IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2025, No. 1, pp. 340–366. DOI:10.46586/tches.v2025.i1.340-366

Extending Randomness-Free First-Order Masking Schemes and Applications to Masking-Friendly S-boxes

Lixuan Wu[1,2], Yanhong Fan[1,2,3], Weijia Wang[3,1,2], Bart Preneel[4] and Meiqin Wang[3,1,2](✉)

[1] School of Cyber Science and Technology, Shandong University, Qingdao, China
yanhongfan@sdu.edu.cn,lixuanwu@mail.sdu.edu.cn
[2] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan, China
[3] Quan Cheng Shandong Laboratory, Jinan, China
{wjwang,mqwang}@sdu.edu.cn
[4] imec-COSIC, KU Leuven, Leuven, Belgium
bart.preneel@esat.kuleuven.be

Abstract.
Masking has emerged as a widely adopted countermeasure against side-channel attacks. However, the implementation of masking schemes faces several challenges, including hardware area, latency and the overhead associated with fresh randomness generation. To eliminate the implementation cost caused by fresh randomness, Shahmirzadi et al. introduced a methodology for constructing 2-share first-order masking schemes without randomness at CHES 2021. In this work, we extend Shahmirzadi et al.'s method to find masked implementations for more S-boxes and further reduce the hardware overhead. We propose the concept of a non-linear compression layer, a comprehensive share assignment strategy based on a linear compression layer, and corresponding optimization techniques. Based on these techniques, we construct the first randomness-free first-order masking schemes for the PRINCE S-box and its inverse, reduce the hardware overhead of masking schemes for multiple S-boxes, and design new masking-friendly S-boxes. Particularly for the SKINNY S-box, the reduction is 21% and 15% in area and power consumption, respectively. To validate the security of masked implementations, we not only employ the automated tools SILVER and PROLEAD but also conduct FPGA-based experiments.

Keywords: Extended first-order masking · Non-linear compression · Friendly masking scheme · PRINCE

1 Introduction

Side-Channel Analysis (SCA) attacks [Koc96, KJJ99] exploit information on timing and power consumption to extract secret variables from cryptographic devices. Due to this discovery, they have attracted considerable attention from researchers and practitioners. Embedded devices running cryptographic algorithms are particularly vulnerable to such attacks. To mitigate the severe threat from SCA attacks, researchers have proposed a large number of countermeasures, among which masking is the most popular due to its sound theoretical foundations.

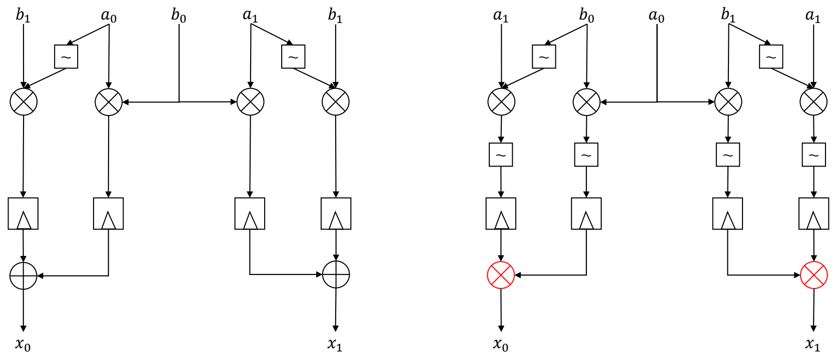
The Ishai-Sahai-Wagner (ISW) d-probing model [ISW03] shows how to correctly implement a masking scheme in a software approach. This model assumes that an adversary can

Received: 2024-07-15 Accepted: 2024-09-15 Published: 2024-12-09

probe the circuit in d locations simultaneously. However, adapting masking schemes for hardware platforms remains challenging, because the d-probing model does not cover some hardware physical defaults. Many masking schemes [Tri03, OMPR05] have been shown to be vulnerable because of glitches, which are regarded as one of the most challenging hardware physical defaults. In this respect, the glitch-extended probing model of Faust et al. [FGP+18] provides a comprehensive formal framework for analyzing and addressing the impact of glitches on hardware designs. This model assumes each probe placed on a combinatorial circuit propagates backward to either the primary inputs or the last synchronization point (e.g., registers).

As the first implementation strategy, Threshold Implementation (TI) [NRR06] is widely studied to resist glitches in hardware implementations as it offers security proofs based on a solid theory. In classic TI, the number of shares for a non-linear function is $td+1$, where t is the algebraic degree of this function and d is the desired security order, which means that any set of up to d intermediate variables does not leak key-dependent information. In order to reduce the number of input shares, [RBN+15, GMK16] have demonstrated that $d+1$ shares are sufficient to achieve d-order security. In this approach, a layer of registers is inserted to block the propagation of glitches within a masking scheme. However, it should be noted that this technique often requires the inclusion of randomness. For instance, the 2-input AND gate with first-order security requires at least 1-bit randomness [GMK16].

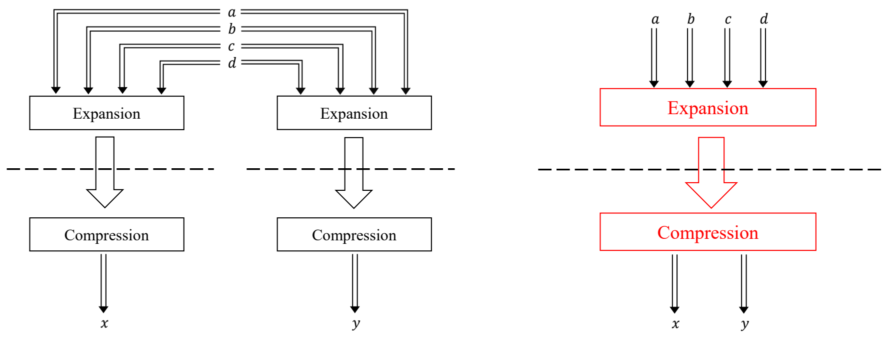
To eliminate the reliance on randomness, Shahmirzadi et al. [SM21a] proposed a methodology to construct randomness-free first-order masking schemes for a Boolean function. Fig. 1(a) shows a masked implementation of a 2-input AND gate, where $a_{0/1}$ and $b_{0/1}$ are the inputs and $x_{0/1}$ are the outputs. This implementation consists of two layers, the one before the registers is called the expansion layer, where certain properties need to be satisfied (e.g., non-completeness [NRR06], identical joint probability distribution [SM21a], etc.), and the one after the registers is called the compression layer, where the results from registers are XOR-ed to generate the final outputs.



(a) Shahmirzadi et al.'s method. (b) Our extended method.

Figure 1: Masked implementations of a 2-input AND gate.

Shahmirzadi et al. [SM21a] then applied this method to larger circuits (rather than an AND gate), such as S-boxes of block ciphers. They proposed to construct masking schemes for each coordinate function of this S-box individually, and then filter out a combination of each masked coordinate function, which is joint uniform [NRS11]. Fig. 2(a) shows a schematic for two coordinate functions of an S-box, where a, b, c and d are the inputs, x and y are the outputs, and the black dashed lines denote the registers. While nullifying randomness holds significant merit, it is imperative to acknowledge the existence of certain S-boxes that Shahmirzadi et al.'s method cannot effectively address. A notable illustration is the PRINCE S-box [BCG+12] and its inverse.



(a) Shahmirzadi et al.'s method. (b) Our extended method.

Figure 2: Masked implementations of two Boolean functions.

1.1 Contributions

This paper extends Shahmirzadi et al.'s method by proposing several new techniques that enable us to find the masked implementations for a wider range of S-boxes but also improve the hardware performance of multiple S-boxes. The contributions[1] are described below.

Masking schemes based on a non-linear compression layer. Taking the 2-input AND gate as an example, demonstrated in Fig. 1(b), we first propose that a non-linear compression layer can also be used to construct a masked Boolean function. Due to the flexibility of non-linear operations, the non-linear compression layer offers two advantages. Firstly, the integration of the non-linear compression layer significantly expands the design space of masking schemes, enabling the construction of more masking schemes. Secondly, the non-linear compression layer is capable of sharing some of the component functions from the expansion layer within a masking scheme. The technique based on shared component functions effectively reduces the number of registers and hardware overhead.

Joint masking schemes for multiple Boolean functions. Due to the increased computational complexity, we propose a joint approach based on the linear compression layer to construct masking schemes for multiple Boolean functions, such as an S-box. In this respect, we introduce a comprehensive share assignment strategy that efficiently assigns the shares of primary inputs to component functions. This strategy can derive all possible share assignment types and enable the construction of more masking schemes in the linear compression scenario. Based on the above strategy, we develop a joint masking scheme, depicted in Fig. 2(b), which consists of a joint expansion layer and a joint compression layer. Compared with the individual method in Fig. 2(a), the joint method allows for the sharing of component functions among multiple masked Boolean functions, effectively reducing the overall number of registers and hardware overhead. Based on the above techniques, we have not only solved the uniformity issues of the PRINCE S-box and its inverse, but also further optimized the hardware overhead of masking schemes for multiple S-boxes. Specifically for the SKINNY S-box, we achieved notable savings of 21% in area and 15% in dynamic power consumption.

Constructing new masking-friendly S-boxes. Since the aforementioned techniques allow for the construction of more efficient first-order masking schemes without randomness, we apply them to construct new masking-friendly S-boxes. Based on the work of [LMC+22] and the proposed masking techniques, we have successfully constructed some

[1] Our material including detailed masking schemes and their HDL codes is given at https://github.com/dfhsdkjc243/dabchxmchaeyufdbj.git

new S-boxes that not only exhibit equivalent cryptographic properties to the PRINCE S-box and its inverse, but also demonstrate improved hardware masking performance.

1.2 Outline

We provide some essential notions in Sect. 2, before describing the fundamental methodology of extending the randomness-free first-order masking schemes. In Sect. 3, we present a detailed description of our proposed techniques, including masking schemes based on a non-linear compression layer, joint masking schemes for multiple Boolean functions, and an application of these techniques to design new masking-friendly S-boxes. To demonstrate the effectiveness of our techniques, we present some case studies and their hardware performance in Sect. 4. In Sect. 5, we evaluate the security of our implementations.

2 Preliminaries

2.1 Notations and Definitions

We denote binary random variables with lower-case italic x, and the i-th share of a variable with x_i. A capital $X(\in \mathbb{F}_2^n, n > 1)$ represents a random binary vector, while X^j denotes the j-th element of the vector X. In a Boolean masking scheme, a secret variable x is divided into s shares, such that their XOR sum is equal to the secret variable x, i.e., $x = \bigoplus_{i=0}^{s-1} x_i$. These s shares are generated in two steps: first, $s-1$ shares are generated randomly and independently, and then the last share x_{s-1} is derived from $x_{s-1} = x \oplus \bigoplus_{i=0}^{s-2} x_i$.

2.2 Probing Security

The masking scheme is a countermeasure that can achieve resistance to side-channel attacks from the algorithm level. The probing security model [ISW03] is an essential theory to evaluate the security of masking schemes. Hardware and software implementations require different masking solutions; this paper focuses on hardware.

In hardware platforms, the path delay of each logic gate can be different, leading to unpredictable signal transitions at the output of a combinatorial circuit. These unpredictable signal transitions that are inherent to hardware implementations are called glitches. To take into account the presence of glitches, Faust et al. [FGP+18] proposed the *glitch-extended* probing model, which is the most popular formal framework in the presence of glitches. In this model, by placing a probe at the output of a gate, the glitch-extended probe returns all values contributing to the probed wire until the synchronization point (e.g., registers).

2.3 Masking with $d + 1$ Shares

According to Threshold Implementations [NRR06], the minimum number of input shares required equals $td + 1$, where t is the algebraic degree of the cryptographic function and d is the desired security order. It has been shown in [RBN+15, GMK16] that we can use $d + 1$ input shares to achieve the same security in hardware implementation. In this implementation scheme, the masked target function is divided into two register-isolated parts, and fresh randomness is introduced. An example for 2-input AND gate $x = f(a, b) = ab$ is given in Eq. (1), where a_0 and a_1 (resp. b_0 and b_1) are the two shares of primary input a (resp. b), x_0 and x_1 are the output shares, and $f_i(.)$, $0 \le i \le 3$ are the *component functions*, the results of which are stored in registers x'_0 to x'_3. The above part is called the *expansion layer*. In the *compression layer*, these values stored in registers are XOR-ed to generate the output shares x_0 and x_1.

$$\begin{array}{llll} f_0(a_0, b_0) & = a_0 b_0 & \rightarrow x'_0 & \\ f_1(a_0, b_1, r) & = a_0 b_1 + r & \rightarrow x'_1 & x'_0 + x'_1 = x_0 \\ \hline f_2(a_1, b_0, r) & = a_1 b_0 + r & \rightarrow x'_2 & x'_2 + x'_3 = x_1 \\ f_3(a_1, b_1) & = a_1 b_1 & \rightarrow x'_3 & \end{array}. \tag{1}$$

In [SM21a], a methodology is introduced to construct 2-share first-order hardware implementations without randomness. The 2-share first-order masking scheme of a 2-input AND gate $x = f(a, b) = ab$ is shown below:

$$\begin{array}{llll} f_0(a_0, b_0) & = a_0 b_0 & \rightarrow x'_0 & \\ f_1(a_0, b_1) & = a_0 b_1 + b_1 & \rightarrow x'_1 & x'_0 + x'_1 = x_0 \\ \hline f_2(a_1, b_0) & = a_1 b_0 & \rightarrow x'_2 & x'_2 + x'_3 = x_1 \\ f_3(a_1, b_1) & = a_1 b_1 + b_1 & \rightarrow x'_3 & \end{array}. \tag{2}$$

The construction of a masking scheme in Eq. (2) can be transformed into the problem of finding a set of component functions that meet the following five properties.

Definition 1 (Non-completeness [NRR06])**.** Every component function is independent of at least one share of the secret variables. This ensures that the evaluation of individual component functions does not reveal any information about the secret variables.

Example. Equation (3) gives an expression of the non-completeness with 4 component functions, where $\hat{a_i}$ and $\hat{b_i}$ are the inputs to the component function f_i, the binary variables λ_i and μ_i determine which share of secret variables is assigned to f_i. For example, the component function f_0 in Eq. (2) is independent of a_1 and b_1, i.e., $\lambda_0 = 0, \mu_0 = 0$.

$$\begin{aligned} \forall i \in \{0, 1, 2, 3\}: \quad & \hat{a_i} = \lambda_i \cdot a_1 + (1 - \lambda_i) \cdot a_0, \\ & \hat{b_i} = \mu_i \cdot b_1 + (1 - \mu_i) \cdot b_0. \end{aligned} \tag{3}$$

Definition 2 (Identical Joint Probability Distribution [SM21a])**.** The joint distribution of a set of variables within a masking scheme remains constant irrespective of the values taken by the secret input variables.

Example. This property for Eq. (2) is reflected by Eq. (4). Here α and β denote the specific values of the two joint distributions, respectively. For example, the joint distribution consisting of the variables x'_0 and x'_1 takes two times $(0, 0)$, once $(1, 0)$, and once $(0, 1)$.

$$\begin{aligned} \forall a, b: \quad & P(x'_0, x'_1) = \alpha, \\ & P(x'_2, x'_3) = \beta. \end{aligned} \tag{4}$$

Definition 3 (Balance [SM21a])**.** In a masking scheme, balance means that regardless of the values of the secret variables, the value taken by each final output share is equal to 0 and 1 the same number of times.

Example. Equation (5) expresses the balance property for Eq. (2). Here $f_{c0} = f_0 + f_1$, where X represents the input variables a_0, b_0 and b_1. Similarly, $f_{c1} = f_2 + f_3$, with Y denoting the input variables a_1, b_0 and b_1. For example, the value of x_0 in Eq. (2) is two times 0 and two times 1, regardless of the values taken by a and b.

$$\begin{aligned} \forall a, b: \quad & |\{X | f_{c0}(X) = 0\}| = |\{X | f_{c0}(X) = 1\}|, \\ & |\{Y | f_{c1}(Y) = 0\}| = |\{Y | f_{c1}(Y) = 1\}|. \end{aligned} \tag{5}$$

Definition 4 (Correctness [NRR06])**.** To ensure the correctness of the masking scheme, the XOR sum of all the final output shares is equal to the target function.

Example. The relation $x_0 + x_1 = f(a,b) = ab$ holds in Eq. (6), where f is the target Boolean function.

$$\forall a, b : f_{c0} + f_{c1} = f. \tag{6}$$

Definition 5 (Uniformity [NRS11])**.** For each secret output value f^*, all possible joint distributions (consisting of all final output shares $f_{c0}, f_{c1}, \cdots$) with $f^* = f_{c0} + f_{c1} + \cdots$ are equally likely to occur.

$$\begin{aligned} & f = f^* : \\ & P(f_{c0}, f_{c1}, \cdots) = \begin{cases} c, & \text{if } (f_{c0}, f_{c1}, \cdots) \text{ is a valid distribution with } f_{c0} + f_{c1} + \cdots = f^*, \\ 0, & \text{else,} \end{cases} \end{aligned} \tag{7}$$

where c is a constant.

Example. As an example of Eq. (2), traversing all inputs and all input shares, the distribution (f_{c0}, f_{c1}) takes six times $(0,0)$ and six times $(1,1)$ when the secret output $f^* = 0$. Similarly for $f^* = 1$, the distribution (f_{c0}, f_{c1}) takes two times $(0,1)$ and two times $(1,0)$.

Based on the above techniques, searching for some masking schemes without randomness is feasible. Due to the limited search space of the linear compression layer, it is recognized that some S-boxes remain unsolved, such as the PRINCE S-box [BCG+12] and its inverse. In this paper, we propose some new techniques (e.g., the non-linear compression layer and joint masking schemes) to extend the design space of masking schemes, which not only address the above challenges but also further optimize the existing masking schemes.

3 New Techniques

This section explains how to extend the first-order masking schemes without randomness. More specifically, we propose a non-linear compression layer to extend the design space of masking schemes in Sect. 3.1, and further propose joint masking schemes for multiple Boolean functions to improve the hardware performance of masking schemes in Sect. 3.2. As an application, Sect. 3.3 gives a workflow for designing new masking-friendly S-boxes that benefit from the proposed extended first-order masking techniques.

3.1 Masking Schemes Based on a Non-linear Compression Layer

As stated in [RBN+15, CGF21], the primary role of the compression layer of masking schemes is to reduce the number of output shares when it exceeds the number of shares of each individual input. We first propose a new technique called a non-linear compression layer in Sect. 3.1.1 to achieve this goal. Section 3.1.2 then provides construction guidelines for a masking scheme utilizing a non-linear compression layer. Additionally, an optimization technique is introduced to reduce the number of registers and enhance hardware performance, especially the area and dynamic power consumption.

3.1.1 The Non-linear Compression Layer

Earlier work [RBN+15, SM21a] have used a compression layer composed of only XOR gates. In other words, there are only linear operations in the compression layer. We propose the notion of an extended compression layer, i.e., this layer is valid as long as it can compress the number of output shares, regardless of the types of logic gates. In other words, the compression layer can be linear or non-linear. Since a linear compression layer has been widely applied, we will not treat them here. Equation (8) illustrates a non-linear compression layer for a 2-input AND gate $x = f(a, b) = ab$.

$$\begin{array}{llll} f_0(a_0, b_0) & = a_0b_0 + 1 & \rightarrow x'_0 & \\ f_1(a_1, b_0) & = a_1b_0 + a_1 + 1 & \rightarrow x'_1 & x'_0x'_1 = x_0 \\ \hline f_2(a_0, b_1) & = a_0b_1 + 1 & \rightarrow x'_2 & x'_2x'_3 = x_1 \\ f_3(a_1, b_1) & = a_1b_1 + a_1 + 1 & \rightarrow x'_3 & \end{array}. \tag{8}$$

Table 1: The distribution of (x'_0, x'_1), (x'_2, x'_3) and (x_0, x_1) with respect to (a, b).

a	b	$4P(x'_0, x'_1)$				$4P(x'_2, x'_3)$				$4P(x_0, x_1)$			
		00	01	10	11	00	01	10	11	00	01	10	11
0	0	0	1	1	2	0	1	1	2	2	0	0	2
0	1	0	1	1	2	0	1	1	2	2	0	0	2
1	0	0	1	1	2	0	1	1	2	2	0	0	2
1	1	0	1	1	2	0	1	1	2	0	2	2	0

Table 2: The computation for $(a, b) = (0, 0)$.

a_0	a_1	b_0	b_1	x'_0	x'_1	x'_2	x'_3	x_0	x_1
0	0	0	0	1	1	1	1	1	1
0	0	1	1	1	1	1	1	1	1
1	1	0	0	1	0	1	0	0	0
1	1	1	1	0	1	0	1	0	0

To validate its correctness and security under the glitch-extended probing model, we list the joint distribution of intermediate values in Table 1 and give an example of how the first row (i.e., $a = 0, b = 0$) is computed in Table 2. By substituting the values of each share of primary inputs (i.e., $a_{0/1}$ and $b_{0/1}$) from Table 1 into Eq. (8), the corresponding values of x_0 and x_1 can be derived. It is evident that $x_0 + x_1 = x$, thereby satisfying the functionality of a two-input AND gate. The following is the security verification. As shown in Eq. (8), this is in line with the non-completeness of masking schemes, because only one share of each input variable is assigned to a component function $f_i(.), 0 \le i \le 3$. When a probe is placed on an output share x_0, it extends into two probes placed on x'_0 and x'_1 respectively. Regardless of the values of a and b, the distribution of $4P(x'_0, x'_1)$ consistently equals $(0, 1, 1, 2)$ in Table 1. In other words, (x'_0, x'_1) is jointly independent of the secret inputs a and b, which is compatible with an identical joint probability distribution of Definition 2. Similarly for the other output share x_1. Table 1 illustrates the balance of this scheme, i.e., regardless of the values of a and b, the output share x_0 always takes twice 0 and twice 1. The same balance property applies to x_1. The uniformity is also fulfilled, because the distribution $4P(x_0, x_1)$ takes six times $(0, 0)$ and six times $(1, 1)$ when $x = 0$, and two times $(0, 1)$ and two times $(1, 0)$ when $x = 1$.

To the best of our knowledge, this is the first time that a non-linear compression layer is applied in a masking scheme. Note that Eq. (8) showcases the feasibility of a non-linear compression layer without implying it as an optimal implementation. Since non-linear operations offer more flexibility compared to linear operations, the application of a non-linear compression layer significantly expands the design space of masking schemes and provides more flexibility.

3.1.2 Optimized Masking Schemes Based on a Non-linear Compression Layer

In a first-order masking scheme for a Boolean function of algebraic degree t, there are at least 2^t component functions in the expansion layer. Based on the flexibility of non-linear operations, a natural question is whether the compression layer can share some of the functions of the expansion layer. Inspired by this idea, we propose a model to construct a masking scheme based on a non-linear compression layer, and give an AND3 example to illustrate the advantage of our techniques. Finally, we describe the search complexity and optimization methods.

The Masking Model Based on a Non-linear Compression Layer. The model includes $m + 2$ Boolean functions, with m functions serving as component functions in the expansion layer, and 2 functions operating in the compression layer. Since the non-linear operations lead to higher computational complexity, we use a SAT solver to implement the model and find the corresponding masking scheme for a given Boolean function. More specifically, the above model is provided in Algorithm 1, where the input consists of an n-bit target Boolean function and the number of component functions, denoted as m. The output is the corresponding masking scheme or no solution. Algorithm 1 outlines the general constraints to constructing a masking scheme, which typically involves m component functions and 2 compression functions. To develop an optimized masking scheme, we simply reduce the number of component functions to a value less than 2^t, where t is the algebraic degree of the n-bit target Boolean function.

The first five lines of Algorithm 1 constrain the non-completeness, i.e., each component function contains at most one share of the secret variable. The binary variables λ_i, μ_i, υ_i, $\cdots$ determine which share is assigned to component function $f_i, 0 \le i \le m-1$. The non-completeness of $\hat{a_i}$ is modeled in Eq. (9), where the value of $\hat{a_i}$ is determined by the binary variable λ_i. More specifically, when λ_i equals 0, $\hat{a_i}$ is assigned a_0; otherwise, it is assigned the value a_1.

$$\forall i \in \{0, 1, \cdots, m-1\} : \quad \hat{a_i} = \lambda_i \cdot a_1 + (1 - \lambda_i) \cdot a_0. \tag{9}$$

Lines 6 to 7 illustrate that $m + 2$ functions are represented as their algebraic normal forms (ANFs), where there are m component functions in the expansion layer (i.e., $f_i, 0 \le i \le m-1$) and 2 compression functions (i.e., f_{c0} and f_{c1}). As an example, Eq. (10) is the corresponding model of a 4-bit component function f_0, where $l_i, 0 \le i \le 15$ determines the specific expression.

$$\begin{aligned} f_0 =& l_0 + l_1 \cdot \hat{a_0} + l_2 \cdot \hat{b_0} + l_3 \cdot \hat{c_0} + l_4 \cdot \hat{d_0} + l_5 \cdot \hat{a_0}\hat{b_0} + l_6 \cdot \hat{a_0}\hat{c_0} + l_7 \cdot \hat{a_0}\hat{d_0} + l_8 \cdot \hat{b_0}\hat{c_0} + \\ & l_9 \cdot \hat{b_0}\hat{d_0} + l_{10} \cdot \hat{c_0}\hat{d_0} + l_{11} \cdot \hat{a_0}\hat{b_0}\hat{c_0} + l_{12} \cdot \hat{a_0}\hat{b_0}\hat{d_0} + l_{13} \cdot \hat{a_0}\hat{c_0}\hat{d_0} + l_{14} \cdot \hat{b_0}\hat{c_0}\hat{d_0} + \\ & l_{15} \cdot \hat{a_0}\hat{b_0}\hat{c_0}\hat{d_0}. \end{aligned} \tag{10}$$

For each value of secret variables $a, b, c, \cdots$ (line 8), line 9 initializes five hash tables (i.e., Hu, Hd, Tu, Td and Tc) which will be used to record the joint distributions in the following operations. Since the values of component functions rely on individual shares of secret variables, line 10 iterates through each possible value of one share. Lines 11 to 15 describe the role of these hash tables, which is to record some joint distributions of component functions. More specifically, the hash table Hu serves the purpose of mapping integer keys ranging from 0 to $2^{\frac{m}{2}}-1$ to the corresponding joint distribution $(f_0, f_1, \cdots, f_{\frac{m}{2}-1})$ for each case. The values stored in Hu are the frequency of occurrence for each respective joint distribution. Similarly, hash table Hd performs the same role but records the joint distribution $(f_{\frac{m}{2}}, f_{\frac{m}{2}+1}, \cdots, f_{m-1})$. As for the hash tables Tu and Td, their keys correspond to the values of compression functions f_{c0} and f_{c1}, respectively. The frequencies of these compression functions are recorded in the respective hash table values. Following a similar approach, the hash table Tc is responsible for recording the cases of the joint distribution (f_{c0}, f_{c1}). Line 16 indicates the correctness of the masking scheme, i.e., the XOR-ed result of final outputs is equal to the target Boolean function f. The model of correctness is shown in Eq. (11), where f_{c0} and f_{c1} are the final output shares and f is the output of the primary Boolean function.

$$\begin{aligned}&\forall a||b||c||\cdots \in \{0,1,\cdots,2^n-1\}, \forall a_0||b_0||c_0||\cdots \in \{0,1,\cdots,2^n-1\}: \\ &f_{c0}+f_{c1}=f.\end{aligned} \tag{11}$$

The subsequent lines 17 to 19 indicate the identical joint distribution, which is modeled as shown in Eq. (12). The frequency of each joint distribution k (consisted of component functions $f_0, f_1, \cdots, f_{\frac{m}{2}-1}$) stored in hash table Hu remains the same regardless of the values taken by secret variable j. This operations apply similarly to joint distribution $(f_{\frac{m}{2}}, f_{\frac{m}{2}+1}, \cdots, f_{m-1})$ and the corresponding hash table Hd.

$$\begin{aligned}&\forall j \in \{0,1,\cdots,2^n-1\}, \forall k \in \left\{0,1,\cdots,2^{\frac{m}{2}}-1\right\}: \\ &Hu^0[k]=Hu^j[k], \quad Hd^0[k]=Hd^j[k].\end{aligned} \tag{12}$$

To maintain balance, lines 20 to 21 constrain that each final output (either 0 or 1) recorded in the hash table Tu and Td, respectively, has an equal frequency. The model of balance is illustrated in Eq. (13). The frequency of 0 (i.e., $Hu^j[0]$) and 1 (i.e., $Hu^j[1]$) for the first output share are equivalent (i.e., $\frac{2^n}{2}=2^{n-1}$), independent of the secret variable j. The second output share and the corresponding hash table Hd^j are similar.

$$\begin{aligned}&\forall j \in \{0,1,\cdots,2^n-1\}: \\ &Tu^j[0]=Tu^j[1], \quad Td^j[0]=Td^j[1].\end{aligned} \tag{13}$$

Lines 22 to 25 constrain the uniformity of the masking schemes, guaranteeing that the frequency of each possible joint distribution (f_{c0}, f_{c1}) is uniform. The model of uniformity is shown in Eq. (14). More specifically, when the output f equals 0, there exist only two cases for the aforementioned distribution: $(0,0)$ and $(1,1)$, with their frequencies recorded as equal in the hash table Tc. Similarly, when the output f equals 1, there are exclusively two values, $(0,1)$ and $(1,0)$, with their frequencies also being the same.

$$\forall j \in \{0,1,\cdots,2^n-1\}: \quad \begin{cases} Tc^j[0]=Tc^j[3], & \text{if } f=0\ , \\ Tc^j[1]=Tc^j[2], & \text{else.}\end{cases} \tag{14}$$

The operations as described (lines 1 to 25) establish non-completeness, correctness, identical joint distribution, balance, and uniformity, thereby ensuring the desired properties and behavior of the masking scheme. Lines 26 to 29 involve the utilization of the SAT solver to run the above model and return the result. The SAT solver aims to determine whether a satisfying assignment exists for the given model. If a solution exists, the SAT

Algorithm 1 The model: whether a masking scheme for an n-bit Boolean function with m component functions and 2 compression functions can be found

```
Input: An $n$-bit Boolean function $f$.
       $m$ : the number of component functions.
Output: Return a masking scheme or NULL.
 1: for $i = 0$ to $m - 1$ do
 2:     $\hat{a_i} = \lambda_i \cdot a_1 + (1 - \lambda_i) \cdot a_0$        ▷ Non-completeness
 3:     $\hat{b_i} = \mu_i \cdot b_1 + (1 - \mu_i) \cdot b_0$
 4:     $\hat{c_i} = \upsilon_i \cdot c_1 + (1 - \upsilon_i) \cdot c_0$
 5:     $\cdots$
 6:     $f_i = l_{i \cdot 2^n} + l_{i \cdot 2^n + 1} \cdot \hat{a_i} + l_{i \cdot 2^n + 2} \cdot \hat{b_i} + l_{i \cdot 2^n + 3} \cdot \hat{c_i} + \cdots$
 7: Model two functions with inputs $(f_0, f_1, \cdots, f_{\frac{m}{2}-1})$ $(resp., (f_{\frac{m}{2}}, f_{\frac{m}{2}+1}, \cdots, f_{m-1}))$:
        $f_{c0} = p_0 + p_1 \cdot f_0 + p_2 \cdot f_1 + p_3 \cdot f_2 + \cdots$
        $f_{c1} = q_0 + q_1 \cdot f_{\frac{m}{2}} + q_2 \cdot f_{\frac{m}{2}+1} + q_3 \cdot f_{\frac{m}{2}+2} + \cdots$
 8: for $a||b||c|| \cdots = j = 0$ to $2^n - 1$ do
 9:     Initialize five hash tables to zero, donated as $Hu^j, Hd^j, Tu^j, Td^j, Tc^j$.
10:     for $a_0||b_0||c_0|| \cdots = 0$ to $2^n - 1$ do
11:         $Hu^j[f_0||f_1|| \cdots ||f_{\frac{m}{2}-1}] = Hu^j[f_0||f_1|| \cdots ||f_{\frac{m}{2}-1}] + 1.$
12:         $Hd^j[f_{\frac{m}{2}}||f_{\frac{m}{2}+1}|| \cdots ||f_{m-1}] = Hd^j[f_{\frac{m}{2}}||f_{\frac{m}{2}+1}|| \cdots ||f_{m-1}] + 1.$
13:         $Tu^j[f_{c0}] = Tu^j[f_{c0}] + 1$
14:         $Td^j[f_{c1}] = Td^j[f_{c1}] + 1$
15:         $Tc^j[f_{c0}||f_{c1}] = Tc^j[f_{c0}||f_{c1}] + 1$
16:         $f_{c0} + f_{c1} = f$        ▷ Correctness
17:     for $k = 0$ to $2^{\frac{m}{2}} - 1$ do
18:         $Hu^0[k] = Hu^j[k]$        ▷ Identical joint distribution
19:         $Hd^0[k] = Hd^j[k]$
20:     $Tu^j[0] = Tu^j[1] = 2^{n-1}$        ▷ Balance
21:     $Td^j[0] = Td^j[1] = 2^{n-1}$
22:     if $f = 0$ then        ▷ Uniformity
23:         $Tc^j[0] = Tc^j[3]$
24:     if $f = 1$ then
25:         $Tc^j[1] = Tc^j[2]$
26: if *Solved the model by SAT solver* then
27:     return a masking scheme;
28: else
29:     return NULL
```

solver returns a masking scheme that fulfills the specified model. Conversely, if no solution is found, it indicates that no valid masking scheme satisfying the model can be derived.

An Illustrative Example of AND3 Gate Based on the Proposed Model. Using the above model, we successfully find a masking scheme for the 3-input AND gate $x = f(a, b, c) = abc$ that requires only $m = 6$ component functions in Eq. (15). It can be seen that the expansion layer contains only 6 component functions, representing a reduction of 2 component functions compared to the common scheme with a linear compression layer. Besides, the algebraic degree of both the expansion and compression layers is 2. In other words, the flexible compression layer reduces the size and decreases the number of operations of the expansion layer, making it more compact. Table 4 illustrates that we achieve a 16% (resp., 34%) reduction in area and dynamic power compared to a common scheme with a linear compression layer.

$$\begin{array}{llll}
f_0(a_0, b_1, c_0) & = a_0c_0 + b_1 & \to x'_0 & \\
f_1(a_0, b_0, c_0) & = a_0c_0 + a_0 + b_0 & \to x'_1 & \\
f_2(a_0, b_1, c_1) & = a_0c_1 + b_1 + c_1 & \to x'_2 & x'_0x'_1 + x'_0x'_2 + x'_1x'_2 = x_0 \\
\hline
f_3(a_1, b_1, c_0) & = a_1c_0 + b_1 & \to x'_3 & x'_3x'_4 + x'_3x'_5 + x'_4x'_5 = x_1 \\
f_4(a_1, b_0, c_0) & = a_1c_0 + a_1 + b_0 & \to x'_4 & \\
f_5(a_1, b_1, c_1) & = a_1c_1 + b_1 + c_1 & \to x'_5 &
\end{array}. \qquad (15)$$

The Search Complexity and Optimization Methods. While the non-linear compression layer expands the design space, it also increases the complexity of the exhaustive search. To demonstrate the practical applicability of our new technique, we provide a discussion on the search complexity and optimization method. To provide a quantitative impact, considering a 4-input Boolean function of algebraic degree 3 in Algorithm 1, there are typically $m + 2 = 2^3 + 2 = 10$ component functions. Given that the search complexity for each component function is 2^4 and two exhaustive searches for secret inputs and input shares, the overall search complexity of Algorithm 1 is at least 2^{48}. Consequently, it is inefficient to implement Algorithm 1 based on an exhaustive search.

To address the unaffordable search complexity, we propose an optimized search method that characterizes Algorithm 1 as the Boolean satisfiability problem (SAT) or satisfiability modulo theories (SMT) and then solves it with SMT/SAT solvers such as STP [GD07] and Cryptominisat [SNC09]. More specifically, taking the CVC formats of the SMT/SAT model of Algorithm 1 as input, the STP solver automates several heuristic preprocessing steps to translate the input into a CNF expression, and then invokes the SAT solver Cryptominisat to address it. Subsequently, the Cryptominisat solver capitalizes on various algorithms such as Gaussian elimination and DPLL [DP60] to optimize and solve the SAT problem. The solver efficiently searches for the desired solution in the subspace by strategically partitioning the search space. In other words, characterizing a problem that cannot be exhaustively searched into an SMT/SAT problem and then solving it using the STP solver is exactly the optimization method for solving this problem. For instance, considering the first coordinate function of the SKINNY S-box [BJK+16] outlined in Eq. (23), the exhaustive search complexity exceeds 2^{48}. Based on our optimization method (i.e., the STP solver), a desired solution can be generated within twenty minutes, equivalent to a search complexity of 2^{40} on a 2.4 GHz CPU.

3.2 Joint Masking Schemes for Multiple Boolean Functions

Due to the increased computational complexity of multiple Boolean functions, this section focuses on joint masking schemes based on the linear compression layer. In this respect, we first introduce a new technique named the share assignment strategy to extend the design space of masking schemes in Sect. 3.2.1. Further, Section 3.2.2 provides construction guidelines for a joint masking scheme utilizing the share assignment strategy, and demonstrates advantages of this joint technique, especially the area and dynamic power consumption.

3.2.1 Share Assignment Strategy

The share assignment strategy is a method to derive all possible share assignment types for a given Boolean function. The share assignment type is the case that shares of some variables are assigned to several component functions. For 2-share first-order masking schemes, Eq. (16) is one of the share assignment types that assigns shares of 4 variables to 8 component functions. To simplify the description, the subscripts for shares of each component function are written in hexadecimal notation; for Eq. (16), it is `02468BCF`.

$$\begin{array}{llll} f_0(a_0,b_0,c_0,d_0), & f_1(a_0,b_0,c_1,d_0), & f_2(a_0,b_1,c_0,d_0), & f_3(a_0,b_1,c_1,d_0), \\ f_4(a_1,b_0,c_0,d_0), & f_5(a_1,b_0,c_1,d_1), & f_6(a_1,b_1,c_0,d_0), & f_7(a_1,b_1,c_1,d_1). \end{array} \quad (16)$$

Table 3: 16 share assignment types for a component function $f_i(.)$.

No.	Share Assignment Type	Indicator
1	$f_i(a_0,b_0,c_0,d_0)$	0
2	$f_i(a_0,b_0,c_0,d_1)$	1
3	$f_i(a_0,b_0,c_1,d_0)$	2
4	$f_i(a_0,b_0,c_1,d_1)$	3
5	$f_i(a_0,b_1,c_0,d_0)$	4
6	$f_i(a_0,b_1,c_0,d_1)$	5
7	$f_i(a_0,b_1,c_1,d_0)$	6
8	$f_i(a_0,b_1,c_1,d_1)$	7
9	$f_i(a_1,b_0,c_0,d_0)$	8
10	$f_i(a_1,b_0,c_0,d_1)$	9
11	$f_i(a_1,b_0,c_1,d_0)$	A
12	$f_i(a_1,b_0,c_1,d_1)$	B
13	$f_i(a_1,b_1,c_0,d_0)$	C
14	$f_i(a_1,b_1,c_0,d_1)$	D
15	$f_i(a_1,b_1,c_1,d_0)$	E
16	$f_i(a_1,b_1,c_1,d_1)$	F

We take the Boolean function $f(a,b,c,d) = c + bd + abc$ as an example to describe the detailed share assignment strategy. Because $f(.)$ is a 4-input Boolean function with algebraic degree 3, we should derive share assignment types where shares of 4 variables (a,b,c,d) are assigned to at least 8 component functions. We denote these 8 component functions as $f_i(.)$, $0 \leq i \leq 7$. For each component function $f_i(.)$, there are 16 different share assignment types as listed in Table 3. Each type corresponds to a certain indicator.

To guarantee that component functions fulfill the non-completeness and correctness of the masking schemes, some shares or share combinations should appear in the share assignment types. For the linear term c of the Boolean function, the shares c_0 and c_1 should be assigned to different component functions. For the quadratic term bd, share combinations b_0d_0, b_0d_1, b_1d_0 and b_1d_1 should fit different component functions. Similarly, for the cubic term abc, share combinations $a_0b_0c_0$, $a_0b_0c_1$, $a_0b_1c_0$, $a_0b_1c_1$, $a_1b_0c_0$, $a_1b_0c_1$, $a_1b_1c_0$ and $a_1b_1c_1$ should appear in different component functions respectively. Based on the above shares and share combinations, without considering the order of share assignment types in different component functions, we wrote a program and obtained 196 classes of share assignment types, one of which is listed in Eq. (16).

When considering the order of share assignment types from $f_0(.)$ to $f_7(.)$, there are $8! = 40\,320$ share assignment types for each class. These results are a huge number of combinations. We propose two observations and a conclusion to reduce this number.

Observation 1. *Assuming that there are m component functions noted as $f_i(.)$, $0 \le i < m$, $f_0(.)$ to $f_{\frac{m}{2}-1}(.)$ are contributing to one output share of the target Boolean function and $f_{\frac{m}{2}}(.)$ to $f_{m-1}(.)$ correspond to the other output share. According to the definitions of identical joint probability distribution and balance, the order of share assignment types from $f_0(.)$ to $f_{\frac{m}{2}-1}(.)$ has no effect on these two properties, and similarly for $f_{\frac{m}{2}}(.)$ to $f_{m-1}(.)$.*

For example, share assignment types `02468BCF` and `20468BCF` (i.e., Eqs. (16) and (17)) are equivalent.

$$\begin{array}{llll} f_0(a_0,b_0,c_1,d_0), & f_1(a_0,b_0,c_0,d_0), & f_2(a_0,b_1,c_0,d_0), & f_3(a_0,b_1,c_1,d_0), \\ f_4(a_1,b_0,c_0,d_0), & f_5(a_1,b_0,c_1,d_1), & f_6(a_1,b_1,c_0,d_0), & f_7(a_1,b_1,c_1,d_1). \end{array} \tag{17}$$

Observation 2. *According to the definitions of correctness and uniformity, exchanging the order of the share assignment types of $(f_0(.), \cdots, f_{\frac{m}{2}-1}(.))$ and of $(f_{\frac{m}{2}}(.), \cdots, f_{m-1}(.))$ has no effect on these two properties.*

For example, share assignment types `02468BCF` and `8BCF0246` (i.e., Eqs. (16) and (18)) are equivalent.

$$\begin{array}{llll} f_0(a_1,b_0,c_0,d_0), & f_1(a_1,b_0,c_1,d_1), & f_2(a_1,b_1,c_0,d_0), & f_3(a_1,b_1,c_1,d_1), \\ f_4(a_0,b_0,c_0,d_0), & f_5(a_0,b_0,c_1,d_0), & f_6(a_0,b_1,c_0,d_0), & f_7(a_0,b_1,c_1,d_0). \end{array} \tag{18}$$

Conclusion 1. *Based on the above two observations, the number of share assignment types for one class is equal to*

$$\delta = \frac{\binom{m}{\frac{m}{2}}}{2}, \tag{19}$$

where m is the number of component functions, $\binom{m}{\frac{m}{2}}$ is the binomial coefficient.

For the Boolean function $f(a,b,c,d) = c + bd + abc$, $m = 8$, the number of concrete share assignment types for one class is equal to $\delta = \frac{\binom{m}{\frac{m}{2}}}{2} = 35$. In total, there are $196 \times 35 = 6860$ share assignment types. Compared to $196 \times 40\,320 = 7\,902\,720$, 6860 share assignment types are acceptable.

By integrating the characteristics of the target Boolean function with some observations, we obtain a comprehensive and efficient share assignment strategy that plays a critical role in the construction of masking schemes.

3.2.2 Joint Masking Schemes

Since each secret variable has 2 shares, there are at least 2^t component functions in the expansion layer of a common masking scheme, where t is the algebraic degree of the Boolean function. It seems that there is limited scope for optimizing the number of component functions from the perspective of a single Boolean function. Considering the case of an n-bit S-box, there are typical $\sum_{i=0}^{n-1} 2^{t_i}$ component functions, where t_i is the algebraic degree of the i-th coordinate function of the S-box. Since these coordinate functions depend on the same primary inputs, we noted that the masking scheme for each coordinate function may not be entirely independent. Inspired by this idea, we propose an optimization technique called joint masking schemes for multiple Boolean functions. We first present a model to construct a joint masking scheme and then give an example to illustrate the advantages of this technique.

The Model for a Joint Masking Scheme. Initially, we focus on constructing a masking scheme for a single Boolean function utilizing the share assignment strategy as proposed in Sect. 3.2.1. The construction guidelines are given in Algorithm 2. Subsequently, we provide an example of Algorithm 3, which presents a model for constructing a joint masking scheme specifically designed for two Boolean functions.

Based on the share assignment strategy in Sect. 3.2.1, Algorithm 2 presents a model for constructing a masking scheme using a linear compression layer. This model takes an n-bit Boolean function as input and generates a masking scheme as output, or indicates if no solution exists. In the first line of Algorithm 2, the algebraic degree t of a given function is determined. Utilizing the share assignment strategy, line 2 generates all possible share assignment types (denoted as sat). Since the compression function is linear, the expansion layer consists of at least 2^t component functions. As depicted in lines 3 to 4, each component function f_i is represented in the form of an ANF, e.g., Eq. (10). The input of each component function $\hat{a_i}, \hat{b_i}, \cdots$ is determined by the corresponding share assignment type sat. For compression functions, line 5 constructs two linear Boolean functions f_{c0} and f_{c1} using $(f_0, f_1, \cdots f_{2^{t-1}-1})$ and $(f_{2^{t-1}}, f_{2^{t-1}+1}, \cdots f_{2^t-1})$ as inputs, respectively. To ensure correctness, identical joint distribution, balance, and uniformity, their constraints follow similar operations as described in Algorithm 1, specifically in lines 8 to 25. In lines 7 to 10, the SAT solver is invoked to search for a masking scheme that satisfies the aforementioned model. If a solution is found, the solver returns a masking scheme; otherwise, it indicates that no solution exists.

Algorithm 2 The model: whether a masking scheme for an n-bit Boolean functions with 2^t component functions and 2 linear compression functions can be found

```
Input: An $n$-bit Boolean function $f$.
Output: Return a masking scheme or NULL.
 1: Determine the algebraic degree of the target function, $t$.
 2: Derive all possible share assignment types $sat$ in Sect. 3.2.1.
 3: for $i = 0$ to $2^t - 1$ do
 4:     $f_i = l_{i\cdot 2^n} + l_{i\cdot 2^n+1} \cdot \hat{a_i} + l_{i\cdot 2^n+2} \cdot \hat{b_i} + l_{i\cdot 2^n+3} \cdot \hat{c_i} + \cdots$
 5: Model two linear compression functions: $f_{c0} = \sum_{i=0}^{2^{t-1}-1} f_i, f_{c1} = \sum_{i=2^{t-1}}^{2^t-1} f_i$.
 6: Referring to lines 8 to 25 of Algorithm 1, constrain the masking properties of target
    functions $f$.
 7: if Solved the model by SAT solver then
 8:     return a masking scheme;
 9: else
10:     return NULL
```

Subsequently, we propose a model to design a joint masking scheme for two Boolean functions. As shown in Algorithm 3, the input of our model consists of two n-bit Boolean functions, denoted as f and g, along with a parameter s that specifies the desired number of shared component functions. The output is either a joint masking scheme or NULL. Algorithm 3 serves as a generic model for constructing a masking scheme for two Boolean functions. When the value of s is set to zero, the algorithm focuses on generating a common masking scheme without any shared component functions. Considering the optimization technique, we can easily incorporate the idea of shared component functions by specifying a value of s greater than zero.

The initial step in Algorithm 3 determines the algebraic degrees t_f and t_g of two target functions. Leveraging the model presented in Algorithm 2, the subsequent line constructs the masking models for Boolean functions f and g individually. Based on the idea of shared component functions, lines 3 to 5 impose constraints to ensure an equal contribution of shared component functions to both output shares. This means that there are $\frac{s}{2}$ shared component functions contributing to one output share, represented as $f_i = g_i$ and $f_{2^{t_f-1}+i} = g_{2^{t_g-1}+i}$, where $0 \leq i \leq \frac{s}{2} - 1$. As a joint masking scheme, Line 6 constrains the joint distribution of output shares $(f_{c0}, f_{c1}, g_{c0}, g_{c1})$ is jointly uniform, as demonstrated in lines 22 to 25 of Algorithm 1. In lines 7 to 10, the SAT solver is employed to analyze the model and provide either a joint masking scheme or indicate the absence of a solution.

Algorithm 3 The model: whether a joint masking scheme for two n-bit Boolean functions with $2^{t_f} + 2^{t_g} - s$ component functions and 4 linear compression functions can be found

```
Input: Two $n$-bit Boolean functions $f$, $g$.
       $s$: the number of shared component functions.
Output: Return a joint masking scheme or NULL.
 1: Determine the algebraic degree of target functions, $t_f, t_g$.
 2: Referring to lines 1 to 6 of Algorithm 2, construct the masking models of Boolean
    functions $f$ and $g$, respectively.
 3: for $i = 0$ to $\frac{s}{2} - 1$ do                    ▷ Shared component functions
 4:     $f_i = g_i$
 5:     $f_{2^{t_f-1}+i} = g_{2^{t_g-1}+i}$
 6: Constrain $(f_{c0}, f_{c1}, g_{c0}, g_{c1})$ is jointly uniform.      ▷ Joint uniformity
 7: if Solved the model by SAT solver then
 8:     return a joint masking scheme;
 9: else
10:     return NULL
```

An Illustrative Example of the Joint Masking Scheme. This optimization called the joint masking scheme is based on the idea of sharing certain component functions in the expansion layer among masking schemes of several Boolean functions. Equation (20) exemplifies the optimization of two masked Boolean functions, where $x = f(a, b, c, d) = ab + c$ and $y = g(a, b, c, d) = bc + d$. In addition to individual masking properties (e.g., non-completeness, identical joint probability distribution, and correctness), these schemes also exhibit joint uniformity in their outputs, denoted as (x_0, x_1, y_0, y_1).

$$
\begin{array}{llll}
f_0(a_0, b_1, c_0, d_0) & = a_0b_0 + b_0c_0 & \to x'_0 & \\
f_1(a_1, b_0, c_0, d_1) & = a_1b_0 + b_0c_0 + c_0 & \to x'_1 & x'_0 + x'_1 = x_0 \\
f_2(a_0, b_0, c_1, d_0) & = a_0b_0 + b_0c_1 + a_0 & \to x'_2 & x'_0 + x'_2 = y_0 \\
\hline
f_3(a_0, b_1, c_1, d_0) & = a_0b_1 + b_1c_1 + d_0 & \to x'_3 & x'_3 + x'_4 = x_1 \\
f_4(a_1, b_1, c_1, d_0) & = a_1b_1 + b_1c_1 + c_1 + d_0 & \to x'_4 & x'_3 + x'_5 = y_1 \\
f_5(a_0, b_1, c_0, d_1) & = a_0b_1 + b_1c_0 + a_0 + d_1 & \to x'_5 &
\end{array}
. \qquad (20)
$$

It is clear that the joint expansion layer in this context utilizes only six component functions, resulting in a reduction of two registers compared to individual masking schemes for the above two Boolean functions. Experimental results in Table 4 indicate that we achieve a 25% (resp., 27%) reduction in area and dynamic power consumption.

3.3 Application: Designing New Masking-Friendly S-boxes

As the only non-linear component in the block cipher, the S-box not only plays a crucial role in the security of the cipher but also affects the hardware performance. Therefore the S-box should be both secure and hardware-friendly. Due to the large energy and area cost to create randomness [BDPVA10, KM23], designing an S-box with a randomness-free masking scheme makes a lot of sense.

Fig. 3 shows the diagram for the construction of new masking-friendly S-boxes. The construction workflow mainly consists of three parts, i.e., generating S-boxes based on [LMC+22], constructing masking schemes using our techniques, and filtering S-boxes with excellent masking hardware performance.

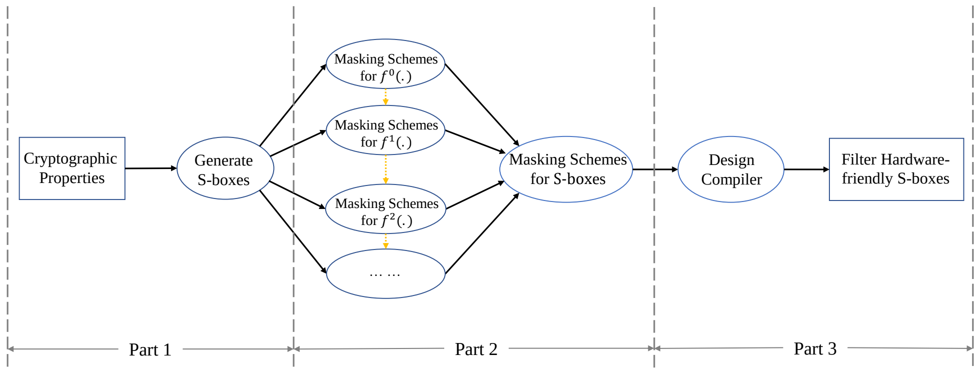

Cryptographic Properties
Generate S-boxes
Masking Schemes for $f^0(.)$
Masking Schemes for $f^1(.)$
Masking Schemes for $f^2(.)$
… …
Masking Schemes for S-boxes
Design Compiler
Filter Hardware-friendly S-boxes
Part 1
Part 2
Part 3

Figure 3: The construction of new S-boxes with friendly masking schemes.

In [LMC+22], Lu et al. proposed an STP-based model to search for S-boxes with specific cryptographic properties, such as differential uniformity [Nyb94], linearity [Nyb95], the frequency of differential uniformity (linearity) and so on. Based on the work of [LMC+22], we apply the STP-based model to construct S-boxes that exhibit the required cryptographic properties (shown in the first ellipse named "Generate S-boxes" in Fig. 3[2]). Next, we construct in part 2 masking schemes using our new techniques. Utilizing the techniques described in Sects. 3.1 and 3.2, we construct (optimized) masking schemes for each coordinate function (i.e., $f^i(.)$, $i = 0, 1, \cdots$) of the S-box. Note that due to the inherent uncertainty, every coordinate function does not always have an optimized masking scheme. If a joint uniform combination of each masked coordinate function can be found, we select this S-box and its masking scheme (shown in the ellipse named "Masking Schemes for S-boxes" in Fig. 3). In part 3, we use the EDA tool Synopsis Design Compiler to evaluate the hardware performance of masking schemes for multiple S-boxes generated in the previous parts, and filter solutions with excellent hardware performance.

4 Case Studies

This section provides some case studies where we have applied our techniques to realize first-order secure implementations of some S-boxes (e.g., PRINCE S-box [BCG+12], PRINCE S-box inverse [BCG+12], SKINNY S-box [BJK+16], Mysterion S-box [JSV17], GIFT S-box [BPP+17], PRESENT S-box [BKL+07], Midori S-box [BBI+15] and AES S-box [DR99]) and construct some new masking-friendly S-boxes.

4.1 PRINCE S-box Inverse

To the best of our knowledge, [SM21a] is the first work to propose 2-share first-order masking schemes without randomness. In [SM21a], the authors provided masking schemes

[2]The part 1 in Fig. 3 can continuously generate S-boxes with the required cryptographic properties.

without randomness for many S-boxes, but failed to find a scheme for the PRINCE S-box inverse `B732FD89A6405EC1`. Based on the coordinate functions in Eq. (21), the authors only found partial uniform combinations, i.e., joint uniformity among the first, third, and fourth masked coordinate functions, but not the second masked coordinate function. By the non-linear compression layer in Sect. 3.1.1, we solve this challenge as follows:

$$\begin{aligned}
f^0(a,b,c,d) &= 1 + d + ab + bc + cd + abd + acd,\\
f^1(a,b,c,d) &= 1 + ac + bc + bd + cd + abc,\\
f^2(a,b,c,d) &= a + c + ab + ac + bc + bd + abc + and,\\
f^3(a,b,c,d) &= 1 + a + b + ab + ac + bc + cd + abc + acd + bcd.
\end{aligned} \tag{21}$$

To the best of our knowledge, this is the first 2-share first-order secure masking scheme without randomness for the PRINCE S-box inverse. Since the non-linear compression layer leads to a higher computational cost, we try to address this challenge using the STP solver. Following the technique proposed in Sect. 3.1.2, we characterize constraints of masking schemes, including non-completeness, identical joint probability distribution, correctness, and joint uniformity. Running on a 48-core AMD EPYC 7302@2.4GHz CPU, the STP solver took 36 hours to successfully generate a masking scheme for the PRINCE S-box inverse. Since it is an automated search of the STP solver, the search time is not fixed. If there are more available solutions in the whole search space, the search time will be shorter. Excluding this solution and starting the STP solver again, more solutions can be generated. Table 4 clearly demonstrates that our work not only provides a formally secure implementation, but also offers better hardware performance (e.g., area, dynamic power and delay), compared to the related work [SM21a].

4.2 PRINCE S-box

Another case study is the PRINCE S-box `BF32AC916780E5D4`, which is also an unresolved challenge in [SM21a]. By using the share assignment strategy proposed in Sect. 3.2.1, we construct the corresponding masking schemes based on a linear compression layer. In other words, we present the first 2-share randomness-free first-order secure masking schemes for the PRINCE S-box. The scheme is described next.

$$\begin{aligned}
f^0(a,b,c,d) &= 1 + c + d + ab + bc + ad + cd + abc,\\
f^1(a,b,c,d) &= 1 + ac + bc + bd + abc + bcd,\\
f^2(a,b,c,d) &= a + d + ab + ad + bd + abd + bcd,\\
f^3(a,b,c,d) &= 1 + b + d + bc + cd + abc + abd + acd.
\end{aligned} \tag{22}$$

Since the PRINCE S-box consists of four cubic Boolean functions (shown in Eq. (22)), it is easy to determine $n = 4, t = 3$ in Algorithm 2. Applying the share assignment strategy proposed in Sect. 3.2.1 to the above four coordinate functions in Eq. (22), we found 5670, 560, 560 and 140 possible share assignment types, respectively. According to lines 3 to 5 of Algorithm 2, each function is encoded in the form of an Algebraic Normal Form (ANF). For each share assignment type, we search for specific expressions for the component functions such that all properties of masking schemes are satisfied in line 6 of Algorithm 2. Running on a 48-core AMD EPYC 7302@2.4GHz CPU, we filter out 11200, 4896, 4896 and 1536 solutions within two hours. Finally, we have found thousands of combinations that satisfy the joint uniformity. While Table 4 shows that the hardware overhead is slightly larger than that of [SM21a], the latter did not pass formal verification due to uniformity issues.

Based on our constructions for the PRINCE S-box and its inverse, we present a 2-share randomness-free round-based design of PRINCE encryption/decryption function, as depicted in Fig. 4. Due to the different assignment types for the above two S-boxes, they are implemented individually, with a multiplexer utilized to select the appropriate one. For example, for the masked second coordinate function of the PRINCE S-box, the share assignment type of the first component function is represented as (a_0, b_0, c_0, d_1), while the corresponding inverse of the PRINCE S-box is (a_0, b_1, c_1, d_0), which will leak the secret variables b, c, and d when the design of [SM21a] is applied, i.e., the component functions of the S-boxes above share a set of registers. Although this design requires a larger hardware area compared to [SM21a], it eliminates the need for randomness and achieves uniformity of the PRINCE cipher.

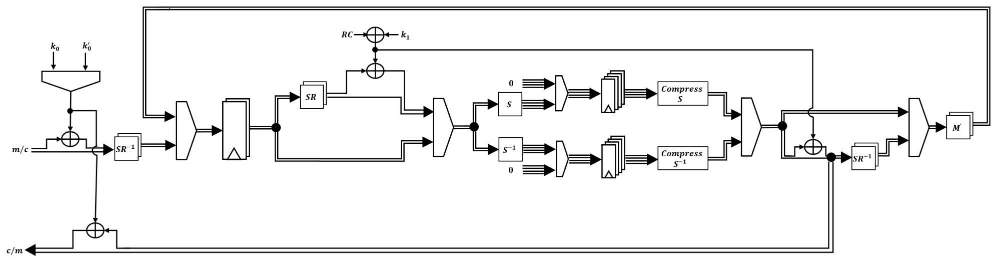


Figure 4: 2-share randomness-free masked round-based PRINCE Enc/Dec function.

4.3 Some Optimized S-boxes

It is well known that masking schemes can be very demanding in terms of hardware resources, especially in terms of area. In this section, we demonstrate the advantages of our techniques of Sects. 3.1.2 and 3.2.2 in generating optimized masking schemes of the S-boxes of SKINNY [BJK+16], Mysterion [JSV17], GIFT [BPP+17], PRESENT [BKL+07], Midori [BBI+15] and AES [DR99]. Since the processing steps are similar for each S-box, we present the details of the optimization techniques for the SKINNY S-box as an example.

$$\begin{aligned}
f^0(a,b,c,d) &= b + c + d + ab + ac + ad + bd + abc + bcd,\\
f^1(a,b,c,d) &= a + d + ab + bc + bd + cd + bcd,\\
f^2(a,b,c,d) &= 1 + b + c + d + bc,\\
f^3(a,b,c,d) &= 1 + a + c + d + cd.
\end{aligned} \tag{23}$$

It is easy to generate the coordinate functions in Eq. (23) from the look-up table of SKINNY S-box: `C6901A2B385D4E7F`. Since the cubic terms of $f^0(.)$ and $f^1(.)$ are different (one is abc, bcd and the other is bcd), we choose the optimization technique based on the non-linear compression layer in Sect. 3.1.2. Utilizing the technique in Sect. 3.2.2, we are capable of optimizing the remaining two coordinate functions $f^2(.)$ and $f^3(.)$. Based on the above CPU, it is not difficult to find desired masked component functions within four hours. Although no jointly uniform solution is found in our experiments, we find partial uniform masking schemes, i.e., the second, third, and fourth masked coordinate functions. To satisfy the uniformity property, we construct a large number of masking schemes without optimization for the first coordinate function (i.e., a non-linear compression layer and a linear compression layer both without optimization). Based on the above masking schemes, thousands of jointly uniform combinations can be successfully found. In other words, our optimization techniques are acting on the second, third, and fourth coordinate functions for the SKINNY S-box. Referring to the study of SKINNY's specification, Fig. 5 gives the schematic of the 2-share round-based SKINNY-64 encryption function without randomness. There are two register stages per cipher round. Since each output bit of the MixColumns (MC) operation is obtained by performing the XOR of at most three bits that belong to three different S-boxes, it is unnecessary to place a register following the compression layer.

In addition to the SKINNY S-box, the reduction in the number of registers yields a substantial improvement in hardware performance, as illustrated in Table 4. In particular, we achieve noteworthy reductions in both area and dynamic power for different S-boxes. For the SKINNY S-box, Mysterion S-box, GIFT S-box, PRESENT S-box and Midori S-box, the area saving amount to 21%, 24%, 15%, 14% and 3%, while the dynamic power consumption is reduced by 15%, 27%, 18%, 15% and 12% for the five above S-boxes, respectively. It is worth noting that in the case of the AES S-box, we share the randomness-free design from [SM21a], which is a tower-field approach for the inversion in $GF(2^8)$. While our techniques lead to an improvement in area (9%) and dynamic power consumption (17%) for the $GF(2^4)$ inversion (as a 4-bit S-box), it should be acknowledged that we reach the same level as [SM21a], i.e., the $GF(2^4)^2$ inversion is probing secure but not jointly uniform. Consequently, the 2-share randomness-free first-order secure masking schemes for the AES S-box remain an unsolved challenge.

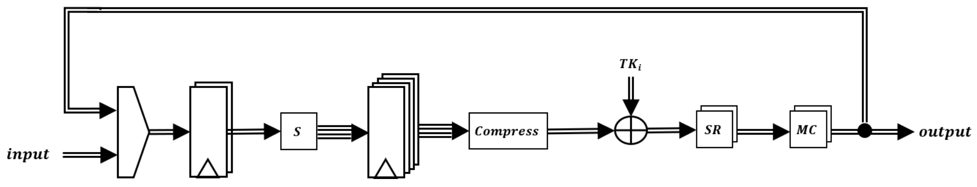


Figure 5: 2-share randomness-free masked round-based SKINNY encryption function.

Table 4: Summary of different 2 shares randomness-free first-order implementations[a].

Target	Formal Verification[b]	Technique Type	Area [GE]	Power [uW]	Delay [ns]	Reference
AND2	✓	LC[c]	32	2.04	0.14	[SM21a]
	✓	NC[d]	31	2.01	0.13	This Work
AND3	✓	LC	72	4.92	0.19	[SM21a]
	✓	NC	60	3.24	0.19	This Work
Two Boolean Functions	✓	LC	83	4.79	0.14	[SM21a]
	✓	JM[e]	63	3.50	0.14	This Work
PRINCE S-box	✗	LC	329	22.36	0.26	[SM21a]
	✓	JM	351	22.92	0.31	This Work
PRINCE S-box Inverse	✗	LC	356	21.86	0.29	[SM21a]
	✓	NC	341	20.69	0.24	This Work
Mysterion S-box	✓	LC	243	14.74	0.22	[SM21a]
	✓	NC+JM	186	10.82	0.19	This Work
SKINNY S-box	✓	LC	252	14.89	0.24	[SM21a]
	✓	NC+JM	200	12.59	0.23	This Work
GIFT S-box	✓	LC	245	15.36	0.20	[SM21a]
	✓	NC+JM	208	12.53	0.20	This Work
PRESENT S-box	✓	LC	313	21.31	0.29	[SM21a]
	✓	JM	269	18.09	0.29	This Work
Midori S-box	✓	LC	293	18.18	0.26	[SM21a]
	✓	NC	283	15.96	0.26	This Work
$GF(2^4)$ Inversion	✓	LC	337	22.53	0.28	[SM21a]
	✓	NC	306	18.72	0.27	This Work

[a] Using Synopsis Design Compiler with NanGate 45 nm.
[b] Verification of the claimed security by SILVER [KSM20].
[c] Linear compression techniques in [SM21a].
[d] Non-linear compression techniques in Sect. 3.1.
[e] Joint masking techniques in Sect. 3.2.

4.4 New S-boxes

To design masking-friendly S-boxes, we do not only consider security metrics (e.g., differential uniformity [Nyb94], linearity [Nyb95], and so on) but also cover hardware performance of the first-order masking schemes without randomness.

As an illustration, we generate a set of new S-boxes by following the workflow depicted in Fig. 3. These S-boxes possess equivalent cryptographic properties to that of the PRINCE S-box and its inverse. Based on [LMC+22], we successfully generated 60 candidate S-boxes within approximately 6 hours. Due to the inherent uncertainty involved in generating masking schemes for both the S-box and its corresponding inverse, we were successful for thirteen of these S-boxes.

Table 5 lists the look-up table descriptions of five of these S-boxes, while Table 6 lists the hardware performance of their corresponding masking schemes. Considering the hardware performance of the PRINCE S-box, S_1, S_2 and S_3 are better candidates among these S-boxes. In terms of their inverses, only S_1 is more favorable than the PRINCE S-box inverse. Based on the above considerations, S_1 is a better candidate.

Table 5: The look-up table descriptions of new S-boxes.

S-box	Look-up Table
S_1	`3158C2D9A4F0E6B7`
S_2	`B48375016CE29DAF`
S_3	`19CD574B0826FAE3`
S_4	`4E516870BA2C39FD`
S_5	`B437D6208591FCAE`

Table 6: Summary of 2 shares randomness-free 1^{st}-order implementations for new S-boxes.

S-box	Performance for S-box			Performance for S-box Inverse		
	Area [GE]	Power [uW]	Delay [ns]	Area [GE]	Power [uW]	Delay [ns]
PRINCE	351	22.92	0.31	341	20.69	0.24
S_1	338	21.33	0.27	332	21.67	0.22
S_2	340	22.19	0.28	346	22.51	0.28
S_3	349	22.56	0.34	370	24.04	0.39
S_4	350	22.07	0.30	359	24.13	0.30
S_5	351	22.16	0.37	343	21.63	0.27

5 Security Analysis

This section presents a comprehensive security analysis of our techniques, providing detailed insights into its theoretical evaluation encompassing the security assessment of individual S-boxes, full ciphers of PRINCE and SKINNY, and their automated analysis utilizing SILVER [KSM20] and PROLEAD [MM22]. Additionally, experimental evaluation is conducted through FPGA-based experiments, which encompass the complete implementations of the PRINCE and SKINNY ciphers.

5.1 Theoretical Evaluation

The theoretical security is established through three parts: the security of the S-boxes, the security of full ciphers, and their automated analysis. The security of the S-boxes is established through their construction principles. As described in Sect. 2.3, the construction of a masking scheme can be understood as the problem of component functions that satisfy certain masking properties, including non-completeness, identical joint probability distribution, correctness, and joint uniformity. Once these masking properties specifically designed to align with the hardware platform are successfully fulfilled, the masking scheme is theoretically consistent with the glitch-extended probing model. The aforementioned discussion presents a theoretical analysis of the S-boxes. The linear components can be trivially implemented to guarantee first-order security. Based on the first-order secure S-boxes and linear components, it is not difficult to construct full ciphers without degrading the security.

To substantiate the aforementioned assertion, we leverage the PRINCE and SKINNY ciphers as examples. Through Propositions 1 and 2, we prove the first-order security of full ciphers under the glitch-extended probing model.

Proposition 1. *Assuming that the masked non-linear components (i.e., S-box and S-box inverse) satisfy the first-order glitch-extended probing security and uniformity, the masked PRINCE cipher in Fig. 4 is first-order secure under the glitch-extended probing model.*

Proof.

i. **Probe the register in the S-box.** When a glitch-extended probe is placed on a register between S and *Compress* S of Fig. 4, this case can be extended as some variables stored in the round state registers that contribute to the value of the probed register. Moreover, the variables stored in the round state registers (i.e., the primary inputs or the outputs of the round function) are uniform. Since this S-box satisfies non-completeness, this probe will not detect the full shares of a secret variable.
ii. **Probe the register in the S-box inverse.** When a glitch-extended probe is placed on a register between S^{-1} and *Compress* S^{-1} of Fig. 4, this case can be represented as some variables in the round state registers that lead to the value of the probed register. Similarly to the above case, this probe does not detect two shares of any secret variable due to the S-box inverse's uniformity and non-completeness.
iii. **Probe the round state register.** The glitch-extended probe on the round state register can observe two signals, one as the output share of the round function and the other as the XOR result involving primary inputs and subkey. Since each bit of round function outputs is generated by some operations on multiple distinct bits of the above XOR result (e.g., SR, S-box and M' and so on), the effect of the above two signals on masking security can be analyzed respectively. Since each output bit of the M'-layer corresponds to an XOR operation involving three bits from distinct S-boxes, it maintains the desired masking security even without the placement of a register layer at the S-box outputs. In other words, one bit of the round function output extends to three output bits from distinct S-boxes. Since each S-box satisfies the first-order security, this signal of round function output does not reveal the secret variable, and the whole round function output is uniform. For the XOR results involving primary inputs and subkey, where one signal of them represents a share of a secret variable, it does not affect the desired security.

□

Proposition 2. *Assuming that the only masked non-linear component (i.e., S-box) is first-order glitch-extended probing secure and uniform, the masked SKINNY cipher in Fig. 5 is first-order secure under the glitch-extended probing model.*

Proof.

i. **Probe the register in the S-box.** When a glitch-extended probe is placed on the register between S and *Compress* of Fig. 5, it can be extended as some S-box inputs stored in the round state registers. Due to the non-completeness of the masked S-box, these inputs do not reveal the full shares of a secret variable.

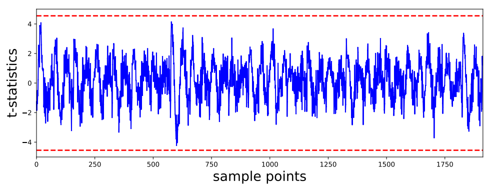
ii. **Probe the round state register.** When a glitch-extended probe is placed on the round state register, two signals can be observed: the output signals of the round function and the primary input. Given that every bit of the round function outputs is a result of some operations performed on several bits of primary inputs (e.g., SR, S-box, MC, and so on), the impact of these two signals on masking security can be analyzed, respectively. Since each output bit of MC operation is generated by performing the XOR of at most three bits from distinct S-boxes, placing a register layer at the S-box outputs is unnecessary. In other words, one bit of the round function output extends to at most three output bits from distinct S-boxes. Since each S-box exhibits first-order security and uniformity, this specific signal of the round function output does not leak any secret variables, and the entire round function outputs remain uniform. In the context of primary inputs, where a specific signal represents a share of a secret variable, it does not compromise the masking security.

□

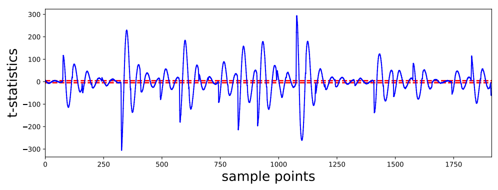
In addition to the above formal security, we also utilized the automated tools SILVER [KSM20] and PROLEAD [MM22] to verify the designs discussed in this paper. By means of SILVER [KSM20] we confirm the first-order security and uniformity of all S-box in this paper, including the PRINCE S-box, the PRINCE S-box inverse, and the SKINNY S-box. With respect to the full ciphers, we conduct PROLEAD-based assessments [MM22] (10 million simulations) to demonstrate the desired security of the PRINCE and SKINNY ciphers.

5.2 Experimental Evaluation

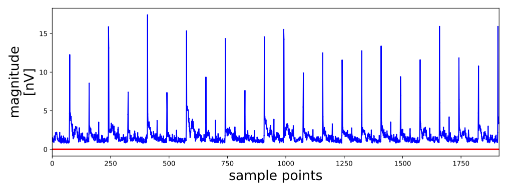
We conduct an FPGA-based experiment to verify the practical security. Our design is implemented on a SAKURA-G board [SAK] with a stable clock frequency of 3 MHz. During the runtime, the power traces are monitored with a PicoScope 5244D oscilloscope at a sampling frequency of 250 MS/s. Regarding the evaluation technique, we apply the non-specific fixed-vs-random t-test described in [SM15]. Further, we employ the confidence interval-based framework [BPG18] to identify quantifiable lower and upper bounds for the leakage. Figures 6 and 7 demonstrate the analysis results using 10 million traces at a confidence level $\alpha = 0.01$ adjusted for family-wise error rate.



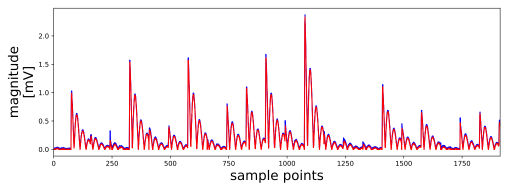
(a) PRNG ON, 1st-order t-test.



(b) PRNG OFF, 1st-order t-test.



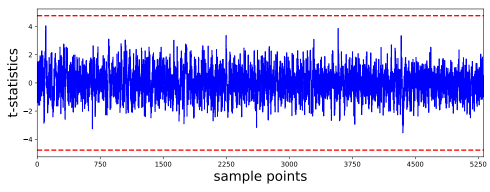
(c) PRNG ON, 1st-order confidence interval.



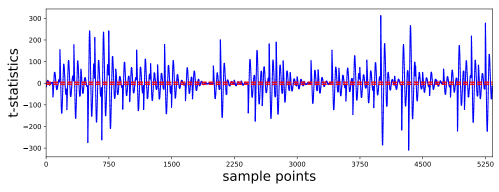
(d) PRNG OFF, 1st-order confidence interval.

Figure 6: Experimental results of our PRINCE round-based design.

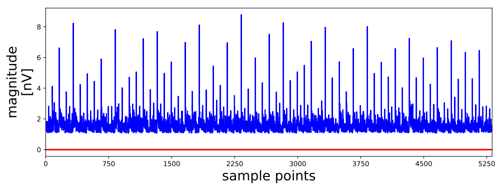
For the PRINCE design, we collected 10 million traces to conduct the t-test and confidence interval methods. As shown in Fig. 6(a), no first-order leakage can be detected in the PRINCE ciphers. Following the plots (i.e., lower bounds in red, upper bounds in blue) in Fig. 6(c), the lower bound of zero for every sample point indicates that our evaluation detected no first-order leakage. This upper bound indicates the maximum possible leakage (with a confidence level of 0.99) at each point. To validate the experimental setup, we collected 100 000 traces and repeated the above analysis when the initial masking is turned off (i.e., the mask for the initial sharing of the plaintext and the key is set to 0). As expected, Figs. 6(b) and 6(d) exhibit detectable first-order leakage.



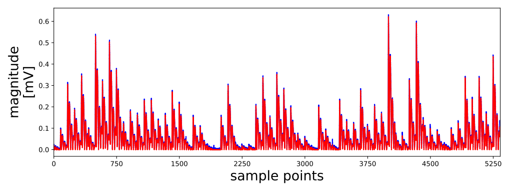
(a) PRNG ON, 1st-order t-test.



(b) PRNG OFF, 1st-order t-test.



(c) PRNG ON, 1st-order confidence interval.



(d) PRNG OFF, 1st-order confidence interval.

Figure 7: Experimental results of our SKINNY-64 round-based design.

The evaluation results of SKINNY-64 using 10 million traces are shown in Fig. 7. Figures. 7(a) and 7(b) present the results of the t-test analysis, which confirm the first-order security of our design and the validity of our experimental setup. The plots of confidence intervals depicted in Figs. 7(c) and 7(d) further reinforce the conclusion that our design exhibits no detectable first-order leakage and our experimental setup is valid.

6 Conclusions

Since first-order masking schemes have lower hardware overhead than higher-order scenarios, this paper focuses on first-order designs without randomness. In this paper, we propose two approaches to extend the randomness-free first-order masking schemes: one approach uses a non-linear compression layer and the other one uses a linear compression layer. The novel non-linear compression layer substantially expands the design space of masking schemes due to the increased flexibility of non-linear operations. For masking schemes based on a linear compression layer, we propose a comprehensive share assignment strategy allowing the construction of masking schemes for more S-boxes. Additionally, we propose optimizations for both the non-linear and linear compression approaches to reduce the hardware overhead. Compared to previous work, our work offers the following improvements. We can construct masking schemes for a wider range of S-boxes, including the PRINCE S-box and its inverse, while also achieving enhanced hardware performance for the S-boxes of SKINNY, Mysterion, and so on. Moreover, our techniques can be used to construct new masking-friendly S-boxes.

Note that our techniques are not necessarily equipped with the following composable properties: SNI [BBD+16] and PINI [CS20]. In addition to the work of [SM21a], Shahmirzadi et al. introduced a methodology to construct second-order masking schemes with almost no fresh randomness [SM21b]. Expanding our techniques to higher-order masking may bring more shared component functions and better hardware performance. Nevertheless, the computational complexity increases significantly. Consequently, proposing novel methods to adapt our proposed techniques to higher-order masking schemes is challenging and meaningful future work.

Acknowledgments

The authors would like to thank the reviewers for their valuable comments and suggestions. This work is supported by the National Key R&D Program of China (Grant No. 2023YFA1009500), the National Natural Science Foundation of China (Grant No. 62032014, U2336207), Department of Science & Technology of Shandong Province, China (No. SYS202201), Quan Cheng Laboratory (Grant No. QCLZD202301, QCLZD202306).

References

[BBD$^+$16] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, Pierre-Yves Strub, and Rébecca Zucchini. Strong non-interference and type-directed higher-order masking. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 116–129, 2016.

[BBI$^+$15] Subhadeep Banik, Andrey Bogdanov, Takanori Isobe, Kyoji Shibutani, Harunaga Hiwatari, Toru Akishita, and Francesco Regazzoni. Midori: A block cipher for low energy. In *Advances in Cryptology–ASIACRYPT 2015: 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29–December 3, 2015, Proceedings, Part II 21*, pages 411–436. Springer, 2015.

[BCG$^+$12] Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knezevic, Lars R Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar, Christian Rechberger, et al. PRINCE–a low-latency block cipher for pervasive computing applications. In *Advances in Cryptology–ASIACRYPT 2012: 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings 18*, pages 208–225. Springer, 2012.

[BDPVA10] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Sponge-based pseudo-random number generators. In *Cryptographic Hardware and Embedded Systems, CHES 2010: 12th International Workshop, Santa Barbara, USA, August 17-20, 2010. Proceedings 12*, pages 33–47. Springer, 2010.

[BJK$^+$16] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY family of block ciphers and its low-latency variant MANTIS. In *Advances in Cryptology–CRYPTO 2016: 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II 36*, pages 123–153. Springer, 2016.

[BKL$^+$07] Andrey Bogdanov, Lars R Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew JB Robshaw, Yannick Seurin, and Charlotte Vikkelsoe. Present: An ultra-lightweight block cipher. In *Cryptographic Hardware and Embedded Systems-CHES 2007: 9th International Workshop, Vienna, Austria, September 10-13, 2007. Proceedings 9*, pages 450–466. Springer, 2007.

[BPG18] Florian Bache, Christina Plump, and Tim Güneysu. Confident leakage assessment—a side-channel evaluation framework based on confidence intervals. In

2018 Design, Automation & Test in Europe Conference & Exhibition (DATE), pages 1117–1122. IEEE, 2018.

[BPP$^+$17] Subhadeep Banik, Sumit Kumar Pandey, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Yosuke Todo. GIFT: A small present: Towards reaching the limit of lightweight encryption. In *Cryptographic Hardware and Embedded Systems–CHES 2017: 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, pages 321–345. Springer, 2017.

[CGF21] Ana Covic, Fatemeh Ganji, and Domenic Forte. Circuit masking: from theory to standardization, a comprehensive survey for hardware security researchers and practitioners. *arXiv preprint arXiv:2106.12714*, 2021.

[CS20] Gaëtan Cassiers and François-Xavier Standaert. Trivially and efficiently composing masked gadgets with probe isolating non-interference. *IEEE Transactions on Information Forensics and Security*, 15:2542–2555, 2020.

[DP60] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *Journal of the ACM (JACM)*, 7(3):201–215, 1960.

[DR99] Joan Daemen and Vincent Rijmen. AES proposal: Rijndael. 1999.

[FGP$^+$18] Sebastian Faust, Vincent Grosso, SM Del Pozo, Clara Paglialonga, and F-X Standaert. Composable masking schemes in the presence of physical defaults & the robust probing model. 2018.

[GD07] Vijay Ganesh and David L Dill. A decision procedure for bit-vectors and arrays. In *Computer Aided Verification: 19th International Conference, CAV 2007, Berlin, Germany, July 3-7, 2007. Proceedings 19*, pages 519–531. Springer, 2007.

[GMK16] Hannes Groß, Stefan Mangard, and Thomas Korak. Domain-Oriented Masking: Compact Masked Hardware Implementations with Arbitrary Protection Order. In Begül Bilgin, Svetla Nikova, and Vincent Rijmen, editors, *Proceedings of the ACM Workshop on Theory of Implementation Security, TIS@CCS 2016 Vienna, Austria, October, 2016*, page 3. ACM, 2016.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In *Advances in Cryptology-CRYPTO 2003: 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003. Proceedings 23*, pages 463–481. Springer, 2003.

[JSV17] Anthony Journault, François-Xavier Standaert, and Kerem Varici. Improving the security and efficiency of block ciphers based on LS-designs. *Designs, Codes and Cryptography*, 82:495–509, 2017.

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Advances in Cryptology—CRYPTO'99: 19th Annual International Cryptology Conference Santa Barbara, California, USA, August 15–19, 1999 Proceedings 19*, pages 388–397. Springer, 1999.

[KM23] David Knichel and Amir Moradi. Composable Gadgets with Reused Fresh Masks – First-Order Probing-Secure Hardware Circuits with only 6 Fresh Masks. *Cryptology ePrint Archive*, 2023.

[Koc96] Paul C Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Advances in Cryptology—CRYPTO'96: 16th Annual International Cryptology Conference Santa Barbara, California, USA August 18–22, 1996 Proceedings 16*, pages 104–113. Springer, 1996.

[KSM20] David Knichel, Pascal Sasdrich, and Amir Moradi. SILVER–statistical independence and leakage verification. In *Advances in Cryptology–ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part I 26*, pages 787–816. Springer, 2020.

[LMC$^+$22] Zhenyu Lu, Sihem Mesnager, Tingting Cui, Yanhong Fan, and Meiqin Wang. An STP-based model toward designing S-boxes with good cryptographic properties. *Designs, Codes and Cryptography*, 90(5):1179–1202, 2022.

[MM22] Nicolai Müller and Amir Moradi. Prolead: A probing-based hardware leakage detection tool. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 311–348, 2022.

[NRR06] Svetla Nikova, Christian Rechberger, and Vincent Rijmen. Threshold implementations against side-channel attacks and glitches. In *International conference on information and communications security*, pages 529–545. Springer, 2006.

[NRS11] Svetla Nikova, Vincent Rijmen, and Martin Schläffer. Secure hardware implementation of nonlinear functions in the presence of glitches. *Journal of Cryptology*, 24:292–321, 2011.

[Nyb94] Kaisa Nyberg. Differentially uniform mappings for cryptography. In Tor Helleseth, editor, *Advances in Cryptology — EUROCRYPT '93*, pages 55–64, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[Nyb95] Kaisa Nyberg. S-boxes and round functions with controllable linearity and differential uniformity. In Bart Preneel, editor, *Fast Software Encryption*, pages 111–130, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.

[OMPR05] Elisabeth Oswald, Stefan Mangard, Norbert Pramstaller, and Vincent Rijmen. A side-channel analysis resistant description of the AES S-box. In *Fast Software Encryption: 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005, Revised Selected Papers 12*, pages 413–423. Springer, 2005.

[RBN$^+$15] Oscar Reparaz, Begül Bilgin, Svetla Nikova, Benedikt Gierlichs, and Ingrid Verbauwhede. Consolidating masking schemes. In *Advances in Cryptology–CRYPTO 2015: 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I 35*, pages 764–783. Springer, 2015.

[SAK] SAKURA. Side-channel Attack User Reference Architecture. http://satoh.cs.uec.ac.jp/SAKURA/index.html.

[SM15] Tobias Schneider and Amir Moradi. Leakage assessment methodology: A clear roadmap for side-channel evaluations. In *Cryptographic Hardware and Embedded Systems–CHES 2015: 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings 17*, pages 495–513. Springer, 2015.

[SM21a] Aein Rezaei Shahmirzadi and Amir Moradi. Re-Consolidating First-Order Masking Schemes Nullifying Fresh Randomness. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(1):305–342, 2021.

[SM21b] Aein Rezaei Shahmirzadi and Amir Moradi. Second-Order SCA Security with almost no Fresh Randomness. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(3):708–755, 2021.

[SNC09] Mate Soos, Karsten Nohl, and Claude Castelluccia. Extending sat solvers to cryptographic problems. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 244–257. Springer, 2009.

[Tri03] Elena Trichina. Combinational logic design for AES subbyte transformation on masked data. *IACR Cryptol. ePrint Arch.*, page 236, 2003.