IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2020, No. 2, pp. 300–326. DOI:10.13154/tches.v2020.i2.300-326

Low-Latency Hardware Masking with Application to AES

Pascal Sasdrich, Begül Bilgin, Michael Hutter, and Mark E. Marson

Rambus Cryptography Research,
425 Market Street, 11th Floor, San Francisco,
CA 94105, United States
{psasdrich,bbilgin,mhutter,mmarson}@rambus.com

Abstract. During the past two decades there has been a great deal of research published on masked hardware implementations of AES and other cryptographic primitives. Unfortunately, many hardware masking techniques can lead to increased latency compared to unprotected circuits for algorithms such as AES, due to the high-degree of nonlinear functions in their designs. In this paper, we present a hardware masking technique which does not increase the latency for such algorithms. It is based on the LUT-based Masked Dual-Rail with Pre-charge Logic (LMDPL) technique presented at CHES 2014. First, we show 1-glitch extended strong non-interference of a nonlinear LMDPL gadget under the 1-glitch extended probing model. We then use this knowledge to design an AES implementation which computes a full AES-128 operation in 10 cycles and a full AES-256 operation in 14 cycles. We perform practical side-channel analysis of our implementation using the Test Vector Leakage Assessment (TVLA) methodology and analyze univariate as well as bivariate t-statistics to demonstrate its DPA resistance level.

Keywords: AES · Low-Latency Hardware · LMDPL · Masking · Secure Logic Styles · Differential Power Analysis · TVLA · Embedded Security

1 Introduction

Masking countermeasures against side-channel analysis [KJJ99] have received a lot of attention over the last two decades. They are popular because they are relatively easy and inexpensive to implement, and their strengths and limitations are well understood using theoretical models and security proofs.

In theory, any masking scheme can provide security with careful implementation using a good understanding of the underlying leakage of the platform. In practice, it can be difficult to precisely model their leakage and a lot of research focuses on finding schemes that are independent of the underlying platform and leakage.

For instance, there are many masking schemes, with their implementations providing practical security when implemented on hardware using standard cells, with no limitation on the exact placing and routing of the circuit [LMW14, PR11, MPL+11, BGN+14a, GMK16, GM18, DRB18]. Some of these masking schemes use polynomial or multiplicative masking, whereas others are based on Boolean masking. The underlying first-order security of many of these implementations can be investigated using the correctness, non-completeness, and uniformity properties introduced for threshold implementations (TI) in [NRR06]. These properties have been extended to provide higher-order security in [BGN+14a] assuming a limited adversary only capable of using univariate leakages [RBN+15]. However, finding hardware implementations of arbitrary orders that are secure in a multivariate setting is still a challenging task today [MMSS19]. Analysing the strong non-interference (SNI)

Received: 2019-10-15 Accepted: 2019-12-15 Published: 2020-03-02

of a gadget under (glitch-extended) probing model is helpful to construct circuits with multivariate security [BBD+16, DRB18, FGP+18].

Despite these challenges, there has been good progress minimizing costs in terms of area requirements for masked implementations. Approaches include optimizing the number of shares required to achieve d^{th}-order security [GMK16, RBN+15, GM18], and serializing the operations to the point of extreme [DMW18]. It is still a challenge to minimize the randomness requirements when we move to theoretical security against higher-order attacks [MMSS19]. However, we can optimize the use of randomness to achieve first-order security [DRB18, Dae17, Sug19].

Unfortunately, until today, low-latency has received less attention as optimization metric. All the state-of-the-art hardware masking techniques increase latency compared to unprotected circuits when applied to high-degree nonlinear functions. However, low-latency is becoming increasingly important for fast and secure payments, the automotive market, and high-performance applications such as in-line memory encryption. Therefore, in this paper we focus on low-latency hardware implementations instead of looking for area and randomness optimizations. Our goal is to develop a low-latency hardware masking scheme which provides both theoretical and practical first-order security against side-channel attacks.

1.1 Previous Work

We discuss below the handful of investigations on low-latency masked implementations which consider multiple rounds per cycle or designs with complex nonlinear layers. The number of investigations gets even lower when we consider generic approaches applicable to any cryptographic algorithm. One such investigation is presented in [ABP+18], where authors show the possibility of composing nonlinear functions of any size in order to achieve low-latency. They increased the number of shares accordingly, to amortize the increased information gathered per probed wire due to glitches. The authors provide a low-latency first-order secure hardware implementation of Keccak where two rounds are implemented in one cycle. Note that the Keccak round function has a quadratic degree which makes it a good candidate for this method. Unfortunately, the technique is impractical for algorithms such as AES, which have high-degree round functions.

In [GIB18], the authors present a generic low-latency solution independent of the security order or the degree of the underlying nonlinear operation. They use Domain-Oriented Masking (DOM), which is known to be secure when the inputs of each nonlinear gadget are shared independently, without the compression layer. They provide a tool that tracks the dependency of the inputs of each gadget, and suggest creating multiple refreshed copies of input variables such that the whole circuit can be implemented without independency failure for any gadget inputs. The method works well for algorithms with low non-linearity such as Ascon, which has a quadratic S-box akin to that of Keccak. Unfortunately, implementing a single shared AES S-box using this technique requires 60kGE using 90nm Low-K UMC process. This is far too costly for most practical applications.

Even when targeting a specific algorithm, achieving low latency appears to be a non-trivial task. For example, in [GC17], the authors present a masked AES implementation requiring three clock cycles per round. This is faster than most other Boolean masked AES implementations which rely on non-completeness for security. However, it was shown in [SBY+18, WM18] that this implementation is insecure. Similarly, in [GSM17] a single-cycle-per-round Keccak implementation is presented, which is then shown to be insecure in [ABP+18]. In fact, even for algorithms that use a small 4-bit S-box and are designed specifically to address low-latency demands such as the PRINCE block cipher [BCG+12, MS16, BKN18], providing a low-latency implementation that is secure against SCA has been shown to be a challenging task.

Until our work presented here, the lowest latency for a practical AES implementation

was presented by Leiserson et al. in [LMW14], and is based on LMDPL. It requires two clock cycles per round and provides security against first-order side-channel analysis. Unfortunately, their security argument relies on a new analysis technique called activity images, which has not been adopted by the community as a standard analysis tool.

1.2 Our Contributions

In this paper, we investigate the theoretical security of the LMDPL gadget and show it is first-order secure under the 1-glitch extended probing model (Sec. 3). This is the first formal analysis of LMDPL gadgets showing their security and composibility in literature. Using this information on how to securely compose LMDPL gadgets, we designed the first practical hardware-masked single-cycle-per-round AES implementation (Sec. 4). We then empirically verified that the claimed security holds in practice using Test Vector Leakage Assessment (TVLA) [GJJR11] (Sec. 5).

2 Preliminaries

In this section, we review state-of-the-art adversary models and requirements used for hardware implementations to provide security against side-channel analysis. We also provide a brief review of the LMDPL gadgets upon which our proposal is based.

We use lower case italic letters to denote single-bit variables, such as $a \in \mathbb{GF}(2)$. We represent multi-bit vectors with lower case bold letters, where each element within a vector is indicated by a superscript. For example, we write $\mathbf{a} = \langle a^1, \ldots, a^m \rangle$, where $\mathbf{a} \in \mathbb{GF}(2^m)$ and $m \geq 1$. We denote vectorial Boolean functions and random variables using upper case bold letters which will be clear from context. We identify sets using calligraphic font. We use the symbols &, $\vee$, $\oplus$, and $\cdot$ to denote bit-wise AND, OR and XOR, and field multiplication respectively. We omit & and $\cdot$ when it is clear from the context or the discussion is independent of the underlying field. We use an overline to denote the complement of a bit, $\overline{a} = a \oplus 1$, and the bit-wise complement of a vector $\overline{\mathbf{a}} = \langle a^1 \oplus 1, \ldots a^m \oplus 1 \rangle$. Finally, we denote the mutual information between $\mathbf{a}$ and $\mathbf{b}$ as $I(\mathbf{a}; \mathbf{b})$.

2.1 Adversary Models and Security Requirements

The exact leakage of an implementation can be affected by various parameters including its inputs, device specifics, exact place and routing, voltage, and temperature. Our goal, similar to many others in literature, is to provide an implementation for which the security is independent of these parameters. In what follows, we describe the abstract models and requirements we use to achieve this.

2.1.1 Masking and d-Probing Security

The d^{th}-order Boolean masking (sharing) of a variable $\mathbf{a} \in \mathbb{GF}(2^m)$ is represented by $s_{\mathbf{a}} = \{\mathbf{a}_i\}_{i=1}^n$ where each share $\mathbf{a}_i \in \mathbb{GF}(2^m)$ and

$$\mathbf{a} = \bigoplus_{i=1}^{n} \mathbf{a}_i = \bigoplus_{i=1}^{n} \langle a_i^1, \ldots, a_i^m \rangle.$$

The number of shares n depends on the particular masking scheme, and is always greater than the security order d. Any combination of at most d shares should not give information about a.

Any function $\mathbf{F}(\mathbf{a}) = \mathbf{x}$ can be implemented using a set of affine operations and multiplications in the corresponding field. Masked calculation of an affine function $\mathbf{A}$ is trivial, as the function can simply operate on each share individually: $\mathbf{A}(\mathbf{a}_i) = \mathbf{x}_i$.

On the other hand, masked multiplication is more challenging. Below we give an example sharing for the function $\mathbf{F}(a, b) = ab$ which use three shares and a random variable r_i, taken from [ISW03].

$$\begin{aligned} t_1 &= (a_1 b_2 \oplus r_1) \oplus a_2 b_1 & x_1 &= a_1 b_1 \oplus r_1 \oplus r_2 \\ t_2 &= (a_1 b_3 \oplus r_2) \oplus a_3 b_1 & x_2 &= a_2 b_2 \oplus t_1 \oplus r_3 \\ t_3 &= (a_2 b_3 \oplus r_3) \oplus a_3 b_2 & x_3 &= a_3 b_3 \oplus t_2 \oplus t_3 \end{aligned} \tag{1}$$

For the rest of the paper, we will be working with circuits operating on masked variables, and will focus mostly on AND gates unless stated otherwise.

Definition 1 (d-probing security [ISW03])**.** *A circuit is d-probing secure if and only if every d-tuple of its intermediate variables is independent of any sensitive variable.*

Note that the set of the equations shown in Eqn. (1) is 1-probing secure as each intermediate variable (in addition to input and output variables) is independent of the sensitive unmasked variables.

It was shown in [DDF14] that for a circuit without any glitches, d-probing security implies security in the noisy-leakage model. This model assumes that each share leaks independently, where there is no cross talk between the shares, and the sum of noisy leakages of each share is provided to the adversary. This model has been shown to match real-world physical leakages [CJRR99, PR13]. However, it has also been shown that there are two major drawbacks.

1. The d-probing security of a gadget such as a masked AND-gate does not imply security of a circuit where these gadgets are composed arbitrarily [CPRR13].
2. The assumption that a circuit does not have any glitches has been shown to be unrealistic, especially on hardware circuits [MPG05].

In what follows, we discuss known solutions to address these drawbacks.

2.1.2 Composability and d-Strong Non-Interference

In [RP10] the authors present an AES implementation using $d + 1$ shares, with the claim of d^{th}-order security. The authors use d-probing secure multiplication gadgets, together with d-probing secure refreshing gadgets to ensure that the shared inputs of any multiplication are independent of each other. However, it is shown in [CPRR13] that this AES implementation and underlying S-box implementation do not provide the claimed security for higher orders. The reason is that these gadgets, even though they are d-probing secure, need special attention for composition.

Definition 2. (Composable gadget) *A d-probing secure gadget is composable if arbitrarily combining d such gadgets results in a d-probing secure circuit.*

Later, it was shown in [BBD+16] that a gadget satisfying d-SNI property as described below is composable. Moreover, any circuit composed of affine gadgets (where share boundaries are not violated, i.e. $\mathbf{A}(\mathbf{a}_i) = \mathbf{x}_i$) and d-SNI gadgets is d-probing secure.

Definition 3 (d-Strong Non-Interference [BBD+16])**.** *A gadget is d-Strong Non-Interfering (d-SNI) if and only if for any set of $p1$ probes on its intermediate values and every set of $p2$ probes on its output shares with $p1 + p2 \leq d$, the totality of the probes can be simulated with $p1$ shares of each input.*

Here *simulation* implies a function which takes $p1$ shares for each input and calculates a joint distribution that is exactly equal to the distribution produced on its d probes by the gadget or algorithm under study [BBD+18].

2.1.3 Security of Circuits with Glitches

Even if a circuit is secure under d-probing model, it might not be secure in practice [MPG05]. This is because the model does not take into account physical effects such as glitches.

The *d-glitch-extended probing* model was created to address the shortcomings of the d-probing model with respect to physical defaults such as glitches [DRB18, FGP+18, RBN+15]. In this model, each glitch-extended probe not only gives information about the probed wire, but also all the variables used to calculate the value of that wire up to the last synchronization point. Note that this is a very strong model covering worst-case leakages which might not occur in practice. However, it does provide a theoretical model one can use with a high degree of confidence. A gadget is shown to be composable under d glitch-extended probing model if it satisfies d glitch-extended SNI (d-GSNI) property as described below.

Definition 4 (d-GSNI [DBR19])**.** *Consider a gadget with $d+1$ input shares $\mathbf{a}_i$, where $i \in \{1, \ldots, d+1\}$. Let $\mathcal{O}$ be any observation set of at most d glitch-extended probes in $\mathbb{GF}(2^m)$. Let $p1$ and $p2$ be the number of intermediate and output probes respectively such that $p1 + p2 \leq d$. The gadget is d-GSNI if for any such $\mathcal{O}$ the following condition holds:*

$$\exists \mathcal{P} \subset \{1, \ldots, d+1\} \text{ and its complement } \hat{\mathcal{P}} \text{ with } |\mathcal{P}| = p1 \text{ such that } I(\mathcal{O}; a_{\hat{\mathcal{P}}} | a_{\mathcal{P}}) = 0.$$

Note that this definition assumes an information-theoretic point of view and is equivalent to Definition 3 if regular probes are used instead of glitch-extended probes. This generality is used in Section 3 to show that the LMDPL gadget is 1-GSNI, and our AES implementation built upon it is secure.

2.2 LMDPL

A new gate-level masking technique, designed to provide first-order security, called LUT-based Masked Dual-rail with Pre-Charge Logic (LMDPL) was presented at CHES 2014 [LMW14]. Here, we describe each linear and non-linear gadget of LMDPL in detail and put them into perspective.

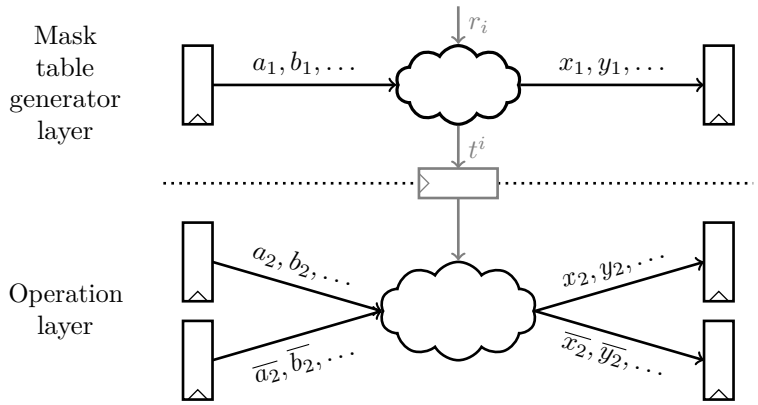


Figure 1: LMDPL gadget at a high level, components present only for nonlinear gadgets are depicted in gray

Each LMDPL gadget uses two-share Boolean masked variables ($s_a = (a_1, a_2)$) together with the complement of the second share ($\overline{a_2}$). The gadgets can be viewed as split into two layers, where the first layer uses only the first share of the input variables and outputs

a single share. It also creates a vector called the *table output* if the gadget is non-linear. The second layer derives the second output share and its complement using the second share of the input variables, their complements, and the table output generated by the first layer when it exists. These layers are referred to as the mask-table generator and the operation layer respectively.

2.2.1 Nonlinear Gadgets

An LMDPL gadget for any nonlinear function $F(a, b) = x$, where $a, b, x \in \mathbb{GF}(2)$, is defined as follows.

The mask-table generation layer gets a random number $r \in \mathbb{GF}(2)$ and assigns it as x_1 (i.e., $x_1 = r$). It also uses this random number together with the first shares of the inputs to create a *table output* vector $\mathbf{t} \in \mathbb{GF}(2^3)$ as follows:

$$\begin{aligned} t^{4+2i+j} &= F(a_1 \oplus j, b_1 \oplus i) \oplus r \\ t^{2i+j} &= F(a_1 \oplus j, b_1 \oplus i) \oplus r \oplus 1, \text{ where } i, j \in \mathbb{GF}(2) \end{aligned} \tag{2}$$

The operation layer uses the synchronized variable $\mathbf{t}$, the second shares of the input and their complements as shown in Figure 2.

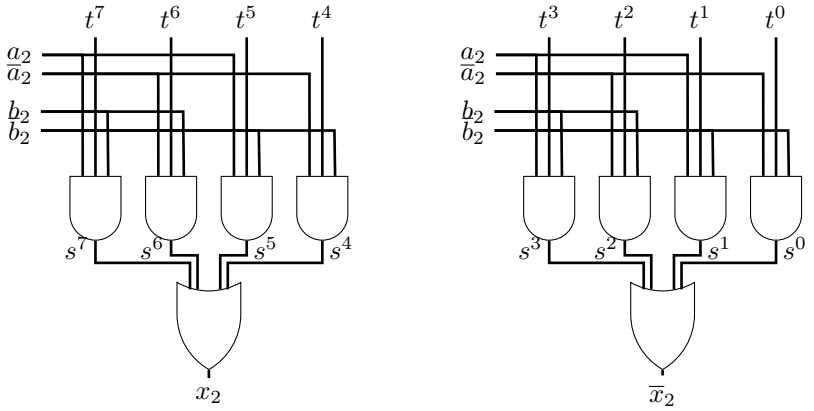


Figure 2: Combinational logic of a nonlinear LMDPL Gadget's operation layer

2.2.2 LMDPL Gadgets for Linear Gates

Similar to other masking schemes, the gadgets for linear operations are very simple and each layer works on only a single share.

NOT. The sharing of $x = NOT(a)$ is calculated as follows:

$$\begin{aligned} x_1 &= a_1, \\ &\cdots\cdots \\ x_2 &= \overline{a_2} \\ \overline{x_2} &= a_2. \end{aligned}$$

XOR. Unlike other masking schemes which build masked XOR gadgets of only standard XOR cells, the LMDPL XOR gadget of $x = a \oplus b$ is implemented using both AND and

OR gates as follows

$$\begin{aligned} x_1 &= a_1 \oplus b_1, \\ &\cdots\cdots\cdots\cdots \\ x_2 &= \overline{a_2} b_2 \vee a_2 \overline{b_2} \\ \overline{x_2} &= \overline{a_2}\,\overline{b_2} \vee a_2 b_2. \end{aligned}$$

2.2.3 Implementation Restrictions and Security Claim

It is important to note two implementation restrictions of LMDPL gadgets. First, all the gadgets use only monotonic gates in the operation layer. Second, each gadget operates in two phases. The first phase is a pre-charging phase, during which each input line to the gadget is set to 0. The second phase is a calculation phase, during which the inputs to the gadgets change from 0 to 1, depending on the data.

Implementing a gadget to have these properties might seem to require careful, potentially custom, circuit design.

However, in [LMW14], Appendix B, the authors point out that ensuring these properties only requires to check that the basic LMDPL cell primitives are preserved after synthesis and P&R. This can be done either by using some compiler-specific attributes such as *don't_touch* or *keep_hierarchy* placed in the high-level design language source code or by including custom synthesis constraint files in the ASIC or FPGA-design flow.

Also, in [LMW14] the authors introduced a new method, called activity image analysis, to analyze the security of a gadget. They used this technique to analyze the toggles of each wire for all possible input sharings of an LMDPL AND gate, and show that the toggle count is constant for all possible input sharings and randomness. The authors also showed the insecurity of an iMDPL AND gate [PKZM07], which is another dual-rail pre-charged logic based gadget using monotonic gates, using activity image analysis.

Finally, the authors used their LMDPL gadget to design an AES implementation which computes each round in two cycles. The first cycle is used for the pre-charging phase, and the second cycle is used for the calculation phase.

3 Security Analysis of the LMDPL Gadget

In this section, we focus formally on the first-order security of LMDPL gadgets using the security notions described in Section 2, and show that the gadgets can be composed securely under the 1-glitch extended probing model. We also describe how these gadgets can be used for low-latency implementations in full generality.

3.1 Nonlinear LMDPL Gadget

In this section, we prove security of the nonlinear LMDPL gadget under the glitch-extended probing model. Our proof benefits from Definition 4, as it can cover both information gathered using probing model and glitch-extended probing model for individual wires.

Observation from each glitch-extended probes. As described in Definitions 3 and 4, we distinguish between probing the output shares which are used as input shares in the following gadgets and the intermediates. Typically, the output probes are considered to be stable values. On the other hand, the intermediates are considered to be unstable, hence requires the usage of glitch-extended probes [FGP+18]. In this paragraph, we focus on the observation from these intermediates. As described in Section 2.1.3, with a glitch-extended probe, an adversary is able to observe not just the probed value, but also all the variables used to calculate that value, up to the last synchronization point. Therefore, probing the

Table 1: Observation from glitch-extended probes

Probes		Observation	Probes		Observation
Intermediate	t^i	$\{a_1, b_1, r\}$	Output	x_1	$\{r\}$
	s^0	$\{t^0\overline{a_2}\overline{b_2}\}$		x_2	$\{ab \oplus r\}$
	s^1	$\{t^1 a_2\overline{b_2}\}$		$\overline{x_2}$	$\{ab \oplus r \oplus 1\}$
	s^2	$\{t^2\overline{a_2} b_2\}$			
	s^3	$\{t^3 a_2 b_2\}$			
	s^4	$\{t^4\overline{a_2}\overline{b_2}\}$			
	s^5	$\{t^5 a_2\overline{b_2}\}$			
	s^6	$\{t^6\overline{a_2} b_2\}$			
	s^7	$\{t^7 a_2 b_2\}$			
	x_2	$\{s^4, s^5, s^6, s^7\}$			
	$\overline{x_2}$	$\{s^0, s^1, s^2, s^3\}$			

end points of the combinational logic just before synchronisation is assumed to give the adversary the most information.

Even though the observation from a glitch-extended probe in the mask-table generator layer is trivial (see the intermediates x_1 and t^i in Table 1), the ones from an operation layer (i.e., x_2 and $\overline{x_2}$) need further investigation. In a traditional setting, where the combinational logic is taken as a black box, a probe on the wire x_2 would give the following observation set:

$$\mathcal{O} = \{a_2, \overline{a_2}, b_2, \overline{b_2}, t^4, t^5, t^6, t^7\}.$$

Clearly $I(\mathcal{O}; a \oplus b)$ is not zero, since $a_2 \oplus b_2 \oplus t^4 \oplus t^7 = a \oplus b \oplus 1$.

As discussed in Section 2.2.3, the operation layer of the LMDPL gadget is designed to operate in a glitch-free manner. Due to the monotonic characteristic of the gates used, each wire can toggle at most once per cycle if the inputs to the operation layer were pre-charged. Note that this statement is true for both the cycle in which we move from a pre-charging phase to calculation phase, and vice versa. However, as has been shown for other gadget construction using dual-rail pre-charge logic with monotonic gates; even though there is no glitching, the gates might still leak information due to difference in timing of this transition [KKT06, SS06]. For the rest of the security analysis we assume that the structure of the operation layer is kept intact using *dont_touch* constraints, with no optimizations rearranging the gates. We also assume that the logic is pre-charged and show the role of additional mask-table generation idea to achieve security.

We now consider how signal delays might affect the analysis. Suppose the delays of the inputs to the OR differ enough for an adversary to measure. Walking backwards on the glitch-extended probe x_2, an adversary who knows the delay of each input wire to the OR gate could potentially learn the values of $\{s^4, s^5, s^6, s^7\}$ based on the timing information of a toggle in x_2. However, walking further backwards to the AND gate, we see that we can not observe each individual input of the AND gate even if we know the output of that gate. Let's take the AND gate resulting in s^4 as an example. Even if we know the delays of t^4, $\overline{a_2}$ and $\overline{b_2}$, due to the nature of an AND gate, we learn that either all of them are one at a certain time or there exists at least one input that is zero. This is the same information gained from observing $s^4 = t^4\overline{a_2}\overline{b_2}$.

We have covered all the intermediates and outputs which can be observed with a single probe, and included them in Table 1. Note also that similar arguments to the ones given above for the evaluation phase of the gadget apply to the pre-charging phase as well.

The gadget is 1-GSNI By Definition 4, we have two types of wires to probe. For $(p1, p2) = (1, 0)$, $\mathcal{O}$ contains one of the intermediate wires listed above. It can be verified

as a simple exercise that for each possible observation set $\mathcal{O}$, $I(.;.) = 0$ is satisfied. This is because each observation set is independent of at least one input share of each variable.

First, suppose an adversary probes one of the t^i, which is computed using a_1, b_1 and r. All these values are independent of (a_2, b_2), so we immediately conclude that an adversary gains no information about the second share.

Next, suppose an adversary probes one of the intermediate values s^i, say $s^0 = t^0\overline{a_2}\overline{b_2}$. We want to to verify that

$$I(\ \mathcal{O} = s^0;\ (a_1, b_1) \mid (a_2, b_2)\ ) = 0.$$

This can be done by creating a 32-row function table for s^0 based on all possible values for (a_1, b_1, a_2, b_2, r). One can then verify that for each value of (a_2, b_2, s^0) appearing in the table, the corresponding values for (a_1, b_1) are uniformly distributed between $\{(0,0),(0,1),(1,0),(1,1)\}$ (see Appendix B). Hence no information about the input share (a_1, b_1) is obtained from knowledge of (a_2, b_2, s^0).

Alternatively, using the simplification rules described in [BBD+16], one can see that $s^i \sim ra_2b_2$, which can be simulated using a single share of each variable.

Finally, suppose an adversary probes the output of one of the OR gates, say $\overline{x_2}$, and acquires the observation set $\mathcal{O} = \{s^0, s^1, s^2, s^3, \overline{x_2}\}$. We can still prove that

$$I(\ \mathcal{O};\ (a_1, b_1) \mid (a_2, b_2)\ ) = 0.$$

As before, create a 32-row function table for all the values in observation set $\mathcal{O}$, based on all possible values for (a_1, b_1, a_2, b_2, r). One can then verify that for each value of $(a_2, b_2, s^0, s^1, s^2, s^3, \overline{x_2})$ appearing in the table, the corresponding values for (a_1, b_1) are uniformly distributed between $\{(0,0),(0,1),(1,0),(1,1)\}$ (see Appendix B).

For $(p1, p2) = (0, 1)$, $\mathcal{O}$ contains one output wire. Each output of the nonlinear LMDPL gadget is randomized by r, making it independent of both input shares. We therefore conclude that the nonlinear LMDPL gadget is 1-GSNI. Additionally, it can easily be verified that the above argument of being 1-GSNI is also correct when the shared inputs depend on each other up to layer separation.

3.2 Composing LMDPL gadgets

Since the nonlinear gadget is 1-GSNI, and the linear gadgets conserve the share boundaries, they can be composed arbitrarily to achieve first-order security.

So far, we discussed composition where the input and output registers of each nonlinear gadget are preserved. Below we argue that multiple nonlinear gadgets can be combined securely without any registers in between.

For masked hardware implementations such as threshold implementations [NRR06] or domain-oriented masking [GMK16], synchronisation between nonlinear layers is imposed in order to avoid glitches cascading from one gadget to another. Synchronization also keeps the glitches in their corresponding share boundaries. This synchronization is typically achieved through the use of registers for the inputs to the gadgets.

On the other hand, if we allow the use of *only* LMDPL gadgets in our circuit where each gadget uses a different random value, we can can eliminate the need for some of the registers. In particular, the mask-table values $\mathbf{t}$ must be synchronized through registers, as they are coming from a circuit with glitches and they cross share boundaries. However, x_2 and $\overline{x_2}$ do not need to be registered in between every nonlinear gadget. As long as the circuit was pre-charged, these values only carry information from a single share, even when the gadget is used with dependent input. This implies that multiple gadgets can be combined if they are pre-charged appropriately. If t^is for each gadget are pre-charged, using a pre-charge register in the first layer of the gadgets' inputs suffices to pre-charge the following gadgets operating in the same cycle.

Finally, since the mask-table generation layer uses only a single share and a random variable, similar to a linear layer, we do not need synchronization of x_1.

With this information, we can safely iterate multiple nonlinear and linear gadgets to calculate functions of high degree.

3.3 Low-Latency Implementations using LMDPL Gadgets

In the original work [LMW14], the authors propose a novel AES hardware implementation with two register stages resulting in a two-step calculation. The first step performs the S-box inversions in $\mathbb{GF}(2^8)$ using LMDPL gadgets. These are the only nonlinear operations in AES. The second step performs all the other round operations of AES, which are linear. This results in a latency of two clock cycles per round function computation. In this architecture, the registers before the inversion are pre-charged every other cycle, while the shared linear calculations are active in between. This maintains the glitch-free architecture for the non-linear operation. In this section, we are going to show how the original concept can be improved to enable the construction of low-latency hardware masked implementations for any cryptographic algorithm.

We start with an observation made at the end of Section 3.2. As long as $\mathbf{t}$ is synchronized and the operation layer is only composed of LMDPL gates that are pre-charged, there is no need for a register between each gadget. Hence, we can keep on accumulating as many gadgets as we would like within one clock cycle. This implies that we can calculate up to n round transformations within one clock cycle, where n depends upon the complexity of the round transformations. Since we need to pre-charge this first register stage, however, we need a second stage that simply holds data during pre-charge before it is carried again to the first register stage. That is, an m-round algorithm can be implemented using $2 \times (m/n) - 1$ cycles where the output is taken from the output of first register stage. This allows for the possibility of unrolled implementations in which the amount of unrolling is restricted not by security concerns, but by concerns such as meeting timing, die area, and the availability of randomness.

With the first observation, we can design an implementation which computes n round transformations within two clock cycles: one pre-charge phase and a calculation phase, each of which is active in only one cycle. In order to improve this further, we propose to use the concept of duality, which is easily achievable for hardware designs. In particular, using duality on the operation layer results in duplication of the entire evaluation circuit. Given this, both partitions operate in an alternating way. One partition is always pre-charging, while the second partition evaluates on the data after being pre-charged in the previous cycle. This gives an implementation of m/n cycles for an m-round algorithm where each phase has an n-round transformation evaluation circuit.

Finally, we use the fact that the mask-table generation layer does not need pre-charging. Hence duplication of the mask-table generation and operating in an alternating manner is not required. Instead, the same logic can serve both partitions of the operation layer. This final enhancement does not improve the latency but helps in area optimization.

In summary, we have shown that the LMDPL gadget is d-GSNI, which makes it composable. We also have shown that it is suitable for generic constructions, and can be used for any cryptographic algorithm to enable a provable secure low-latency hardware masking.

4 Case Study: Low-Latency SCA-Protected AES

In this section, we highlight the most important aspects of a practical implementation of a masked, round-based AES architecture using LMDPL. Before investigating the security in

terms of side-channel protection, we give a detailed description of our architecture and provide area and performance results derived for a $28nm$ ASIC technology node.

4.1 Design Considerations

AES is one of the most predominantly deployed symmetric-key block ciphers, and is used in many security critical applications. It is a Substitution-Permutation Network (SPN), with a 128-bit block size. It supports 128, 192, and 256-bit keys and, depending on the key size, runs for 10, 12 or 14 rounds respectively. For the remainder of this work, we will focus on the 128-bit and 256-bit keys, as they are used most widely. Note, however, that the selection of different key sizes does not affect the round computation architecture, and only requires changes to the round key computation in the design.

One of our main goals is to design a combined architecture which can perform SCA-protected AES-128 and AES-256 operations with a maximum latency of 10 and 14 cycles respectively. Each round instance supports both the encryption and decryption operations. The atomic sub-functions are masked and implemented in parallel, allowing the design to perform an entire round computation and update of the internal state registers within a single cycle.

Since some applications and modes of operation only require either the encryption or decryption direction, we also implement encryption-only and decryption-only variants of our architecture by removing unnecessary parts of the design. Moreover, we also discuss variants where key protection is disabled. This results in a smaller implementation, which accepts and processes keys as a single share, rather than accepting and processing multiple keys shares.

Our architecture is designed to provide protection against first-order, univariate side-channel attacks. We empirically verified our design using a practical SCA-evaluation setup and state-of-the-art leakage assessment methodologies. To verify that our negative results for leakage were not due to errors in our evaluation setup, our implementation includes a switch which can turn off the masking by setting the randomness generator to output all zeros. By observing leakage in this mode of operation we verify that our evaluation setup is collecting and processing data correctly.

4.2 Architecture Details

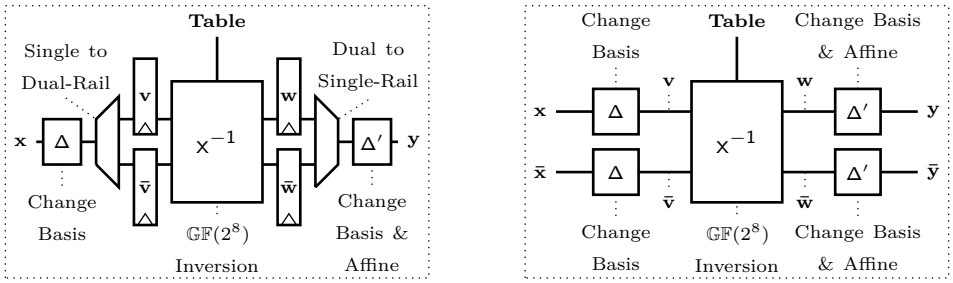
This section follows a bottom-up approach to provide the low-level architectural details of our AES implementation. First, we outline the details of our masked S-box architecture. This is the most critical component of the design, as well as the most difficult to protect against side-channel attacks. Our architecture uses LMDPL gadgets to protect the nonlinear operations of the S-box calculation.

We then describe the architecture of the full round function, including the remaining atomic sub-functions. After that we present details on the optional key expansion protection. Finally, we give an overview of the entire architecture for our AES accelerator.

4.2.1 Low-latency S-box Architecture

In a first step, we started to optimize and improve the architecture of the S-box as the most critical component in terms of SCA security in order to reduce the latency. Figure 3 shows a comparison between the original LMDPL-based S-box construction presented in [LMW14] and our proposal for a low-latency S-box construction. Both implementations are based on the S-box described in [Can05].

As shown in Figure 3a, each S-box instance comprises two register stages framing the core module, i.e., the inversion in $\mathbb{GF}(2^8)$ using dedicated LMDPL cells. To this end, a single computation of the S-box substitution takes two cycles and follows the



(a) Original **(b)** Low-latency

Figure 3: Comparison of the original and our low-latency LMDPL S-box.

principle of separate clock cycles for pre-charge and evaluation phase. In addition, the field inversion requires pre-computed tables depending on the mask share to generate the (masked) inversion result. Due to the construction of the LMDPL cells and the principle of pre-charge and evaluation phases, the field inversion is entirely built and implemented in dual-rail logic and requires a second inverted input while providing the correct and inverted results as output.

However, reducing the latency of the original S-box construction is not easily achievable by removing the internal register stages. Since the field inversion circuit is constructed of the LMDPL gates and implemented in dual-rail logic style, pre-charging the circuit before evaluating on input data is still required and crucial to preserve the security properties of the masking scheme. For this reason, we decided to implement the entire S-box in dual-rail logic style, as outlined in Figure 3b, and move the duty of pre-charging the field inversion unit to the external module instantiating our low-latency S-box construction.

4.2.2 Low-Latency Round Computation

In addition to the low-latency S-box architecture entirely implemented using a dual-rail logic style, we decided to extend the dual-rail logic to the entire data path responsible for a full round computation of the AES cipher. This extension includes sub-modules that perform the shifting of rows, the addition of round keys, and the mixing of columns entirely on dual-rail encoded data for both encryption and decryption directions[1].

However, the combination and integration of the encryption and decryption path in the same circuitry required the design and application of special monotonic dual-rail logic gates, e.g., a dual-rail multiplexer and a dual-rail XOR with one gated input, in order to preserve the pre-charging properties throughout the entire data path (Appendix A).

To this end, a preceding register stage holds the dual-rail encoded inputs of the round computation and allows to pre-charge the entire round function circuitry if cleared during the pre-charge phase.

4.2.3 Protected Low-Latency AES Architecture

Duplication of the register and round function allows the first instance to pre-charge while the second instance, which was pre-charged in the previous cycle, is available for the evaluation phase. Using an alternating flow of pre-charging and evaluation for the two modules, our architecture ensures the processing and update of the internal AES state in every cycle. This results in a encryption-decryption combined, masked, single-cycle per round AES architecture.

[1]Potentially, subsequent logic of the S-box could be reduced to single-rail for area and performance optimization, but at risk of diminishing the security in particular for higher-order side-channel analysis. Note, however, that we decided to not implement nor analyze this approach but leave it as future work.

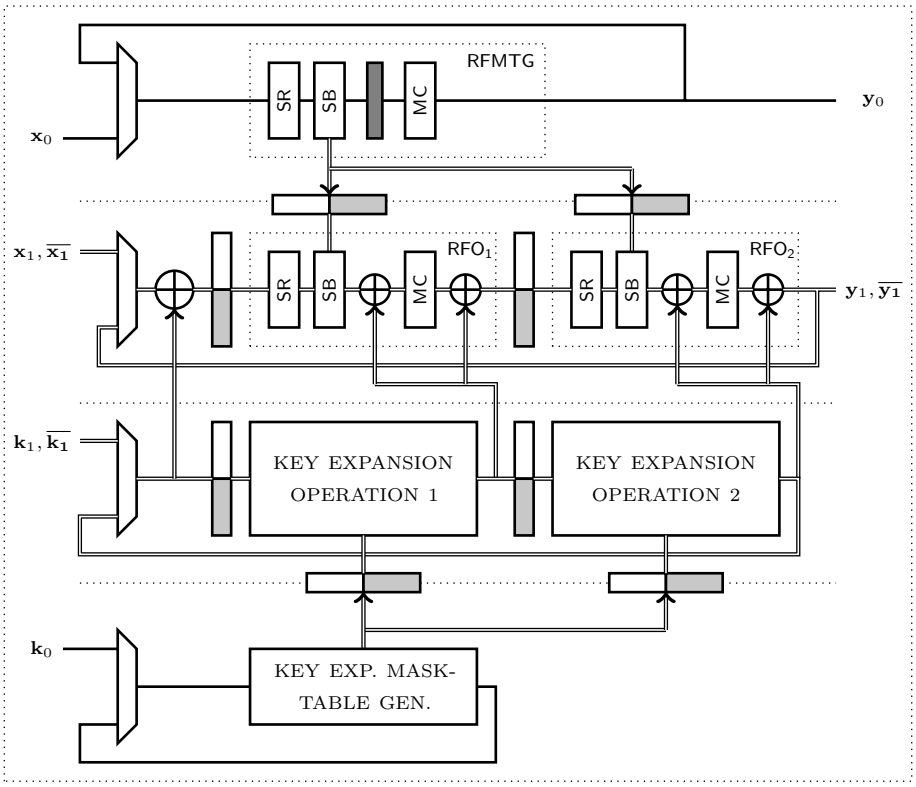


Figure 4: Basic Architecture of the Protected Low-Latency AES. Logic flows left to right and in the direction of the arrows.

Figure 4 outlines the basic architecture of our protected single-cycle per round AES hardware implementation. Well recognizable, the data path in the center of the figure uses two round functions based on dual-rail logic cells, separated by register stages to avoid idle cycles due to the pre-charge and evaluation scheme. In addition, our architecture can leverage the original mask share update and table generation design which already allows to generate a new table in every cycle. By writing the table to two separate registers, depending on the currently active round function, we can still ensure the pre-charging of the table without duplication of the table generation logic.

Encryption-only and decryption-only variants. For some applications, having a combined encryption-decryption architecture might not be necessary. For these scenarios, we opted to implement encryption-only and decryption-only variants by removing the unnecessary parts of the data path shown in Figure 4. In particular, this will reduce the area demands of our architecture and also helps to improve the critical path delay and maximum frequency.

Key expansion protection variant. By default our architecture uses a shared key schedule. Hence, the key schedule is protected under a threat model in which the adversary has full control over the key changing process and the key is not protected by other means at a system level. For this, we can apply the same principle of duplication to the key expansion module to provide a shared round key in every cycle without violating the pre-charge and evaluation principle. However, if certain applications can waive additional SCA-protection of the key expansion module, we can reduce the area demands again

Table 2: Area and power results at 100 MHz after synthesis using GF28nm.

Module	Protected Key Expansion			Standard Key Expansion		
	Enc./Dec. [kGE]	Enc. [kGE]	Dec. [kGE]	Enc./Dec. [kGE]	Enc. [kGE]	Dec. [kGE]
Cryptographic Engine (AES)	**174.4**	**157.5**	**167.2**	**136.3**	**123.1**	**129.2**
control & connection	0.4	0.5	0.5	0.4	0.4	0.5
data mask-table generation	14.9	12.2	12.9	14.1	11.7	12.2
data operation layer	114.4	103.8	109.1	114.3	103.8	109.0
key mask-table generation	5.3	5.1	5.3	-	-	-
key operation layer	39.4	35.9	39.4	7.5	7.2	7.5
Entropy Engine (PRNG)	**14.8**	**14.8**	**14.8**	**11.2**	**11.2**	**11.2**
Power Consumption [mW]	**4.728**	**4.517**	**4.661**	**3.627**	**3.494**	**3.608**

and implement a standard key expansion module without special consideration of key protection and sharing.

4.3 Implementation Results

This section summarizes and presents area and performance results of our different hardware implementations that we derived after synthesis using an ASIC design flow. All designs have been described and implemented using Verilog as Hardware Description Language (HDL) and were compiled and synthesized using Synopsys Design Compiler K-2015.06-SP3-1. For synthesis, we used a standard cell library from Global Foundry with $28nm$ RVT cells. All designs were synthesized with a target frequency of 100 MHz and area results are normalized in terms of Gate Equivalents (GE) using the size of $0.624\,\mu m^2$ for the standard two-input NAND gate of the selected library.

Table 2 shows the results of our different implementations. In total, we implemented six different designs: two cores supporting encryption and decryption, two cores supporting encryption-only operations, and two cores supporting decryption-only operations. For each of these options on the direction of operation, the first core uses a protected key expansion while the second core uses a standard key expansion and only processes plain keys internally.

The results show that our encryption-only variants require about 10 % less area while the decryption-only variant saves 5 % compared to the combined design. In addition, using a protected key expansion increases the area requirements by about 28 % compared to the standard key expansion.

In terms of maximum frequency, our core meets all path timings using a target clock of 400 MHz. Note that we used only Standard VT cells, i.e., regular RVT cells, for the synthesis and picked worst case (slow-slow) corner case results. Higher frequencies of up to 600 MHz can be obtained using a mix of standard and low-voltage threshold (LVT) cells and paying the price of a higher power consumption.

It is worth to mention that all our area and performance numbers include numbers for the core and an internal PRNG as well. The PRNG is used to generate the necessary masks for the LMDPL countermeasures. Masks are needed in every clock cycle, so we decided to integrate a PRNG based on Keccak-f, similar to [BDPA10], which did not impact critical path in our design. The PRNG has an internal state of 650-1050 bits (depending on the core configuration) and is periodically fed with a 128-bit entropy seed. In every cycle we are able to fetch up to 976 bits. This provided enough diffusion for LMDPL masking purposes, as is demonstrated in the following section.

4.4 Comparison to Related Work

We now compare our solution to related work. For the comparison, we consider different masked implementations and compare their area, latency, and randomness requirements, for a single AES S-box only[2].

Table 4 lists related work in comparison with our design. Our work requires 3.48 kGEs of area which includes the S-box operation logic itself (1,137 GEs), the mask table generation (611 GEs), and the 288-bit mask-table register (1,728 GEs). The latency requirement is one clock cycle and it needs 36 bits of fresh randomness.

Admittedly, neither the area nor the randomness requirement of our single S-box implementation is the smallest in literature. Moreover, the duality of the state and key registers creates extra burden in terms of area when we look at the overall design. However, in this paper, our goal is to provide a low-latency implementation for which our implementation is the best presented so far.

The only other work reporting similar latency numbers (single-cycle execution) was that of Gross et al. [GIB18]. Their S-box implementation is based on Domain-Oriented Masking (DOM) and expands all mask shares until no domain collisions are found. The shares are then re-combined again to $d + 1$ shares after the S-box computation is done. This requires a costly register stage. They reported an area requirement of about 60 kGEs and need 2,048 bits of randomness. This makes their solution hard to justify for practical implementations. Note that a fully parallel AES implementation would theoretically need about 1 million gates and 32 kBits of randomness. However, they also proposed a two cycle per round S-box implementation which requires 6.74 kGEs of area and 416 bits of randomness.

Leiserson et al. [LMW14] proposed an LMDPL-based AES S-box which requires 2.83 kGEs. Because of the same underlying masking technique, randomness requirements are the same. Bilgin et al. [BGN+14b, BGN+15] presented results of a Threshold Implementation of AES. There are many other designs based on the same or similar masking techniques, including [MPL+11, GC17, DCRB+16, UHA17, Sug19, WM18], but all of them require at least three clock cycles to perform a masked S-box computation.

In De Meyer et al. [DRB18], the authors presented a low-latency S-box based on multiplicative masking. Their design only needs 1.69 kGEs, requires 2 cycles of latency, and only 19 bits of randomness. However, when integrated in a full AES design, their design requires three additional clock cycles to handle the zero-value problem that usually arises when using this type of masking.

5 Side-Channel Analysis of Our Design

This section presents results of practical side-channel analysis of our low-latency masked AES implementation (version with protected key expansion and support for encryption and decryption). We analyzed the security of our proposal using real-world power measurements. We applied the Test Vector Leakage Assessment (TVLA) methodology [GJJR11] for this purpose and examined design up to the fourth-order statistical moment in a univariate attack setting and also analyzed the first-order moment in a bivariate attack setting to gain good confidence in its DPA resistance level.

Measurement Setup. We used a custom FPGA prototyping platform for side-channel analysis. The platform assembles a Xilinx Kintex-7 FPGA, has external SRAM memory to store and load cipher input and output data, a USB-to-serial connection, power and IO pins, and a dedicated SMA connector to measure the power consumption of the FPGA. We measured the voltage drop across a measurement resistor using an 8-bit 1 GHz Tektronix

[2] Note that these implementations use different CMOS libraries.

Table 3: Comparison of masked AES S-box implementations and their performance.

Works	Area [kGE]	Latency [cycles]	Randomness [bits]
Gross et al. [GIB18]	60.73	1	2,048
Gross et al. [GIB18]	6.74	2	416
Moradi et al. [MPL+11]	4.24	4	48
Wegener and Moradi. [WM18]	4.20	16	0
Bilgin et al. [BGN+14b]	3.71	3	44
Sugawara [Sug19]	3.50	4	0
Ghoshal and De Cnudde [GC17]	2.91	3	20
Bilgin et al. [BGN+15]	2.84	3	32
Leiserson et al. [LMW14]	2.83	2	36
Gross et al. [GM18]	2.20	8	18
De Cnudde et al. [DCRB+16]	1.98	6	54
De Meyer et al. [DRB18]	1.69	2+3	19
Ueno et al. [UHA17]	1.42	5	64
Our Solution	**3.48**	**1**	**36**

digital oscilloscope (DPO7104C). All data in this section have been sampled using 1 GS/s and a 500 MHz bandwidth low-pass filter. Using our setup, we are able to capture 50,000 AES operations in a single power trace. No amplifiers have been used in our experiments. The target frequency of the core inside the FPGA was set to 50 MHz.

Our setup is further composed of a control PC that connects to the FPGA platform. All the data, keys, and random number seed values are stored in SRAM before starting a DPA acquisition. Our FPGA platform is loading and storing input and output data from SRAM using a few clock cycles, which allows fast data processing and power trace acquisitions.

In order to avoid input and output leaking in our TVLA tests, we provide the data to and from the FPGA in Boolean shares. The core under test (AES) or logic inside the FPGA does not mask or unmask the inputs or outputs respectively. Instead, our control PC splits up the inputs in shares and combines the shares again when receiving the output shares.

Mask Generation. As opposed to many related work, we consider mask generation as part of the overall system (core under test). We therefore decided to include a Pseudo-Random Number Generator (PRNG) inside the FPGA that is used to generate all necessary mask values for our LMDPL gadgets inside our AES implementation.

This PRNG is seeded with a random 128-bit value in the beginning of the measurement. The initial sharing of the input data and the key is performed externally and fed to the core. The LMDPL mask values however are generated by the PRNG that runs in parallel to the AES core. We believe that the advantage of analyzing the entire system including mask generation in the field out-weighs the disadvantage of the lower Signal-to-Noise (SNR) ratio of the acquisition setup because of adding the PRNG as noise source. We are able to evaluate both the core and the entropy quality of the mask generation logic without needing to re-evaluate the DPA resistance level after the core has been deployed in a larger "stand-alone" system in practice.

To verify correctness of our setup, we implemented a *maskDisable* control signal at the interface level of our prototype implementation to disable the DPA protection. If asserted, the output of the PRNG is gated to zero, which causes the core to leak information in side channels (unprotected). If not asserted, the core is fed with random masks (protected).

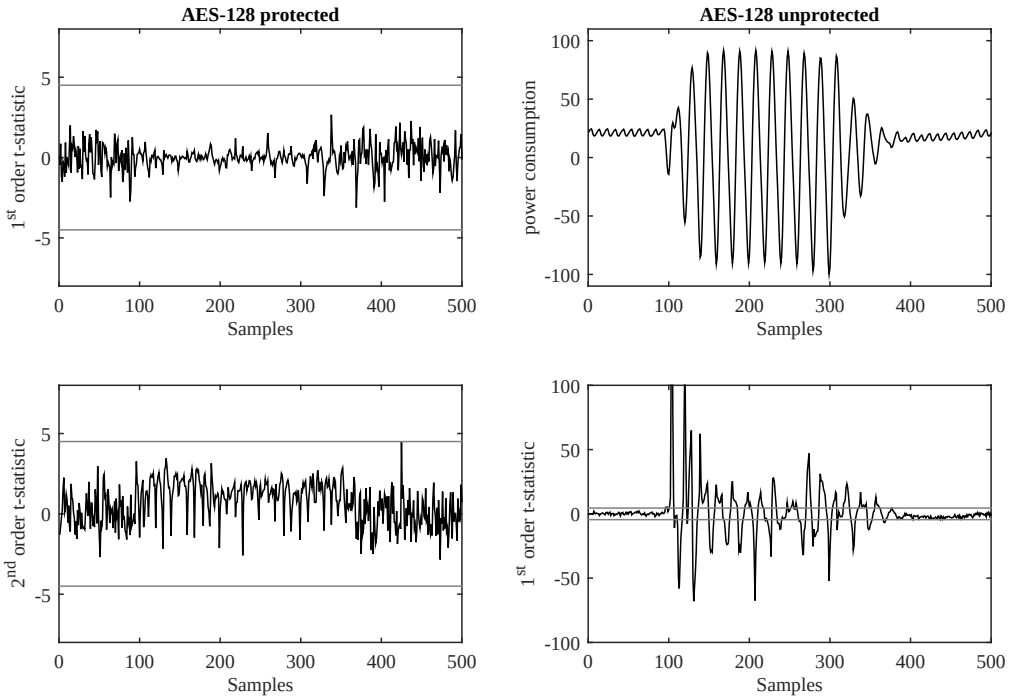


Figure 5: Univariate fixed-vs.-random t-test results for AES-128. The left two plots show the results for the first two statistical order moments using 100 million traces when mask generation is enabled. The upper right plot shows the average trace over 1 million encryption operations, the lower right plot shows the result for the first-order statistical moment using 1 million traces when mask generation is disabled.

5.1 Univariate Analysis

We now discuss the DPA resistance level of our implementation against univariate attacks. We analyzed each sample point independently and evaluated if information was leaking across an entire AES operation. Our security target was to resist up to 100 million power traces which we present for first- and second-order statistical moments.

Afterwards, we analyze the resistance level of the implementation using more than 100 million traces.

Results using 100 million traces. In the following, we investigate non-specific tests using a fixed-vs.-random Welch t-test. Figure 5 shows the results for AES-128. The two plots on the left side show the result for a t-test using 100 million traces in case the mask generation logic is enabled. Both the first- and the second-order statistical moments do not contain significant leakage and are within the 4.5 sigma interval. The same results were obtained for higher moments as well.

The two plots on the right side show our results when the mask generation logic was gated to output all zeros. The upper right plot shows the average trace over 1 million traces. The ten rounds of AES can be clearly identified. The initial and last cycle shows lower mean power consumption and is related to the input and output loading of the data and key. The lower right plot shows the first-moment t-statistic using 1 million traces. Leakages are observable throughout the operation with an initial leak of up to 220 sigma which can be explained by the lower noise level in the first cycle when inputs are loaded and AES is idle. Also, round functions are implemented in dual-rail logic, hiding some

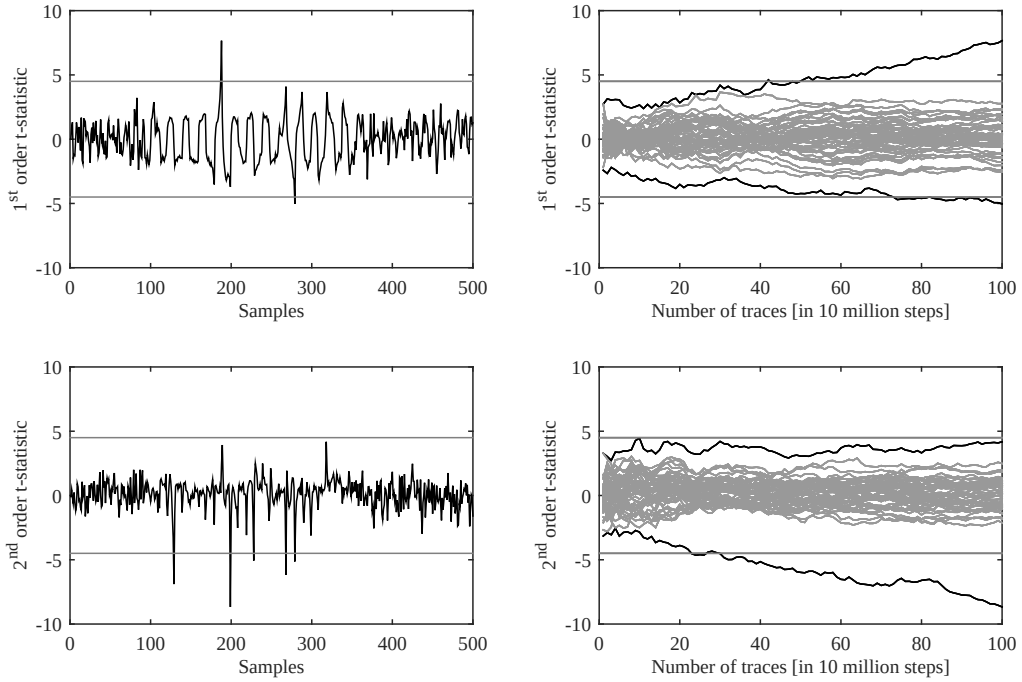


Figure 6: Univariate fixed-vs.-random t-test results for AES-128 using 1 billion traces. The left two plots show the results for the first two statistical order moments. The right two plots show the t-statistics as a function of the number of traces.

leakage. We also analyzed higher-order moments and can confirm leakage with sigma values of more than 30. Similar results have been obtained when using 256-bit keys.

Interpretation of the Results. Our results show that our implementation resists univariate attacks up to 100 million traces with high probability. There was no significant leakage in any of the observed statistical moments. Our LMDPL masking countermeasure is constructed using a Boolean masking scheme where secrets are split up into two Boolean shares. In addition, the use of dual-rail logic provides an efficient hiding countermeasure. The dual-rail logic style both lowers the signal level and adds noise, resulting in a very low signal-to-noise ratio. The combination of both masking and hiding is the reason we not only get good resistance levels in the first order but also in higher orders as well.

Results using 1 billion traces. We decided to evaluate our implementation using more than 100 million traces and performed 1 billion trace acquisitions. The main reason for this was to clarify when the device starts to leak information and what the actual resistance level is. Additionally, we wanted to clarify whether some of the peaks shown in the previous results were anomalies or truly indicated information leakage. For example, we wanted to see whether or not the peak at sample point 420 in the second-moment results in Figure 5 would cross the significance border of 4.5 sigma when collecting more traces.

Figure 6 shows the results using 1 billion traces. The left two plots show the t-test statistics of the first two statistical moments. Interestingly, we can identify first and second order leaks. The position of the highest first order leak, however, does not leak in the second-order case. We also observe that there are 5 peaks crossing the 4.5 boundary in the second order test, versus 2 in the first order test.

The two plots on the right side show the leakage as a function of the number of traces. It can be seen that the implementation starts leaking in the first order after about 400

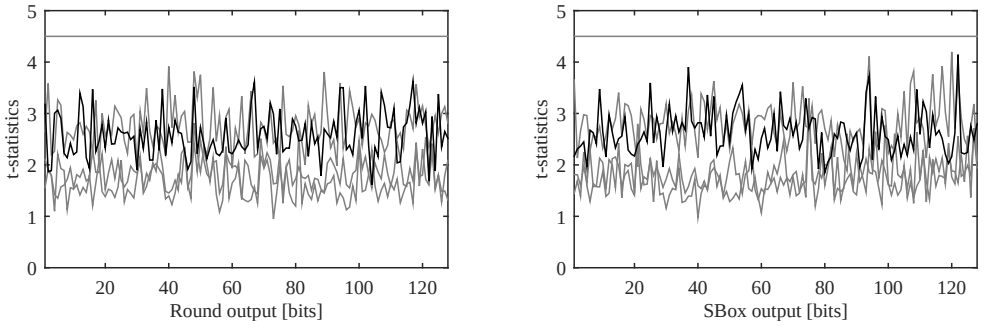


Figure 7: Univariate results of specific t-tests using 2 billion traces targeting the fifth round of AES-128 (absolute t-statistics of round output and S-box output). First-order results are drawn in black, second-order up to fourth-order results are drawn in gray.

million traces. Leakage in the second order starts showing up after about 220 million traces. We also analyzed the third and fourth statistical moments, but could not identify any leakages.

We also targeted specific intermediates of AES to test if we could extract enough information to reveal the key using a practical attacking scenario. We targeted the output of the round and the output of all 16 S-boxes for round 5 of AES-128, and round 7 of AES-256. For these tests, we decided to collect twice as much traces and acquired 2 billion traces instead of 1 billion, because key-extraction attacks usually require more traces to exploit leakage that may was identified earlier in the analysis process, for example, using non-specific fixed-vs-random t-tests. Figure 7 shows the results for the first four statistical order moments (mean, variance, skewness, and kurtosis). We plotted the absolute t-statistics over all 128 bits and the 4.5 sigma border is marked with a gray horizontal line. All of the key-extraction attempts were unsuccessful and we were unable to recover any key bits.

Interpretation of the Results. The first-order leakages that we observed can be explained by the fact that our current leakage models assume an independent power consumption of the shares in masked implementations. In practice, it has been shown that this might not be the case. For example, in [CEM18] the authors identified leakages in a 2-share masked implementation due to fluctuations of the voltage supply. They conducted the experiments on linear functions only which means that these physical effects are independent of the underlying masking scheme and can be applied to LMDPL as well. Similar effects have been also described in [CBG+17] where coupling, signal crosstalk, and IR-drop effects impact the security of masked implementations.

In terms of second-order analysis, we expect that our implementation shows leakage because our LMDPL masking countermeasure uses only 2 shares. However, the use of dual-rail logic and its power equalization property pushed the resistance level against higher-order attacks beyond 100 million traces. Noise has a large impact in higher-order statistics, and our design doubles the number of S-boxes to achieve single-cycle operations. In addition, the AES key-schedule is protected as well, requiring 4 additionally masked AES S-boxes. Furthermore, our PRNG produces 976 random bits during each cycle (256 bits to refresh the input-data mask and the 128/256-bit key, and 720 bits for the 16+4 LMDPL S-box implementations) which added enough noise in the higher order statistics to prevent extraction of useful information up to our targeted 100 million trace resistance level.

5.2 Multivariate Analysis

We also analyzed our implementation using bivariate statistics. For the analysis, we used the same settings as for the univariate tests. We also decided to calculate the bivariate statistics over the entire AES power trace, using 500 sample points instead of only a few AES rounds.

TVLA testing was done as follows. First, we normalized every sample point by subtracting the corresponding mean sample points. Then, we combined the leakage samples into a single variable by multiplication. A first-order t-test was performed on the resulting traces to evaluate if there was leakage in the mean of the joined sample distributions. Figure 8 shows the results for AES-128 encryption operations. It shows the 500 sample points on the x-axis and the y-axis. All gray dots show t-statistics within the 4.5 sigma interval whereas red and blue dots show significant t-statistics larger than ± 4.5 sigma. There are two triangles/corners in the plot. The diagonal line, which separates the two triangles, represents squared sample variables which equals to the variance (univariate second-order moments) of the sample distributions. All other variables above or below the diagonal line represent the mean samples of the bivariate distributions.

The lower triangle represents the results using 100k traces when the PRNG was turned off. In this case, we can identify significant bi-variate leaks especially around sample point 100 which is where the input data is leaking strongly, c.f., lower right plot in Figure 5). Leakages can also be observed at other locations in the lower triangle as well, for example, at sample point 225 and around 285. Note that every (univariate) sample point that is leaking information will show leakages in all combined sample point distributions as well, which is the reason for the corner-shaped leakage patterns in the lower left triangle shown in Figure 8. The given leakages prove that our setup is working correctly and that the implementation is leaking the LMDPL intermediates as expected.

The upper right triangle shows the results for 1 billion traces where the PRNG was turned on (countermeasure enabled). Most of the t-statistics are gray and within the 4.5 sigma interval, but small leakages can be identified around the diagonal line (not only on the line itself), for example, at sample point 110 and 180. That means that the implementation shows bi-variate leakage as we expected but the observed leakage is very weak. One billion traces had to be used for these leakages to show up in the t-statistics.

6 Conclusions

In this paper, we presented the first practical, hardware-masked, single-cycle-per-round AES implementation, proved its first-order security under the d-glitch extended probing model, and verified our design empirically.

We researched the current state-of-the-art masking techniques, and decided to focus our efforts on the LMDPL gadget. After reviewing security concepts and models, we showed that the LMDPL gadget is first-order secure using the d-GSNI property. Using this information on how to compose LMDPL gadgets, we designed a secure hardware-masked AES implementation which computes a single round with a latency of one clock cycle. We empirically verified the security of our implementation by collecting 100 million power traces and analyzing them based on the TVLA methodology. We also collected and analyzed up to two billion traces to see at what point the design starts leaking.

While our design uses two shares and is only intended to be secure against first-order analysis, it demonstrated significant higher-order resistance as well. We believe much of this resistance is due to the amount of noise in our single-cycle per round AES design. LMDPL might not exhibit the same level of higher-order resistance if it is used in algorithms and designs with smaller and/or fewer S-boxes. We believe that extending LMDPL to higher-order masking should be an interesting and useful topic for further research.

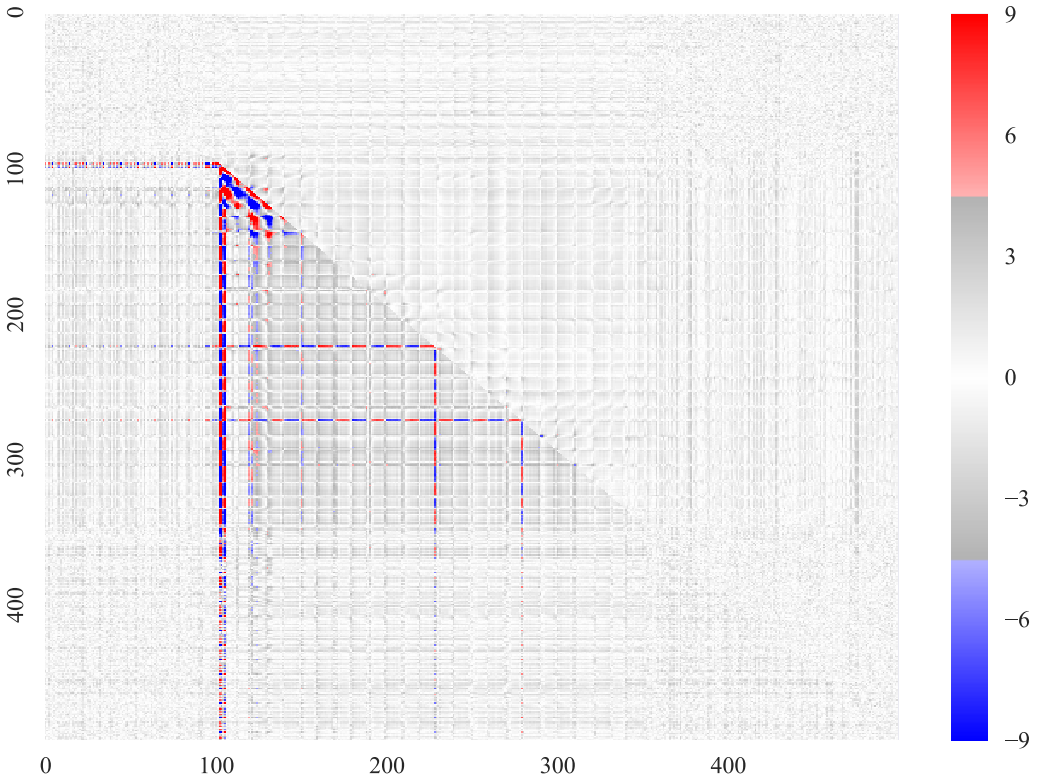


Figure 8: Bivariate fixed-vs.-random t-test results for AES-128. The lower left corner shows the results when mask generation is disabled using 100k traces. The upper right corner shows the results when mask generation is enabled using 1 billion traces.

References

[ABP+18] Victor Arribas, Begül Bilgin, George Petrides, Svetla Nikova, and Vincent Rijmen. Rhythmic Keccak: SCA security and low latency in HW. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(1):269–290, 2018.

[BBD+16] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, Pierre-Yves Strub, and Rébecca Zucchini. Strong non-interference and type-directed higher-order masking. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 116–129. ACM, 2016.

[BBD+18] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, François-Xavier Standaert, and Pierre-Yves Strub. Improved parallel mask refreshing algorithms: Generic solutions with parametrized non-interference & automated optimizations. *IACR Cryptology ePrint Archive*, 2018:505, 2018.

[BCG+12] Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knezevic, Lars R. Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar, Christian Rechberger, Peter Rombouts, Søren S. Thomsen, and Tolga Yalçin. PRINCE - A low-latency block cipher for pervasive computing applications

- extended abstract. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, volume 7658 of *Lecture Notes in Computer Science*, pages 208–225. Springer, 2012.

[BDPA10] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Sponge-Based Pseudo-Random Number Generators. In *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*, pages 33–47, 2010.

[BGN+14a] Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. Higher-order threshold implementations. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II*, volume 8874 of *Lecture Notes in Computer Science*, pages 326–343. Springer, 2014.

[BGN+14b] Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. A more efficient AES threshold implementation. In David Pointcheval and Damien Vergnaud, editors, *Progress in Cryptology – AFRICACRYPT 2014*, pages 267–284, Cham, 2014. Springer International Publishing.

[BGN+15] Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Nikov Ventzislav, and Vincent Rijmen. Trade-offs for threshold implementations illustrated on AES. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(7):1188–1200, July 2015.

[BKN18] Dušan Božilov, Miroslav Knežević, and Ventzislav Nikov. Optimized threshold implementations: Securing cryptographic accelerators for low-energy and low-latency applications. Cryptology ePrint Archive, Report 2018/922, 2018.

[Can05] David Canright. A very compact s-box for AES. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 441–455. Springer, 2005.

[CBG+17] Thomas De Cnudde, Begül Bilgin, Benedikt Gierlichs, Ventzislav Nikov, Svetla Nikova, and Vincent Rijmen. Does coupling affect the security of masked implementations? In *Constructive Side-Channel Analysis and Secure Design - 8th International Workshop, COSADE 2017, Paris, France, April 13-14, 2017, Revised Selected Papers*, pages 1–18, 2017.

[CEM18] Thomas De Cnudde, Maik Ender, and Amir Moradi. Hardware masking, revisited. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(2):123–148, 2018.

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999.

[CPRR13] Jean-Sébastien Coron, Emmanuel Prouff, Matthieu Rivain, and Thomas Roche. Higher-order side channel security and mask refreshing. In Shiho Moriai, editor, *Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers*, volume 8424 of *Lecture Notes in Computer Science*, pages 410–424. Springer, 2013.

[Dae17] Joan Daemen. Changing of the guards: A simple and efficient method for achieving uniformity in threshold sharing. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, volume 10529 of *Lecture Notes in Computer Science*, pages 137–153. Springer, 2017.

[DBR19] Lauren De Meyer, Begül Bilgin, and Oscar Reparaz. Consolidating security notions in hardware masking. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(3):119–147, 2019.

[DCRB+16] Thomas De Cnudde, Oscar Reparaz, Begül Bilgin, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. Masking AES with d+1 shares in hardware. In *Proceedings of the 2016 ACM Workshop on Theory of Implementation Security*, TIS '16, pages 43–43, New York, NY, USA, 2016. ACM.

[DDF14] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 423–440. Springer, 2014.

[DMW18] Lauren De Meyer, Amir Moradi, and Felix Wegener. Spin me right round rotational symmetry for FPGA-specific AES. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(3):596–626, 2018.

[DRB18] Lauren De Meyer, Oscar Reparaz, and Begül Bilgin. Multiplicative masking for AES in hardware. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(3):431–468, 2018.

[FGP+18] Sebastian Faust, Vincent Grosso, Santos Merino Del Pozo, Clara Paglialonga, and François-Xavier Standaert. Composable masking schemes in the presence of physical defaults & the robust probing model. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(3):89–120, 2018.

[GC17] Ashrujit Ghoshal and Thomas De Cnudde. Several masked implementations of the Boyar-Peralta AES s-box. In *Progress in Cryptology - INDOCRYPT 2017 - 18th International Conference on Cryptology in India, Chennai, India, December 10-13, 2017, Proceedings*, pages 384–402, 2017.

[GIB18] Hannes Groß, Rinat Iusupov, and Roderick Bloem. Generic low-latency masking in hardware. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(2):1–21, 2018.

[GJJR11] Gilbert Goodwill, Benjamin Jun, Josh Jaffe, and Pankaj Rohatgi. A testing methodology for side-channel resistance validation. NIST NIAT Workshop, 2011.

[GM18] Hannes Groß and Stefan Mangard. A unified masking approach. *J. Cryptographic Engineering*, 8(2):109–124, 2018.

[GMK16] Hannes Groß, Stefan Mangard, and Thomas Korak. Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order. In Begül Bilgin, Svetla Nikova, and Vincent Rijmen, editors, *Proceedings of the ACM Workshop on Theory of Implementation Security, TIS@CCS 2016 Vienna, Austria, October, 2016*, page 3. ACM, 2016.

[GSM17] Hannes Groß, David Schaffenrath, and Stefan Mangard. Higher-order side-channel protected implementations of KECCAK. In Hana Kubátová, Martin Novotný, and Amund Skavhaug, editors, *Euromicro Conference on Digital System Design, DSD 2017, Vienna, Austria, August 30 - Sept. 1, 2017*, pages 205–212. IEEE Computer Society, 2017.

[ISW03] Yuval Ishai, Amit Sahai, and David A. Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *CRYPTO 1999, Santa Barbara, California, USA, August 15-19.*, pages 388–397, 1999.

[KKT06] Konrad J. Kulikowski, Mark G. Karpovsky, and Alexander Taubin. Power attacks on secure hardware based on early propagation of data. In *Proceedings of the 12th IEEE International Symposium on On-Line Testing*, IOLTS '06, pages 131–138, Washington, DC, USA, 2006. IEEE Computer Society.

[LMW14] Andrew J. Leiserson, Mark E. Marson, and Megan A. Wachs. Gate-level masking under a path-based leakage metric. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*, volume 8731 of *Lecture Notes in Computer Science*, pages 580–597. Springer, 2014.

[MMSS19] Thorben Moos, Amir Moradi, Tobias Schneider, and François-Xavier Standaert. Glitch-resistant masking revisited or why proofs in the robust probing model are needed. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(2):256–292, 2019.

[MPG05] Stefan Mangard, Thomas Popp, and Berndt M. Gammel. Side-channel leakage of masked CMOS gates. In Alfred Menezes, editor, *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*, volume 3376 of *Lecture Notes in Computer Science*, pages 351–365. Springer, 2005.

[MPL+11] Amir Moradi, Axel Poschmann, San Ling, Christof Paar, and Huaxiong Wang. Pushing the limits: A very compact and a threshold implementation of AES. In *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, pages 69–88, 2011.

[MS16] Amir Moradi and Tobias Schneider. Side-Channel Analysis Protection and Low-Latency in Action - Case Study of PRINCE and Midori. In Jung Hee

Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 517–547, 2016.

[NRR06] Svetla Nikova, Christian Rechberger, and Vincent Rijmen. Threshold implementations against side-channel attacks and glitches. In *ICICS 2006, Raleigh, NC, USA, December 4-7.*, pages 529–545, 2006.

[PKZM07] Thomas Popp, Mario Kirschbaum, Thomas Zefferer, and Stefan Mangard. Evaluation of the masked logic style MDPL on a prototype chip. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 81–94. Springer, 2007.

[PR11] Emmanuel Prouff and Thomas Roche. Higher-order glitches free implementation of the AES using secure multi-party computation. *IACR Cryptology ePrint Archive*, 2011:413, 2011.

[PR13] Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*, pages 142–159. Springer, 2013.

[RBN+15] Oscar Reparaz, Begül Bilgin, Svetla Nikova, Benedikt Gierlichs, and Ingrid Verbauwhede. Consolidating masking schemes. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 764–783. Springer, 2015.

[RP10] Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of AES. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*, volume 6225 of *Lecture Notes in Computer Science*, pages 413–427. Springer, 2010.

[SBY+18] Danilo Sijacic, Josep Balasch, Bohan Yang, Santosh Ghosh, and Ingrid Verbauwhede. Towards efficient and automated side channel evaluations at design time. In Lejla Batina, Ulrich Kühne, and Nele Mentens, editors, *PROOFS 2018, 7th International Workshop on Security Proofs for Embedded Systems, colocated with CHES 2018, Amsterdam, The Netherlands, September 13, 2018*, volume 7 of *Kalpa Publications in Computing*, pages 16–31. EasyChair, 2018.

[SS06] Daisuke Suzuki and Minoru Saeki. Security evaluation of DPA countermeasures using dual-rail pre-charge logic style. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, volume 4249 of *Lecture Notes in Computer Science*, pages 255–269. Springer, 2006.

[Sug19] Takeshi Sugawara. 3-share threshold implementation of AES s-box without fresh randomness. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(1):123–145, 2019.

[UHA17] Rei Ueno, Naofumi Homma, and Takafumi Aoki. Toward more efficient DPA-resistant AES hardware architecture based on threshold implementation. In Sylvain Guilley, editor, *Constructive Side-Channel Analysis and Secure Design*, pages 50–64, Cham, 2017. Springer International Publishing.

[WM18] Felix Wegener and Amir Moradi. A first-order SCA resistant AES without fresh randomness. In Junfeng Fan and Benedikt Gierlichs, editors, *Constructive Side-Channel Analysis and Secure Design - 9th International Workshop, COSADE 2018, Singapore, April 23-24, 2018, Proceedings*, volume 10815 of *Lecture Notes in Computer Science*, pages 245–262. Springer, 2018.

A New LMDPL Gadgets

Here, we describe two additional gadgets that we designed to implement our low-latency AES. Similar to other LDMPL gadgets, these should also be implemented using monotonic gates and be pre-charged to provide first-order security.

A.1 Selective XOR

The function under consideration is

$$x = a \oplus (be).$$

Here e act as an enabling bit that is constant through the whole design (e.g., chooses encryption or decryption operation) and do not depend on a secret. Hence, the corresponding LMDPL gadget is very similar to that of an XOR gadget.

$$\begin{aligned} x_1 &= a_1 \oplus b_1 e, \\ &\cdots\cdots\cdots\cdots\cdots\cdots \\ x_2 &= e(\overline{a_2} b_2 \vee a_2 \overline{b_2}) \vee \overline{e} a_2 \\ \overline{x_2} &= e(\overline{a_2}\,\overline{b_2} \vee a_2 b_2) \vee \overline{e a_2}. \end{aligned}$$

A.2 Multiplexer

Here we have a simple multiplexer

$$x = a \oplus s(a \oplus b)$$

where s is a variable that does not depend on the sensitive value (e.g., last round selection). The corresponding LMDPL gadget is constructed as follows:

$$\begin{aligned} x_1 &= a_1 \oplus s(a_1 \oplus b_1), \\ &\cdots\cdots\cdots\cdots\cdots\cdots \\ x_2 &= s b_2 \vee \overline{s} a_2 \\ \overline{x_2} &= s\overline{b_2} \vee \overline{s a_2}. \end{aligned}$$

B Exemplary Glitch Extended Probe Table

Table 4: Showing $I(\ \mathcal{O};\ (a_1, b_1) \mid (a_2, b_2)\ ) = 0$ for the LMDPL AND gadget.

a_1	b_1	r	a_2	b_2	s^0	s^1	s^2	s^3	$\overline{x_2}$	(a_2, b_2, s^0)	$(a_2, b_2, s^0, s^1, s^2, s^3, \overline{x_2})$
0	0	0	0	0	1	0	0	0	1	1	17
0	0	0	0	1	0	0	1	0	1	2	37
0	0	0	1	0	0	1	0	0	1	4	73
0	0	0	1	1	0	0	0	0	0	6	96
0	0	1	0	0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	2	32
0	0	1	1	0	0	0	0	0	0	4	64
0	0	1	1	1	0	0	0	1	1	6	99
0	1	0	0	0	1	0	0	0	1	1	17
0	1	0	0	1	0	0	1	0	1	2	37
0	1	0	1	0	0	0	0	0	0	4	64
0	1	0	1	1	0	0	0	1	1	6	99
0	1	1	0	0	0	0	0	0	0	0	0
0	1	1	0	1	0	0	0	0	0	2	32
0	1	1	1	0	0	1	0	0	1	4	73
0	1	1	1	1	0	0	0	0	0	6	96
1	0	0	0	0	1	0	0	0	1	1	17
1	0	0	0	1	0	0	0	0	0	2	32
1	0	0	1	0	0	1	0	0	1	4	73
1	0	0	1	1	0	0	0	1	1	6	99
1	0	1	0	0	0	0	0	0	0	0	0
1	0	1	0	1	0	0	1	0	1	2	37
1	0	1	1	0	0	0	0	0	0	4	64
1	0	1	1	1	0	0	0	0	0	6	96
1	1	0	0	0	0	0	0	0	0	0	0
1	1	0	0	1	0	0	1	0	1	2	37
1	1	0	1	0	0	1	0	0	1	4	73
1	1	0	1	1	0	0	0	1	1	6	99
1	1	1	0	0	1	0	0	0	1	1	17
1	1	1	0	1	0	0	0	0	0	2	32
1	1	1	1	0	0	0	0	0	0	4	64
1	1	1	1	1	0	0	0	0	0	6	96