Cryptanalysis of Efficient Masked Ciphers: Applications to Low Latency

Tim Beyne*[1], Siemen Dhooghe*[1], Amir Moradi*[2] and Aein Rezaei Shahmirzadi*[2]

[1] imec-COSIC, ESAT, KU Leuven, Leuven, Belgium
firstname.lastname@esat.kuleuven.be
[2] Ruhr University Bochum, Horst Görtz Institute for IT Security, Bochum, Germany
firstname.lastname@rub.de

Abstract. This work introduces second-order masked implementation of LED, Midori, Skinny, and Prince ciphers which do not require fresh masks to be updated at every clock cycle. The main idea lies on a combination of the constructions given by Shahmirzadi and Moradi at CHES 2021, and the theory presented by Beyne et al. at Asiacrypt 2020. The presented masked designs only use a minimal number of shares, i.e., three to achieve second-order security, and we make use of a trick to pair a couple of S-boxes to reduce their latency. The theoretical security analyses of our constructions are based on the linear-cryptanalytic properties of the underlying masked primitive as well as SILVER, the leakage verification tool presented at Asiacrypt 2020. To improve this cryptanalytic analysis, we use the *noisy probing model* which allows for the inclusion of noise in the framework of Beyne et al. We further provide FPGA-based experimental security analysis confirming second-order protection of our masked implementations.

Keywords: Hardware · Linear Cryptanalysis · Masking · Probing Security · Side-Channel Analysis · Threshold Implementations

1 Introduction

Ever since the introduction of differential power analysis by Kocher *et al.* [KJJ99] in 1999, the cryptographic hardware community has been looking for countermeasures to protect embedded devices. The benefits as well as difficulties of masking as a countermeasure against side-channel analysis attacks, have been proven through several scientific articles and experimental investigations. Masked implementations can be made efficient towards a cost function like area or latency, and their security can be proven using abstractions such as the probing model.

Nevertheless, masking typically relies on fresh randomness in order to provide security in the probing model. The generation of this randomness is often costly, and its security requirements are currently not very well known. The cost of this generation is also not often reported in academic literature, leading to a biased view of the efficiency of certain countermeasures.

For first-order security, one can use threshold implementations as proposed in 2006 by Nikova *et al.* [NRR06]. Thanks to the property that these maskings maintain uniformity, it is possible to build secure masked circuits which do not require any fresh randomness.

*Authors list in alphabetical order; see https://www.ams.org/profession/leaders/culture/CultureStatement04.pdf

Recently, due to techniques of Daemen [Dae17] and Shahmirzadi and Moradi [SM21a], such maskings have become easier to design and their costs have decreased. However, their extension to higher-order security is not straightforward. Higher-order threshold implementations were first proposed by Bilgin *et al.* [BGN+14] and later shown to be not fully secure by Reparaz [Rep15].

In CHES 2021, Shahmirzadi and Moradi [SM21b] proposed several efficient second-order masking of popular symmetric primitives that require a significantly lower amount of randomness compared to other known masked designs. Nevertheless, except for Keccak, their masked designs still use fresh randomness.

At Asiacrypt 2020, Beyne *et al.* [BDZ20] proposed a framework to evaluate the security of higher-order threshold implementations. They applied their framework to design a second-order implementation of the LED cipher [GPPR11]. However, the underlying masked design requires seven shares – making it inefficient. Further, their security claims, which have not been validated in practice, are based on a too restrictive security model. They consider adversaries who can collect a large number of traces, *e.g.* 2^{121}. Additionally, their security lower bound holds for adversaries observing the exact values on wires. This might be a bit far from reality, where we are typically limited to around 100 million noisy traces.

As a result, there is a need for second-order masked circuits which do not require fresh randomness while achieving small area and low latency. To accomplish this, we need security models which are closer to practice and better masking techniques.

Contributions. In this work, we use a generalization of the bounded-query probing model from Beyne *et al.* [BDZ20] called the *noisy probing model* which bears some similarity to the model by Dziembowski *et al.* [DFH+16]. In this model, it is possible to restrict the power of adversaries by modeling probing results as a noisy leakage function of some underlying wire-values. An important feature of the model is that, like the model of Beyne *et al.*, it leads to a concrete security analysis framework based on linear cryptanalysis. This is enabled by Theorem 1, which strictly generalizes [BDZ20, Theorem 1] to the noisy probing model. Theorem 1 anticipates the design of maskings that achieve higher-order probing security, including third-order and higher, without fresh randomness. Specifically, under the independent leakage assumption, the result implies a trade-off between the weakening linear-cryptanalytic properties of the masking and the increasing noise level as the security order increases. From a practical viewpoint, the model allows us to obtain more realistic and improved security bounds for the concrete second-order secure sharings that we propose.

We propose two masking techniques that allow for a concrete security analysis in the noisy probing model and that allow the randomness in each masked S-box to be reused. These techniques remove an important limitation from the work of Beyne *et al.* that required each shared function to be second-order non-complete and uniform, which led to a high overhead in area and latency. The first technique ensures that the masking is still non-complete after two register stages which allows for the use of $d + 1$ sharings, meaning the use of a minimal number of shares. However, it requires two register stages per quadratic function and this harms the latency of the designs. As such, we introduce a second technique where two masked S-boxes are paired together. Namely, one masked S-box is fed with the inputs of another in order to temporarily introduce randomness in its design. This additional randomness is used to ensure second-order non-completeness of the design using fewer register stages.

By combining the improved security model with the new masking techniques, we provide second-order secure maskings of LED, Midori, Skinny, and Prince which require no fresh randomness. Instead, the implementation receives some random bits at the start of the encryption and remains unchanged during the execution. Their performance figures

are listed in Table 1 on page 18. We should highlight that we verified the second-order security of our masked S-boxes using SILVER [KSM20] under the glitch-extended probing model and of our implementations by FPGA-based practical experiments. The different case studies show that our techniques are applicable to a wide range of symmetric-key primitives. Our hardware implementations (HDL code), are provided in full in GitHub .

2 Preliminaries

This section recalls a number of standard concepts related to Boolean masking (Section 2.1), as well as a number of useful tools in probability theory (Section 2.2), and information on $d+1$ sharings (Section 2.3).

2.1 Boolean Masking and Threshold Implementations

Boolean masking is a technique based on splitting each secret variable x F_2 in the circuit into shares $\bar{x} = (x_1, x_2, \ldots, x_{s_x})$ such that $x = \sum_{i=1}^{s_x} x_i$ over F_2. A random Boolean masking of a fixed secret is uniform if all sharings of that secret are equally likely.

In this work, we make use of threshold implementations as proposed by Nikova *et al.* [NRR06]. This approach has been extended to capture higher-order univariate attacks by Bilgin *et al.* [BGN+14]. In the following, the main properties of threshold implementations are reviewed.

Let $\bar{F}$ be a layer in the threshold implementation corresponding to a part of the circuit $F : \mathsf{F}_2^n \to \mathsf{F}_2^m$. The function $\bar{F} : \mathsf{F}_2^{ns_x} \to \mathsf{F}_2^{ms_y}$, where we assume s_x shares per input bit and s_y shares per output bit, will be called a *sharing* of F. The i^{th} share of the function $\bar{F}$ is denoted by $F_i : \mathsf{F}_2^{ns_x} \to \mathsf{F}_2^m$, for i $\{1, .., s_y\}$. Sharings can have a number of properties that are relevant in the security argument for a threshold implementation; these properties are summarized in Definition 1.

Definition 1 (Properties of sharings [NRR06, BGN+14])**.** Let $F : \mathsf{F}_2^n \to \mathsf{F}_2^m$ be a function and $\bar{F} : \mathsf{F}_2^{ns_x} \to \mathsf{F}_2^{ms_y}$ be a sharing of F. The sharing $\bar{F}$ is said to be

1. *correct* if $\sum_{i=1}^{s_{F(x)}} F_i(\bar{x}) = F(\sum_{i=1}^{s_x} x_i)$ for all $x_1, \ldots, x_{s_x}$ F_2^n,
2. d^{th}*-order non-complete* if any function in d or fewer shares F_i depends on at most $s_x - 1$ input shares,
3. *uniform* if $\bar{F}$ maps a uniform random sharing of any x F_2^n to a uniform random sharing of $F(x)$ F_2^m.

2.2 Probability Theory and Fourier Analysis

Throughout the paper, random variables are denoted in boldface. The probability space will always be clear from the context. The average of a random variable $\mathbf{x}$ is denoted by $\mathsf{E}\mathbf{x}$. The probability mass or density function of $\mathbf{x}$ will be denoted by $p_{\mathbf{x}}$. In the proof of Theorem 1, we will use the Kullback-Leibler divergence from a random variable $\mathbf{x}$ to a random variable $\mathbf{y}$ on the same probability space. This measure of dissimilarity is defined as the average logarithmic likelihood ratio:

$$D_{\mathrm{KL}}(p_{\mathbf{y}} \,||\, p_{\mathbf{x}}) = \mathsf{E}_{\mathbf{y}} \log(p_{\mathbf{y}}(\mathbf{y})/p_{\mathbf{x}}(\mathbf{y}))\,,$$

where log is the natural logarithm and the average is with respect to $\mathbf{y}$.

Most of the probability distributions in this paper are discrete distributions on F_2^n. In the analysis of such distributions, it is often convenient to work with the Fourier transformations of probability mass functions. As will be discussed in Section 3.4, this

is closely related to the well-known technique of linear cryptanalysis [TG91, Mat93]. In general, the Fourier transformation of a function $\mathsf{F}_2^n \to \mathsf{C}$ can be defined as in Definition 2 below.

Definition 2. Let $f : \mathsf{F}_2^n \to \mathsf{C}$ a complex-valued function on F_2^n. The Fourier transformation of f is a function $\widehat{f} : \mathsf{F}_2^n \to \mathsf{C}$ defined by

$$\widehat{f}(u) = \sum_{x \ \mathsf{F}_2^n} (-1)^{u \ x} f(x) .$$

Equivalently, $\widehat{f}$ is the representation of f in the basis of functions $x \to (-1)^{u \ x}$ for $u \quad \mathsf{F}_2^n$.

The Euclidean norm of a function $f : \mathsf{F}_2^n \to \mathsf{C}$ will be denoted by $f \ _2$. Since the Fourier transformation is orthogonal up to a factor $2^{n/2}$, it holds that $\widehat{f} \ _2 = 2^{n/2} \ f \ _2$. This result is known as Parseval's theorem.

2.3 Masking with $d + 1$ Shares

It has been shown that the implementation cost of threshold implementations is high, particularly at higher orders due to using a large number of input shares, leading to higher area overhead and requiring a significant amount of fresh randomness when composing the functions [MPL+11, CBR+15]. More precisely, the number of input shares depends on the algebraic degree of the target Boolean function, which potentially scales to higher implementation costs. There have been two independent works trying to make the number of input shares independent of the algebraic degree [RBN+15, GMK16]. They proposed methodologies to use $d+1$ input shares for d^{th}-order security while maintaining glitch resistance in hardware platforms with the same level of security that threshold implementations offer. Due to using a lower number of input shares, these constructions have smaller area overhead and latency. These techniques generally demand fresh randomness to achieve non-completeness in contrast to threshold implementations where fresh masks might be needed to fulfill uniformity. A masked Boolean function following the $d+1$ sharing method is divided into two separate parts by a register layer to avoid the propagation of glitches.

Based on the technique presented by Groß *et al.* [GMK16], a two-share variant of a 2-input AND gate $f(a, b) = x$ can be realized as: (the horizontal line is purely cosmetic)

$$\begin{array}{lll} f_0(a_0, b_0) = a_0 b_0 & \to x_0 & \\ f_1(a_0, b_1, r) = a_0 b_1 + r & \to x_1 & x_0 + x_1 = x_0 \\ \hline f_2(a_1, b_0, r) = a_1 b_0 + r & \to x_2 & x_2 + x_3 = x_1 \\ f_3(a_1, b_1) = a_1 b_1 & \to x_3 & \end{array}, \tag{1}$$

where a_0, a_1, b_0, b_1 are the input shares, r is a single-bit of fresh randomness, and x_0, x_1 are the output shares. The functions f_l are known as *coordinate functions* whose result should be stored in registers. The part that generates output shares by XORing the registers' output is known as the *compression layer*. To achieve first-order $d+1$ hardware implementations without fresh randomness, a methodology has been introduced by Shahmirzadi and Moradi [SM21a] which they also extended to second-order designs [SM21b].

3 A Noisy Probing Model

In this section, an extension of the bounded-query probing model from [BDZ20] will be introduced. In the modified model, the adversary can probe the circuit but it obtains noisy rather than exact results. This is a more realistic model and thus allows for a tighter security analysis.

The noisy probing model resembles the *noisy leakage model* first introduced by Chari *et al.* [CJRR99] and extended by Prouff and Rivain [PR13]. The main difference between the two models is in the information given to the adversary. In the noisy leakage model, the adversary is given a noisy function of all wire values in the circuit. In our model, as in the (glitch-extended) probing model, an adversary can only probe the circuit locally. However, unlike in the probing model, the adversaries' probes reveal only a noisy leakage function of the wire values. That makes the model similar to the noisy probing model from Dziembowski *et al.* [DFH+16]. However, the models differ in the way noisy leakage functions are defined. In addition, as opposed to the model of Dziembowski *et al.*, our model is purely information-theoretic, non-asymptotic, and limits the number of queries that can be made by the adversary. Moreover, in this work, we apply the noisy probing model to masking applications instead of re-keying applications. Nevertheless, we believe that reusing the term is justified.

3.1 Security Model

We first introduce the bounded-query probing model from Beyne *et al.* [BDZ20]. In this model, the security of a circuit C with input k against a t-threshold-probing adversary is quantified by means of a left-or-right security game as follows. The challenger picks a random bit b and provides an oracle $\mathcal{O}^b$, to which adversary $\mathcal{A}$ is given query access. The adversary queries the oracle by choosing up to t wires to probe, we denote this set by $\mathcal{P}$, and sending it to the oracle along with the inputs k_0 and k_1. Note that we consider the input of the circuit to consist of both the plaintext and the key. The oracle responds by giving back the probed wire values of $C(k_b)$. After a total of q queries, the adversary responds to the challenger with a guess for b. For $b \quad \{0,1\}$, denote the result of the adversary after interacting with the oracle $\mathcal{O}^b$ using q queries by $\mathcal{A}^{\mathcal{O}^b}$. For left-or-right security, the advantage of the adversary $\mathcal{A}$ is then defined as

$$\mathrm{Adv}_{t\text{-thr}}(\mathcal{A}) = \left| \Pr[\mathcal{A}^{\mathcal{O}^0} = 1] - \Pr[\mathcal{A}^{\mathcal{O}^1} = 1] \right|.$$

Since we are working on hardware, we extend the above model to include the effect of glitches. These effects can result in significant leakage that is not accounted for by the standard probing model, see for example the attacks of Mangard *et al.* on several masked AES implementations [MPO05]. Whereas one of the adversary's probes normally results in the value of a single wire, a glitch-extended probe allows obtaining the values of all wires in a bundle. As glitches occur in the logic between two memory gates, they are stopped by registers. In other words, glitches do not propagate through memory gates. As a result, a glitch-extended probe returns all values leading to the probed wire until registers are reached. This extension was originally proposed in the work of Reparaz *et al.* [RBN+15] and later formalized by Faust *et al.* [FGP+18].

In this work, we adapt the above model by changing the oracle. More specifically, we extend the notion of a probe to a *noisy probe*. Instead of giving back the exact values on the wire/or bundle, the noisy probe returns a *noisy leakage function* of the values. The formal definition of noisy leakage functions is given in Section 3.2. The new security model is depicted in Figure 1. The advantage of a noisy t-threshold probing adversary $\mathcal{A}$ will be denoted by $\mathrm{Adv}_{\text{noisy } t\text{-thr}}(\mathcal{A})$.

In practice, the above model relates to an attacker performing a t^{th}-order attack on *traces*. The attacker only has a limited number of traces which relates to a limited number of queries. In the above security model, the adversary can pick two secret values for the masked circuit which resembles a *fixed vs. fixed* t-test. Because the adversary can pick the secret value, we will model this as a public value throughout the work. Instead, the security of the countermeasure is based on the randomness used to mask it.

In Section 5, we provide several second-order maskings of symmetric primitives. For these case studies we provide upper bounds on the advantage of probing adversaries. Using

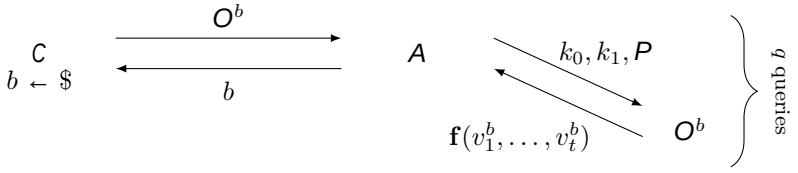


Figure 1: The privacy model for glitch-extended t-threshold-probing security consisting of a challenger C, an adversary A, a left-right oracle O^b, two inputs k_0, k_1, a set of probes P, and a noisy leakage function $\mathbf{f}(v_1^b, \ldots, v_t^b)$ of the probed wire values $v_1^b, \ldots, v_t^b$ in the circuit $C(k_b)$.

the fact that there is noise on the side-channel measurements we can relax that bound and provide much more efficient randomness-free sharings.

3.2 Noisy Leakage Functions

In order to introduce the noisy probing model, it is necessary to introduce the notion of *noisy leakage functions*. Let d and $m_1, \ldots, m_d$ be positive integers. In Definition 3, the Hamming weight of a vector $x = (x_1, \ldots, x_d) \quad \prod_{i=1}^{d} \mathsf{F}_2^{m_i}$ is denoted by $\mathsf{wt}(x) = |\{1 \le i \le d \mid x_i = 0\}|$. Furthermore, the set of vectors of weight i will be denoted by $B_d(i)$, *i.e.* the *Hamming circle* of radius i:

$$B_d(i) = \{x \quad \prod_{i=1}^{d} \mathsf{F}_2^{m_i} \mid \mathsf{wt}(x) = i\}\,.$$

Definition 3 provides a quantitative description of noisy leakage functions that will be useful to obtain our main theoretical result regarding the noisy probing model, *i.e.* Theorem 1. In this definition, $\mathbf{f}$ is a random function over the set of functions from $\prod_{i=1}^{d} \mathsf{F}_2^{m_i}$ to Ω. For example, in the Hamming weight leakage model with additive Gaussian noise $\Omega = \mathsf{R}$. The function $q_{\mathbf{f}}$ can be interpreted as a measure of similarity between the probability density functions $p_{\mathbf{f}(x_1)}$ and $p_{\mathbf{f}(x_2)}$ of the noisy leakage under secrets x_1 and x_2. The reciprocal noise parameters $\lambda_1, \ldots, \lambda_d$ then upper bound the Euclidean norm of the restriction of the Fourier transform of $q_{\mathbf{f}}$ to Hamming circles of successively increasing weights.

Definition 3 (Noisy leakage function)**.** Let $\Omega \quad \mathsf{R}^n$ be a measurable set. A d^{th} order $(\lambda_1, \ldots, \lambda_d)$-noisy leakage function $\mathbf{f}$ is a random function from $\prod_{i=1}^{d} \mathsf{F}_2^{m_i}$ to Ω such that

$$1/\lambda_i \ge \frac{1}{|B_d(i)|} \left(\sum_{u,v \quad B_d(i)} \widehat{q}_{\mathbf{f}}(u,v)^2 \right)^{1/2},$$

for $i = 1, \ldots, d$ where $\widehat{q}_{\mathbf{f}}$ is the Fourier transform of

$$q_{\mathbf{f}}(x_1, x_2) = \int_{\Omega} \frac{p_{\mathbf{f}(x_1)}(y)\, p_{\mathbf{f}(x_2)}(y)}{\mathsf{E}\,|\mathbf{f}^{-1}(y)|}\, \mathrm{d}y\,,$$

with $p_{\mathbf{f}(x)}$ the probability density function of $\mathbf{f}(x)$.

The noise parameters $\lambda_1, \ldots, \lambda_d$ characterize the level of the noise, with λ_i in particular representing the 'noise-level' when values from i probes are combined. In principle, the noise parameters could be computed empirically from estimates of the probability distributions of the leakage (i.e. trace points) under all possible secrets. A practical evaluation of these parameters is left as future work, and we shall rely on plausible estimates instead.

Relation to Other Leakage Functions In previous work, other definitions of noisy functions have been proposed, in particular in the context of the *noisy leakage model* by Duc *et al.* [DDF14]. There, a statistical distance is used to measure the noise on the random function. Some examples of statistical distances which can be used in this context are found in the work by Prest *et al.* [PGMP19]. In our work, we deviate from using these statistical distances for a more natural fit for Theorem 1.

The above definition for noisy leakage functions can be computed explicitly for concrete leakage models, as we will illustrate for the "Hamming weight plus Gaussian noise" model. Figure 2 shows the value of λ_1 for the Hamming-weight leakage model defined by $\mathbf{f}(x) = \mathsf{wt}(x) + \mathbf{e}$ with $\mathbf{e}$ $N(0, \sigma^2)$ and $\mathsf{wt}(x)$ the bitwise Hamming-weight of x. Unsurprisingly, a larger standard deviation σ results in a larger noise parameter λ_1.

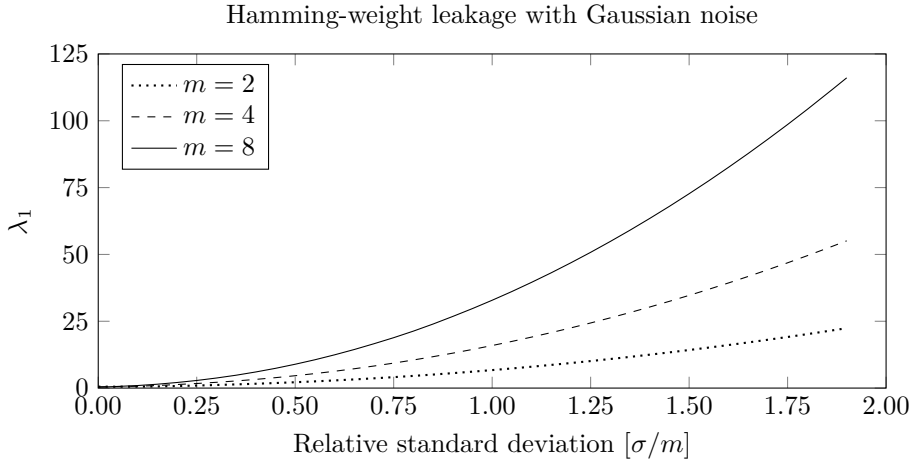


Figure 2: Noise parameter λ_1 as a function of the relative standard-deviation σ/m for the noisy Hamming-weight leakage function defined by $\mathbf{f}(x) = \mathsf{wt}(x) + \mathbf{e}$ with $\mathbf{e}$ $N(0, \sigma^2)$.

Noise Amplification The real-world value of the probing model relies in large part on the premise that higher-order side-channel attacks require a (geometrically) increasing number of traces to perform. Hence, it must be the case that combining information from different probes increases the noise. The following lemma shows that such an amplification of noise indeed occurs, provided that the leakage functions of the probes are independent. The latter assumption is commonly referred to as the independent-leakage assumption and was first stated by Dziembowski and Pietrzak [DP08].

Lemma 1 (Noise amplification)**.** *Let $\mathbf{f}_1, \ldots, \mathbf{f}_d$ be mutually independent λ-noisy first-order leakage functions with $\mathbf{f}_i$ from $\mathsf{F}_2^{m_i}$ to Ω for $i = 1, \ldots, d$. The random function $\mathbf{g}$ defined by*

$$\mathbf{g}(x_1, \ldots, x_d) = (\mathbf{f}_1(x_1), \ldots, \mathbf{f}_d(x_d)) .$$

is a $(\lambda, \lambda^2, \ldots, \lambda^d)$-noisy leakage function of order d from $\prod_{i=1}^d \mathsf{F}_2^{m_i}$ to Ω^d.

Proof. By the independence of $\mathbf{f}_1, \ldots, \mathbf{f}_d$, it holds that $p_{\mathbf{g}(x_1,\ldots,x_d)} = \prod_{i=1}^d p_{\mathbf{f}_i(x_i)}$. Hence, the function $q_{\mathbf{g}}$ from Definition 3 satisfies $q_{\mathbf{g}} = \prod_{i=1}^d q_{\mathbf{f}_i}$ and $\widehat{q}_{\mathbf{g}} = \prod_{i=1}^d \widehat{q}_{\mathbf{f}_i}$. It follows that for any l $\{1, \ldots, d\}$,

$$\sum_{u,v \; B_d(l)} \widehat{q}_{\mathbf{g}}(u,v)^2 = \sum_{\substack{I \; [d] \\ |I|=l}} \prod_{i \; I} \frac{1}{(2^{m_i} - 1)^2} \sum_{u,v \; \mathsf{F}_2^{m_i} \setminus \{0\}} \widehat{q}_{\mathbf{f}_i}(u,v)^2 .$$

Since $1/\lambda^2 \geq \sum_{u,v \ \mathsf{F}_2^{m_i} \setminus \{0\}} \widehat{q}_{\mathbf{f}_i}(u,v)^2/(2^{m_i}-1)^2$ and $|B_d(l)| = \binom{d}{l}$, it holds that

$$\frac{1}{|B_d(l)|}\left(\sum_{u,v \ \ B_d(l)} \widehat{q}_{\mathbf{g}}(u,v)^2\right)^{1/2} \leq \frac{1}{\binom{d}{l}}\left(\sum_{\substack{I \ [d] \\ |I|=l}} \prod_{i \ I} 1/\lambda^2\right)^{1/2} \leq 1/\lambda^l \,.$$

Hence, $\mathbf{g}$ is $(\lambda, \lambda^2, \ldots, \lambda^d)$-noisy. □

A noisy threshold probing adversary A will be said to have independent λ-noisy probes if its probes jointly yield a leakage function $\mathbf{g}$ of the form described in Lemma 1. The functions $\mathbf{f}_1, \ldots, \mathbf{f}_d$ can then be interpreted as the leakage functions for the individual probes.

3.3 Bound on the Advantage

In order to use the security model from Section 3.1 in practice, it is necessary to be able to bound the advantage of adversaries in terms of some properties of the masking that can be computed or estimated. For the noiseless bounded-query probing model, [BDZ20, Theorem 1] provides such a bound in terms of the linear-cryptanalytic properties of the masking. However, the latter theorem is not applicable to the new noisy probing model from Section 3.1. Hence, in Theorem 1, we provide a generalization of [BDZ20, Theorem 1]. The latter theorem corresponds to the case $\lambda_1 = \ldots = \lambda_t = 1$.

Similar to [BDZ20, Theorem 1], Theorem 1 below assumes that any probed wire value can be labeled as 'good' or 'bad'. The values labeled 'good' jointly reveal nothing about the secret. The 'bad' values may reveal secret information, but the leakage can be bounded in terms of $\lambda_1, \ldots, \lambda_t$ and $\varepsilon_1, \ldots, \varepsilon_t$. The parameters $\lambda_1, \ldots, \lambda_t$ are determined by physical aspects such as the leakage model and noise level. The parameters $\varepsilon_1, \ldots, \varepsilon_t$ are instead determined by the mathematical properties of the masking. Specifically, it will be shown in Section 3.4 how these parameters can be determined using linear cryptanalysis.

Theorem 1. *Let A be a noisy t-threshold-probing adversary for a circuit C. Take $\lambda_1, \ldots, \lambda_t \geq 1$, and $\varepsilon_1, \ldots, \varepsilon_t \leq 1$ as non-negative real numbers. Assume that for every query made by A on the oracle O^b with result $\mathbf{z}$, there exists a partitioning (depending only on the probe positions) of the probed wire values into two random variables $\mathbf{x}$ ('good') and $\mathbf{y}$ ('bad') such that*

1. *The noisy leakage function $\mathbf{f}$ such that $\mathbf{z} = \mathbf{f}(\mathbf{x}, \mathbf{y})$ is $(\lambda_1, \ldots, \lambda_t)$-noisy.*
2. *The conditional probability distribution $p_{\mathbf{y}|\mathbf{x}}$ satisfies $\mathsf{E}_{\mathbf{x}} \ \widehat{p}_{\mathbf{y}|\mathbf{x}} \mathbb{1}_{B_t(d)} \ \frac{2}{2} \leq \varepsilon_d$ for all $d \quad \{1, \ldots, t\}$, with $\mathbb{1}_{B_t(d)}$ the indicator function of the set $B_t(d)$ and $\widehat{p}_{\mathbf{y}|\mathbf{x}} \mathbb{1}_{B_t(d)}$ the restriction of $\widehat{p}_{\mathbf{y}|\mathbf{x}}$ to this set.*
3. *Any t-threshold-probing adversary for the same circuit C and making the same oracle queries as A, but which only receives the 'good' wire values (*i.e. *corresponding to $\mathbf{x}$) for each query, has advantage zero.*

The advantage of A can be upper bounded as

$$\mathrm{Adv}_{\mathsf{noisy}\ t\text{-}\mathsf{thr}}(\mathsf{A}) \leq \max_{1 \leq d \leq t} \sqrt{2q\,\varepsilon_d/\lambda_d}\,,$$

where q is the number of queries to the oracle O^b.

The proof of Theorem 1 relies on the following technical lemma. Informally, for a $(\lambda_1, \ldots, \lambda_d)$-noisy leakage function $\mathbf{f}$, Lemma 2 upper bounds the dissimilarity between $\mathbf{f}(\mathbf{x})$ and $\mathbf{f}(\mathbf{x}\,)$ with $\mathbf{x}$ and $\mathbf{x}\,$ independent random variables and $\mathbf{x}\,$ uniform random. The

dissimilarity is measured using the Kullback-Leibler divergence and the bound is expressed in terms of the noise parameters $\lambda_1, \ldots, \lambda_d$ of $\mathbf{f}$ and the Fourier transformation of the probability mass function of $\mathbf{x}$.

Lemma 2. *Let $\mathbf{x}$ be a random variable on $V = \prod_{i=1}^{d} \mathsf{F}_2^{m_i}$ with probability mass function $p_{\mathbf{x}}$ and $\mathbf{f}$ a $(\lambda_1, \ldots, \lambda_d)$-noisy leakage function from V to Ω. Let $\mathbf{x}$ be random variable uniform random on V and $\mathbf{f}$ a noisy leakage function independent from and identically distributed as $\mathbf{f}$. It holds that*

$$D_{\mathrm{KL}}(p_{\mathbf{f}(\mathbf{x})} \,||\, p_{\mathbf{f}\,(\mathbf{x}\,)}) \leq \sum_{i=1}^{d} \frac{\widehat{p}_{\mathbf{x}} \mathbb{1}_{B_d(i)}{}_2^2}{\lambda_i} \frac{|B_d(i)|}{|V|}\,.$$

Proof. Let $\mathbf{y} = \mathbf{f}(\mathbf{x})$ and $\mathbf{y}\ = \mathbf{f}\,(\mathbf{x}\,)$. The goal is to upper bound the Kullback-Leibler divergence (see Section 2.2)

$$D_{\mathrm{KL}}(p_{\mathbf{y}} \,||\, p_{\mathbf{y}}\,) = \int_{\Omega} p_{\mathbf{y}}(y) \log \frac{p_{\mathbf{y}}\,(y)}{p_{\mathbf{y}}\,(y)}\,\mathrm{d}y\,.$$

By the law of total probability, it holds that

$$p_{\mathbf{y}}\,(y) = \sum_{x\ V} p_{\mathbf{y}\ |\mathbf{x}\ =x}(y)\, p_{\mathbf{x}}\,(x) = \mathsf{E}\,|\mathbf{f}^{-1}(y)|/|V|\,.$$

Indeed, $p_{\mathbf{x}}\,(x) = 1/|V|$ for all $x \quad V$ because $\mathbf{x}$ is uniform random on V. Hence, by Jensen's inequality (since $x \to \log x$ is a concave function),

$$D_{\mathrm{KL}}(p_{\mathbf{y}} \,||\, p_{\mathbf{y}}\,) = \int_{\Omega} p_{\mathbf{y}}(y) \log \left(\frac{|V|\, p_{\mathbf{y}}(y)}{\mathsf{E}\,|\mathbf{f}^{-1}(y)|} \right) \mathrm{d}y \leq \log \int_{\Omega} \frac{|V|\, p_{\mathbf{y}}^2(y)}{\mathsf{E}\,|\mathbf{f}^{-1}(y)|}\,\mathrm{d}y\,.$$

The values $p_{\mathbf{y}}^2(y)$ will now be computed. By the law of total probability, it holds that

$$p_{\mathbf{y}}(y) = \sum_{x\ V} p_{\mathbf{y}|\mathbf{x}=x}(y)\, p_{\mathbf{x}}(x) = \sum_{x\ V} \varrho_y(x)\, p_{\mathbf{x}}(x)\,,$$

with $\varrho_y(x) = p_{\mathbf{f}(x)}(y)$. The Fourier inversion formula $p_{\mathbf{x}}(x) = \sum_{u\ V} (-1)^{u\ x}\, \widehat{p}_{\mathbf{x}}(u)/|V|$ then yields

$$|V|\, p_{\mathbf{y}}(y) = \sum_{u\ V} \widehat{p}_{\mathbf{x}}(u) \sum_{x\ V} (-1)^{u\ x}\, \varrho_y(x) = \mathsf{E}|\mathbf{f}^{-1}(y)| + \sum_{u\ V\setminus\{0\}} \widehat{p}_{\mathbf{x}}(u)\, \widehat{\varrho}_y(u)\,,$$

where the second equality follows from $\widehat{p}_{\mathbf{x}}(0) = |V|$ and $\sum_{x\ V} p_{\mathbf{f}(x)}(y) = \mathsf{E}|\mathbf{f}^{-1}(y)|/|V|$. Hence,

$$\frac{|V|\, p_{\mathbf{y}}^2(y)}{\mathsf{E}|\mathbf{f}^{-1}(y)|} = \frac{\mathsf{E}|\mathbf{f}^{-1}(y)|}{|V|} + \frac{2\sum_{u\ V\setminus\{0\}} \widehat{p}_{\mathbf{x}}(u)\, \widehat{\varrho}_y(u)}{|V|} + \frac{\left(\sum_{u\ V\setminus\{0\}} \widehat{p}_{\mathbf{x}}(u)\, \widehat{\varrho}_y(u)\right)^2}{|V|\,\mathsf{E}\,|\mathbf{f}^{-1}(y)|}\,.$$

Next, we consider the integral of $|V|\, p_{\mathbf{y}}^2(y)/\mathsf{E}|\mathbf{f}^{-1}(y)|$ with respect to y. By linearity of integration, it suffices to consider each of the above three terms separately. For the first term, it holds that

$$\int_{\Omega} \frac{\mathsf{E}|\mathbf{f}^{-1}(y)|}{|V|}\,\mathrm{d}y = \mathsf{E} \int_{\Omega} \frac{|\mathbf{f}^{-1}(y)|}{|V|}\,\mathrm{d}y = 1\,.$$

Indeed, conditioned on $\mathbf{f}$, every $x \quad V$ is mapped to exactly one $y \quad \Omega$ under $\mathbf{f}$. The second term is zero since

$$\int_{\Omega} \sum_{u\ V\setminus\{0\}} \widehat{p}_{\mathbf{x}}(u)\, \widehat{\varrho}_y(u)\,\mathrm{d}y = \sum_{u\ V\setminus\{0\}} \widehat{p}_{\mathbf{x}}(u) \int_{\Omega} \widehat{\varrho}_y(u)\,\mathrm{d}y = 0\,.$$

Indeed, $\int_\Omega \varrho_y \, \mathrm{d}y \equiv \mathbb{1}$ and $\widehat{\mathbb{1}}(u) = 0$ for all $u = 0$. Summing the three terms, it follows that

$$\begin{aligned} D_{\mathrm{KL}}(p_{\mathbf{y}} \mid\mid p_{\mathbf{y}}) &= \log\left(1 + \int_\Omega \frac{\left(\sum_{u \ \ V\setminus\{0\}} \widehat{p}_{\mathbf{x}}(u)\,\widehat{\varrho}_y(u)\right)^2}{|V|\,\mathsf{E}\,|\mathbf{f}^{-1}(y)|}\,\mathrm{d}y\right) \\ &\le \int_\Omega \frac{\left(\sum_{u \ \ V\setminus\{0\}} \widehat{p}_{\mathbf{x}}(u)\,\widehat{\varrho}_y(u)\right)^2}{|V|\,\mathsf{E}\,|\mathbf{f}^{-1}(y)|}\,\mathrm{d}y\,, \end{aligned}$$

where the inequality $\log(1+x) \le x$ was used. By expanding the square, it follows that

$$\begin{aligned} D_{\mathrm{KL}}(p_{\mathbf{y}} \mid\mid p_{\mathbf{y}}) &\le \sum_{u,v \ \ V\setminus\{0\}} \widehat{p}_{\mathbf{x}}(u)\widehat{p}_{\mathbf{x}}(v)\,\frac{1}{2^m}\int_\Omega \frac{\widehat{\varrho}_y(u)\widehat{\varrho}_y(v)}{\mathsf{E}\,|\mathbf{f}^{-1}(y)|}\,\mathrm{d}y \\ &\le \frac{1}{2^m}\sum_{u,v \ \ V\setminus\{0\}} \widehat{p}_{\mathbf{x}}(u)\widehat{p}_{\mathbf{x}}(v)\,\widehat{q}_{\mathbf{f}}(u,v)\,, \end{aligned}$$

where $q_{\mathbf{f}}$ is the function defined in Definition 3:

$$q_{\mathbf{f}}(x_1,x_2) = \int_\Omega \frac{p_{\mathbf{f}(x_1)}(y)\,p_{\mathbf{f}(x_2)}(y)}{\mathsf{E}\,|\mathbf{f}^{-1}(y)|}\,\mathrm{d}y\,.$$

Grouping by weight and applying the Cauchy-Schwarz inequality yields

$$\begin{aligned} D_{\mathrm{KL}}(p_{\mathbf{y}} \mid\mid p_{\mathbf{y}}) &\le \frac{1}{|V|}\sum_{i=1}^{d}\sum_{u,v \ \ B_d(i)} \widehat{p}_{\mathbf{x}}(u)\widehat{p}_{\mathbf{x}}(v)\,\widehat{q}(u,v) \\ &\le \frac{1}{|V|}\sum_{i=1}^{d} \widehat{p}_{\mathbf{x}}\mathbb{1}_{B_d(i)}\ {}_2^2 \left(\sum_{u,v \ \ B_d(i)} \widehat{q}(u,v)^2\right)^{1/2} \\ &\le \frac{1}{|V|}\sum_{i=1}^{d} \widehat{p}_{\mathbf{x}}\mathbb{1}_{B_d(i)}\ {}_2^2\,\frac{|B_d(i)|}{\lambda_i}\,, \end{aligned}$$

by definition of the noise parameters $\lambda_1, \ldots, \lambda_d$ (Definition 3). □

The proof of Theorem 1 can now be stated. The first part of the proof is similar to that of the original theorem for noiseless t-threshold-probing adversaries from [BDZ20] and consists of a standard game-hopping argument. The second part of the argument is based on Lemma 2, which replaces the simpler [BDZ20, Lemma 1].

Proof (of Theorem 1). Consider the following two additional games:

1. Game 't-thr-good' is a modification of the t-threshold probing game in which the oracle O^b replaces the 'bad' values in each query by uniform random values. In this game, A essentially only receives information about the 'good' wire values.

2. In the game 'Δ-bad', the adversary chooses a secret input k and is given access to an oracle with the same noisy t-threshold-probing interface as O^b. This oracle is either a noisy t-threshold-probing oracle for the real circuit with input k, or a modification thereof in which the 'bad' values in each query are replaced by uniform random bits. The goal is to distinguish between these two cases.

We construct an adversary B for the game 'Δ-bad' by running the noisy t-threshold-probing adversary A. Specifically, B picks a uniform random bit b and forwards the corresponding secret k_b chosen by A to its challenger. Adversary B reports the oracle as real if and only if A correctly recovers b. Hence, by the triangle inequality,

$$\mathrm{Adv}_{\mathsf{noisy}\ t\text{-}\mathsf{thr}}(\mathsf{A}) \le \mathrm{Adv}_{t\text{-}\mathsf{thr}\text{-}\mathsf{good}}(\mathsf{A}) + 2\mathrm{Adv}_{\Delta\text{-}\mathsf{bad}}(\mathsf{B}).$$

The factor two in front of $\mathrm{Adv}_{\Delta\text{-bad}}(\mathcal{B})$ is due to our definition of 'advantage', *i.e.* the absolute difference between the winning and failure probabilities of $\mathcal{B}$. It is given that $\mathrm{Adv}_{t\text{-thr-good}}(\mathcal{A}) = 0$, so it suffices to upper bound $\mathrm{Adv}_{\Delta\text{-bad}}(\mathcal{B})$.

Denote the result of the i^{th} query of $\mathcal{B}$ to its oracle by $\mathbf{z}_i$ when $\mathcal{B}$ interacts with the real noisy threshold probing oracle and by $\mathbf{z}_i$ when $\mathcal{B}$ interacts with the (partially) randomized oracle. Let $\delta_{\mathsf{TV}}(\cdot,\cdot)$ denote the total variation distance and $\bigotimes$ the tensor product. The distinguishing advantage of the adversary $\mathcal{B}$ is then upper bounded by

$$\begin{aligned}\mathrm{Adv}_{\Delta\text{-bad}}(\mathcal{B}) &\leq \delta_{\mathsf{TV}}\big(\textstyle\bigotimes_{i=1}^{q} p_{\mathbf{z}_i}, \bigotimes_{i=1}^{q} p_{\mathbf{z}_i}\big)\\ &\leq \sqrt{\frac{1}{2} D_{\mathrm{KL}}\big(\textstyle\bigotimes_{i=1}^{q} p_{\mathbf{z}_i} \quad \bigotimes_{i=1}^{q} p_{\mathbf{z}_i}\big)}\\ &\leq \sqrt{\frac{q}{2} \max_{1\leq i\leq q} D_{\mathrm{KL}}\big(p_{\mathbf{z}_i} \quad p_{\mathbf{z}_i}\big)}, \end{aligned} \tag{2}$$

where the second inequality is due to Pinsker.

Since $\mathcal{B}$ makes exactly the same queries to its oracle as $\mathcal{A}$, the wire values probed in the i^{th} query of $\mathcal{B}$ can also be partitioned into 'good' and 'bad' wire values. Denote these values by $\mathbf{x}_i$ and $\mathbf{y}_i$ respectively when $\mathcal{B}$ is interacting with the real threshold probing oracle, and by $\mathbf{x}_i$ and $\mathbf{y}_i$ when $\mathcal{B}$ interacts with the (partially) randomized oracle.

There exists a $(\lambda_1, \ldots, \lambda_t)$-noisy leakage function $\mathbf{f}_i$ such that the result of the i^{th} oracle query satisfies $\mathbf{z}_i = \mathbf{f}_i(\mathbf{x}_i, \mathbf{y}_i)$ and $\mathbf{z}_i = \mathbf{f}_i\,(\mathbf{x}_i, \mathbf{y}_i)$ with $\mathbf{f}_i$ and $\mathbf{f}_i$ independent and identically distributed. By definition of 'Δ-bad', the random variables $\mathbf{x}_i$ and $\mathbf{x}_i$ have the same probability distribution. Consequently,

$$D_{\mathrm{KL}}\big(p_{\mathbf{z}_i} \quad p_{\mathbf{z}_i}\big) = \mathsf{E}_{\mathbf{t}}\, D_{\mathrm{KL}}\big(p_{\mathbf{z}_i|\mathbf{x}_i=\mathbf{t}} \quad p_{\mathbf{z}_i|\mathbf{x}_i=\mathbf{t}}\big).$$

Up to an inconsequential reordering of bits, the values $(\mathbf{x}_i, \mathbf{y}_i)$ can be considered to be elements of $\prod_{d=1}^{t} \mathsf{F}_2^{m_d}$ with m_d the number of bits reached by probe d and $\sum_{d=1}^{t} m_d = n$ the total number of bits. Hence, it follows from Lemma 2, $\sum_{d=1}^{t} |B_t(d)| < 2^n$ and the definition of ε_d that

$$D_{\mathrm{KL}}\big(p_{\mathbf{z}_i} \quad p_{\mathbf{z}_i}\big) \leq \sum_{d=1}^{t} \mathsf{E}_{\mathbf{x}_i} \frac{\widehat{p}_{\mathbf{y}_i|\mathbf{x}_i} \mathbb{1}_{B_t(d)}}{\lambda_d}^{\,2}_{\,2} \frac{|B_t(d)|}{2^n} \leq \max_{1\leq d\leq t} \varepsilon_d/\lambda_d\,.$$

It then follows from (2) that

$$\mathrm{Adv}_{\Delta\text{-bad}}(\mathcal{B}) \leq \max_{1\leq d\leq t} \sqrt{\frac{q\,\varepsilon_d}{2\lambda_d}}.$$

Hence, we conclude that

$$\mathrm{Adv}_{\text{noisy } t\text{-thr}}(\mathcal{A}) \leq 2\mathrm{Adv}_{\Delta\text{-bad}}(\mathcal{B}) \leq \max_{1\leq d\leq t} \sqrt{2q\,\varepsilon_d/\lambda_d}.$$

□

3.4 Cryptanalysis of Higher-Order Threshold Implementations

The security bound obtained in Theorem 1 depends on the parameters $\varepsilon_1, \ldots, \varepsilon_t$. These values will be determined by performing linear cryptanalysis of the masked cipher. This section provides a brief summary of the main concepts from [BDZ20] that are necessary for this analysis. Since the maskings in this paper target second-order security, we assume $t = 2$.

Linear Masking Schemes For any linear masking scheme, there exists a vector space $\mathsf{V} \subseteq \mathsf{F}_2^\ell$ of valid sharings of zero. More specifically, an F_2-linear secret sharing scheme is an algorithm that maps a secret $x \in \mathsf{F}_2^n$ to a random element of a corresponding coset of the vector space V. Let $\rho : \mathsf{F}_2^n \to \mathsf{F}_2^\ell$ be a map that sends secrets to their corresponding coset representative. For convenience, we denote $\mathsf{V}_a = a + \mathsf{V}$.

Let $\bar{G}$ be a correct sharing of a function $G : \mathsf{F}_2^n \to \mathsf{F}_2^n$ in the sense of Definition 1. Fix any $x \in \mathsf{F}_2^n$ and let $a = \rho(x)$ and $b = \rho(G(x))$. The correctness property implies that $\bar{G}(\mathsf{V}_a) \subseteq \mathsf{V}_b$. It follows that the restriction $F : \mathsf{V}_a \to \mathsf{V}_b$ of $\bar{G}$ defined by $F(x) = \bar{G}(x)$ is a well-defined function.

Linear Cryptanalysis of Masked Ciphers Linear cryptanalysis is closely related to the propagation of the Fourier transformation of a probability distribution under a function $F : \mathsf{V}_a \to \mathsf{V}_b$. This leads to the notion of correlation matrices due to Daemen *et al.* [DGV94]. The action of F on probability distributions can be described by a linear operator. The coordinate representation of this operator with respect to the standard basis $\{\delta_x\}_{x \in V}$ may be called the *transition matrix* of F. Following [Bey18], the correlation matrix of F is then the same operator expressed with respect to the Fourier basis. The (absolute) correlation matrix of a sharing can be defined as follows. Note that it only depends on the spaces V_a and V_b, not on the specific choice of the representatives a and b.

Definition 4 (Correlation matrix)**.** For a subspace $\mathsf{V} \subseteq \mathsf{F}_2^\ell$, let $F : \mathsf{V} \to \mathsf{V}$ be a function. The correlation matrix C^F of F is a real $|\mathsf{V}| \times |\mathsf{V}|$ matrix with coordinates indexed by elements $u, v \in \mathsf{F}_2^n/\mathsf{V}$ and equal to

$$C^F_{v,u} = \frac{1}{|\mathsf{V}|} \sum_{x \in \mathsf{V}} (-1)^{u \cdot x + v \cdot F(x)} .$$

For a function $F : \mathsf{V}_a \to \mathsf{V}_b$, its *absolute* correlation matrix $|C^F|$ is defined by[1] $|C^F_{v,u}| = |C^F_{v,u}|$ for all $u, v \in \mathsf{F}_2^\ell/\mathsf{V}$, where $F'(x) = F(x + a) + b$.

In Definition 4, the vector space $\mathsf{V}^\perp = \{x \in \mathsf{F}_2^\ell \mid \forall v \in \mathsf{V} : v \cdot x = 0\}$ is the orthogonal complement of V. The quotient space $\mathsf{F}_2^\ell/\mathsf{V}$ is by definition the vector space of cosets of V. A vector $x + \mathsf{V} \in \mathsf{F}_2^\ell/\mathsf{V}$ can be concisely denoted by x. For $u \in \mathsf{F}_2^\ell/\mathsf{V}$ and $x \in \mathsf{V}$, the expression $u \cdot x$ is well-defined. Consequently, Definition 4 above is proper.

The relation between Definition 4 and linear cryptanalysis is as follows: the coordinate $|C^F_{v,u}|$ is equal to the absolute correlation of a linear approximation over F with input mask u and output mask v. That is, $|C^F_{v,u}| = |2 \Pr[v \cdot F(\mathbf{x}) = u \cdot \mathbf{x}] - 1|$ for $\mathbf{x}$ uniform random on V_a. An important difference with ordinary linear cryptanalysis is that, for shared functions, the masks u and v correspond to equivalence classes. This formalizes the intuitive observation that masks which differ by a vector orthogonal to the space V lead to identical correlations.

The link between ε_1 and ε_2 and linear cryptanalysis is completed by Theorem 2 below. It shows that the coordinates of $\widehat{p}_{\mathbf{z}}$ are entries of the correlation matrix of the state-transformation between the specified probe locations. In Theorem 2, the restriction of $x \in \mathsf{V}_a$ to an index set $I = \{i_1, \ldots, i_m\}$ is denoted by $x_I = (x_{i_1}, \ldots, x_{i_m}) \in \mathsf{F}_2^{|I|}$. This definition depends on the specific choice of the representative a, but the result of Theorem 2 does not.

Theorem 2 ([BDZ20], §5.2)**.** *Let $F : \mathsf{V}_a \to \mathsf{V}_b$ be a function with $\mathsf{V} \subseteq \mathsf{F}_2^\ell$ and $I, J \subseteq \{1, \ldots, \ell\}$. For $\mathbf{x}$ uniform random on V_a and $\mathbf{y} = F(\mathbf{x})$, let $\mathbf{z} = (\mathbf{x}_I, \mathbf{y}_J)$. The Fourier transformation of the probability mass function of $\mathbf{z}$ then satisfies $|\widehat{p}_{\mathbf{z}}(u, v)| = |C^F_{\widetilde{v}, \widetilde{u}}|$, where $\widetilde{u}, \widetilde{v} \in \mathsf{F}_2^\ell/\mathsf{V}$ are such that $\widetilde{u}_I = u$, $\widetilde{u}_{[\ell] \setminus I} = 0$, $\widetilde{v}_J = v$ and $\widetilde{v}_{[\ell] \setminus J} = 0$.*

[1] Here, we abuse notation since $C^F_{v,u}$ itself cannot be defined in a way that does not depend on the choice of representatives for u and v.

Theorem 2 relates the linear approximations of F to $\widehat{p}_{\mathbf{z}}(u, v)$ and hence provides a method to upper bound $\widehat{p}_{\mathbf{z}}\mathbb{1}_{B_2(2)}{}_2$ and therefore ε_2 based on linear cryptanalysis. Upper bounding the absolute correlations $|C^F_{\tilde{v},\tilde{u}}|$ is nontrivial in general. However, the piling-up principle [Mat93, TG91] can be used to obtain heuristic estimates.

Finally, note that the maskings discussed in Section 5 do not necessarily satisfy the uniformity property (see Definition 1) in each layer – but if necessary, we extend the adversary's probes to guarantee the uniformity of the probed values. This implies that $\widehat{p}_{\mathbf{z}}\mathbb{1}_{B_2(1)}{}_2 = 0$ and consequently $\varepsilon_1 = 0$.

4 Masking Techniques

Shahmirzadi and Moradi [SM21b] presented second-order sharings of Skinny, Midori, Present, and Prince that require only a few bits of randomness per cycle. We use the theory from Beyne *et al.* [BDZ20] (summarized in Section 3.4) to create second-order sharings which require no fresh randomness. To that end, we propose two techniques to mask S-box layers. The first technique splits each quadratic function into two stages and uses non-completeness and uniformity properties. The second method builds further on the first by pairing masked S-boxes to reduce their latency.

4.1 Technique 1: Non-Completeness over Two Stages

In order to apply the theory of Beyne *et al.*, the masking of the S-box must have a sufficiently small maximum absolute correlation in the sense of Definition 4. Although the work of Shahmirzadi and Moradi gives efficient uniform sharings, their sharings do not have this property. For example, consider their uniform sharing of the AND gate using two bits of randomness (r_0, r_1) [SM21b, Sect. 3.2.1]:

$$
\begin{array}{lll}
f_0(a_0, b_0) = a_0b_0 + b_0 & \to x_0 & \\
f_1(a_0, b_1) = a_0b_1 & \to x_1 & \to x_0 + x_1 + x_2 = x_0 \\
f_2(a_0, b_2) = a_0b_2 + b_2 + r_0 & \to x_2 & \\
\hline
f_3(a_1, b_0) = a_1b_0 + a_1 + r_1 & \to x_3 & \\
f_4(a_1, b_1) = a_1b_1 & \to x_4 & \to x_3 + x_4 + x_5 = x_1 \\
f_5(a_1, b_2) = a_1b_2 + a_1 + b_2 & \to x_5 & \\
\hline
f_6(a_2, b_0) = a_2b_0 + b_0 + r_1 & \to x_6 & \\
f_7(a_2, b_1) = a_2b_1 & \to x_7 & \to x_6 + x_7 + x_8 = x_2 \\
f_8(a_2, b_2) = a_2b_2 + r_0 & \to x_8 &
\end{array}
$$

Consider the first output share after two cycles: $x_0 = a_0b + b_0 + b_2 + r_0$. Given that b is considered a constant, the above sharing only linearly combines its input shares. Instead, we want nonlinear terms, such as a_0b_0, to occur in the output. This property will result in Lemmas 3 to 6 in Section 5 and will be used to guarantee the multivariate security of our maskings. We thus search for uniform sharings of quadratic functions which have this nonlinearity property. In this work, we use the first-order non-complete and uniform maskings by Bilgin *et al.* [BNN+12] as a starting point to create low-randomness second-order maskings.

The original work of Beyne *et al.* requires that each stage in the masked design is second-order non-complete and uniform. This requirement comes at a high price in terms of the number of shares and thus in area cost. Instead, we relax this requirement by making sharings which are uniform only every two stages similar to the sharings by Shahmirzadi and Moradi. Since we will reuse randomness between S-boxes, we additionally require that our sharings still achieve first-order non-completeness after two stages.

We start from the non-complete and uniform sharings of Bilgin *et al.* and divide them into two stages such that each stage is second-order non-complete using the ring re-masking technique of Reparaz *et al.* [RBN+15], but we only refresh the cross terms. We illustrate our approach by giving a uniform masking of an AND-XOR gate, which maps (a, b, c) to $ab + c$. The masking uses random bits $r_0, \ldots, r_5$.

$$
\begin{array}{llll}
f_0(a_0, b_0, c_0) = a_0b_0 + c_0 & \to x_0 & \\
f_1(a_0, b_1) = a_0b_1 + r_5 + r_0 & \to x_1 & \to x_0 + x_1 + x_2 = x_0 \\
f_2(a_1, b_0) = a_1b_0 + r_0 + r_1 & \to x_2 & \\
\hline
f_3(a_1, b_1, c_1) = a_1b_1 + c_1 & \to x_3 & \\
f_4(a_1, b_2) = a_1b_2 + r_1 + r_2 & \to x_4 & \to x_3 + x_4 + x_5 = x_1 \\
f_5(a_2, b_1) = a_2b_1 + r_2 + r_3 & \to x_5 & \\
\hline
f_6(a_2, b_2, c_2) = a_2b_2 + c_2 & \to x_6 & \\
f_7(a_0, b_2) = a_0b_2 + r_3 + r_4 & \to x_7 & \to x_6 + x_7 + x_8 = x_2 \\
f_8(a_2, b_0) = a_2b_0 + r_4 + r_5 & \to x_8 &
\end{array}
$$

Each arrow denotes a register stage. Note that the first output share equals $a_0b_0 + a_0b_1 + a_1b_0 + c_0 + r_1 + r_5$ which is still non-complete, but now also contains nonlinear terms.

In order to ensure the second-order non-completeness of the second stage of the sharing, we used fresh randomness in a ring refreshing configuration. Using the theory from Section 3, it will be shown in Section 5 that this randomness can be reused in every S-box. This can be seen as follows. Probing two S-boxes with jointly uniformly shared inputs as in Figure 3 only gives a non-complete set of input shares due to the masking's properties even when removing the randomness r from the construction.

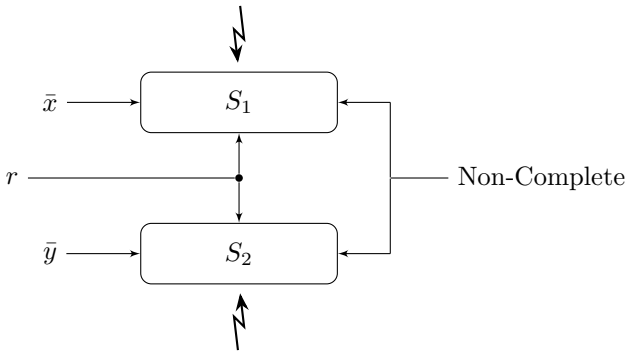


Figure 3: Two masked S-boxes sharing the same randomness. Probes are indicated by lightning signs. Since both S-boxes are non-complete without the randomness r, the probes do not return secret information.

4.2 Technique 2: Paired Masked S-boxes

The technique from Section 4.1 allows us to reduce the number of shares and still create randomness-free second-order sharings. Indeed, in Section 5 we will apply the above technique to design low-area, low-randomness sharings of LED, Midori-64, Skinny, and Prince. However, each quadratic function requires two register stages and this increases the latency of the masking.

Removing the second register layer breaks the non-completeness property over two stages that was outlined above, causing a failure to achieve second-order probing security. Adding randomness to the design is not an option as we intend to reuse all randomness

in each S-box. Instead, we pair two S-boxes that use each other's inputs to help achieve second-order probing security. We also use different randomness for each S-box in the pair, but this randomness can be re-used for each pair. The overall configuration is depicted in Figure 4.

Essentially, the idea is that if two probes are placed in the same pair of masked S-boxes, the randomness (r_0, r_1) ensures that the probes do not observe secret information. When probing two different pairs using the same randomness, the inputs from the paired S-boxes act as fresh randomness.

The above trick should remind the reader of the changing of the guards technique due to Daemen [Dae17], as we are using the inputs of one masked S-box in another one. However, here we are not solving the uniformity of the sharing. Instead, we are using the inputs to ensure the non-completeness property of the sharing.

However, we should be careful with the above trick as the second property required to apply the theory by Beyne *et al.* is that the masking needs to have good diffusion. It will be shown in Section 5 that this helps to increase the number of active masked S-boxes. In case the output of the masked S-box depends on its paired second S-box, the diffusion properties of the masked cipher are altered. Instead, we add the inputs of a second S-box such that the dependency disappears after two stages. Below, we provide an example of the masking technique for two paired AND-XOR gates:

$f_0(a_0, b_0, c_0) = a_0b_0 + c_0 + k_0 + l_0$	$\rightarrow x_0$	
$f_1(a_0, b_1) = a_0b_1 + r_5 + r_0 + k_0$	$\rightarrow x_1$	$x_0 + x_1 + x_2 = x_0$
$f_2(a_1, b_0) = a_1b_0 + r_0 + r_1 + l_0$	$\rightarrow x_2$	
$f_3(a_1, b_1, c_1) = a_1b_1 + c_1 + k_0 + l_0$	$\rightarrow x_3$	
$f_4(a_1, b_2) = a_1b_2 + r_1 + r_2 + k_0$	$\rightarrow x_4$	$x_3 + x_4 + x_5 = x_1$
$f_5(a_2, b_1) = a_2b_1 + r_2 + r_3 + l_0$	$\rightarrow x_5$	
$f_6(a_2, b_2, c_2) = a_2b_2 + c_2 + k_0 + l_0$	$\rightarrow x_6$	
$f_7(a_0, b_2) = a_0b_2 + r_3 + r_4 + k_0$	$\rightarrow x_7$	$x_6 + x_7 + x_8 = x_2$
$f_8(a_2, b_0) = a_2b_0 + r_4 + r_5 + l_0$	$\rightarrow x_8$	

$f_0(k_0, l_0, m_0) = k_0l_0 + m_0 + a_0 + b_0$	$\rightarrow y_0$	
$f_1(k_0, l_1) = k_0l_1 + r_{11} + r_6 + a_0$	$\rightarrow y_1$	$y_0 + y_1 + y_2 = y_0$
$f_2(k_1, l_0) = k_1l_0 + r_6 + r_7 + b_0$	$\rightarrow y_2$	
$f_3(k_1, l_1, m_1) = k_1l_1 + m_1 + a_0 + b_0$	$\rightarrow y_3$	
$f_4(k_1, l_2) = k_1l_2 + r_7 + r_8 + a_0$	$\rightarrow y_4$	$y_3 + y_4 + y_5 = y_1$
$f_5(k_2, l_1) = k_2l_1 + r_8 + r_9 + b_0$	$\rightarrow y_5$	
$f_6(k_2, l_2, m_2) = k_2l_2 + m_2 + a_0 + b_0$	$\rightarrow y_6$	
$f_7(k_0, l_2) = k_0l_2 + r_9 + r_{10} + a_0$	$\rightarrow y_7$	$y_6 + y_7 + y_8 = y_2$
$f_8(k_2, l_0) = k_2l_0 + r_{10} + r_{11} + b_0$	$\rightarrow y_8$	

In red, we denote the added randomness which can be re-used for each pair of AND-XOR sharings. In blue, we denote the paired gate's input. The diffusion of the masking remains unaffected, for example, the output $x_0 = a_0b_0 + a_0b_1 + a_1b_0 + c_0 + r_1 + r_5$ does not depend on (k_0, l_0).

In Section 5, we apply the technique requiring two register stages per quadratic function and the paired S-box technique to several case studies. More specifically, we investigate LED in Section 5.1, Midori in Section 5.2, Skinny in Section 5.3, and Prince in Section 5.4.

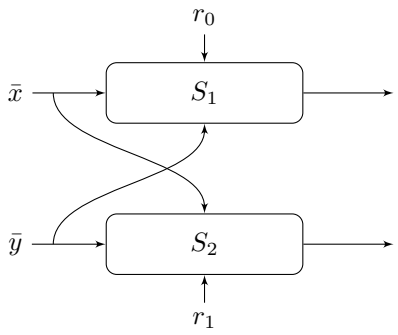


Figure 4: Two paired masked S-boxes.

5 Case Studies

In this section, we apply the two masking techniques from Sections 4.1 and 4.2, to the ciphers LED, Midori, Skinny, and Prince. We use SILVER [KSM20] to study the security of the masked S-boxes and the theory from Section 3 for the security analysis of the entire masked primitive. Based on this analysis, we can conclude that our maskings remain secure even against a noisy-probing adversary that can make up to 100 million (2^{27}) queries. The practical analysis in Section 6 supports this conclusion.

In the rest of the paper, we use the same notation as Bilgin *et al.* [BNN+12], who classified all 4-bit invertible S-boxes and analyzed their maskings. Based on the study, 4-bit quadratic bijections are classified in six classes up to affine equivalence, namely Q_4^4, Q_{12}^4, Q_{293}^4, Q_{294}^4, Q_{299}^4, and Q_{300}^4. This classification was also provided for cubic bijections, denoted by C_i^4.

5.1 LED

LED is a 64-bit block cipher designed by Guo *et al.* [GPPR11]. The cipher's state is divided into 16 four-bit cells. The variant considered here has a 128-bit master key, from which subkeys are derived using a nibble-wise permutation. The cipher consists of 12 steps, each comprising four rounds. These rounds consist of the parallel application of the Present S-box [BKL+07], a `ShiftRows` step and a column-by-column multiplication with an MDS matrix.

Masking. The S-box S is given by the hexadecimal lookup table `c56b90ad3ef84712` and belongs to the cubic class C_{266}^4. There are several ways to decompose the S-box into quadratic bijections, we chose the decomposition $S = A_3 \circ Q_{12}^4 \circ A_2 \circ Q_{12}^4 \circ A_1$ with A_1, A_2 and A_3 affine and Q_{12}^4 (lookup table `0123456789cdefab`) quadratic. The affine maps are defined by

$$A_1 : (a,\ b,\ c,\ d) \to (a,\ d,\ b,\ b+c+1), \qquad (\texttt{894501cdab6723ef})$$
$$A_2 : (a,\ b,\ c,\ d) \to (c+d,\ c,\ a,\ a+b), \qquad (\texttt{0c843fb71d952ea6})$$
$$A_3 : (a,\ b,\ c,\ d) \to (a+d+1,\ b+c,\ a+c+d,\ b+c+d+1)\,. \qquad (\texttt{9c3672d841ebaf05})$$

In our first design, we make a second-order probing-secure masked version of the S-box in such a way that it remains first-order secure without fresh masks. This enables us to reuse the fresh masks in all S-boxes and in all rounds, which will be discussed in more detail in the security analysis paragraph below. To guarantee the security of the design, the outputs of A_1, A_2, and the second application of Q_{12}^4 should be stored in registers. Otherwise, the first-order non-completeness would be violated. Additionally, the masking

of Q_{12}^4 also needs a register layer before compression, leading to a 5-stage design of the masked LED S-box. The full description of the coordinate functions and how they are compressed together is given in Appendix A.

To improve the latency, we can apply our second masking technique (introduced in Section 4.2) and remove two register layers to make a 3-stage design. To this end, we integrate the middle and output affine functions, A_2 and A_3, at the output of the quadratic functions. Namely, $S = G \circ F \circ A_1$ with $F = A_2 \circ Q_{12}^4$ (lookup table `0e843bd71f952ac6`) where

$$F : (a,\ b,\ c,\ d) \to (bd + c + d,\ bd + c,\ a,\ bd + cd + a + b),$$

and $G = A_3 \circ Q_{12}^4$ (lookup table `9c3672d841af05eb`) where

$$G : (a,\ b,\ c,\ d) \to (a + d + 1,\ cd + b + c,\ bd + a + c + d,\ cd + b + c + d + 1).$$

The second-order probing secure maskings of F and G are given in detail in Appendix C. Note that the composition of A_1 and Q_{12}^4 contains a coordinate function containing all quadratic monomials in three input variables. Hence, it does not have a 3-share non-complete and uniform sharing [BNN+12]. For this reason, we implement the input affine map A_1 separately such that the output of A_1 is stored in a register to ensure non-completeness. We then pair two S-boxes, following the second technique described in Section 4.2 and add randomness to the paired design. These random masks can be reused for each pair of S-boxes in the entire encryption.

Architecture. The design architecture of our fully-pipelined round-based second-order LED is depicted in Figure 5. In this design, spanning 5 clock cycles, no S-boxes are paired and each S-box is implemented independently. Note that we do not place any further registers to implement the cipher as one of the register stages can be seen as the state register. As stated before, the fresh masks for one S-box can be re-used for all S-boxes in all rounds. Hence, we generate 24 random bits at the start and store them for the entire encryption, avoiding the need to update the random bits every clock cycle. The structure of the second design is similar to Figure 5, except that two S-boxes are paired together and each round requires two fewer clock cycles, *i.e.* A_2 and A_3 are integrated into the quadratic bijections and the register layers after them are removed. Since we cannot share random bits between the paired S-boxes, we need 72 random bits at the start of each encryption. These bits remain the same throughout the execution of the cipher like in the other design. The synthesis results for our designs are shown in Table 1. Using polynomial masking as the underlying masking scheme, the authors of [CBRN14] presented a second-order secure PRESENT S-box whose randomness complexity and latency are extremely high. Recently, a design with 7 input shares has been proposed by Beyne *et al.* [BDZ20] with a high area overhead. Notably, our designs need fewer fresh masks with lower area and higher throughput. Note that the number of fresh masks reported in Table 1 includes the initial sharing and the key schedule.

Security Analysis. We first assess the security of one round of the maskings. For this, we use the SILVER verification tool [KSM20]. Since the paired S-box is too large for the tool to handle, we have split the verification into two parts. For the first part, we removed the fresh masks from our S-box construction and checked their first-order security in the glitch-extended probing model. Second, we verified the second-order security of one S-box with fresh masks. In both cases, the security was confirmed by SILVER. Due to the first part, placing probes in two different pairs is covered. The second part of the verification together with using fresh randomness in one pair covers the case where the two probes are placed in the same pair. We conclude that all probe positions in a single round of the primitive can be labeled as 'good' when applying Theorem 1.

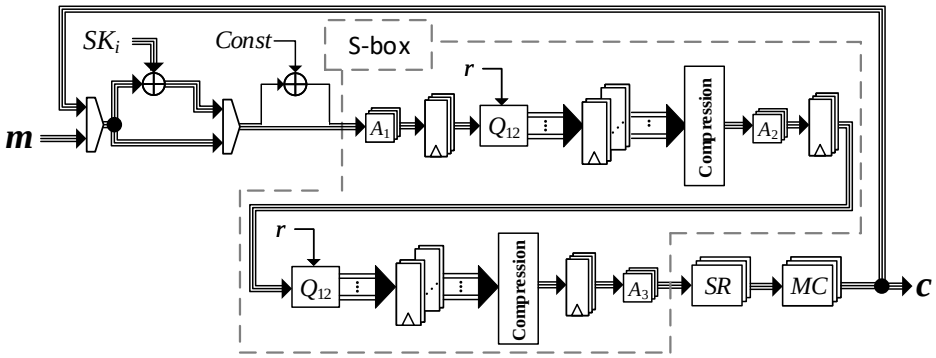


Figure 5: Design architecture of our round-based second-order LED-128 encryption function.

Since the key schedule is an affine function, a probe on the key-addition operation depends on at most one share of the key. As a result, we can consider the shares of the key as 'good' values when applying Theorem 1. The same reasoning applies to the additional randomness that is reused across S-boxes.

Before Theorem 1 can be applied to obtain a bound on the security against noisy 2-threshold probing adversaries, the analysis of the 'bad' wire values must be completed. This is the most difficult part of the security analysis and requires determining ε_1 and

Table 1: Performance figures of different implementations.
(using Synopsis Design Compiler, and UMC 90 standard cell library, excluding RNGs)

Design	Security Order	No. of Shares	Fresh Masks/ Encryption [bit]	Area [kGE]	Delay [ns]	Latency [cycles]	Throughput [MB/s]
LED-128							
[CBRN14][a]	2	5	520	8.3	-	149	-
[BDZ20][b]	2	7	664	28.7	3.82	192	43.6
This work	2	3	408	14.9	2.37	240	70.3
This work	2	3	456	16.3	2.04	144	81.7
Midori-64							
[SM21b]	2	3	2432	15.5	2.86	64	174.8
This work	2	3	408	13.9	2.94	64	170
This work	2	3	456	16.6	2.95	48	169.5
Skinny-64-64							
[SM21b]	2	3	4352	10.6	1.22	128	204.9
This work	2	3	296	12.4	1.33	128	188
This work	2	3	320	12.3	1.33	96	188
Prince							
[SM21b]	2	3	3456	19.4	3.11	84	214.3
[BKN19][c]	2	3	21120	13.4	4.00	72	27.7
[BKN19][c]	2	5	12032	18.7	4.10	72	27.1
[BKN19][c,d]	2	3	41856	32.4	3.42	24	194.9
[BKN19][c,d]	2	8	34496	177.6	3.54	24	188.3
This work	2	3	454	19.5	3.08	72	216.4
This work	2	3	584	20.3	3.41	48	195.5

[a] just an S-box and using NanGate 45
[b] implemented and synthesized by ourselves
[c] using TSMC 90
[d] without S-box decomposition

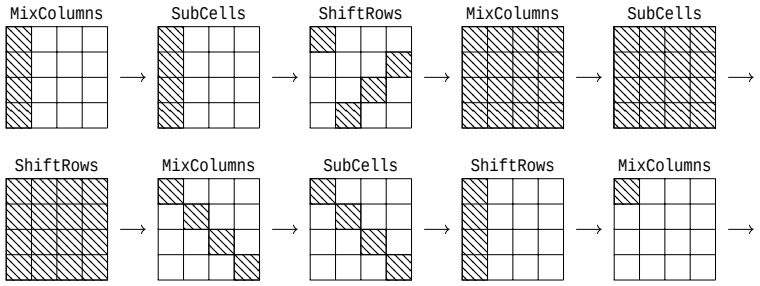


Figure 6: Linear trail with a minimal number of active S-boxes over four rounds of LED, with only one active column in the input- and output masks. The figure shows the nonzero cells in the output masks of each transformation.

ε_2. Due to the first-order security of the masking, $\varepsilon_1 = 0$. The value of ε_2 is determined by the correlation of linear approximations resulting from probes placed in two different rounds. Our estimate of ε_2 is obtained using the principles outlined in Section 3.4 and relies on the piling-up principle. The analysis holds for both maskings of LED.

To upper bound the correlation of linear trails, the following upper bound on the maximum absolute correlation of linear approximations over the masked LED S-box will be used. The verification of this result is done using software that is included as supplementary material.

Lemma 3. *Let $\bar{S} : \mathsf{V}_a \to \mathsf{V}_b$ be any restriction of the sharing of S defined above. Denote its absolute correlation matrix by $|C^{\bar{S}}|$. For any $u, v \quad \mathsf{F}_2^\ell/\mathsf{V}$ such that $u = 0$, it holds that $\left|C^{\bar{S}}_{u,v}\right| \leq 2^{-2}$.*

A single probe on the LED state results in a value that depends either on one column of the state (when placed at the `MixColumns` step) or on the input or output of two (paired) S-boxes in the same column. Hence, it suffices to consider masks (in the sense of linear cryptanalysis) that activate at most one column of the input and output state of the trail. As a result, we search for the activity pattern between two columns of the LED state activating the least number of S-boxes. This analysis was already made for the LED masking of Beyne *et al.* [BDZ20]. There, it was shown that all such two- and three-round trails have correlation zero. For more than three rounds, the best linear trails span four rounds and activate 24 S-boxes. This follows from the design of LED, which follows the wide-trail strategy [DR01]. An example of the activity pattern of such an optimal trail between two columns is shown in Figure 6.

Thus, for probes placed in rounds i and $i + r$ with $r \geq 3$, the relevant linear trails all have at least 24 active S-boxes. Hence, using Lemma 3, the correlations of these trails are bounded by 2^{-48}. By the piling-up principle, we expect a similar bound for the correlation of linear approximations. It then follows that the 2-norm of the nontrivial Fourier coefficients of the observed bits $\mathbf{z}$ can be upper bounded by

$$\varepsilon_2 := \left\| \widehat{p}_{\mathbf{z}} \mathbb{1}_{B_2(2)} \right\|_2^2 \leq |\operatorname{supp} \widehat{p}_{\mathbf{z}}| \left\| \widehat{p}_{\mathbf{z}} \mathbb{1}_{B_2(2)} \right\|_\infty^2 \leq 2^{64} 2^{-96} = 2^{-32},$$

where we have used the inequality $|\operatorname{supp} \widehat{p}_{\mathbf{z}}| \leq 2^{64}$, which follows from the fact that the observed value $\mathbf{z}$ consists of at most 64 bits in the glitch-extended probing model: if a coordinate in $\bar{S}$ is read, at most 12 shares are learned; if an output of the shared linear layer is probed, at most 32 shares are observed.

The above analysis motivates the following security claim, which relies only on the accuracy of the piling-up principle (that resulted in the estimate $\varepsilon_2 \leq 2^{-32}$).

Security Claim 1. *Let A be a noisy 2-threshold probing adversary for the masking of LED described in this section. If A makes at most q queries and the probes of A are*

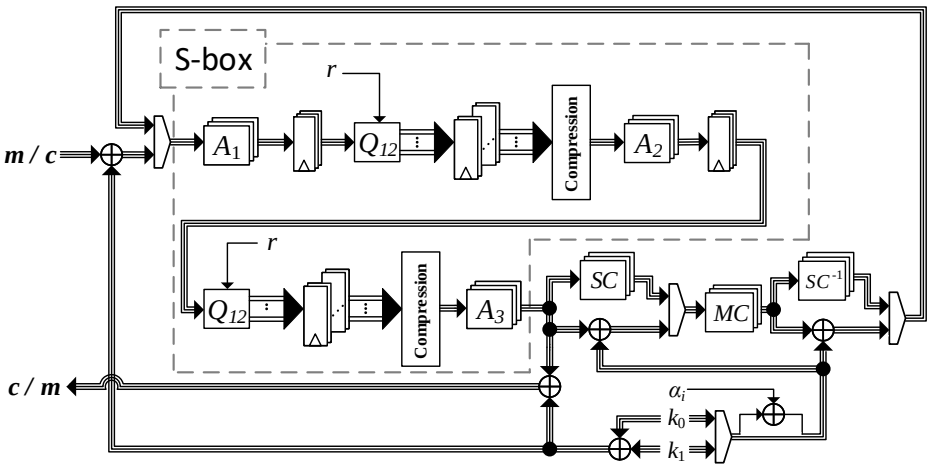


Figure 7: Design architecture of our round-based second-order MIDORI-64 encryption/decryption function.

independent and λ-noisy, then the advantage of A is bounded by (assuming piling-up)

$$\mathrm{Adv}_{\text{noisy 2-thr}}(\mathsf{A}) \leq \sqrt{\frac{q}{\lambda^2\, 2^{31}}}\,.$$

Due to the above security bound, the advantage of any noisy-probing adversary making at most 2^{27} queries is at most 2^{-8} as long as $\lambda \geq 2^6$. Based on Figure 12d from the practical evaluation on FPGA, we observe third-order univariate leakage of the masked PRINCE with 100 million traces. Assuming similar leakage for the masked LED, we can assume that there is a third-order distinguisher with advantage one. As a result, we have that $1 \leq \sqrt{2q/\lambda^3}$ for $q = 2^{27}$ and thus $\lambda \leq 2^9$. Note that we expect much higher noise parameters in ASIC implementations.

5.2 Midori

In this section, we consider the MIDORI-64 block cipher by Banik *et al.* [BBI+15]. It has a 64-bit state that is divided into 4-bit cells and a 128-bit key. Round keys are derived by alternately using the left or right half of the master key. MIDORI-64 uses a 4-bit cubic S-box which can be decomposed into two quadratic functions. The diffusion layer consists of a permutation of the 4-bit cells and the application of an involutive binary quasi-MDS matrix. We note that MIDORI-64 has been broken by Todo *et al.* [TLS16]. We choose this cipher to provide more variety in the applications of our masking techniques, but we do not recommend the use of the cipher.

Masking. MIDORI's 4-bit S-box S is given by the lookup table `cad3ebf789150246` and is affine equivalent to the class C^4_{266}. We use the same S-box decomposition as presented by Moradi *et al.* [MS16]. Namely, we can decompose the S-box as $S = A_3 \circ Q^4_{12} \circ A_2 \circ Q^4_{12} \circ A_1$ with

$$\begin{aligned}
A_1 &: (a,\ b,\ c,\ d) \to (b,\ a,\ d,\ a+c), && (\texttt{0a1b82934e5fc6d7})\\
A_2 &: (a,\ b,\ c,\ d) \to (b+d,\ b,\ a,\ a+c+1), && (\texttt{84b70c3f95a61d2e})\\
A_3 &: (a,\ b,\ c,\ d) \to (c,\ a,\ c+d,\ b+1). && (\texttt{8a02df57ce469b13})
\end{aligned}$$

The general structure of our 4-stage masked S-box can be seen in Figure 7. Notably, the output of A_1 and A_2 should be stored in registers. The masking of Q^4_{12} also needs a

register layer before compression. Their sharing is given in Appendix A. This would lead to a design with a latency of 4 clock cycles.

To reduce the latency of our design by removing the register layer after A_2, we follow the technique described in Section 4.2 and pair two masked S-boxes. More precisely, similar to LED, we integrate A_2 (respectively A_3) with Q_{12}^4 and define $S = G \circ F \circ A_1$ with $F = A_2 \circ Q_{12}^4$ (lookup table `84b70c3f951d2ea6`) as

$$F : (a,\ b,\ c,\ d) \to (bd + cd + c + d,\ bd + cd + b,\ a,\ bd + a + c + 1),$$

and $G = A_3 \circ Q_{12}^4$ (lookup table `9c3672d841af05eb`) as

$$G : (a,\ b,\ c,\ d) \to (bd + c,\ a,\ bd + c + d,\ bd + cd + b + 1).$$

If we would integrate the affine map A_1 into the first quadratic bijection Q_{12}^4, then we would face the same difficulty as observed for LED. That is, all quadratic monomials of three input variables would appear in one coordinate function. Hence, we keep the affine map A_1 in the decomposition and store its output in a register. Since we removed the register after the compression layer of F, we pair two S-boxes to ensure their second-order probing security. As discussed in Section 4.2, this requires adding independent fresh masks to the two S-boxes in the pair. These fresh masks can be reused for all S-box pairs in the cipher. The maskings of F and G are given in Appendix E.

Architecture. The design architecture of our round-based second-order MIDORI-64 implementation, which supports both encryption and decryption, is illustrated in Figure 7. Note that in this design, the affine functions are implemented separately and the S-boxes are not paired. In this design, 24-bit fresh masks are generated at the start of encryption and remain unchanged during the execution. These fresh masks are used in all S-boxes. The same holds for the low latency design. However, the number of fresh masks is higher, *i.e.* the design should receive 72-bit fresh randomness along with the shared input and key. The synthesis results for this design can also found in Table 1. As a comparison to the state of the art, a second-order secure MIDORI-64 is presented in Shahmirzadi and Moradi [SM21b]. It requires more randomness than our designs. Furthermore, our designs have roughly the same delay and area overhead.

Security Analysis. We first investigate the probing security of our S-box designs and of one round of the masked primitive. This verification is performed by the SILVER tool [KSM20] and was done in the same way as for LED. From the verification, we can conclude the second-order probing security of one round of the masked LED. Hence, we conclude that all probe positions in a single round of the primitive can be labeled as 'good' following Theorem 1. Similarly, since the key schedule is an affine function, the same labeling applies for it.

As for LED, the perfect first-order security of the masking implies $\varepsilon_1 = 0$. To estimate ε_2, a similar argument as in the case of LED will be used. However, there is one important novelty in the analysis of MIDORI below: rather than assuming that the entire key is constant, which was sufficient for the analysis of LED, we rely on the fact that the adversary can observe only a few key bits. This significantly reduces the correlation of the best trail.

We first compute the maximum absolute correlation of the masked MIDORI S-box. The following lemma provides an upper bound, which can be verified using the software in the supplementary material.

Lemma 4. *Let $\bar{S} : \mathsf{V}_a \to \mathsf{V}_b$ be any restriction of the sharing of the MIDORI S-box S defined above. Denote its absolute correlation matrix by $|C^{\bar{S}}|$. For any $u, v \quad \mathsf{F}_2^\ell/\mathsf{V}$ such that $u = 0$, it holds that $\left|C^{\bar{S}}_{u,v}\right| \leq 2^{-2}$.*

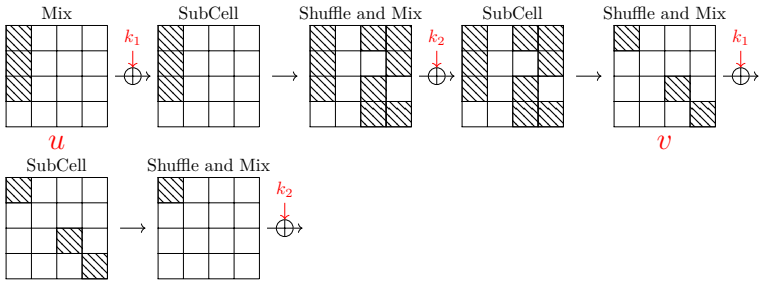


Figure 8: Linear trail with a minimal number of active S-boxes over four rounds of MIDORI, with only one active column in the input- and output masks. If the randomness of the keys k_1 and k_2 is taken into account, then the trail has correlation zero.

As in the analysis of LED, a probe can only activate one column of the MIDORI state. For fixed keys, the best trails between two columns essentially follow the ones given by Banik *et al.* [BBI+15, Fig. 9] and span four rounds activating 15 S-boxes. Figure 8 depicts such a trail. Although the correlation of this trail is quite small ($\leq 2^{-30}$), this is not sufficient to obtain a good bound on ε_2 due to the potentially large size of $\operatorname{supp} \widehat{p}_{\mathbf{z}}$.

However, taking into account the randomness of the key, the correlation of all four and five round trails is necessarily zero. Indeed, each of the adversary's probes reveals bits from at most one cell of the shared round keys k_1 and k_2 (note that these bits are labeled 'good'). To ensure a zero mask on the key k_1, the masks u and v indicated in Figure 8 must satisfy $u_i = v_i$ for all cells $i \quad \{1, \ldots, 16\} \setminus \{j\}$, where cell j is the probed cell. Since only column j of u can be nonzero, the same must be true for v. However, this is impossible by the two-round diffusion of MIDORI, which ensures that at least three columns of v are nonzero. This implies that the correlation of any trail over four rounds is zero. The same reasoning applies to five rounds; the only difference is that two cells of k_1 are known and two columns of u and v can be nonzero.

Banik *et al.* [BBI+15, Tbl. 7] showed that a linear trail over 5 rounds of MIDORI must activate at least 23 S-boxes. Hence, for probes placed in rounds i and $i+r$ with $r \geq 6$, the relevant linear trails all have at least 23 active S-boxes. Note that this is an underestimate, since it does not take into account the influence of the key-addition and the fact that only one column of the input and output state can be active. Based on this, it can be concluded that the absolute correlation of the best trail is at most 2^{-46}. It follows that the 2-norm of the nontrivial Fourier coefficients of the observed bits $\mathbf{z}$ can be upper bounded by

$$\varepsilon_2 := \left\| \widehat{p}_{\mathbf{z}} \mathbb{1}_{B_2(2)} \right\|_2^2 \leq |\operatorname{supp} \widehat{p}_{\mathbf{z}}| \left\| \widehat{p}_{\mathbf{z}} \mathbb{1}_{B_2(2)} \right\|_\infty^2 \leq 2^{48} \, 2^{-92} = 2^{-44},$$

where we have used the inequality $|\operatorname{supp} \widehat{p}_{\mathbf{z}}| \leq 2^{48}$, which follows from the fact that the observed value $\mathbf{z}$ consists of at most 48 bits in the glitch-extended probing model: if a coordinate in $\bar{S}$ is read, at most 12 shares are learned; if an output of the shared linear layer is probed, at most 24 shares are observed.

The above analysis motivates the following security claim, which relies only on the accuracy of the piling-up principle.

Security Claim 2. *Let A be a noisy 2-threshold probing adversary for the masking of MIDORI described in this section. If A makes at most q queries and the probes of A are independent and λ-noisy, then the advantage of A is bounded by (assuming piling-up)*

$$\mathrm{Adv}_{\text{noisy 2-thr}}(\mathsf{A}) \leq \sqrt{\frac{q}{\lambda^2 \, 2^{43}}}.$$

Given that the adversary can make up to 2^{27} queries, even without noise the above

bound yields a maximum advantage of 2^{-8}. The noise can only decrease the advantage of the adversary.

5.3 SKINNY

In this section, we consider the tweakable block cipher SKINNY from Beierle *et al.* [BJK+16] with a 64-bit state and a 64-bit tweakey. The state is divided into 4-bit cells which are processed using the cubic PICCOLO S-box [SIH+11], a `ShiftRows` operation and a lightweight matrix multiplication with the columns of the state. This matrix only has branch number two. The tweakey is processed using affine operations.

Masking. The S-box S of SKINNY is given by the lookup table `c6901a2b385d4e7f` and belongs to the cubic class $\mathcal{C}^4_{223}$. We use the same decomposition as Shahmirzadi and Moradi [SM21b], namely $S = A_3 \circ Q^4_{294} \circ A_2 \circ Q^4_{294} \circ A_1$. All affine functions are bit-permutations up to addition by a constant. The affine functions are defined as

$$A_1 : (a,\ b,\ c,\ d) \to (a+1,\ d+1,\ b+1,\ c+1), \qquad (\texttt{feba7632dc985410})$$
$$A_2 : (a,\ b,\ c,\ d) \to (d,\ c,\ b,\ a), \qquad (\texttt{084c2a6e195d3b7f})$$
$$A_3 : (a,\ b,\ c,\ d) \to (b+1,\ a+1,\ c+1,\ d+1). \qquad (\texttt{fdecb9A875643120})$$

The full description of the coordinate functions and how we realized a second-order secure version of Q^4_{294} is provided in Appendix B. The general structure of the 4-stage masked S-box is similar to MIDORI's S-box. More precisely, we placed register layers at the input and output of the first application of Q^4_{294} as well as before the compression layers. However, we add 16 bits of additional fresh masks to refresh the output of every pair of S-boxes in a column to compensate for the weak diffusion layer of SKINNY. Note that it is also possible to avoid this extra randomness, as the randomness of the masked key can serve as a substitute as in the analysis of MIDORI in Section 5.2. However, recall that the tweak part of the tweakey can be public and does not necessarily need to be masked. As a result, the complete construction uses a total of 40-bits of fresh masks.

We then used our second masking technique to improve the latency by pairing two S-boxes similar to MIDORI-64 and LED. Namely, the first application of Q^4_{294} in the first S-box receives one share from the first application of Q^4_{294} in the second S-box as input and vice versa. We note that, since all affine layers are bit-permutations, they do not need to be integrated with the quadratic functions in the S-box. The sharing of this construction is provided in Appendix D. In this way, we can omit the register layer after the first Q^4_{294}. In summary, we provide a three-share second-order probing-secure realization of the SKINNY S-box with 3 register stages making use of 48-bit fresh masks. Again, an additional 16-bits of fresh randomness should be used in the design to refresh the output of each pair of S-boxes to avoid multivariate leakage. The total of 64-bit fresh randomness can be re-used in every pair of S-boxes.

Architecture. The design architecture of our fully-pipelined round-based second-order SKINNY-64 is depicted in Figure 9. Each round is performed in 4 clock cycles. Note that no further register layer is necessary and one of the register layers in the S-box construction can be seen as the state register. As stated before, 40-bit fresh masks should be given to the design at the start of the encryption. The low latency design needs one fewer clock cycle per round at the cost of additional fresh randomness, *i.e.* 64-bit fresh randomness should be fed to the design. Table 1 shows the corresponding performance figures. Recently, a second-order secure design of SKINNY-64-64 was presented by Shahmirzadi and Moradi [SM21b]. It requires more fresh masks per encryption than our design. Furthermore, our low latency design is 25% faster in the terms of clock cycles. Note that the SKINNY key schedule is a

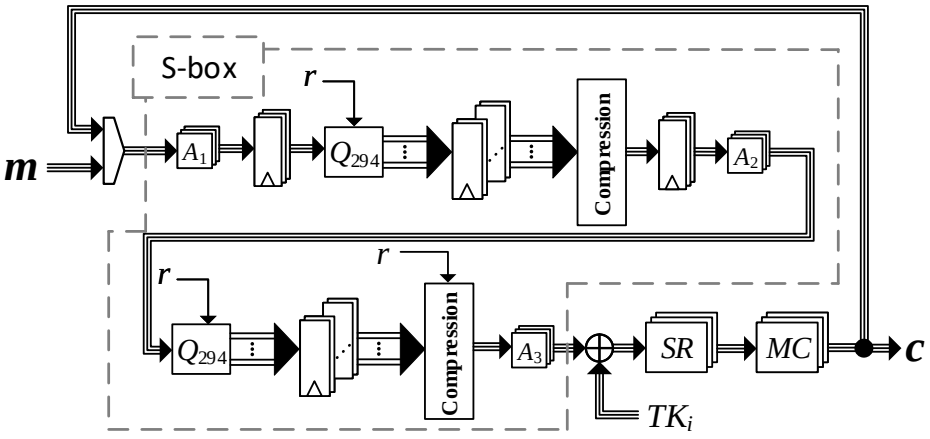


Figure 9: Design architecture of our round-based second-order SKINNY-64 encryption function.

linear function and other variants with larger key sizes can be easily implemented even though we only constructed SKINNY-64-64, *i.e.* with a 64-bit key size.

Security Analysis. We start with the security analysis of one round of the primitive. This has been analyzed by SILVER [KSM20] following the approach outlined in the security analysis of LED. However, for this case we were also able to evaluate the second-order probing security of the paired S-boxes by SILVER due to the fact that the SKINNY S-box is rather simple. Moreover, the number of fresh masks is also lower compared to our MIDORI and PRESENT S-box low latency designs. We can thus label all probe positions placed in a single round of the masked SKINNY as 'good' when applying Theorem 1. Again, since the tweakey schedule is an affine function, we can also consider the shared tweakey variables as 'good'.

We then switch to the multiple-round probing security analysis and consider the 'bad' probe positions for Theorem 1. Since the masking is perfect first-order secure, we have that $\varepsilon_1 = 0$. We now investigate ε_2. We first bound the maximum absolute correlation of the masked SKINNY S-box in Lemma 5. This result can be verified using the software in the supplementary material of the submission.

Lemma 5. *Let $\bar{S} : \mathsf{V}_a \to \mathsf{V}_b$ be any restriction of the sharing of S defined above. Denote its absolute correlation matrix by $|C^{\bar{S}}|$. For any $u, v \quad \mathsf{F}_2^\ell/\mathsf{V} \quad$ such that $u = 0$, it holds that $\left|C^{\bar{S}}_{u,v}\right| \leq 2^{-2}$.*

As mentioned above, we add some additional randomness to avoid trails with nonzero correlation over two or three rounds of SKINNY resulting from two probes. In particular, we add these 16 extra random bits after each S-box pair and use a development version of ArxPy[2], equipped with the SMT solver Boolector [NPB15], to search for the best trail resulting from two probes, taking into account that only one column of the state can be active in the input- and output masks. Remarkably, we find that the best trail found by the software covers 6 rounds and activates at least 34 masked S-boxes. Hence, the correlations of these trails are bounded by 2^{-68}. It then follows that ε_2 can be upper bounded by

$$\varepsilon_2 := \ \widehat{p}_{\mathbf{z}} \mathbb{1}_{B_2(2)} \ {}_2^2 \leq |\mathrm{supp}\, \widehat{p}_{\mathbf{z}}| \ \widehat{p}_{\mathbf{z}} \mathbb{1}_{B_2(2)} \ {}_\infty^2 \leq 2^{24} 2^{-136} = 2^{-112} \,,$$

[2] https://github.com/ranea/ArxPy

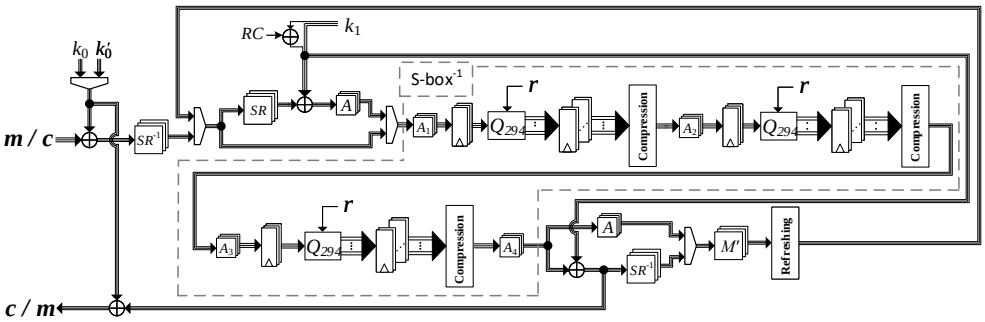


Figure 10: Design architecture of our round-based second-order PRINCE encryption/decryption function.

where we have used the inequality $|\text{supp}\,\widehat{p}_{\mathbf{z}}| \leq 2^{24}$, which follows from the fact that the observed value $\mathbf{z}$ consists of at most 24 bits in the glitch-extended probing model: if a coordinate in $\bar{S}$ is read, at most 12 shares are learned; if an output of the shared linear layer is probed, at most 12 shares are observed.

The above analysis motivates the following security claim, which relies only on the accuracy of the piling-up principle.

Security Claim 3. *Let A be a noisy 2-threshold probing adversary for the masking of SKINNY described in this section. If A makes at most q queries and the probes of A are independent and λ-noisy, then the advantage of A is bounded by (assuming piling-up)*

$$\text{Adv}_{\text{noisy 2-thr}}(\mathsf{A}) \leq \sqrt{\frac{q}{\lambda^2\, 2^{111}}}\,.$$

It is clear that, even without noise, the above bound achieves the desired security level for up to $q \leq 2^{27}$ queries.

5.4 PRINCE

PRINCE is a low-latency and energy-efficient cipher introduced by Borghoff *et al.* [BCG+12]. The cipher's state is divided into 16 four-bit cells and the cipher uses a 128-bit key which is split in two 64-bit subkeys. The nonlinear layer of PRINCE uses a cubic S-box which can be split into three quadratic functions and whose inverse is affine equivalent to itself. The diffusion layer consists of a `ShiftRows` step mixed with an involutive quasi-MDS matrix.

Masking. PRINCE uses both its S-box S, given by `bf32ac916780e5d4` as a lookup table, and its inverse S^{-1}, given by `b732fd89a6405ec1`, during encryption and decryption. The S-box and its inverse are both affine equivalent to the cubic class C^4_{223} [MS16]. Based on the study published by Moradi and Schneider [MS16], we can decompose the inverse S-box as $S^{-1} = A_4 \circ Q^4_{294} \circ A_3 \circ Q^4_{294} \circ A_2 \circ Q^4_{294} \circ A_1$ with

$$\begin{aligned}
A_1 &: (a,\ b,\ c,\ d) \to (b,\ a,\ c,\ a+d+1), && (\texttt{8293c6d70a1b4e5f})\\
A_2 &: (a,\ b,\ c,\ d) \to (d,\ c,\ b+1,\ a+1), && (\texttt{c480e6a2d591f7b3})\\
A_3 &: (a,\ b,\ c,\ d) \to (c,\ c+d,\ b,\ a+b), && (\texttt{08c43bf72ae619d5})\\
A_4 &: (a,\ b,\ c,\ d) \to (a+b,\ a+c+1,\ b+d,\ c). && (\texttt{21748bde6530cf9a})
\end{aligned}$$

We can use the same sharing of Q^4_{294} as in our masking of SKINNY to make a second-order masking of the inverse S-box using 38 bits of fresh masks (see Appendix B). However, we make a slight change to the middle quadratic function: we moved some linear terms

of the direct sharing in order to improve the maximum absolute correlation of the S-box (Lemma 6). The general architecture of our design is shown in Figure 10. Namely, each masking of Q_{294}^4 requires a register layer before compression. Further, a register layer should be placed at the output of A_1, A_2, and A_3. This results in a design with 6 register stages. To avoid leakage at the reflection layer of PRINCE, we add 32 bits of extra fresh randomness after the `MixColumns` operation. More precisely, we fully refresh a column consisting of 4 nibbles and re-use the same masks to refresh the other columns as well. It is interesting to note that PRINCEv2 [BEK+20] does not require this extra randomness due to its key addition in the reflection layer. Additionally, the alternating use of two round keys eases the security analysis of its masking.

Finally, note that the S-box and its inverse are affine equivalent as $S = A \circ S^{-1} \circ A$ which allows us to also implement the S-box proper. The affine layer A is given by

$$A : (a,\ b,\ c,\ d) \rightarrow (a+b+d+1,\ a+1,\ d,\ c+1). \qquad (\texttt{b8a93021edfc6574})$$

To improve the latency of our design, we integrated the affine functions into the quadratic bijection Q_{294}^4 and write the S-box inverse as $S = H \circ G \circ F$ with $F = A_2 \circ Q_{294}^4 \circ A_1$ (lookup table `d850ba32c149e76f`) as

$$F : (a,\ b,\ c,\ d) \rightarrow (a+d+1,\ c,\ ac+cd+a+c+1,\ ad+b+1),$$

$G = A_3 \circ Q_{294}^4$ (lookup table `08c43bf72a6ed591`) as

$$G : (a,\ b,\ c,\ d) \rightarrow (c,\ c+d,\ cd+b,\ ad+b+1),$$

and $H = A_4 \circ Q_{294}^4$ (lookup table `21748bde65039afc`) as

$$H : (a,\ b,\ c,\ d) \rightarrow (bd+cd+a+b,\ bd+a+c+1,\ cd+b+d,\ c).$$

Following the technique described in Section 4.2, we pair four S-boxes in each column. Namely, the F and G functions of the i^{th} S-box with $0 \leq i \leq 3$ are paired with the F function of the $(i+1 \mod 4)^{\text{th}}$ S-box and the G function of the $(i+2 \mod 4)^{\text{th}}$ S-box. The full expression of the coordinate functions and how we paired the functions are given in detail in Appendix F. Based on this approach, the latency of the masked S-box is reduced to 3 clock cycles.

Architecture. Figure 10 depicts the design architecture of our fully-pipelined round-based second-order PRINCE supporting both encryption and decryption. In this design, each round is calculated over 6 clock cycles using a total of 70 bits of fresh masks. Namely, 38 bits for the S-box and 32 bits for refreshing after the `MixColumns`. For the low latency design, *i.e.* the 3-stage construction, we need to place a register layer at the input of the S-box to function as the state register. This design requires more fresh masks. *i.e.* 168 bits for four paired S-boxes and 32 bits for refreshing after the `MixColumns`. Table 1 shows the corresponding performance figures. Compared to state of the art, our designs requires fewer fresh masks per encryption while maintaining similar area and throughput.

Security Analysis. We start by analyzing the security of a single round of the masked PRINCE. This verification was done using SILVER [KSM20]. With SILVER, we examined the first- and second-order security of our designs as outlined in Section 5.1. The verification was successful, meaning that we can label all probe positions in one round of the masking as 'good'. Since the key-schedule is an affine function, it is possible to label all shares of the key as 'good' values when applying Theorem 1. Similarly, all of the random bits used in the S-boxes may be labeled 'good'.

We then move to the 'bad' probe positions in Theorem 1. Since the masking is perfect first-order secure, we have $\varepsilon_1 = 0$. We thus focus on ε_2. We first upper bound the

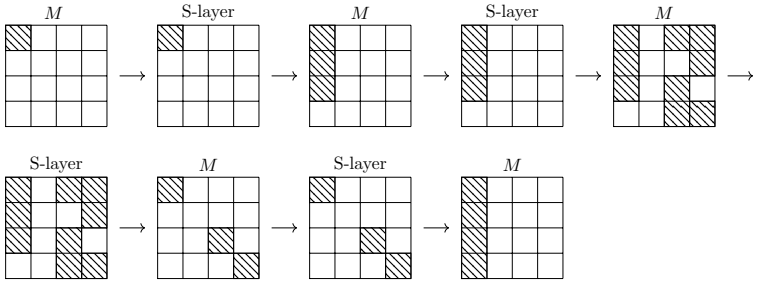


Figure 11: Linear trail over PRINCE starting with one active cell in the first diffusion layer including the key schedule.

maximum absolute correlation of the masked PRINCE S-box. This can be verified using software included in the submission.

Lemma 6. *Let $\bar{S} : \mathsf{V}_a \to \mathsf{V}_b$ be any restriction of the sharing of S defined above. Denote its absolute correlation matrix by $|C^{\bar{S}}|$. For any $u, v \quad \mathsf{F}_2^\ell/\mathsf{V}$ such that $u = 0$, it holds that $\left|C^{\bar{S}}_{u,v}\right| \leq 2^{-1.678}$.*

A probe in the state of PRINCE either results in an active column when probing the diffusion layer or in up to three active cells in a column when probing the masked S-box (due to the pairing technique). Similar to MIDORI, we include the key schedule in our analysis. Using an argument similar to the one used for MIDORI, it can be shown that the randomness of the masked key ensures that a trail with nonzero correlation must cover at least five rounds. In Table 2, we show how many S-boxes are activated given the number of active cells in the first column of the diffusion layer over 2 rounds.

Table 2: The number of active S-boxes, the number of bits observed by a probe, and a bound on c, the square correlation times the support of the observation, over two rounds of the masked PRINCE given how many cells are activated by a probe.

# cells	active S-boxes	support (bits)	c
2	8	17	$2^{-10.84}$
3	12	25	2^{-16}
4	16	33	2^{-21}

Activating only one cell results in the activity pattern with 16 active S-boxes shown in Figure 11. This trail is not possible due to the key. Indeed, this would require that at least two columns of the mask on the state after the third and fourth linear layer are equal. Furthermore, any other activity pattern starting with one activation in M must activate at least five additional S-boxes, making it inferior to the two-cell pattern from Table 2.

As a result, the best trail spans at least five rounds and offers $\varepsilon_2 = 2^{-21.68}$ (the square of the best result of Table 2).

The above analysis motivates the following security claim, which relies only on the accuracy of the piling-up principle.

Security Claim 4. *Let A be a noisy 2-threshold probing adversary for the masking of PRINCE described in this section. If A makes at most q queries and the probes of A are independent and λ-noisy, then the advantage of A is bounded by (assuming piling-up)*

$$\mathrm{Adv}_{\mathsf{noisy\ 2\text{-}thr}}(\mathsf{A}) \leq \sqrt{\frac{q}{\lambda^2\, 2^{20.68}}}\,.$$

In order to achieve an advantage of at most 2^{-8} for up to 2^{27} noisy-probing queries, we require that $\lambda > 2^{11.16}$. A similar argument to the one given in Section 5.1 would not work here, as from the practical evaluation we can conclude that $\lambda < 2^9$. However, we repeat that we expect a much higher noise parameter on ASIC. Moreover, a more detailed security analysis can provide a better bound.

6 Experimental Analysis

In addition to all theoretical analyses, for the sake of completeness, we have conducted experimental analyses. To this end, we have taken our full cipher implementation of PRINCE introduced in Section 5.4, which needs 6 clock cycles per cipher round and in total 70 bits of fresh randomness. We implemented this design on a Xilinx Spartan-6 FPGA of SAKURA-G evaluation board [GIS14] and supplied the device with a stable 6 MHz clock. For each required fresh mask bit, we instantiated a Linear Feedback Shift Register (LFSR) with the feedback polynomial $x^{31} + x^{28} + 1$, which – following the instruction given by De Meyer *et al.* [MMW18] – can be efficiently realized in Xilinx FPGAs by means of only three 6-to-1 Look-Up Tables (LUTs). For each given plaintext to be encrypted, the LFSRs are just activated for one clock cycle to be updated. In other words, the entire 70-bit fresh masks stay unchanged until the encryption is terminated.

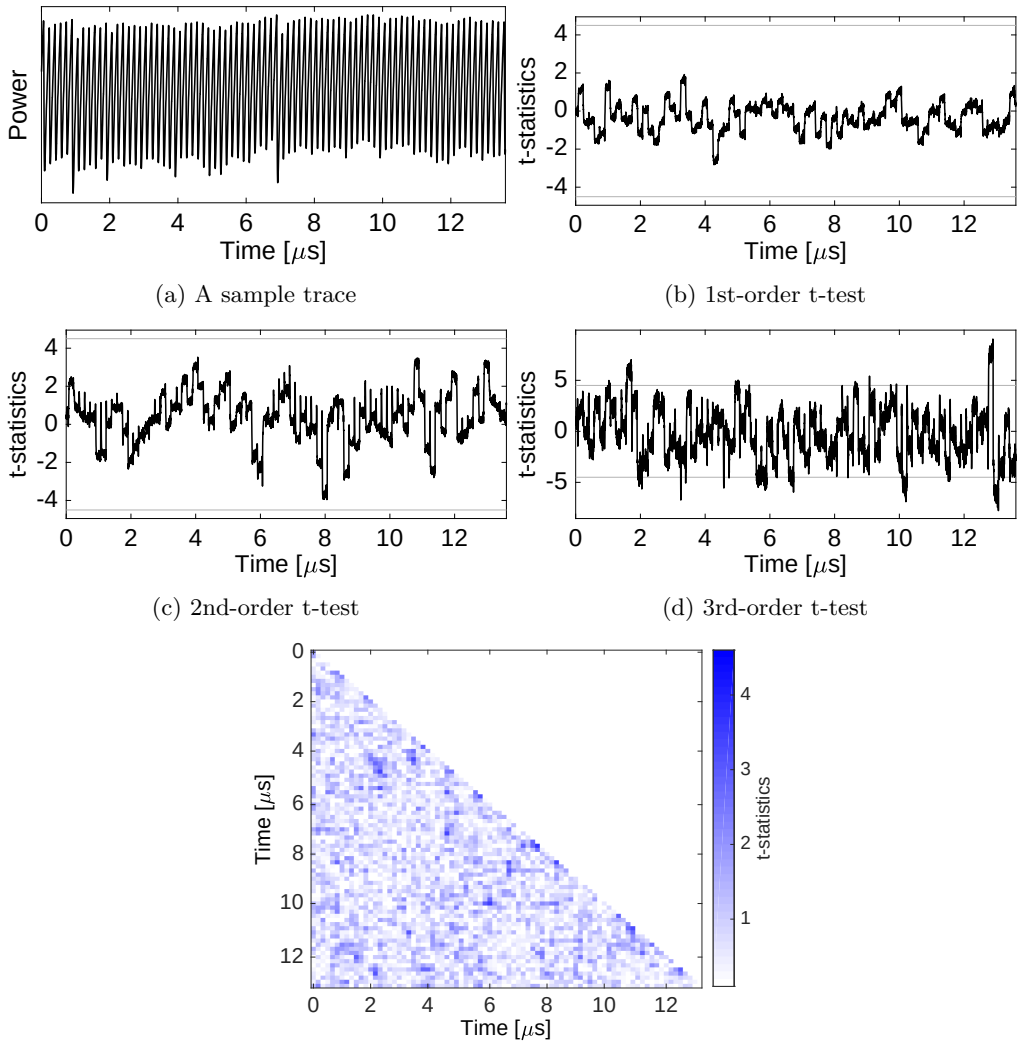
Using a digital oscilloscope at a sampling rate of 500 MS/s, we collected power consumption traces by monitoring the voltage drop over a 1 Ω shunt resistor placed on the VDD path of the target FPGA. We followed the measurement strategy explained by Schneider and Moradi [SM15] to conduct fixed versus random t-test (also known as TVLA [CDG+13]), where the encryption engine receives either a fixed or random plaintext while the key is constant during these measurements. Such an analysis is supposed to detect SCA leakages without performing any key-recovery attack. Note that all inputs to the device (resp. the output of the device) are provided in a masked form using 3 shares.

We performed four different analyses, with the first one being an ordinary t-test on each sample point individually, *i.e.* first-order univariate. For second-order univariate, we first made the traces mean-free (for each group of fixed and random individually), and then squared each mean-free sample point prior to running the same t-test. The same process has been performed for third-order univariate while cubing each mean-free sample point instead of squaring. For a bivariate second-order t-test, an individual t-test for each combination of every two possible sample points should be performed (by multiplying the corresponding mean-free sample points). Since performing such a high number of individual tests is not feasible (particularly in our case that every power consumption signal has 7 000 sample points), we followed the same trick used in [CRB+16, SM21b] by down-sampling the traces by taking a sample point for each clock cycle (carefully selected at the middle of the cycle). This allows us to perform individual t-tests for every possible combination of two clock cycles.

The corresponding results are shown in Figure 12, confirming our theoretical analyses. In short, we do not detect any first-order or second-order leakage, either univariate or bivariate. As expected, the design exhibits third-order leakage, as depicted in Figure 12d. Since the third-order univariate t-test already showed detectable leakage, we omitted the extension of our multivariate analysis to the third-order case.

7 Conclusions

In this work, we used an extension of the bounded-query probing model from Beyne *et al.* [BDZ20], called the *noisy probing model*. This model can be seen as a hybrid between the threshold probing model and the noisy leakage model. We have shown that the



(a) A sample trace

(b) 1st-order t-test

(c) 2nd-order t-test

(d) 3rd-order t-test

(e) 2nd-order bivariate t-test

Figure 12: Experimental analysis of the masked PRINCE from Section 5.4 using 100 million traces.

concrete security of maskings can be analyzed within this model using linear cryptanalysis, extending the results of Beyne *et al.* [BDZ20]. The inclusion of noise makes the model more realistic and allows for relaxed design constraints.

We proposed two techniques to create second-order low-randomness masked designs. The first relaxes the joint need for second-order non-completeness and uniformity. Thanks to this relaxation, we can make low-randomness masked designs with a minimal number of shares. The second technique extends the first one by pairing two masked S-boxes. This technique allows reducing the number of register stages, thus improving the latency of the masked designs.

We applied these two techniques to several case studies, including LED, MIDORI, SKINNY, PRINCE, where the efficiency of the constructed designs can be seen in Table 1. While conducting theoretical security analyses by the noisy probing model, we confirmed our claims by FPGA-based experiments conducted on an exemplary construction of ours.

Acknowledgment. We thank Adrián Ranea for his help on building the SMT model for SKINNY. Tim Beyne and Siemen Dhooghe are supported by a PhD Fellowship from the Research Foundation – Flanders (FWO). The work described in this paper has been supported in part by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972 and through the project 406956718 SuCCESS.

A 3-share $\mathcal{Q}_{12}^4$ with 12-bit Fresh Masks

$F(a,\ b,\ c,\ d) = (x,\ y,\ z,\ t)$ with lookup table `0123456789cdefab`

$x = f(a,b,c,d) = a$
$y = g(a,b,c,d) = bd + cd + b$
$z = h(a,b,c,d) = bd + c$
$t = k(a,b,c,d) = d$

$$
\begin{aligned}
a_1 &\rightarrow x_0\\
a_2 &\rightarrow x_1\\
a_0 &\rightarrow x_2
\end{aligned}
$$

$g_0(d_1,c_1,b_1)$	$= d_1c_1 + d_1b_1$	$\rightarrow y_0$	
$g_1(d_1,c_2,b_2)$	$= d_1c_2 + d_1b_2 + b_2 + r_0 + r_1$	$\rightarrow y_1$	$y_0 + y_1 + y_2 = y_0$
$g_2(d_2,c_1,b_1)$	$= d_2c_1 + d_2b_1 + r_1 + r_2$	$\rightarrow y_2$	
$g_3(d_2,c_2,b_2)$	$= d_2c_2 + d_2b_2$	$\rightarrow y_3$	
$g_4(d_2,c_0,b_0)$	$= d_2c_0 + d_2b_0 + b_0 + r_2 + r_3$	$\rightarrow y_4$	$y_3 + y_4 + y_5 = y_1$
$g_5(d_0,c_2,b_2)$	$= d_0c_2 + d_0b_2 + r_3 + r_4$	$\rightarrow y_5$	
$g_6(d_0,c_0,b_0)$	$= d_0c_0 + d_0b_0$	$\rightarrow x_6$	
$g_7(d_0,c_1,b_1)$	$= d_0c_1 + d_0b_1 + b_1 + r_4 + r_5$	$\rightarrow y_7$	$y_6 + y_7 + y_8 = y_2$
$g_8(d_1,c_0,b_0)$	$= d_1c_0 + d_1b_0 + r_5 + r_0$	$\rightarrow y_8$	

$h_0(d_1,b_1,c_2)$	$= d_1b_1 + c_2$	$\rightarrow z_0$	
$h_1(d_1,b_2)$	$= d_1b_2 + r_6 + r_7$	$\rightarrow z_1$	$z_0 + z_1 + z_2 = z_0$
$h_2(d_2,b_1)$	$= d_2b_1 + r_7 + r_8$	$\rightarrow z_2$	
$h_3(d_2,b20,c_0)$	$= d_2b_2 + c_0$	$\rightarrow z_3$	
$h_4(d_2,b_0)$	$= d_2b_0 + r_8 + r_9$	$\rightarrow z_4$	$z_3 + z_4 + z_5 = z_1$
$h_5(d_0,b_2)$	$= d_0b_2 + r_9 + r_{10}$	$\rightarrow z_5$	
$h_6(d_0,b_0,c_1)$	$= d_0b_0 + c_1$	$\rightarrow z_6$	
$h_7(d_1,b_0)$	$= d_1b_0 + r_{10} + r_{11}$	$\rightarrow z_7$	$z_6 + z_7 + z_8 = z_2$
$h_8(d_0,b_1)$	$= d_0b_1 + r_{11} + r_6$	$\rightarrow z_8$	

$$
\begin{aligned}
d_1 &\rightarrow t_0\\
d_2 &\rightarrow t_1\\
d_0 &\rightarrow t_2
\end{aligned}
$$

B 3-share $\mathcal{Q}^4_{294}$ with 12-bit Fresh Masks

$F(a,\ b,\ c,\ d) = (x,\ y,\ z,\ t)$ with lookup table `0123456789baefdc`

$x = f(a,b,c,d) = bd + a$
$y = g(a,b,c,d) = cd + b$
$z = h(a,b,c,d) = c$
$t = k(a,b,c,d) = d$

$$
\begin{array}{lll}
f_0(b_1,d_1,a_1) = b_1d_1 + a_1 & \to x_0 & \\
f_1(b_1,d_2) = b_1d_2 + r_0 + r_1 & \to x_1 & x_0 + x_1 + x_2 = x_0 \\
f_2(b_2,d_1) = b_2d_1 + r_1 + r_2 & \to x_2 & \\
\hline
f_3(b_2,d_2,a_2) = b_2d_2 + a_2 & \to x_3 & \\
f_4(b_2,d_1) = b_2d_1 + r_2 + r_3 & \to x_4 & x_3 + x_4 + x_5 = x_1 \\
f_5(b_1,d_2) = b_1d_2 + r_3 + r_4 & \to x_5 & \\
\hline
f_6(b_0,d_0,a_0) = b_0d_0 + a_0 & \to x_6 & \\
f_7(b_0,d_1) = b_0d_1 + r_4 + r_5 & \to x_7 & x_6 + x_7 + x_8 = x_2 \\
f_8(b_1,d_0) = b_1d_0 + r_5 + r_0 & \to x_8 &
\end{array}
$$

$$
\begin{array}{lll}
g_0(d_1,c_1,b_2) = d_1c_1 + b_1 & \to y_0 & \\
g_1(d_1,c_2) = d_1c_2 + r_6 + r_7 & \to y_1 & y_0 + y_1 + y_2 = y_0 \\
g_2(d_2,c_1) = d_2c_1 + r_7 + r_8 & \to y_2 & \\
\hline
g_3(d_2,c_2,b_1) = d_2c_2 + b_2 & \to y_3 & \\
g_4(d_2,c_0) = d_2c_0 + r_8 + r_9 & \to y_4 & y_3 + y_4 + y_5 = y_1 \\
g_5(d_0,c_2) = d_0c_2 + r_9 + r_{10} & \to y_5 & \\
\hline
g_6(d_0,c_0,b_0) = d_0c_0 + b_0 & \to y_6 & \\
g_7(d_0,c_1) = d_0c_1 + r_{10} + r_{11} & \to y_7 & y_6 + y_7 + y_8 = y_2 \\
g_8(d_1,c_0) = d_1c_0 + r_{11} + r_6 & \to y_8 &
\end{array}
$$

$$
\begin{array}{ll}
c_1 \to z_0 \\
c_2 \to z_1 \\
c_0 \to z_2
\end{array}
$$

$$
\begin{array}{ll}
d_1 \\
\quad\vdots & \to t_0 \\
c_2 & \\
d_2 \\
\quad\vdots & \to t_1 \\
c_2 & \\
d_0 & \to t_2
\end{array}
$$

C 3-share Present S-Box with 36-bit Fresh Masks

$S = G \circ F \circ A_1$ with lookup table `c56b90ad3ef84712`

$A_1(a,\ b,\ c,\ d) = (a,\ d,\ b,\ b + c + 1)$ with lookup table `894501cdab6723ef`

$F(a,\ b,\ c,\ d) = (x,\ y,\ z,\ t)$ with lookup table `0e843bd71f952ac6`

$x = f(a,b,c,d) = bd + c + d$ $\qquad$ $y = g(a,b,c,d) = bd + c$

$z = h(a,b,c,d) = a$ $\qquad$ $t = u(a,b,c,d) = bd + cd + a + b$

$(k,\ l,\ m,\ n)$ are the input variables of the neighboring S-box's F function.

$f_0(d_1, b_1)$	$= d_1b_1 + k_0 + l_0$	$\rightarrow x_0$	
$f_1(d_1, b_2)$	$= d_1b_2 + d_1 + r_0 + r_1 + k_0$	$\rightarrow x_1$	$x_0 + x_1 + x_2 \rightarrow x_0$
$f_2(d_2, b_1, c_2)$	$= d_2b_1 + c_2 + r_1 + r_2 + l_0$	$\rightarrow x_2$	
$f_3(d_2, b_2)$	$= d_2b_2 + k_0 + l_0$	$\rightarrow x_3$	
$f_4(d_2, b_0)$	$= d_2b_0 + d_2 + r_2 + r_3 + k_0$	$\rightarrow x_4$	$x_3 + x_4 + x_5 \rightarrow x_1$
$f_5(d_0, b_2, c_0)$	$= d_0b_2 + c_0 + r_3 + r_4 + l_0$	$\rightarrow x_5$	
$f_6(d_0, b_0)$	$= d_0b_0 + k_0 + l_0$	$\rightarrow x_6$	
$f_7(d_0, b_1)$	$= d_0b_1 + d_0 + r_4 + r_5 + k_0$	$\rightarrow x_7$	$x_6 + x_7 + x_8 \rightarrow x_2$
$f_8(d_1, b_0, c_1)$	$= d_1b_0 + c_1 + r_5 + r_0 + l_0$	$\rightarrow x_8$	
$g_0(d_1, b_1)$	$= d_1b_1 + m_0 + n_0$	$\rightarrow y_0$	
$g_1(d_1, b_2)$	$= d_1b_2 + r_6 + r_7 + m_0$	$\rightarrow y_1$	$y_0 + y_1 + y_2 \rightarrow y_0$
$g_2(d_2, b_1, c_2)$	$= d_2b_1 + c_2 + r_7 + r_8 + n_0$	$\rightarrow y_2$	
$g_3(d_2, b_2)$	$= d_2b_2 + m_0 + n_0$	$\rightarrow y_3$	
$g_4(d_2, b_0)$	$= d_2b_0 + r_8 + r_9 + m_0$	$\rightarrow y_4$	$y_3 + y_4 + y_5 \rightarrow y_1$
$g_5(d_0, b_2, c_0)$	$= d_0b_2 + c_0 + r_9 + r_{10} + n_0$	$\rightarrow y_5$	
$g_6(d_0, b_0)$	$= d_0b_0 + m_0 + n_0$	$\rightarrow y_6$	
$g_7(d_0, b_1)$	$= d_0b_1 + r_{10} + r_{11} + m_0$	$\rightarrow y_7$	$y_6 + y_7 + y_8 \rightarrow y_2$
$g_8(d_1, b_0, c_1)$	$= d_1b_0 + c_1 + r_{11} + r_6 + n_0$	$\rightarrow y_8$	
		$a_1 + m_1 + n_1 \rightarrow z_0$	
		$a_2 + m_1 \rightarrow z_1$	
		$a_0 + n_1 \rightarrow z_2$	
$u_0(d_1, b_1, c_1)$	$= d_1b_1 + d_1c_1 + k_1 + l_1$	$\rightarrow t_0$	
$u_1(d_1, b_2, c_2)$	$= d_1b_2 + d_1c_2 + b_2 + r_{12} + r_{13} + k_1$	$\rightarrow t_1$	$t_0 + t_1 + t_2 \rightarrow t_0$
$u_2(d_2, b_1, c_1, a_1)$	$= d_2b_1 + d_2c_1 + a_1 + r_{13} + r_{14} + l_1$	$\rightarrow t_2$	
$u_3(d_2, b_2, c_2)$	$= d_2b_2 + d_2c_2 + k_1 + l_1$	$\rightarrow t_3$	
$u_4(d_2, b_0, c_0)$	$= d_2b_0 + d_2c_0 + b_0 + r_{14} + r_{15} + k_1$	$\rightarrow t_4$	$t_3 + t_4 + t_5 \rightarrow t_1$
$u_5(d_0, b_2, c_2, a_2)$	$= d_0b_2 + d_0c_2 + a_2 + r_{15} + r_{16} + l_1$	$\rightarrow t_5$	
$u_6(d_0, b_0, c_0)$	$= d_0b_0 + d_0c_0 + k_1 + l_1$	$\rightarrow t_6$	
$u_7(d_0, b_1, c_1)$	$= d_0b_1 + d_0c_1 + b_1 + r_{16} + r_{17} + k_1$	$\rightarrow t_7$	$t_6 + t_7 + t_8 \rightarrow t_2$
$u_8(d_1, b_0, c_0, a_0)$	$= d_1b_0 + d_1c_0 + a_0 + r_{17} + r_{12} + l_1$	$\rightarrow t_8$	

$G(a,\ b,\ c,\ d) = (x,\ y,\ z,\ t)$ with lookup table `9c3672d841af05eb`
$x = f(a,b,c,d) = a + d + 1$
$y = g(a,b,c,d) = cd + b + c$
$z = h(a,b,c,d) = bd + a + c + d$
$t = k(a,b,c,d) = cd + b + c + d + 1$

$$\begin{aligned} 1 + a_1 + d_1 &\rightarrow x_0 \\ a_2 + d_2 &\rightarrow x_1 \\ a_0 + d_0 &\rightarrow x_2 \end{aligned}$$

$g_0(d_1, c_1) = d_1c_1$	$\rightarrow y_0$		
$g_1(d_1, c_2, b_2) = d_1c_2 + b_2 + c_2 + r_{18} + r_{19}$	$\rightarrow y_1$	$y_0 + y_1 + y_2 = y_0$	
$g_2(d_2, c_1) = d_2c_1 + r_{19} + r_{20}$	$\rightarrow y_2$		
$g_3(d_2, c_2) = d_2c_2$	$\rightarrow y_3$		
$g_4(d_2, c_0, b_0) = d_2c_0 + b_0 + c_0 + r_{20} + r_{21}$	$\rightarrow y_4$	$y_3 + y_4 + y_5 = y_1$	
$g_5(d_0, c_2) = d_0c_2 + r_{21} + r_{22}$	$\rightarrow y_5$		
$g_6(d_0, c_0) = d_0c_0$	$\rightarrow y_6$		
$g_7(d_0, c_1, b_1) = d_0c_1 + b_1 + c_1 + r_{22} + r_{23}$	$\rightarrow y_7$	$y_6 + y_7 + y_8 = y_2$	
$g_8(d_1, c_0) = d_1c_0 + r_{23} + r_{18}$	$\rightarrow y_8$		
$h_0(d_1, b_1) = d_1b_1$	$\rightarrow z_0$		
$h_1(d_1, b_2, a_1) = d_1b_2 + a_1 + d_1 + r_{24} + r_{25}$	$\rightarrow z_1$	$z_0 + z_1 + z_2 = z_0$	
$h_2(d_2, b_1, c_2) = d_2b_1 + c_2 + r_{25} + r_{26}$	$\rightarrow z_2$		
$h_3(d_2, b_2) = d_2b_2$	$\rightarrow z_3$		
$h_4(d_2, b_0, a_2) = d_2b_0 + a_2 + d_2 + r_{26} + r_{27}$	$\rightarrow z_4$	$z_3 + z_4 + z_5 = z_1$	
$h_5(d_0, b_2, c_0) = d_0b_2 + c_0 + r_{27} + r_{28}$	$\rightarrow z_5$		
$h_6(d_0, b_0) = d_0b_0$	$\rightarrow z_6$		
$h_7(d_0, b_1, a_0) = d_0b_1 + a_0 + d_0 + r_{28} + r_{29}$	$\rightarrow z_7$	$z_6 + z_7 + z_8 = z_2$	
$h_8(d_1, b_0, c_1) = d_1b_0 + c_1 + r_{29} + r_{24}$	$\rightarrow z_8$		
$k_0(d_1, c_1) = d_1c_1 + 1$	$\rightarrow t_0$		
$k_1(d_1, c_2, b_2) = d_1c_2 + b_2 + c_2 + d_1 + r_{30} + r_{31}$	$\rightarrow t_1$	$t_0 + t_1 + t_2 = t_0$	
$k_2(d_2, c_1) = d_2c_1 + r_{31} + r_{32}$	$\rightarrow t_2$		
$k_3(d_2, c_2) = d_2c_2$	$\rightarrow t_3$		
$k_4(d_2, c_0, b_0) = d_2c_0 + b_0 + c_0 + d_2 + r_{32} + r_{33}$	$\rightarrow t_4$	$t_3 + t_4 + t_5 = t_1$	
$k_5(d_0, c_2) = d_0c_2 + r_{33} + r_{34}$	$\rightarrow t_5$		
$k_6(d_0, c_0) = d_0c_0$	$\rightarrow t_6$		
$k_7(d_0, c_1, b_1) = d_0c_1 + b_1 + c_1 + d_0 + r_{34} + r_{35}$	$\rightarrow t_7$	$t_6 + t_7 + t_8 = t_2$	
$k_8(d_1, c_0) = d_1c_0 + r_{35} + r_{30}$	$\rightarrow t_8$		

D 3-share $\mathcal{Q}^4_{294}$ with 12-bit Fresh Masks

$F(a,\ b,\ c,\ d) = (x,\ y,\ z,\ t)$ with lookup table `0123456789baefdc`

$x = f(a,b,c,d) = bd + a$

$y = g(a,b,c,d) = cd + b$

$z = h(a,b,c,d) = c$

$t = k(a,b,c,d) = d$

$(k,\ l,\ m,\ n)$ are the output variables of the neighboring S-box's input affine function.

$$
\begin{array}{lll|l}
f_0(b_1,d_1,a_1) = b_1d_1 + a_1 + k_0 + l_0 & \rightarrow x_0 & & \\
f_1(b_1,d_2) = b_1d_2 + r_0 + r_1 + k_0 & \rightarrow x_1 & x_0 + x_1 + x_2 = x_0 & \\
f_2(b_2,d_1) = b_2d_1 + r_1 + r_2 + l_0 & \rightarrow x_2 & & \\
\hline
f_3(b_2,d_2,a_2) = b_2d_2 + a_2 + k_0 + l_0 & \rightarrow x_3 & & \\
f_4(b_2,d_1) = b_2d_1 + r_2 + r_3 + k_0 & \rightarrow x_4 & x_3 + x_4 + x_5 = x_1 & \\
f_5(b_1,d_2) = b_1d_2 + r_3 + r_4 + l_0 & \rightarrow x_5 & & \\
\hline
f_6(b_0,d_0,a_0) = b_0d_0 + a_0 + k_0 + l_0 & \rightarrow x_6 & & \\
f_7(b_0,d_1) = b_0d_1 + r_4 + r_5 + k_0 & \rightarrow x_7 & x_6 + x_7 + x_8 = x_2 & \\
f_8(b_1,d_0) = b_1d_0 + r_5 + r_0 + l_0 & \rightarrow x_8 & &
\end{array}
$$

$$
\begin{array}{lll|l}
g_0(d_1,c_1,b_2) = d_1c_1 + b_1 + m_0 + n_0 & \rightarrow y_0 & & \\
g_1(d_1,c_2) = d_1c_2 + r_6 + r_7 + m_0 & \rightarrow y_1 & y_0 + y_1 + y_2 = y_0 & \\
g_2(d_2,c_1) = d_2c_1 + r_7 + r_8 + n_0 & \rightarrow y_2 & & \\
\hline
g_3(d_2,c_2,b_1) = d_2c_2 + b_2 + m_0 + n_0 & \rightarrow y_3 & & \\
g_4(d_2,c_0) = d_2c_0 + r_8 + r_9 + m_0 & \rightarrow y_4 & y_3 + y_4 + y_5 = y_1 & \\
g_5(d_0,c_2) = d_0c_2 + r_9 + r_{10} + n_0 & \rightarrow y_5 & & \\
\hline
g_6(d_0,c_0,b_0) = d_0c_0 + b_0 + m_0 + n_0 & \rightarrow y_6 & & \\
g_7(d_0,c_1) = d_0c_1 + r_{10} + r_{11} + m_0 & \rightarrow y_7 & y_6 + y_7 + y_8 = y_2 & \\
g_8(d_1,c_0) = d_1c_0 + r_{11} + r_6 + n_0 & \rightarrow y_8 & &
\end{array}
$$

$$
\begin{array}{l}
c_1 \rightarrow z_0 \\
c_2 \rightarrow z_1 \\
c_0 \rightarrow z_2
\end{array}
$$

$$
\begin{array}{l}
d_1 \rightarrow t_0 \\
d_2 \rightarrow t_1 \\
d_0 \rightarrow t_2
\end{array}
$$

E 3-share Midori S-Box with 36-bit Fresh Masks

$S = G \circ F \circ A_1$ with lookup table `cad3ebf789150246`
$A_1(a,\ b,\ c,\ d) = (b,\ a,\ d,\ a+c)$ with lookup table`0a1b82934e5fc6d7`

$F(a,\ b,\ c,\ d) = (x,\ y,\ z,\ t)$ with lookup table `84b70c3f951d2ea6`
$x = f(a,b,c,d) = bd + cd + b + d$ $\quad$ $y = g(a,b,c,d) = bd + cd + b$
$z = h(a,b,c,d) = a$ $\quad$ $t = u(a,b,c,d) = bd + a + c + 1$
$(k,\ l,\ m,\ n)$ are the input variables of the neighboring S-box's F function.

$f_0(d_1,b_1,c_1)$	$= d_1b_1 + d_1c_1 + k_0 + l_0$	$\rightarrow x_0$	
$f_1(d_1,b_2,c_2,d_1)$	$= d_1b_2 + d_1c_2 + b_2 + d_1 + r_0 + r_1 + k_0$	$\rightarrow x_1$	$x_0 + x_1 + x_2 \rightarrow x_0$
$f_2(d_2,b_1,c_1)$	$= d_2b_1 + d_2c_1 + r_1 + r_2 + l_0$	$\rightarrow x_2$	
$f_3(d_2,b_2,c_2)$	$= d_2b_2 + d_2c_2 + k_0 + l_0$	$\rightarrow x_3$	
$f_4(d_2,b_0,c_0,d_2)$	$= d_2b_0 + d_2c_0 + b_0 + d_2 + r_2 + r_3 + k_0$	$\rightarrow x_4$	$x_3 + x_4 + x_5 \rightarrow x_1$
$f_5(d_0,b_2,c_2)$	$= d_0b_2 + d_0c_2 + r_3 + r_4 + l_0$	$\rightarrow x_5$	
$f_6(d_0,b_0,c_0)$	$= d_0b_0 + d_0c_0 + k_0 + l_0$	$\rightarrow x_6$	
$f_7(d_0,b_1,c_1,d_0)$	$= d_0b_1 + d_0c_1 + b_1 + d_0 + r_4 + r_5 + k_0$	$\rightarrow x_7$	$x_6 + x_7 + x_8 \rightarrow x_2$
$f_8(d_1,b_0,c_0)$	$= d_1b_0 + d_1c_0 + r_5 + r_0 + l_0$	$\rightarrow x_8$	
$g_0(d_1,b_1,c_1)$	$= d_1b_1 + d_1c_1 + m_0 + n_0$	$\rightarrow y_0$	
$g_1(d_1,b_2,c_2)$	$= d_1b_2 + d_1c_2 + b_2 + r_6 + r_7 + m_0$	$\rightarrow y_1$	$y_0 + y_1 + y_2 \rightarrow y_0$
$g_2(d_2,b_1,c_1)$	$= d_2b_1 + d_2c_1 + r_7 + r_8 + n_0$	$\rightarrow y_2$	
$g_3(d_2,b_2,c_2)$	$= d_2b_2 + d_2c_2 + m_0 + n_0$	$\rightarrow y_3$	
$g_4(d_2,b_0,c_0)$	$= d_2b_0 + d_2c_0 + b_0 + r_8 + r_9 + m_0$	$\rightarrow y_4$	$y_3 + y_4 + y_5 \rightarrow y_1$
$g_5(d_0,b_2,c_2)$	$= d_0b_2 + d_0c_2 + r_9 + r_{10} + n_0$	$\rightarrow y_5$	
$g_6(d_0,b_0,c_0)$	$= d_0b_0 + d_0c_0 + m_0 + n_0$	$\rightarrow y_6$	
$g_7(d_0,b_1,c_1)$	$= d_0b_1 + d_0c_1 + b_1 + r_{10} + r_{11} + m_0$	$\rightarrow y_7$	$y_6 + y_7 + y_8 \rightarrow y_2$
$g_8(d_1,b_0,c_0)$	$= d_1b_0 + d_1c_0 + r_{11} + r_6 + n_0$	$\rightarrow y_8$	
	a_1	$\rightarrow z_0$	
	a_2	$\rightarrow z_1$	
	a_0	$\rightarrow z_2$	
$u_0(d_1,b_1)$	$= d_1b_1 + 1 + k_1 + l_1$	$\rightarrow t_0$	
$u_1(d_1,b_2,c_2)$	$= d_1b_2 + c_2 + r_{12} + r_{13} + k_1$	$\rightarrow t_1$	$t_0 + t_1 + t_2 \rightarrow t_0$
$u_2(d_2,b_1,a_1)$	$= d_2b_1 + a_1 + r_{13} + r_{14} + l_1$	$\rightarrow t_2$	
$u_3(d_2,b_2)$	$= d_2b_2 + k_1 + l_1$	$\rightarrow t_3$	
$u_4(d_2,b_0,c_0)$	$= d_2b_0 + c_0 + r_{14} + r_{15} + k_1$	$\rightarrow t_4$	$t_3 + t_4 + t_5 \rightarrow t_1$
$u_5(d_0,b_2,a_2)$	$= d_0b_2 + a_2 + r_{15} + r_{16} + l_1$	$\rightarrow t_5$	
$u_6(d_0,b_0)$	$= d_0b_0 + k_1 + l_1$	$\rightarrow t_6$	
$u_7(d_0,b_1,c_1)$	$= d_0b_1 + c_1 + r_{16} + r_{17} + k_1$	$\rightarrow t_7$	$t_6 + t_7 + t_8 \rightarrow t_2$
$u_8(d_1,b_0,a_0)$	$= d_1b_0 + a_0 + r_{17} + r_{12} + l_1$	$\rightarrow t_8$	

$G(a,\ b,\ c,\ d) = (x,\ y,\ z,\ t)$ with lookup table `8a02df57ce9b1346`
$x = f(a,b,c,d) = bd + c$
$y = g(a,b,c,d) = a$
$z = h(a,b,c,d) = bd + c + d$
$t = k(a,b,c,d) = bd + cd + b + 1$

$$
\begin{array}{lll|lll}
f_0(d_1,b_1) & = d_1b_1 & & \to x_0 & & \\
f_1(d_1,b_2,c_2) & = d_1b_2 + c_2 + r_{18} + r_{19} & & \to x_1 & & x_0 + x_1 + x_2 = x_0 \\
f_2(d_2,b_1) & = d_2b_1 + r_{19} + r_{20} & & \to x_2 & & \\
\hline
f_3(d_2,b_2) & = d_2b_2 & & \to x_3 & & \\
f_4(d_2,b_0,c_0) & = d_2b_0 + c_0 + r_{20} + r_{21} & & \to x_4 & & x_3 + x_4 + x_5 = x_1 \\
f_5(d_0,b_2) & = d_0b_2 + r_{21} + r_{22} & & \to x_5 & & \\
\hline
f_6(d_0,b_0) & = d_0b_0 & & \to x_6 & & \\
f_7(d_0,b_1,c_1) & = d_0b_1 + c_1 + r_{22} + r_{23} & & \to x_7 & & x_6 + x_7 + x_8 = x_2 \\
f_8(d_1,b_0) & = d_1b_0 + r_{23} + r_{18} & & \to x_8 & &
\end{array}
$$

$$
\begin{array}{lll}
a_1 & \to y_0 \\
a_2 & \to y_1 \\
a_0 & \to y_2
\end{array}
$$

$$
\begin{array}{lll|lll}
h_0(d_1,b_1) & = d_1b_1 & & \to z_0 & & \\
h_1(d_1,b_2) & = d_1b_2 + d_1 + r_{24} + r_{25} & & \to z_1 & & z_0 + z_1 + z_2 = z_0 \\
h_2(d_2,b_1,c_2) & = d_2b_1 + c_2 + r_{25} + r_{26} & & \to z_2 & & \\
\hline
h_3(d_2,b_2) & = d_2b_2 & & \to z_3 & & \\
h_4(d_2,b_0) & = d_2b_0 + d_2 + r_{26} + r_{27} & & \to z_4 & & z_3 + z_4 + z_5 = z_1 \\
h_5(d_0,b_2,c_0) & = d_0b_2 + c_0 + r_{27} + r_{28} & & \to z_5 & & \\
\hline
h_6(d_0,b_0) & = d_0b_0 & & \to z_6 & & \\
h_7(d_0,b_1) & = d_0b_1 + d_0 + r_{28} + r_{29} & & \to z_7 & & z_6 + z_7 + z_8 = z_2 \\
h_8(d_1,b_0,c_1) & = d_1b_0 + c_1 + r_{29} + r_{24} & & \to z_8 & &
\end{array}
$$

$$
\begin{array}{lll|lll}
k_0(d_1,b_1,c_1) & = d_1b_1 + d_1c_1 + 1 & & \to t_0 & & \\
k_1(d_1,b_2,c_2) & = d_1b_2 + d_1c_2 + b_2 + r_{30} + r_{31} & & \to t_1 & & t_0 + t_1 + t_2 = t_0 \\
k_2(d_2,b_1,c_1) & = d_2b_1 + d_2c_1 + r_{31} + r_{32} & & \to t_2 & & \\
\hline
k_3(d_2,b_2,c_2) & = d_2b_2 + d_2c_2 & & \to t_3 & & \\
k_4(d_2,b_0,c_0) & = d_2b_0 + d_2c_0 + b_0 + r_{32} + r_{33} & & \to t_4 & & t_3 + t_4 + t_5 = t_1 \\
k_5(d_0,b_2,c_2) & = d_0b_2 + d_0c_2 + r_{33} + r_{34} & & \to t_5 & & \\
\hline
k_6(d_0,b_0,c_0) & = d_0b_0 + d_0c_0 & & \to t_6 & & \\
k_7(d_0,b_1,c_1) & = d_0b_1 + d_0c_1 + b_1 + r_{34} + r_{35} & & \to t_7 & & t_6 + t_7 + t_8 = t_2 \\
k_8(d_1,b_0,c_0) & = d_1b_0 + d_1c_0 + r_{35} + r_{30} & & \to t_8 & &
\end{array}
$$

F 3-share Prince S-Box Inverse with 42-bit Fresh Masks

$S = H \circ G \circ F$ with lookup table `cad3ebf789150246`

$F(a,\ b,\ c,\ d) = (x,\ y,\ z,\ t)$ with lookup table `d850ba32c149e76f`
$x = f(a,b,c,d) = 1 + a + d$ $\qquad y = g(a,b,c,d) = c$
$z = h(a,b,c,d) = ac + cd + a + c + 1$ $\qquad t = u(a,b,c,d) = ad + b + 1$
$(k,\ l,\ m,\ n)$ are the input variables of the neighboring S-box's F function.

$$
\begin{aligned}
1 + a_1 + d_1 + c_2 &\rightarrow x_0\\
a_2 + d_2 + c_2 &\rightarrow x_1\\
a_0 + d_0 &\rightarrow x_2\\
\\
c_1 &\rightarrow y_0\\
c_2 &\rightarrow y_1\\
c_0 &\rightarrow y_2
\end{aligned}
$$

$h_0(d_1,c_1,a_1) = d_1c_1 + a_1c_1 + a_1 + 1 + k_0 + l_0$	$\rightarrow z_0$		
$h_1(d_1,c_2,a_1) = d_1c_2 + a_1c_2 + c_2 + r_0 + r_1 + k_0$	$\rightarrow z_1$	$z_0 + z_1 + z_2$	$\rightarrow z_0$
$h_2(d_2,c_1,a_2) = d_2c_1 + a_2c_1 + r_1 + r_2 + l_0$	$\rightarrow z_2$		
$h_3(d_2,c_2,a_2) = d_2c_2 + a_2c_2 + a_2 + k_0 + l_0$	$\rightarrow z_3$		
$h_4(d_2,c_0,a_2) = d_2c_0 + a_2c_0 + c_0 + r_2 + r_3 + k_0$	$\rightarrow z_4$	$z_3 += z_4 + z_5$	$\rightarrow z_1$
$h_5(d_0,c_2,a_0) = d_0c_2 + a_0c_2 + r_3 + r_4 + l_0$	$\rightarrow z_5$		
$h_6(d_0,c_0,a_0) = d_0c_0 + a_0c_0 + a_0 + k_0 + l_0$	$\rightarrow z_6$		
$h_7(d_0,c_1,a_0) = d_0c_1 + a_0c_1 + c_1 + r_4 + r_5 + k_0$	$\rightarrow z_7$	$z_6 + z_7 + z_8$	$\rightarrow z_2$
$h_8(d_1,c_0,a_1) = d_1c_0 + a_1c_0 + r_5 + r_0 + l_0$	$\rightarrow z_8$		
$u_0(d_1,b_1) = d_1a_1 + 1 + m_0 + n_0$	$\rightarrow t_0$		
$u_1(d_1,b_2,b_1) = d_1a_2 + b_1 + r_6 + r_7 + m_0$	$\rightarrow t_1$	$t_0 + t_1 + t_2$	$= t_0$
$u_2(d_2,b_1) = d_2a_1 + r_7 + r_8 + n_0$	$\rightarrow t_2$		
$u_3(d_2,b_2) = d_2a_2 + m_0 + n_0$	$\rightarrow t_3$		
$u_4(d_2,b_0,b_2) = d_2a_0 + b_2 + r_8 + r_9 + m_0$	$\rightarrow t_4$	$t_3 + t_4 + t_5$	$= t_1$
$u_5(d_0,b_2) = d_0a_2 + r_9 + r_{10} + n_0$	$\rightarrow t_5$		
$u_6(d_0,b_0) = d_0a_0 + m_0 + n_0$	$\rightarrow t_6$		
$u_7(d_0,b_1,b_0) = d_0a_1 + b_0 + r_{10} + r_{11} + m_0$	$\rightarrow t_7$	$t_6 + t_7 + t_8$	$= t_2$
$u_8(d_1,b_0) = d_1a_0 + r_{11} + r_6 + n_0$	$\rightarrow t_8$		

$G(a,\ b,\ c,\ d) = (x,\ y,\ z,\ t)$ with lookup table `08c43bf72a6ed591`
$x = f(a,b,c,d) = c$
$y = g(a,b,c,d) = c + d$
$z = h(a,b,c,d) = cd + b$
$t = k(a,b,c,d) = bd + cd + a + b$
$(o,\ p,\ q,\ s)$ are the input variables of the neighboring S-box's G function.

$$
\begin{array}{lll}
c_1 & \to x_0 & \\
c_2 & \to x_1 & \\
c_0 & \to x_2 & \\
 & & \\
c_1 + d_1 + q_1 & \to y_0 & \\
c_2 + q_1 + s_1 & \to y_1 & y_0 + y_1 = y_0 \\
d_2 & \to y_2 & y_2 = y_1 \\
c_0 + d_0 + s_1 & \to y_3 & y_3 = y_2
\end{array}
$$

$$
\begin{array}{lll}
h_0(d_1,c_1,b_1) = d_1c_1 + b_1 + q_0 + s_0 & \to z_0 & \\
h_1(d_1,c_2) = d_1c_2 + r_{12} + r_{13} + q_0 & \to z_1 & z_0 + z_1 + z_2 = z_0 \\
h_2(d_2,c_1) = d_2c_1 + r_{13} + r_{14} + s_0 & \to z_2 & \\
\hline
h_3(d_2,c_2,b_2) = d_2c_2 + b_2 + q_0 + s_0 & \to z_3 & \\
h_4(d_2,c_0) = d_2c_0 + r_{14} + r_{15} + q_0 & \to z_4 & z_3 + z_4 + z_5 = z_1 \\
h_5(d_0,c_2) = d_0c_2 + r_{15} + r_{16} + s_0 & \to z_5 & \\
\hline
h_6(d_0,c_0,b_0) = d_0c_0 + b_0 + q_0 + s_0 & \to z_6 & \\
h_7(d_0,c_1) = d_0c_1 + r_{16} + r_{17} + q_0 & \to z_7 & z_6 + z_7 + z_8 = z_2 \\
h_8(d_1,c_0) = d_1c_0 + r_{17} + r_{12} + s_0 & \to z_8 &
\end{array}
$$

$$
\begin{array}{lll}
k_0(d_1,b_1,c_1,a_1) = d_1b_1 + d_1c_1 + b_1 + a_1 + o_0 + p_0 & \to t_0 & \\
k_1(d_1,b_2,c_2) = d_1b_2 + d_1c_2 + r_{18} + r_{19} + o_0 & \to t_1 & t_0 + t_1 + t_2 = t_0 \\
k_2(d_2,b_1,c_1) = d_2b_1 + d_2c_1 + r_{19} + r_{20} + p_0 & \to t_2 & \\
\hline
k_3(d_2,b_2,c_2,a_2) = d_2b_2 + d_2c_2 + b_2 + a_2 + o_0 + p_0 & \to t_3 & \\
k_4(d_2,b_0,c_0) = d_2b_0 + d_2c_0 + r_{20} + r_{21} + o_0 & \to t_4 & t_3 + t_4 + t_5 = t_1 \\
k_5(d_0,b_2,c_2) = d_0b_2 + d_0c_2 + r_{21} + r_{22} + p_0 & \to t_5 & \\
\hline
k_6(d_0,b_0,c_0,a_0) = d_0b_0 + d_0c_0 + b_0 + a_0 + o_0 + p_0 & \to t_6 & \\
k_7(d_0,b_1,c_1) = d_0b_1 + d_0c_1 + r_{22} + r_{23} + o_0 & \to t_7 & t_6 + t_7 + t_8 = t_2 \\
k_8(d_1,b_0,c_0) = d_1b_0 + d_1c_0 + r_{23} + r_{18} + p_0 & \to t_8 &
\end{array}
$$

$H(a,\ b,\ c,\ d) = (x,\ y,\ z,\ t)$ with lookup table `21748bde65039afc`

$x = f(a,b,c,d) = bd + cd + a + b$

$y = g(a,b,c,d) = bd + a + c + 1$

$z = h(a,b,c,d) = cd + b + d$

$t = k(a,b,c,d) = c$

$f_0(d_1, c_1, b_1)$	$= d_1b_1 + d_1c_1 + b_1$	$\rightarrow x_0$	
$f_1(d_1, c_2, b_2, a_1)$	$= d_1b_2 + d_1c_2 + a_1 + r_{24} + r_{25}$	$\rightarrow x_1$	$x_0 + x_1 + x_2 = x_0$
$f_2(d_2, c_1, b_1)$	$= d_2b_1 + d_2c_1 + r_{25} + r_{26}$	$\rightarrow x_2$	
$f_3(d_2, c_2, b_2)$	$= d_2b_2 + d_2c_2 + b_2$	$\rightarrow x_3$	
$f_4(d_2, c_0, b_0, a_2)$	$= d_2b_0 + d_2c_0 + a_2 + r_{26} + r_{27}$	$\rightarrow x_4$	$x_3 + x_4 + x_5 = x_1$
$f_5(d_0, c_2, b_2)$	$= d_0b_2 + d_0c_2 + r_{27} + r_{28}$	$\rightarrow x_5$	
$f_6(d_0, c_0, b_0)$	$= d_0b_0 + d_0c_0 + b_0$	$\rightarrow x_6$	
$f_7(d_0, c_1, b_1, a_0)$	$= d_0b_1 + d_0c_1 + a_0 + r_{28} + r_{29}$	$\rightarrow x_7$	$x_6 + x_7 + x_8 = x_2$
$f_8(d_1, c_0, b_0)$	$= d_1b_0 + d_1c_0 + r_{29} + r_{24}$	$\rightarrow x_8$	
$g_0(d_1, b_1, a_1)$	$= d_1b_1 + a_1 + 1$	$\rightarrow y_0$	
$g_1(d_1, b_2)$	$= d_1b_2 + r_{30} + r_{31}$	$\rightarrow y_1$	$y_0 + y_1 + y_2 = y_0$
$g_2(d_2, b_1, c_2)$	$= d_2b_1 + c_2 + r_{31} + r_{32}$	$\rightarrow y_2$	
$g_3(d_2, b_2, a_2)$	$= d_2b_2 + a_2$	$\rightarrow y_3$	
$g_4(d_2, b_0)$	$= d_2b_0 + r_{32} + r_{33}$	$\rightarrow y_4$	$y_3 + y_4 + y_5 = y_1$
$g_5(d_0, b_2, c_0)$	$= d_0b_2 + c_0 + r_{33} + r_{34}$	$\rightarrow y_5$	
$g_6(d_0, b_0, a_0)$	$= d_0b_0 + a_0$	$\rightarrow y_6$	
$g_7(d_0, b_1)$	$= d_0b_1 + r_{34} + r_{35}$	$\rightarrow y_7$	$y_6 + y_7 + y_8 = y_2$
$g_8(d_1, b_0, c_1)$	$= d_1b_0 + c_1 + r_{35} + r_{30}$	$\rightarrow y_8$	
$h_0(d_1, b_1, c_1)$	$= d_1c_1 + b_1$	$\rightarrow z_0$	
$h_1(d_1, b_2, c_2)$	$= d_1c_2 + c_2 + d_1 + r_{36} + r_{37}$	$\rightarrow z_1$	$z_0 + z_1 + z_2 = z_0$
$h_2(d_2, b_1, c_1)$	$= d_2c_1 + r_{37} + r_{38}$	$\rightarrow z_2$	
$h_3(d_2, b_2, c_2)$	$= d_2c_2 + b_2$	$\rightarrow z_3$	
$h_4(d_2, b_0, c_0)$	$= d_2c_0 + d_2 + r_{38} + r_{39}$	$\rightarrow z_4$	$z_3 + z_4 + z_5 = z_1$
$h_5(d_0, b_2, c_2)$	$= d_0c_2 + c_2 + r_{39} + r_{40}$	$\rightarrow z_5$	
$h_6(d_0, b_0, c_0)$	$= d_0c_0 + b_0$	$\rightarrow z_6$	
$h_7(d_0, b_1, c_1)$	$= d_0c_1 + d_0 + r_{40} + r_{41}$	$\rightarrow z_7$	$z_6 + z_7 + z_8 = z_2$
$h_8(d_1, b_0, c_0)$	$= d_1c_0 + r_{41} + r_{36}$	$\rightarrow z_8$	
		$c_1 \rightarrow t_0$	
		$c_2 \rightarrow t_1$	
		$c_0 \rightarrow t_2$	

References

[BBI+15] Subhadeep Banik, Andrey Bogdanov, Takanori Isobe, Kyoji Shibutani, Harunaga Hiwatari, Toru Akishita, and Francesco Regazzoni. Midori: A block cipher for low energy. In *ASIACRYPT 2015*, volume 9453 of *Lecture Notes in Computer Science*, pages 411–436. Springer, 2015.

[BCG+12] Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knezevic, Lars R. Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar, Christian Rechberger, Peter Rombouts, Søren S. Thomsen, and Tolga Yalçin. PRINCE - A low-latency block cipher for pervasive computing applications - extended abstract. In *ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 208–225. Springer, 2012.

[BDZ20] Tim Beyne, Siemen Dhooghe, and Zhenda Zhang. Cryptanalysis of masked ciphers: A not so random idea. In *ASIACRYPT 2020*, volume 12491 of *Lecture Notes in Computer Science*, pages 817–850. Springer, 2020.

[BEK+20] Dusan Bozilov, Maria Eichlseder, Miroslav Knezevic, Baptiste Lambin, Gregor Leander, Thorben Moos, Ventzislav Nikov, Shahram Rasoolzadeh, Yosuke Todo, and Friedrich Wiemer. Princev2 - more security for (almost) no overhead. In *SAC 2020*, volume 12804 of *Lecture Notes in Computer Science*, pages 483–511. Springer, 2020.

[Bey18] Tim Beyne. Block cipher invariants as eigenvectors of correlation matrices. In *ASIACRYPT 2018*, volume 11272 of *Lecture Notes in Computer Science*, pages 3–31. Springer, 2018.

[BGN+14] Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. Higher-order threshold implementations. In *ASIACRYPT 2014*, volume 8874 of *Lecture Notes in Computer Science*, pages 326–343. Springer, 2014.

[BJK+16] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY family of block ciphers and its low-latency variant MANTIS. In *CRYPTO 2016*, volume 9815 of *Lecture Notes in Computer Science*, pages 123–153. Springer, 2016.

[BKL+07] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: an ultra-lightweight block cipher. In *CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007.

[BKN19] Dusan Bozilov, Miroslav Knezevic, and Ventzislav Nikov. Optimized Threshold Implementations: Minimizing the Latency of Secure Cryptographic Accelerators. In *CARDIS 2019*, volume 11833 of *Lecture Notes in Computer Science*, pages 20–39. Springer, 2019.

[BNN+12] Begül Bilgin, Svetla Nikova, Ventzislav Nikov, Vincent Rijmen, and Georg Stütz. Threshold implementations of all 3 $\times$3 and 4 $\times$4 s-boxes. In *CHES 2012*, volume 7428 of *Lecture Notes in Computer Science*, pages 76–91. Springer, 2012.

[CBR+15] Thomas De Cnudde, Begül Bilgin, Oscar Reparaz, Ventzislav Nikov, and Svetla Nikova. Higher-order threshold implementation of the AES s-box. In

CARDIS 2015, volume 9514 of *Lecture Notes in Computer Science*, pages 259–272. Springer, 2015.

[CBRN14] Thomas De Cnudde, Begül Bilgin, Oscar Reparaz, and Svetla Nikova. Higher-Order Glitch Resistant Implementation of the PRESENT S-Box. In *BalkanCryptSec 2014*, volume 9024 of *Lecture Notes in Computer Science*, pages 75–93. Springer, 2014.

[CDG+13] Jeremy Cooper, Elke DeMulder, Gilbert Goodwill, Joshua Jaffe, Gary Kenworthy, Pankaj Rohatgi, et al. Test vector leakage assessment (TVLA) methodology in practice. In *International Cryptographic Module Conference*, volume 20, 2013.

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In *CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999.

[CRB+16] Thomas De Cnudde, Oscar Reparaz, Begül Bilgin, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. Masking AES with d+1 Shares in Hardware. In *CHES 2016*, volume 9813 of *Lecture Notes in Computer Science*, pages 194–212. Springer, 2016.

[Dae17] Joan Daemen. Changing of the guards: A simple and efficient method for achieving uniformity in threshold sharing. In *CHES 2017*, volume 10529 of *Lecture Notes in Computer Science*, pages 137–153. Springer, 2017.

[DDF14] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. In *EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 423–440. Springer, 2014.

[DFH+16] Stefan Dziembowski, Sebastian Faust, Gottfried Herold, Anthony Journault, Daniel Masny, and François-Xavier Standaert. Towards sound fresh re-keying with hard (physical) learning problems. In *CRYPTO 2016*, volume 9815 of *Lecture Notes in Computer Science*, pages 272–301. Springer, 2016.

[DGV94] Joan Daemen, René Govaerts, and Joos Vandewalle. Correlation matrices. In *Fast Software Encryption: Second International Workshop. Leuven, Belgium, 14-16 December 1994, Proceedings*, volume 1008 of *Lecture Notes in Computer Science*, pages 275–285. Springer, 1994.

[DP08] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 293–302. IEEE Computer Society, 2008.

[DR01] Joan Daemen and Vincent Rijmen. The wide trail design strategy. In *Cryptography and Coding, 8th IMA International Conference, Cirencester, UK, December 17-19, 2001, Proceedings*, volume 2260 of *Lecture Notes in Computer Science*, pages 222–238. Springer, 2001.

[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.

[GIS14] Hendra Guntur, Jun Ishii, and Akashi Satoh. Side-channel attack user reference architecture board SAKURA-G. In *GCCE 2014*, pages 271–274. IEEE, 2014.

[GMK16] Hannes Groß, Stefan Mangard, and Thomas Korak. Domain-Oriented Masking: Compact Masked Hardware Implementations with Arbitrary Protection Order. In *Theory of Implementation Security - TIS@CCS 2016*, page 3. ACM, 2016.

[GPPR11] Jian Guo, Thomas Peyrin, Axel Poschmann, and Matthew J. B. Robshaw. The LED block cipher. In *CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*, pages 326–341. Springer, 2011.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[KSM20] David Knichel, Pascal Sasdrich, and Amir Moradi. SILVER - statistical independence and leakage verification. In *ASIACRYPT 2020*, volume 12491 of *Lecture Notes in Computer Science*, pages 787–816. Springer, 2020.

[Mat93] Mitsuru Matsui. Linear cryptanalysis method for DES cipher. In *EUROCRYPT '93*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer, 1993.

[MMW18] 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.

[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 *EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 69–88. Springer, 2011.

[MPO05] Stefan Mangard, Norbert Pramstaller, and Elisabeth Oswald. Successfully attacking masked AES hardware implementations. In *CHES 2005*, volume 3659 of *Lecture Notes in Computer Science*, pages 157–171. Springer, 2005.

[MS16] Amir Moradi and Tobias Schneider. Side-channel analysis protection and low-latency in action - - case study of PRINCE and midori -. In *ASIACRYPT 2016*, volume 10031 of *Lecture Notes in Computer Science*, pages 517–547, 2016.

[NPB15] Aina Niemetz, Mathias Preiner, and Armin Biere. Boolector 2.0 system description. *Journal on Satisfiability, Boolean Modeling and Computation*, 9:53–58, 2015.

[NRR06] Svetla Nikova, Christian Rechberger, and Vincent Rijmen. Threshold implementations against side-channel attacks and glitches. In *Information and Communications Security, 8th International Conference, ICICS 2006*, volume 4307 of *Lecture Notes in Computer Science*, pages 529–545. Springer, 2006.

[PGMP19] Thomas Prest, Dahmun Goudarzi, Ange Martinelli, and Alain Passelègue. Unifying leakage models on a rényi day. In *CRYPTO 2019*, volume 11692 of *Lecture Notes in Computer Science*, pages 683–712. Springer, 2019.

[PR13] Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In *EUROCRYPT 2013*, 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 *CRYPTO 2015*, volume 9215 of *Lecture Notes in Computer Science*, pages 764–783. Springer, 2015.

[Rep15] Oscar Reparaz. A note on the security of higher-order threshold implementations. *IACR Cryptol. ePrint Arch.*, page 1, 2015.

[SIH+11] Kyoji Shibutani, Takanori Isobe, Harunaga Hiwatari, Atsushi Mitsuda, Toru Akishita, and Taizo Shirai. Piccolo: An ultra-lightweight blockcipher. In *CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*, pages 342–357. Springer, 2011.

[SM15] Tobias Schneider and Amir Moradi. Leakage Assessment Methodology - A Clear Roadmap for Side-Channel Evaluations. In *CHES 2015*, volume 9293 of *Lecture Notes in Computer Science*, pages 495–513. Springer, 2015.

[SM21a] Aein Rezaei Shahmirzadi and Amir Moradi. Re-consolidating first-order masking schemes nullifying fresh randomness. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(1):305–342, 2021.

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

[TG91] Anne Tardy-Corfdir and Henri Gilbert. A known plaintext attack of FEAL-4 and FEAL-6. In Joan Feigenbaum, editor, *CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 172–181. Springer, 1991.

[TLS16] Yosuke Todo, Gregor Leander, and Yu Sasaki. Nonlinear invariant attack - practical attack on full scream, iscream, and midori64. In *ASIACRYPT 2016*, volume 10032 of *Lecture Notes in Computer Science*, pages 3–33, 2016.