LLTI: Low-Latency Threshold Implementations

Victor Arribas*†, Zhenda Zhang† and Svetla Nikova†
* Rambus Inc., varribas@rambus.com
† KU Leuven, imec - COSIC, Belgium
firstname.lastname@esat.kuleuven.be

Abstract*—With the enormous increase in portable cryptographic devices, physical attacks are becoming similarly popular. One of the most common physical attacks is Side-Channel Analysis (SCA), extremely dangerous due to its non-invasive nature. Threshold Implementations (TI) was proposed as the first countermeasure to provide provable security in masked hardware implementations. While most works on hardware masking are focused on optimizing the area requirements, with the newer and smaller technologies area is taking a backseat, and low-latency is gaining importance. In this work, we revisit the scheme proposed by Arribas *et al.* in TCHES 2018 to secure unrolled implementations. We formalize and expand this methodology, to devise a masking scheme, derived from TI, designed to secure hardware implementations optimized for latency named Low-Latency Threshold Implementations (LLTI). By applying the distributive property and leveraging a divide-and-conquer strategy, we split a non-linear operation in layers which are masked separately. The result is a more efficient scheme than the former TI for any operation of algebraic degree greater than two, achieving great optimizations both in terms of speed and area. We compare the performance of first-order LLTI with first-order TI in securing a cubic gate and a degree-7 AND gate without using any registers in between. We achieve a 137% increase in maximum frequency and a 60% reduction in area for the cubic gate, and 3131 times reduction in area in the case of a degree-7 AND gate compared to TI. To further illustrate the power of our scheme we take a low-latency PRINCE implementation from the literature and, by simply changing the secure S-box with the LLTI version, we achieve a 46% max. frequency improvement and a 38% area reduction. Moreover, we apply LLTI to a secure a low-latency AES implementation and compare it with the TI version, achieving a 6.9 times max. freq. increase and a 47.2% area reduction.

I. Introduction

With the exponential increase of IoT devices, Side-Channel Analysis (SCA) arises as one of the most threatening physical attacks. It exploits the side channels produced by the electronic device, such as the power consumption, the electromagnetic radiations, or the computation time. The most common attack of this kind is the Differential Power Analysis (DPA) [1]. Several strategies or countermeasures exist to help preventing side-channel attacks. One countermeasure that has caught great attention for the past years in the literature is masking [2], [3], [4], [5], [6], [7], [8], [9]. It is a technique derived from secret-sharing and multi-party computation (MPC), in which secret variables are split into multiple pieces or “shares”, to make the information leaked independent of sensitive data.

Cryptographic algorithms combine linear and non-linear operations to achieve the desired security. The only non-linear part in symmetric cryptographic primitives is usually the S-box. S-boxes have been designed with implementation costs in mind, but not taking into account the costs of side-channel resistance. A recent manuscript, from Bilgin *et al.* [10] stresses the importance of taking into account SCA resistance at design time, where the number of multiplications and the logical depth are the deciding factors to achieve efficient protected implementations. The logic depth has a direct impact on the latency of secure hardware implementations, which are prone to increase the unexpected dependencies caused by glitches [3]. Additionally, [10] analyzes larger S-boxes, with six and seven bits, limiting their investigations to quadratic operations due to the latency penalty when securing operations with a higher algebraic degree.

The perfect example of a construction with a big AND depth is the AES S-box, which has algebraic degree 7. There are multiple strategies to implement a masked AES S-box, the most common being the tower filed decomposition [11], [12], [13], [14], [8]. An also popular approach is the power maps decomposition [15], [4], [16], [17]. All these methods need several register layers to cope with the high non-linearity. To the best of our knowledge, there is no previous work securing the AES S-box by applying straightforward masking to the Algebraic Normal Form (ANF). Most of the literature regarding the masking of AES focuses on area efficient implementations. Only a few recent works proposed low-latency AES implementations [9], [18]. As chip manufacturing technology becomes smaller, the area constraints grow secondary, and lower latency implementations gain importance.

A. Related Work

The most popular type of masking used in hardware implementations is Boolean masking, where the pieces of the secret, or shares, are combined using an XOR operation to retrieve the unmasked data. Within this work, we focus on this type of masking. One of the first masked implementations is the one from Trichina [19], often referred to as the Trichina AND gate. A more formal study from Ishai, Shahai, and Wagner (ISW) [2], proposes the first security proofs and provable secure scheme for SCA protection at arbitrary order. They introduce the probing model, a widely used model in the design of masking primitives. Nevertheless, these techniques were not secure in hardware, since they did not capture in their models the hardware non-ideal behavior, *i.e.*, glitches.

The first masking scheme to provide first-order provable security in the presence of glitches was Threshold Implementations (TI) [3], later expanded to Higher-Order Threshold implementation (HOTI) [6], to cope with higher orders of security. These schemes are often referred as $td+1$ masking. In a subsequent manuscript by Reparaz *et al.* [7], it was shown that the properties defined by HOTI were not enough to protect against multivariate attacks. Consolidating Masking Schemes (CMS) [7] proposes a reduction of the number of shares to achieve the same degree of security, in line with the proposal from [2]. This reduction additionally requires independence among the input variables to preserve the security warranties. Following CMS, Domain Oriented Masking (DOM) [8], improves on the randomness needed in CMS. These schemes are often referred as $d+1$ masking. Finally, the paper from Moos *et al.* [20] shows that both CMS and DOM present flaws from third-order onwards.

On low-latency masking, the manuscript from Arribas *et al.* [21] presents the first low-latency TI applied to secure an unrolled implementation of Keccak. Following closely, Generic Low-Latency Masking (GLM) from Gross *et al.* [9] uses $d+1$ masking as the main building block. The latest work by Sasdrich *et al.* [18] uses an alternative technique based on gate-level masking, known as LUT-based Masked Dual-Rail with Pre-charge Logic (LMDPL).

B. Our Contribution

In the cases where the AND depth equals to two, it is common practice to secure each AND gate separately, placing one register in between, as done in [22]. Hence, the greater the AND depth the more registers are needed. Current masking methodologies do not provide the means to directly secure high-algebraic-degree operations, such as the one found in the AES S-box, harming the latency of the design [10]. In this manuscript, we close this gap, proposing a methodology to mask any-logic-depth circuit without a single register, while still achieving competitive area results.

We present *Low-Latency Threshold Implementations* (LLTI), a masking methodology suitable for securing low-latency implementations, derived from the well-known masking scheme TI. The scheme we propose builds upon the concepts on unrolled implementations presented in Sect. 4 from previous work by Arribas *et al.* [21]. The scheme presented by Arribas *et al.* was utilized for securing unrolled implementations, proposing the first steps to devise the masking strategy needed within each unrolled round or layer. Within this manuscript, we formalize and extend the concepts from Arribas *et al.* to optimize the masking of higher-degree operations, and prove its applicability to any order of security and any algebraic degree.

We demonstrate how, by using our methodology, we can achieve the same minimal bounds for input and output shares as proposed in TI, with more efficient implementation results. Moreover, we prove that the optimization strategy used in [21] can be used at any security order and for any algebraic degree. Our scheme provides the means to mask any-degree operation without registers in between. It achieves realistic and competitive circuit complexity, compared to the direct sharing from [3], [6]. With our scheme we can not only implement efficiently high-algebraic-degree operations, but we can also use it to optimize the area requirements and the maximum frequency. LLTI works at the algorithmic level, so no synthesis or place-and-route constraints are needed for its implementation. It is important, however, to ensure that no optimizations happen during synthesis that could harm the security of the design.

Throughout this manuscript, we use the concept of latency to refer both to the number of clock cycles (or number of register layers), and the time from input to output. The reason behind this is that different communities use this term in distinct ways. On the one hand, the masking community uses it as cycle count or registers needed within a gadget ([9], [18], [10]), while, on the other hand, the works proposing PRINCE (or other full-cipher) implementations use latency as time from input to output ([23], [24], [25]). With this new scheme we aim to optimize both. The main feature of LLTI is to reduce the number of register layers. However, its application results in significant improvements in maximum frequency with respect to TI, leading to an enhanced input-output computation time.

Traditionally, in digital design, there is always a trade-off between area and latency: improving the design latency incurs in area penalty. In this work, we show how the application of LLTI minimizes this area increase in comparison with traditional strategies (like TI) when used for low-latency applications. Moreover, we show how it additionally increases the overall speed by further reducing the maximum frequency. We illustrate the advantages of LLTI by taking primitives implemented with low-latency in mind and comparing them to the corresponding TI secured version. We can see that while keeping the (low-) latency constant, LLTI highly reduces the area and frequency compared to the TI version.

Applying the concepts of the proposed scheme, we show a straightforward example of a first-order secure cubic AND gate (regularly used in symmetric primitives). Compared to using the direct sharing of traditional TI, with LLTI we achieve a maximum frequency improvement of 137% and a reduction in area of more than 60%. Additionally, we provide a sharing of a degree-7 operation, which is, to the best of our knowledge, the first sharing suitable to mask the AES S-box without adding any registers in between. This sharing, applied on a single monomial, is more than 3 100 times smaller compared to the direct sharing if we were to use TI. To show the performance of LLTI when applied to realistic implementations, we experiment with PRINCE and AES designs. By simply swapping the secure S-box of a state-of-the-art PRINCE implementation targeting low-latency with an LLTI version, we achieve a 46% max. frequency improvement and a 38% area reduction. Moreover, we propose an LLTI AES with a single register layer and zero online randomness and compare it with the corresponding TI version, resulting in almost 7 times higher max. freq. and a reduction of 47% in area when using LLTI. We practically evaluate the security of both versions for up to 100 million traces.

The next sections of the paper are organized as follows. In Sect. II, we present more in detail related preliminary works. Then, in Sect. III, we introduce all the core concepts and proofs of our scheme. In the following Sect. IV, we illustrate different applications of our scheme. Finally, in Sect. V, we present several implementations with LLTI.

II. Preliminaries

A. Notation

In this paper, we work in $GF(2)$, meaning that sums and multiplications correspond to XOR and AND gates in hardware.

We describe any-algebraic-degree non-linear operations as AND gates of different number of inputs. We use t to denote the algebraic degree, and d for the order of security.

We focus entirely on Boolean masking, and thus, we represent a shared variable as $\boldsymbol{x} = (x_1, \ldots, x_s)$ such that $x = x_1 \oplus \ldots \oplus x_s$, where s refers to the number of shares and the number of input shares of a function. When referring to the number of output shares of a function we use s_o. Thus, the sharing of any function is denoted with the tuple (s, t, d, s_o). High algebraic degrees are designated with T, and F_{T} represents a function of such algebraic degree. Then, s_{T} refers to the input shares to this function, and d_{T} to the desired overall security order of F_{T}.

In addition to the masking notation, an equivalent set covering notation is used in the proofs presented in the following sections. We represent the corresponding set covering for the sharing tuple (s, t, d, s_o) as $\langle \mathcal{C}(s, t, d), s_o \rangle$, such that $\left|\mathcal{C}(s, t, d)\right| = s_o$. We also provide the sharing of a variable or a function with a set notation, *e.g.*, the sharing of the variable $\boldsymbol{x} = (x_1, \ldots, x_s)$ would be represented as $\{1, \ldots, s\}$. This notation is typically used in subsets of a covering set, where each subset represents the shares used in a shared function. For instance, a shared function with two output shares and three input shares could be represented as $\{\{1, 2\}, \{2, 3\}\}$, where the first shared function depends on shares 1 and 2, and the second one depends on shares 2 and 3. For clarity, we simplify this representation with the following $\{12, 23\}$.

B. Adversarial Model

The adversarial model used throughput this manuscript is the glitch extended d-probing model, first introduced in [7], and refined in [26]. It is also known as $(1, 0, 0)-$robust d-probing model [27]. Additionally, we assume independent leakage among shares. It was shown in [28], that a circuit which is secure in the d-probing model where each calculation is treated separately, is also secure against dth-order SCA.

C. Threshold Implementations

Threshold Implementations (TI) [3] require an implementation to fulfill the following three properties in order to be first-order SCA secure:

- **Correctness**: a shared function $\mathbf{f}$ such that $f_i(\boldsymbol{x}) = y_i$, where $i = 1, \ldots, s$, is correct if $\sum y_i = y = f(x)$.
- **Non-completeness**: a shared function $\mathbf{f}$ is non-complete if every component function f_i is independent of at least one input share.
- **Uniformity**: Given a uniform sharing at the input of $\mathbf{f}$, the resulting output sharing must conform a uniform distribution as well.

The paper by Bilgin *et al.* [6] extended the original version of TI to Higher-Order Threshold Implementations (HOTI), to provide resistance against higher-order attacks. The former definition of non-completeness is expanded to:

*Definition 1 (d^{th}-**Non-completeness** [6]):* A shared function $\mathbf{f}$ is d^{th}-order non-complete if any combination of up to d component functions f_i is indep. of at least one input share.

This work proposes minimum bounds for the number of input and output shares to comply with the definition above as:

$$s_i \geq td + 1, \quad s_o \geq \binom{td+1}{t}. \tag{1}$$

Example: We provide an example of a first-order TI of the AND/XOR gate $z = a \oplus bc$, with $(s, t, d, s_o) = (3, 2, 1, 3)$:

$$\begin{aligned} z_1 &= a_1 \oplus b_1c_1 \oplus b_1c_2 \oplus b_2c_1, \\ z_2 &= a_2 \oplus b_2c_2 \oplus b_2c_3 \oplus b_3c_2, \\ z_3 &= a_3 \oplus b_3c_3 \oplus b_1c_3 \oplus b_3c_1. \end{aligned} \tag{2}$$

D. Set Coverings and Non-completeness

The non-completeness property of TI is reformulated in terms of set coverings by Petrides [29]. He leverages this theory to prove new concepts, which we use in the subsequent sections to aid our proofs. Below, we summarize his contributions.

Definition 2 (non-complete set coverings [29]): A dth-order non-complete set covering $\mathcal{C}^{\mathrm{nc}}(s, t, d)$ is a set of subsets from the universe of the inputs, $\mathcal{U}_s = \{1, \ldots, s\}$, such that:

1) each t-subset of $\mathcal{U}_s$ is a subset of at least one element of $\mathcal{C}^{\mathrm{nc}}(s, t, d)$,
2) each element of $\mathcal{C}^{\mathrm{nc}}(s, t, d)$ has size at least t, and
3) a minimum of $d + 1$ elements of $\mathcal{C}^{\mathrm{nc}}(s, t, d)$ are needed to cover $\mathcal{U}_s$.

Item 1 ensures correctness, item 2 prevents redundant elements not contributing to correctness from appearing, and item 3 ensures non-completeness. Note that the set covering theory does not capture the uniformity property of TI, nor necessary remasking. There can be several such non-complete set coverings, which can be used as a guideline for obtaining a non-complete TI for any algebraic degree t. Each subset represents a shared function, and each element of the subset is a share on which the sub-function is allowed to depend.
Example: We define $\mathcal{C}^{\mathrm{nc}}(3, 2, 1) = \{12, 23, 13\}$, which is the set covering corresponding to the sharing from Eq. (2). It is trivial to see that it satisfies all the conditions from Def. 2, and that can be used to generate a non-complete sharing of $z = a + bc$.

$\mathcal{C}^{\mathrm{nc}}(s, t, d)$ is a particular non-complete covering. We denote the set of all such coverings as $\mathcal{NC}(s, t, d)$, its subset of coverings with the smallest cardinality possible as $\widehat{\mathcal{NC}}(s, t, d)$, and a non-complete set covering with minimal cardinality by $\widehat{\mathcal{C}}^{\mathrm{nc}}(s, t, d)$. Finally, $\mathcal{U}_{s,t}$ is the set of all t-subsets of $\mathcal{U}_s$.

1) Basic Results [29]: The first proposition presents the minimum number of input shares to achieve non-completeness:

Proposition 2.1: If $s \geq td + 1$ then $\mathcal{U}_{s,t} \subseteq \mathcal{NC}(s, t, d)$.
Subsequently, conclusions regarding the trivial case, where $s = td + 1$, are presented. By using the trivial value of s, we obtain non-complete set coverings with minimal cardinality:

Lemma 2.1.1: $\widehat{\mathcal{NC}}(s, t, d) = \mathcal{U}_{td+1,t}$.

Thus, a single non-complete set covering of minimal cardinality is represented as follows:

Corollary 2.1.1: $\langle \widehat{\mathcal{C}}^{\mathrm{nc}}(td + 1, t, d), \binom{td+1}{t} \rangle$.
Finally, the subsequent proposition establishes an upper bound for the size of subsets within a set covering, which represents the maximum number of shares a sub-function of a shared function can depend on.

Proposition 2.2: For every $\mathcal{S} \in \mathcal{C}^{\text{nc}}(s,t,d)$ it holds that

$$\left|\mathcal{S}\right| \leq s-(d-1)t-1. \quad (3)$$

2) Extensions and Expansions [29]: In here, Petrides looks at concepts corresponding with bounds beyond the trivial ones defined in Eq. (1). The following lemma shows how by increasing the number of input shares, the number of output shares remains constant.

Lemma 2.2.1: If $s \geq td+1$ then for every $\mathcal{C}^{\text{nc}}(s,t,d)$ there exists $\mathcal{C}^{\text{nc}}(s+1,t,d)$ of equal cardinality.
Example: The set covering from the example above $\mathcal{C}^{\text{nc}}(3,2,1)$ can be extended to $\mathcal{C}^{\text{nc}}(4,2,1) = \{124, 134, 23\}$. From Prop. 2.2 we see that in this new case $\left|\mathcal{S}\right| \leq 4-(1-1)2-1 = 3$.

Furthermore, by increasing the number of input shares, we can also reduce the number of output shares or the cardinality of the resulting set covering:

Corollary 2.2.1: For $s \geq td+1$ we have $\left|\widehat{\mathcal{C}}^{\text{nc}}(s,t,d)\right| \geq \left|\widehat{\mathcal{C}}^{\text{nc}}(s+1,t,d)\right|$.
The previous example has already minimum number of outputs, so no further increase in s will reduce the size of the set covering beyond 3. Nevertheless, this is extremely useful for higher orders, where $\left|\mathcal{C}^{\text{nc}}(s,t,d)\right| \geq s$.

In summary, Petrides defines the non-complete set covering, reaching the same basic results for the trivial case of TI as [6] in Eq. (1). This is then extended to the non-trivial case when the number of input shares is increased beyond the minimum of the trivial case, with the respective change in the number of output shares to accommodate for non-completeness and uniformity. Up to now, no systematic way of finding TI non-trivial sharings has been found [29], [30].

E. Unrolled Implementations

The method proposed by Arribas *et al.* [21] is used to secure an unrolled implementation of Keccak, where two rounds are computed within one clock cycle. To achieve this, they proposed to split the non-linear operations of the unrolled rounds into different layers. Those layers are secured separately, with some conditions precomputed to ensure that the subsequent union preserves the security requirements. This allows the designer to follow a divide-and-conquer procedure instead of solving the problem of securing a complex operation as a whole, which would be highly challenging or even impossible. Traditionally, these functions are secured placing a register in between and separately masking the smaller non-linear operations. In [21], the authors propose a methodology to secure such functions without placing registers in between. The designer is free to choose how to split the more complex non-linear operations into a total of N layers, each layer containing simpler non-linear operations. The n-th layer is denoted as R^n, its input shares s_{R^n} are expressed as s_n for better readability, the degree of security of this layer is referred as d_{R^n}, and its algebraic degree as t_{R^n}.

The idea is that every layer is secured with a different sharing scheme of a certain security order provided by the following recipe:

$$d_{R^n} = \begin{cases} d_F & \text{if } n = N, \\ t_{R^{n+1}} * d_{R^{n+1}} & \text{if } n < N \end{cases} \quad (4)$$

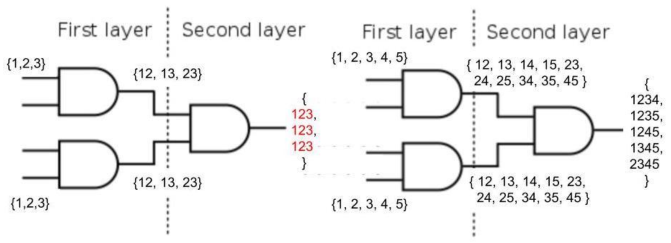


Figure 1: Layer-wise first-order sharing of a combinational quartic operation with an insecure sharing (left), and the secure counterpart (right) from applying the guidelines of Eq. (4) from [21], equivalent to a trivial LLTI with T = 4. The dashed lines represent the layers separation.

By defining the last layer's security requirements, the designer can recursively compute backwards the non-completeness degree needed in previous layers to achieve the desired security order at the output. The shares of every layer are constructed following the constraints from Eq. (4). Finally, the last layer conforms a *compression sharing*. This kind of sharing can be found for example in [31] and [13], and is defined in [21]:

*Definition 3 (**Compression sharing [21]**):* A compression sharing is characterized by having $s > s_o$.

The authors of [21] use a quartic, or degree-four gate, as an example to illustrate the application of Eq. (4). This represents the algebraic degree resulting from removing the register layer usually employed to split two quadratic rounds. This is the case of Keccak, which is the algorithm analyzed in [21]. Fig. 1 presents this example, used to show how to leverage Eq. (4) to devise the masking strategy to secure such function. On the one hand, it shows an example in which the two layers are secured independently with three input shares (left), which makes it impossible to maintain the non-completeness property given the combinations from both layers. On the other hand, the secure version is presented using five input shares, derived from applying the concepts from Eq. (4). As we can see, in the end every output share depends on four input shares out of five, ensuring non-completeness.

Finally, the unrolled implementation can also be formulated as non-complete set coverings. The covering set for layer R^n is denoted as $\langle \mathcal{C}_n(s_n, t_n, d_n), s_{o_n} \rangle$. Sometimes, when the parameters of such covering are clear from the context, or not relevant at that moment, we use $\mathcal{C}_n$ for the sake of simplicity.

F. Unbalanced sharings

Unbalanced sharings have already been used in [26] App. A, but they were not properly introduced until the work of Wei *et al.* [32], used to devise a TI implementation of SM4.

*Definition 4 (**Unbalanced sharings [32]**):* In existing sharing schemes, every input variable has the same number of shares. However, it is perfectly possible that a sharing scheme in which different input variables are split into different numbers of shares still maintains the three essential properties for TIs. Such a scheme is named an unbalanced sharing scheme.

III. Low-Latency Threshold Implementations

In this section, we discuss our methodology. We present first an example where the methodology of Arribas *et al.* [21] does not achieve optimal results, and how to improve it. As a result, we propose a new sharing for a cubic operation with no registers in between, which at the same time represents the most basic element of our scheme. Then, we introduce *Low-Latency Threshold Implementations* (LLTI). We prove that our methodology achieves minimal sharings for any security order d, and any algebraic degree t, achieving the same bounds as with TI. Additionally, we demonstrate how by increasing the number of input shares we can achieve more optimal sharings. Finally, we address the issue of uniformity and randomness.

A. Sub-optimal cases of [21]

The scheme proposed by Arribas *et al.* can be applied to any non-linear operation, but it is not directly applicable to odd prime algebraic degrees to achieve the minimal sharing. We fix this and extend our methodology to obtain minimal sharings for any degree.

To illustrate the cases where a sub-optimal sharing is achieved when following the guidelines from Eq. (4), we present an example of a decomposed simple cubic gate for first-order security. With TI we can trivially share a cubic operation with $s = 3 \cdot 1 + 1 = 4$ shares for $d = 1$ (from Eq. (1)). However, when decomposing the cubic gate, the resulting two layers are composed by quadratic operations, so the result would be $s = 5$ shares for first-order security by applying Eq. (4), compared to the four input and output shares from the direct case.

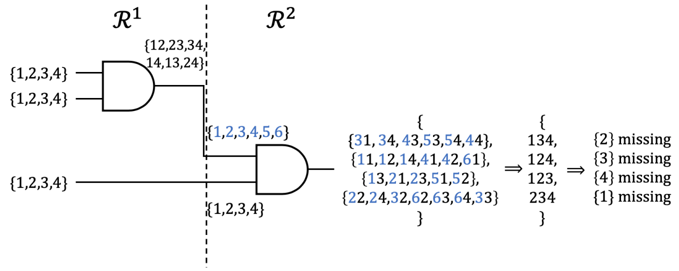


Figure 2: LLTI sharings for the two layers decomposition of a combinational cubic operation, for first-order security using unbalanced sharings with four and six shares.

In Fig. 2, we show that it is possible to achieve the minimum number of shares also from the layered decomposition, including the non-complete sharings for both layers. The two inputs to the second AND gate are split with a different number of shares: 6 and 4, respectively. The sharing in this layer is a compression sharing, and, additionally, an *unbalanced sharing*.

Unbalanced sharings are operated in the same way as regular sharings. However, note that the order of the elements within the subsets of the final covering matters, as opposed to the case when the sharings are symmetric. From the example in Fig. 2, since the bottom input is not driven by any other operation before, there are no further dependencies introduced in this line. Hence, it is enough to have six and four shares in respective inputs, instead of the 10 input shares as in the example from [21] shown in Fig. 1. As we see from the picture, the order in the indexes at the output of R^N is important, since one comes from a 6-shares sharing which at the same time has further dependencies, while the other comes from a 4-shares sharing equivalent to $\mathcal{U}_s$. Thus, the covering set for an unbalanced sharing must be expressed using all the cross-products from the two input sharings. With this example we show that the strategy proposed in [21] is not always optimal.

B. Low-Latency Threshold Implementations

In their work, De Meyer *et al.* [26] proved that non-completeness is a necessary condition for a masking scheme to provide provable security. Thus, we focus on finding efficient strategies to achieve the optimal non-complete sharing for a higher algebraic degree operation. We benefit from the set coverings theory from Sect. II-D to prove that we can achieve minimal non-complete coverings for any degree t function.

Our scheme builds upon the one in [21], providing a structured way of splitting the high-algebraic-degree non-linear operations into layers of lower ones to achieve more efficient sharings. As we have seen from the counterexample in Sect III-A, we need to take into account cases when the tree decomposition is not symmetric. Fig. 2 corresponds to such a case, namely the decomposition of a cubic gate, of prime algebraic degree. On the contrary, Fig. 1 corresponds to the decomposition of a quartic operation with a composite degree, which results in a symmetric tree of smaller operations. Thus, we differentiate between two cases to cover all possible operations: first, we define the scheme for t composite, and subsequently, we extend the method for the cases where t is prime.

1) Composite algebraic degree: Given a non-linear function with a high algebraic degree T composite, denoted as F_{T}, we can split this operation into simpler non-linear operations following its prime factors decomposition (PFD), such that

$$\mathrm{T} = \prod_{i}^{N} t_i,$$

where every t_i prime factor designates a layer R^i, for a total of N layers. Given the degree of security d_{T} of $F_{\mathrm{T}} = R^N \circ \cdots \circ R^1$, the resulting sharing for each layer has a similar structure as Eq. (4). The decomposition and the corresponding sharing for each layer are defined as follows:

$$\forall t_i \in \text{PFD}(\mathrm{T}) \text{ with } i \in [1, N],\ \exists R^i : \tag{5}$$
$$d_{R^i} = \begin{cases} d_{\mathrm{T}} & \text{if } i = N, \\ t_{R^{i+1}} * d_{R^{i+1}} & \text{if } i < N \end{cases}$$

Lemma 1: Provided that the decomposition of F_{T} is a symmetric tree of operations, the number of input shares of the first layer R^1, obtained from applying Eq. (5), is equivalent to the minimal input shares resulting from applying Eq. (1) to the function F_{T}, *i.e.*,

$$s_1 \geq t_1 \cdot d_1 + 1 = \mathrm{T} \cdot d_{\mathrm{T}} + 1.$$

In such a case, there exists always a compression sharing for the layer R^N, such that it is a minimal d_{T}^{th}-order non-complete covering $\widehat{\mathcal{C}}^{\mathrm{nc}}_N(s_N, t_N, d_N)$ with:

$$\begin{aligned}\widehat{\mathcal{C}}^{\mathrm{nc}}_N(s_N, t_N, d_N) &= \widehat{\mathcal{C}}^{\mathrm{nc}}_{F_{\mathrm{T}}}(\mathrm{T}d_{\mathrm{T}}+1, \mathrm{T}, d_{\mathrm{T}}) \Rightarrow \\ \langle \widehat{\mathcal{C}}^{\mathrm{nc}}_N(s_N, t_N, d_N) &, \binom{\mathrm{T}d_{\mathrm{T}}+1}{\mathrm{T}} \rangle.\end{aligned}$$

Proof: In the first layer R^1, the sharing scheme is a set covering in the form $\mathcal{C}_1(s_1, t_1, d_1)$, such that $s_1 \geq t_1 \cdot d_1 + 1$. From the definition in Eq. (5), we have that $d_1 = t_2 \cdot d_2$, resulting in $s_1 \geq t_1 \cdot (t_2 \cdot d_2) + 1$. If we keep going until recursively reaching the last layer, and from Eq. (1), we have that

$$s_1 \geq t_1 \cdot (t_2 \cdot \ldots (t_{N-1} \cdot (t_N \cdot d_N))) + 1 = (\prod_i^N t_i) * d_N + 1 = \mathrm{T}d_{\mathrm{T}} + 1$$

From this result, and together with Lemma 2.1.1, we conclude that the sharing of R^1 uses the minimal input shares corresponding to F_{T}.

Now we have to prove that the sharing for the last layer can be a non-complete compression sharing achieving the corresponding minimal number of output shares of F_{T}. From Def. 2 (3), we know that, for the last covering set to satisfy $d_{\mathrm{T}}^{\mathrm{th}}$-order non-completeness, we need $|\mathcal{C}_N| \geq d_{\mathrm{T}} + 1$. From Prop. 2.1, and taking the minimum number of elements at the input and output of R^N, it would result in $s_N = t_n \cdot (|\mathcal{C}_N| - 1) + 1$. Since $s_N = |\mathcal{C}_{N-1}|$, and $s_{N-1} = t_{N-1} \cdot (|\mathcal{C}_{N-1}| - 1) + 1$, we can write that $s_N = t_N \cdot t_{N-1} \cdot (|\mathcal{C}_N| - 1) + 1$. By recursively reaching the first layer, we will have that

$$s_1 = (\prod_1^N t_i) \cdot \underbrace{(|\mathcal{C}_N| - 1)}_{|\mathcal{C}_N| = d_{\mathrm{T}}+1} + 1 \Rightarrow s_1 = (\prod_1^N t_i) * d_{\mathrm{T}} + 1 = \mathrm{T}d_{\mathrm{T}} + 1. \tag{6}$$

Thus, together with the first part of the proof, we can affirm that with the resulting number of input shares in R^1, $\mathcal{C}_N(s_N, t_N, d_N) \subseteq \mathcal{NC}(s_N, t_N, d_N = d_{\mathrm{T}})$, *i.e.*, is a $(d_N = d_{\mathrm{T}})^{th}$-order non-complete covering. Finally, as the result from Eq. (6) corresponds with the minimum number of input shares, from Cor. 2.1.1 we conclude that

$$\begin{aligned}&\langle \widehat{\mathcal{C}}^{\mathrm{nc}}_N(s_N, t_N, d_N), \binom{\prod_i t_i * d_{\mathrm{T}} + 1}{\prod_i t_i} \rangle \Leftrightarrow \\ &\Leftrightarrow \langle \widehat{\mathcal{C}}^{\mathrm{nc}}_{F_{\mathrm{T}}}(\mathrm{T}d_{\mathrm{T}}+1, \mathrm{T}, d_{\mathrm{T}}), \binom{\mathrm{T}d_{\mathrm{T}}+1}{\mathrm{T}} \rangle.\end{aligned}$$

■

2) Prime algebraic degree: When T is prime, there is no further decomposition possible. In this case, to split F_{T} into layers of simpler operations, we follow the above procedure to decompose $F_{\mathrm{T}-1}$ instead, where $\mathrm{T}-1$ will always be composite. To the output of the symmetric tree derived from $F_{\mathrm{T}-1}$, we just add an extra 2-input AND gate with the last input to realize a decomposition tree for F_{T}. Fig. 3 illustrates this concept with an example for $\mathrm{T} = 7$. Hence, we have that R^{N-1} is the last layer of $F_{\mathrm{T}-1}$'s decomposition, and R^N includes the last AND gate to complete F_{T}. It is important to note that the sharing in the last layer of $F_{\mathrm{T}-1}$ is not a compression sharing anymore. In this case, the sharing in R^N is both an unbalanced sharing and a compression sharing. The sharing of the first input to the last AND gate will have $|\widehat{\mathcal{C}}^{\mathrm{nc}}_{F_{\mathrm{T}-1}}(s_{\mathrm{T}-1}, \mathrm{T}-1, d_{\mathrm{T}})|$ number of input shares, while the corresponding sharing for the second input will have s_1 input shares, conforming the unbalanced sharing. Additionally, the number of output shares will be $\binom{\mathrm{T}d_{\mathrm{T}}+1}{\mathrm{T}}$, resulting in a compression sharing.

We can apply Eq. (5) to $F_{\mathrm{T}-1}$'s decomposition to get the different sharings for the first R^{N-1} layers, with $d_{N-1} = d_N = d_{\mathrm{T}}$ to achieve a full non-complete sharing. By definition, with this split, we always have that $t_N = 2$, and $d_N = d_{\mathrm{T}}$. To ensure non-completeness is preserved when adding the last AND gate to form F_{T}, it suffices to add d_{T} additional elements to $\mathcal{U}_s^{\mathrm{T}-1}$, which refers to the number of elements in the universe of $F_{\mathrm{T}-1}$. This way it is also possible to achieve the minimum number of output shares, as shown in Fig. 2. These concepts are formalized in the following lemma:

Lemma 2: Given F_{T}, with T prime, the decomposition of layers R^1 to R^{N-1} corresponds to the decomposition of $T-1$, where the conditions from Eq. (5) for $F_{\mathrm{T}-1}$ apply, with $d_{N-1} = d_N = d_{\mathrm{T}}$. Finally, an extra layer R^N is appended to realize F_{T}, with $t_N = 2$ and $d_N = d_{\mathrm{T}}$. The sharing of R^N is both a compression and an unbalanced sharing, and it suffices to have

$$|\mathcal{U}_s^{\mathrm{T}}| = |\mathcal{U}_s^{\mathrm{T}-1}| + d_{\mathrm{T}}$$

to achieve $\widehat{\mathcal{C}}^{\mathrm{nc}}_{F_{\mathrm{T}}}(s, \mathrm{T}, d_{\mathrm{T}})$.

Proof: From Lemma 1 we know that the decomposition of $F_{\mathrm{T}-1}$ results in a minimal non-complete covering $\widehat{\mathcal{C}}^{\mathrm{nc}}_{F_{\mathrm{T}-1}}(s_{\mathrm{T}-1}, \mathrm{T}-1, d_{\mathrm{T}})$. From Eq. (1), in the minimal case we have that

$$s_{\mathrm{T}-1} = (\mathrm{T}-1) \cdot d_{\mathrm{T}} + 1 = \underbrace{\mathrm{T} \cdot d_{\mathrm{T}} + 1}_{s_{\mathrm{T}}} - d_{\mathrm{T}} \Rightarrow s_{\mathrm{T}} = s_{\mathrm{T}-1} + d_{\mathrm{T}},$$

from which we confirm that $|\mathcal{U}_s^{\mathrm{T}}| = |\mathcal{U}_s^{\mathrm{T}-1}| + d_{\mathrm{T}}$.

Then, we have to prove that with this number of input shares and following the guidelines for $F_{\mathrm{T}-1}$ we can achieve a non-complete minimal covering. From Def. 2 (2) and Prop. 2.2, for the trivial case

$$\begin{aligned}&\forall \mathcal{S} \in \widehat{\mathcal{C}}^{\mathrm{nc}}_{F_{\mathrm{T}-1}}(s_{\mathrm{T}-1}, \mathrm{T}-1, d_{\mathrm{T}}): \\ &|\mathcal{S}| = t_1 \cdot t_2 \cdot \ldots \cdot t_{R^{N-1}} = \prod_i^{R^{N-1}} t_i.\end{aligned} \tag{7}$$

Every $\mathcal{S}$ contains at most $\mathrm{T}-1$ elements of $\mathcal{U}_s^{\mathrm{T}}$, and since they conform a non-complete covering, any d_{T} combinations of sets contain at most $d_{\mathrm{T}} \cdot \prod_i^{N-1} t_i = d_{\mathrm{T}} \cdot (\mathrm{T}-1)$ elements.

Subsequently, an extra layer R^N is added to complete F_{T}. Since the non-linear operation in R^N is a simple 2-input AND gate, and the second input set covering corresponds to $\mathcal{U}_s^{\mathrm{T}}$:

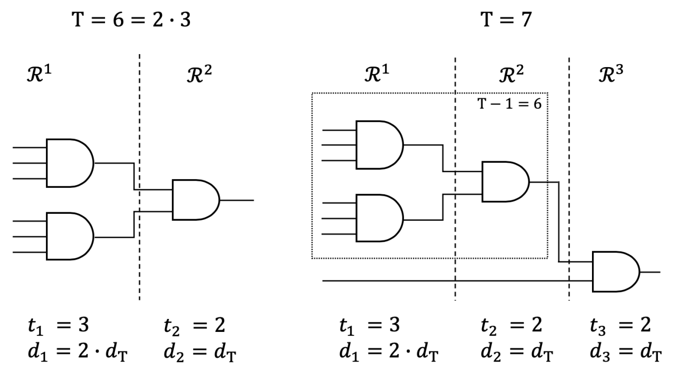
$$\forall \mathcal{S} \in \mathcal{C}_{F_{\mathrm{T}}}(s_{\mathrm{T}}, \mathrm{T}, d_{\mathrm{T}}): |\mathcal{S}| \leq t_1 \cdot t_2 \cdot \ldots \cdot t_{R^{N-1}} + 1 = \prod_i^{N-1} t_i + 1.$$

Every $\mathcal{S}$ contains at most $\mathrm{T}-1+1=\mathrm{T}$ elements. Similarly as before, any d_{T} combinations of sets will contain at most $d_{\mathrm{T}} \cdot (\prod_i^{N-1} t_i + 1) = d_{\mathrm{T}} \cdot \mathrm{T}$ components. As we have that

$$s_{\mathrm{T}} = s_{\mathrm{T}-1} + d_{\mathrm{T}} = (\mathrm{T}-1)\cdot d_{\mathrm{T}} + 1 + d_{\mathrm{T}} = \\ \mathrm{T}\cdot d_{\mathrm{T}} - d_{\mathrm{T}} + 1 + d_{\mathrm{T}} = d_{\mathrm{T}}\cdot \mathrm{T} + 1, \tag{8}$$

Def. 2 (3) is satisfied, and we conclude that $\mathcal{C}_{F_{\mathrm{T}}}(s_{\mathrm{T}}, \mathrm{T}, d_{\mathrm{T}}) \subseteq \mathcal{NC}(s_{\mathrm{T}}, \mathrm{T}, d_{\mathrm{T}})$. Finally, from this reasoning and together with Cor. 2.1.1, we conclude that indeed $\widehat{\mathcal{C}}^{\text{nc}}_{F_{\mathrm{T}}}(s_{\mathrm{T}}, \mathrm{T}, d_{\mathrm{T}})$ is a minimal non-complete set covering. ■

Fig. 3 illustrates two examples, for $\mathrm{T}=6$ and $\mathrm{T}=7$, respectively. It includes the decomposition in several layers, together with the conditions that each layer must satisfy to achieve an overall minimal non-complete covering.



(a) $\mathrm{T}=6$ (b) $\mathrm{T}=7$

Figure 3: Trivial LLTI security requirements for combinational composite and prime Ts examples, following the application of Eq. (5) and Lemma 2.

In the case of $\mathrm{T}=6$ in Fig. 3a, the trivial TI sharing for first-order security given by Eq. (1) we have $s=7$ and $s_o=7$, or $\langle \widehat{\mathcal{C}}^{\text{nc}}_{F_6}(7,6,1), \rangle$. Then, if we build the set covering following the layered decomposition, the first thing to notice is that $s_1 = 7 = s$, as stated in Lemma 1. The resulting set coverings for each layer are $\langle \widehat{\mathcal{C}}^{\text{nc}}_1(7,3,2), 35\rangle$ and $\langle \widehat{\mathcal{C}}^{\text{nc}}_2(35,2,1), 7\rangle$. For overall second-order security, with $d_{\mathrm{T}}=2$, if we calculate the trivial TI sharing for F_6, we have that:

$$s = 6\cdot 2 + 1 = 13 \tag{9}$$

$$s_o = \binom{6\cdot 2+1}{6} = 1716 \Rightarrow \langle \widehat{\mathcal{C}}^{\text{nc}}_{F_6}(13,6,2), 1716\rangle$$

Similarly, if we calculate the coverings with LLTI we have that $\langle \widehat{\mathcal{C}}^{\text{nc}}_1(13,3,4), 286\rangle$ and $\langle \widehat{\mathcal{C}}^{\text{nc}}_2(286,2,2), 1716\rangle$.

Now, we do the same exercise for $\mathrm{T}=7$ from Fig. 3b. The trivial TI sharing for first-order would have $s=8$ and $s_o=8$, or $\langle \widehat{\mathcal{C}}^{\text{nc}}_{F_7}(8,7,1), 8\rangle$. Then, the set coverings for the layered decomposition would be: for R^1, following Lemma 2,

$$\langle \widehat{\mathcal{C}}^{\text{nc}}_1(s_{\mathrm{T}-1} + d_{\mathrm{T}}, t_1, d_1) = \widehat{\mathcal{C}}^{\text{nc}}_1(7+1,2,3), \binom{8}{3} = 56\rangle;$$

for R^2,

$$\langle \widehat{\mathcal{C}}^{\text{nc}}_2(56,2,1), \binom{56}{2} = 1540\rangle;$$

and finally, for R^3, the unbalanced and compression sharing,

$$\langle \widehat{\mathcal{C}}^{\text{nc}}_3([1540,8],2,1), 8\rangle,$$

which has the minimal cardinality associated with F_7, as stated from Lemma 2. We can follow the same procedure for $d_{\mathrm{T}} = 2$. On the one hand, the TI set covering for F_7 would be $\langle \widehat{\mathcal{C}}^{\text{nc}}_{F_7}(15,7,2), 6435\rangle$. On the other hand, the set coverings corresponding to LLTI would result in $\langle \widehat{\mathcal{C}}^{\text{nc}}_1(13+2,3,4), 455\rangle$, $\langle \widehat{\mathcal{C}}^{\text{nc}}_2(455,2,2), 103285\rangle$, and $\langle \widehat{\mathcal{C}}^{\text{nc}}_3([103285,15],2,2), 6435\rangle$. With these examples, and the theory presented in this section we have shown that with LLTI we can achieve the same minimal bounds defined by TI in Eq. (1).

C. Optimizing LLTI

We should not forget that the numbers from the examples above refer to the number of shares. We can see that, already for second-order security, the cardinality of the sets grows enormously. However, these are only the examples corresponding to the trivial LLTI constructions, with the minimum number of input shares. In this section, we show how to keep the intermediate coverings cardinality from growing disproportionately by increasing the input shares. Throughout the rest of the manuscript we use trivial LLTI (presented above) to refer to LLTI constructed with sharings that strictly follow the guidelines of Eq. (1). The optimized LLTI or LLTI (presented below) refers to the construction of the scheme with non-trivial sharings, as the ones described in Sect. II-D.

1) Preventing intermediate expansions: In [29], Petrides studies more in-depth the cases where the set coverings do not correspond to the trivial cases, which correspond to directly applying Eq. (1). As we see from Cor. 2.2.1, it is possible to reduce the cardinality of a set only by increasing the number of input shares. This technique was already used in [21] to convert the $5 \rightarrow 10 \rightarrow 5$ sharing to a $6 \rightarrow 6 \rightarrow 6$ one, avoiding the expansion of the shares in between. As explained in [6], the expansion does not entail a problem for uniformity, since then the sharing goes back to the original size within the same cycle. Nevertheless, it helps to optimize the area requirements. In the following, we prove that it is possible to prevent the expansion of shares in every layer independently, in a similar fashion as in [21], for any order of security and any algebraic degree without affecting the non-completeness.

Lemma 3: Given $F_{\mathrm{T}} = R^N \circ \cdots \circ R^1$ with T composite, we can achieve constant cardinality for every set covering across the different layers by increasing the number of input shares s_1 enough, such that they are still non-complete set coverings.

Proof: Given the case where $s < s_o$, we can always achieve $\langle \mathcal{C}(s+S,t,d), s+S = s_o\rangle$ by applying Lemma 2.2.1. Applying this concept consecutively to every layer, we can achieve constant cardinality across the N layers. The non-completeness follows from applying Cor. 2.2.1 together with Lemma 1 to every layer as follows:

$$\left|\mathcal{C}^{\text{nc}}(s,t,d)\right| \geq \left|\mathcal{C}^{\text{nc}}(s+1,t,d)\right| \geq \\ \geq \left|\mathcal{C}^{\text{nc}}(s+2,t,d)\right| \geq \ldots \geq \left|\mathcal{C}^{\text{nc}}(s+S,t,d)\right|.$$

Thus, we conclude that $\mathcal{C}(s+S,t,d) \subseteq \mathcal{NC}(s+S,t,d)$ and $\langle \mathcal{C}^{\text{nc}}(s+S,t,d), s+S \rangle$. ■

For first-order security, $s = s_o$, so this technique is important to prevent intermediate expansions from growing out of control. For higher orders of security, it is always the case that $s_o > s$ for the trivial sharing, which means extra registers and fresh randomness may be needed to bring it back to $s = s_o$. Moreover, as we can see in the example from Fig. 3, the intermediate expansions of shares rapidly grows out of control. Thus, this property becomes of utmost importance in those cases to achieve sharings of reasonable sizes. For the non-trivial cases, which correspond to the cases where s does not correspond to the minimal one from Eq. (1), there is no bound for the minimal s_o anymore, and which is not straightforward to find.

2) Prime T*:* In the case where T is prime, Lemmas 2 and 3 are not enough to ensure non-completeness when seeking constant cardinality. For the non-trivial case, from Def. 2 (2) and Prop. 2.2, we have that $t_i \leq |\mathcal{S}_i| \leq s - (d_i - 1)t_i - 1 = k$ for a given R^i. Thus, Eq. (7) does not hold anymore, and the dependencies at the input of R^N are not precisely bound. The cause of this is the variability introduced by the changing number of input shares to seek constant cardinality. Layers R^1 to R^{N-1} adhere to the constraints given by Eq. (5), and hence, from Lemma 3, constant cardinality can be achieved without harming the non-completeness on those layers. Hence, only an additional constraint for layer R^N is needed. As we have seen from Lemma 2, this problem does not exist in the trivial case, since the universe of the inputs is strictly bound to $|\mathcal{U}_s^{\mathrm{T}}| = |\mathcal{U}_s^{\mathrm{T}-1}| + d_{\mathrm{T}}$. Despite not being sufficient, the guidelines from this lemma are still useful and can be used as a guide in the search for a more optimal sharing in the case of T prime. However, it does not guarantee any more that $\mathcal{C}_{F_{\mathrm{T}}}(s_{\mathrm{T}}, \mathrm{T}, d_{\mathrm{T}}) \subseteq \mathcal{NC}(s_{\mathrm{T}}, \mathrm{T}, d_{\mathrm{T}})$. It is necessary to include one additional constraint.

Proposition 1: To achieve $\mathcal{C}^{\text{nc}}_{F_{\mathrm{T}}}(s_{\mathrm{T}}, \mathrm{T}, d_{\mathrm{T}})$ for T prime, the covering at layer R^{N-1} has to satisfy that

$$\forall \mathcal{S} \subseteq \mathcal{C}^{\text{nc}}_{N-1}(s_{N-1}, N-1, d_N) : |\mathcal{S}| < \left\lceil \frac{s_1}{d_{\mathrm{T}}} \right\rceil - 1, \quad (10)$$

for $\mathcal{U}_s^{\mathrm{T}}$.

Proof: Suppose the maximum number of dependencies concerning $\mathcal{U}_s^{\mathrm{T}}$ at the output of layer R^{N-1} is denoted with $|\mathcal{S}_{N-1}|$. Since T is prime, $t_N = 2$ and the second input sharing is equivalent to $\mathcal{U}_s^{\mathrm{T}}$. Then, as we have seen in Lemma 2, the dependencies at the output of R^N are at most $|\mathcal{S}_{N-1}| + 1$, *i.e.*, $|\mathcal{S}_N| \leq |\mathcal{S}_{N-1}| + 1$. Given the security order d_{T}, we know that every output combination will have at most $(|\mathcal{S}_{N-1}| + 1) \cdot d_{\mathrm{T}}$. To ensure non-completeness, we need that $s_1 > (|\mathcal{S}_{N-1}| + 1) \cdot d_{\mathrm{T}}$, which brings us to

$$|\mathcal{S}_{N-1}| < \left\lceil \frac{s_1}{d_{\mathrm{T}}} \right\rceil - 1.$$

■

3) Achieving minimal circuit complexity: As concluded in Lemma 3, it is always possible to achieve constant covering cardinality across layers. However, this does not necessarily mean that this results in the most optimal implementation. To achieve this constant cardinality, one has to increase the number of input shares, which also impacts the linear layers (and other logic) surrounding the high-algebraic-degree non-linear operation under consideration. Thus, in some cases, it may be a better idea to moderately increase the number of inputs, such that there is still a constrained expansion of shares in the middle layers, to achieve the best compromise, taking into account the surrounding logic as well. It is then up to the designer to find a trade-off between the trivial case, with enormous expansion in the middle layers and the least number of input shares (with the corresponding necessity of compressing s_o), and the constant cardinality case, where there is no expansion in the middle layers but where the input shares increase might drastically affect the overall architecture. It is noteworthy that increasing the number of shares is one of the strategies used in [30] to achieve uniformity. Therefore, the increase in the number of input shares can lead not only to improve the area performance but also to achieve uniformity, making it worthwhile even if more surrounding logic is needed.

4) Merging the N coverings: A final step is needed when using alternative sharings different from the trivial ones. The coverings at each layer must be compatible with each other, *i.e.*, that the composition of them results in a non-complete covering. For the trivial cases from Sect. III-B this is straightforward, since input and output shares are decided based on the worst-case expansions, where every set combination corresponds to a different element in the next layer. A more careful procedure is needed in the optimized case.

D. Uniformity and Resharing

With the concepts presented above, we ensure non-completeness is fulfilled, and hence, security within the same cycle. For first-order security, it suffices to additionally satisfy uniformity to achieve overall security. To achieve uniformity there are several strategies we can follow [30], including increasing the number of input shares, adding fresh randomness, or duplicating different cross-terms, such that correctness still holds. For LLTI, we can benefit from the optimization strategy of Sect. III-C to also find a uniform sharing.

For higher orders of security, ensuring uniformity is not enough to prevent multivariate attacks, as it was shown in [33]. Hence, we need to add extra randomness for remasking to keep the security guarantees. It was noted in [20] that the security claims of the ring refreshing from CMS [7] do not hold as of third-order. Similarly for the refreshing proposed in [8], which is broken from second-order on. In light of these results, for higher-orders, we propose to use the ISW refreshing after every non-linear operation masked with LLTI and before any registers layer (if any) that may be placed in between. In [20] it is shown as well that composing TI with an SNI refreshing does not result in an SNI gadget. However, in this work we do not seek to achieve SNI-ness, but glitch-probing security. Nevertheless, we stress that this is only a recommendation, and it is not in the scope of this paper to provide an optimal refreshing. As mentioned in [8], this is still an open question. Similarly, the optimization of the additional randomness requirements for securing higher-order implementations is not within the scope of this paper. In any case, LLTI could naturally benefit from any such proposals to reduce randomness or ensure composability.

IV. Applications

As the name suggests, LLTI aims at achieving low-latency masked implementations. The main applications of LLTI are to secure higher algebraic degree functions such that no registers are needed in between, and to secure unrolled implementations. Nevertheless, it can be used to secure any composition of multiple functions. In this section, we show that we can achieve higher performance by applying LLTI already with the trivial cases (Sect. III-B) than with the classic TI. Moreover, we show that the trivial cases can be improved greatly according to Sect. III-C, to achieve yet more optimal implementations. Applying these concepts, we devise a first-order sharing for $\mathrm{T} = 7$, which is, to the best of our knowledge, the first sharing capable of directly securing the AES S-box's ANF.

A. High Algebraic Degree Functions

In this section, we show that the application of LLTI not only makes the masking of functions with $t > 2$ possible without incurring in further cycle penalties, but that the area reduction is also significant. First, we discuss the circuit complexity resulting from a masked implementation, giving an example for a simple cubic AND gate masked with TI and LLTI, and then show that the greater the security order and T, the greater is the complexity reduction when applying LLTI. Finally, we present highly optimized first-order coverings for different values of $\mathrm{T} = 5, 6, 7$.

1) Circuit complexity when masking: We can calculate the number of AND and XOR operations of a given sharing based on the number of input and output shares, and the algebraic degree. The number of AND gates and XOR gates needed to share a single monomial of degree t is given in Eq. (11):

$$\begin{aligned} \#AND &= (t-1)\cdot s^t \\ \#XOR &= s^t - s_o. \end{aligned} \tag{11}$$

The number of AND gates is given by all resulting cross-products s^t from the multiplication, all of them being degree t, which translates into $(t-1)$ multiplications per cross-product. These cross-products are XORed together and distributed among the different output shares. For the unbalanced sharing, since we have two different number of input shares, the cross-products are given by $[s_1 \cdot s_2]$, Eq. (11) resulting in:

$$\begin{aligned} \#AND &= [s_1\cdot s_2] \\ \#XOR &= [s_1\cdot s_2] - s_o. \end{aligned} \tag{12}$$

Finally, to calculate the gate numbers for a scheme applying LLTI, it suffices to apply these equations layer-wise and add the results together.

Example: To illustrate the power of LLTI, we show an example on a single cubic operation, *i.e.*, a 3-input AND gate in $GF(2)$, $F_3 = abc$. We evaluate the number of operations needed to mask it following the traditional TI (direct method), using Eqs. (1), versus the operations needed using our proposed method. Directly masking F_3, we get that

$$(s,t,d,s_o) = (4,3,1,4) \Rightarrow \begin{cases} \#AND = 2\cdot 4^3 = 128 \\ \#XOR = 4^3 - 4 = 60 \end{cases} \tag{13}$$

The number of gates needed in the example from Fig. 2 is

$$\begin{aligned} R^1: &\quad (4,2,1,6) &&\Rightarrow \begin{cases} \#AND = (2-1)\cdot 4^2 = 16 \\ \#XOR = 4^2 - 6 = 10 \end{cases} \\ R^2: &\quad ([6,4],2,1,4) &&\Rightarrow \begin{cases} \#AND = (6\cdot 4) = 24 \\ \#XOR = 6\cdot 4 - 4 = 20 \end{cases} \\ F_3: &\quad (4,3,1,4) &&\Rightarrow \begin{cases} \#AND = 40 \\ \#XOR = 30 \end{cases} \end{aligned} \tag{14}$$

From Eq. (13), we obtain that sharing a cubic operation for first-order security with a direct sharing requires 188 gates, whereas with LLTI (Eq. (14)) it needs a total of 70 gates. This entails a 62.77% reduction in circuit complexity. As described in [10], a large number of symmetric primitives include non-linear cubic operations of AND depth two, including Prince [25], Present [22], or the inverse in $GF(16)$, often used in masking the AES S-box when implemented with tower field decomposition.

2) What is the reason for this significant reduction?: The reason lies in the basics of the distributive property. While the direct method expands every cross-product and then compresses them back, LLTI expands only the elements for the first multiplication, and compresses those back before continuing with the next multiplications. The equations below present these implications for the previous first-order example of $F_3 = abc$:

$$\begin{aligned} &\mathrm{TI}: \\ abc &= (a_1 \oplus a_2 \oplus a_3 \oplus a_4)\cdot(b_1 \oplus b_2 \oplus b_3 \oplus b_4)\cdot \\ &\quad (c_1 \oplus c_2 \oplus c_3 \oplus c_4) \\ &= a_1b_1c_1 \oplus a_1b_1c_2 \oplus \ldots \oplus a_4b_4c_4 = F_3 \\ &\mathrm{LLTI}: \\ ab &= (a_1 \oplus a_2 \oplus a_3 \oplus a_4)\cdot(b_1 \oplus b_2 \oplus b_3 \oplus b_4) \\ &= a_1b_1 \oplus a_1b_2 \oplus \ldots \oplus a_4b_4 = w \\ wc &= (w_1 \oplus w_2 \oplus w_3 \oplus w_4 \oplus w_5 \oplus w_6)\cdot \\ &\quad (c_1 \oplus c_2 \oplus c_3 \oplus c_4) \\ &= w_1c_1 \oplus w_1c_2 \oplus \ldots \oplus w_6c_4 = z \end{aligned}$$

We show an extremely simple example that perfectly illustrates the reason behind this great reduction in logic gates:

$$\begin{aligned} &(1+2+3)\cdot(1+2+3) = 6\cdot 6 = 36 \quad \Rightarrow \begin{cases} \#\times = 1, \\ \#+ = 4 \end{cases} \\ &1\cdot 1 + 1\cdot 2 + 1\cdot 3 + 2\cdot 1 + 2\cdot 2 + \\ &+2\cdot 3 + 3\cdot 1 + 3\cdot 2 + 3\cdot 3 = 36 \quad \Rightarrow \begin{cases} \#\times = 9, \\ \#+ = 8 \end{cases} \end{aligned} \tag{15}$$

From this example, it is clear that the expansion of the cross-products needs more "resources" to compute the same operation. With the direct sharing, all cross-products are distributed among the output shares. In LLTI, the sharings for the smaller degree operations are computed separately, avoiding the high amount of cross-products produced with high values of t.

With Eqs. (13) and (14) we have shown the area improvements for first-order and $\mathrm{T} = 3$. In the following, we calculate some additional circuit complexity numbers for other algebraic degrees to illustrate that the area improvements are more dramatic the greater the values of T and d_{T} are. In this comparison, presented in Tab. I, we have only included the trivial cases (as described in Sect. III-B) since for them we can define the sharings without searching for them.

As we can see from Tab. I, for $\mathrm{T}=7$ we can achieve 455 times reduction for 1^{st}-order security, compared to the 2.68 from the example above with $\mathrm{T}=3$. Although most of these numbers are impractical, this comparison shows the drastic reduction we can achieve with LLTI, making the design of these sharings one step closer to a practical implementation.

3) Further optimizing the area: According to Eq. (11), the circuit complexity increases notably with the number of input shares, with more severity the greater the algebraic degree. To reduce the area of the implementation of a given F_T operation we have to minimize the number of input shares on each layer. From Prop. 2.1, we know that the direct case uses the minimum number of shares to achieve a non-complete sharing, which means that the circuit complexity for this method can not be reduced further. For LLTI, for the same reason, we cannot reduce further the area requirements of R^1. However, we can reduce the area for all subsequent layers by applying the concepts from Sect. III-C. By increasing the number of input shares of R^1, we can reduce the number of output shares, which correspond to the number of input shares for R^2. We can proceed similarly for subsequent layers, to reduce the overall costs for the implementation of the non-linear operation.

This offers a great search space for the designer, to find the best trade-offs for a particular case. The expansion of input shares in R^1 translates in the reduction of input shares for subsequent layers, and ultimately, on the reduction of the area of F_{T}. However, the increase of input shares also entails a consequent increase of shares in the surrounding logic. Hence, the designer should find a balance between the area requirements of the non-linear operation in isolation, and the whole circuit. An example of this optimization procedure can be found in [21], where the coverings $\langle\widehat{\mathcal{C}}^{\mathrm{nc}}(5,2,2),10\rangle \rightarrow \langle\widehat{\mathcal{C}}^{\mathrm{nc}}(10,2,1),5\rangle$ were optimized to $\langle\mathcal{C}^{\mathrm{nc}}(6,2,2),6\rangle \rightarrow \langle\mathcal{C}^{\mathrm{nc}}(6,2,1),6\rangle$, increasing by one the number of input and output shares.

Computing the numbers as in Tab. I for a monomial of the $6\rightarrow 6\rightarrow 6$ sharing, we get a total number of gates of 198, a 1.4 times reduction compared to the 275 from the $5\rightarrow 10\rightarrow 5$ sharing. Moreover, this entails a total of 12.6 times reduction w.r.t. TI sharing with one input share less (see the first row from Tab. I). The final implementation of [21], despite having an extra share, achieved a better overall area and maximum frequency performance compared to the trivial case.

4) The problem of finding $min(s+s_o)$: For the cases where s is not the minimal value, for first-order security, it is always the case that $s_o \geq td+1$, no matter how much one increases the number of input shares with respect to the trivial case. For higher orders, there is no longer a clear relationship between s and s_o. The problem of finding the minimum $(s+s_o)$ for $s > td+1$ was already introduced in [6]. To date, it is still an open question, and there are no formulas that establish a relationship between them. A small step forward is taken in [29], where the construction for second-order security of a quadratic operation with $s=s_o=6$ is proven to be the minimal one, *i.e.*, $\langle\widehat{\mathcal{C}}^{\mathrm{nc}}(6,2,2),6\rangle$. Additionally, for $t=3$ and $s=9$, the bound $s_o \leq 13$ is given.

This problem is equivalent to the set cover problem in combinatorics and complexity theory, considered an NP-problem, and which has been studied for a long time [34], [35], [36], [37], [38]. These works propose similar bounds for the number of elements and the minimal number of subsets needed to entirely cover them. This corresponds exactly to our problem, with the exception that we need to include the non-completeness constraint. In these works, a (v,k,t)-covering design is a collection of k-element subsets of $\{1,2,\ldots,v\}$, such that any t-element subset is contained in at least one block. In our notation, (v,k,t) corresponds to $(s,|\mathcal{S}|,t)$. The upper bounds for the coverings are given individually for every particular case, depending on v,k, and t. These bounds are found using different strategies, like greedy algorithms, induced from other coverings, with the help of finite geometry, etc. In [39], we can find the upper bounds for $t=\{2,3,4,5\}$, depending on the different values of v and k.

Similar strategies and results are needed to find optimal TI sharings, including non-completeness as a constraint, and for the parameters (s,t,d). The problem of finding the minimum bound for $s+s_o$ is beyond the scope of this manuscript and remains as an interesting subject for future works. However, our methodology can be used already to bound the search to $t\leq 3$. The layered decomposition allows a divide-and-conquer strategy, where, instead of finding the bounds for $t=6$, for instance, with the corresponding exponential increase in cross-terms, we can just search the minimum coverings for the two layers with $t=3$ and $t=2$, respectively. Once the minimal bound is established, it suffices to find a sharing within those bounds such that the merge of both coverings conforms a non-complete covering. For the case when T is prime, the algorithm should also add the constraint given in Prop. 1.

5) Optimized sharings for $\mathrm{T}=5,6,7$: Here, we present highly optimized first-order non-complete coverings for $\mathrm{T}=5,6,$ and, 7, following the guidelines from Sect. III-C to reduce the gate requirements further compared to the ones presented in Tab. I. To the best of our knowledge, these are the first sharings given for such algebraic degrees. To allow checking the validity of our results, we provide the reader the algorithms we used to build these coverings, the detailed constructions, and corresponding non-completeness checks in [40].

For the intermediate sharings, we use sharings from previous papers or take the coverings from the set covering literature. The covering sets described in [39] can be found in [41]. Note that these coverings are not constructed with the non-completeness property in mind, so it is important to check whether they satisfy it. While tailored bounds to our problem are still missing, this database is a great source for guiding the designer to get the sharing needed. Nevertheless, smaller coverings may be achieved by specifically searching the targeted set coverings with non-completeness as an initial constraint.

a) $\mathrm{T}=5$: It suffices to increase the input shares by one (as from Lemma 2) with respect to the optimal case for $\mathrm{T}=4$ from [21] to find a non-complete covering. The covering for the first layer is taken from the coverings of [41]. We choose a covering set such that it is second-order non-complete, according to the constraints from Eq. (5). Furthermore, the

Table I: Gate complexity comparison between TI and the trivial cases of LLTI for T = 4, 5, 6, 7 applied to a single monomial, for first- and second-order security, obtained by applying Eqs. (11) and (12).

	TI		trivial LLTI		Reduction
	Covering	Size (g)	Coverings	Size (g)	(g(TI)/g(LLTI))
$d_{\mathrm{T}}=1$					
T = 4	$\langle\widehat{\mathcal{C}}^{\mathrm{nc}}(5,4,1),5\rangle$	2495	$2\times\langle\widehat{\mathcal{C}}^{\mathrm{nc}}(5,2,2),10\rangle$ $\langle\widehat{\mathcal{C}}^{\mathrm{nc}}(10,2,1),5\rangle$	275	×9
T = 5	$\langle\widehat{\mathcal{C}}^{\mathrm{nc}}(6,5,1),6\rangle$	39K	$2\times\langle\widehat{\mathcal{C}}^{\mathrm{nc}}(6,2,2),15\rangle$ $\langle\widehat{\mathcal{C}}^{\mathrm{nc}}(15,2,1),105\rangle$ $\langle\widehat{\mathcal{C}}^{\mathrm{nc}}([105,6],2,1),6\rangle$	1.7K	×23
T = 6	$\langle\widehat{\mathcal{C}}^{\mathrm{nc}}(7,6,1),7\rangle$	706K	$2\times\langle\widehat{\mathcal{C}}^{\mathrm{nc}}(7,3,2),35\rangle$ $\langle\widehat{\mathcal{C}}^{\mathrm{nc}}(35,2,1),7\rangle$	4.4K	×160
T = 7	$\langle\widehat{\mathcal{C}}^{\mathrm{nc}}(8,7,1),8\rangle$	14.7M	$2\times\langle\widehat{\mathcal{C}}^{\mathrm{nc}}(8,3,2),56\rangle$ $\langle\widehat{\mathcal{C}}^{\mathrm{nc}}(56,2,1),1540\rangle$ $\langle\widehat{\mathcal{C}}^{\mathrm{nc}}([1540,8],2,1),8\rangle$	32.3K	×455
$d_{\mathrm{T}}=2$					
T = 4	$\langle\widehat{\mathcal{C}}^{\mathrm{nc}}(9,4,2),126\rangle$	26K	$2\times\langle\widehat{\mathcal{C}}^{\mathrm{nc}}(9,2,4),36\rangle$ $\langle\widehat{\mathcal{C}}^{\mathrm{nc}}(36,2,2),126\rangle$	2.7K	×9.6
T = 5	$\langle\widehat{\mathcal{C}}^{\mathrm{nc}}(11,5,2),462\rangle$	804K	$2\times\langle\widehat{\mathcal{C}}^{\mathrm{nc}}(11,2,4),55\rangle$ $\langle\widehat{\mathcal{C}}^{\mathrm{nc}}(55,2,2),1485\rangle$ $\langle\widehat{\mathcal{C}}^{\mathrm{nc}}([1485,11],2,2),462\rangle$	37K	×22
T = 6	$\langle\widehat{\mathcal{C}}^{\mathrm{nc}}(13,6,2),1716\rangle$	29M	$2\times\langle\widehat{\mathcal{C}}^{\mathrm{nc}}(13,3,4),286\rangle$ $\langle\widehat{\mathcal{C}}^{\mathrm{nc}}(286,2,2),1716\rangle$	174K	×167
T = 7	$\langle\widehat{\mathcal{C}}^{\mathrm{nc}}(15,7,2),6435\rangle$	1.2G	$2\times\langle\widehat{\mathcal{C}}^{\mathrm{nc}}(15,3,4),455\rangle$ $\langle\widehat{\mathcal{C}}^{\mathrm{nc}}(455,2,2),103285\rangle$ $\langle\widehat{\mathcal{C}}^{\mathrm{nc}}([103285,15],2,2),6435\rangle$	3.4M	×353

choice of this covering has to comply with the restriction from Prop. 1, so that the dependencies created in the second layer are within the required bounds. Finally, the unbalanced and compression sharing outputs the minimum number of shares. The resulting set coverings for each layer are:

$$\begin{aligned} R^1 :&\quad \langle\mathcal{C}_1(7,2,2),7\rangle \\ R^2 :&\quad \langle\mathcal{C}_2(7,2,1),21\rangle \\ R^3 :&\quad \langle\mathcal{C}_3([21,7],2,1),7\rangle \end{aligned}$$

According to Eqs. (11) and (12), the total gates required to realize this sharing is 546, 71 times smaller than the TI sharing, and 3 times smaller than the trivial case.

b) T = 6: To design this covering we follow the layered decomposition for T = 6 from Fig. 3a. For the first layer we need a second-order non-complete covering for $t_1 = 3$. Petrides [29] precisely proposes in his work such a covering, so we start from this set. Since second-order non-completeness is ensured at the output of R^1, the compression sharing for the second layer is therefore constructed ensuring that there is always one element from the input share missing:

$$\begin{aligned} R^1 :&\quad \langle\mathcal{C}_1(9,3,2),13\rangle \\ R^2 :&\quad \langle\mathcal{C}_2(13,2,1),9\rangle \end{aligned} \tag{16}$$

The total number of AND and XOR gates to implement this sharing is 4 677 gates. This entails a total of 151 times reduction with respect to TI, given in Tab. I. In this case, however, the number of gates needed for the LLTI trivial case is smaller.

We can go even further and apply LLTI again in the first two cubic AND gates (See Fig. 3), to reduce the area of the first layer. Now we have a case analogous to that of Fig. 2, where we decompose R^1 into R^{11} and R^{12}. Finding the set covering for this case is easier since we already have the output shares indexes of R^{12}, defined by the output shares of the former R^1. It remains to find a suitable covering for R^{11}, such that when added the dependency from applying R^{12}, the final combination sets can be distributed into the outputs of R^1. R^{11} is devised by manually reducing the trivial sharing (with $s_o = \binom{9}{2} = 36$) as much as possible to 32 output sets. Hence, the R^1 set covering $\langle\mathcal{C}_1(9,3,2),13\rangle$ can be split as follows:

$$\begin{aligned} R^{11} :&\quad \langle\mathcal{C}_{11}(9,2,2),32\rangle \\ R^{12} :&\quad \langle\mathcal{C}_{12}([32,9],2,2),13\rangle \end{aligned} \tag{17}$$

The combined number of AND and XOR gates to implement the corresponding sharing counting R^2 from Eq. (16) is 1 715, a final reduction of more than 400 times with respect to TI, and more than 2.5 times reduction with respect to the LLTI trivial case and the previous LLTI optimized case.

c) T = 7: As shown in Fig. 3b, the constraints for the first two layers are the same as in T = 6. The trivial bound according to Eq. (1) is $s = 8$, which is equivalent to the outcome of applying Lemma 2. We may increase the number of inputs by one, and proceed to calculate the set coverings of every layer with $s = 9$. The sharing we used from Petrides for the previous case also satisfies these constraints. However, it does not comply with the requirement from Prop. 1. The dependencies after the first layer are $\left|\mathcal{S}\right| = 5$ number of shares, which after the second non-linear operation turns into at most 10 dependencies with respect to the input shares. From Eq. (10), the covering from the second layer must have $\left|\mathcal{S}\right| < \frac{9}{1} - 1 = 8 \not> 10$, with respect to $\mathcal{U}_s$. To find an alternative covering we go back to the database from [41]. We could not find any set with $s = 9$ such that the previous condition is fulfilled, so we go up to $s = 10$, where we succeed to find a suitable non-complete covering. To construct the R^2 sharing, we can again search for a non-complete covering with 30 inputs from [41], or construct it ourselves. We decide to construct the covering set ourselves with non-completeness in mind. We know that the maximum number of dependencies at the output of R^2 is $\left|\mathcal{S}\right| = 8$, and that the last layer adds just one extra dependency. Thus, ensuring that all output shares of R^2 have at least two shares missing from $\mathcal{U}_s$ will allow us achieve overall non-completeness. We calculate the dependencies of each combination at R^2 and get the complementary set ($\mathcal{S}^c$) with respect to $\mathcal{U}_s$. Then, we look at all different combinations such that $\left|\mathcal{S}^c\right| = 2$, and define them as the output shares baskets of R^2. By definition, the last layer adds a dependency to each element, which means that $\left|\mathcal{S}_3\right| = 9$. Since $s = 10$ it suffices to identify a single missing share at every output share of R^3 and distribute the cross-products according to this missing index dependency. A more detailed step-by-step commented code to generate this sharing is included in [40]. The set covering corresponding to the three layers is:

$$\begin{aligned} R^1 : &\quad \langle \mathcal{C}_1(10,3,2),30\rangle \\ R^2 : &\quad \langle \mathcal{C}_2(30,2,1),45\rangle \\ R^3 : &\quad \langle \mathcal{C}_3([45,10],2,1),10\rangle \end{aligned} \tag{18}$$

Calculating the gate complexity with Eqs. (11), (12), we need a total of 8 585 gates to construct the covering defined in Eq. (18). This entails a 3.76 times reduction with respect to LLTI's trivial case, and a total of 1718 times reduction compared to TI.

Similarly as with $\mathrm{T} = 6$, we reduce this covering further by again applying LLTI on the first cubic gates. By applying Eq. (10) to R^{11}, we obtain that $\left|\mathcal{S}\right| = 3$ is allowed. However, with such a sharing, when the extra dependency added from R^{12} is taken into account, it is impossible to distribute all combinations according to R^1 output shares. Thus, R^{11} is just the trivial sharing, where $s_o = \binom{10}{2} = 45$, and $\left|\mathcal{S}\right| = 2$. Finally, $\langle \mathcal{C}_1(10,3,2),30\rangle$ can be split in:

$$\begin{aligned} R^{11} : &\quad \langle \mathcal{C}_{11}(10,2,2),45\rangle \\ R^{12} : &\quad \langle \mathcal{C}_{12}([45,10],2,2),30\rangle \end{aligned}$$

Achieving an overall gate count of 4 695, taking into account R^2 and R^3 from Eq. (18). This means a reduction of 1.8 time with respect to the previous case, a reduction of more than x6 times comparing with the trivial case, and a final reduction of 3131 times with respect to TI.

Table II: Summary of final first-order LLTI coverings for $\mathrm{T} = 3, 5, 6, 7$ applied to a single monomial.

Degree	Layer	Covering	Size	Reduction times (w.r.t)	
			(gates)	TI	Trivial LLTI
$\mathrm{T} = 3$	R^1 R^2	$\langle \mathcal{C}_1(4,2,1),6\rangle$ $\langle \mathcal{C}_2([6,4],2,1),4\rangle$	70	×2.68	
$\mathrm{T} = 5$	R^1 R^2 R^3	$\langle \mathcal{C}_1(7,2,2),7\rangle$ $\langle \mathcal{C}_2(7,2,1),21\rangle$ $\langle \mathcal{C}_3([21,7],2,1),7\rangle$	546	×71	×3
$\mathrm{T} = 6$	R^{11} R^{12} R^2	$\langle \mathcal{C}_{11}(9,2,2),32\rangle$ $\langle \mathcal{C}_{12}([32,9],2,2),13\rangle$ $\langle \mathcal{C}_2(13,2,1),9\rangle$	1 715	×400	×2.5
$\mathrm{T} = 7$	R^{11} R^{12} R^2 R^3	$\langle \mathcal{C}_{11}(10,2,2),45\rangle$ $\langle \mathcal{C}_{12}([45,10],2,2),30\rangle$ $\langle \mathcal{C}_2(30,2,1),45\rangle$ $\langle \mathcal{C}_3([45,10],2,1),10\rangle$	4 695	×3131	×6.9

Within this section we have proposed highly optimized first-order sharings for $\mathrm{T} = 3, 5, 6, 7$ ($\mathrm{T} = 4$ already proposed in [21]), summarized in Tab. II. The results presented in this section can be used to mask higher algebraic degrees of complex constructions efficiently, even if the AND depth is greater than one. Moreover, the covering corresponding to $\mathrm{T} = 7$ can be used to directly mask the AES S-box ANF without registers, which is, to the best of our knowledge, the first sharing to allow it.

B. *Securing a full cipher with LLTI*

Previous sections focus on how to apply LLTI to a non-linear function, which is the key element to secure in a cipher. Below, we provide a recipe on how to apply LLTI to an entire cipher to achieve a fully secured implementation:

- **Number of shares**: when implementing an SCA-secure design, we first need to determine the desired security order. The non-linear operation is the limiting factor of the implementation, *i.e.* the most difficult element to secure. Thus, the security order together with the algebraic degree of the non-linear function will define the number of shares to employ in the design. By plugging these parameters in Eq. (5) and following the instructions from Sect. III, we can obtain the complete sharing of the non-linear function and the number of shares for the overall implementation. In this work, we have presented a first-order sharing for each basic multiplication of algebraic degree up to $t = 7$, which are included in [40]. These sharings can be use directly to secure any function of such algebraic degree.
- **Refreshing**: then, we should include a refreshing layer to the masked non-linear function, as described in Sect. III-D.
- **Full implementation**: finally, since linear operations are performed share-wise, every linear block from the design should have the same number of copies as number of shares in the design. Similarly, MUXes, buses, and Primary Inputs and Outputs should be also replicated as many times as the number of resulting shares from the application of LLTI to the non-linear operation.

Overall, securing a cipher with LLTI follows the same major steps as with any masking scheme, with the key differences in how to secure the non-linear layer.

C. Unrolled Implementations

The second application possible for LLTI is to secure unrolled implementations, as it was done in the work of Arribas *et al.* [21], where they proposed a first-order secure unrolled implementation of Keccak.

The typical design flow to secure a (simple) S-box, is to take the Algebraic Normal Form (ANF) and substitute each variable with the shared representation, distributing the cross-terms among the output shares respecting the security requirements to realize the shared function. This is impossible to do with unrolled implementations, since we cannot get a single equation to define two (or more) rounds at once. Hence, the only alternative to mask the unrolled expression is by masking each round independently and then merging them, as in LLTI.

The equivalent overall degree t for a single round of the unrolled implementation is calculated by looking at the corresponding non-linear operations of each layer. Then, the strategy from Sect. III is applied to look for the sharings. The linear operations can be trivially included in the covering of the corresponding layer, since they do not create dependencies among different shares. They may create additional dependencies among different variables of the inputs within one share, but even if they do so, that is still not a problem since TI does not require independent inputs.

V. Implementations and Case Studies

In this section we present the results of applying LLTI to different primitives. We begin with the multiplication, the most basic non-linear operation in symmetric key primitives. Then, we present results for widely used ciphers, including example for PRINCE and AES.

A. Platform and Evaluation

The synthesis results were obtained with the Synopsis Design Compiler v2013.12, using the NANGATE 45nm Open Cell Library [42]. The `exact_map` compile option is used to prevent optimization across modules. Our implementation is deployed on a Xilinx Spartan6 FPGA on a Sakura-G evaluation board. For the synthesis, the `KEEP_HIERACHY` option is enabled in Xilinx ISE to prevent optimization across modules and shares in particular. The power measurements is taken on a dedicated output on Sakura-G board.

We perform a non-specific leakage detection test (TVLA) from Goodwill *et al.* [43]. The power measurements are split in two sets: the first set $\mathcal{S}_0$ receives fixed plaintexts and the second set $\mathcal{S}_1$ random plaintexts. The two sets of measurements are compared using the t-test statistic:

$$t = \frac{\mu(\mathcal{S}_0) - \mu(\mathcal{S}_1)}{\sqrt{\frac{\sigma^2(\mathcal{S}_0)}{|\mathcal{S}_0|} + \frac{\sigma^2(\mathcal{S}_1)}{|\mathcal{S}_1|}}} \tag{19}$$

The t-test verifies whether two populations have the same *mean*, *i.e.* its null hypothesis H_0 states that "the sets $\mathcal{S}_0$ and $\mathcal{S}_1$ are drawn from populations with the same mean." Large absolute values of this t-statistic indicate that the null hypothesis can be rejected with a high degree of confidence.

In order to verify that our setup is sound, we first perform TVLA to an unprotected implementation, by turning off the PRNG. With 100 000 power traces, we see clear evidence of leakage (see Fig. 5, left). When we turn on the PRNG, the masked AES by LLTI is secure up 100 million traces. (see Fig. 5, right). Second- and third-order leakage are detected, as expected. A threshold level of 4.5 is often used to decide whether the hypothesis should be rejected. In practice, this threshold should not be considered a hard limit. Fig. 5 shows that the first-order t-statistic reaches the "threshold" at some time samples. However, Fig. 4 demonstrates clearly that the first-order t-statistic does not grow over 100 million traces, and thus that we can say with confidence that the implementation is first-order secure with this many traces.

B. Multiplication

To show the advantages of our methodology, we implement several masked AND gates following the guidelines proposed in the previous section. We have implemented a cubic AND gate and a degree-7 AND gate, for both TI and LLTI for first-order security. Additionally, we have implemented the cubic AND gate with the method explained in [21] for comparison purposes. Tab. III presents these results. As in [9], we refer to a combinational circuit as a circuit needing 0 clock cycles to compute the result. Similarly, a circuit with a single register layer needs 1 clock cycle. For the security analysis, we evaluate non-completeness and uniformity with VerMI [44], and, additionally, include a mutual information (MI) analysis.

Table III: Comparison between TI and LLTI for several protected AND gates implementations for first-order of security.

	Method	Area $[GE]$	Rand.* $[bpc]$	# Reg. layers	Max. Freq. $[MHz]$	NC	Unif.	MI
				T = 3				
AND3 4sh.	TI	258	3	0	690	✓	✓	0.1379
AND3 5sh.	Rhythm.	134	4	0	1538	✓	✓	0.1399
AND3 4sh.	LLTI	**97**	3	0	1639	✓	✓	0.1379
				T = 7				
AND7 4sh.	TI	709	9	1	1176	✓	?	?
AND7 4sh.	LLTI	**357**	9	1	**1613**	✓	?	?
AND7 10sh.	LLTI	**5466**	≤ 9	**0**	375	✓	?	?

* Cost of fresh random bits per cycle (bpc) during computation.

The first three rows of Tab. III depict the results for the masked cubic gate, for TI, [21], and LLTI, respectively. With LLTI, the area complexity is reduced a 62% and a 28% w.r.t respective other techniques, as expected from the calculations made in Sect. IV-A. Furthermore, with this method we also achieve a 137% increase in the maximum frequency with LLTI compared to TI. In the case of TI, numerous cross-products have to be compressed to the given number of output shares, creating a deep XOR gates tree. With LLTI the total number of cross-products to compress is highly reduced, which leads to the corresponding improvement in maximum frequency. To ensure uniformity we add some extra bits of randomness. Note that in the examples we are only masking a single AND gate, which is highly unbalanced by definition. For other functions, it may be possible to reduce this number with (or even without) increasing the number of shares. Finally, we have calculated the mutual information of these examples. We have calculated as well the MI of an insecure 2-bit AND gate with simply two shares and a secure version with three shares from [30] to serve as reference. The results are 0.393 and 0.311, respectively. We can see how the implementations from Tab. III considerably reduce the MI information compared to these cases.

The second part of Tab. III presents the different implementations for a 7-input AND gate. The last row implementation corresponds to the function illustrated in Fig. 3b. The previous two rows implement the same function but with a register layer between R^1 and R^2, which leave only cubic operations at both sides of the register. To mask these cubic gates we benefit from the results of masking the previous cubic gates, the first version using TI, and the second using LLTI, both with 4 shares. The area complexity is almost 50% smaller for the case with LLTI, and the maximum frequency is 37% larger.

The last implementation is a masked 7-input AND gate without any register in between. So far, there is no masked implementation of the AES S-box purely combinational. Here we only present the implementation of a single monomial of the S-box ANF, but this implementation can be trivially leveraged to implement the remaining monomials to realize a full masked S-box implementation evaluated within the same clock cycle. As we can see, the reduction in maximum frequency is quite significant. Not only the critical path is split in two by the register, but also the number of shares is greatly reduced, leading to a great reduction of XOR gates needed to compress all the cross-products. It is important to note that this notorious decrease in the frequency would not happen in a full implementation, since this operation would no longer belong to the critical path. Hence, the designer should ponder what is the impact of that one cycle reduction, and how much is the maximum frequency affected.

For these implementations, it was not feasible to evaluate the uniformity nor the MI. Since the 7-bit AND gate can be decomposed in three cubic operations, as shown in Fig. 3, we propose a total of 9 random bits, although further analysis is required since this composition does not guarantee uniformity.

C. PRINCE

To further illustrate the power of LLTI, we take existing masked implementations of PRINCE using TI and exclusively modify the TI cubic sharing with the LLTI version proposed in Fig. 2. In particular, we use the one from [24] with the lowest latency. PRINCE has been designed with latency in mind, unfortunately threatened by the use of registers that typically appear in decomposed SBoxes or in $d+1$-like masking. Thus, PRINCE highly benefits from the use of LLTI as we will see in the following. Multiple previous works in the literature have presented side-channel protected implementations of PRINCE, like [23], [24], [25].

As we can see from the first two rows in Tab. IV, while maintaining the same number of cycles, we increase the maximum frequency by 46% and reduce the computation time of the implementation by 31% with simply applying LLTI to the protected S-box. Furthermore, this also reduces the total area by 38% compared to the TI version. Since LLTI improves the performance of the core element of the implementation, any shared implementation of the multiple ciphers in the literature using a cubic function would also result in similar performance optimizations.

Table IV: SCA protected PRINCE implementations.

PRINCE 1^{st} order SCA	Method	Area $[GE]$	Rand. $[bpc]$	Cycles	Max. Freq. $[MHz]$	Time (ns)
1. No S-box dec.	LLTI	25857	48	**12**	**488**	**24.6**
2. No S-box dec. [24]*	TI	41628	48	**12**	335	35.8
3. No S-box dec. [25]	$d+1$	11596	48	24	376	63.8
4. S-box dec. [45]	TI	14153	**0**	36	268	134
5. S-box dec.[23]	TI	9292	**0**	40	250	160
6. S-box dec. [45]	$d+1$	**8701**	24	72	260	277

* Original implementation source code recompiled with the NANGATE45 library.

D. AES

Finally, we analyze the benefits of applying LLTI to AES. So far, we can only find two implementations of the AES S-box in the literature with a single layer of registers, both using different flavors of masking. The first one, from Gross *et al.* [9], employs the proposed Generic Low-Latency Masking (GLM), a masking scheme at the algorithmic level. The second one, from Sasdrich *et al.* [18], uses LMDPL, a gate level masking scheme. Hereunder, we present a new alternative using LLTI.

The AES S-box consists of an inversion in $GF(2^8)$, and an affine layer. The inversion in $GF(2^8)$, which has algebraic degree seven, can be decomposed into two simpler cubic functions: $x^{-1} = x^{254} = (x^{26})^{49}$ [16], [46]. We leverage this decomposition to split our implementation into two stages with one register layer in between the two functions. Instead of splitting further these cubic functions, we directly mask them with a 1^{st}-order non-complete $td+1$ sharing.

Thanks to the non-completeness property, the affine transformation can be applied to each share independently, and compose with the non-linear part without using register. To illustrate the great performance optimization that LLTI can provide, we design our AES implementation using both TI and LLTI for comparison. Tab. V presents the results of both our implementations, together with previous works AES secure implementations for reference.

Table V: 1^{st}-order SCA secure AES Implementation Cost.

Design	Area* $[kGE]$ S-box	 Total	Rand. $[bpc]$	# Regs. S-box	Cycles Total	Max. Freq.* $[MHz]$	Time* (ns)
4 shares LLTI	25.78	**36.49**	**0**	**1**	216	277	779
4 shares TI	58.41	69.12	0	1	216	40	5 400
Bilgin *et al.* [13]	2.84	8.12	32	2	246	-	-
De Cnudde *et al.* [14]	1.98	**6.68**	54	5	276	-	-
Gross *et al.* [9]	60.76	-	2 048	**1**	-	356	-
Gross *et al.* [9]	6.74	-	416	2	-	**584**	-
Wegener-Moradi [47]	4.20	7.60	**0**	16	2 804	-	-
Sugawara [48]	3.50	17.10	**0**	4	266	-	-
Sasdrich *et al.* [18]	3.48	157.50	720	**1**	**10**	400	25

* Different designs used different standard cell libraries.

As we can see from Tab. V, our implementations achieve minimal latency, with the S-box implemented with a single layer of registers. Moreover, no additional randomness besides the initial sharing is needed. We would like to stress that the purpose of these implementations is not to present a new or the best state-of-the-art AES implementation, but to illustrate the power of LLTI. As we can see from the first two rows, with LLTI we achieve 47.2% reduction in area and a 6.9 times increase in maximum frequency w.r.t. the TI implementation. The implementation from Gross *et al.* [9] using GLM is based

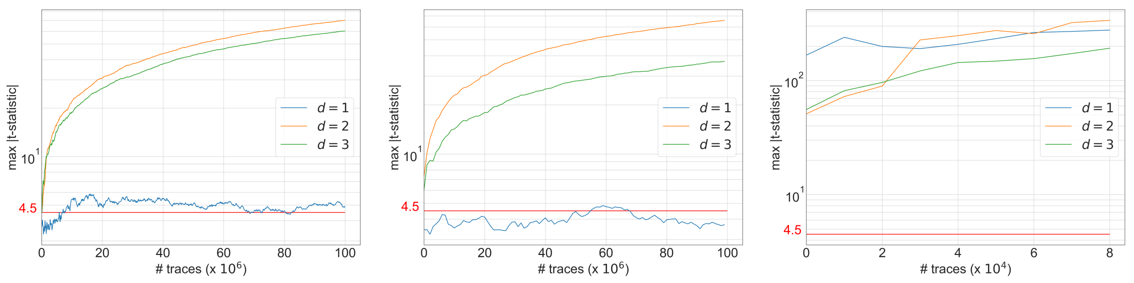


Figure 4: Max t-test value in logarithmic scale. From left to right: masks-on AES-TI, masks-on AES-LLTI, mask-off.

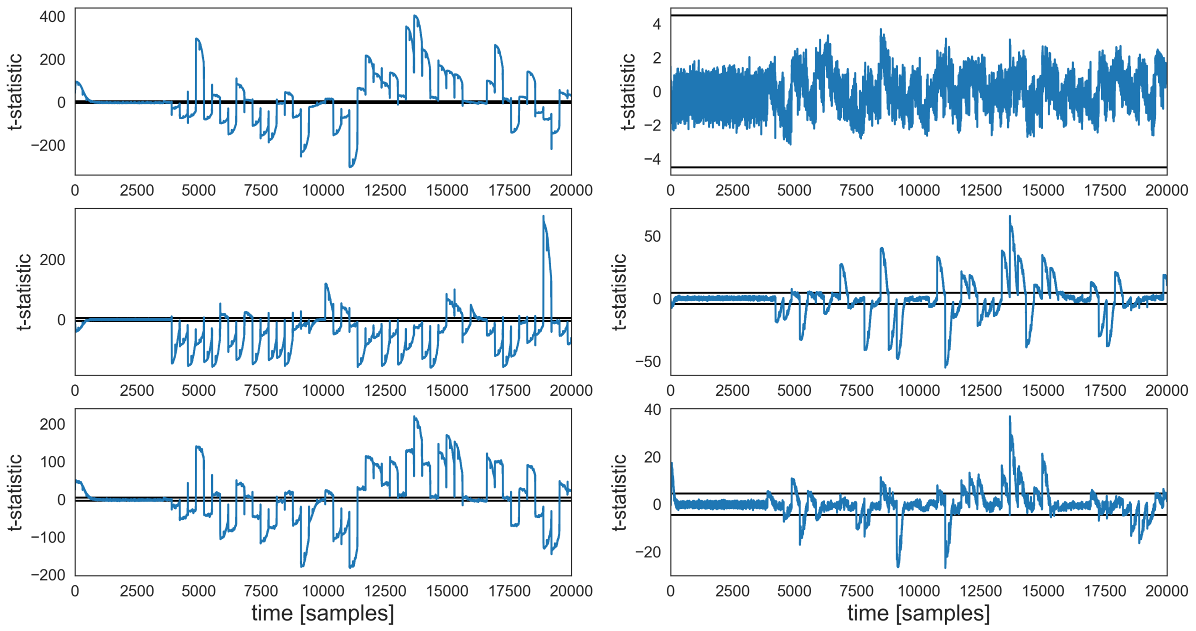


Figure 5: TVLA results for unmasked AES with 100k traces (left) and masked AES-LLTI with 100 million traces (right). Top to bottom: 1^{st}-, 2^{nd}- and 3^{rd}-order t-statistic

on $d+1$ masking, contrary to the $td+1$ masking of LLTI. Since they are both maskings at the algorithmic level, it is easier to compare between them. We can see that with LLTI we manage to significantly improve both area and online randomness.

The latest design from Sasdrich *et al.* [18] proposes an efficient round-based implementation computed in 10 cycles. However, it is difficult to have a one-to-one comparison since Sasdrich *et al.* employ a gate level masking scheme. Although our implementation is slower due to the serialized design, it performs much better in terms of online randomness, using zero bits compare to the 720 bits per cycle from Sasdrich *et al.*'s design. This allows us to avoid any PRNG placed next to our design, which greatly affects the overall area and power performance. Moreover, the use of a PRNG creates additional vulnerabilities, easily exploitable by the attackers. Finally, compared to the implementations using zero online randomness, the latency in cycles of our S-box is reduced 16 and 4 times respectively, at the cost of larger area. Every implementation has different trade-offs, and we should choose the most fitting one tailored to the desired application. Our implementation is presented as a new alternative to the state-of-the-art.

We use TVLA to evaluate the security of both our implementations, deeming them secure for up to 100 million traces. Figs. 4 and 5 present these results. In the case of the TI implementation, we can see small leakage, but since it does not grow with the number of traces (as opposed to $d = 2, 3$) and stays close to the threshold we can say that this leakage would not be exploitable by an attacker. On the contrary, we can see that with LLTI this leakage no longer appears. From Fig. 4, we can see that LLTI not only does not harm the security of the implementation compared to TI, but it even reduces the amount of leakage. This can be attributed to LLTI's reduction in the glitching activity. Glitches introduce unexpected (and unwanted) combinations, which threaten the security of masking in hardware [49]. As we have seen from Eq. (15) LLTI considerably reduces the XOR trees that are needed to compress all cross-terms to the number of output shares. This leads to a considerable reduction in glitching activity, which potentially translates into better security warranties.

VI. Conclusion

In this manuscript, we have presented Low-Latency Threshold Implementations (LLTI), a new masking methodology suitable for low-latency applications. It is derived and inherits the properties of TI, achieving a considerable improvement in the area, latency, and maximum frequency performances for any operation with an algebraic degree greater than two. For a first-order masked cubic AND gate, very often used in symmetric primitives, we achieved a 137% increase in maximum frequency and a 60% reduction in area compared to TI. Furthermore, we have presented different sharings for operations with $\mathrm{T} = 3, 5, 6$, and 7, achieving a remarkable 3131 times reduction in gate complexity for a first-order masking of a degree-seven monomial compared to TI. To show the practical impact of LLTI, we applied it to both examples of PRINCE and AES. We took a previously proposed PRINCE implementation and simply substituting the S-box with the new LLTI version, we achieved substantial improvements in both area and latency. Similar gains could be achieved when securing any cipher featuring a cubic AND operation. Finally, we proposed a new AES design with a one-cycle S-box and its TI equivalent, obtaining, with LLTI, an implementation with half the area and seven times greater max. frequency.

VII. Acknowledgements

This work was supported by CyberSecurity Research Flanders with reference number VR20192203.

References

[1] P. C. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *CRYPTO*. Springer, 1999.

[2] Y. Ishai, A. Sahai, and D. A. Wagner, "Private circuits: Securing hardware against probing attacks," in *CRYPTO*. Springer, 2003.

[3] S. Nikova, C. Rechberger, and V. Rijmen, "Threshold implementations against side-channel attacks and glitches," in *ICICS*. Springer, 2006.

[4] E. Prouff and T. Roche, "Higher-order glitches free implementation of the AES using secure multi-party computation protocols," in *CHES*. Springer, 2011.

[5] S. Nikova, V. Rijmen, and M. Schläffer, "Secure hardware implementation of nonlinear functions in the presence of glitches," *J. Cryptology*, 2011.

[6] B. Bilgin, B. Gierlichs, S. Nikova, V. Nikov, and V. Rijmen, "Higher-order threshold implementations," in *ASIACRYPT*. Springer, 2014.

[7] O. Reparaz, B. Bilgin, S. Nikova, B. Gierlichs, and I. Verbauwhede, "Consolidating masking schemes," in *CRYPTO*. Springer, 2015.

[8] H. Groß, S. Mangard, and T. Korak, "An efficient side-channel protected AES implementation with arbitrary protection order," in *CT-RSA*. Springer, 2017.

[9] H. Groß, R. Iusupov, and R. Bloem, "Generic low-latency masking in hardware," *Trans. CHES*, vol. 2018(2).

[10] 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," in *IACR Transactions on Symmetric Cryptology*, vol. 2020,1.

[11] D. Canright, "A very compact s-box for AES," in *CHES*, 2005.

[12] A. Moradi, A. Poschmann, S. Ling, C. Paar, and H. Wang, "Pushing the limits: A very compact and a threshold implementation of AES," in *EUROCRYPT*. Springer, 2011.

[13] B. Bilgin, B. Gierlichs, S. Nikova, V. Nikov, and V. Rijmen, "Trade-offs for threshold implementations illustrated on AES," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 34, no. 7, 2015.

[14] T. D. Cnudde, O. Reparaz, B. Bilgin, S. Nikova, V. Nikov, and V. Rijmen, "Masking AES with d+1 shares in hardware," in *CHES*. Springer, 2016.

[15] M. Rivain and E. Prouff, "Provably secure higher-order masking of AES," in *CHES*. Springer, 2010.

[16] F. Wegener and A. Moradi, "Yet another size record for AES: A first-order SCA secure AES s-box based on $\(\mathrm {GF}(2^8)\)$ multiplication," in *CARDIS*. Springer, 2018.

[17] L. De Meyer, V. Arribas, S. Nikova, V. Nikov, and V. Rijmen, "M&m: Masks and macs against physical attacks," *Trans. CHES*, vol. 2019(1).

[18] P. Sasdrich, B. Bilgin, M. Hutter, and M. E. Marson, "Low-latency hardware masking with application to AES," *Trans. CHES*, vol. 2020(2).

[19] E. Trichina, "Combinational logic design for AES subbyte transformation on masked data," *IACR Cryptol. ePrint Arch.*, 2003.

[20] T. Moos, A. Moradi, T. Schneider, and F. Standaert, "Glitch-resistant masking revisited or why proofs in the robust probing model are needed," *Trans. CHES*, vol. 2019(2).

[21] V. Arribas, B. Bilgin, G. Petrides, S. Nikova, and V. Rijmen, "Rhythmic keccak: SCA security and low latency in HW," *Trans. CHES*, vol. 2018(1).

[22] A. Poschmann, A. Moradi, K. Khoo, C. Lim, H. Wang, and S. Ling, "Side-channel resistant crypto for less than 2, 300 GE," *J. Cryptology*, vol. 24, no. 2, 2011.

[23] A. Moradi and T. Schneider, "Side-channel analysis protection and low-latency in action - case study of PRINCE and Midori," in *ASIACRYPT*, 2016.

[24] D. Bozilov, V. Nikov, and V. Rijmen, "Design trade-offs in threshold implementations," in *ICECS*. IEEE, 2019, pp. 751–754.

[25] D. Bozilov, M. Knezevic, and V. Nikov, "Optimized threshold implementations: Minimizing the latency of secure cryptographic accelerators," in *CARDIS*. Springer, 2019.

[26] L. De Meyer, B. Bilgin, and O. Reparaz, "Consolidating security notions in hardware masking," *Trans. CHES*, vol. 2019(3).

[27] S. Faust, V. Grosso, S. M. D. Pozo, C. Paglialonga, and F. Standaert, "Composable masking schemes in the presence of physical defaults & the robust probing model," *Trans. CHES*, vol. 2018, no. 3, 2018.

[28] A. Duc, S. Faust, and F. Standaert, "Making masking security proofs concrete - or how to evaluate the security of any leaking device," in *EUROCRYPT*. Springer, 2015.

[29] G. Petrides, "On non-completeness in threshold implementations," in *TIS@CCS*. ACM, 2019, pp. 24–28.

[30] B. Bilgin, "Threshold implementations : as countermeasure against higher-order differential power analysis," Ph.D. dissertation, 2015.

[31] B. Bilgin, B. Gierlichs, S. Nikova, V. Nikov, and V. Rijmen, "A more efficient AES threshold implementation," in *AFRICACRYPT*, 2014.

[32] M. Wei, S. Sun, Z. Wei, and L. Hu, "Unbalanced sharing: a threshold implementation of SM4," in *Science China Information Sciences*, vol. 64. Springer, 2021.

[33] O. Reparaz, "A note on the security of higher-order threshold implementations," *IACR Cryptol. ePrint Arch.*, vol. 2015.

[34] J. Schönheim, "On coverings." *Pacific J. Math.*, vol. 14, no. 4, 1996.

[35] D. M. Gordon, O. Patashnik, and G. Kuperberg, "New constructions for covering designs," *Journal of Combinatorial Designs*, no. 4, 1995(3).

[36] D. M. Gordon, O. Patashnik, G. Kuperberg, and J. Spencer, "Asymptotically optimal covering designs," *J. Comb. Theory(A)*, no. 2, 1996(75).

[37] C. Dai, P. C. Li, and M. Toulouse, "A cooperative multilevel tabu search algorithm for the covering design problem," in *Artificial Evolution*. Springer, 2005.

[38] F. Montecalvo, "Some constructions of general covering designs," *Electr. J. Comb.*, vol. 19, no. 3, p. P28, 2012.

[39] C. J. Colbourn and J. H. Dinitz, *Handbook of Combinatorial Designs, Second Edition (Discrete Mathematics and Its Applications)*. Chapman & Hall/CRC, 2006.

[40] J. Notebook, "LLTI sharings: https://colab.research.google.com/drive/1BoJSX7zgiBcbYsC-1Jv1u83RM7ExbEY2?usp=sharing."

[41] D. M. Gordon, "Covering designs, https://www.dmgordon.org/cover/."

[42] NANGATE, "The NanGate 45nm Open Cell Library," available at https://github.com/The-OpenROAD-Project/OpenROAD-flow-scripts/tree/master/flow/platforms/nangate45.

[43] G. Goodwill, B. Jun, J. Jaffe, and P. Rohatgi, "A testing methodology for side-channel resistance validation," September 2011.

[44] V. Arribas, S. Nikova, and V. Rijmen, "Vermi: Verification tool for masked implementations," in *ICECS*. IEEE, 2018.

[45] D. Bozilov, M. Knezevic, and V. Nikov, "Optimized threshold implementations: Securing cryptographic accelerators for low-energy and low-latency applications," *IACR Cryptol. ePrint Arch.*, vol. 2018, p. 922, 2018.

[46] F. Wegener, L. De Meyer, and A. Moradi, "Spin me right round rotational symmetry for FPGA-specific AES: Extended version," *Journal of Cryptology*, 2020.

[47] F. Wegener and A. Moradi, "A first-order SCA resistant AES without fresh randomness," in *COSADE*, 2018.

[48] T. Sugawara, "3-Share Threshold Implementation of AES S-box without Fresh Randomness," *Trans. CHES*, vol. 2019(1).

[49] S. Mangard, N. Pramstaller, and E. Oswald, "Successfully attacking masked AES hardware implementations," in *CHES*, ser. Lecture Notes in Computer Science, vol. 3659. Springer, 2005, pp. 157–171.