic functions have different numbers of logic gates. Shahmirzadi and Moradi directly used the cubic S-box, and we found a higher number of AND gates and XOR gates for the implementation because the ANF equations contain many redundant logic units, for example, the $x_2 \times x_1$ operation and $x_1 \oplus x_0$ operation in the y_0 function, which also appear in the y_1 function and y_2 function, respectively. Poschmann et al. [12] decomposed the S-box into G and F to reduce the algebraic degree of the S-box and also reduce the number of AND logic gates, but there are still some redundant logic units in ANF equations of G and F, for example, $x_2 \oplus x_1$ operation in the g_2 function and the g_3 function. And Kutzner et al. [15] decomposed the

Table 3: Number of times that output shares occur for all possible three input shares.

Input shares	$(y_{0,0}, y_{0,1}, y_{0,2})$								$(y_{1,0}, y_{1,1}, y_{1,2})$							
(x_0, x_1, x_2, x_3)	0,0,0	0,0,1	0,1,0	0,1,1	1,0,0	1,0,1	1,1,0	1,1,1	0,0,0	0,0,1	0,1,0	0,1,1	1,0,0	1,0,1	1,1,0	1,1,1
0,0,0,0	64	0	0	64	0	64	64	0	64	0	0	64	0	64	64	0
0,0,0,1	0	64	64	0	64	0	0	64	64	0	0	64	0	64	64	0
0,0,1,0	64	0	0	64	0	64	64	0	0	64	64	0	64	0	0	64
0,0,1,1	0	64	64	0	64	0	0	64	0	64	64	0	64	0	0	64
0,1,0,0	64	0	0	64	0	64	64	0	0	64	64	0	64	0	0	64
0,1,0,1	0	64	64	0	64	0	0	64	0	64	64	0	64	0	0	64
0,1,1,0	64	0	0	64	0	64	64	0	64	0	0	64	0	64	64	0
0,1,1,1	0	64	64	0	64	0	0	64	64	0	0	64	0	64	64	0
1,0,0,0	64	0	0	64	0	64	64	0	64	0	0	64	0	64	64	0
1,0,0,1	0	64	64	0	64	0	0	64	0	64	64	0	64	0	0	64
1,0,1,0	0	64	64	0	64	0	0	64	64	0	0	64	0	64	64	0
1,0,1,1	64	0	0	64	0	64	64	0	0	64	64	0	64	0	0	64
1,1,0,0	64	0	0	64	0	64	64	0	0	64	64	0	64	0	0	64
1,1,0,1	64	0	0	64	0	64	64	0	64	0	0	64	0	64	64	0
1,1,1,0	0	64	64	0	64	0	0	64	64	0	0	64	0	64	64	0
1,1,1,1	0	64	64	0	64	0	0	64	0	64	64	0	64	0	0	64
	$(y_{2,0}, y_{2,1}, y_{2,2})$								$(y_{3,0}, y_{3,1}, y_{3,2})$							
(x_0, x_1, x_2, x_3)	0,0,0	0,0,1	0,1,0	0,1,1	1,0,0	1,0,1	1,1,0	1,1,1	0,0,0	0,0,1	0,1,0	0,1,1	1,0,0	1,0,1	1,1,0	1,1,1
0,0,0,0	64	0	0	64	0	64	64	0	0	64	64	0	64	0	0	64
0,0,0,1	0	64	64	0	64	0	0	64	64	0	0	64	0	64	64	0
0,0,1,0	64	0	0	64	0	64	64	0	64	0	0	64	0	64	64	0
0,0,1,1	64	0	0	64	0	64	64	0	64	0	0	64	0	64	64	0
0,1,0,0	0	64	64	0	64	0	0	64	64	0	0	64	0	64	64	0
0,1,0,1	0	64	64	0	64	0	0	64	0	64	64	0	64	0	0	64
0,1,1,0	0	64	64	0	64	0	0	64	0	64	64	0	64	0	0	64
0,1,1,1	64	0	0	64	0	64	64	0	0	64	64	0	64	0	0	64
1,0,0,0	0	64	64	0	64	0	0	64	64	0	0	64	0	64	64	0
1,0,0,1	64	0	0	64	0	64	64	0	0	64	64	0	64	0	0	64
1,0,1,0	0	64	64	0	64	0	0	64	0	64	64	0	64	0	0	64
1,0,1,1	0	64	64	0	64	0	0	64	0	64	64	0	64	0	0	64
1,1,0,0	64	0	0	64	0	64	64	0	0	64	64	0	64	0	0	64
1,1,0,1	64	0	0	64	0	64	64	0	64	0	0	64	0	64	64	0
1,1,1,0	64	0	0	64	0	64	64	0	64	0	0	64	0	64	64	0
1,1,1,1	0	64	64	0	64	0	0	64	64	0	0	64	0	64	64	0

S-box into two identical G functions. Jati et al. [17] decomposed the S-box into G and F. The ANF equations of G and F also have redundant logic units, for example, $x_0 \oplus x_1$ operation in the g_0 function and the g_3 function. We use MiniSat to decompose the S-box, which also reduces the algebraic degree of the S-box, while the ANF equations do not contain redundant logic units, thus minimizing the number of logic gates, as shown in Section 3.1. When a TI of S-box is implemented, the functions in the ANF equations are split into several shared functions independent of each other, and then the output of these shared functions is calculated together to get the output of the original functions, and the realization is always required to satisfy the three secure properties of TI [13]. MS-LW-TI constructs the primitives to implement each function from the three secure properties of TI. Meanwhile, MS-LW-TI can resist first-order glitch-extended probing attacks like the above schemes. When the ANF equations of the S-box require fewer logic gates, the number of logic gates required for its TI will also be reduced, thus reducing the overall protection resources [12]. We count the number of AND gates and XOR gates in the ANF equations of S-boxes and the ANF equations of TI of S-boxes, respectively. The comparison results between MS-LW-TI and existing TI are shown in Table 4. From the results in Table 4, we find that MS-LW-TI has the least number of AND gates and XOR gates.

4.4. Experiments of MS-LW-TI

4.4.1. Evaluation of MS-LW-TI. We have proved that MS-LW-TI satisfies the first-order glitch-extended probing secure. To evaluate the information leakage of the design implementation, we implement our scheme on the Spartan-6 FPGA of the

Table 4: Total number of AND and XOR gates in MS-LW-TI and the existing TIs.

Design	Number of shares	ANF for S-box		ANF for TI of S-box		Registers
		AND gates	XOR gates	AND gates	XOR gates	
PRESENT [23]	2	23	23	200	164	28
PRESENT [22]	2	8	17	32	47	32
PRESENT (this work)	2	4	14	16	47	16
GIFT (this work)	2	4	10	16	32	16
PICCOLO (this work)	2	4	15	16	35	16
PRESENT [12]	3	9	19	81	105	12
PRESENT [15]	3	8	17	72	93	12
PRESENT (this work)	3	4	14	36	67	12
GIFT [17]	3	5	15	45	69	12
GIFT [18]	3	5	15	45	69	12
GIFT [19]	3	5	15	45	69	12
GIFT (this work)	3	4	10	36	50	12
PICCOLO (this work)	3	4	15	36	63	12

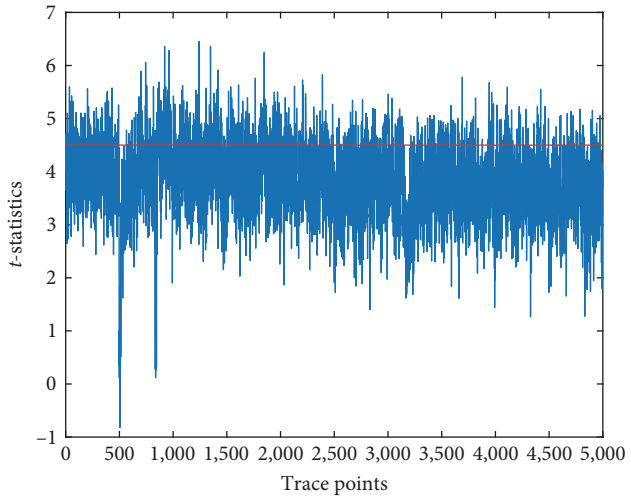

Figure 3: Unprotected GIFT S-box.

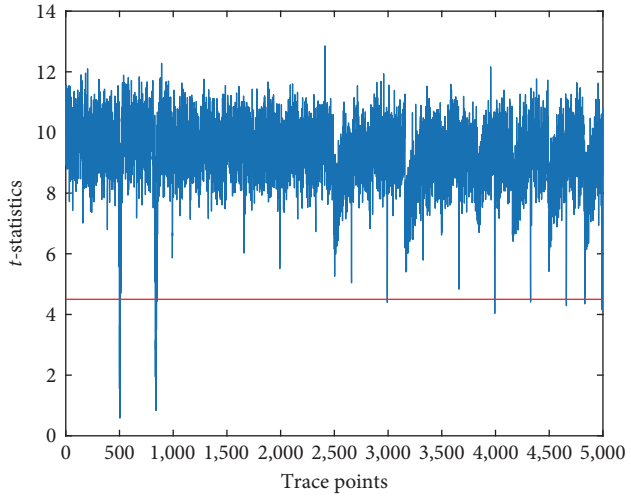

Figure 4: Unprotected PRESENT S-box.

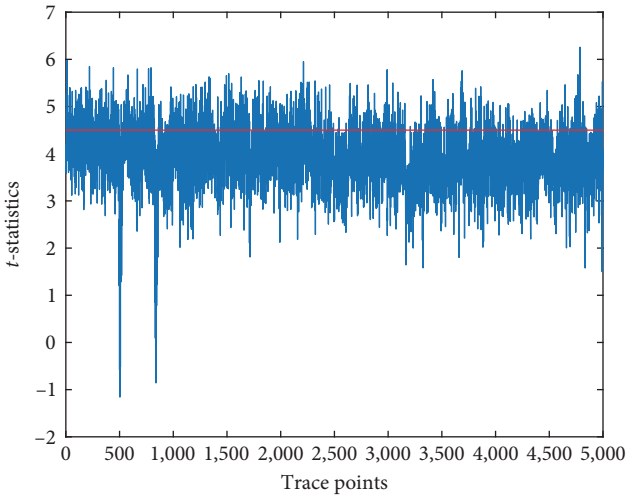

Figure 5: Unprotected PICCOLO S-box.

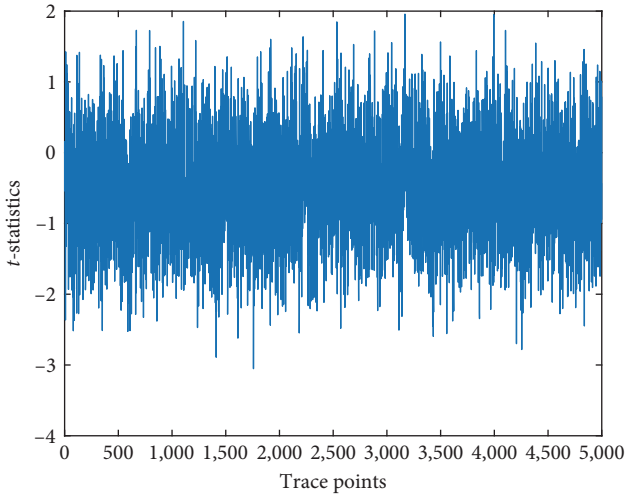

Figure 6: Two shares MS-LW-TI of GIFT S-box.

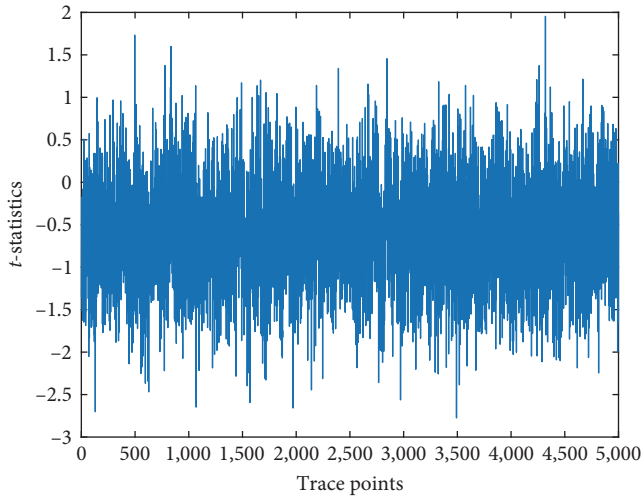

FIGURE 7: Three shares MS-LW-TI of GIFT S-box.

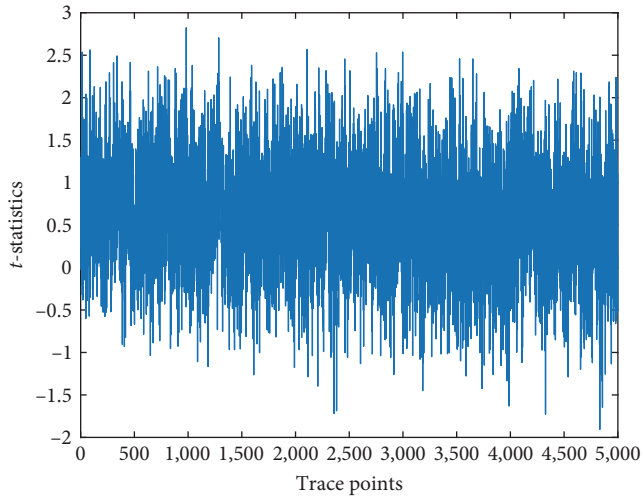

FIGURE 8: Two shares MS-LW-TI of PRESENT S-box.

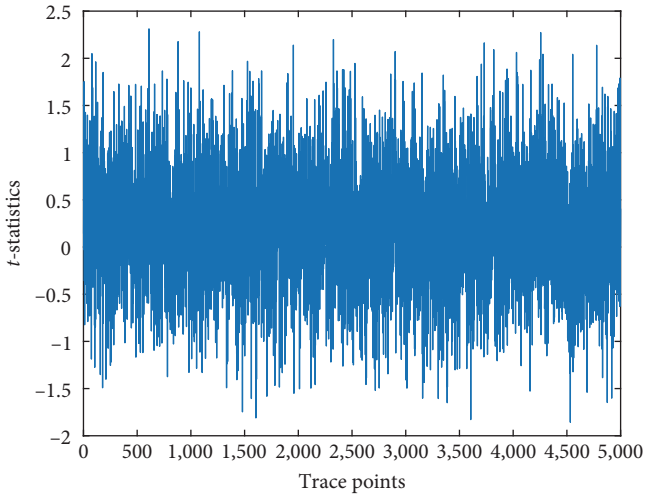

FIGURE 9: Three shares MS-LW-TI of PRESENT S-box.

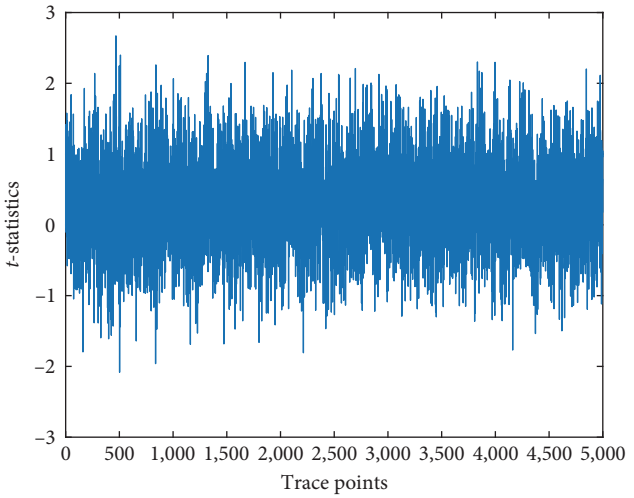

FIGURE 10: Two shares MS-LW-TI of PICCOLO S-box.

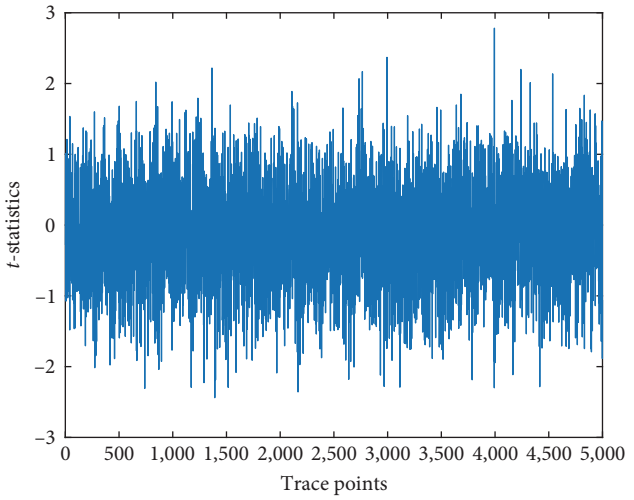

FIGURE 11: Three shares MS-LW-TI of PICCOLO S-box.

SAKURA-G board [31]. Meanwhile, we use the PicoScope 5000 Series to collect the power consumption traces at a sampling rate of 500 MS/s, and the number of sampling points of a power consumption trace is 5,000 points. The trigger signal goes high at the 10% position and drops to low after one clock cycle. The test vector leakage assessment (TVLA) is a general evaluation technique used to evaluate the security of masking schemes [32]. In the evaluation experiment, we collected the power consumption traces by conducting a reliable fixed-versus-random t-test, and then we evaluated whether there is a difference in the mean values of the distributions of the two sets of traces to determine whether there is information leakage. The TVLA threshold of ± 4.5 is selected based on [33] (confidence interval is 99.999%). We use the TVLA to perform the unprotected and first-order MS-LW-TI of the GIFT, PRESENT, and PICCOLO S-boxes.

We sample 200,000 power consumption traces for the unprotected S-boxes. The target FPGA receives unprotected

Table 5: Comparison MS-LW-TI with the existing TIs to resist SCA.

Design	Number of shares	Correctness	Noncompleteness	Uniformity	TVLA	dth order
PRESENT [23]	2	Yes	Yes	Yes	—	First-order
PRESENT [22]	2	Yes	Yes	Yes	Yes	First-order
PRESENT (this work)	2	Yes	Yes	Yes	Yes	First-order
GIFT (this work)	2	Yes	Yes	Yes	Yes	First-order
PICCOLO (this work)	2	Yes	Yes	Yes	Yes	First-order
PRESENT [12]	3	Yes	Yes	Yes	—	First-order
PRESENT [15]	3	Yes	Yes	Yes	—	First-order
PRESENT (this work)	3	Yes	Yes	Yes	Yes	First-order
GIFT [17]	3	Yes	Yes	Yes	Yes	First-order
GIFT [18]	3	Yes	Yes	Yes	Yes	First-order
GIFT [19]	3	Yes	Yes	Yes	Yes	First-order
GIFT(this work)	3	Yes	Yes	Yes	Yes	First-order
PICCOLO (this work)	3	Yes	Yes	Yes	Yes	First-order

Table 6: Comparison MS-LW-TI with the existing TIs, mapped for Spartan-6 (XC6SLX75).

Design	Number of shares	LUTs	FFs	Slices	Maximum frequency (MHz)	Fresh masks
PRESENT [23]	2	22	26	8	229.779	0
PRESENT [22]	2	18	30	10	525.486	0
PRESENT (this work)	2	17	16	6	351.494	0
GIFT (this work)	2	14	16	6	369.959	0
PICCOLO (this work)	2	13	16	6	527.704	0
PRESENT [12]	3	26	12	8	326.797	0
PRESENT [15]	3	30	12	12	375.516	0
PRESENT (this work)	3	20	9	6	343.289	0
GIFT [17]	3	16	12	7	414.938	0
GIFT [18]	3	16	12	7	414.938	0
GIFT [19]	3	16	12	7	414.938	0
GIFT (this work)	3	15	9	5	351.247	0
PICCOLO (this work)	3	14	6	8	539.084	0

input and issues output also in the same unprotected form. From Figures 3–5, the values of t-statistics are relatively large, more than 4.5. The difference between fixed and random data leads to information leakage of input operations, logical operations, and output operations for running the unprotected S-boxes on the target FPGA. This result is as expected and demonstrates a sufficiently high signal-to-noise ratio (SNR) for the experimental setup [17].

As we know, as the amount of traces increases, the t-statistics may also become larger. We sample 5,000,000 power consumption traces for first-order MS-LW-TI of the S-boxes. These traces have 25 times the number of unprotected S-boxes in the same experimental setup. The target FPGA receives shared input and issues output also in the same sharing form. From Figures 6–11, the values of t-statistics are small, which are in the range of ± 4.5. This result is also as expected and demonstrates there is no information leakage. On the other hand, the masking schemes can reduce the SNR in the implementation [34]. Considering protection against SCA, we compare the MS-LW-TI with the existing work in terms of the properties of TI, TVLA evaluation, and security order, respectively. Through Table 5, we consider that MS-LW-TI can resist first-order SCA.

4.4.2. Performance of MS-LW-TI. The implementation of MS-LW-TI is based on the Design Suite 14.7 with Xilinx Spartan-6 (XC6SLX75) FPGAs for S-boxes, which are used in the existing TI schemes [35, 36]. When these TI schemes map a hardware design to an FPGA, the authors count the number of occupied slices as a metric for size. Every slice contains four LUTs and eight FFs. In terms of resources, we implement the first-order MS-LW-TI of the GIFT S-box. The scheme of two shares requires 14 LUTs, 16 FFs, and six slices, and the scheme of three shares only requires 15 LUTs, nine FFs, and five slices. In terms of performance, the scheme of two shares needs 3 clock cycles and the maximum frequency is 369.959 MHz, the scheme of three shares needs three clock cycles and the maximum frequency is 351.247 MHz. We implement the first-order MS-LW-TI of the PRESENT S-box. The scheme of two shares only requires 17 LUTs, 16 FFs, six slices, and 351.494 MHz, the scheme of three shares requires 20 LUTs, nine FFs, six slices, and

343.289 MHz. The first-order MS-LW-TI S-box of PRESENT requires three clock cycles. Furthermore, we implement the first-order MS-LW-TI of the PICCOLO S-box. The scheme of two shares requires 13 LUTs, 16 FFs, six slices, and 527.704 MHz, the scheme of three shares only requires 14 LUTs, six FFs, eight slices, and 539.084 MHz. The first-order MS-LW-TI S-box of PICCOLO only requires two clock cycles.

The MS-LW-TI has lower resources in hardware implementation. Since the S-boxes are optimized, we obtain the least logic gates to construct the protection of S-boxes based on the primitives. The S-box optimizations of Bilgin et al. [27] and Cassiers et al. [28] have not been constructed into TI, and we do not consider the protection resources and the performance to compare with them. Compared with the TI of Shahmirzadi and Moradi, our two-shares scheme reduces 5 LUTs, 10 FFs, and two slices, which has a 22% reduction of LUTs, 38% reduction of FFs, and 25% reduction of slices with only two additional clock cycles. Compared with the TI of Jati et al. [17], our three-shares scheme reduces one LUT, three FFs, and two slices, which has a 6% reduction of LUTs, 25% reduction of FFs, and 28% reduction of slices with only one additional clock cycle. The detailed results are shown in Table 6. The designs expressed in Verilog Hardware Description Language (HDL) and the experimental results of the schemes are based on the implementation (post-place & route) of Design Suite 14.7 with Xilinx Spartan-6 FPGAs (XC6SLX75, Area Optimization Synthesis Mode) in Table 6.

5. Conclusion

TI is a very important masking scheme and can guarantee no leakage when glitches happen. The TI design of S-boxes is the key issue to protecting lightweight block ciphers such as PRESENT, GIFT, and PICCOLO. However, there is a high decomposition complexity and more resource consumption for TI of S-boxes of lightweight block ciphers.

To solve the problems, we propose a general, efficient, and low-resource TI scheme named MS-LW-TI. The S-box after MiniSat contains only AND gates, OR gates, and XOR gates, which are constituted into two basic primitives. Meanwhile, the two primitives can be extended to any logic gate functions of S-boxes. We build the $(d+1)$ and $(2d+1)$ first-order MS-LW-TI for the S-boxes, which satisfy the first-order glitch-extended probing secure. Thus, we take the GIFT S-box as an example to demonstrate the design, implementation, and safety analysis of MS-LW-TI in detail. By TVLA evaluation on 5,000,000 traces, the MS-LW-TI is no information leakage.

In this paper, we designed the lightweight implementation of $(d+1)$ and $(2d+1)$ first-order MS-LW-TI for the S-box of PRESENT, GIFT, and PICCOLO without any fresh randomness. The results of experiments show that MS-LW-TI has fewer resources than existing TI schemes. The MS-LW-TI is the smallest one presently, which is suitable for lightweight block cipher to resist first-order SCA in hardware implementation.

Appendix

A. The Logic Gate Functions of PRESENT S-Box by MiniSat.

The x_0, x_1, x_2, and x_3 are input variables, the y_0, y_1, y_2, and y_3 are output variables, the t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, and t_9 are intermediate variables:

$$\begin{aligned}&S(x_3,x_2,x_1,x_0)=(y_3,y_2,y_1,y_0),\\&t_1=x_2\oplus x_1, t_2=x_1\times t_1, t_3=x_0\oplus t_2, y_3=x_3\oplus t_3,\\&t_4=t_1\times t_3, t_5=t_1\oplus y_3, t_6=t_4\oplus x_1, t_7=x_3\parallel t_6,\\&y_2=t_5\oplus t_7, t_8=t_6\oplus x_3\oplus 1, y_0=y_2\oplus t_8, t_9=t_8\parallel t_5, y_1=t_3\oplus t_9.\end{aligned} \tag{A.1}$$

B. The Logic Gate Functions of PICCOLO S-Box by MiniSat.

The x_0, x_1, x_2, and x_3 are input variables, the y_0, y_1, y_2, and y_3 are output variables, the t_0, t_1, t_4, and t_5 are intermediate variables:

$$\begin{aligned}&S(x_3,x_2,x_1,x_0)=(y_3,y_2,y_1,y_0),\\&t_0=x_0\parallel x_1, t_1=x_1\parallel x_2\oplus 1, y_0=x_3\oplus t_0\oplus 1, y_1=t_1\oplus x_0,\\&t_4=y_0\parallel y_1, t_5=y_0\parallel x_2\oplus 1, y_2=t_5\oplus x_1\oplus 1, y_3=t_4\oplus x_2\oplus 1.\end{aligned} \tag{B.1}$$

C. The ANF Equations of Two Shares of the MS-LW-TI for GIFT S-Box.

The $x_{(0,0)}$, $x_{(0,1)}$, $x_{(1,0)}$, $x_{(1,1)}$, $x_{(2,0)}$, $x_{(2,1)}$, $x_{(3,0)}$, and $x_{(3,1)}$ are input shares, the $y_{(0,0)}$, $y_{(0,1)}$, $y_{(1,0)}$, $y_{(1,1)}$, $y_{(2,0)}$, $y_{(2,1)}$, $y_{(3,0)}$, and $y_{(3,1)}$ are output shares:

$$\begin{aligned}&S\left(x_{(3,1)},x_{(3,0)},x_{(2,1)},x_{(2,0)},x_{(1,1)},x_{(1,0)},x_{(0,1)},x_{(0,0)}\right)\\&=\left(y_{(3,1)},y_{(3,0)},y_{(2,1)},y_{(2,0)},y_{(1,1)},y_{(1,0)},y_{(0,1)},y_{(0,0)}\right),\end{aligned} \tag{C.1}$$

$$\begin{aligned}&t'_{(2,0)}=x_{(2,0)}\times x_{(3,0)}\oplus x_{(2,0)}\oplus x_{(3,0)}\oplus x_{(1,0)},\\&t'_{(2,1)}=x_{(2,0)}\times x_{(3,1)},\\&t'_{(2,2)}=x_{(2,1)}\times x_{(3,0)},\\&t'_{(2,3)}=x_{(2,1)}\times x_{(3,1)}\oplus x_{(2,1)}\oplus x_{(3,1)}\oplus x_{(1,1)},\\&t_{(2,0)}=t'_{(2,0)}\oplus t'_{(2,1)}, t_{(2,1)}=t'_{(2,2)}\oplus t'_{(2,3)},\end{aligned} \tag{C.2}$$

$$\begin{aligned}&t'_{(3,0)}=x_{(1,0)}\times x_{(3,0)}\oplus x_{(2,0)},\\&t'_{(3,1)}=x_{(1,0)}\times x_{(3,1)},\\&t'_{(3,2)}=x_{(1,1)}\times x_{(3,0)},\\&t'_{(3,3)}=x_{(1,1)}\times x_{(3,1)}\oplus x_{(2,1)},\\&t_{(3,0)}=t'_{(3,0)}\oplus t'_{(3,1)}, t_{(3,1)}=t'_{(3,2)}\oplus t'_{(3,3)},\end{aligned} \tag{C.3}$$

$$y_{(3,0)}=x_{(0,1)}\oplus t_{(2,1)}\oplus 1, y_{(3,1)}=x_{(0,0)}\oplus t_{(2,0)}, \tag{C.4}$$

$$y_{(2,0)} = y_{(3,1)} \oplus t_{(3,1)} \oplus 1, y_{(2,1)} = y_{(3,0)} \oplus t_{(3,0)}, \quad \text{(C.5)}$$

$$\begin{aligned} y'_{(0,0)} &= x_{(0,0)} \times t_{(3,0)} \oplus x_{(3,0)}, \\ y'_{(0,1)} &= x_{(0,0)} \times t_{(3,1)}, \\ y'_{(0,2)} &= x_{(0,1)} \times t_{(3,0)}, \\ y'_{(0,3)} &= x_{(0,1)} \times t_{(3,1)} \oplus x_{(3,1)}, \\ y_{(0,0)} &= y'_{(3,0)} \oplus y'_{(3,1)}, y_{(0,1)} = y'_{(3,2)} \oplus y'_{(3,3)}, \end{aligned} \quad \text{(C.6)}$$

$$\begin{aligned} y'_{(1,0)} &= y_{(0,0)} \times y_{(3,0)} \oplus t_{(2,0)}, \\ y'_{(1,1)} &= y_{(0,0)} \times y_{(3,1)}, \\ y'_{(1,2)} &= y_{(0,1)} \times y_{(3,0)}, \\ y'_{(1,3)} &= y_{(0,1)} \times y_{(3,1)} \oplus t_{(2,1)}, \\ y_{(1,0)} &= y'_{(1,0)} \oplus y'_{(1,1)}, y_{(1,1)} = y'_{(1,2)} \oplus y'_{(1,3)}, \end{aligned} \quad \text{(C.7)}$$

D. The ANF Equations of Three Shares of the MS-LW-TI for GIFT S-Box.

The $x_{(0,0)}$, $x_{(0,1)}$, $x_{(0,2)}$, $x_{(1,0)}$, $x_{(1,1)}$, $x_{(1,2)}$, $x_{(2,0)}$, $x_{(2,1)}$, $x_{(2,2)}$, $x_{(3,0)}$, $x_{(3,1)}$, and $x_{(3,2)}$ are input shares, the $y_{(0,0)}$, $y_{(0,1)}$, $y_{(0,2)}$, $y_{(1,0)}$, $y_{(1,1)}$, $y_{(1,2)}$, $y_{(2,0)}$, $y_{(2,1)}$, $y_{(2,2)}$, $y_{(3,0)}$, $y_{(3,1)}$, and $y_{(3,2)}$ are output shares:

$$\begin{aligned} &S\left(x_{(3,2)}, x_{(3,1)}, x_{(3,0)}, x_{(2,2)}, x_{(2,1)}, x_{(2,0)}, x_{(1,2)}, x_{(1,1)}, x_{(1,0)}, x_{(0,2)}, x_{(0,1)}, x_{(0,0)}\right) \\ &= \left(y_{(3,2)}, y_{(3,1)}, y_{(3,0)}, y_{(2,2)}, y_{(2,1)}, y_{(2,0)}, y_{(1,2)}, y_{(1,1)}, y_{(1,0)}, y_{(0,2)}, y_{(0,1)}, y_{(0,0)}\right), \end{aligned} \quad \text{(D.1)}$$

$$\begin{aligned} t_{(2,0)} &= x_{(2,1)} \times x_{(3,1)} \oplus x_{(2,1)} \times x_{(3,2)} \oplus x_{(2,2)} \times x_{(3,1)} \oplus x_{(2,1)} \oplus x_{(3,1)} \oplus x_{(1,1)}, \\ t_{(2,1)} &= x_{(2,2)} \times x_{(3,2)} \oplus x_{(2,2)} \times x_{(3,0)} \oplus x_{(2,0)} \times x_{(3,2)} \oplus x_{(2,2)} \oplus x_{(3,2)} \oplus x_{(1,2)}, \\ t_{(2,2)} &= x_{(2,0)} \times x_{(3,0)} \oplus x_{(2,0)} \times x_{(3,1)} \oplus x_{(2,1)} \times x_{(3,0)} \oplus x_{(2,0)} \oplus x_{(3,0)} \oplus x_{(1,0)}, \end{aligned} \quad \text{(D.2)}$$

$$\begin{aligned} t_{(3,0)} &= x_{(1,1)} \times x_{(3,1)} \oplus x_{(1,1)} \times x_{(3,2)} \oplus x_{(1,2)} \times x_{(3,1)} \oplus x_{(2,1)}, \\ t_{(3,1)} &= x_{(1,2)} \times x_{(3,2)} \oplus x_{(1,2)} \times x_{(3,0)} \oplus x_{(1,0)} \times x_{(3,2)} \oplus x_{(2,2)}, \\ t_{(3,2)} &= x_{(1,0)} \times x_{(3,0)} \oplus x_{(1,0)} \times x_{(3,1)} \oplus x_{(1,1)} \times x_{(3,0)} \oplus x_{(2,0)}, \end{aligned} \quad \text{(D.3)}$$

$$\begin{aligned} y_{(3,0)} &= x_{(0,1)} \oplus t_{(2,1)} \oplus 1, y_{(3,1)} = x_{(0,2)} \oplus t_{(2,2)}, y_{(3,2)} \\ &= x_{(0,0)} \oplus t_{(2,0)}, \end{aligned} \quad \text{(D.4)}$$

$$\begin{aligned} y_{(2,0)} &= y_{(3,1)} \oplus t_{(3,1)} \oplus 1, y_{(2,1)} = y_{(3,2)} \oplus t_{(3,2)}, y_{(2,2)} \\ &= y_{(3,0)} \oplus t_{(3,0)}, \end{aligned} \quad \text{(D.5)}$$

$$\begin{aligned} y_{(0,0)} &= x_{(0,1)} \times t_{(3,1)} \oplus x_{(0,1)} \times t_{(3,2)} \oplus x_{(0,2)} \times t_{(3,1)} \oplus x_{(3,1)}, \\ y_{(0,1)} &= x_{(0,2)} \times t_{(3,2)} \oplus x_{(0,2)} \times t_{(3,0)} \oplus x_{(0,0)} \times t_{(3,2)} \oplus x_{(3,2)}, \\ y_{(0,2)} &= x_{(0,0)} \times t_{(3,0)} \oplus x_{(0,0)} \times t_{(3,1)} \oplus x_{(0,1)} \times t_{(3,0)} \oplus x_{(3,0)}, \end{aligned} \quad \text{(D.6)}$$

$$\begin{aligned} y_{(1,0)} &= y_{(0,1)} \times y_{(2,1)} \oplus y_{(0,1)} \times y_{(2,2)} \oplus y_{(0,2)} \times y_{(2,1)} \oplus t_{(2,1)}, \\ y_{(1,1)} &= y_{(0,2)} \times y_{(2,2)} \oplus y_{(0,2)} \times y_{(2,0)} \oplus y_{(0,0)} \times y_{(2,2)} \oplus t_{(2,2)}, \\ y_{(1,2)} &= y_{(0,0)} \times y_{(2,0)} \oplus y_{(0,0)} \times y_{(2,1)} \oplus y_{(0,1)} \times y_{(2,0)} \oplus t_{(2,0)}. \end{aligned} \quad \text{(D.7)}$$

E. The Implementation Process of Logic Gate Functions of PRESENT S-Box.

The x_0, x_1, x_2, and x_3 are input variables, the y_0, y_1, y_2, and y_3 are output variables, the t_1, t_3, t_5, t_6, and t_8 are intermediate variables:

$$\begin{aligned} &S(x_3, x_2, x_1, x_0) = (y_3, y_2, y_1, y_0) \\ &t_1 = x_2 \oplus x_1, t_3 = x_1 \times t_1 \oplus x_0, y_3 = x_3 \oplus t_3, \\ &t_6 = t_1 \times t_3 \oplus x_1, t_5 = t_1 \oplus y_3, t_8 = t_6 \oplus x_3 \oplus 1, \\ &y_2 = x_3 \times t_6 \oplus x_3 \oplus t_6 \oplus t_5, y_0 = y_2 \oplus t_8, y_1 = t_8 \times t_5 \oplus t_8 \oplus t_5 \oplus t_3. \end{aligned} \quad \text{(E.1)}$$

F. The Implementation Process of Logic Gate Functions of PICCOLO S-Box.

The x_0, x_1, x_2, and x_3 are input variables, the y_0, y_1, y_2, and y_3 are output variables:

$$\begin{aligned} &S(x_3, x_2, x_1, x_0) = (y_3, y_2, y_1, y_0) \\ &y_0 = x_0 \times x_1 \oplus x_1 \oplus x_0 \oplus x_3 \oplus 1, y_1 = x_1 \times x_2 \oplus x_1 \oplus x_2 \oplus x_0 \oplus 1, \\ &y_2 = y_0 \times x_2 \oplus y_0 \oplus x_2 \oplus x_1, y_3 = y_0 \times y_1 \oplus y_0 \oplus y_1 \oplus x_2 \oplus 1. \end{aligned} \quad \text{(F.1)}$$

Data Availability

Data are available upon request from the authors.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Key R&D Program of China (No. 2022YFB3103800).

References

[1] A. Bogdanov, L. R. Knudsen, G. Leander et al., "Present: an ultra-lightweight block cipher," in *Cryptographic Hardware and Embedded Systems-CHES 2007: 9th International Workshop*, pp. 450–466, Springer, Berlin, Heidelberg, Vienna, Austria, September 2007.

[2] S. Banik, S. K. Pandey, T. Peyrin, Y. Sasaki, S. M. Sim, and Y. Todo, "Gift: a small present: towards reaching the limit of lightweight encryption," in *Cryptographic Hardware and Embedded Systems–CHES 2017: 19th International Conference*, pp. 321–345, Springer, Cham, Taipei, Taiwan, September 2017.

[3] K. Shibutani, T. Isobe, H. Hiwatari, A. Mitsuda, T. Akishita, and T. Shirai, "Piccolo: an ultra-lightweight blockcipher," in *Cryptographic Hardware and Embedded Systems–CHES 2011: 13th International Workshop*, pp. 342–357, Springer, Berlin, Heidelberg, Nara, Japan, September 2011.

[4] J. Guo, T. Peyrin, A. Poschmann, and M. Robshaw, "The LED block cipher," in *Cryptographic Hardware and Embedded Systems–CHES 2011: 13th International Workshop*, pp. 326–341, Springer, Berlin, Heidelberg, Nara, Japan, September 2011.

[5] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology-CRYPTO'99: 19th Annual International Cryptology Conference*, pp. 388–397, Springer, Berlin, Heidelberg, Santa Barbara, California, USA, August 1999.

[6] A. Biryukov, D. Dinu, and J. Großschädl, "Correlation power analysis of lightweight block ciphers: from theory to practice," in *Applied Cryptography and Network Security: 14th International Conference, ACNS 2016*, pp. 537–557, Springer, Cham, Guildford, UK, June 2016.

[7] D. Shanmugam, R. Selvam, and S. Annadurai, "Differential power analysis attack on SIMON and LED block ciphers," in *Security, Privacy, and Applied Cryptography Engineering: 4th International Conference, SPACE 2014*, pp. 110–125, Springer, Cham, Pune, India, October 2014.

[8] O. Lo, W. J. Buchanan, and D. Carson, "Correlation power analysis on the PRESENT block cipher on an embedded device," in *ARES '18: Proceedings of the 13th International Conference on Availability, Reliability and Security*, pp. 1–6, Association for Computing Machinery, Hamburg, Germany, August 2018.

[9] Y. Ou and L. Li, "Research on a high-order AES mask anti-power attack," *IET Information Security*, vol. 14, no. 5, pp. 580–586, 2020.

[10] S. Nikova, C. Rechberger, and V. Rijmen, "Threshold implementations against side-channel attacks and glitches," in *International Conference on Information and Communications Security*, pp. 529–545, Springer, Berlin, Heidelberg, 2006.

[11] A. Juels and S. A. Weis, "Authenticating pervasive devices with human protocols," in *Advances in Cryptology–CRYPTO 2005: 25th Annual International Cryptology Conference*, pp. 293–308, Springer, Berlin, Heidelberg, Santa Barbara, California, USA, August 2005.

[12] A. Poschmann, A. Moradi, K. Khoo, C.-W. Lim, H. Wang, and S. Ling, "Side-channel resistant crypto for less than 2,300 GE," *Journal of Cryptology*, vol. 24, no. 2, pp. 322–345, 2011.

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

[14] B. Bilgin, B. Gierlichs, S. Nikova, V. Nikov, and V. Rijmen, "Higher-order threshold implementations," in *Advances in Cryptology–ASIACRYPT 2014: 20th International Conference on the Theory and Application of Cryptology and Information Security*, pp. 326–343, Springer, Berlin, Heidelberg, Kaoshiung, Taiwan, ROC, December 2014.

[15] S. Kutzner, P. H. Nguyen, A. Poschmann, and H. Wang, "On 3-share threshold implementations for 4-bit S-boxes," in *Constructive Side-Channel Analysis and Secure Design: 4th International Workshop, COSADE 2013*, pp. 99–113, Springer, Berlin, Heidelberg, Paris, France, March 2013.

[16] Y. Yao, M. Yang, P. Kiaei, and P. Schaumont, "Dimming down LED: an open-source threshold implementation on light encryption device (LED) block cipher," 2021.

[17] A. Jati, N. Gupta, A. Chattopadhyay, S. K. Sanadhya, and D. Chang, "Threshold implementations of GIFT: a trade-off analysis," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2110–2120, 2020.

[18] V. Satheesh and D. Shanmugam, "Secure realization of lightweight block cipher: a case study using GIFT," in *Security, Privacy, and Applied Cryptography Engineering: 8th International Conference, SPACE 2018*, pp. 85–103, Springer, Cham, Kanpur, India, December 2018.

[19] A. Caforio, D. Collins, S. Banik, and F. Regazzoni, "A small GIFT-COFB: lightweight bit-serial architectures," in *International Conference on Cryptology in Africa*, pp. 53–77, Springer, Cham, 2022.

[20] O. Reparaz, B. Bilgin, S. Nikova, B. Gierlichs, and I. Verbauwhede, "Consolidating masking schemes," in *Cryptology–CRYPTO 2015: 35th Annual Cryptology Conference*, pp. 764–783, Springer, Berlin, Heidelberg, Santa Barbara, CA, USA, August 2015.

[21] H. Gross, S. Mangard, and T. Korak, "Domain-oriented masking: compact masked hardware implementations with arbitrary protection order," in *TIS '16: Proceedings of the 2016 ACM Workshop on Theory of Implementation Security*, p. 3, Association for Computing Machinery, Vienna, Austria, October 2016.

[22] C. Chen, M. Farmani, and T. Eisenbarth, "A tale of two shares: why two-share threshold implementation seems worthwhile—and why it is not," in *Advances in Cryptology–ASIACRYPT 2016: 22nd International Conference on the Theory and Application of Cryptology and Information Security*, pp. 819–843, Springer, Berlin, Heidelberg, Hanoi, Vietnam, December 2016.

[23] A. Rezaei Shahmirzadi and A. Moradi, "Re-consolidating first-order masking schemes: nullifying fresh randomness," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2021, no. 1, pp. 305–342, 2020.

[24] S. Faust, V. Grosso, S. Merino Del Pozo, C. Paglialonga, and F.-X. Standaert, "Composable masking schemes in the presence

of physical defaults & the robust probing model," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2018, no. 3, pp. 89–120, 2018.

[25] K. Stoffelen, "Optimizing S-box implementations for several criteria using SAT solvers," in *International Conference on Fast Software Encryption*, pp. 140–160, Springer, Berlin, Heidelberg, July 2016.

[26] N. T. Courtois, D. Hulme, and T. Mourouzis, "Solving circuit optimisation problems in cryptography and cryptanalysis," *Cryptology ePrint Archive*, 2011.

[27] B. Bilgin, L. De Meyer, S. Duval, I. Levi, and F.-X. Standaert, "Low AND depth and efficient inverses: a guide on S-boxes for low-latency masking," *IACR Transactions on Symmetric Cryptology*, vol. 2020, no. 1, pp. 144–184, 2020.

[28] G. Cassiers, B. Grégoire, I. Levi, and F.-X. Standaert, "Hardware private circuits: from trivial composition to full verification," *IEEE Transactions on Computers*, vol. 70, no. 10, pp. 1677–1690, 2021.

[29] B. Bilgin, B. Gierlichs, S. Nikova, V. Nikov, and V. Rijmen, "Trade-offs for threshold implementations Illustrated on AES," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 7, pp. 1188–1200, 2015.

[30] S. Dhooghe, S. Nikova, and V. Rijmen, "Threshold implementations in the robust probing model," in *Proceedings of ACM Workshop on Theory of Implementation Security Workshop*, pp. 30–37, Association for Computing Machinery, London, United Kingdom, November 2019.

[31] SAKURA, "Side-channel attack user reference architecture," 2024, Last accessed on: 1-16 https://satoh.cs.uec.ac.jp/SAKURA/index.html.

[32] B. J. Gilbert Goodwill, J. Jaffe, P. Rohatgi, and Cryptography Research Inc, "A testing methodology for side-channel resistance validation," *NIST Non-Invasive Attack Testing Workshop*, vol. 7, pp. 115–136, 2011.

[33] G. Becker, J. Cooper, E. DeMulder et al., "Test vector leakage assessment (TVLA) methodology in practice," *International Cryptographic Module Conference*, vol. 1001, Article ID 13, 2013.

[34] S. Guilley, A. Heuser, and O. Rioul, "Codes for side-channel attacks and protections," in *Codes, Cryptology and Information Security: Second International Conference, C2SI 2017*, pp. 35–55, Springer, Cham, Rabat, Morocco, Proceedings-In Honor of Claude Carlet 2, April 2017.

[35] F. Wegener, L. De Meyer, and A. Moradi, "Spin me right round rotational symmetry for FPGA-specific AES: extended version," *Journal of Cryptology*, vol. 33, no. 3, pp. 1114–1155, 2020.

[36] A. R. Shahmirzadi, D. Božilov, and A. Moradi, "New first-order secure aes performance records," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2021, no. 2, pp. 304–327, 2021.