Threshold Implementations with Non-Uniform Inputs

Siemen Dhooghe$^{[0000-0003-0591-7355]}$ and
Artemii Ovchinnikov$^{[0000-0002-9035-523X]}$

COSIC, KU Leuven, Leuven, Belgium
firstname.lastname@esat.kuleuven.be

Abstract Modern block ciphers designed for hardware and masked with Threshold Implementations (TIs) provide provable security against first-order attacks. However, the application of TIs leaves designers to deal with a trade-off between its security and its cost, for example, the process to generate its required random bits. This generation cost comes with an increased overhead in terms of area and latency. Decreasing the number of random bits for the masking allows to reduce the aforementioned overhead.
We propose to reduce the randomness to mask the secrets, like the plaintext. For that purpose, we suggest ***relaxing the requirement for the uniformity*** of the input shares and ***reuse randomness*** for their masking in first-order TIs. We apply our countermeasures to first-order TIs of the Prince and Midori64 ciphers with three shares. Since the designs with non-uniform masks are no longer perfect first-order probing secure, we provide further analysis by calculating bounds on the advantage of a noisy threshold-probing adversary. We then make use of the PROLEAD tool, which implements statistical tests verifying the robust probing security to compare its output with our estimates. Finally, we evaluate the designs on FPGA to highlight the practical security of our solution. We observe that their security holds while requiring four times less randomness over uniform TIs.

Keywords: FPGA, Masking, Probing Security, Threshold Implementations, Uniformity

1 Introduction

For the past two decades after Kocher *et al.* [16] presented differential power analysis in 1999, one example of a side-channel attack, the development of protected cryptographic hardware devices has turned into an important goal for researchers and designers. To that end, masking has become a reliable countermeasure. In masking, a secret is split into several parts to confound the correlation between its value and some physical characteristics, like its power consumption. However, to eliminate the mentioned statistical dependence, the shares of the masking have to be uniform random such that they are independent of the secret.

Threshold Implementations (TIs), introduced in 2006 by Nikova *et al.* [21], allow to preserve this uniformity of the masks and provide for first-order security. As a result, one only needs some fixed number of random bits to be generated at the initial stage and no fresh random bits are needed for further re-masking. The process of randomness generation needed for this uniformity lacks attention in terms of cost and security from the research community. As a result, to be on the safe side, researchers recommend heavy-duty random number generators based on standardised cryptographic primitives. However, in practice, lighter non-cryptographic generators are used whose security remains unknown. Instead of deducting which lightweight random number generator can be used, designers of masking schemes focus on reducing the total number of random bits required for the masking to be secure. The security of this question can more easily be verified by using the probing model by Ishai *et al.* [15] which has become the standard model for masking countermeasures.

The research on reducing the randomness cost for maskings has significantly progressed. The work by Shahmirzadi and Moradi [23] reduces the fresh randomness needed for first-order designs using only two shares while their follow-up work [24] does the same for second-order designs. The work by Beyne *et al.* [7] in 2020 introduces the bounded-query probing model which allows to obtain a concrete bound on the adversary's advantage by means of a security framework based on linear cryptanalysis. In 2021, Beyne *et al.* [6] applied their scheme to make a low-randomness masked AES. This work was followed up in 2022 [5] to include the noise on probes to improve the bounds on the adversary and to improve the efficiency of their designs. For all three previous works, the security framework was always applied to second-order masked designs but never to the first-order designs. Its application to first-order designs provides an opportunity of using non-uniform randomness to generate secret shares. Via careful analysis, it can provide a way to reduce the cost of the initial randomness for TIs.

Contributions. In this paper, we investigate TIs of lightweight ciphers and their security when given non-uniform inputs. We provide a framework, based on the works by Beyne *et al.* [5, 7], to show when the implementations are secure and when they are not, and provide practical evidence to support it.

For the analysis, we work with the Prince [10] and Midori64 [1] ciphers and take previously established TIs of them, namely the one by Bozilov *et al.* [11] for Prince and by Moradi *et al.* [18] for Midori64. The goal of the analysis is to reduce the initial randomness needed to mask the plaintext for both ciphers. To that end, we investigate cases where we can re-use randomness for the initial masking and provide, for each cipher, insecure and secure cases using the same number of total random bits. This shows that it is not the total entropy of the masked input which counts, but instead, its relation to the properties of the masking of the cipher. We demonstrate that using the security framework via a trail based approach where we obtain bounds on a probing adversary's advantage.

In order to complete the research, we provide practical experiments on top of the previous mentioned theoretical analysis. We test the designs with non-uniform masked inputs using two different approaches. First, we use the PRO-

LEAD tool by Müller and Moradi [19] which allows for a noiseless statistical leakage evaluation based on the glitch-extended probing model. Second, we provide the results of a practical evaluation of our designs on FPGA. For the observed power consumption, we apply first-order statistical fixed vs. random t-tests [2]. The results of those tests are closer to practice as the noise is included in the sampled traces. As an end result, we show that the TIs by Bozilov *et al.* and by Moradi *et al.* can be used with a non-uniform masked input reducing the total needed randomness four times over while still providing practical security of up to 50M traces.[1]

2 Preliminaries

In this section, we introduce the probing security model together with the basics of masking and threshold implementations. We also introduce the cryptanalysis-based evaluation we use to determine which non-uniform inputs of threshold implementations are secure.

2.1 Glitch-Extended Bounded-Query Probing Security

We first introduce the bounded-query probing model [7] with a noisy probing variant [5]. The security model is depicted in Figure 1.

In this model, the security of a circuit C with input k against a t-threshold-probing adversary is quantified as follows. The challenger picks a random bit b and provides an oracle $\mathcal{O}^b$ (the masked circuit with b hardwired), to which adversary $\mathcal{A}$ is given query access. The adversary queries the oracle by choosing up to t wires of the masked circuit to probe, we denote this set by $\mathcal{P}$, and sends it to the oracle along with the inputs (for a cipher, both key and plaintext) k_0 and k_1. The oracle responds by giving back a noisy leakage function $\mathbf{f}$ (following the definition given in [5]) of the glitch-extended probed wire values of the masked circuit with input k_b. After a total of q queries, the adversary responds to the challenger with a guess for b. For $b \in \{0, 1\}$, 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

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

Since we are working on hardware, we make use of the glitch-extended probing model by Faust *et al.* [13]. 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, with the limit that glitches do not propagate through memory gates.

[1] The HDL representations of the constructed ciphers will be made publicly available on `https://github.com/KULeuven-COSIC/TIs_with_Non-Uniform_Inputs`

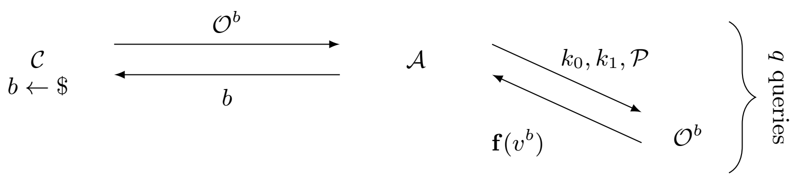


Figure 1: [5] The privacy model for the glitch-extended t-threshold-probing security consisting of a challenger $\mathcal{C}$, an adversary $\mathcal{A}$, a left-right oracle $\mathcal{O}^b$, two inputs k_0, k_1, a set of probes $\mathcal{P}$, and a noisy leakage function $\mathbf{f}(v^b)$ of the probed wire values v^b in the circuit $C(k_b)$.

2.2 Boolean Masking and Threshold Implementations

Boolean masking is a technique based on splitting each secret variable $x \in \mathbb{F}_2$ in the circuit into shares $\bar{x} = (x^0, x^1, \ldots, x^{s_x-1})$ such that $x = \sum_{i=0}^{s_x-1} x^i$ over $\mathbb{F}_2$. A random Boolean masking of a fixed secret is uniform if all maskings of that secret are equally likely. There are several approaches to masking a circuit. In this work, we make use of threshold implementations proposed by Nikova *et al.* [22].

Let $\bar{F}$ be a masked function in the threshold implementation corresponding to the unmasked function $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$. Then $\bar{F}$ can have the following properties.

Definition 1 (Threshold implementations [22]). *Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ be a function and $\bar{F} : \mathbb{F}_2^{ns_x} \to \mathbb{F}_2^{ms_y}$ be a masking of F. The masking $\bar{F}$ is said to be*

1. correct *if $\forall x^0, \ldots, x^{s_x-1} \in \mathbb{F}_2^n$, $\sum_{i=0}^{s_y-1} F^i(x^0, \ldots, x^{s_x-1}) = F(\sum_{i=0}^{s_x-1} x_i)$,*
2. non-complete *if any function share F^i depends on at most s_x-1 input shares,*
3. uniform *if $\bar{F}$ maps a uniform random masking of any $x \in \mathbb{F}_2^n$ to a uniform random masking of $F(x) \in \mathbb{F}_2^m$.*

2.3 Linear Cryptanalysis of Maskings

To prove the first-order probing security of a circuit, it is sufficient to show that the probed values consist only of uniform randomness and public values. To that end, we make use of the theory by Beyne *et al.* [7] which bounds the distribution of probed values (in a bounded query model) by their Fourier distribution. We also make use of the later work [5], where this framework was expanded to include the noise on the traces. We recall the main results of these works.

Bound on the Advantage. We first discuss the bound on a bounded-query probing adversary which uses λ-noisy probes. We refer to [5] on the specific definition of the used noisy leakage function. In the rest of this description, the noise parameter λ characterises the level of the noise. 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.

We assume that any probed wire value can be labelled as 'good' or 'bad'. The values labelled 'good' jointly reveal nothing about the secret. The 'bad' values may reveal secret information, but the leakage can be bounded in terms of λ and ε. The parameter λ is determined by physical aspects such as the leakage model and noise level. The parameter ε is instead determined by the mathematical properties of the masking. Specifically, it will be shown later how these parameters can be determined using linear cryptanalysis. Below is the definition of the bound on a first-order noisy probing adversary given a bound on ε and λ.

Theorem 1 **([5]).** *Let $\mathcal{A}$ be a noisy threshold-probing adversary for a circuit C. Take $\lambda \geq 1$, and $\varepsilon \leq 1$ as non-negative real numbers. Assume that for every query made by $\mathcal{A}$ on the oracle $\mathcal{O}^b$ 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 λ-noisy.*
2. *The conditional probability distribution $p_{\mathbf{y}|\mathbf{x}}$ satisfies $\mathbb{E}_{\mathbf{x}} \|\widehat{p}_{\mathbf{y}|\mathbf{x}}\|_2^2 \leq \varepsilon$.*
3. *Any t-threshold-probing adversary for the same circuit C and making the same oracle queries as $\mathcal{A}$, but which only receives the 'good' wire values (*i.e.* corresponding to $\mathbf{x}$) for each query, has advantage zero.*

The advantage of $\mathcal{A}$ can be upper bounded as

$$\mathrm{Adv}_{\mathsf{noisy}}(\mathcal{A}) \leq \sqrt{2q\,\varepsilon/\lambda}\,,$$

where q is the number of queries to the oracle $\mathcal{O}^b$.

The security bound obtained in Theorem 1 depends on the parameter ε. This value will be determined by performing linear cryptanalysis of the masked cipher. Essentially, this follows regular linear cryptanalysis, except that masking schemes naturally incorporate linear relations (namely that the sum of the shares form the secret). As a result, the basic definitions of linear cryptanalysis need to be adapted to work over a quotient space where, in short, the last share is removed to avoid the previously mentioned linear relation. Viewing linear cryptanalysis over this quotient space is justified by the non-completeness property of threshold implementations, namely that a probe does not view all shares of a secret at once, and as a result, we only investigate relations over non-complete sets of shares.

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

We use the following definition of correlation matrices of a masking.

Definition 2 (Correlation matrix). *For a subspace $\mathbb{V} \subseteq \mathbb{F}_2^\ell$, let $F : \mathbb{V} \to \mathbb{V}$ be a function. The correlation matrix C^F of F is a real $|\mathbb{V}| \times |\mathbb{V}|$ matrix with coordinates indexed by elements $u, v \in \mathbb{F}_2^n/\mathbb{V}^\perp$ and equal to*

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

for a function $F' : \mathbb{V}_a \to \mathbb{V}_b$ with $F'(x) = F(x + a) + b$.

The link between ε from Theorem 1 and linear cryptanalysis is completed by the theorem 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 $x \in \mathbb{V}_a$ to an index set $I = \{i_1, \ldots, i_m\}$ is denoted by $x_I = (x_{i_1}, \ldots, x_{i_m}) \in \mathbb{F}_2^{|I|}$. This definition depends on the specific choice of the representative a, but the result of the theorem does not.

Theorem 2 ([7], §5.2). *Let $F : \mathbb{V}_a \to \mathbb{V}_b$ be a function with $\mathbb{V} \subset \mathbb{F}_2^\ell$ and $I, J \subset \{1, \ldots, \ell\}$. For $\mathbf{x}$ uniform random on $\mathbb{V}_a$ 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 $|\widehat{p}_{\mathbf{z}}(u, v)| = |C^F_{\tilde{v}, \tilde{u}}|$, where $\widetilde{u}, \widetilde{v} \in \mathbb{F}_2^\ell/\mathbb{V}^\perp$ are such that $\widetilde{u}_I = u$, $\widetilde{u}_{[\ell]\setminus I} = 0$, $\widetilde{v}_J = v$ and $\widetilde{v}_{[\ell]\setminus J} = 0$.*

The above theorem relates the linear approximations of F to $\widehat{p}_{\mathbf{z}}(u, v)$ and hence provides a method to upper bound ε based on linear cryptanalysis.

Applying the Bound with Non-Uniform Inputs. The analysis by Beyne *et al.* originally applied to threshold implementations working on a uniform input or consisting of uniform functions. In this work, we extend this framework by analysing threshold implementations with a non-uniform input, namely an input which is shared via a non-uniform function. More specifically, we use a limited number of random bits to mask the input of the threshold implementation and analyse the impact on its first-order security. While previous works focus on reducing the online randomness of a masking, we instead propose to use the cryptanalysis technique to reduce the randomness requirement at the start of the masking.

We model this limited-random input by considering a non-uniform input encoder *Enc* (shown in Figure 2) which takes in the circuit's input k (e.g. plaintext and key) and random bits $\mathbf{r}$ (modelled as shares of zero and as 'bad' values), and provides a shared input for the masked circuit. In particular, where this shared input is larger in size than the randomness that was given as input.

Due to *Enc* being a non-uniform function, the correlation matrix's entries $C^{Enc}_{\widetilde{v},0}$ for $\widetilde{v} \in \mathbb{F}_2^\ell/\mathbb{V}^\perp$ are non-zero. As a result, when $|\widehat{p}_{\mathbf{y}_J}(v)| = |C^H_{\widetilde{v},0}| \neq 0$ for $H = F \circ Enc$ (some non-uniform function which maps the limited randomness $\mathbf{r}$ to the probed values), $\widetilde{v}_J = v$ and $\widetilde{v}_{[\ell]\setminus J} = 0$ where a single probe is placed on $\mathbf{y}_J$, it is possible that this probe reveals a secret. The specific value $\varepsilon = |\widehat{p}_{\mathbf{y}_J}(v)|$ determines the advantage of the first-order probing adversary via Theorem 1. In other words, due to the threshold implementation using a non-uniform input,

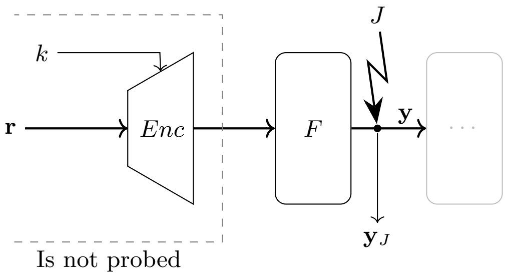


Figure 2: Depiction of the non-uniform encoder *Enc* masking the input (e.g., plaintext and key) k using a few uniform random bits $\mathbf{r}$.

a first-order probing attack is possible. However, we show the probability of success is limited in function of ε.

In the rest of the work, we will look at trails over F (which is a uniform function) where we can pick non-zero input linear masks with certain conditions depending on how the input of the cipher was masked. We thus analyse the trails of the masked cipher ending in a single probe position with conditions on the input mask related to how it is masked.

Cautionary Note. In this work, we use the piling-up principle [17, 26] to obtain estimates of ε using a trail-based approach (which is often used in the field of cryptanalysis). However, since *Enc* is a non-uniform (read non-balanced) function, a zero input mask will correlate to several output masks. As a result, and due to the first S-box layer, many input masks of F from Figure 2 are related to an output mask. As a result, the actual correlations may differ from the trail-based estimates. More specifically, we make the assumption that the correlation of the probed values is determined by a trail with an outstanding value. In case a trail with high correlation is found, we can assume that there is an attack. However, we emphasise that the absence of such trails does not trivially imply that no attack is possible. For that reason, we use this piling-up principle as a heuristic to find and verify promising non-uniform inputs for the threshold implementations. We then base ourselves on a practical verification to analyse the promising candidates via tools like PROLEAD [19] for a noiseless verification and via FPGA experiments for more realistic and noisy verification.

3 Analysis of TIs with Non-Uniform Inputs

We analyse threshold implementations of the Midori64 and Prince ciphers which are given non-uniform masked inputs. The threshold implementations consist of uniform functions which do not need fresh randomness for their computation. The security calculations of the non-uniform inputs are done via the cryptanalytic technique explained in Section 2.3. In essence, this analysis allows to bound the deviation of a masking from uniform. When this deviation is low, the masking is "almost uniform" and we can expect the masking to still be first-order secure. The analysis is mainly based on trail-based techniques where we investigate the activity patterns of the non-uniform inputs to discover whether there are weak probing points in the masked implementation. We then estimate the number of queries (or traces) a bounded-query probing adversary needs in order to get advantage one.

3.1 Masked Prince with Non-Uniform Inputs

We start the analysis with a threshold implementation of the Prince cipher with a non-uniform input. We provide the details of the cipher and its masking as well as a secure and an insecure example of a non-uniform masked input. We provide the theoretical analysis of both cases. The experimental analysis is found in Section 4.

Prince Cipher. Prince [10] is an AES-like cipher which consists of a 64-bit state divided in 4x4 rosters of nibbles and a 128-bit key. The S-box is a 4-bit cubic function and the linear layer consists of a MixColumns operation with a quasi-MDS matrix and a ShiftRows operation. The key schedule is simple where the first and last whitening keys contain the first 64-bits of the master key and the other round keys form the last 64-bits of the master key. The cipher consists of 12 nonlinear layers, where the first half applies the S-box and the second half applies the inverse S-box. The S-box and its inverse are affine equivalent.

Masking Details. Consider the threshold implementation of the Prince cipher, we choose the design presented in work by Bozilov *et al.* [11], where the S-box is decomposed using three identical quadratic functions

$$S = A_4 \circ Q_{294} \circ A_3 \circ Q_{294} \circ A_2 \circ Q_{294} \circ A_1$$

with A_1, A_2, A_3, A_4 affine layers and Q_{294} a quadratic representative of a particular affine equivalence class as described by Bilgin *et al.* [8,9]. Since the inverse S-box of Prince used in the second part of the algorithm is an affine equivalent of the regular S-box, only two additional affine layers are required to implement it. The masking of the above functions is achieved via direct sharing using three shares. Similarly, the masking of Prince's linear layer is done share-wise. Both are non-complete and uniform, and such that a register layer is placed after the only nonlinear layer, namely after Q_{294}. Since the cipher is implemented in parallel and calculates regular and inverse diffusion layers simultaneously, the chosen approach may significantly decrease the security with non-uniform inputs against noisy-probing adversary. To reduce the probability of leakage, we add two more registers in the design. One register is added before inverse diffusion to disable it, as it is unnecessary until the end of fifth round. The other register is added after ShiftRows operation. The details on the S-box and its inverse decomposition and maskings of its quadratic substitutions are given in Appendix A.

The architecture of the masking follows a round-based design. In total, the original version needs 36 cycles and does not require fresh randomness for its computation. Our modified version needs 48 clock cycles, due to the presence of new registers. The schematic of the design can be found in Figure 5 in [11]. The data path is 64×3 bits width indicating a masked state. Round constants and the key are XORed with the first share of the state.

The threshold implementation of the S-box has interesting cryptanalytic properties which we can use when evaluating the security of the cipher with non-uniform inputs. Namely, the masked S-box has a nontrivial upper bound on the maximum absolute correlation.

Lemma 1 (Correlation of the Masked Prince S-box). *Let $\bar{S} : \mathbb{V}_a \to \mathbb{V}_b$ be any restriction of the sharing of the masked Prince S-box S. Denote its absolute correlation matrix by $|C^{\bar{S}}|$. For any $u, v \in \mathbb{F}_2^\ell / \mathbb{V}^\perp$ such that $u \neq 0$, it holds that $\left|C^{\bar{S}}_{u,v}\right| \leq 2^{-1.41}$.*

The above result will be used in the analysis of non-uniform inputs.

For simplicity of the analysis, we keep the key as a constant (as a result, we do not need to consider trails including the key schedule). Meaning that for the theoretical and experimental analysis, we do not consider the influence of the key and instead only consider a non-uniform masking of the plaintext. We note that with this analysis, the security of the masked cipher including the key (given that the key is uniformly masked) is also included (with the possible effect that the masked key improves the security due to the increased entropy).

An Insecure Non-Uniform Input. We first detail the non-uniform masking of the plaintext which shows negative experimental results in PROLEAD and on FPGA as featured in Section 4. For the masking of the plaintext we use 32 random bits (versus 128 bits for a uniform three-sharing), and we re-use this randomness row-by-row. We depict this as follows

$$\begin{pmatrix} r_1 & r_2 & r_3 & r_4 \\ r_1 & r_2 & r_3 & r_4 \\ r_1 & r_2 & r_3 & r_4 \\ r_1 & r_2 & r_3 & r_4 \end{pmatrix}, \tag{1}$$

with r_i eight bits of (ideal) randomness. Namely, each cell in a column of the state is masked using the same randomness.

We analyse the effect of this non-uniform input masking using the linear cryptanalytic techniques detailed in Section 2.3. To recap, we use a trail based approach. To that end, we study the activity patterns through the masked Prince given the non-uniform input (or given the uniform input with the non-uniform encoder *Enc* from Figure 2). When we call some parts of the state *active*, we mean the non-zero linear approximations are applied to those parts. We consider (masked) cells of the cipher's state as main indicators of the activity propagation. Using the resulting trails, we find the dependency between the probed values and the initially masked input secret. Due to the way the plaintext is shared, when activating a cell masked with r_i, we have to activate at least one other cell masked with the same randomness r_i. This results in the constraint that each row either has no cells activated or at least two cells activated. We then analyse the resulting activity patterns. We stop the activity pattern when a single probe can cover the output activity. Namely, in Prince, this is when only one input of a single MixColumns function is active or when only one S-box is active.

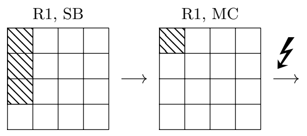


Figure 3: The trail for the threshold implementation of Prince with the non-uniform input from Eq. (1) activating at most 3 masked S-boxes in the first round. SB stands for SubCell, MC for MixColumn. The lighting indicates a single-bit probe in the active cell before it and will be omitted on the following pictures.

Considering the described approach, we can see in Figure 3 that the input of the first MixColumns function is already non-uniform. As a result, at most three masked S-boxes are in the trail (the branch number of Prince's MixColumns minus one), each with a maximum absolute correlation of $2^{-1.41}$ following Lemma 1. When probing the S-box, a glitch-extended probe can view up to 16 bits due to the parallel architecture. A probe placed after MixColumns reveals 9 bits, because due to affine layers, each glitch-extended probe can view several bits of cells in the first round. Considering the bound from Theorem 1, we find that $\varepsilon \approx 1$. As a result, the non-uniformity is so high that no relevant bound can be found. Thus, this method should be insecure and, in practice, it should leak. In Sections 4.1 and 4.2, we have implemented this case study in PROLEAD and on FPGA and observed this leakage.

A Secure Non-Uniform Input. The previous example already shows that badly chosen non-uniform inputs can lead to insecurities. We now detail a non-uniform masking (with the same entropy as the insecure example) of the plaintext which shows positive experimental results in PROLEAD and on FPGA as detailed in Sections 4.1 and 4.2.

For this example, we again need 32 random bits which are re-used now in a different, row-wise, manner. We mask the plaintext as follows

$$\begin{pmatrix} r_1 & r_1 & r_1 & r_1 \\ r_2 & r_2 & r_2 & r_2 \\ r_3 & r_3 & r_3 & r_3 \\ r_4 & r_4 & r_4 & r_4 \end{pmatrix}, \tag{2}$$

with r_i eight bits of (ideal) randomness.

Applying the same strategy as for the insecure case, we find that the trail activating the least number of masked S-boxes ends in an S-box operation of the round three and activates 12 masked S-boxes. The trail is depicted in Figure 4.

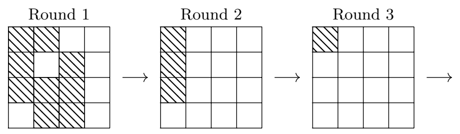


Figure 4: The best trail for the threshold implementation of Prince with the non-uniform input from Eq. (2) activating 12 masked S-boxes.

Considering the above trail, we calculate the advantage of a bounded-query noisy-probing adversary. A probe in a masked S-box views at most 16 bits, namely when propagating through the affine layers and branching, the probe returns two shares of two clock cycles. Moreover, the best trail activates 12 masked S-boxes each with a maximum absolute correlation of $2^{-1.41}$ following Lemma 1. As a result,

$$\varepsilon := \|\widehat{p}_{\mathbf{z}} - \delta_0\|_2^2 \leq |\operatorname{supp} \widehat{p}_{\mathbf{z}}| \, \|\widehat{p}_{\mathbf{z}} - \delta_0\|_\infty^2 \leq 2^{16} \, 2^{-33.84} = 2^{-17.84} \, .$$

The above calculation gives the following bound on the advantage of a noisy-probing adversary.

$$\mathrm{Adv}_{\mathsf{2\text{-}thr}}(\mathcal{A}) \leq \sqrt{\frac{q}{\lambda 2^{16.84}}} \, ,$$

where λ is the addition of noise that would be observed during practical experiments. It was mentioned in the paper of Beyne *et al.* that the noise that was observed during evaluation on an FPGA was bounded by $\lambda < 2^9$. Given that we take around 50M $\approx 2^{25}$ traces, the above bound looks to be a promising candidate to be tested. In Section 4, we provide PROLEAD and FPGA experimental results.

Given that Prince's MixColumns works on four cells at a time and since we require the input of this operation to be (close-to) uniform, using less than 32-bits of randomness to mask the input would likely lead to insecure designs. Nevertheless, a carefully chosen masking of the S-box and the input might allow for a further reduction in cost. Such an optimisation would be non-trivial and we leave it as an open problem.

3.2 Masked Midori64 with Non-Uniform Inputs

We apply the analysis on a second case study. Namely, we investigate non-uniform inputs to a threshold implementation of the Midori64 cipher. Like in the Prince example, we use no less than 32 bits to mask the plaintext because we aim to preserve the uniformity of the first MixColumns operation.

Midori64 Cipher. Midori [1] is a block cipher optimised for low-energy usage. The S-box is also used in other block ciphers including CRAFT [4] and MANTIS [3]. In this work, we specifically look at the Midori64 variant which has a 128-bit key and a 64-bit state that is split into 4-bit cells. An involutive binary quasi-MDS matrix together with a permutation of the 4-bit cells form the diffusion layer and it uses a 4-bit cubic S-box as the non-linear layer. Midori64 has a simple key schedule where each round either the left or right half of the master key is XORed to the state of the cipher.

Masking Details. To create the first-order secure masking of Midori64 we adopt the approach described in the work by Moradi *et al.* [18] with a change of the decomposition choice where we switch the quadratic class Q_{12} for the classes Q_{294} and Q_{299}. The architecture comprises only the encryption of the Midori64 and follows a pipelined structure as depicted in Figure 5.

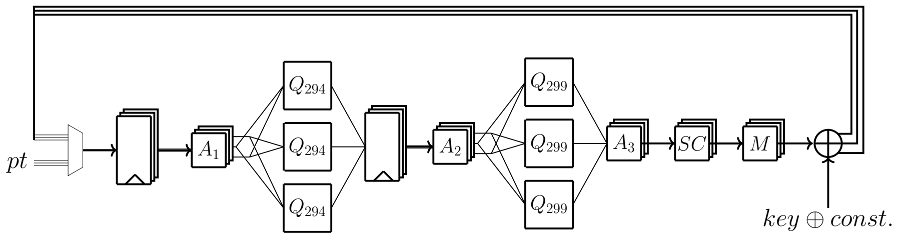


Figure 5: Midori64 encryption round-based architecture.

Midori64 S-box is affine equivalent to the cubic class C_{266} and can be decomposed into two quadratic bijections. The decomposition we choose for our experiments is

$$S = A_3 \circ Q_{299} \circ A_2 \circ Q_{294} \circ A_1$$

with the affine functions A_1, A_2, A_3 and the quadratic classes Q_{299}, Q_{294}. More details on the decomposition and its masking are given in Appendix B. The pipelined architecture implies two register stages: one placed before the S-box and another one inside of it to split the nonlinear layers. It increases the latency of the implementation; nevertheless allowing to encrypt two plaintexts at the same time. The chosen design needs 32 clock cycles to perform one encryption and does not require fresh randomness for its computation. Similar to the design of the threshold implementation of Prince, the key is considered a constant. As a result, the secure example of a non-uniform input below would also be secure in case the key is (uniformly) masked.

The threshold implementation of the S-box has the following cryptanalytic property.

Lemma 2 (Correlation of the Masked Midori64 S-box). *Let $\bar{S} : \mathbb{V}_a \to \mathbb{V}_b$ be any restriction of the masking of S defined above. Denote its absolute* *correlation matrix by $|C^{\bar{S}}|$. For any $u, v \in \mathbb{F}_2^\ell/\mathbb{V}^\perp$ such that $u \neq 0$, it holds that $|C^{\bar{S}}_{u,v}| \leq 2^{-2}$.*

An Insecure Non-Uniform Input. An insecure example is designed following the same principle as for the corresponding case in Section 3.1 for Prince. We initialise the state with 32-bits of randomness placing them as follows

$$\begin{pmatrix} r_1 & r_2 & r_3 & r_4 \\ r_2 & r_1 & r_4 & r_3 \\ r_3 & r_4 & r_1 & r_2 \\ r_4 & r_3 & r_2 & r_1 \end{pmatrix} . \tag{3}$$

With the above masking of the plaintext, an input to an S-box on the second round is non-uniform. The trail to such an S-box activates three cells with the same randomness because of the diffusion layer.

To verify whether the advantage of the noisy probing adversary is high, we use the analysis from Section 2.3. Recall from Lemma 2 that a masked S-box has a maximum absolute correlation of 2^{-2} and that we activate only three of them. A probe placed in the S-box after the first round will reveal at most 8 bits of information because of the Q_{299} quadratic layer. If the probe is placed after the MixColumn operation, it views at most $8 \cdot 3 = 24$ bits due to the quasi-MDS matrix and the absence of a register between Q_{299} and the MixColumns layer. Thus, we find that $\varepsilon \approx 1$. As a result, we expect to quickly observe leakage in this case. In Sections 4.1 and 4.2, we have implemented this case study in PROLEAD and on FPGA and observed this leakage.

A Secure Non-Uniform Input. For the secure non-uniform masking of the plaintext, we need again a total of 32 random bits and reuse the randomness over the rows the same way as it was done in Section 3.1 for the secure non-uniform input example. We represent the masking of the plaintext as a matrix

$$\begin{pmatrix} r_1 & r_1 & r_1 & r_1 \\ r_2 & r_2 & r_2 & r_2 \\ r_3 & r_3 & r_3 & r_3 \\ r_4 & r_4 & r_4 & r_4 \end{pmatrix} , \tag{4}$$

with r_i eight bits of randomness.

We analyse activity patterns for Midori64 which end in a single S-box or a single active column and which start with the constraint that a row either has no activations or at least two active cells. The best trail with these constraints activates at least 12 masked S-boxes and is depicted in Figure 6.

As we already mentioned for the insecure case, a probe placed after Mix-Column can observe up to 24 wires and the masked S-box has a maximum absolute correlation of 2^{-2}. From the above, we find that

$$\varepsilon := \|\widehat{p}_{\mathbf{z}} - \delta_0\|_2^2 \leq |\operatorname{supp} \widehat{p}_{\mathbf{z}}| \, \|\widehat{p}_{\mathbf{z}} - \delta_0\|_\infty^2 \leq 2^{24}\, 2^{-48} = 2^{-24} ,$$

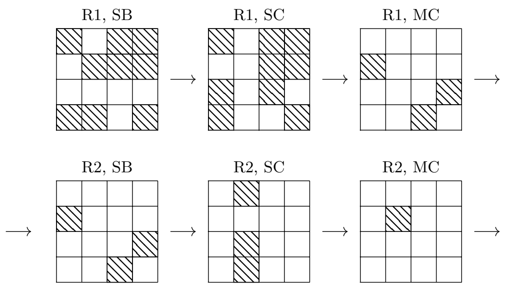


Figure 6: An example of the best trail for the masked Midori64 with a non-uniform input activating 12 masked S-boxes in 2 rounds (denoted R1 and R2). SB stands for SubCell, SC for ShuffleCell, and MC for MixColumn.

which provides the following advantage for a non-uniform input masking

$$\mathrm{Adv}_{\mathsf{2\text{-}thr}}(\mathcal{A}) \leq \sqrt{\frac{q}{\lambda 2^{23}}}\,.$$

Similar to Prince's secure case, the above advantage shows the masking is a promising candidate for practical verification. Its results are found in Section 4.

We also tested another way to re-use the same amount of initial random bits (32-bits), namely a column-wise masking similar to the masking in Eq. (1) of Prince (which was an insecure example). The case of a column-wise masking for Midori64 provides the same security bound as for the row-wise masking which is detailed above. This case was also tested in PROLEAD and lead to a secure result. We provide these test results in Appendix C.4.

4 Practical Evaluation and Efficiency

In this section, we provide experimental analyses of the proposed security claims for both the Prince cipher from Section 3.1 and the Midori64 cipher from Section 3.2. The section is divided into three parts: a noiseless verification is done using the PROLEAD tool from Müller *et al.* [19] in Section 4.1, a more realistic noisy result is obtained from FPGA experiments in Section 4.2, and an efficiency analysis is done in Section 4.3.

4.1 PROLEAD Experiments

PROLEAD is a recently developed automated tool which allows to analyse the statistical independence of simulated intermediates probed by a robust-probing adversary following the definition by Faust *et al.* [12]. Among its benefits are the abilities to handle full masked cipher implementations and to detect flaws related to the occurrence of glitches. The tool does not require any power model as it processes only gate-level netlists.

We use PROLEAD to check the correctness of our assumptions about the threshold-probing adversary advantage claimed in Sections 3.1 and 3.2 following the analysis detailed in Section 2.3. Since the tool does not consider noise during statistical evaluation, we omit the noise parameter λ used in the Theorem 1 (we set $\lambda = 1$). To evaluate any leakage, PROLEAD uses a likelihood ratio G-test of independence [25] as a statistical hypothesis test based on multinomial distributions. It tests the goodness of fit of observed frequencies to their expected frequencies. The method is applied to contingency tables containing the distributions for all glitch-extended probing sets generated for the design. The sizes of those probing sets depend on the glitch-extended probing model which can reveal many bits under a single probe making the tables significantly large. For example, the maximum number of probes per largest set is 32 and 39[2] for the Midori64 and Prince designs, respectively.

PROLEAD operates in two modes, namely a normal mode and a compact mode. The difference lies in the contingency table calculation. In normal mode, the columns of the distribution tables are calculated using the concatenated values of all probed bits from the particular set that are called *labels*. Each label represents an entry of a contingency table where the frequency of the label's appearance per group is stored. Thus, it is possible to acquire the tables with up to 2^{n_p} columns, where n_p equals the number of wires within one probe in the glitch-extended probing set. In compact mode, the labels for the distribution tables are provided based on the Hamming weight of a probing set which allows for a more computation friendly verification at the cost of accuracy. To summarise, the normal mode provides more accurate information on the observed leakage, whereas it is possible to conduct more experiments for larger inputs in the compact mode.

We begin the evaluation process with checking our implementations in a compact mode to verify their security against first-order glitch-extended probing adversaries when uniform randomness is used. We perform up to 100M simulations. The results for those and the following experiments are shown in Table 1. We find that the implementations with uniform inputs do not leak during the simulations as would be expected. Excerpts with the results from the PROLEAD reports can be found in Appendix C.

Then, we use PROLEAD to evaluate leakage for the same designs with non-uniform inputs. Since the tool is only designed to create uniform maskings of the input, we add an intermediate module to our design, that is excluded from the probing list, to reuse generated randomness in composition of the non-uniform shares. For the cases with "insecure" inputs, the tool shows immediate[3] leakage significantly exceeding the threshold for the statistical tests in both modes which

[2] The size of the largest set may include control logic wires of the algorithm (like counter, start, and select). These bits do not contribute to the advantage of the noisy threshold-probing adversary.

[3] Simulations in PROLEAD are split into several iterations with a chosen step. Here, we set a step of 1M and 128k traces for the compact and normal modes, respectively.

is in line with the theoretical analyses in Section 3. Moreover, the leakage is observed only in some particular rounds of the ciphers (see Table 1).

Finally, we test the "secure" non-uniform designs. Again, we observe growing leakage during the simulations. However, this time the threshold is achieved only after significantly more experiments, and the leakage is detected during the later rounds in accordance with our trails from Section 3. We pay closer attention to the tests in normal mode, since those are more related to a threshold-probing adversary model, and see the leakage again. This time the number of traces needed to conclude the insecurity of a design is closely related to the bounds we proposed considering the absence of noise ($\lambda = 1$).

Table 1: Results of PROLEAD experiments of the Prince masking detailed in Section 3.1 and of the Midori64 cipher from Section 3.2. We detail after how many experiments PROLEAD finds leakage and in which clock cycles the leakage occurs (including two cycles to initialise the cipher) and in which round of the cipher the leakage occurs.

Cipher	Case	Mode	Passed	#Traces	#Cycle	#Round
Prince	Uniform	compact	✓	100M	NA	NA
	"Insecure" Non-Uniform	compact	✗	1M	4,...,10	1,2,3
		normal	✗	128k	4,...,10	1,2,3
	"Secure" Non-Uniform	compact	✗	48M	10	3
		normal	✗	3.8M	10	3
Midori64	Uniform	compact	✓	100M	NA	NA
	"Insecure" Non-Uniform	compact	✗	1M	5,6,7	2,3
		normal	✗	128k	5,6,7	2,3
	"Secure" Non-Uniform	compact	✗	2M	7	3
		normal	✗	6.4M	8	3

4.2 FPGA Experiments

For the practical experiments, we used a Xilinx Spartan-6 FPGA on a SAKURA-G evaluation board [14] and supplied the device with a stable 6.144 MHz clock. For the randomness generation, we focus on providing comprehensive results whether randomness can be re-used when masking the input of the cipher. As a result, we use a heavy cryptographic random number generator, namely one based on AES-128, to ensure the limited randomness given to the threshold implementation is of good quality such that it does not bias our results.

We collect power consumption traces by monitoring the voltage drop over a 1Ω shunt resistor placed on the VDD path of the target FPGA and using a digital oscilloscope at a sampling rate of 500 MS/s. For the measurement's analysis, we follow TVLA requirements [2] to conduct first-order fixed vs. random plaintext t-tests. The encryption is performed up to 50M times receiving either masked fixed or masked random plaintexts. This technique allows to detect the first-order side-channel leakage without implementing an actual key-recovery attack.

We perform an analysis for each example introduced in Section 3. To do the first-order t-test, we collect 4000 and 3000 sample points (the number of points that comprise a complete waveform record, determined by the amount of data that can be captured by an oscilloscope.) for the Prince and Midori64 implementations, respectively. The corresponding results are shown in Figures 7 and 8 including the absolute t-value changes through the number of traces sampled.

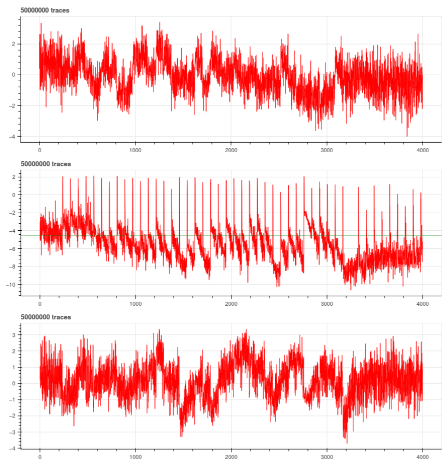
From Figure 7, we observe that the threshold implementation of Prince with a uniform input is secure (as expected). As a sanity check, we also evaluated the implementation with the random number generator turned off to find that the implementation leaks (again, as expected). The "insecure" non-uniform input leaks almost immediately and we see a spike of leakage each cycle of the implementation. However, the "secure" non-uniform input does not leak even at 50M traces. Together with the theoretical analysis and the PROLEAD analysis, we can conclude that it is viable to use the threshold implementation of Prince with such a non-uniform input.

From Figure 8, we observe similar results for the threshold implementation of the Midori64 cipher. It is interesting to note that the "insecure" non-uniform input only leaks in the second round of the cipher, but then becomes secure. This can be explained from the strong cryptanalytic properties of its diffusion layer and its masked S-box. As the computation of the cipher continues, the non-uniform randomness is processed more-and-more via a good cryptographic function making it less-and-less distinguishable from uniform random. We note that Prince's behaviour with the "insecure" non-uniform input should be the same, but that due to the multiplexers in the architecture the same leakage from the first round is repeated each cycle. Again, the "secure" non-uniform input shows no leakage in the t-test showing that it is possible to work with the threshold implementation without giving a full entropy input.

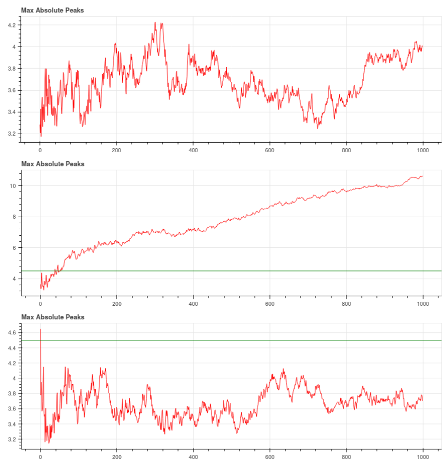
4.3 Efficiency Comparison

We quickly provide the efficiency measures of the threshold implementations of the Prince and Midori64 ciphers from Sections 3.1 and 3.2 though we note that we worked with tried-and-tested threshold implementations and that their efficiency was not the goal of the work. Table 2 provides the cost in terms of area, latency, and total random bits all designs use. We note that while we report on the randomness cost in bits, we have not investigated the related cost of the RNG which generates this randomness.

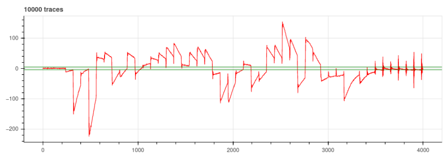
For the Midori64 case, we find that the non-uniform input allows a four times reduction of the total randomness cost without requiring a trade-off in area or latency. However, the Prince masking requires an extra register layer in order to have a sufficient theoretical bound. We have performed leakage tests with a



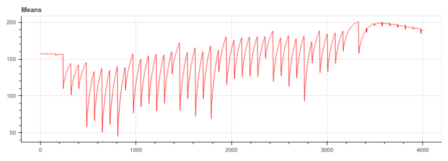
(a) **T-test**. **Top**: Uniform Rand.; **Middle**: "Insecure" Non-Uniform Rand.; **Bottom**: "Secure" Non-Uniform Rand.



(b) **T-value**. **Top**: Uniform Rand.; **Middle**: "Insecure" Non-Uniform Rand.; **Bottom**: "Secure" Non-Uniform Rand.

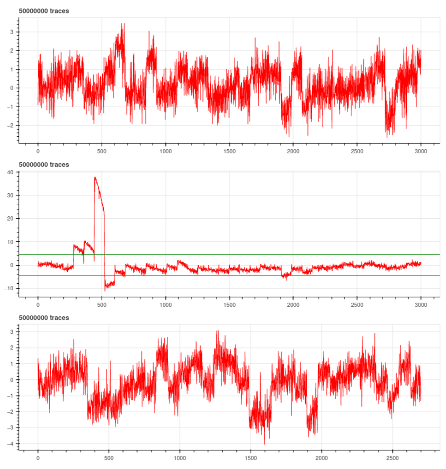


(c) **T-test, masks off**

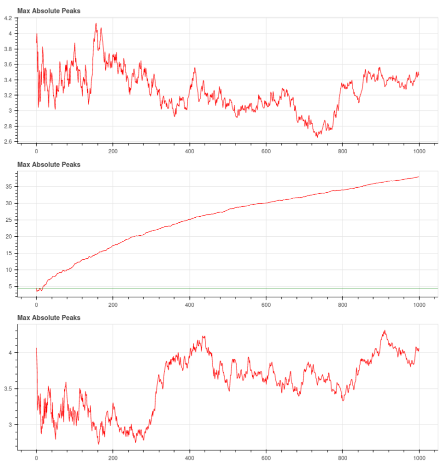


(d) **Sample trace**

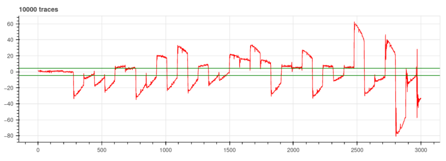
Figure 7: Masked Prince implementation with 50M traces where we test the implementation with uniform randomness and the non-uniform case studies from Section 3.1. We show sample traces with masks off and masks on and we show the final t-test together with the maximum absolute t-test value evolution over the number of traces.



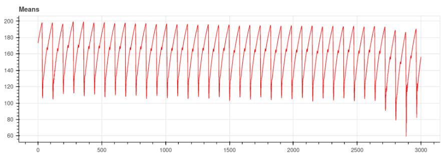
(a) **T-test**. **Top**: Uniform Rand.; **Middle**: “Insecure” Non-Uniform Rand.; **Bottom**: “Secure” Non-Uniform Rand.



(b) **T-value**. **Top**: Uniform Rand.; **Middle**: “Insecure” Non-Uniform Rand.; **Bottom**: “Secure” Non-Uniform Rand.



(c) **T-test, masks off**



(d) **Sample trace**

Figure 8: Masked Midori64 implementation with 50M traces where we test the implementation with uniform randomness and the case studies from Section 3.2. We show sample traces with masks off and masks on and we show the final t-test together with the maximum absolute t-test value evolution over the number of traces.

non-uniform input without this extra layer and have seen secure results. Thus, we believe the cheaper Prince masking with the non-uniform input can still be used in practice and that the discrepancy comes from the theoretical framework's overestimation on glitches, similar to the observations from [6] on glitches in a masked AES S-box.

Table 2: The efficiency measures in area, latency, and total randomness cost of both the Prince and Midori64 threshold implementations using uniform or non-uniform randomness with the `NANGATE` 45nm Open Cell Library [20].

Cipher	#Shares	Area (GE)	Latency (Cycles)	Rand. (Bits)
Prince	3	8353	36	128
		11050	48	32
Midori64	3	7324	32	128
		7324	32	32

5 Conclusion

In this paper, we have shown that using a non-uniform masked input for a threshold implementation can remain first-order secure in face of a practical evaluation. We have also shown that the non-uniform masking itself should be chosen carefully and differently for each symmetric primitive following the principles from this paper's security framework based on linear cryptanalysis and the noisy probing model.

We presented secure and insecure examples of non-uniform initial maskings for established threshold implementations of the Prince and Midori64 ciphers. We verified the examples using the PROLEAD tool and in practice by implementing them on FPGA. The vulnerabilities of the insecure examples were quickly detected and its leakage coincided with the provided bounds from the theoretical analyses. The secure examples did not show any leakage up to 50 million traces which shows that we can securely reduce the entropy of the initial masking four times over.

While the scope of this paper was limited to first-order secure implementations, we pose the open problem of continuing the investigation of the security of higher-order threshold implementations with a non-uniform masked input. Moreover, we pose the important problem of creating tools to automate this paper's security analysis such that more complex examples, like using LFSRs to generate the initial masking, can be investigated.

Acknowledgements. We thank Tim Beyne for the interesting discussions. This work was supported by CyberSecurity Research Flanders with reference number VR20192203.

References

1. Banik, S., Bogdanov, A., Isobe, T., Shibutani, K., Hiwatari, H., Akishita, T., Regazzoni, F.: Midori: A block cipher for low energy. In: ASIACRYPT (2). Lecture Notes in Computer Science, vol. 9453, pp. 411–436. Springer (2015)
2. Becker, G.T., Cooper, J., DeMulder, E.K., Goodwill, G., Jaffe, J., Kenworthy, G., Kouzminov, T., Leiserson, A.J., Marson, M.E., Rohatgi, P., Saab, S.: Test vector leakage assessment (tvla) methodology in practice (2013)
3. Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: The SKINNY family of block ciphers and its low-latency variant MANTIS. In: CRYPTO (2). Lecture Notes in Computer Science, vol. 9815, pp. 123–153. Springer (2016)
4. Beierle, C., Leander, G., Moradi, A., Rasoolzadeh, S.: CRAFT: lightweight tweakable block cipher with efficient protection against DFA attacks. IACR Trans. Symmetric Cryptol. **2019**(1), 5–45 (2019)
5. Beyne, T., Dhooghe, S., Moradi, A., Shahmirzadi, A.R.: Cryptanalysis of efficient masked ciphers: Applications to low latency. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2022**(1), 679–721 (2022)
6. Beyne, T., Dhooghe, S., Ranea, A., Sijacic, D.: A low-randomness second-order masked AES. In: AlTawy, R., Hülsing, A. (eds.) Selected Areas in Cryptography - 28th International Conference, SAC 2021, Virtual Event, September 29 - October 1, 2021, Revised Selected Papers. Lecture Notes in Computer Science, vol. 13203, pp. 87–110. Springer (2021). https://doi.org/10.1007/978-3-030-99277-4_5, https://doi.org/10.1007/978-3-030-99277-4_5
7. Beyne, T., Dhooghe, S., Zhang, Z.: Cryptanalysis of masked ciphers: A not so random idea. In: Moriai, S., Wang, H. (eds.) Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12491, pp. 817–850. Springer, Berlin (2020). https://doi.org/10.1007/978-3-030-64837-4_27, https://doi.org/10.1007/978-3-030-64837-4_27
8. Bilgin, B., Nikova, S., Nikov, V., Rijmen, V., Stütz, G.: Threshold implementations of all 3 ×3 and 4 ×4 s-boxes. In: Prouff, E., Schaumont, P. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7428, pp. 76–91. Springer (2012). https://doi.org/10.1007/978-3-642-33027-8_5, https://doi.org/10.1007/978-3-642-33027-8_5
9. Bilgin, B., Nikova, S., Nikov, V., Rijmen, V., Stütz, G.: Threshold implementations of all 3x3 and 4x4 s-boxes. IACR Cryptol. ePrint Arch. p. 300 (2012), http://eprint.iacr.org/2012/300
10. Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E.B., Knezevic, M., Knudsen, L.R., Leander, G., Nikov, V., Paar, C., Rechberger, C., Rombouts, P., Thomsen, S.S., Yalçin, T.: PRINCE - A low-latency block cipher for pervasive computing applications - extended abstract. In: ASIACRYPT. Lecture Notes in Computer Science, vol. 7658, pp. 208–225. Springer (2012)
11. Bozilov, D., Knezevic, M., Nikov, V.: Optimized threshold implementations: securing cryptographic accelerators for low-energy and low-latency applications. J. Cryptogr. Eng. **12**(1), 15–51 (2022). https://doi.org/10.1007/s13389-021-00276-5, https://doi.org/10.1007/s13389-021-00276-5

12. Faust, S., Grosso, V., Pozo, S.M.D., Paglialonga, C., Standaert, F.: Composable masking schemes in the presence of physical defaults and the robust probing model. IACR Cryptol. ePrint Arch. p. 711 (2017), http://eprint.iacr.org/2017/711
13. Faust, S., Grosso, V., Pozo, S.M.D., Paglialonga, C., Standaert, F.: Composable masking schemes in the presence of physical defaults & the robust probing model. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2018**(3), 89–120 (2018)
14. Guntur, H., Ishii, J., Satoh, A.: Side-channel attack user reference architecture board SAKURA-G. In: IEEE 3rd Global Conference on Consumer Electronics, GCCE 2014, Tokyo, Japan, 7-10 October 2014. pp. 271–274. IEEE (2014). https://doi.org/10.1109/GCCE.2014.7031104, https://doi.org/10.1109/GCCE.2014.7031104
15. Ishai, Y., Sahai, A., Wagner, D.: Private circuits: Securing hardware against probing attacks. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 463–481. Springer, Heidelberg (Aug 2003). https://doi.org/10.1007/978-3-540-45146-4˙27
16. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M.J. (ed.) Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings. Lecture Notes in Computer Science, vol. 1666, pp. 388–397. Springer (1999). https://doi.org/10.1007/3-540-48405-1_25, https://doi.org/10.1007/3-540-48405-1_25
17. Matsui, M.: Linear cryptanalysis method for DES cipher. In: EUROCRYPT '93. Lecture Notes in Computer Science, vol. 765, pp. 386–397. Springer (1993)
18. Moradi, A., Schneider, T.: Side-channel analysis protection and low-latency in action - - case study of PRINCE and midori -. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10031, pp. 517–547 (2016). https://doi.org/10.1007/978-3-662-53887-6_19, https://doi.org/10.1007/978-3-662-53887-6_19
19. Müller, N., Moradi, A.: PROLEAD A probing-based hardware leakage detection tool. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2022**(4), 311–348 (2022). https://doi.org/10.46586/tches.v2022.i4.311-348, https://doi.org/10.46586/tches.v2022.i4.311-348
20. NANGATE: The NanGate 45nm Open Cell Library, version: PDKv1.3_v2010_12.Apache.CCL. Available at https://github.com/The-OpenROAD-Project/OpenROAD-flow-scripts/tree/master/flow/platforms/nangate45
21. Nikova, S., Rechberger, C., Rijmen, V.: Threshold implementations against side-channel attacks and glitches. In: Ning, P., Qing, S., Li, N. (eds.) Information and Communications Security, 8th International Conference, ICICS 2006, Raleigh, NC, USA, December 4-7, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4307, pp. 529–545. Springer (2006). https://doi.org/10.1007/11935308_38, https://doi.org/10.1007/11935308_38
22. Nikova, S., Rechberger, C., Rijmen, V.: Threshold implementations against side-channel attacks and glitches. In: Ning, P., Qing, S., Li, N. (eds.) ICICS 06. LNCS, vol. 4307, pp. 529–545. Springer, Heidelberg (Dec 2006)
23. Shahmirzadi, A.R., Moradi, A.: Re-consolidating first-order masking schemes nullifying fresh randomness. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2021**(1), 305–342 (2021). https://doi.org/10.46586/tches.v2021.i1.305-342, https://doi.org/10.46586/tches.v2021.i1.305-342

24. Shahmirzadi, A.R., Moradi, A.: Second-order SCA security with almost no fresh randomness. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2021**(3), 708–755 (2021). https://doi.org/10.46586/tches.v2021.i3.708-755, https://doi.org/10.46586/tches.v2021.i3.708-755
25. Sokal, R., Rohlf, F.: Biometry: The Principles and Practice of Statistics in Biological Research. W. H. Freeman (1981), https://books.google.be/books?id=C-OTQgAACAAJ
26. Tardy-Corfdir, A., Gilbert, H.: A known plaintext attack of FEAL-4 and FEAL-6. In: Feigenbaum, J. (ed.) CRYPTO '91. Lecture Notes in Computer Science, vol. 576, pp. 172–181. Springer (1991)

A Prince S-box Masking

The Prince S-box has its lookup table $[B, F, 3, 2, A, C, 9, 1, 6, 7, 8, 0, E, 5, D, 4]$ and is decomposed as follows

$$S = A_1 \circ Q_{294} \circ A_2 \circ Q_{294} \circ A_3 \circ Q_{294} \circ A_4 \,.$$

The inverse S-box has its lookup table $[B, 7, 3, 2, F, D, 8, 9, A, 6, 4, 0, 5, E, C, 1]$ and is decomposed as

$$S^{-1} = A_5 \circ Q_{294} \circ A_2 \circ Q_{294} \circ A_3 \circ Q_{294} \circ A_6 \,.$$

The inputs of the following 4-bit functions are designated with a nibble (x, y, z, w), where x is the most significant bit and w s the least significant bit.

$$\begin{aligned}
A_1 &= [1 + x + z; 1 + y; z + w; z] \\
A_2 &= [x + z + w; 1 + x + z; 1 + y + z + w; x + y + z + w] \\
A_3 &= [w; z; y; x] \\
A_4 &= [1 + x + y + z + w; x; 1 + x + z + w; w] \\
A_5 &= [1 + y; x + z; 1 + y + w; 1 + z + w] \\
A_6 &= [1 + x + w; y; w; 1 + z] \,,
\end{aligned}$$

with

$$\begin{aligned}
A_1 &= [C, E, 7, 5, 8, A, 3, 1, 4, 6, F, D, 0, 2, B, 9] \\
A_2 &= [6, D, 9, 2, 5, E, A, 1, B, 0, 4, F, 8, 3, 7, C] \\
A_3 &= [0, 8, 4, C, 2, A, 6, E, 1, 9, 5, D, 3, B, 7, F] \\
A_4 &= [A, 1, 0, B, 2, 9, 8, 3, 4, F, E, 5, C, 7, 6, D] \\
A_5 &= [B, 8, E, D, 1, 2, 4, 7, F, C, A, 9, 5, 6, 0, 3] \\
A_6 &= [9, 3, 8, 2, D, 7, C, 6, 1, B, 0, A, 5, F, 4, E] \,.
\end{aligned}$$

The above affine functions are masked share-by-share. The quadratic layer $Q_{294} = [a, b, c, d] = [x, y, z + xy, w + xz]$ is masked as follows

$$\begin{aligned}
a^i &= x^i \\
b^i &= y^i \\
c^i &= z^i + x^i y^i + x^i y^{i+1} + x^{i+1} y^i \\
d^i &= w^i + x^i z^i + x^i z^{i+1} + x^{i+1} z^i \,,
\end{aligned}$$

for the shares $i \in \{0, 1, 2\}$, where the convention is used that superscripts wrap around at two. The above masking is uniform and non-complete.

B Midori64 S-box Masking

The Midori64 S-box has its lookup table $[C, A, D, 3, E, B, F, 7, 8, 9, 1, 5, 0, 2, 4, 6]$ and is decomposed as

$$S = A_1 \circ Q_{299} \circ A_2 \circ Q_{294} \circ A_3 \,.$$

The inputs of the following 4-bit functions are designated with a nibble (x, y, z, w), where x is the most significant bit and w s the least significant bit.

$$\begin{aligned}
A_1 &= [1 + x + y + z; 1 + x + y + w; 1 + x + y + z + w; y + w] \\
A_2 &= [w; x; y; z] \\
A_3 &= [1 + y + w; 1 + y + z + w; w; x + z + w] \,,
\end{aligned}$$

with

$$\begin{aligned}
A_1 &= [E, 9, 4, 3, 1, 6, B, C, 0, 7, A, D, F, 8, 5, 2] \\
A_2 &= [0, 8, 1, 9, 2, A, 3, B, 4, C, 5, D, 6, E, 7, F] \\
A_3 &= [C, 3, 9, 6, 0, F, 5, A, D, 2, 8, 7, 1, E, 4, B] \,.
\end{aligned}$$

The above affine functions are masked share-by-share. The quadratic layer $Q_{294} = [a, b, c, d] = [x, y, z + xy, w + xz]$ is masked as follows

$$\begin{aligned}
a^{i-1} &= x^i \\
b^{i-1} &= y^i \\
c^{i-1} &= z^i + x^i y^i + x^i y^{i+1} + x^{i+1} y^i \\
d^{i-1} &= w^i + x^i z^i + x^i z^{i+1} + x^{i+1} z^i \,,
\end{aligned}$$

and the quadratic layer $Q_{299} = [a, b, c, d] = [x, y + xy + xz, z + xy + xz + xw, w + xy + xw]$ is masked as

$$\begin{aligned}
a^{i-1} &= x^i \\
b^{i-1} &= y^i + (x^i y^i + x^i y^{i+1} + x^{i+1} y^i) + (x^i z^i + x^i z^{i+1} + x^{i+1} z^i) \\
c^{i-1} &= z^i + (x^i y^i + x^i y^{i+1} + x^{i+1} y^i) + (x^i z^i + x^i z^{i+1} + x^{i+1} z^i) \\
&\quad + (x^i w^i + x^i w^{i+1} + x^{i+1} w^i) \\
d^{i-1} &= w^i + (x^i y^i + x^i y^{i+1} + x^{i+1} y^i) + (x^i w^i + x^i w^{i+1} + x^{i+1} w^i) \,.
\end{aligned}$$

for the shares $i \in \{0, 1, 2\}$, where the convention is used that superscripts wrap around at two. The above maskings are uniform and non-complete.

C PROLEAD Experiments

The PROLEAD tool outputs data in two formats: tables with an overview of the current status of the evaluation process and reports for each simulation step. We further provide shortened versions of the tables for our experiments and add them with the overview of some probing sets from the reports. In the following tables, we will write probe sets (such as _00480_[35] (7)). Such sets include several register values from the implementation together with the cycle of the computation (in this example seven).

C.1 Designs with Uniform randomness

Table 3: **Evaluation info**: Uniform randomness.

Cipher	#Standard Probes	#Extended Probes	Security Order	#Probing Sets	Maximum #Probes per Set
Prince	17600	20880	1	9080	101
Midori64	50880	29120	1	14480	32

Table 4: **Evaluation results**: Prince, uniform case.

Elapsed Time (s)	Required RAM (GB)	Processed Simulations	Probe Set with Highest Information Leakage	-log10(p)	Status
101	26.23	1000000	_03951_[61] (18)	4.887129	OKAY
202	26.23	2000000	_04211_[21] (22)	7.141349	LEAKAGE
304	26.23	3000000	_04081_[32] (6)	5.383576	LEAKAGE
406	26.23	4000000	_04211_[39] (27)	3.671223	OKAY
9655	26.23	99000000	_03951_[53] (40)	4.154447	OKAY
9751	26.23	100000000	_04081_[19] (1)	4.118873	OKAY

Table 5: **Evaluation results**: Midori64, uniform case.

Elapsed Time (s)	Required RAM (GB)	Processed Simulations	Probe Set with Highest Information Leakage	-log10(p)	Status
89	33.32	1000000	Sbox[15].register.in[5] (20)	4.004151	OKAY
179	33.32	2000000	ciphertext_1[63] (9)	2.891629	OKAY
269	33.32	3000000	Sbox[7].register.in[1] (4)	3.397437	OKAY
8560	33.32	99000000	_00226_[19] (1)	4.151201	OKAY
8646	33.32	100000000	_00226_[19] (1)	4.091332	OKAY

C.2 Midori64, Non-Uniform randomness

Table 6: **Evaluation info**: Midori64, non-uniform cases.

Case	Mode	#Stand. Probes	#Extend. Probes	#Probe Sets	Maximum #Probes per Set
“Insecure”	compact	43200	29100	14480	32
	normal	10800	7280	3620	
“Secure”	compact	43200	29120	14480	
	normal	10800	7280	3620	

Table 7: **Evaluation results**: Midori64, “insecure”, normal mode.

Elapsed Time (s)	Required RAM (GB)	Processed Simulations	Probe Set with Highest Information Leakage	-log10(p)	Status
34	8.24	128000 / 438057	_00208_[63] (5)	inf	LEAKAGE
457	45.70	1280000 / 1346575	_00208_[63] (5)	inf	LEAKAGE

Table 8: **Evaluation results**: Midori64, “secure”, normal mode.

Elapsed Time (s)	Required RAM (GB)	Processed Simulations	Probe Set with Highest Information Leakage	-log10(p)	Status
33	9.21	128000 / 438083	Sbox[7].register.in[5] (6)	3.912678	OKAY
73	13.74	256000 / 615365	Sbox[14].register.in[5] (10)	3.354810	OKAY
4318	273.38	6272000 / 2767077	Sbox[9].register.in[4] (8)	4.880834	OKAY
4473	282.42	6400000 / 2790153	Sbox[9].register.in[4] (8)	5.094253	LEAKAGE

C.3 Prince, Non-Uniform randomness

Table 9: **Evaluation info**: Prince, non-uniform cases.

Case	Mode	#Stand. Probes	#Extend. Probes	#Probe Sets	Maximum #Probes per Set
"Insecure"	compact	16500	14300	8380	39
	normal	7425	6435	3771	
"Secure"	compact	16500	14300	8380	
	normal	7425	6435	3771	

Table 10: **Evaluation results**: Prince, "insecure", normal mode.

Elapsed Time (s)	Required RAM (GB)	Processed Simulations	Probe Set with Highest Information Leakage	-log10(p)	Status
46	10.26	128000 / 293647	beta_reg_3.in[22](4)	inf	LEAKAGE
276	45.70	768000 / 316035	beta_reg_3.in[22](4)	inf	LEAKAGE

Table 11: **Evaluation results**: Prince, "secure", normal mode.

Elapsed Time (s)	Required RAM (GB)	Processed Simulations	Probe Set with Highest Information Leakage	-log10(p)	Status
48	11.05	128000 / 293811	_05111_ (4)	3.485201	OKAY
95	11.79	256000 / 313129	beta_reg_2.in[53] (6)	3.070294	OKAY
1385	11.99	3968000 / 316035	_04193_[16] (10)	5.983958	LEAKAGE
1431	11.99	4096000 / 316035	_04193_[17] (10)	5.668399	LEAKAGE

C.4 Midori64, "Secure" Non-Uniform case, Column-wise

Table 12: **Evaluation info**: Midori64, column-wise.

Case	Mode	#Stand. Probes	#Extend. Probes	#Probe Sets	Maximum #Probes per Set
Column-wise	compact	43200	29120	14480	32
	normal	6480	4368	2172	

Table 13: **Evaluation results**: Midori64, column-wise, compact mode.

Elapsed Time (s)	Required RAM (GB)	Processed Simulations	Probe Set with Highest Information Leakage	-log10(p)	Status
94	33.34	1000000	ciphertext_1[35] (27)	4.696523	OKAY
187	33.34	2000000	Sbox[3].register.in[1] (20)	3.945218	OKAY
9222	33.34	101000000	Sbox[7].register.in[9] (8)	4.789482	OKAY
9312	33.34	102000000	Sbox[7].register.in[9] (8)	5.164452	LEAKAGE
13539	33.34	150000000	_00480_[35] (7)	8.223624	LEAKAGE

Table 14: **Evaluation results**: Midori64, column-wise, normal mode.

Elapsed Time (s)	Required RAM (GB)	Processed Simulations	Probe Set with Highest Information Leakage	-log10(p)	Status
21	7.71	128000 / 438075	Sbox[9].register.in[1] (8)	2.371509	OKAY
46	10.60	256000 / 615407	_00480_[14] (3)	2.493538	OKAY
6551	238.73	9984000 / 3316723	_00220_[61] (3)	3.992771	OKAY
6712	239.51	10112000 / 3332201	_00220_[61] (3)	3.959946	OKAY