LFSRs and Boolean Masking: An In-depth Security Analysis

Anna Guinet[1], Jan Schoone[2,3], Niklas Höher[1], Dina Hesse[1] and Tim Güneysu[1]

[1] Ruhr University Bochum, Bochum, Germany, firstname.lastname@rub.de
[2] LIACS, Leiden University, Leiden, the Netherlands
[3] Radboud University, Nijmegen, the Netherlands, jan.schoone@proton.me

Abstract. Masking is a widely adopted countermeasure to protect cryptographic implementations from side-channel attacks. Subsequent research has focused on designing masking schemes and formally proving their security, notably through the development of automated tools, within models abstracting the reality of a side-channel analysis. These designs rely on an external source of randomness; however, there is currently no consensus on the choice of (pseudo-)random number generators for masking. To the best of our knowledge, existing formal proofs for masking security do not consider particular choices of random number generators, but rather assume that they yield uniformly distributed and independent random variables. In that context, we introduce the first verification framework that jointly analyzes a pseudo-random number generator— specifically, but not limited to, a linear feedback shift register—and a masking scheme, in the d-probing model. Our framework relies on the Walsh-Hadamard transform by drawing on techniques from linear cryptanalysis, which we extend to the robust probing model. We demonstrate our method on 4-bit and 8-bit S-boxes, provide a detailed analysis of the formal verification outcomes, and corroborate the findings with practical evaluations on an FPGA.

Keywords: probing model · masking · PRNG · LFSR · Walsh-Hadamard transform

1 Introduction

One of the most popular techniques against Side-Channel Analysis (SCA), such as power analysis [KJJ99], is *masking* [CJRR99, GP99]. Its core principle is to split every sensitive variable of a cryptographic algorithm into multiple *shares*, such that only their combination reveals the original variable. To formally prove the security of masked schemes against SCA, abstract models are used to define the capabilities of a side-channel adversary. In this context, a common model is the d-probing model [ISW03], where an adversary can observe up to d circuit signals, or *probes*, to gain some knowledge about some secret variables. However, concerns have been raised about its accuracy in capturing low-level physical effects of hardware implementations, which motivated the introduction of the robust d-probing model [FGM+18]. Both models have attracted a lot of attention of the research community, leading to the development of automated tools that verify the security of masking schemes within these models, e.g., [BGI+18, BBC+19, BMRT22, RBFSG22, MM22, BFG+25].

Essentially, masking schemes rely on a fundamental assumption: using an ideal external source of randomness to create the shares of the secret variables and refresh intermediate computations within the scheme. In formal security verification (through manual proofs or automated tools), those random bits are simply assumed to be independent and identically distributed. Unfortunately, randomness generation substantially raises the cost of implementing the masking countermeasure and comes with practical limitations. That is

why part of the research community focuses on designing low-randomness masking schemes [Sug18, SM21, BDRS22, SDM23, GKM+24, JMB25, JSB25] and randomness generators for masking [PYR+16, CGZ20]. For common applications, a True Random Number Generator (TRNG) is generally considered too costly for masking; as a result Pseudo-Random Number Generators (PRNGs) (or hybrid schemes where PRNGs are seeded with TRNGs) are usually preferred. For PRNGs, recent work lists the various primitives used in the literature in [UHIM24, CMM+24]. Among them, Linear Feedback Shift Registers (LFSRs) seem like viable and secure candidates at first, since their hardware cost is rather low [CMM+24] and they have desirable statistical properties [Gol67]. Hence, LFSRs have been used by several research studies to assess the security of their proposed masking schemes in practice, by using a 31-bit [DMW18, SM21, KSM22, KM22a, DSM22] or a 64-bit [KM22a] LFSR. However, to date, no security verification has formally investigated in detail the joint effect of using a specific PRNG in a Boolean masking scheme.

With respect to LFSRs, Cassiers et *al.* [CMM+24] recently warn against their use in general masking contexts without "specific and carefully crafted case studies", and argue that "the randomness generation [from LFSRs] and the masking scheme must be analyzed jointly" because the intrinsic linear structure of LFSRs might cause their outputs to cancel in $\mathbb{F}_2$. To this end, we propose a formal approach to analyzing PRNG-based randomness in masked implementations, with a focus on LFSRs. Our method is designed to exploit linear structures of Boolean functions and detect variable cancellations, and it can be applied to any source of randomness as long as its algebraic expression is known.

The idea of exploiting linear structures of functions carried by the probes, for evaluating the side-channel resistance of a scheme, underlies other independently proposed frameworks. However, they do not jointly account for the PRNG used and, thus, analyze functions under another variable parameterization than in this work and do not consider PRNG variable cancellations. This is the case for the robust d-probing security verification of Molteni and Zaccaria [MZ20], that also lacks a proof connecting their method to probing security, which we provide, and INDIANA [BFG+25], through the concept of indistinguishability. Similarly, Rebecca [BGI+18], and its successors COCO [GHP+21] and COCOALMA [HB21], employ the Fourier expansion to estimate the linear structures of the probes, whereas we determine them exactly. Finally, Beyne et *al.* [BDZ20] use this approach to provide bounds in their proposed abstract probing model.

Our contribution. We introduce a method for formally analyzing the joint security of masking schemes and the output of compact PRNGs, as their source of randomness, within the robust d-probing model. In particular, our approach exploits the linear structure of Boolean functions, by adapting a commonly used technique to identify weaknesses in S-boxes, namely *linear cryptanalysis* [Mat94]. We assume an adversary to not have direct access to the inner processing of the PRNG, as in [CMM+24]. We suppose that the PRNG manipulates in some defined way its initial state (seed) coefficients to produce output variables. In short, our framework consists of substituting the shares and refresh variables in the algebraic expressions of the probes with the original secret variables and the initial state (seed) variables of the PRNG. Then, we conduct an algebraic analysis, based on the linear cryptanalysis methodology, to assess whether the resulting Boolean functions depend only on the original secret inputs, to evaluate d-(robust)-probing security. It allows us to determine if the studied Boolean functions are well-approximated by linear functions of their inputs, and in particular, of only the original secret values.

We provide a general and finer-grained framework, which makes it possible for any Boolean masking scheme, (1) to exploit the linear structure of the LFSR and determine whether some of its output values cancel out, (2) to determine whenever (robust) d-probing security is compromised, (3) depending on the chosen characteristic polynomial of the LFSR and (4) regardless of the seed quality provided to the LFSR, given that it is not the

all-zero state. Indeed, while [CMM+24] raised some concerns about the use of LFSRs for masking, the question of how to jointly analyze the security of their random outputs and a masking scheme in general remains open. We aim to bridge that gap; we show for the first time with a theoretical analysis and practical validation, that we can use an LFSR with a lower seed length than the number of required random bits in a masking scheme to remain secure. We expect that our methodology can be used to reduce effectively the cost of the overall masking implementation in actual applications. While designers consider the costs of masking schemes alone (e.g., in terms of area, latency, random bits, in hardware), we contribute to the discussion about the cost of the used PRNG for masking, as initiated by [CMM+24], and to identify small PRNGs that provide robust d-probing security. We present additional insights from comparing our theoretical analyses with practical evaluations. Furthermore, our work highlights that characteristic polynomials of the same degree (defining the LFSR length) can have a different impact on the security of the overall scheme, which is validated with practical evaluations on hardware implementations. We explain our method for an LFSR in either Fibonacci or Galois mode and point out that with respect to linear algebraic analysis, there is no advantage of one over the other.

Outline. First, we provide background information about our circuit model, masking and (robust) d-probing security, followed by a general description of LFSRs with respect to linear algebra and some of their properties in Section 3. Next, we present our linear cryptanalysis method in Section 4 and link it to the (robust) d-probing security notions. We also present a fine-grained application of our framework on small S-boxes masked at the first-order, namely the 4-bit and 8-bit SKINNY S-box [BJK+16] in Section 5. We also provide the results of our verification on a first-order masked AES S-box [AES01] in Section B.2. We validate this analysis by practical assessments in Section 6 for the hardware implementation of the first-order masked 4-bit SKINNY S-box using the HPC3 gadget [KM22b].

Extended version. The extended version of this paper presents additional theoretical security results for the second-order masked implementation of the 4-bit SKINNY S-box in Section 5 and Section B.2. Regarding the side-channel evaluations, we further validate our methodology through new measurements on the first-order masked 8-bit SKINNY S-box hardware implementation using the HPC3 gadget in Section 6. We also extend the practical security analysis of the first-order masked 4-bit SKINNY S-box for selected LFSR lengths in Section 6.

2 Preliminaries

2.1 Notation

We write vectors as $\vec{x}$ and a set in calligraphic font $\mathcal{S}$. The finite field with p elements is denoted by $\mathbb{F}_p$. We write $[a, b]_{\mathbb{N}}$ for the closed interval of natural numbers between a and b: $[a, b]_{\mathbb{N}} := \{x \in \mathbb{N} \mid a \leq x \leq b\}$. We write $\mathbb{P}[E]$ for the probability of the event E happening. A random probabilistic evaluation is indicated by $\xleftarrow{\$}$. We represent the addition modulo 2 (also called XOR) by $\oplus$ to distinguish it from the addition $+$ of real numbers ($\mathbb{R}$).

2.2 Circuit model

The implementation of an algorithm is referred to as a *circuit* C. It is composed of logical gates to perform the AND, NOT and XOR operations on Boolean values, memory gates (registers), probabilistic gates to output a random uniform value, and constant gates to provide a deterministic value. Wires connect the individual gates and carry Boolean values.

2.3 Masking

One of the most widely adopted countermeasure against SCA is *masking*, which was independently introduced by Chari et *al.* [CJRR99] and by Goubin and Patarin [GP99] in 1999. The core idea is to split a secret Boolean variable $x \in \mathbb{F}_2$ into a vector of *shares* $\vec{x} = (x_0, \ldots, x_d) \in (\mathbb{F}_2)^{d+1}$ such that $x = \bigoplus_{i=0}^{d} x_i$, where d represents the *order* of the masking. The shares x_i are chosen in such a way that it prevents the adversary from recovering the original secret x with the knowledge of any subset of at most d shares.

Encoding and Decoding. To create the shares of x at any order d, values z_i are drawn uniformly random from a source of randomness such that $\forall i \in [1, d]_{\mathbb{N}}$, $z_i \xleftarrow{\$} \mathbb{F}_2$, $x_i = z_i$ and $x_0 = x \oplus \bigoplus_{i=1}^{d} x_i$. This probabilistic procedure is called *encoding* (or sharing). Similarly, the recovery of the original secret x from the shares is named *decoding*. Both the encoding and decoding procedures are assumed to not be part of the masking scheme.

Gadget. To preserve secret-sharing in an algorithm, all operations are transformed to work on a (sub)set of shares and produce shared results, while maintaining the same functionality as when operating on unshared variables. A popular approach is using *gadgets* [ISW03], which are individual operations, or a set of operations, designed for securely manipulating shared values. At the d-th order, a gadget takes $d + 1$ inputs that are called *input shares* and produces $d + 1$ outputs or *output shares*. More broadly, we use the term circuit C to also refer to a masked implementation using several gadgets.

2.4 d-Probing Security

To not rely on experimental leakage assessments of circuits that are setup-dependent, we reason about and *prove* security in models representing an abstract view of the capabilities of a side-channel adversary. A simple, efficient and popular model is the *d-probing model* [ISW03], where an attacker can observe up to d values carried by the wires that are the evaluation of Boolean expressions. Those adversarial observations are called *probes*.

A circuit is *d-probing secure* if and only if the knowledge gained from observing up to d probes is statistically independent of any secret value. Its formal definition is provided below as stated in [KSM20], which is a formalization of the original idea from [ISW03].

Definition 1 (d-probing security)**.** Let C be a circuit and $\vec{X} = (X_0, \ldots, X_d)$ be a vector of discrete random variables where X_i represents which value the share $x_i \in \mathbb{F}_2$ takes. Similarly, let $\vec{Q} = (Q_0, \ldots, Q_{d-1})$ be a vector of d discrete random variables where Q_i describes the value of the probe q_i located on a wire of the circuit C. Let $\vec{x}, \vec{q}$ be arbitrary vectors of values of resp. shares x_i and probes q_i. The circuit C is *d-probing secure* if:

$$\mathbb{P}[\vec{Q} = \vec{q} \mid \vec{X} = \vec{x}] = \mathbb{P}[\vec{Q} = \vec{q}],$$

i.e., we cannot learn about all $d + 1$ secret shares from only d probes.

To ensure d-probing security within a circuit or gadget, we may need to draw additional independent and identically distributed random values to add them to some intermediate computations. This *refreshes* the underlying Boolean expressions.

2.5 Robust d-Probing Security

The d-probing model was extended in [FGM+18] to the *robust d-probing model* to account for additional leakage induced by physical effects in hardware implementations, namely *glitches*, *transitions* at register stages and *coupling* between wires. To capture these, new probes are created and added to the vector of probes $\vec{Q}$, as briefly described below.

Glitch. A glitch occurs when the inputs of a gate arrive asynchronously due to different path delays. As a result, an adversary may observe a transient, unexpected output value before it stabilizes. We define *glitch-extended probes* as new probes located on any input of the function carried by the probed wire up to the last synchronization point (register).

Transition. A transition is the observable power consumption difference in a memory cell of a register between two consecutive clock cycles, revealing the change in stored values. As a result, a *transition-extended probe* is a new probe put on the input of the memory element of a register whose output is probed.

Coupling. A coupling refers to the interaction between adjacent wires that may deviate the signal of one of the considered wires from its intended value (cross-talk). Thus, *coupling-extended probes* are new probes located on the wires adjacent to the probed one.

The probes are extended to account for the described physical effects, forming a worst-case scenario security model. Recent observations [MLM24] paved the way for a less conservative approach with the *robust-but-relaxed probing model* [MM24], which depends on the actual processed values and exact gate behavior of the considered circuit. However, for the sake of generality, we provide the mathematical foundations for the former conservative approach, but is easily adaptable to the needs of security practitioners.

3 Linear-Feedback Shift Register (LFSR)

LFSRs are primitives that can be used as PRNGs to provide randomness in masking schemes. They are among the most inexpensive PRNG options to implement in hardware [CMM+24] and their sequences of output bits satisfy desired statistical properties [Gol67]. However, a framework to provide a fine-grained formal security analysis when used in masking schemes, is currently lacking in the literature. We aim to bridge this gap by proposing a dedicated verification method in the (robust) d-probing model.

An LFSR is a finite state automaton of a state of length n that produces a periodic sequence over a finite field satisfying a linear recurrence relation [Gol67, Kle13, Can25]. In this work, because we are interested in manipulating algebraic expressions over $\mathbb{F}_2$, we write the state of the LFSR as a polynomial of degree n with coefficients in $\mathbb{F}_2$. For that, we describe the finite field of 2^n elements with an irreducible polynomial chosen to be primitive. The output sequence of the LFSR is then called an *m-sequence*, see below.

Definition 2 (m-sequence [Gol67])**.** An LFSR defined by a primitive polynomial of degree n over $\mathbb{F}_2$, produces a periodic sequence of length $2^n - 1$. It is said to be a maximal-length sequence, or m-sequence for short (m stands for maximal-length).

The primitive polynomial defining the LFSR is called the *characteristic* or *feedback polynomial*. In the remainder of the paper, we use the term characteristic polynomial and we assume that all studied n-bit LFSRs produce a m-sequence, i.e., the characteristic polynomial is always assumed to be primitive.

Our research questions are: *When is d-(robust)-probing security breached if we consider the randomness of a Boolean-masked circuit to be drawn from an LFSR?* and *Can we securely use an LFSR of smaller length than the required number of random bits for the masking scheme?* To answer them, we first define LFSRs with respect to linear algebra and we derive the linear recurrences for both modes of operations: Galois and Fibonacci. Thereafter, we present a brief overview of its statistical properties, a known linear algebra attack and its impact on security in the transition-extended d-probing model. For the sake of generality and convenience in this section, we assume every LFSR to have a register of length n bits. We also consider that the LFSR initial state coefficients are not equal to all zero values, as it would otherwise causes it to output only zeros.

3.1 Characteristic Polynomial

The general form of the primitive characteristic polynomial p for $\mathbb{F}_{2^n}$ is

$$p(X) = p_0 \oplus p_1\, X \oplus p_2\, X^2 \oplus \ldots \oplus p_{n-1}\, X^{n-1} \oplus p_n\, X^n$$

where $p_i \in \mathbb{F}_2$, for $i \in [1, n-1]_{\mathbb{N}}$, is the value of the i-th polynomial coefficient and $p_0 = p_n = 1$. When $p_i = 1$ for a n-bit LFSR, it is called a tap at position i.

3.2 Galois LFSR

We first present the Galois LFSR. It computes the next state of the LFSR by multiplying its current state by a root α of the primitive polynomial $p(X)$ in the standard polynomial basis $\langle \alpha_0, \alpha_1, \ldots, \alpha_{n-1} \rangle = \langle 1, \alpha, \ldots, \alpha^{n-1} \rangle$ of $\mathbb{F}_{2^n}$, such that

$$p(\alpha) = 1 \oplus p_1\, \alpha \oplus p_2\, \alpha^2 \oplus \ldots \oplus p_{n-1}\, \alpha^{n-1} \oplus \alpha^n = 0\,. \tag{1}$$

We denote the initial state, or seed, of the LFSR by S^0. In general, the j-th state of the LFSR S^j in the polynomial basis is

$$S^j(\alpha) = s_0^j\, \alpha_0 \oplus s_1^j\, \alpha_1 \oplus s_2^j\, \alpha_2 \oplus \ldots \oplus s_{n-1}^j\, \alpha_{n-1}\,, \tag{2}$$

where $s_i^j \in \mathbb{F}_2$ for all $i \in [0, n-1]_{\mathbb{N}}, j \in [0, 2^n - 2]_{\mathbb{N}}$, is the value of the i-th bit of the state S^j. The Galois LFSR's next state is computed by $S^{j+1}(\alpha) = \alpha S^j(\alpha)$.

From that, we easily derive the recurrence relation between the coefficients of a state S^{j+1} and the preceding state S^j from

$$\begin{aligned}
S^{j+1}(\alpha) = \alpha\, S^j(\alpha) &= \alpha\, (s_0^j \oplus s_1^j \alpha \oplus \ldots \oplus s_{n-1}^j \alpha^{n-1})\,, \\
&= s_0^j\, \alpha \oplus s_1^j\, \alpha^2 \oplus \ldots \oplus s_{n-1}^j\, \alpha^n \qquad \text{see Eq. (1)}, \\
&= s_0^j\, \alpha \oplus s_1^j\, \alpha^2 \oplus \ldots \oplus s_{n-1}^j\, (1 \oplus p_1\, \alpha \oplus \ldots \oplus p_{n-1}\, \alpha^{n-1})\,, \\
&= s_{n-1}^j \oplus (s_0^j \oplus s_{n-1}^j\, p_1)\, \alpha \oplus \ldots \oplus (s_{n-2}^j \oplus s_{n-1}^j\, p_{n-1})\, \alpha^{n-1}\,, \\
&= s_0^{j+1} \oplus s_1^{j+1}\, \alpha \oplus \ldots \oplus s_{n-1}^{j+1}\, \alpha^{n-1}\,.
\end{aligned}$$

In other words, we have:

$$\begin{cases} s_{i+1}^{j+1} &= s_i^j \oplus s_{n-1}^j\, p_{i+1} \text{ for } i \in [0, n-2]_{\mathbb{N}}, \\ s_0^{j+1} &= s_{n-1}^j\,. \end{cases} \tag{3}$$

The outputs of the Galois LFSR are the coefficients s_{n-1}^j for consecutive states S^j $(j > 0)$.

3.3 Fibonacci LFSR

For the Fibonacci LFSR, we work in the *dual* of the polynomial basis $\langle \beta_0, \beta_1, \ldots, \beta_{n-1} \rangle$ of $\mathbb{F}_{2^n}$. It computes the next state by multiplying the current state of the LFSR by a root of the primitive polynomial p with respect to this dual basis. In short, we use the same characteristic polynomial p as in the Galois case[1] and we compute $S^{j+1} = \alpha\, S^j$ with dual coordinates. To convert to the dual polynomial basis, we first need the *field trace* map Tr defined by

$$\mathrm{Tr}_{\mathbb{F}_{2^n}/\mathbb{F}_2}(\alpha) = \alpha \oplus \alpha^2 \oplus \alpha^{2^2} \oplus \ldots \oplus \alpha^{2^{n-1}}\,.$$

[1] Another convention uses both the reflected polynomial of p and initial state. For example, in $\mathbb{F}_2[X]$, the reflected polynomial of $X^4 \oplus X^3 \oplus 1$ is $X^4 \oplus X \oplus 1$; both are primitive.

It maps elements of $\mathbb{F}_{2^n}$ to $\mathbb{F}_2$ and is linear over $\mathbb{F}_2$. The relation between the polynomial and its dual basis is

$$\mathrm{Tr}(\alpha_i\,\beta_j) = \begin{cases} 0, & \text{if } i \neq j \\ 1, & \text{otherwise.} \end{cases}$$

We refer to McEliece's book [McE87, Chapters 8 and 9] for further details.[2]

Then, we write the state from Eq. (2) in the dual polynomial basis to operate the LFSR in Fibonacci mode. We now name this state R^j (note that $R^j(\beta) = S^j(\alpha)$) satisfying

$$R^j(\beta) = r_0^j\,\beta_0 \oplus r_1^j\,\beta_1 \oplus r_2^j\,\beta_2 \oplus \ldots \oplus r_{n-1}^j\,\beta_{n-1}\,,$$

such that, for all $i \in [0, n-1]_{\mathbb{N}}$,

$$s_i^j = \mathrm{Tr}(R^j(\beta)\,\beta_i) \quad \text{and} \quad r_i^j = \mathrm{Tr}(S^j(\alpha)\,\alpha_i)\,, \tag{4}$$

where the $S^j(\alpha)$ and s_i^j come from the Galois representation.

Remark 1. Note that $R^j(\beta) = S^j(\alpha)$ gives an equivalence between the Galois and Fibonacci LFSRs from an algebraic perspective.

From that, we derive the recurrence relation between the coefficients of the state R^j and the next one R^{j+1}, for $i \in [0, n-2]_{\mathbb{N}}$,

$$r_i^{j+1} = \mathrm{Tr}(S^{j+1}(\alpha)\,\alpha_i) = \mathrm{Tr}(\alpha\,S^j(\alpha)\,\alpha_i) = \mathrm{Tr}(S^j(\alpha)\,\alpha_{i+1}) \;\Leftrightarrow\; r_i^{j+1} = r_{i+1}^j \text{ from Eq. (4)}$$

and for $n-1$:

$$\begin{aligned} r_{n-1}^{j+1} &= \mathrm{Tr}(S^{j+1}(\alpha)\,\alpha_{n-1}) = \mathrm{Tr}(\alpha\,S^j(\alpha)\,\alpha_{n-1}) = \mathrm{Tr}(S^j(\alpha)\,\alpha_n)\,, \\ \Leftrightarrow r_{n-1}^{j+1} &= \mathrm{Tr}(S^j(\alpha)\cdot(\alpha_0 \oplus p_1\,\alpha_1 \oplus \ldots \oplus p_{n-1}\,\alpha_{n-1})) \quad \text{from Eq. (1)}\,, \\ &= \mathrm{Tr}(S^j(\alpha)\cdot\alpha_0) \oplus \mathrm{Tr}(S^j(\alpha)\cdot p_1\,\alpha_1) \oplus \ldots \oplus \mathrm{Tr}(S^j(\alpha)\cdot p_{n-1}\,\alpha_{n-1}), \\ &= \mathrm{Tr}(S^j(\alpha)\cdot\alpha_0) \oplus p_1\;\mathrm{Tr}(S^j(\alpha)\cdot\alpha_1) \oplus \ldots \oplus p_{n-1}\;\mathrm{Tr}(S^j(\alpha)\cdot\alpha_{n-1})\,, \\ &= r_0^j \oplus p_1\,r_1^j \oplus \ldots \oplus p_{n-1}\,r_{n-1}^j\,, \end{aligned}$$

which corresponds to the sum of the coefficients r_i^j on the tap positions. In short, we have

$$\begin{cases} r_i^{j+1} & = r_{i+1}^j\,, \quad \text{for } i \in [0, n-2]_{\mathbb{N}}; \\ r_{n-1}^{j+1} & = r_0^j \oplus \bigoplus_{k=1}^{n-1} p_k\,r_k^j\,. \end{cases} \tag{5}$$

The outputs of the Fibonacci LFSR are r_0^j coefficients for consecutive states R^j $(j > 0)$.

3.4 Discussion on Statistical and Algebraic Properties

In the following, we provide a literature summary of the statistical properties of LFSRs to motivate their practical use. We also present a known linear algebra attack (or known-plaintext attack against one LFSR [Kle13, PPG24]) explaining why LFSRs are currently discarded in cryptographic schemes, but might be still relevant for masking. Finally, we highlight key facts about their linear recurrences and discuss observations from [CMM+24].

3.4.1 Statistical Properties

In 1967, Golomb stated in [Gol67] three randomness properties that are satisfied by any m-sequence of length $2^n - 1$ (see Definition 2), which are:

[2] For a concrete example in $\mathbb{F}_{2^4}$, the polynomial basis is $\langle\alpha_0, \alpha_1, \alpha_2, \alpha_3\rangle = \langle 1, \alpha, \alpha^2, \alpha^3\rangle$ and its dual basis is $\langle\beta_0, \beta_1, \beta_2, \beta_3\rangle = \langle 1 \oplus \alpha, 1 \oplus \alpha^2 \oplus \alpha^3, \alpha \oplus \alpha^3, \alpha \oplus \alpha^2\rangle$, computed with the field trace map.

(P1) The said output sequence contains 2^{n-1} ones and $2^{n-1}-1$ zeros;

(P2) Let a *run* be any number of consecutive identical bits, e.g., 000 is a run of length 3. For an m-sequence, the number of runs of $n-k$ bits is equal to 2^{k-1} for $k \in [1, n-1]_{\mathbb{N}}$ and 1 for $k = 0$.

(P3) The autocorrelation of an m-sequence $t_1, \ldots, t_{2^n-1}$ with any circularly shifted version of itself by $\tau \neq 0 \pmod{2^n - 1}$ positions has the same value. It measures the agreement between the sequences $\{t_{j+\tau}\}$ and $\{t_j\}$.

The first postulate implies that the LFSR generates a maximal-length sequence that is close to uniform. The second postulate means that shorter runs are more frequent than longer ones, and the third postulate signifies that past and future output values are uncorrelated to the considered sequence. These are all desirable properties of a PRNG.

While these properties are fundamental for a PRNG suitable for practical use, additional features of randomly generated numbers are typically assessed using statistical test suites, like, for example, NIST SP 800-22 [BRS+10], Diehard [Mar08] and Dieharder [BEB18], TestU01 [LS07], ENT [Wal08] and PractRand [DH]. Each test suite contains multiple tests and some of those are particularly designed to detect linear structures over $\mathbb{F}_2$. An example is the binary matrix rank test that divides the sequence in fixed-size matrices and compute their ranks, which is present with some variation in the NIST SP 800-22, Diehard, PractRand or TestU01 test suites. The authors of [SW13] prove that this test fails as intended for two 32-bit LFSRs with the NIST SP 800-22 test suite. Furthermore, the authors of [FYC24] compare a 32-bit LFSR output to two hardware RNGs through several test suites. They show that the LFSR fails more tests than the others. They also investigate post-processing methods to improve the random quality of the output.

On a side note, while statistical test suites are useful to detect sequences of numbers that are deemed not random enough, their implementations can fall short [SOMK22] or issues have been raised concerning the interpretation of their results [MS15]. We refer to [FYC24] for more of their shortcomings.

3.4.2 Linear Algebra Attack

From an algebraic perspective, it is possible to recover the values of the coefficients of the initial state (seed) with the first n output bits of an n-bit LFSR—if the primitive polynomial is known—by solving a system of n linear equations. If the primitive polynomial is unknown, an adversary needs n additional output bits to recover its coefficients [Kle13, PPG24]. Therefore, given a sufficient amount of output bits, an adversary can produce the entire LFSR sequence, effectively breaking the randomness generation.

Transitions in the d-Robust Probing Model. If the primitive polynomial is known, the authors of [CMM+24] highlight that, for an n-bit LFSR, its initial state coefficients could be retrieved by probing $n/2$ registers due to transitional effects in the robust d-probing model, which allow to extend the probe on the output of a register to its input (see Section 2.5). However, this attack does not hold when $2d < n$ in this model, where d is the number of probes. Indeed, it is not possible to uniquely solve a system of $2d$ linear equations of n variables if $2d < n$. Note that an adversary learns about a new coefficient for each probe only in the best-case scenario. Concretely, it means that to protect against $d = 3$ probes here, we need use at least a 7-bit LFSR, which is a rather inexpensive requirement.

3.4.3 Linear Recurrences

The authors of [CMM+24] wrote the following linear recurrence for an n-bit Fibonacci LFSR with initial state coefficients $r_0, \ldots, r_{n-1}$: there exist $a_i \in \mathbb{F}_2$ such that $r_{i+n} = \bigoplus_{j=0}^{n-1} a_j r_{i+j}$,

which corresponds to our Eq. (5). Indeed, their coefficients a_i correspond to the ones of the chosen primitive polynomial of degree n (in our notation, p_i) and the newly created coefficients of the following LFSR state are indexed by i (i.e., their r_n equals r^1_{n-1} in our notation, their r_{n+1} corresponds to our r^2_{n-1}, etc.). From that, the authors of [CMM+24] point out (using their notation) that $\bigoplus_{j=0}^{n} a_j r_{i+j} = 0$ and argue that it can break d-probing security. We note that this result comes from the definition of the linear recurrences of a Fibonacci LFSR where $r^1_{n-1} \oplus \bigoplus_{k=0}^{n-1} p_k\, r^0_k = 0$ (see Eq. (5)). In light of this concern, the primary purpose of our verification method is to evaluate combinations of output variables, determine whether they cancel out, and detect whether they compromise d-probing security.

In general, we note that the sum of $n+1$ (or n) outputs bits of an n-bit LFSR does not necessarily add up to zero. We consider a small counterexample with a 3-bit Fibonacci LFSR with $p(X) = X^3 + X + 1$ and initial state coefficients r^0_0, r^0_1, r^0_2. The first $n+1 = 4$ outputs bits are $(r^1_0, r^2_0, r^3_0, r^4_0) = (r^0_1, r^0_2, r^0_0 \oplus r^0_1, r^0_1 \oplus r^0_2)$, which sum is $r^0_0 \oplus r^0_1$. If we add the $n = 3$ first outputs, we get $r^0_0 \oplus r^0_2$. The same holds for a Galois LFSR.

However, the sum of $2^n - 1$ output bits of an n-bit LFSR in either mode is necessarily zero as a direct consequence of Golomb's first randomness property (see Section 3.4.1). This is caught by our security analysis and flagged if d-probing security is breached.

Concerning linear dependencies of an m-sequence (different than linear recurrences) as mentioned in [CMM+24], if we consider n consecutive variables $t_0, \ldots, t_{n-1}$ of an n-bit LFSR at any location in its output sequence, then they can be considered random variables, i.e., functions with co-domain $\mathbb{F}_2$, and form a vector space. From there, one might find scalars $\gamma_0, \ldots, \gamma_{n-1}$ that are not all zeros such that $\bigoplus_{i=0}^{n-1} t_i\, \gamma_{n-1}$ and deduct that the functions $t_0, \ldots, t_{n-1}$ are linearly dependent. However, it remains unclear how this result could be leveraged to compromise d-probing security.

3.4.4 Summary and Implications

As a result, an LFSR is generally not recommended *without a carefully conducted analysis* for cryptographic use or masking due to its linear structure, despite its desirable statistical properties. However, when such analysis is performed, we argue that LFSRs are potential randomness generators in the context of masking, where the adversary's observations are limited and the research community tends to minimize the use of randomness by designing low-randomness masking schemes [Sug18, SM21, BDRS22, SDM23, GKM+24]. In that scenario, the relevant question becomes how to carefully select LFSR parameters to ensure security? To this end, we provide a framework to formally analyze the security of masking schemes whose the source of randomness is an LFSR with varying lengths and characteristic polynomials. Our security analysis is also applicable to other randomness generators, including the ones with nonlinear structures like Trivium [Can06].

4 Linear Cryptanalysis

In this work, we perform a detailed analysis of the (robust) d-probing security implications of using random variables drawn consecutively from a single LFSR to create secret shares and to refresh gadgets. That said, the presented method can be extended to other scenarios, such as using one LFSR for encoding and another for refreshing, selecting non-consecutive output bits from an LFSR for both operations, employing a Nonlinear Feedback Shift Register (NLFSR), or to test any other minimal PRNG design.

To exploit the inherent linear recurrence relation of LFSRs, we apply and adapt a well-known technique in symmetric cryptography, namely linear cryptanalysis [Mat94, KR11], to compute *linear approximations* of the probe algebraic expressions holding with a certain probability, or *correlation*. This method relies on the application of the Walsh-Hadamard Transform on Boolean functions. More precisely, the key steps of our verification are first,

to write the Algebraic Normal Forms (ANFs) of the probes as functions of the original secret and the initial state variables of the PRNG. Then, we compute the Walsh-Hadamard Transform (WHT) and search for non-zero correlations between the probe expressions and linear combinations of only the original secret variables. The latter indicate that the probes can be well-approximated by linear combinations of unmasked secret variables, thereby signifying that d-probing security is compromised. For that, we first present the WHT for Boolean functions, then for vectorial Boolean functions to verify higher-order probing security. In addition, we present the detailed steps of our method with a theorem linking the WHT to the direct notion of statistical independence from the definition of d-probing security (see Definition 1).

4.1 Walsh-Hadamard Transform

First, we present the Walsh-Hadamard Transform for Boolean functions to derive which linear combination of input variables holds with the highest probability. We then define it for vectorial Boolean functions (S-boxes).

4.1.1 Boolean Functions

The WHT is the application of the discrete Fourier Transform to Boolean functions. It gives an integer vector of length 2^n that measures the distance between a Boolean function $f\colon \mathbb{F}_2^n \to \mathbb{F}_2$ and all possible linear functions from $\mathbb{F}_2^n$ to $\mathbb{F}_2$, or linear approximations of the Boolean function f. The vector itself is called the Walsh spectrum of the function f. It is generally extended to provide the decomposition into affine functions as well (i.e., linear functions with $\oplus 1$).

More formally, all linear functions in $\mathbb{F}_2^n$ are represented by $\vec{u}^\intercal \vec{x}$ where $\vec{u} \in \mathbb{F}_2^n$ is a vector of Boolean values that is called an input *mask* in symmetric cryptography, or a *spectral coordinate*. To be explicit, we have $\vec{u}^\intercal \vec{x} = u_0 x_0 \oplus u_1 x_1 \oplus \ldots \oplus u_{n-1} x_{n-1} \in \mathbb{F}_2$. The input mask selects the coordinates $x_i, i \in [0, n-1]_{\mathbb{N}}$, of an input vector $\vec{x}$ when $u_i = 1$, thereby creating a linear function of the input variables. For a function $f\colon \mathbb{F}_2^n \to \mathbb{F}_2$, the WHT of f with the input mask $\vec{u}$ is defined by the map $\mathrm{W}_f\colon \mathbb{F}_2^n \to \mathbb{Z}$,

$$\mathrm{W}_f(\vec{u}) = \sum_{\vec{x} \in \mathbb{F}_2^n} (-1)^{f(\vec{x}) \oplus \vec{u}^\intercal \vec{x}}\,, \tag{6}$$

which outputs the Walsh coefficient between the linear form $\vec{u}^\intercal \vec{x}$ (or mask $\vec{u}$) and the function f. In other words, it counts the number of agreements ($+1$ per agreement) and disagreements (-1 per disagreement) between the output vector of the Boolean function f and $\vec{u}^\intercal \vec{x}$. We iterate this computation for all possible $\vec{u}$ to obtain a vector called the Walsh spectrum of f

$$\vec{\mathrm{Sp}}_f = (\mathrm{W}_f(\vec{u}_0), \mathrm{W}_f(\vec{u}_1), \ldots, \mathrm{W}_f(\vec{u}_{2^n-1}))\,, \tag{7}$$

where $\vec{u}_i$ represents an input mask in $\mathbb{F}_2^n$ indexed by their binary vector representation. A fast computation of the Walsh Spectrum is obtained through the Fast Walsh-Hadamard Transform (FWHT) algorithm with time complexity $O(n \cdot 2^n)$, instead of $O(2^{2n})$ for the naive algorithm, for a truth table of size 2^n [Jou09].

From the Walsh coefficient, we compute the correlation between f and the linear function encoded by $\vec{u}$ with the map $\mathrm{Cor}_f\colon \mathbb{F}_2^n \to [-1, +1]$,

$$\mathrm{Cor}_f(\vec{u}) = \frac{1}{2^n}\,\mathrm{W}_f(\vec{u})\,, \tag{8}$$

In this context, $\mathrm{Cor}_f(\vec{u}) = 1$ represents a perfect agreement and $\mathrm{Cor}_f(\vec{u}) = -1$ a perfect disagreement between between f with a given linear Boolean function $\vec{u}^\intercal \vec{x}$. If a correlation

coefficient is not equal to zero, the functions are correlated to some extent. When the correlation is a negative value, we simply add $\oplus 1$ to the linear function $\vec{u}^\intercal \vec{x}$ to have an affine function with a positive correlation value to f.

If we re-write it differently, we have

$$\begin{aligned}\mathrm{Cor}_f(\vec{u}) &= \mathbb{P}[f(\vec{x}) \oplus \vec{u}^\intercal \vec{x} = 0] - \mathbb{P}[f(\vec{x}) \oplus \vec{u}^\intercal \vec{x} = 1]\,,\\ &= 2\,\mathbb{P}[f(\vec{x}) = \vec{u}^\intercal \vec{x}] - 1\,,\\ &=: 2\,\mathrm{p}_f(\vec{u}) - 1\,,\end{aligned}$$

with $\mathrm{p}_f(\vec{u})$ the joint agreement probability between the Boolean function f and the linear one represented by $\vec{u}^\intercal \vec{x}$. The intuition is that, the higher the absolute value of the correlation between f and a linear form $\vec{u}^\intercal \vec{x}$ is, the higher the dependency between them. An example to illustrate those notions is given in Table 7 in Section A for the multiplication of two Boolean variables $f\colon \mathbb{F}_2^2 \to \mathbb{F}_2,\ (x_1, x_0) \mapsto x_0 x_1$.

4.1.2 Vectorial Boolean Functions

Having defined the WHT for Boolean functions, we now consider combinations of multiple probes to ensure higher-order security ($d > 1$). We thereby extend the previous definitions by introducing correlation matrices. They were first mentioned by Daemen [DGV95] for S-boxes, and their use to assess d-probing security was mentioned by Molteni and Zaccaria in [MZ20] for a masking scheme only, then extended to compositional notions in [MZC22]. It is also the main foundation of the bounded-query model proposed by Beyne *et al.* [BDZ20] as a variant of the d-probing model. However, none of these works consider the joint security analysis with the source of randomness.

In general, for a vectorial Boolean function $f\colon \mathbb{F}_2^n \to \mathbb{F}_2^m$, we consider linear approximations for the component functions of f and their linear combinations. For that, we define an output mask $\vec{v} \in \mathbb{F}_2^m$ to select the component functions and we extend the definition of WHT (from Eq. (6)) to $\mathrm{W}_f\colon \mathbb{F}_2^m \times \mathbb{F}_2^n \to \mathbb{Z}$ as follows:

$$\mathrm{W}_f(\vec{v}, \vec{u}) = \sum_{\vec{x} \in \mathbb{F}_2^n} (-1)^{\vec{v}^\intercal f(\vec{x}) \oplus \vec{u}^\intercal \vec{x}}\,. \tag{9}$$

From this, we can create a correlation matrix Cor_f indexed by the input and output masks where the coefficients are computed similarly to Eq. (8) by

$$\begin{aligned}\mathrm{Cor}_f(\vec{v}, \vec{u}) = \frac{1}{2^n}\,\mathrm{W}_f(\vec{v}, \vec{u}) &= 2\,\mathbb{P}[\vec{v}^\intercal f(\vec{x}) = \vec{u}^\intercal \vec{x}] - 1\,,\\ &=: 2\,\mathrm{p}_f(\vec{v}, \vec{u}) - 1\,,\end{aligned}$$

with $\mathrm{p}_f(\vec{v}, \vec{u})$ the joint agreement probability between the linear functions $\vec{v}^\intercal f(\vec{x})$ and $\vec{u}^\intercal \vec{x}$. For higher-order probing security ($d > 1$), the component functions of f are the ANFs of the individual probes and the output masks $\vec{v}$ are created to combine at most d probes (i.e., the functions $\vec{v}^\intercal f(\vec{x})$ are all possible linear combinations of at most d probes).

4.2 (Robust) Probing Security and Linear Cryptanalysis

In this section, we present our verification method to jointly analyze the randomness generator and a masking scheme, and we provide a proof to link our use of linear cryptanalysis to d-probing security. Afterwards, we describe how the WHT can be leveraged to extend the probes according to the d-robust probing model.

4.2.1 Verification Steps

For our verification, we first gather the algebraic expressions of the probes from the considered circuit, and we convert them to their ANFs. Then, we proceed with the following steps described for an LFSR, however, it is applicable to other PRNGs:

1. **Variable parameterization.** In the probe ANFs, we first substitute the input shares with their encoding definitions (see Section 2.3), and replace all encoding and refreshing random variables with the LFSR output variables. Next, we express these outputs as functions of the LFSR's initial state variables, using the recurrence relations in either Galois (see Eq. (3)) or Fibonacci (see Eq. (5)) mode, for a known characteristic polynomial. In short, we express the ANFs of the probes as functions of the original secret inputs and the LFSR seed variables;

2. **Applying the WHT.** In a second step, we exploit the inherent linear relations of the resulting ANFs by computing the linear approximations, with the Walsh-Hadamard transform, of the internal and output probes in the gadgets (see Section 2.3) for various characteristic polynomials of different degrees. At the order d, we create combinations of up to d probes $\vec{v}^\intercal f(\vec{x})$ to build the correlation matrix Cor_f.

3. **Checking d-probing security.** Finally, for each considered $\vec{v}^\intercal f(\vec{x})$, we look for non-zero correlation values $\mathrm{Cor}_f(\vec{v}, \vec{u})$ where the input mask $\vec{u}$ contains only the original secret variables. In other words, we assess whether the probe ANFs can be well-approximated by a linear combination of original secret variables.

For the third step, if the correlation values are not zero for these secret-only linear approximations, it means that the (combination of) probe(s) is statistically dependent on secret inputs; therefore, d-probing security is compromised. We clarify and prove this relation in Section 4.2.2. In addition, we provide detailed first- and second-order examples of our framework in Section B.1 for one of the studied masking schemes in Section 5. Note that we are only interested in a few selected values of the correlation matrix (or Walsh spectrum). Moreover, because we look for secret-only linear approximations of the probe expressions that do not depend on the LFSR initial state variables, our verification is agnostic to the values of the seed, as long as it is not the all-zero state to not lock the LFSR at zero. The presented verification method assumes that the SCA adversary knows the description of the masking scheme and the use of the LFSR for encoding and refreshing.

A variant of the WHT, called Fourier expansion, is used by the formal verification tools Rebecca [BGI+18] and COCO [GHP+21]. However, in the latter, they only estimate the Fourier/Walsh-Hadamard coefficients that characterize statistical independence (without a formal proof), and they do not consider a particular choice of random generator. One could adapt their method with these estimations to make statements for possibly larger LFSRs due to the potential reduction in computational complexity. However, we present here a method with all exact linear Boolean functions to leave no margin of error.

The WHT is used in another automated verification tool, that is, INDIANA [BFG+25] to reason about robust d-probing security for verify a masking scheme only. For that purpose, INDIANA works with the notion of indistinguishability, instead of statistical independence. They define the leakage of the circuit, given a set of probes $\mathcal{Q}$, to be 0, if and only if for all (extended) probes in $\mathcal{Q}$ and all possible inputs, the WHT of a specifically defined difference function is 0. While their result seems similar to ours, we focus instead on comparing the correlations, defined as the probability of f being equal to an input mask minus the probability that it differs from it. Although both methods use the WHT and compare probe values to input masks, we believe our method is more elementary by working directly with statistical independence for d-probing security.

The authors of [BDZ20] introduce a new framework and probing security model, namely the bounded-query probing model. It is closely similar to the d-probing model, where upon

plaintext and key inputs and a set of probes (positions), an oracle is queried a finite (hence bounded) number of times. The results from the queries are the values carried by the probes. In this model, provided the expected value of the probability distribution of the probes w.r.t. the inputs is low enough, and assuming the adversary retrieves only outcomes from the queries that reveal some information, there is a proven upper bound for the advantage of the d-probing adversary. Using the WHT to compute linear approximations of the function underlying the circuit, the said expected value can be upper bounded using linear cryptanalaysis methods, thus making the upper bound for the advantage of the adversary more precise. However, we present our verification in the d-probing model [ISW03], and not the bounded-query probing model, as it was proven that being secure in the d-probing translates into being secure into more realistic models [DDF14].

4.2.2 d-Probing Security: Statistical Independence and Correlation

We explicitly link the WHT and the statistical independence as used in Section 4.2.1, as it has not been proven and clarified in the literature [BGI+18, BDZ20, MZ20, BFG+25]. We fill that gap in Theorem 1 for Boolean functions. For that, we first recall the definition of statistical independence and prove a useful lemma regarding linear maps.

Definition 3. Two Boolean functions $f\colon \mathbb{F}_2^n \to \mathbb{F}_2$ and $g\colon \mathbb{F}_2^n \to \mathbb{F}_2$ are *statistically independent* if for any outputs $a, b \in \mathbb{F}_2$, we have

$$\mathbb{P}[f(\vec{x}) = a, g(\vec{x}) = b] = \mathbb{P}[f(\vec{x}) = a] \cdot \mathbb{P}[g(\vec{x}) = b].$$

Lemma 1. *Let $\vec{u} \in \mathbb{F}_2^n$ with $\vec{u} \neq \vec{0}$. Then we have $\mathbb{P}[\vec{u}^\intercal \vec{x} = 0] = \mathbb{P}[\vec{u}^\intercal \vec{x} = 1] = \frac{1}{2}$.*

Proof. The set $\{\vec{x} \in \mathbb{F}_2^n : \vec{u}^\intercal \vec{x} = 0\}$ has co-dimension one, hence contains 2^{n-1} elements. □

Theorem 1. *Let $f\colon \mathbb{F}_2^n \to \mathbb{F}_2$ be a Boolean function and let $u \in \mathbb{F}_2^n$. We write $\vec{u}^\intercal(\text{-})$ for the linear map $\vec{x} \mapsto \vec{u}^\intercal \vec{x}$. Then the link between $\mathrm{Cor}_f(\vec{u})$ and the statistical independence of f and $\vec{u}^\intercal(\text{-})$ as Boolean functions is given by:*

(a) If $\vec{u} = \vec{0}$, then f and $\vec{u}^\intercal(\text{-})$ are statistically independent (regardless of $\mathrm{Cor}_f(\vec{u})$).

(b) If $\vec{u} \neq \vec{0}$, then f and $\vec{u}^\intercal(\text{-})$ are statistically independent if and only if $\mathrm{Cor}_f(\vec{u}) = 0$.

Proof. (a) This is straightforward, since $\vec{u}^\intercal \vec{x} = 0$ for all $\vec{x}$ when $\vec{u} = \vec{0}$: Therefeore we have $\mathbb{P}[f(\vec{x}) = a, \vec{u}^\intercal \vec{x} = 1] = 0$ and $\mathbb{P}[f(\vec{x}) = a, \vec{u}^\intercal \vec{x} = 0] = \mathbb{P}[f(\vec{x}) = a]$.

(b) $\Rightarrow$:) Assume that f and $\vec{u}^\intercal(\text{-})$ are statistically independent. Then we have

$$\begin{aligned}
\mathrm{Cor}_f(\vec{u}) &= 2\,\mathbb{P}[f(\vec{x}) = \vec{u}^\intercal \vec{x}] - 1\,, \\
&= 2\,(\mathbb{P}[f(\vec{x}) = 0, \vec{u}^\intercal \vec{x} = 0] + \mathbb{P}[f(\vec{x}) = 1, \vec{u}^\intercal \vec{x} = 1]) - 1\,, \\
&= 2\,(\mathbb{P}[\vec{u}^\intercal \vec{x} = 0]\ \mathbb{P}[f(\vec{x}) = 0] + \mathbb{P}[\vec{u}^\intercal \vec{x} = 1]\ \mathbb{P}[f(\vec{x}) = 1]) - 1\,, \\
&= 2\left(\frac{1}{2}\,\mathbb{P}[f(\vec{x}) = 0] + \frac{1}{2}\,\mathbb{P}[f(\vec{x}) = 1]\right) - 1 = 0
\end{aligned}$$

where the second-to-last equality comes from Lemma 1.

$\Leftarrow$:) Now, assume $\mathrm{Cor}_f(\vec{u}) = 0$. We have: $0 = \mathrm{Cor}_f(\vec{u}) = 2\mathbb{P}[f(\vec{x}) = \vec{u}^\intercal \vec{x}] - 1$. Hence, we have $\mathbb{P}[f(\vec{x}) = \vec{u}^\intercal \vec{x}] = \frac{1}{2}$. This gives us:

$$\mathbb{P}[f(\vec{x}) = 0, \vec{u}^\intercal \vec{x} = 0] + \mathbb{P}[f(\vec{x}) = 1, \vec{u}^\intercal \vec{x} = 1] = \frac{1}{2}, \quad \text{and} \tag{10}$$

$$\mathbb{P}[f(\vec{x}) = 0, \vec{u}^\intercal \vec{x} = 1] + \mathbb{P}[f(\vec{x}) = 1, \vec{u}^\intercal \vec{x} = 0] = \frac{1}{2}. \tag{11}$$

Additionally, from Lemma 1, we find:

$$\mathbb{P}[f(\vec{x}) = 0, \vec{u}^\intercal \vec{x} = 0] + \mathbb{P}[f(\vec{x}) = 1, \vec{u}^\intercal \vec{x} = 0] = \frac{1}{2}, \quad \text{and} \tag{12}$$

$$\mathbb{P}[f(\vec{x}) = 1, \vec{u}^\intercal \vec{x} = 1] + \mathbb{P}[f(\vec{x}) = 0, \vec{u}^\intercal \vec{x} = 1] = \frac{1}{2}. \tag{13}$$

Combining Equations (10), (11), (12) and (13), we find that

$$\begin{aligned} &\mathbb{P}[f(\vec{x}) = 0, \vec{u}^\intercal \vec{x} = 0] = \mathbb{P}[f(\vec{x}) = 0, \vec{u}^\intercal \vec{x} = 1], \quad \text{and} \\ &\mathbb{P}[f(\vec{x}) = 1, \vec{u}^\intercal \vec{x} = 0] = \mathbb{P}[f(\vec{x}) = 1, \vec{u}^\intercal \vec{x} = 1]. \end{aligned}$$

Now we are going to prove that

$$\mathbb{P}[f(\vec{x}) = a, \vec{u}^\intercal \vec{x} = b] = \mathbb{P}[f(\vec{x}) = a] \cdot \mathbb{P}[\vec{u}^\intercal \vec{x} = b]$$

to show statistical independence. We only focus here on the case where $(a, b) = (0, 0)$, the other cases are derived in a similar fashion. We have:

$$\mathbb{P}[f(\vec{x}) = 0] = \mathbb{P}[f(\vec{x}) = 0, \vec{u}^\intercal \vec{x} = 0] + \mathbb{P}[f(\vec{x}) = 0, \vec{u}^\intercal \vec{x} = 1] = 2\mathbb{P}[f(\vec{x}) = 0, \vec{u}^\intercal \vec{x} = 0]$$

and $\mathbb{P}[\vec{u}^\intercal \vec{x} = 0] = \frac{1}{2}$ by Lemma 1. Together, as required, we get

$$\mathbb{P}[f(\vec{x}) = 0] \cdot \mathbb{P}[\vec{u}^\intercal \vec{x} = 0] = 2 \cdot \mathbb{P}[f(\vec{x}) = 0, \vec{u}^\intercal \vec{x} = 0] \cdot \frac{1}{2} = \mathbb{P}[f(\vec{x}) = 0, \vec{u}^\intercal \vec{x} = 0]. \quad \square$$

Remark 2. In particular, if $\vec{u} = \vec{0}$ and $\mathrm{Cor}_f(\vec{u}) = 0$, then f is balanced:

$$0 = \mathrm{Cor}_f(\vec{u}) = 2 \cdot \mathbb{P}[f(\vec{x}) = \vec{u}^\intercal \vec{x}] - 1 \implies \mathbb{P}[f(\vec{x}) = 0] = \frac{1}{2},$$

hence also $\mathbb{P}[f(\vec{x}) = 1] = \frac{1}{2}$.

We highlight some considerations for higher-order security verification below.

Remark 3. About vectorial Boolean functions, note that each coordinate of the correlation matrix contains the correlation value of two functions. Usually, the columns are indexed by input masks and the rows by output masks in linear cryptanalysis. The theorem above is then applicable to check whether one input mask (linear approximation of inputs) is statistically independent of one output mask (linear approximation of probes).

Remark 4. We point out that the correlation matrix computes linear combinations of the probes (output masks). It implies that a correlation value speaks about the agreement probability of the combination of Boolean functions (here, probes) and may provide a less fine analysis than looking at the joint distribution of the said functions. However, we can tackle this issue by considering all literals of our Boolean functions. To illustrate our point, let x_0, x_1, x_2 be shares of a secret x for $d = 2$ and z a refreshing variable. Let $q_1 = x_0 \oplus z$ and $q_2 = x_1 x_2$ be two probes of a (trivially insecure) gadget. We see that the joint probability distribution (q_1, q_2) is not statistically independent from (x_0, x_1, x_2). Despite that, in the correlation matrix, to verify second-order probing security, the value of the correlation of $q_1 \oplus q_2$ with $x_0 \oplus x_1 \oplus x_2$ is zero, meaning that no security violation is found for this approximation. However, when combining q_2 with x_0, we find it correlated with value $1/2$ to $x_0 \oplus x_1 \oplus x_2$: the gadget then fails second-order security.

4.2.3 Robust Extension

We present, in our framework, the probe extensions for the robust probing model.

Glitches. A probe can be easily extended for glitches with the WHT under some conditions. For a strictly *linear* Boolean function, one needs to manually add the monomials of its algebraic expressions to the probes to verify to run the security analysis. For example, if $f(a, b, c, d) = a \oplus b \oplus c$, then a practitioner creates new extended probes on a, b, c and adds them to the existing probes to run the verification. If the function is perfectly nonlinear,[3] then applying the WHT extends the probe by definition since it provides all linear approximations of its inputs. For example, if $f(a, b, c, d) = ab$, then the application of the WHT outputs the approximations that all have the same agreement probability $\mathrm{p}_f(\vec{u})$, where $\vec{u}^\intercal \vec{x} \in \{0, a, b, a \oplus b \oplus 1\}$. A function that is not perfectly nonlinear is a mix between the two cases: the linear terms are added to all linear approximations of the nonlinear terms. In this regard, glitch-extended probes are constructed in the same manner as done by Rebecca [BGI+18] with Fourier expansions. Indeed, for non-linear functions, their Fourier expansion method naturally takes into account glitches (like ours), and linear functions are dealt with as though they are non-linear: Meaning that, e.g., when probing a function $x \oplus y$, the probe is extended to probing x, y and $x \oplus y$. A similar approach is used in [BDZ20] and [BFG+25].

Transitions. A transition-extended probe can be created by summing of the probed output of a register with its input. Then, this transition-extended probe is added to the probes to verify. For example, if the probed register at clock cycle t corresponds to the function f_t, then, to account for the transition, we apply the WHT on the sum $f_t \oplus f_{t-1}$.

Coupling. Coupling-extended probes are new probes put on the adjacent wires of the probed wire. To study the local effect of a coupling, the application of the WHT on the probed wire expression (e.g., f_{w_i}) and the ones of its adjacent neighbors (e.g., $f_{w_{i-1}}, f_{w_{i+1}}$) creates, by definition, the correlation matrix Cor_f with all their linear combinations, to capture their influence on each other's signal values. We then consider the correlation values of the output masks of this matrix corresponding to the linear combinations of the probed wire and its adjacent ones (here, f_{w_i}, $f_{w_i} \oplus f_{w_{i-1}}$, $f_{w_i} \oplus f_{w_{i+1}}$, $f_{w_i} \oplus f_{w_{i-1}} \oplus f_{w_{i+1}}$).

5 Case Study: First- and Second-order Masked S-boxes

We showcase our method with a detailed theoretical analysis of the first-order masked hardware implementations of the 4-bit and 8-bit S-box in SKINNY using different multiplication gadgets, as well as the first-order masked 8-bit AES S-box implementation [AES01] using the HPC3 [KM22b] multiplication gadget. For the extended version of this paper, we add the results for the second-order masked 4-bit SKINNY S-box circuit with various multiplication gadgets. We base our analysis on the implementations generated in Verilog by the automated hardware masking design tool HADES from Buschkowski et *al.* [BLH+25]. Assuming the random variables are uniformly distributed and independent, the first- and second-order masked implementations achieve first- and second-order probing security, respectively. We symbolically translate these circuits into Sagemath version 9.5 scripts to extract and analyze the ANF of the probes, and then evaluate their WHT. For each circuit, we assume that the sharing and refreshing were made by drawing random values from a single LFSR, from which we vary the length. Furthermore, we test for each LFSR length different characteristic polynomials. The outcome of the verification for a single characteristic polynomial is identical in Galois and Fibonacci mode (see Remark 1), therefore checking for only one mode is sufficient. Then, we evaluate whether the resulting scheme is glitch-extended first- or second-probing secure with our method (see Section 4.2.1). The

[3] A Boolean function f that is perfectly nonlinear is called a bent function: All the absolute Walsh coefficient values $|\mathrm{W}_f(\vec{u})|$ in its spectrum $\vec{\mathrm{Sp}}_f$ are equal.

circuit description and verification scripts are provided in the supplementary material.[4] As stated before, we assume that an adversary cannot probe the internal state of the LFSR: The attacker may only probe the output bits of the LFSR while being processed during the implementation execution. However, the adversary knows the used characteristic polynomial and how the LFSR outputs are used.

First, we provide a brief description of the gadgets used in our masked examples, then we present the studied S-boxes, before presenting the results of our security analysis. In the remainder of the paper, we denote the i-th original secret input and output by x_i and y_i, respectively. The j-th share of an unmasked input x_i is denoted by $x_{i,j}$; the same convention is applied to outputs.

5.1 Gadgets

We consider the first-order masked implementations of the two described SKINNY S-boxes and the AES S-box, and the second-order masked implementation of the 4-bit SKINNY S-box. We follow the generalization of the principles of Private Circuits from [ISW03] by using gadgets to mask the three logical gates in $\mathbb{F}_2$ for our circuits: the two-input addition XOR, the two-input multiplication AND and the one-input negation NOT.

Masking *linear operations* in a secure manner is trivial. Let x_0, x_1 be two secret variables in $\mathbb{F}_2$. We can trivially mask at the d-th order their addition $y_0 = x_0 \oplus x_1$ by the following share-wise operations: $\forall i \in [0, d]_{\mathbb{N}},\ y_{0,i} = x_{0,i} \oplus x_{1,i}$. In a similar fashion, the negation of a secret value $x_0 \mapsto x_0 \oplus 1 = y_0$ can be done by computing $\forall i \in [1, d]_{\mathbb{N}},\ y_{0,i} = x_{0,i}$ and $y_0 = x_{0,0} \oplus 1$. Both gadgets do not need to be refreshed (i.e., use extra random bits).

The challenge lies in designing secure *nonlinear gadgets*. For the multiplication in $\mathbb{F}_2$, we consider four popular d-probing composable secure gadgets: DOM [GMK16], HPC1 [CGLS21], HPC2 [CGLS21] and HPC3 [KM22b]. We refer to [KSM20, KM22b] for their detailed security comparison and to [KM22b, CSV24, BLH+25] for a comparison w.r.t. hardware design metrics. We highlight that, at the first order, the DOM and HPC2 gadgets require one refresh bit per gadget, whereas HPC1 and HPC3 require two bits. At the second order, the number of refresh bits per gadget increases to three for DOM and HPC2, five for HPC1, and six for HPC3.

5.2 SKINNY S-Boxes

The SKINNY family by Beierle et *al.* [BJK+16] offers a family of lightweight tweakable block ciphers with two 4-bit and 8-bit S-boxes. We consider these as our case studies. As the SKINNY S-boxes are compact by design, we use HADES to create a straightforward pipelined masked implementation of the component Boolean functions. Both S-boxes are masked at the first order ($d = 1$) with the four multiplication gadgets, and the 4-bit S-box variant at the second order ($d = 2$) with the same gadgets for this extended version.

5.2.1 4-bit SKINNY S-box

The smallest S-box of SKINNY operates on four bits. The first-order masked hardware implementation of this S-box contains 4 AND, 4 XOR and 8 NOT gadgets. It requires 4 random variables to encode the input shares. For refreshing the multiplication gadgets, a total of 4 random variables is required for the implementations with the DOM and HPC2 gadgets, while 8 random variables are needed for those with the HPC1 and HPC3 gadgets. In total, the first-order masked implementations with the DOM and HPC2 gadgets need 8 bits of randomness, whereas the ones with the HPC1 and HPC3 gadgets require 12 bits.

[4] The files for the first-order masked circuits are included in the artifact of the TCHES paper, while those for the circuits masked at both probing orders are available at https://github.com/anna-guinet/lfsr-boolean-masking

For the extended paper, we consider the second-order masked hardware implementations of this S-box. The number of gadgets stays the same, but it requires 8 random variables to encode the input shares. For the multiplication gadgets, the total numbers of refreshing random variables are as follows: 12 for the DOM and HPC2 implementations, 20 for the HPC1 implementation, and 24 for the HPC3 implementation. Overall, the second-order masked designs need 20 bits of randomness with the DOM and HPC2 gadgets, 28 bits with the HPC1 gadget, and 32 bits with the HPC3 gadget.

5.2.2 8-bit SKINNY S-box

The other SKINNY S-box variant operates on eight bits. The first-order masked hardware implementation of this S-box is composed of 8 AND, 8 XOR and 16 NOT gadgets. The encoding of the input takes 8 random variables. To refresh the multiplication gadgets, 8 random variables are required for the implementations with the DOM and HPC2 gadgets, whereas 16 are required for those with the HPC1 and HPC3 gadgets. In total, 16 random variables are used for the first-order masked implementation with the DOM and HPC2 gadgets, while 24 are employed for those with the HPC1 and HPC3 gadgets.

5.3 AES S-Box

The AES block cipher was standardized in 2001 by the NIST [AES01] and is a popular cryptographic algorithm. AES uses an 8-bit S-box, which can be implemented in different ways. For a review of the hardware-implementation of the AES S-box with respect to hardware metrics and SCA security, we refer to the recent survey of Curlin et *al.* [CHCL25]. We choose to focus on the rather compact S-box design of Boyar and Peralta [BP12] by considering its first-order masked pipelined implementation with the HPC3 gadget, as generated by HADES. In total, this masked implementation is composed of 34 AND, 94 XOR and 4 NOT gadgets; and it requires 8 bits of randomness to encode the inputs and 68 bits of randomness to refresh the inner computations in the HPC3 gadgets.

5.4 Use of LFSR Outputs

In all implementation examples, we use one LFSR as a source of randomness for encoding and refreshing. To mask any of the S-boxes at any order, we first encode the original inputs x_i by drawing the first output bits of the chosen LFSR. Indeed, for a primary input x_i, the encoding is given by $x_{i,0} = x_i \oplus x_{i,1} \oplus \ldots \oplus x_{i,d}$ and, for $j \in [1, d]_{\mathbb{N}}$, $x_{i,j}$ is drawn from either a Galois or a Fibonacci LFSR. At the first-order ($d = 1$), the output bits from the chosen LFSR are used to create the shares $x_{i,1}$ for each i-th secret input: $x_{i,1} = s_{n-1}^{i+1}$ (Galois LFSR) or r_0^{i+1} (Fibonacci LFSR), where i ranges from 0 to the S-box size minus one. For the second-order ($d = 2$), we first create all the shares $x_{i,1}$ then all the shares $x_{i,2}$ by drawing LFSR output bits, for the secret inputs x_i in ascending order of i. After that, additional outputs bits are drawn from the LFSR to refresh the multiplication gadgets, following the same order provided by the circuits generated by HADES. For example, if we use an n-bit Galois LFSR for the first-order masked 4-bit SKINNY S-box implementation, we draw $(s_{n-1}^1, s_{n-1}^2, s_{n-1}^3, s_{n-1}^4)$ to encode the sharing of the four secret inputs. Then, according to HADES, the first multiplication to be processed is $(x_3 \oplus 1)(x_2 \oplus 1)$ in the expression of the output bit y_3 (with most significant bit ordering); therefore, to refresh this particular gadget at the first order, we draw the next LFSR output s_{n-1}^5 for a DOM or a HPC2 gadget, or (s_{n-1}^5, s_{n-1}^6) for HPC1 or HPC3. Then, we drawn the next LFSR outputs for the following multiplications in the circuit. We refer to the supplementary material for a detailed description of the use of the LFSR outputs (see Footnote 4).

5.5 Verification Results

We report the results of our theoretical analysis of the first-order masked 4-bit SKINNY S-box with the DOM, HPC1, HPC2 and HPC3 gadgets in Table 1, and the first-order masked 8-bit SKINNY S-box with the HPC3 gadget in Table 2. We provide the outcome of the analysis for the first-order masked 8-bit SKINNY S-box with the DOM, HPC1 and HPC2 gadgets in Table 8 in Section B.2. For the first-order masked AES S-box, we present two complementary tables for LFSRs of length $n \in [4, 20]_{\mathbb{N}}$ due to time: Table 9 and Table 10 (see Section B.2) list the LFSRs for which non-zero correlation and zero correlation values have been found with linear approximations containing only the original secret input, respectively. In this extended version of the paper, the verification for the 4-bit SKINNY S-box masked at the second order is added with the four multiplication gadgets, as detailed later in this section. Because we probe the input shares, the refreshing variables, and the inputs and outputs of the gadgets—including the intermediate computations of the multiplication gadgets—the outcome of our verification in this section is therefore glitch-extended (see Section 4.2.3).

We write $\mathcal{P}_n$ for the set of all characteristic polynomials of degree n, taken from the list compiled by Koopman [Koo]. $\mathcal{P}_n^{\neq 0}$ and $\mathcal{P}_n^{=0}$ are subsets of $\mathcal{P}_n$ for which non-zero correlations and zero correlation with linear combinations of original secret values were found for the given scheme, respectively. It means that a characteristic polynomial in $\mathcal{P}_n^{\neq 0}$ breaks d-probing security for a particular scheme when used as described in Section 5.4, but one in $\mathcal{P}_n^{=0}$ does not. For conciseness, we report the polynomials found in $\mathcal{P}_n^{\neq 0}$ and $\mathcal{P}_n^{=0}$ in hexadecimal notation, where each bit represents the coefficient of a polynomial over $\mathbb{F}_2$, starting from the most significant bit (e.g., $\texttt{4E}_{16} = \texttt{1001110}_2$ represents $p(X) = X^7 + X^4 + X^3 + X^2 + 1$). Along with the polynomials $p \in \mathcal{P}_n^{\neq 0}$, we report in parentheses their highest absolute non-zero correlation value $\max_p|\mathrm{Cor}|$ found with a secret-only linear approximation on the $-\log_2$ scale (e.g., $-\log_2(1/8) = 3$).

For the first-order masked 4-bit SKINNY S-box, we report the correlation values for an LFSR length up to the number of required random bits in the circuit plus two (i.e., we run the analysis for $n \in [4, 10]_{\mathbb{N}}$ for the implementations masked with the DOM and HPC2 gadgets, and $n \in [4, 14]_{\mathbb{N}}$ with the HPC1 and HPC3 gadgets). For the first-order masked 8-bit SKINNY S-box, we provide the outcomes of the verification for all gadgets for LFSRs up to length 20 due to time. Finally, for the first-order masked AES S-box, we provide the verification results for LFSRs up to length 20 in Section B.2. Since the number of characteristic polynomials grows rapidly with the length n, we limited our analysis to the first 500 or 200 LFSRs of a given length n according to the list provided in [Koo], where the characteristic polynomials are sorted in ascending numeric order in base 16. This is highlighted by an asterisk mark $*$ in the column $|\mathcal{P}_n|$ in Table 2, and Table 8, Table 9 and Table 10 from Section B.2.

In Table 1, for first-order masked 4-bit SKINNY S-box implementations, we group the results for the DOM and HPC2 gadgets, since they have the same characteristic polynomials in $\mathcal{P}_n^{\neq 0}$ for degree $n \in [4, 10]_{\mathbb{N}}$ with the same absolute maximal non-zero correlations $\max_p|\mathrm{Cor}|$ (except for $\texttt{1B}_{16}$ when $n = 5$). However, in both circuits, we point out that different probes are compromising security and are approximated by distinct secret-only linear approximations with non-zero correlation. For the first-order implementations of the 8-bit SKINNY S-box, we also report that the same characteristic polynomials are first-probing insecure for the DOM and HPC2 gadgets for length $n \in [4, 20]_{\mathbb{N}}$ where the absolute maximal non-zero correlation values are similar, except for $n = 5, 7, 11, 12$, as shown in Table 8 in Section B.2. For both S-boxes, those results with DOM and HPC2 differ from the ones of the implementations with HPC1 and HPC3, which do not exhibit mutual similarity either. Therefore, the ability of an LFSR to ensure glitch-extended first-order probing security for a masked SKINNY S-box is gadget-dependent.

For both SKINNY S-boxes at the first masking order, based on Table 1, Table 2, and

Table 1: First-order masked 4-bit SKINNY S-box with DOM, HPC1/2/3 gadgets.

LFSR n	**Gadget**	$\lvert\mathcal{P}_n^{\neq 0}\rvert/\lvert\mathcal{P}_n\rvert$	$p \in \mathcal{P}_n^{\neq 0}$ **in hex.** ($\max_p\lvert\mathrm{Cor}\rvert$ **in** $-\log_2$ **scale)**
4	DOM, HPC2	2 / 2	`9` (1), `C` (1)
	HPC1	1 / 2	`9` (1)
	HPC3	2 / 2	`9` (3), `C` (1)
5	DOM, HPC2	4 / 6	`12` (3), `17` (2), `1E` (3), DOM: `1B` (2), HPC2: `1B` (1)
	HPC1	4 / 6	`14` (1), `17` (3), `1B` (2), `1E` (1)
	HPC3	5 / 6	`12` (2), `14` (2), `17` (2), `1B` (3), `1D` (2)
6	DOM, HPC2	1 / 6	`2D` (2)
	HPC1	4 / 6	`21` (2), `30` (1), `36` (1), `39` (3)
	HPC3	3 / 6	`21` (1), `2D` (1), `30` (1)
7	DOM, HPC2	2 / 18	`48` (2), `5C` (3)
	HPC1	5 / 18	`47` (2), `48` (1), `55` (3), `69` (3), `7B` (2)
	HPC3	9 / 18	`4E` (1), `53` (2), `5F` (2), `60` (1), `69` (1), `77` (2), `78` (3), `7B` (1), `7E` (3)
8	DOM, HPC2	0 / 16	
	HPC1	1 / 16	`95` (2)
	HPC3	2 / 16	`96` (3), `B2` (3)
9	DOM, HPC2	0 / 48	
	HPC1	1 / 48	`191` (2)
	HPC3	5 / 48	`119` (1), `152` (2), `198` (2), `19D` (2), `1E6` (3)
10	DOM, HPC2	0 / 60	
	HPC1, HPC3	0 / 60	
11	HPC1, HPC3	1 / 176	HPC1: `415` (2), HPC3: `47A` (3)
12	HPC1, HPC3	0 / 144	
13	HPC1, HPC3	0 / 630	
14	HPC1, HPC3	0 / 756	

Table 8 in Section B.2, the larger the LFSR length n, the less likely we are to find a polynomial in $\mathcal{P}_n^{\neq 0}$ (i.e., an 'insecure' LFSR). We observe a similar trend for the AES S-box (see Table 9 and Table 10 in Section B.2). In addition, for the SKINNY S-boxes, we see the absolute maximal correlation values range between 1/2 and 1/32. For the first-order masked 4-bit SKINNY S-box, no LFSR of length equal to, or greater than, the number of required random bits for the masked implementation, is found to violate glitch-extended first-probing security in Table 1.

The verification outcome is interesting for the 4-bit SKINNY S-box with the HPC3 gadget at first order. First, we remark that, for this S-box, the lowest absolute maximal non-zero correlation value found is 1/8 across all gadgets. Concerning HPC3, we observe that one of the two 4-bit LFSRs breaks first-probing security with this rather low $\max_p|\mathrm{Cor}| = 1/8$ (while one of the two is labeled secure for HPC1). We found at least five insecure 7-bit and 9-bit LFSRs with $\max_p|\mathrm{Cor}| \geq 1/8$, but only two 8-bit ones with $\max_p|\mathrm{Cor}| = 1/8$. No 10-bit LFSRs is found insecure (same for HPC1), and only one 11-bit LFSR with this relatively low $\max_p|\mathrm{Cor}| = 1/8$ is labeled insecure. It means that, in theory, we have potentially 449 secure characteristic polynomials to choose from of degree $n \in [5, 12]_{\mathbb{N}}$. We recall that this circuit requires 12 bits of randomness to encode and refresh. For this masked scheme, we provide a detailed explanation of our framework for $X^7 + X^6 + 1$ and

$d = 1$, and $X^7 + X^4 + 1$ and $d = 2$, in Section B.1. However, some preliminary intuition suggests that a 5-bit LFSR (or even a 4-bit one for HPC1) may not be secure in practice, despite the successful analysis: The register length appears too small, raising the question of whether a practical threshold exists for the LFSR length. We further explore this design with a practical evaluation in the following section to validate our verification and intuition.

Table 2: First-order masked 8-bit SKINNY S-box with HPC3 gadget.

LFSR n	$\lvert\mathcal{P}_n^{\neq 0}\rvert / \lvert\mathcal{P}_n\rvert$	$p \in \mathcal{P}_n^{\neq 0}$ **in hex.** ($\max_p \lvert\mathrm{Cor}\rvert$ **in** $-\log_2$ **scale)**
4	2 / 2	`9` (1), `C` (1)
5	6 / 6	`12` (1), `14` (1), `17` (1), `1B` (2), `1D` (1), `1E` (1)
6	6 / 6	`21` (2), `2D` (1), `30` (5), `33` (2), `36` (1), `39` (3)
7	16 / 18	`41` (5), `44` (4), `47` (4), `48` (2), `4E` (1), `53` (1), `5C` (3), `5F` (2), `60` (1), `65` (2), `69` (4), `6A` (1), `72` (2), `78` (2), `7B` (3), `7E` (3)
8	5 / 16	`96` (1), `AF` (4), `B1` (4), `B4` (3), `F3` (2)
9	16 / 48	`110` (1), `119` (2), `13B` (3), `13E` (4), `151` (3), `152` (2), `167` (3), `16D` (1), `17A` (3), `17C` (2), `191` (3), `198` (2), `1B5` (5), `1B6` (1), `1C7` (3), `1E3` (2)
10	8 / 60	`240` (2), `245` (3), `291` (2), `2C2` (3), `312` (2), `33C` (2), `3A3` (4), `3AC` (3)
11	9 / 176	`44A` (2), `56A` (3), `581` (4), `6E4` (4), `713` (3), `78C` (3), `79B` (4), `7C8` (3), `7CB` (2)
12	2 / 144	`CDE` (3), `E26` (3)
13	14 / 630	`11AC` (2), `13C1` (3), `14AA` (1), `1616` (4), `1675` (5), `17B2` (3), `1860` (2), `18F6` (3), `1A3D` (3), `1C01` (3), `1CCB` (2), `1DFF` (2), `1E7E` (4), `1F34` (1)
14	8 / 756	`2604` (2), `2A01` (2), `2D21` (4), `2E5F` (1), `2F6D` (2), `3536` (3), `36D8` (2), `3D40` (4)
15	3 /1800	`7BFF` (1), `7E0F` (2), `7FF7` (2)
16	5 /2048	`801C` (2), `9861` (3), `C00A` (3), `CF48` (5), `FC00` (2)
17	0 /500*	
18	0 /500*	
19	0 /200*	
20	0 /200*	

Second-order verification. We provide the security verification results for the 4-bit SKINNY S-box masked at the second order with the DOM, HPC1, HPC2 and HPC3 gadgets. We run the theoretical security analysis for LFSRs of length $n \in [4, 18]_{\mathbb{N}}$, where we limited our analysis to the first 500 and 300 LFSRs for $n = 17$ and $n = 18$, respectively. Similarly to the results presented for the AES S-box, we display the LFSRs for which non-zero and zero correlation values have been reported with secret-only linear approximations in separate tables. For the LFSRs with non-zero correlation values ($p \in \mathcal{P}_n^{\neq 0}$), we present the verification results with the HPC3 gadget in Table 3. We show the polynomials $p \in \mathcal{P}_n^{\neq 0}$ for DOM, HPC1 and HPC2 in Table 11 in Section B.2. In Table 12, we list the polynomials $p \in \mathcal{P}_n^{=0}$ for all four multiplication gadgets. For readability purposes, in Table 3 and Table 11, we only present a selection of the characteristic polynomials in $\mathcal{P}_n^{\neq 0}$ with the highest and lowest absolute maximal correlation values for a given n.

Similarly to the first-order verification results, we observe that the higher the LFSR length, the less likely we are to find an LFSR that breaks second-order security for those masked implementations. In addition, no LFSR of length $4 \leq n \leq 12$ is labeled as secure, which is a higher threshold than that observed for the first-order masked implementation

of this S-box, as one would expect. Moreover, the security results for the second-order implementations are clearly gadget-dependent as they exhibit no apparent similarity.

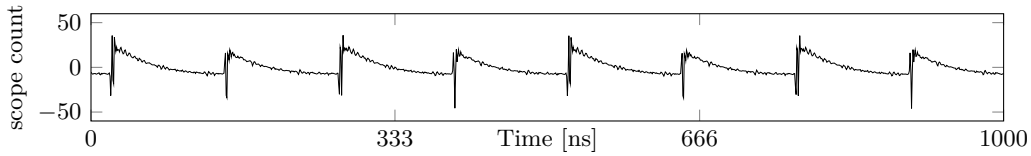
Table 3: Second-order masked 4-bit SKINNY S-box with HPC3 gadget.

LFSR n	**Gadget**	$\lvert\mathcal{P}_n^{\neq 0}\rvert / \lvert\mathcal{P}_n\rvert$	$p \in \mathcal{P}_n^{\neq 0}$ **in hex. with highest and lowest** ($\max_p \lvert\mathrm{Cor}\rvert$ **in** $-\log_2$ **scale) for a given** n
4	HPC3	2 / 2	Highest (0) | Lowest (0): 9, C
5	HPC3	6 / 6	Highest (0) | Lowest (0): 12, 14, 17, 1B, 1D, 1E
6	HPC3	6 / 6	Highest (0)| Lowest (0): 21, 2D, 30, 33, 36, 39
7	HPC3	18 / 18	Highest (0): 41, 44, 48, 4E, 53, 55, 5F, 60, 65, 69, 6A, 78, 7E | Lowest (1): 47, 5C, 72, 77, 7B
8	HPC3	16 / 16	Highest (0): 95, 96, AF, B1, B2, B8, E7, FA Lowest (1): 8E, A6, B4, C3, C6, D4, E1, F3
9	HPC3	48 / 48	Highest (0): 108, 119, 151, 152, 15E, 167, 16D, 17A, 1A7, 1B5, 1B9, 1C7, 1E3, 1F1 | Lowest (1) first 12: 10D, 110, 116, 12C, 12F, 134, 137, 13B, 13E, 143, 14A, 157
10	HPC3	60 / 60	Highest (0): 216, 232, 2CB, 31B, 344, 3A6, 3C6, 3D8, 3F9 | Lowest (1) first 16: 204, 20D, 213, 237, 240, 245, 262, 26B, 273, 279, 27F, 286, 28C, 291, 298, 29E
11	HPC3	176 / 176	Highest (0): 402, 44F, 475, 500, 70E, 7E9 Lowest (2): 5DE
12	HPC3	144 / 144	Highest (0): 8B0, B23, C4B, CDE, D7A, EC4 Lowest (2): 83E, 8B5, 965, 9D1, A27, BA2, BD6, CBD, CD8, CF3, D15, E5D, F17, F33
13	HPC3	630 / 630	Highest (0): 15CD, 169D, 172E, 1784, 1925, 1A34, 1D9A, 1F07, 1F34, 1F51, 1F7C, 1FC1, 1FF1 Lowest (2) first 14: 1013, 1037, 1052, 1057, 1070, 1086, 108A, 109D, 10B3, 10DF, 10E6, 10EF, 10F1, 111E
14	HPC3	756 / 756	Highest (0): 2A07, 36DE, 3A74, 3ECB | Lowest (3): 2850, 28AF, 2CD0, 2F23, 3517, 3995, 3E16, 3E80, 3F3A
15	HPC3	1800 /1800	Highest (0): 4008, 4218, 5556, 62AD, 6D6E, 7007, 71DB 74EB, 789A, 7C07, 7FF7 | Lowest (4): 5688
16	HPC3	2032 /2048	Highest (0): 810C, AFAA, F636, FE07, FE08 Lowest (4) first 14: 80A7, 85F2, 8CBA, 8D2B, 8E74, 9225, 95CC, 9627, 986E, 9B8C, 9CCC, 9DA2, 9DF7, 9EA4
17	HPC3	469 /500*	Highest (0): 10007, 10019, 1002A, 1020C, 10809, 10818 | Lowest (4) first 10: 10046, 101E1, 102F6, 10346, 103B6, 10534, 1055B, 10564, 10575, 105BC
18	HPC3	220 /300*	Highest (1) first 10: 2001F, 2006D, 20073, 20109, 20218, 20250, 20290, 204B2, 20A01, 20B03 Lowest (4) first 10: 20105, 2018B, 20346, 20394, 203A8, 20477, 204D2, 20638, 2073A, 207E2

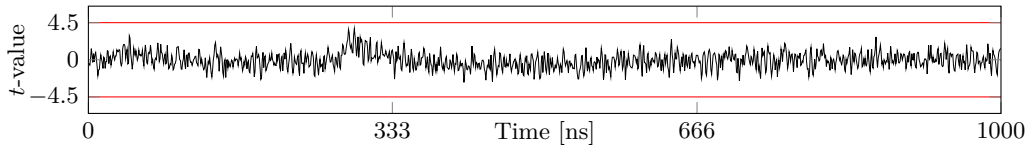
6 Practical Evaluation

To validate our findings, we perform an SCA on a Field Programmable Gate Array (FPGA) implementation of the first-order masked 4-bit SKINNY S-box using the HPC3 gadget, which was generated by HADES. In this extended version of the paper, we provide additional

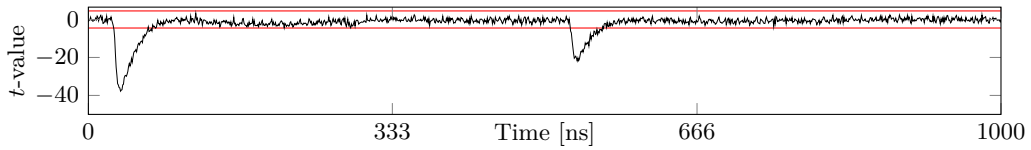
measurements for this S-box implementation as well as an SCA of the first-order masked 8-bit SKINNY S-box with the same setup. The results of both analyses are presented later in this section; we here present the evaluation for the 4-bit SKINNY S-box masked at the first order. For our practical evaluation, we follow the commonly used Test Vector Leakage Assessment (TVLA) methodology [SM15] and apply a fixed-vs-random t-test with a threshold of ± 4.5 to determine whether the use of a given LFSR is secure for this circuit. We expect that using characteristic polynomials from the set $\mathcal{P}_n^{\neq 0}$, reported in Table 1 (see Section 5.5) for the 4-bit SKINNY S-box, will exceed this threshold in the first-order t-test, indicating the potential presence of exploitable side-channel leakage. Conversely, we presume that the characteristic polynomials from the set $\mathcal{P}_n^{=0}$ remain below the t-test threshold. Note that, because we are analyzing a first-order masked design, peaks exceeding ± 4.5 in the second-order t-test are expected.



(a) Mean trace.



(b) First-order t-test results.



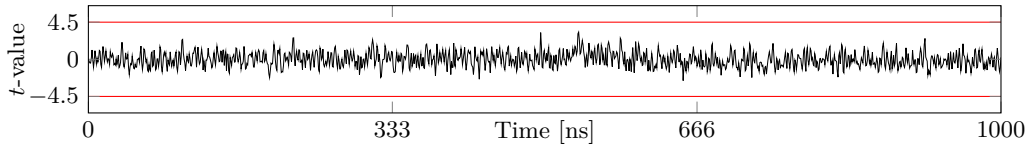
(c) Second-order t-test results.

Figure 1: No first-order t-test leakage using non-LFSR randomness for the first-order masked 4-bit SKINNY S-box (250 million traces).

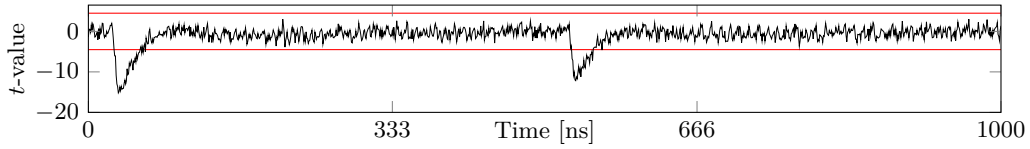
The measurement setup consists of a Sakura-G evaluation board [GIS14], which is specifically designed for low-noise side-channel evaluation and contains a Xilinx Spartan-6 target FPGA clocked at 4MHz. We amplify the voltage drop over the 1Ω shunt using a ZFL-2000GH+ Low Noise Amplifier (LNA) and digitize the power traces using a Spectrum M4 capture card at a sample rate of 1.25GS/s with 8-bit resolution. To ensure that there are no transitional effects between multiple secret values in the pipelined design, we wait for the current execution to finish and fully exit the S-box pipeline before starting the next measurement after an additional timeout. While clearing the pipeline, we always pass in fresh randomness to avoid any leakage due to transitions between different secret-dependent values under the same randomness.

To provide some assurance that the 4-bit SKINNY S-box implementation itself is secure in practice, we perform an initial measurement consisting of 250 million traces using randomness from an AES-128 instance in counter mode, which is seeded by the kernel randomness source of the control PC. A mean trace and the corresponding t-test results are shown in Fig. 1: As expected, no first-order leakage is visible, while clear second-order peaks can be observed.

For each tested characteristic polynomial, we then record 100 million fixed/random traces in total. During each individual execution, the studied LFSR is freshly seeded. The LFSR output is then used to create the initial sharing of the S-box inputs and to supply the necessary randomness for the circuit, employed as described in Section 5.4.

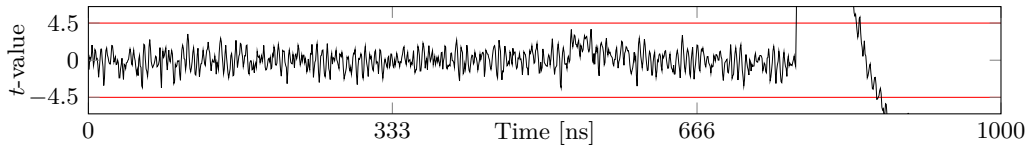


(a) First-order t-test results.

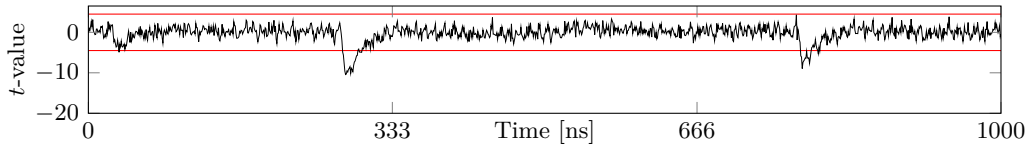


(b) Second-order t-test results.

Figure 2: No first-order t-test leakage using the Galois LFSR $X^7 + X^6 + X^3 + X + 1$ (65 in hex.) for the first-order masked 4-bit SKINNY S-box (100 million traces).



(a) First-order t-test results.



(b) Second-order t-test results.

Figure 3: First- and second-order t-test leakage using the Galois LFSR $X^7 + X^6 + 1$ (60 in hex.) for the first-order masked 4-bit SKINNY S-box (100 million traces).

We run several side-channel evaluations for LFSRs of length $n \in [5, 12]_{\mathbb{N}}$ in Galois mode for the first-order masked 4-bit SKINNY S-box. For each length n, we choose—if this configuration exists—one characteristic polynomial $p \in \mathcal{P}_n^{=0}$ ('secure' LFSR) and one $p \in \mathcal{P}_n^{\neq 0}$ ('insecure' LFSR) based on the findings in Table 1 from Section 5.5. Otherwise, we select one polynomial in the set $\mathcal{P}_n^{=0}$ if all LFSRs for a given n are secure. Then, we assess in practice whether the selected LFSRs in our studied circuit exceed the t-test threshold. We sum up this evaluation in Table 4, where we report the chosen polynomial p in hexadecimal representation, whether they are secure (✓) or not (✗) in theory w.r.t. first-probing security, and whether they are secure (✓) or not (✗) according to the first-order t-test in practice. Specifically, we show the t-test results of two 7-bit LFSRs: one defined by the characteristic polynomial $X^7 + X^6 + X^3 + X + 1$ (65 in hexadecimal representation) in Fig. 2 and another one by $X^7 + X^6 + 1$ (60 in hex. repr.) in Fig. 3.

We provide the remaining evaluation results of Table 4 for $n = 6, 8, 9, 10, 11, 12$ in Section B.4. As predicted by our verification, the first 7-bit LFSR does not pass the first-order t-test, while the second one does. We report the same behavior for our chosen

Table 4: Evaluated LFSRs in theory and practice (100 million traces) as PRNGs for first-order masked 4-bit SKINNY S-box with HPC3 gadget.

	LFSR Evaluation													
n	5		6		7		8		9		10	11		12
p in hex.	1E	12	33	2D	65	60	8E	96	110	119	20D	458	47A	E08
Theory	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✓	✗	✓
Practice	✗	✗	✗	✗	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓

LFSRs of length $n = 9, 10, 12$. For the selected LFSRs of length 5 and 6 in Table 4, we observe that, in each case, both do not pass the first-order t-test: We conclude that, for this circuit, the minimum LFSR length is $n = 7$ to ensure glitch-extended first-order-probing security, validating the intuition of a minimal entropy threshold in this case. We tested those same LFSRs of length 6 and 7 in Fibonacci mode: The outcome is identical (see Section B.4). Interestingly, for $n = 8$ (resp. $n = 11$), the insecure LFSR with characteristic polynomial 96 (resp. 47A) in hex. passes the first-order t-test. We recall that for $n = 8$ (resp. $n = 11$), we found only two (resp. one) insecure LFSR(s) with the relatively low absolute maximal correlation value of 1/8 (see Table 1).

These practical results either do not show the theoretical leakage (as may occur) or suggest a relation between the LFSR length n, the correlation value and/or the ratio of insecure LFSRs found for a given n that would ensure practical security, which is not currently captured by the d-probing model itself. Along with the 6-bit LFSR evaluation outcome, further investigation should be conducted to capture these relations and rule out (or in) some LFSR lengths. From those evaluations, we deduct that there are at least 445 possible LFSRs to choose from, for $n \in [5, 12]_{\mathbb{N}}$, to ensure practical security for this masked scheme and number of traces. Moreover, we practically showcase that a masking scheme requiring 12 bits of randomness for sharing and refreshing can pass a first-order t-test with an LFSR whose length is lower than 12, and it can be efficiently leveraged to lower the implementation cost of masking.

4-bit SKINNY S-box: $n = 8$ and $n = 11$. For this extended version, we further investigate the practical evaluation outcomes for LFSRs of length $n = 8$ and $n = 11$. Indeed, in these two cases, we observe that the first-order t-tests show no leakage for the insecure LFSRs (i.e., characteristic polynomials 96 and 47A in hex.) for 100 million traces. Therefore, we run additional evaluations of 250 million traces each. We sum up our findings in Table 5 and we display the t-test results in Section B.4.1. For $n = 8$, the practical results for 250 million traces match the theoretical verification: It means that the insecure LFSR with characteristic polynomial 96 is then found insecure, while the secure LFSR 8E remains secure. However, for $n = 11$, both LFSRs labeled as secure and insecure in theory pass the first-order t-test for 250 million traces. It confirms some relation between the LFSR length and the correlation value, as noted previously.

Table 5: Additional theoretical and practical comparison of selected LFSRs for $n \in \{8, 11\}$ as PRNGs for a first-order masked 4-bit SKINNY S-box with HPC3 gadget.

		LFSR Evaluation			
n		8		11	
p in hex.	# of traces	8E	96	458	47A
Theory		✓	✗	✓	✗
Practice	100 million	✓	✓	✓	✓
	250 million	✓	✗	✓	✓

8-bit SKINNY S-box. For the extended version of this paper, we include the practical evaluation of the 8-bit SKINNY S-box masked at the first-order with the HPC3 multiplication gadget. Indeed, we run several side-channel evaluations for LFSRs of length $n \in [6, 16]_{\mathbb{N}}$ in Galois mode, in a similar fashion to the 4-bit SKINNY S-box. First, we ensure that the first-order masked design is secure in practice (see Fig. 22 in Section B.5). After that, based on the verification results of Table 2 from Section 5.5, we select one secure LFSR ($p \in \mathcal{P}_n^{=0}$) and one insecure LFSR ($p \in \mathcal{P}_n^{\neq 0}$), except for $n = 6$ where no secure LFSR is found. Then, for each case, we collect 100 million traces, and we check whether the selected LFSRs exceed the first-order t-test threshold. We summarize our theoretical and practical findings in Table 6. The results are straightforward for this 8-bit S-box: Our theoretical verification matches the practical evaluations for $n = 6, 7, 8, 9, 10, 11, 13, 14, 16$. Curiously, for $n = 12$ (resp. $n = 15$), the insecure LFSR with characteristic polynomial `E26` (resp. `7BFF`) in hex. passes the first-order t-test in practice. We observe the same phenomenon for particular LFSR lengths as seen with the 4-bit SKINNY S-box implementation. It is noteworthy that, only for $n = 12$ (resp. $n = 15$), the ratio of theoretical insecure LFSRs for this length is slightly lower than the previous and next integers, according to Table 2. We encourage additional investigation to better account for those cases.

Table 6: Evaluated LFSRs in theory and practice (100 million traces) as PRNGs for a first-order masked 8-bit SKINNY S-box with HPC3 gadget.

	LFSR Evaluation										
n	6	7		8		9		10		11	
p in hex.	`2D`	`55`	`60`	`95`	`96`	`168`	`16D`	`262`	`240`	`446`	`44A`
Theory	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗
Practice	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗
n	12		13		14		15		16		
p in hex.	`A8E`	`E26`	`1159`	`1F34`	`2269`	`2A01`	`4144`	`7BFF`	`8260`	`FC00`	
Theory	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	
Practice	✓	✓	✓	✗	✓	✗	✓	✓	✓	✗	

7 Conclusion

In this work, we present a verification framework to formally analyze the joint d-(robust)-probing security of a masking scheme with a PRNG of simple algebraic description. For our use cases, we consider the randomness generation of a single LFSR for both encoding and refreshing, though our framework also supports other configurations. We demonstrate that both LFSR modes, Galois and Fibonacci, lead to the same security verification assessments. We showcase our method by verifying three S-boxes masked with different gadgets at the first- and second-order, for a total of 13 masked circuits. It highlights that different gadgets lead to different security outcomes for the same circuit. Furthermore, we showed that the larger the LFSR-length, the less likely it is to find an insecure configuration. We validated our theoretical verification with practical evaluations on an FPGA. Through this, we prove and evaluate that we can securely use an LFSR whose length is smaller than the required number of random bits for a given scheme. This outcome demonstrates that we can possibly further reduce the cost of masking by having small LFSRs for randomness generation, which are cost-efficient PRNGs in hardware. In addition, we find that, in practice, there is a minimal entropy threshold below which no LFSR-length is secure for a small 4-bit S-box, and possible cases that exhibit security for a 4-bit and an 8-bit S-box, regardless of the d-probing security analysis. Our intuition is that some additional information is

not captured by the d-probing model to formalize these observations, plausibly linking the correlation value/agreement probability with the LFSR state length: We encourage future investigation in this direction. Our work could also be further extended by considering other PRNGs for theoretical and practical evaluations. Another future direction is how to efficiently run the presented method to a full cryptographic algorithm or higher-order masking schemes. Overall, we aim to extend the discussion around the formal verification of a masking scheme with a PRNG by having provided new insights about the use of LFSRs for Boolean masking.

Acknowledgments

We would like to express our gratitude to Lorenzo Casalino and Jan Richter-Brockmann for their early support. We would also like to thank the TCHES reviewers for their comments and suggestions. Jan Schoone was affiliated with Radboud University from the development until the acceptance of the TCHES paper, and with Leiden University during the preparation and publication of this extended version. Anna Guinet and Tim Güneysu have been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972. Niklas Höher, Dina Hesse and Tim Güneysu have been supported by the German Federal Ministry of Research, Technology and Space through the projects SASPIT (16KIS1859), DI-OCDCpro (16ME0942) and DI-SIGN-HEP (16KIS2073), and have been funded by the European Commission under the grant agreement number 101070374 (CONVOLVE).

References

[AES01] Advanced Encryption Standard (AES). National Institute of Standards and Technology, NIST FIPS PUB 197, U.S. Department of Commerce, November 2001.

[BBC+19] Gilles Barthe, Sonia Belaïd, Gaëtan Cassiers, Pierre-Alain Fouque, Benjamin Grégoire, and François-Xavier Standaert. maskVerif: Automated verification of higher-order masking in presence of physical defaults. In Kazue Sako, Steve Schneider, and Peter Y. A. Ryan, editors, *ESORICS 2019, Part I*, volume 11735 of *LNCS*, pages 300–318. Springer, Cham, September 2019.

[BDRS22] Tim Beyne, Siemen Dhooghe, Adrián Ranea, and Danilo Sijacic. A low-randomness second-order masked AES. In Riham AlTawy and Andreas Hülsing, editors, *SAC 2021*, volume 13203 of *LNCS*, pages 87–110. Springer, Cham, September / October 2022.

[BDZ20] Tim Beyne, Siemen Dhooghe, and Zhenda Zhang. Cryptanalysis of masked ciphers: A not so random idea. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part I*, volume 12491 of *LNCS*, pages 817–850. Springer, Cham, December 2020.

[BEB18] Robert G Brown, Dirk Eddelbuettel, and David Bauer. Dieharder. *Duke University Physics Department Durham, NC*, 27708, 2018.

[BFG+25] Christof Beierle, Jakob Feldtkeller, Anna Guinet, Tim Güneysu, Gregor Leander, Jan Richter-Brockmann, and Pascal Sasdrich. INDIANA - verifying (random) probing security through indistinguishability analysis. In Serge Fehr and Pierre-Alain Fouque, editors, *EUROCRYPT 2025, Part VIII*, volume 15608 of *LNCS*, pages 33–63. Springer, Cham, May 2025.

[BGI+18] Roderick Bloem, Hannes Groß, Rinat Iusupov, Bettina Könighofer, Stefan Mangard, and Johannes Winter. Formal verification of masked hardware implementations in the presence of glitches. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 321–353. Springer, Cham, April / May 2018.

[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 Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 123–153. Springer, Berlin, Heidelberg, August 2016.

[BLH+25] Fabian Buschkowski, Georg Land, Niklas Höher, Jan Richter-Brockmann, Pascal Sasdrich, and Tim Güneysu. HADES: Automated hardware design exploration for cryptographic primitives. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2025(4):1–45, 2025.

[BMRT22] Sonia Belaïd, Darius Mercadier, Matthieu Rivain, and Abdul Rahman Taleb. IronMask: Versatile verification of masking security. In *2022 IEEE Symposium on Security and Privacy*, pages 142–160. IEEE Computer Society Press, May 2022.

[BP12] Joan Boyar and René Peralta. A small depth-16 circuit for the AES S-box. In *SEC*, volume 376 of *IFIP Advances in Information and Communication Technology*, pages 287–298. Springer, 2012.

[BRS+10] Lawrence Bassham, Andrew Rukhin, Juan Soto, James Nechvatal, Miles Smid, Stefan Leigh, Mark Levenson, Mark Vangel, Nathanael Heckert, and David Banks. A statistical test suite for random and pseudorandom number generators for cryptographic applications, 2010-09-16 2010.

[Can06] Christophe De Cannière. Trivium: A stream cipher construction inspired by block cipher design principles. In Sokratis K. Katsikas, Javier Lopez, Michael Backes, Stefanos Gritzalis, and Bart Preneel, editors, *ISC 2006*, volume 4176 of *LNCS*, pages 171–186. Springer, Berlin, Heidelberg, August / September 2006.

[Can25] Anne Canteaut. *Linear Feedback Shift Register*, pages 1428–1431. Springer Nature Switzerland, Cham, 2025.

[CGLS21] Gaëtan Cassiers, Benjamin Grégoire, Itamar Levi, and François-Xavier Standaert. Hardware private circuits: From trivial composition to full verification. *IEEE Trans. Computers*, 70(10):1677–1690, 2021.

[CGZ20] Jean-Sébastien Coron, Aurélien Greuet, and Rina Zeitoun. Side-channel masking with pseudo-random generator. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 342–375. Springer, Cham, May 2020.

[CHCL25] Phaedra Sophia Curlin, Jeff Heiges, Calvin Chan, and Tamara Silbergleit Lehman. A survey of hardware-based AES Sboxes: Area, performance, and security. *ACM Comput. Surv.*, 57(9), April 2025.

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 398–412. Springer, Berlin, Heidelberg, August 1999.

[CMM+24] Gaëtan Cassiers, Loïc Masure, Charles Momin, Thorben Moos, Amir Moradi, and François-Xavier Standaert. Randomness generation for secure hardware masking - unrolled trivium to the rescue. *CiC*, 1(2):4, 2024.

[CSV24] Gaëtan Cassiers, François-Xavier Standaert, and Corentin Verhamme. Low-latency masked gadgets robust against physical defaults with application to ascon. *IACR TCHES*, 2024(3):603–633, 2024.

[DDF14] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 423–440. Springer, Berlin, Heidelberg, May 2014.

[DGV95] Joan Daemen, René Govaerts, and Joos Vandewalle. Correlation matrices. In Bart Preneel, editor, *FSE'94*, volume 1008 of *LNCS*, pages 275–285. Springer, Berlin, Heidelberg, December 1995.

[DH] Chris Doty-Humphrey. Practrand. https://pracrand.sourceforge.net/. Accessed: 03-Jan-2026.

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

[DSM22] Siemen Dhooghe, Aein Rezaei Shahmirzadi, and Amir Moradi. Second-order low-randomness d + 1 hardware sharing of the AES. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 815–828. ACM Press, November 2022.

[FGM+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 TCHES*, 2018(3):89–120, 2018.

[FYC24] Cameron Foreman, Richie Yeung, and Florian J. Curchod. Statistical testing of random number generators and their improvement using randomness extraction. *CoRR*, abs/2403.18716, 2024.

[GHP+21] Barbara Gigerl, Vedad Hadzic, Robert Primas, Stefan Mangard, and Roderick Bloem. Coco: Co-design and co-verification of masked software implementations on CPUs. In Michael Bailey and Rachel Greenstadt, editors, *USENIX Security 2021*, pages 1469–1468. USENIX Association, August 2021.

[GIS14] Hendra Guntur, Jun Ishii, and Akashi Satoh. 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*, pages 271–274. IEEE, 2014.

[GKM+24] Barbara Gigerl, Franz Klug, Stefan Mangard, Florian Mendel, and Robert Primas. Smooth passage with the guards: Second-order hardware masking of the AES with low randomness and low latency. *IACR TCHES*, 2024(1):309–335, 2024.

[GMK16] Hannes Groß, Stefan Mangard, and Thomas Korak. Domain-Oriented Masking: Compact masked hardware implementations with arbitrary protection order. In *TIS@CCS*, page 3. ACM, 2016.

[Gol67] Solomon W. Golomb. *Shift Register Sequences*. Holden-Day, 1967.

[GP99] Louis Goubin and Jacques Patarin. DES and differential power analysis (the "duplication" method). In Çetin Kaya Koç and Christof Paar, editors, *CHES'99*, volume 1717 of *LNCS*, pages 158–172. Springer, Berlin, Heidelberg, August 1999.

[HB21] Vedad Hadzic and Roderick Bloem. COCOALMA: A versatile masking verifier. In *FMCAD*, pages 1–10. IEEE, 2021.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 463–481. Springer, Berlin, Heidelberg, August 2003.

[JMB25] Vahid Jahandideh, Bart Mennink, and Lejla Batina. Probing secure composability without fresh randomness: Theory and application to Ascon. *IACR TCHES*, 2025(4):972–1009, Sep. 2025.

[Jou09] Antoine Joux. *Algorithmic cryptanalysis*. Chapman and Hall/CRC, 2009.

[JSB25] Vahid Jahandideh, Jan Schoone, and Lejla Batina. Clustering approach for higher-order deterministic masking. Cryptology ePrint Archive, Report 2025/273, 2025.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 388–397. Springer, Berlin, Heidelberg, August 1999.

[Kle13] Andreas Klein. *Linear Feedback Shift Registers*, pages 17–58. Springer London, London, 2013.

[KM22a] David Knichel and Amir Moradi. Composable gadgets with reused fresh masks first-order probing-secure hardware circuits with only 6 fresh masks. *IACR TCHES*, 2022(3):114–140, 2022.

[KM22b] David Knichel and Amir Moradi. Low-latency hardware private circuits. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 1799–1812. ACM Press, November 2022.

[Koo] Philip Koopman. Maximal length LFSR feedback terms. https://users.ece.cmu.edu/~koopman/lfsr/. Accessed: 03-Jan-2026.

[KR11] Lars R. Knudsen and Matthew Robshaw. *The Block Cipher Companion*. Information Security and Cryptography. Springer, 2011.

[KSM20] David Knichel, Pascal Sasdrich, and Amir Moradi. SILVER - statistical independence and leakage verification. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part I*, volume 12491 of *LNCS*, pages 787–816. Springer, Cham, December 2020.

[KSM22] David Knichel, Pascal Sasdrich, and Amir Moradi. Generic hardware private circuits towards automated generation of composable secure gadgets. *IACR TCHES*, 2022(1):323–344, 2022.

[LS07] Pierre L'Ecuyer and Richard Simard. TestU01: A C library for empirical testing of random number generators. 33(4), August 2007.

[Mar08] George Marsaglia. The marsaglia random number CDROM including the diehard battery of tests of randomness. http://www.stat.fsu.edu/pub/diehard/, 2008. Accessed: 03-Jan-2026.

[Mat94] Mitsuru Matsui. Linear cryptanalysis method for DES cipher. In Tor Helleseth, editor, *EUROCRYPT'93*, volume 765 of *LNCS*, pages 386–397. Springer, Berlin, Heidelberg, May 1994.

[McE87] Robert J. McEliece. *Finite Fields for Computer Scientists and Engineers*. Springer US, 1987.

[MLM24] Nicolai Müller, Daniel Lammers, and Amir Moradi. A deep analysis of two glitch-free hardware masking schemes SESYM and LMDPL. *IACR TCHES*, 2024(3):76–98, 2024.

[MM22] Nicolai Müller and Amir Moradi. PROLEAD A probing-based hardware leakage detection tool. *IACR TCHES*, 2022(4):311–348, 2022.

[MM24] Nicolai Müller and Amir Moradi. Robust but relaxed probing model. *IACR TCHES*, 2024(4):451–482, 2024.

[MS15] Kinga Marton and Alin Suciu. On the interpretation of results from the NIST statistical test suite. *Science and Technology*, 18(1):18–32, 2015.

[MZ20] Maria Chiara Molteni and Vittorio Zaccaria. On the spectral features of robust probing security. *IACR TCHES*, 2020(4):24–48, 2020.

[MZC22] Maria Chiara Molteni, Vittorio Zaccaria, and Valentina Ciriani. ADD-based spectral analysis of probing security. In *DATE*, pages 987–992. IEEE, 2022.

[PPG24] Christof Paar, Jan Pelzl, and Tim Güneysu. *Stream Ciphers*, pages 37–72. Springer Berlin Heidelberg, Berlin, Heidelberg, 2024.

[PYR+16] Stjepan Picek, Bohan Yang, Vladimir Rozic, Jo Vliegen, Jori Winderickx, Thomas De Cnudde, and Nele Mentens. PRNGs for masking applications and their mapping to evolvable hardware. In *CARDIS*, volume 10146 of *Lecture Notes in Computer Science*, pages 209–227. Springer, 2016.

[RBFSG22] Jan Richter-Brockmann, Jakob Feldtkeller, Pascal Sasdrich, and Tim Güneysu. VERICA - verification of combined attacks automated formal verification of security against simultaneous information leakage and tampering. *IACR TCHES*, 2022(4):255–284, 2022.

[SDM23] Aein Rezaei Shahmirzadi, Siemen Dhooghe, and Amir Moradi. Low-latency and low-randomness second-order masked cubic functions. *IACR TCHES*, 2023(1):113–152, 2023.

[SM15] Tobias Schneider and Amir Moradi. Leakage assessment methodology - A clear roadmap for side-channel evaluations. In Tim Güneysu and Helena Handschuh, editors, *CHES 2015*, volume 9293 of *LNCS*, pages 495–513. Springer, Berlin, Heidelberg, September 2015.

[SM21] Aein Rezaei Shahmirzadi and Amir Moradi. Second-order SCA security with almost no fresh randomness. *IACR TCHES*, 2021(3):708–755, 2021.

[SOMK22] Marek Sýs, Lubomír Obrátil, Vashek Matyás, and Dusan Klinec. A bad day to die hard: Correcting the dieharder battery. *Journal of Cryptology*, 35(1):3, January 2022.

[Sug18] Takeshi Sugawara. 3-share threshold implementation of AES S-box without fresh randomness. *IACR TCHES*, 2019(1):123–145, 2018.

[SW13] Rafał Stępień and Janusz Walczak. Statistical analysis of the LFSR generators in the NIST STS test suite. *Computer applications in electrical engineering*, 11, 2013.

[UHIM24] Rei Ueno, Naofumi Homma, Akiko Inoue, and Kazuhiko Minematsu. SoK: Pseudorandom generation for masked cryptographic implementation. Cryptology ePrint Archive, Report 2024/1968, 2024.

[Wal08] John Walker. A pseudorandom number sequence test program. https://www.fourmilab.ch/random/, 2008. Accessed: 03-Jan-2026.

A Walsh-Hadamard Transform: A Toy Example

We consider the Boolean function $f\colon \mathbb{F}_2^2 \to \mathbb{F}_2;\ (x_1, x_0) \mapsto x_0 x_1$ as an example to provide the intuition behind the WHT and illustrate the definitions from Section 4 in Table 7.

Table 7: Relation between the multiplication and linear functions in $\mathbb{F}_2$. The agreements between the linear functions and f are highlighted in green and the disagreements in red. The Walsh Spectrum is $\vec{\mathrm{Sp}}_f = (+2, +2, +2, -2)$. It means that the functions $0, x_0, x_1, x_0 \oplus x_1 \oplus 1$ equally likely approximate f with a probability of $3/4$ to be correct.

Input		**Output**	**Linear functions** $\vec{u}^\intercal \vec{x}$			
		$f(x_1, x_0)$	$\vec{u}_0 = (0,0)$	$\vec{u_1} = (0,1)$	$\vec{u_2} = (1,0)$	$\vec{u_3} = (1,1)$
x_1	x_0	$= x_0 x_1$	0	x_0	x_1	$x_0 \oplus x_1$
0	0	**0**	0	0	0	0
0	1	**0**	0	1	0	1
1	0	**0**	0	0	1	1
1	1	**1**	0	1	1	0
Walsh coefficients W_f			$+2$	$+2$	$+2$	-2
Correlations Cor_f			$1/2$	$1/2$	$1/2$	$-1/2$
Probabilities p_f			$3/4$	$3/4$	$3/4$	$1/4$

We explore the link between f and x_0 with joint probabilities since $\mathrm{Cor}_f(\vec{u}_1) = 1/2$:

$$\Pr[f = 0, x_0 = 0] = 1/2\,, \qquad \Pr[f = 0, x_0 = 1] = 1/4\,,$$
$$\Pr[f = 1, x_0 = 0] = 0\,, \qquad \Pr[f = 1, x_0 = 1] = 1/4\,,$$

We see that the joint probabilities are not fully uniform, meaning here that f is statistically dependent from x_0, and vice versa. Note that we have

$$\Pr[f = 0, x_0 = 0] + \Pr[f = 1, x_0 = 1] = 3/4 = \mathrm{p}_f(\vec{u_1}) = \frac{1}{2}(\mathrm{Cor}_f(\vec{u}_1) + 1).$$

We also compute the conditional probabilities that are not fully uniform either:

$$\Pr[f = 0 \mid x_0 = 0] = 1\,, \qquad \Pr[f = 0 \mid x_0 = 1] = 1/2\,,$$
$$\Pr[f = 1 \mid x_0 = 0] = 0\,, \qquad \Pr[f = 1 \mid x_0 = 1] = 1/2\,,$$

and

$$\Pr[x_0 = 0 \mid f = 0] = 2/3\,, \qquad \Pr[x_0 = 0 \mid f = 1] = 0\,,$$
$$\Pr[x_0 = 1 \mid f = 0] = 1/3\,, \qquad \Pr[x_0 = 1 \mid f = 1] = 1\,.$$

We can then create a distinguisher to recover the value of x_0 from the value of f.

B Verification: Additional Details and Results

B.1 Detailed Examples

B.1.1 First-order verification

We illustrate the steps of our verification from Section 4.2.1 to assess the use of an LFSR in a masking scheme. For our example, we consider the first-order 4-bit SKINNY S-box circuit with the HPC3 gadget and the LFSR defined by $X^7 + X^6 + 1$ (60) in Galois mode. The Boolean component functions of this S-box are (with most significant bit first ordering):

$$y_3 = x_0 \oplus (x_3 \oplus 1)\,(x_2 \oplus 1) \qquad y_2 = x_3 \oplus (x_2 \oplus 1)\,(x_1 \oplus 1),$$
$$y_1 = x_2 \oplus (x_1 \oplus 1)\,(y_3 \oplus 1) \qquad y_0 = x_1 \oplus (y_3 \oplus 1)\,(y_2 \oplus 1),$$

where x_i and y_i, $i \in [0,3]_{\mathbb{N}}$, are the original secret inputs and outputs, respectively. The first-order masked implementation of this S-box with HPC3 requires 12 bits of randomness (see Section 5.2.1). Therefore, the Galois LFSR generates 12 variables $s_6^1, s_6^2, \ldots, s_6^{12}$ (see Section 3.2). As reported in Table 1, the absolute maximal correlation value found for X^7+X^6+1, with a linear function of original secret input variables, is $1/2$ ($-\log_2(1/2) = 1$). The corresponding probe is the first output share of the addition $x_2 \oplus (x_1 \oplus 1)(y_3 \oplus 1)$, i.e., the first share of y_1, namely $y_{1,0}$. From our symbolic representation in Sagemath of the masked circuit provided by HADES, the ANF of $y_{1,0}$ with the shared inputs $x_{0,0}, x_{0,1}, x_{1,0}, \ldots, x_{3,1}$ and the refreshing variables $z_0, z_1, \ldots, z_7$ is

$$x_{0,0}x_{1,0} \oplus x_{0,0} \oplus x_{0,1}x_{1,0} \oplus x_{0,1} \oplus x_{1,0}x_{2,0}x_{3,0} \oplus x_{1,0}x_{2,0}x_{3,1} \oplus x_{1,0}x_{2,0} \oplus x_{1,0}x_{2,1}x_{3,0}$$
$$\oplus x_{1,0}x_{2,1}x_{3,1} \oplus x_{1,0}x_{2,1} \oplus x_{1,0}x_{3,0} \oplus x_{1,0}x_{3,1} \oplus x_{2,0}x_{3,0} \oplus x_{2,0}x_{3,1} \oplus x_{2,1}x_{3,0} \oplus x_{2,1}x_{3,1}$$
$$\oplus x_{2,1} \oplus x_{3,0} \oplus x_{3,1} \oplus z_4 \oplus z_5\,.$$

Step 1. In this ANF, we substitute the shared inputs by their encoding expression with the variables of the corresponding output state of the chosen LFSR, e.g., we have $(x_{0,0}, x_{0,1}) = (x_0 \oplus s_6^1, s_6^1)$ for the first input. We perform the same operation on the refreshing values; we thus have $(z_0, z_1, \ldots, z_7) = (s_6^5, s_6^6, \ldots, s_6^{12})$. Then, we reiterate the process in the resulting ANFs by substituting the LFSR output state variables by the LFSR initial state (seed) variables according to the recurrence relation of the chosen mode (here, Eq. (3)). As a result, we obtain the following ANF for $y_{1,0}$ with the original secret inputs x_0, x_1, x_2, x_3 and the seed variables $s_0^0, s_1^0, s_2^0, \ldots, s_6^0$:

$$x_0x_1 \oplus x_0s_6^0 \oplus x_0s_5^0 \oplus x_0s_4^0 \oplus x_0 \oplus x_1x_2x_3 \oplus x_1x_2 \oplus x_1x_3 \oplus x_2x_3s_6^0 \oplus x_2x_3s_5^0 \oplus x_2x_3s_4^0$$
$$\oplus x_2x_3 \oplus x_2s_6^0 \oplus x_2s_5^0 \oplus x_2s_4^0 \oplus x_3s_6^0 \oplus x_3s_5^0 \oplus x_3s_4^0 \oplus x_3\,.$$

Step 2. We apply the WHT on the above expression. From the computed Walsh spectrum, we find the below linear approximations which hold with non-zero correlation values:

- x_2 with a correlation value being equal to $1/2$;
- x_0, $x_0 \oplus x_2 \oplus 1$, $x_0 \oplus x_3$, $x_0 \oplus x_2 \oplus x_3$, $x_0 \oplus x_1 \oplus s_6^0 \oplus s_5^0 \oplus s_4^0$, $x_0 \oplus x_1 \oplus x_3 \oplus s_6^0 \oplus s_5^0 \oplus s_4^0$, with a correlation value being equal to $1/4$, among others.

Step 3. Finally, we check whether there are non-zero correlation linear approximations with only the original secret inputs x_i to assess probing security. In this example, we find one of those with correlation value $1/2$ and four with correlation value $1/4$. Note that the outcome of the third step is identical for the same characteristic polynomial in Fibonacci mode (see Remark 1). Therefore, first-probing security of this masking scheme is broken when using an LFSR with characteristic polynomial $X^7 + X^6 + 1$.

B.1.2 Second-order verification

For the extended version of this paper, we provide the details of our second-order verification from Section 4.2.1 to determine the security of using a specific LFSR in a masking scheme. For this example, we still consider the 4-bit SKINNY S-box implementation with the HPC3 gadget and the LFSR defined by $X^7 + X^4 + 1$ (`48` in hex.) in Galois mode, but masked at the second-order. This second-order masked implementation requires 32 bits of randomness: The Galois LFSR thus generates 32 variables $s_6^1, s_6^2, \ldots, s_6^{32}$ (see Section 3.2). To encode the inputs, we first choose to create the second shares of the inputs $(x_{0,1}, x_{1,1}, x_{2,1}, x_{3,1}) = (s_6^1, s_6^2, s_6^3, s_6^4)$, then the third shares such that $(x_{0,2}, x_{1,2}, x_{2,2}, x_{3,2}) = (s_6^5, s_6^6, s_6^7, s_6^8)$.

For the second-order verification, we begin by performing the first-order security analysis as illustrated in Section B.1.1, where we consider the ANF of a single probe and we compute their WHT according to Section 4.1.1. Then, for the higher-order verification, we consider linear combinations of up to d probes as described in Section 4.1.2. In this example, we create all the possible linear combinations of up to two probes to assess second-order probing security.

The absolute maximal correlation value for $X^7 + X^4 + 1$ (`48` in hex.), with a linear function of original secret input variables, is 1 ($-\log_2(1) = 0$) as reported in Table 3. It is found for 14 distinct linear combinations of two probes. We detail one of those combinations with one secret-only linear approximation, which holds with a correlation value of 1. We call this combination c, which is the addition of the first output share of the negation (NOT gate) $x_3 \oplus 1$ and the third refreshing value z_2. According to the symbolic representation of this masked circuit in Sagemath, its ANF is:

$$c = (x_{3,0} \oplus 1) \oplus z_2 \,.$$

Step 1. In this ANF, like in the first-order verification example, we substitute the shared inputs by their encoding expression with the variables of the corresponding output state of the chosen LFSR, and we replace the refreshing variables by the following output bits of the same LFSR. It means that we have

$$x_{3,0} = x_3 \oplus x_{3,1} \oplus x_{3,2} = x_3 \oplus s_6^4 \oplus s_6^8 \quad \text{and} \quad z_2 = s_6^{11} \,.$$

Then, we perform a second substitution in the resulting ANF: the LFSR output variables are replaced by the LFSR initial state variables, as described by the recurrence relation (see Section 3.2 for the Galois mode). In more details, we have

$$x_{3,0} = x_3 \oplus (s_5^0 \oplus s_2^0) \oplus (s_5^0 \oplus s_4^0 \oplus s_1^0) = x_3 \oplus s_4^0 \oplus s_2^0 \oplus s_1^0 \quad \text{and} \quad z_2 = s_4^0 \oplus s_2^0 \oplus s_1^0 \,.$$

Consequently, the resulting algebraic expression of c is $c = x_3 \oplus 1$ as function of the original secret inputs and the seed variables.

Step 2. From there, the application of the WHT on the above ANF is straightforward: $x_3 \oplus 1$ is the only linear approximation of c, which has a correlation value of 1.

Step 3. As a final step, we check whether the identified non-zero correlation linear approximation is expressed with only the original secret inputs x_i to assess probing security: This is true in this case. As a result, we conclude that second-probing security of this masking scheme with the LFSR of with characteristic polynomial $X^7 + X^4 + 1$ is compromised. Note that the individual probes on $x_{3,0} \oplus 1$ and z_2 are not correlated with linear approximations of only original secret inputs.

B.2 Further First-Order Verification Results

We provide additional verification results to assess glitch-extended d-probing security when using a single LFSR to share and refresh a masking scheme. We present the list of characteristic polynomials for which non-zero correlation values were found with linear approximations of only the original secret inputs ($p \in \mathcal{P}_n^{\neq 0}$):

- in Table 8 for the first-order masked 8-bit SKINNY S-box with the DOM, HPC1 and HPC2 gadgets, with an LFSR length $n \in [4, 20]_{\mathbb{N}}$. To be concise, we only report the absolute maximal correlation values found with such secret-only linear approximations. The term 'Gad.' refers to the gadget used in the circuit.
- in Table 9 for the first-order masked 8-bit AES S-box with the HPC3 gadget, with LFSR lengths $n \in [4, 20]_{\mathbb{N}}$. For readability, we only report the characteristic polynomials in $\mathcal{P}_n^{\neq 0}$ with the highest and lowest absolute maximal correlation values for a given n.

To complement the latter, we report up to the first 50 LFSRs with zero correlation with secret-only linear approximations ($p \in \mathcal{P}_n^{=0}$), in Table 10 for $n \in [11, 20]_{\mathbb{N}}$.

In the aforementioned tables, we may only analyze the first 500 or 200 characteristic polynomials from the list provided in [Koo] as indicated by the symbol $*$ in the column $|\mathcal{P}_n|$, since the number of characteristic polynomials grows rapidly with the length n.

Table 8: First-order masked 8-bit SKINNY S-box with DOM, HPC1-2 gadgets.

LFSR n	**Gad.**	$\lvert\mathcal{P}_n^{\neq 0}\rvert / \lvert\mathcal{P}_n\rvert$	$p \in \mathcal{P}_n^{\neq 0}$ **in hex.** ($\max_p \lvert\mathrm{Cor}\rvert$ **in** $-\log_2$ **scale**)
4	DOM	2 / 2	9 (3), C (1)
	HPC1	2 / 2	9 (1), C (1)
	HPC2	2 / 2	9 (3), C (1)
5	DOM	6 / 6	12 (2), 14 (2), 17 (2), 1B (4), 1D (3), 1E (3)
	HPC1	6 / 6	12 (1), 14 (1), 17 (2), 1B (1), 1D (2), 1E (1)
	HPC2	6 / 6	12 (1), 14 (2), 17 (2), 1B (4), 1D (3), 1E (3)
6	DOM	2 / 6	33 (3), 39 (3)
	HPC1	6 / 6	21 (2), 2D (2), 30 (2), 33 (2), 36 (1), 39 (1)
	HPC2	2 / 6	33 (3), 39 (3)
7	DOM	8 / 18	41 (1), 47 (4), 48 (2), 53 (2), 69 (4), 72 (2), 78 (4), 7B (3)
	HPC1	9 / 18	44 (3), 47 (3), 53 (1), 55 (4), 5F (2), 72 (5), 77 (4), 7B (4), 7E (1)
	HPC2	8 / 18	41 (1), 47 (4), 48 (1), 53 (2), 69 (4), 72 (1), 78 (4), 7B (3)
8	DOM	2 / 16	95 (4), B1 (3)
	HPC1	4 / 16	B8 (2), C3 (3), E1 (2), F3 (3)
	HPC2	2 / 16	95 (4), B1 (3)
9	DOM	2 / 48	14A (4), 1EA (5)
	HPC1	12 / 48	108 (3), 137 (3), 151 (4), 157 (2), 167 (3), 17C (3), 1B5 (1), 1B6 (1), 1B9 (3), 1C2 (4), 1EC (2), 1FD (3)
	HPC2	2 / 48	14A (4), 1EA (5)
10	DOM	3 / 60	2C2 (3), 38E (3), 3AC (2)
	HPC1	5 / 60	232 (5), 2B5 (3), 312 (1), 31B (2), 3AC (1)
	HPC2	3 / 60	2C2 (3), 38E (3), 3AC (2)

Table 8: First-order masked 8-bit SKINNY S-box with DOM, HPC1-2 gadgets. (Continued)

LFSR n	**Gad.**	$\lvert\mathcal{P}_n^{\neq 0}\rvert/\lvert\mathcal{P}_n\rvert$	$p \in \mathcal{P}_n^{\neq 0}$ **in hex. (**$\max_p\lvert\mathrm{Cor}\rvert$ **in** $-\log_2$ **scale)**
11	DOM	4 / 176	`500` (2), `588` (3), `5BE` (4), `6AA` (2)
	HPC1	5 / 176	`415` (2), `454` (4), `4E3` (2), `578` (4), `70E` (3)
	HPC2	4 / 176	`500` (1), `588` (3), `5BE` (4), `6AA` (2)
12	DOM	1 / 144	`C77` (3)
	HPC1	4 / 144	`834` (5), `AE2` (2), `C9A` (4), `EA8` (4)
	HPC2	1 / 144	`C77` (2)
13	DOM	3 / 630	`12EB` (4), `186C` (3), `1B65` (5)
	HPC1	5 / 630	`12E7` (4), `18FF` (4), `1B24` (3), `1B8D` (4), `1CFB` (3)
	HPC2	3 / 630	`12EB` (4), `186C` (3), `1B65` (5)
14	DOM	2 / 756	`2C20` (3), `36DE` (5)
	HPC1	6 / 756	`2307` (3), `230E` (4), `2945` (5), `36DD` (1), `376D` (2), `3B2F` (3)
	HPC2	2 / 756	`2C20` (3), `36DE` (5)
15	DOM	1 /1800	`4901` (3)
	HPC1	3 /1800	`6517` (4), `6C06` (3), `73FB` (3)
	HPC2	1 /1800	`4901` (3)
16	DOM	0 /2048	
	HPC1	2 /2048	`CBFE` (2), `FC00` (2)
	HPC2	0 /2048	
17	DOM	0 /500*	
	HPC1	2 /500*	`10400` (2), `10405` (2)
	HPC2	0 /500*	
18	DOM	0 /500*	
	HPC1	0 /500*	
	HPC2	0 /500*	
19	DOM	0 /200*	
	HPC1	0 /200*	
	HPC2	0 /200*	
20	DOM	0 /200*	
	HPC1	0 /200*	
	HPC2	0 /200*	

Table 9: First-order masked 8-bit AES S-box with HPC3 gadget.

LFSR n	$\lvert\mathcal{P}_n^{\neq 0}\rvert/\lvert\mathcal{P}_n\rvert$	$p \in \mathcal{P}_n^{\neq 0}$ **in hex. with highest and lowest (**$\max_p\lvert\mathrm{Cor}\rvert$ **in** $-\log_2$ **scale) for a given** n
4	2 / 2	Highest (0) | Lowest (0): `9`, `C`
5	6 / 6	Highest (0) | Lowest (0): `12`, `14`, `17`, `1B`, `1D`, `1E`
6	6 / 6	Highest (0): `21`, `2D`, `30`, `36`, `39` | Lowest (2): `33`
7	18 / 18	Highest (0): `44`, `60`, `69`, `7E` | Lowest (4): `48`, `72`
8	16 / 16	Highest (0): `95`, `96`, `A6`, `FA` | Lowest (5): `8E`, `B4`, `C6`, `D4`, `E1`, `E7`

Table 9: First-order masked 8-bit AES S-box with HPC3 gadget. (Continued)

LFSR n	$\lvert\mathcal{P}_n^{\neq 0}\rvert / \lvert\mathcal{P}_n\rvert$	$p \in \mathcal{P}_n^{\neq 0}$ **in hex. with highest and lowest (**$\max_p \lvert\mathrm{Cor}\rvert$ **in** $-\log_2$ **scale) for a given** n
9	48 / 48	Highest (0): 143, 1E5 | Lowest (6): 13E, 15B, 17C, 18A, 1C2
10	60 / 60	Highest (1): 273, 291, 2C7, 2E3 | Lowest (7): 232, 38B, 3C6
11	176 / 176	Highest (0): 56A, 5CA, 6B8 | Lowest (8): 43D, 503, 6DE
12	141 / 144	Highest (1): 939, BC4, CBD | Lowest (8): 829, 8A7, 8B5, 8C2, 98E, 9D1, A03, A33, A8E, AE2, BE0, CD8, CF3, D45, DAB, DC8, DDF, E5D, E62, E91, EA1, F33, F47, F88
13	491 / 630	Highest (0): 1657, 1DB7 | Lowest (9): 10DF, 1BF0
14	433 / 756	Highest (0): 3B29 | Lowest (9): 300A, 3388, 35F6, 3E13
15	175 /500*	Highest (1): 4836, 4C04 | Lowest (9): 4908
16	87 /500*	Highest (4): 8B4D | Lowest (9): 84D7
17	53 /500*	Highest (1): 10004 | Lowest (8): 1002A, 10051, 10171, 10247, 10346, 103F4, 10405, 10632, 106AE, 106BA, 1074D, 10774, 10874, 10881, 108EE, 10931, 10BB7, 10C29, 10CB9, 10D00, 10D56, 10E17, 10E77, 10F3E
18	24 /500*	Highest (5): 21179 Lowest (8): 20766, 209AD, 209C8, 20CC4, 21264, 214B3, 21D08
19	7 /200*	Highest (1): 40013, 40037, 40070 Lowest (8): 400EF, 40326, 4044B
20	2 /200*	Highest (8) | Lowest (8): 805EA, 80CC1

Table 10: First-order masked 8-bit AES S-box with HPC3 gadget.

LFSR n	$\lvert\mathcal{P}_n^{=0}\rvert / \lvert\mathcal{P}_n\rvert$	**Up to the first** 50 $p \in \mathcal{P}_n^{=0}$ **in hexadecimal notation**
11	0 / 176	
12	3 / 144	AEE, CDE, EA8
13	139 / 630	100D, 1045, 104F, 10C4, 10CB, 10FD, 112E, 114B, 11AC, 11B7, 11CA, 11D1, 11DD, 1230, 1236, 1241, 125F, 1263, 128B, 1296, 12AA, 12B8, 12BB, 12C3, 12CA, 1308, 1325, 1343, 137A, 13AE, 13C1, 13EF, 141D, 1474, 149A, 149F, 14A0, 14BE, 14C9, 14FA, 1519, 1591, 15A1, 15CB, 1616, 1620, 1662, 1668, 1697, 16A2
14	323 / 756	201C, 2054, 2057, 2086, 2089, 20A1, 20D3, 20F8, 212B, 2142, 214E, 2163, 2190, 219F, 21CA, 21D7, 21E4, 2239, 227D, 2295, 229C, 22AC, 22B7, 22CF, 22DB, 22EB, 22FF, 2307, 231A, 232C, 2331, 2338, 2352, 2362, 23A7, 23D3, 23F4, 23F7, 2421, 242D, 243F, 2441, 248E, 2496, 249C, 24B4, 24BB, 24C3, 24F0, 251A
15	325 /500*	4001, 4008, 4016, 402F, 403B, 4040, 4043, 4049, 4052, 4061, 4073, 407A, 4080, 408A, 4092, 40AB, 40B0, 40B6, 40D0, 40D3, 40DC, 40E6, 4109, 411D, 4122, 413F, 4144, 4147, 416C, 418D, 4195, 41AF, 41B1, 41B4, 41B8, 41C5, 41CC, 41E8, 4211, 4217, 4218, 4233, 4236, 4241, 425A, 426F, 427E, 428E, 4293, 4299

Table 10: First-order masked 8-bit AES S-box with HPC3 gadget. (Continued)

LFSR n	$\lvert\mathcal{P}_n^{=0}\rvert/\lvert\mathcal{P}_n\rvert$	**Up to the first** 50 $p \in \mathcal{P}_n^{=0}$ **in hexadecimal notation**
16	413 /500*	8016, 801F, 8029, 805E, 806B, 8097, 80A7, 80AE, 80CB, 80D0, 80D6, 810A, 810C, 8112, 812E, 8136, 8142, 8148, 8150, 8172, 81A5, 81B8, 81C3, 81C6, 81D1, 81EE, 81FC, 8233, 8241, 8244, 8248, 8260, 8299, 82A3, 82B4, 82C3, 82E1, 82EE, 82F5, 8320, 8325, 8329, 8361, 83B6, 83BC, 83C1, 83F8, 8430, 845F, 846A
17	447 /500*	10007, 10010, 10016, 10019, 1001F, 10034, 1003D, 10046, 10057, 1005D, 10062, 1007A, 10085, 10086, 1008C, 10092, 1009E, 100AB, 100B0, 100B3, 100B6, 100BF, 100C1, 100E0, 100E3, 100E5, 100EC, 100F8, 10106, 10111, 10114, 10118, 1011B, 10122, 10135, 1013C, 1013F, 10141, 10148, 1015A, 10163, 1016F, 10174, 1017E, 10184, 10188, 1018B, 1018D, 10193, 10199
18	476 /500*	20013, 2001F, 20026, 2003D, 20040, 20073, 20076, 20083, 200A7, 200C8, 200F1, 200F4, 200F7, 20105, 20109, 20130, 2013A, 20155, 20178, 2018B, 20195, 20196, 201BB, 201C3, 201CA, 201CC, 201D4, 201D8, 201E2, 201EB, 201F0, 2020C, 20218, 2021E, 2022D, 20244, 20250, 20290, 202C9, 202CF, 202E2, 202ED, 202F0, 202FF, 20304, 20315, 2033E, 20346, 20394, 20398
19	193 /200*	4001F, 40023, 40029, 4002C, 40031, 4003E, 40049, 40057, 400A1, 400B0, 400B5, 400BA, 400C2, 400CE, 400D0, 400D6, 400D9, 400DF, 400E3, 400E6, 40105, 40109, 4010C, 40112, 40118, 40127, 40128, 40135, 40136, 40141, 40142, 4014B, 40150, 40190, 401A3, 401B4, 401B7, 401CF, 401D2, 401D4, 401E4, 401EB, 401ED, 401EE, 401F9, 4021B, 40228, 4023A, 4023F, 40244
20	198 /200*	80004, 80029, 80032, 80034, 8003D, 80079, 800B3, 800B6, 800BF, 800C7, 800DF, 80111, 80114, 80118, 8015C, 80199, 801A9, 801AC, 801B7, 801E1, 801FA, 801FF, 80211, 80242, 8024B, 80260, 80263, 80266, 80274, 80284, 80295, 802CF, 802F6, 802F9, 80302, 80315, 8031C, 8032A, 80338, 8033D, 8035B, 8036E, 80379, 8038A, 8039B, 803A4, 803BC, 803EF, 803FB, 8040C

B.3 Further Second-Order Verification Results

For the extended version of this paper, we add the following verification results tables for the second-order masked 4-bit SKINNY S-box with an LFSR length $n \in [4, 18]_{\mathbb{N}}$:

- in Table 11, we list the LFSRs for which non-zero correlation values were found with linear approximations of only the original secret inputs ($p \in \mathcal{P}_n^{\neq 0}$), for the implementations with the DOM, HPC1 and HPC2 gadgets. For the sake of conciseness, we only provide the characteristic polynomials in $\mathcal{P}_n^{\neq 0}$ with the highest and lowest absolute maximal correlation values for a given n.

- in Table 12, we list up to the first 24 LFSRs which are second-order probing secure ($p \in \mathcal{P}_n^{=0}$), for the implementations with the DOM, HPC1, HPC2 and HPC3 gadgets.

In those tables, we may only analyze the first 500 or 300 characteristic polynomials from the list provided in [Koo] as indicated by the symbol $*$ in the column $|\mathcal{P}_n|$. We remind that those reported LFSRs are secure or insecure given our specific choice of use of their output bits (see Section 5.4).

Table 11: Second-order masked 4-bit SKINNY S-box with DOM, HPC1/2 gadgets.

LFSR n	**Gadget**	$\lvert\mathcal{P}_n^{\neq 0}\rvert / \lvert\mathcal{P}_n\rvert$	$p \in \mathcal{P}_n^{\neq 0}$ **in hex. with highest and lowest** ($\max_p \lvert\mathrm{Cor}\rvert$ **in** $-\log_2$ **scale) for a given** n
4	DOM	2 / 2	Highest (0) | Lowest (0): 9, C
	HPC1	2 / 2	Highest (0) | Lowest (0): 9, C
	HPC2	2 / 2	Highest (0) | Lowest (0): 9, C
5	DOM	6 / 6	Highest (0): 12, 17, 1B, 1E | Lowest (1): 14, 1D
	HPC1	6 / 6	Highest (0) | Lowest (0): 12, 14, 17, 1B, 1D, 1E
	HPC2	6 / 6	Highest (0) | Lowest (0): 12, 14, 17, 1B, 1D, 1E
6	DOM	6 / 6	Highest (1) | Lowest (1): 21, 2D, 30, 33, 36, 39
	HPC1	6 / 6	Highest (0): 21, 2D, 30, 33, 39 | Lowest (1): 36
	HPC2	6 / 6	Highest (0): 21, 2D, 30, 33, 39 | Lowest (1): 36
7	DOM	18 / 18	Highest (0): 44, 48 | Lowest (1): 41, 47, 4E, 53, 55, 5C, 5F, 60, 65, 69, 6A, 72, 77, 78, 7B, 7E
	HPC1	18 / 18	Highest (0): 41, 44, 47, 48, 53, 55, 60, 69, 78 Lowest (1): 4E, 5C, 5F, 65, 6A, 72, 77, 7B, 7E
	HPC2	18 / 18	Highest (0): 41, 44, 47, 48, 53, 5F, 60, 65, 6A, 72, 77, 78, 7B | Lowest (1): 4E, 55, 5C, 69, 7E
8	DOM	16 / 16	Highest (1) | Lowest (1): 8E, 95, 96, A6, AF, B1, B2, B4, B8, C3, C6, D4, E1, E7, F3, FA
	HPC1	16 / 16	Highest (0): 8E, 95, 96, B1, E1, E7, FA Lowest (1): A6, AF, B2, B4, B8, C3, C6, D4, F3
	HPC2	16 / 16	Highest (0): AF, C6, E1, E7 Lowest (1): 8E, 95, 96, A6, B1, B2, B4, B8, C3, D4, F3, FA
9	DOM	48 / 48	Highest (0): 108, 1B5 | Lowest (2): 1F1
	HPC1	48 / 48	Highest (0): 108, 12C, 1B5, 1F1, 1FD | Lowest (1) first 20: 10D, 110, 116, 119, 12F, 134, 137, 13B, 13E, 143, 14A, 151, 152, 157, 15B, 15E, 167, 168, 16D, 17A
	HPC2	48 / 48	Highest (0): 108, 110, 12C, 152, 157, 16D, 1AD, 1B5, 1B9, 1C7, 1DA | Lowest (1) first 15: 10D, 116, 119, 12F, 134, 137, 13B, 13E, 143, 14A, 151, 15B, 15E, 167, 168
10	DOM	60 / 60	Highest (0): 3A6, 3C6 Lowest (2): 237, 245, 262, 273, 2B5, 2C7, 2F2, 2FD, 327, 33C, 35A, 36F, 38B, 39C, 3B1, 3BE, 3ED, 3FC
	HPC1	60 / 60	Highest (0): 232, 321, 39C, 3A6, 3C6 Lowest (2): 26B, 286, 2CB, 3AA
	HPC2	60 / 60	Highest (0): 291, 298, 309, 3A6, 3C6, 3F9 Lowest (2): 33C, 38B, 3B1, 3BE
11	DOM	176 / 176	Highest (1) first 15: 402, 415, 416, 423, 431, 432, 438, 458, 467, 475, 47A, 494, 4A2, 4AD, 4BA | Lowest (3): 44F, 486, 4BF, 56F, 5A5, 5B2, 65F, 6CF, 6EB, 6FA, 740
	HPC1	176 / 176	Highest (0): 402, 475, 68E, 6DE, 7C1, 7C8 | Lowest (2) first 20: 415, 432, 44A, 458, 49D, 4C7, 4D6, 503, 514, 53F, 577, 57D, 584, 5A5, 5BE, 5C3, 5C5, 5DB, 5EE, 5F6
	HPC2	176 / 176	Highest (0): 402, 475, 509, 542, 68E Lowest (3): 486, 56F, 72A

Table 11: Second-order masked 4-bit SKINNY S-box with DOM, HPC1/2 gadgets. (Continued)

LFSR n	**Gadget**	$\lvert\mathcal{P}_n^{\neq 0}\rvert / \lvert\mathcal{P}_n\rvert$	$p \in \mathcal{P}_n^{\neq 0}$ **in hex. with highest and lowest** ($\max_p \lvert\mathrm{Cor}\rvert$ **in** $-\log_2$ **scale) for a given** n
12	DOM	144 / 144	Highest (1): 829, 834, 8A7, 9CC, 9D4, A03, B23, B2C, B52, B5E, C4B, CDE | Lowest (4): BBA, F28
	HPC1	144 / 144	Highest (0): D7A Lowest (3): 8B0, 9A6, B52, C9A, D68, E4F, E52
	HPC2	144 / 144	Highest (0): 891 | Lowest (3): 83D, 89D, 91E, 93F, 965, A27, B2A, BA2, BBA, BE0, C4B, CB2, CE4, D15, DDF, DE0, E26, E62, E67, EA1, F22, F28, F47, F71, F8D
13	DOM	568 / 630	Highest (1) first 14: 100D, 101A, 1029, 1061, 1079, 10A1, 110C, 1209, 1284, 1308, 1403, 140C, 1411, 1418 Lowest (4) first 14: 113F, 1159, 117B, 11AA, 1205, 120F, 12B8, 12E2, 1316, 13BC, 13C8, 141D, 1442, 1471
	HPC1	625 / 630	Highest (0): 169D, 172E, 1F7C Lowest (4): 1112, 1214, 1290, 14F9, 15D9, 1646, 1709, 17AF, 18BD, 1949, 1AAE, 1D39, 1E55, 1EB4
	HPC2	558 / 630	Highest (1) first 14: 100D, 101A, 1029, 1037, 1052, 1061, 1079, 108A, 10A1, 10F8, 110C, 11B7, 11F0, 1209 Lowest (4) first 14: 113F, 1159, 117B, 1205, 120F, 1316, 13BC, 13C8, 1442, 1471, 14BE, 14CF, 1510, 15EC
14	DOM	502 / 756	Highest (1) first 14: 201C, 2029, 2079, 20A1, 2973, 2A01, 300A, 301E, 3059, 3088, 3119, 3408, 36DE, 376D Lowest (4) first 14: 22EB, 2313, 2433, 25E5, 25FD, 2652, 265E, 26D6, 2724, 27CF, 27DD, 29DF, 29E3, 2A1C
	HPC1	704 / 756	Highest (1) first 14: 2029, 2067, 2075, 2221, 230E, 236D, 23BF, 240C, 242D, 25B5, 25CB, 2604, 260E, 2623 Lowest (4) first 14: 205D, 2171, 2184, 21CA, 229C, 2367, 23E0, 248E, 25AD, 25B6, 25BC, 25D0, 25EA, 271B
	HPC2	495 / 756	Highest (1) first 14: 201C, 2029, 2079, 2086, 20A1, 20CE, 20DA, 20EA, 20EC, 214E, 2184, 2295, 23F7, 242D Lowest (4) first 14: 2313, 2433, 25E5, 25FD, 2652, 265E, 26D6, 2724, 27CF, 27DD, 29DF, 29E3, 2A1C, 2AF4
15	DOM	740 /1800	Highest (0): 4008 Lowest (4) first 14: 4165, 41E2, 426F, 43DF, 43EF, 442E, 4482, 4484, 4493, 44D7, 45B6, 45EA, 4623, 4739
	HPC1	1252 /1800	Highest (0): 4008, 548C Lowest (4) first 14: 400B, 4049, 40AE, 40B6, 40E6, 41B4, 4211, 427B, 42AF, 42D8, 4304, 430E, 4315, 4332
	HPC2	725 /1800	Highest (0): 4008, 7C07 Lowest (4) first 14: 4165, 426F, 43EF, 442E, 4482, 4493, 44D7, 45EA, 4623, 4739, 473A, 4750, 4787, 47B2

Table 11: Second-order masked 4-bit SKINNY S-box with DOM, HPC1/2 gadgets. (Continued)

LFSR n	**Gadget**	$\lvert\mathcal{P}_n^{\neq 0}\rvert / \lvert\mathcal{P}_n\rvert$	$p \in \mathcal{P}_n^{\neq 0}$ **in hex. with highest and lowest** ($\max_p \lvert\text{Cor}\rvert$ **in** $-\log_2$ **scale) for a given** n
16	DOM	338 /2048	Highest (1): 801C, 8029, 810C, C00A, FE07 Lowest (4) first 14: 82C3, 82F5, 83B5, 88B7, 89BC, 8A49, 8D72, 8D84, 8F2F, 8F38, 927F, 95ED, 965C, 99F9
	HPC1	893 /2048	Highest (0): 924C, B5B5, D5D6, D86C Lowest (4) first 14: 810A, 8117, 818E, 8241, 8361, 83B6, 83BC, 846F, 847D, 84C5, 84F9, 85E9, 860E, 8753
	HPC2	331 /2048	Highest (0): FE07 Lowest (4) first 14: 82C3, 83B5, 88B7, 89BC, 8A49, 8D72, 8D84, 8F2F, 8F38, 95ED, 965C, 99F9, 9D94, 9F6F
17	DOM	192 /500*	Highest (1): 10007, 10019, 10051, 10188, 1020C, 10319, 10400, 10501 | Lowest (4) first 12: 103A7, 103B6, 1048E, 10564, 105AE, 106C7, 106D5, 106F1, 1072E, 1076F, 10774, 10787
	HPC1	122 /500*	Highest (0): 10007 | Lowest (4) first 12: 10010, 1009E, 100BF, 100E3, 10111, 101E8, 10293, 10323, 103A7, 103B6, 104C3, 1053D
	HPC2	185 /500*	Highest (1) first 10: 10007, 10019, 1002A, 10051, 10188, 1020C, 10233, 10319, 10400, 10428 Lowest (4) first 10: 10293, 103B6, 1048E, 10564, 106C7, 106F1, 1072E, 1076F, 10774, 10787
18	DOM	41 /300*	Highest (1): 201CC, 20400, 20A01 Lowest (4): 205B3, 20787, 20CB6, 20DAF, 20DBE, 20FB6
	HPC1	58 /300*	Highest (1): 20026, 200C8, 20304 | Lowest (4): 20315, 204B2, 20814, 2082D, 2117A, 212B0, 213B4, 214AE
	HPC2	39 /300*	Highest (1): 201CC, 20400, 20A01, 20C4C, 20E30 Lowest (4): 205B3, 20787, 20CB6, 20DAF, 20DBE, 20FB6

Table 12: Second-order masked 4-bit SKINNY S-box with the four gadgets.

LFSR n	**Gadget**	$\lvert\mathcal{P}_n^{=0}\rvert / \lvert\mathcal{P}_n\rvert$	**Up to the first** 24 $p \in \mathcal{P}_n^{=0}$ **in hex. notation**
13	DOM	62 / 630	10B3, 10E0, 1112, 11C3, 121E, 1228, 1260, 1263, 1302, 131F, 1325, 133B, 133D, 1358, 1394, 13A7, 13D0, 13E5, 1433, 1456, 1466, 148D, 149A, 14BB
	HPC1	5 / 630	11B8, 135B, 145F, 1BCA, 1EE1
	HPC2	72 / 630	10B3, 10E0, 1112, 11C3, 121E, 1228, 1260, 1263, 12BB, 12E2, 1302, 131F, 1325, 133B, 133D, 1358, 1375, 1394, 13A7, 13D0, 13E5, 1433, 1456, 1466
	HPC3	0 / 630	

Table 12: Second-order masked 4-bit SKINNY S-box with the four gadgets. (Continued)

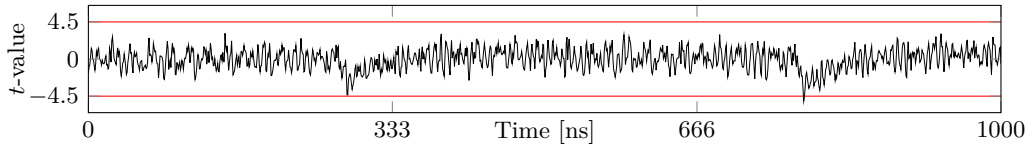
LFSR n	**Gadget**	$\lvert\mathcal{P}_n^{=0}\rvert / \lvert\mathcal{P}_n\rvert$	**Up to the first** 24 $p \in \mathcal{P}_n^{=0}$ **in hex. notation**
14	DOM	254 / 756	2057, 205D, 2106, 213F, 2142, 2171, 2177, 21D7, 21F6, 2205, 2239, 2269, 226A, 226F, 227D, 229C, 22CF, 22D2, 22DB, 22F9, 22FF, 2307, 231A, 2323
	HPC1	52 / 756	2057, 205E, 20D3, 20EC, 2269, 2421, 2430, 243F, 24BB, 25F8, 26D9, 26E0, 270A, 28C5, 2968, 2992, 29CD, 29DF, 2A9B, 2AC1, 2B14, 2C73, 2DE2, 2E5F
	HPC2	261 / 756	2057, 205D, 2106, 213F, 2142, 2171, 2177, 21BE, 21D7, 21F6, 2205, 2239, 2269, 226A, 226F, 227D, 229C, 22CF, 22D2, 22DB, 22F9, 22FF, 2307, 231A
	HPC3	0 / 756	
15	DOM	1060 /1800	402F, 403B, 4043, 4067, 4073, 407A, 40EF, 4122, 4147, 414D, 41A3, 41AF, 421B, 423C, 4241, 424B, 425A, 427B, 427E, 42AF, 42BD, 432A, 432C, 433E
	HPC1	548 /1800	4016, 4073, 407A, 40B0, 40D0, 40E5, 4109, 413F, 414D, 418B, 418D, 4199, 41A3, 41A6, 41AF, 41B1, 41B8, 41E2, 420C, 4217, 421B, 4236, 423C, 4241
	HPC2	1075 /1800	402F, 403B, 4043, 4067, 4073, 407A, 40EF, 4122, 4147, 414D, 41A3, 41AF, 421B, 423C, 4241, 424B, 425A, 427B, 427E, 42AF, 42BD, 42C6, 432A, 432C
	HPC3	0 /1800	
16	DOM	1710 /2048	806B, 80A7, 80DF, 8112, 8117, 8136, 8142, 8150, 8172, 8248, 825F, 82EE, 8325, 8329, 8345, 8406, 845F, 846A, 846F, 8471, 8478, 847D, 849C, 84BE
	HPC1	1155 /2048	8016, 801C, 8097, 80A7, 80AE, 80D0, 80D6, 810C, 8112, 812E, 8136, 8148, 81A5, 81B4, 81C3, 81C6, 81CF, 81D1, 81EE, 81FC, 8214, 822B, 8244, 8248
	HPC2	1717 /2048	806B, 80A7, 80DF, 8112, 8117, 8136, 8142, 8150, 8172, 8248, 825F, 82EE, 8325, 8329, 8345, 8406, 845F, 846A, 846F, 8471, 8478, 847D, 849C, 84BE
	HPC3	16 /2048	84C5, 84D2, 8562, 8A0D, 8FC1, 99E2, 9DF8, A23B, B20F, BA34, D58F, D6A7, DE14, DE7B, EAD5
17	DOM	308 /500*	10004, 10010, 1001F, 1003D, 10046, 10057, 1005D, 1007A, 100B0, 100B3, 100F8, 10106, 10114, 10118, 1011B, 10122, 10135, 1013C, 1013F, 10141, 1015A, 1016F, 10171, 1017E
	HPC1	378 /500*	10004, 10016, 10019, 1001F, 1002A, 10046, 1004C, 10057, 1005D, 1007A, 10086, 10092, 100B0, 100B3, 100E0, 100E5, 100EC, 100F8, 10106, 10114, 10118, 10122, 10135, 1013C
	HPC2	315 /500*	10004, 10010, 1001F, 1003D, 10046, 10057, 1005D, 1007A, 100B0, 100B3, 100F8, 10106, 10114, 10118, 1011B, 10122, 10135, 1013C, 1013F, 10141, 1015A, 1016F, 10171, 1017E
	HPC3	31 /500*	1017E, 1018B, 10471, 1047E, 10532, 10537, 1054C, 10561, 105AE, 106C7, 1075A, 10799, 1079C, 107D4, 1091F, 1097C, 109BA, 10BA0, 10BD4, 10BD7, 10BF3, 10C29, 10D39, 10DC3

Table 12: Second-order masked 4-bit SKINNY S-box with the four gadgets. (Continued)

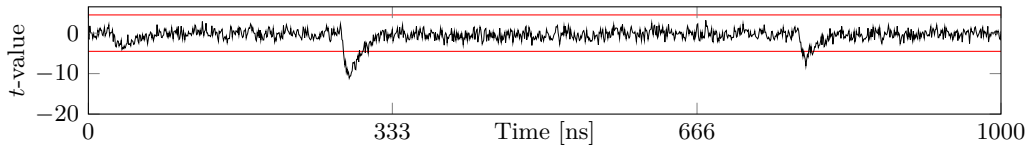
LFSR n	Gadget	$\lvert\mathcal{P}_n^{=0}\rvert / \lvert\mathcal{P}_n\rvert$	Up to the first 24 $p \in \mathcal{P}_n^{=0}$ in hex. notation
18	DOM	259 /300*	20013, 2001F, 20026, 2003D, 20040, 2006D, 20073, 20076, 20083, 200A7, 200F1, 200F4, 200F7, 20105, 20109, 20130, 2013A, 20155, 20178, 20196, 201C3, 201D4, 201E2, 201EB
	HPC1	242 /300*	2001F, 2006D, 20083, 200A7, 200F4, 2013A, 20155, 20178, 2018B, 20195, 201BB, 201CA, 201D4, 201D8, 201E2, 201F0, 2021E, 2022D, 2023C, 20244, 20250, 20290, 202C9, 202CF
	HPC2	261 /300*	20013, 2001F, 20026, 2003D, 20040, 2006D, 20073, 20076, 20083, 200A7, 200F1, 200F4, 200F7, 20105, 20109, 20130, 2013A, 20155, 20178, 20195, 20196, 201C3, 201D4, 201E2
	HPC3	80 /300*	2022D, 20244, 202F0, 20304, 2041D, 20465, 2047E, 20482, 204E7, 2050E, 2054A, 20562, 2056B, 2059D, 205EC, 205F8, 20625, 206C2, 206F7, 20728, 2073C, 207A0, 207CF, 20899

B.4 Further Evaluation Results for the 4-bit SKINNY S-box

We present, in this section, the remaining practical evaluation outcomes reported in Table 4 (see Section 6) for the first-order masked 4-bit SKINNY S-box implementation with the HPC3 multiplication gadget. All the presented side-channel evaluations are reported for 100 million traces. In more details, we present the leakage assessments for LFSRs of length 6 in Galois mode in Fig. 4 and Fig. 5, and in Fibonacci mode in Fig. 6 and Fig. 7. Then, the results for the LFSRs of length 7 in Fibonacci mode are shown in Fig. 8 and Fig. 9. Furthermore, we provide the evaluation outcomes for LFSRs in Galois mode of length 8 in Fig. 10 and Fig. 11, length 9 in Fig. 12 and Fig. 13, length 10 in Fig. 14, length 11 in Fig. 15 and Fig. 16, length 12 in Fig. 17.

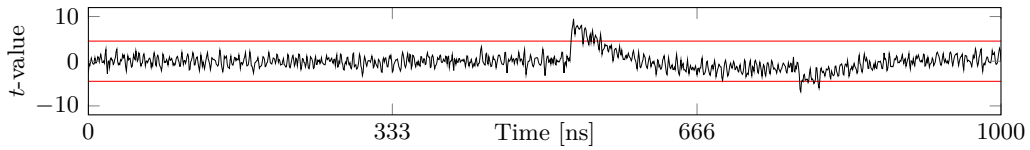


(a) First-order t-test results.

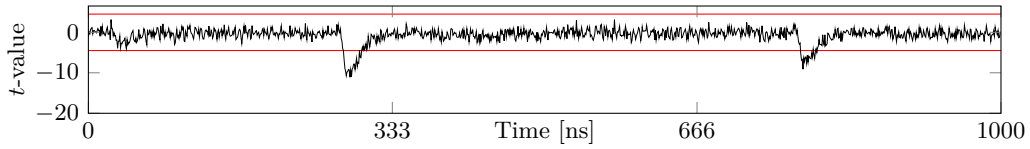


(b) Second-order t-test results.

Figure 4: First- and second-order t-test leakage using the Galois LFSR $X^6+X^5+X^2+X+1$ (33 in hex.) for the first-order masked 4-bit SKINNY S-box (100 million traces).

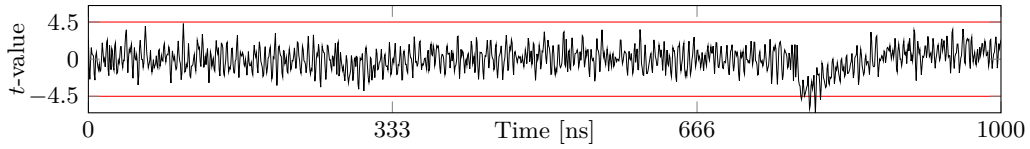


(a) First-order t-test results.

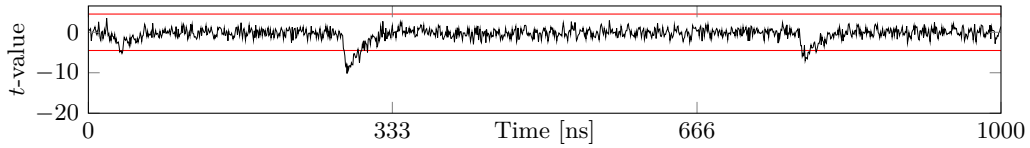


(b) Second-order t-test results.

Figure 5: First- and second-order t-test leakage using the Galois LFSR $X^6+X^4+X^3+X+1$ (`2D` in hex.) for the first-order masked 4-bit SKINNY S-box (100 million traces).

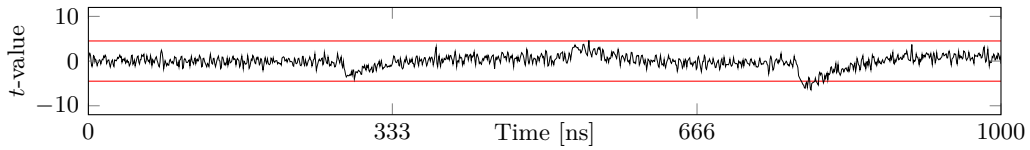


(a) First-order t-test results.

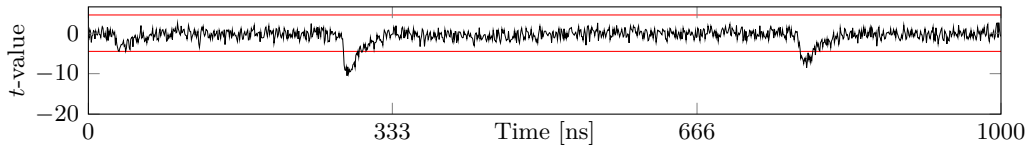


(b) Second-order t-test results.

Figure 6: First- and second-order t-test leakage using the Fibonacci LFSR $X^6+X^5+X^2+X+1$ (`33` in hex.) for the first-order masked 4-bit SKINNY S-box (100 million traces).

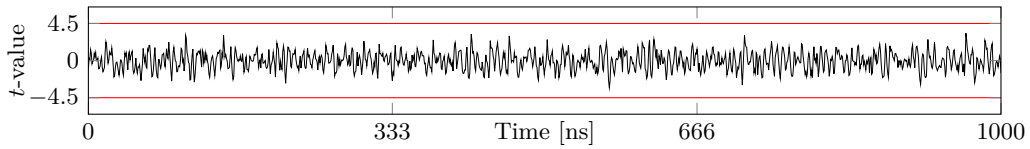


(a) First-order t-test results.

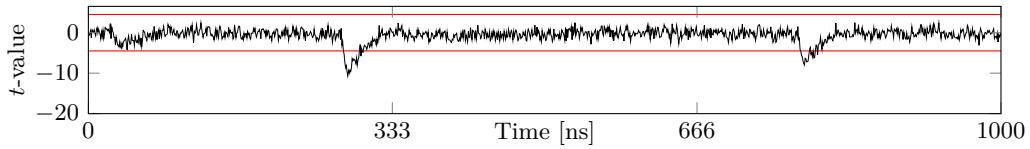


(b) Second-order t-test results.

Figure 7: First- and second-order t-test leakage using the Fibonacci LFSR $X^6+X^4+X^3+X+1$ (`2D` in hex.) for the first-order masked 4-bit SKINNY S-box (100 million traces).

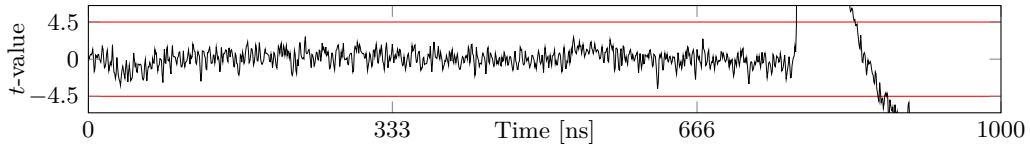


(a) First-order t-test results.

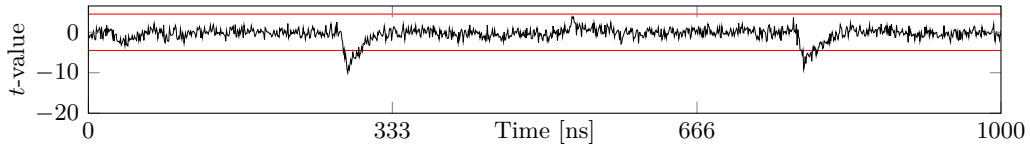


(b) Second-order t-test results.

Figure 8: No first-order t-test leakage using the Fibonacci LFSR $X^7 + X^6 + X^3 + X + 1$ (`65` in hex.) for the first-order masked 4-bit SKINNY S-box (100 million traces).

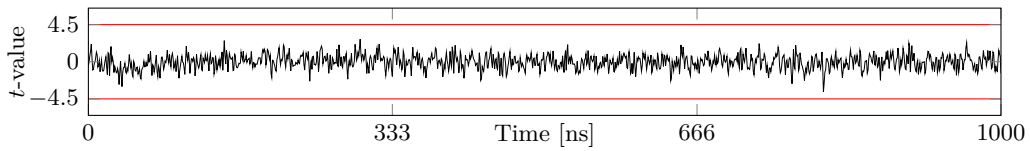


(a) First-order t-test results.

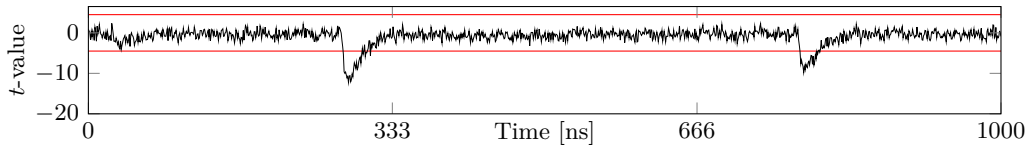


(b) Second-order t-test results.

Figure 9: First- and second-order t-test leakage using the Fibonacci LFSR $X^7 + X^6 + 1$ (`60` in hex.) for the first-order masked 4-bit SKINNY S-box (100 million traces).

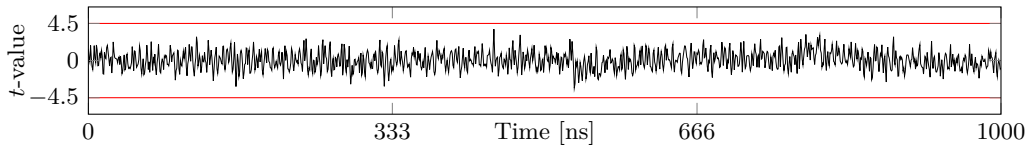


(a) First-order t-test results.

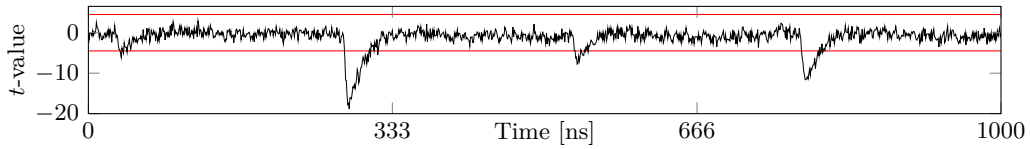


(b) Second-order t-test results.

Figure 10: No first-order t-test leakage using the Galois LFSR $X^8 + X^4 + X^3 + X^2 + 1$ (`8E` in hex.) for the first-order masked 4-bit SKINNY S-box (100 million traces).

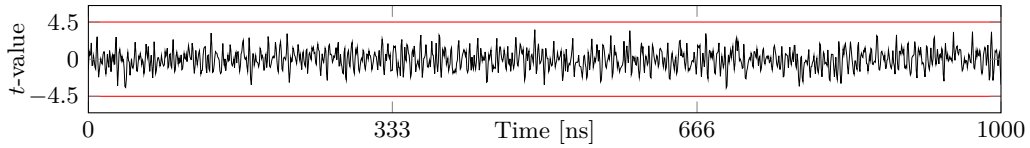


(a) First-order t-test results.

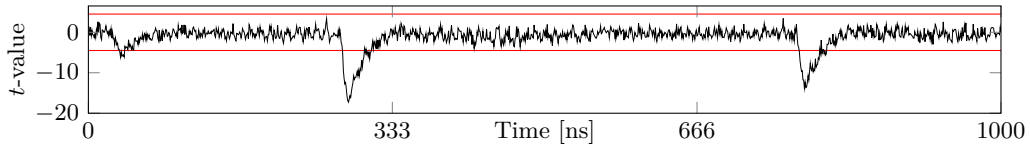


(b) Second-order t-test results.

Figure 11: No first-order t-test leakage using the Galois LFSR $X^8 + X^5 + X^3 + X^2 + 1$ (`96` in hex.) for the first-order masked 4-bit SKINNY S-box (100 million traces).

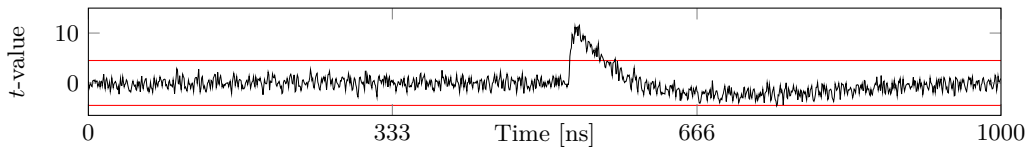


(a) First-order t-test results.

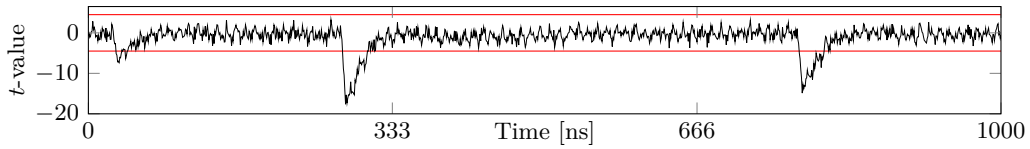


(b) Second-order t-test results.

Figure 12: No first-order t-test leakage using the Galois LFSR $X^9 + X^5 + 1$ (`110` in hex.) for the first-order masked 4-bit SKINNY S-box (100 million traces).

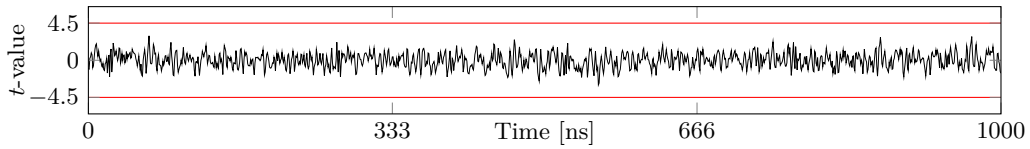


(a) First-order t-test results.

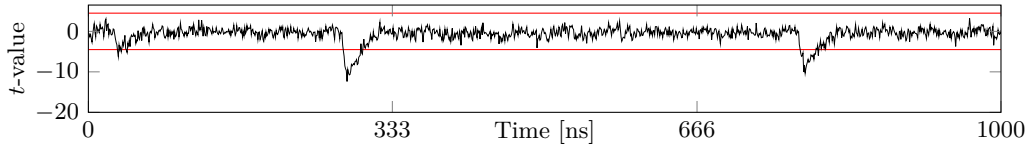


(b) Second-order t-test results.

Figure 13: First- and second-order t-test leakage using the Galois LFSR $X^9+X^5+X^4+X+1$ (`119` in hex.) for the first-order masked 4-bit SKINNY S-box (100 million traces).

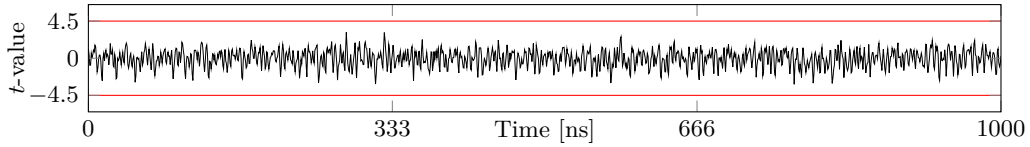


(a) First-order t-test results.

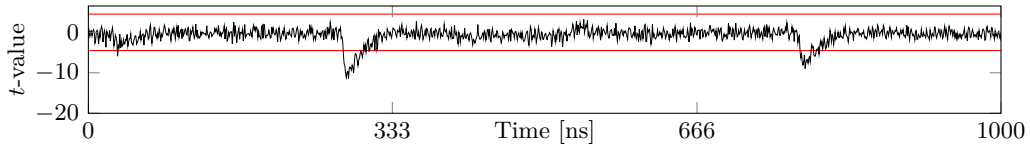


(b) Second-order t-test results.

Figure 14: No first-order t-test leakage using the Galois LFSR $X^{10} + X^4 + X^3 + X + 1$ (`20D` in hex.) for the first-order masked 4-bit SKINNY S-box (100 million traces).

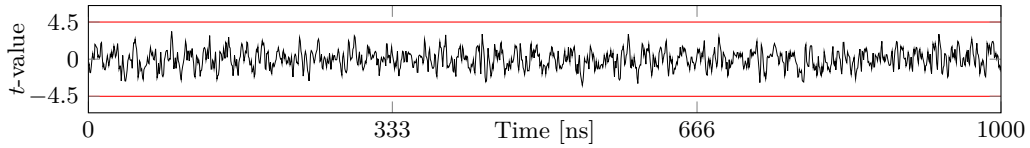


(a) First-order t-test results.

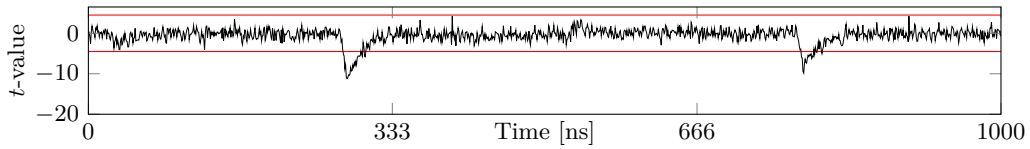


(b) Second-order t-test results.

Figure 15: No first-order t-test leakage using the Galois LFSR $X^{11} + X^7 + X^5 + X^4 + 1$ (`458` in hex.) for the first-order masked 4-bit SKINNY S-box (100 million traces).

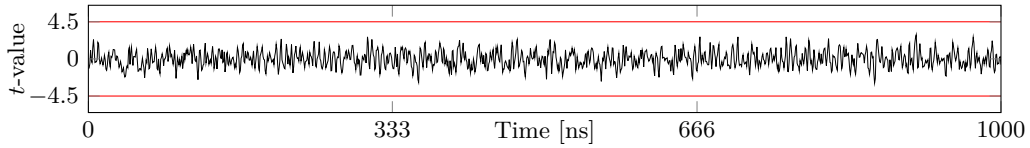


(a) First-order t-test results.

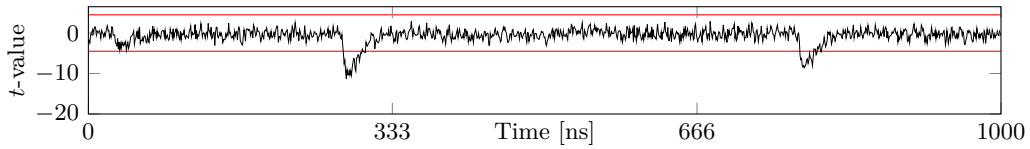


(b) Second-order t-test results.

Figure 16: No first-order t-test leakage using the Galois LFSR $X^{11} + X^7 + X^6 + X^5 + X^4 + X^2 + 1$ (`47A` in hex.) for the first-order masked 4-bit SKINNY S-box (100 million traces).



(a) First-order t-test results.

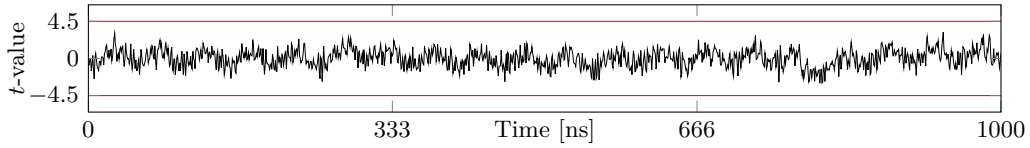


(b) Second-order t-test results.

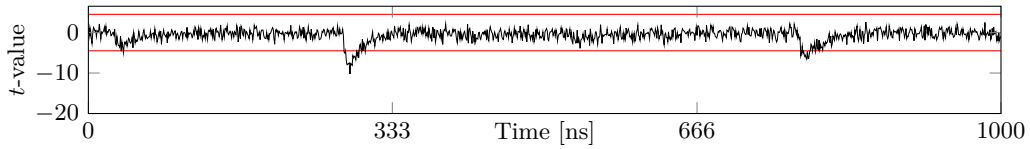
Figure 17: No first-order t-test leakage using the Galois LFSR $X^{12} + X^{11} + X^{10} + X^4 + 1$ (`E08` in hex.) for the first-order masked 4-bit SKINNY S-box (100 million traces).

B.4.1 Additional Evaluation Results for $n = 8$ and $n = 11$

For the extended version of this paper, we append side-channel evaluations for 250 million traces for the first-order masked 4-bit SKINNY S-box with HPC3, as reported in Table 5 (see Section 6). Those results are reported for the LFSRs in Galois mode of length 8 in Fig. 18 and Fig. 19, and of length 11 in Fig. 21 and Fig. 20.

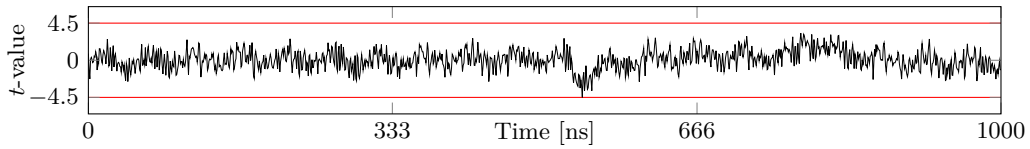


(a) First-order t-test results.

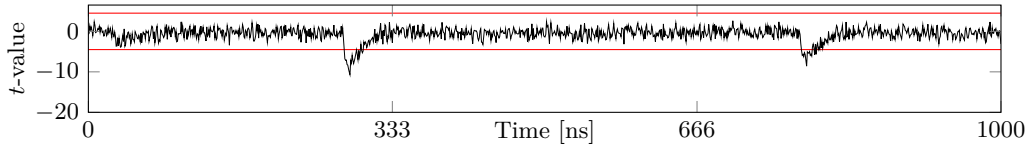


(b) Second-order t-test results.

Figure 18: No first-order t-test leakage using the Galois LFSR $X^8 + X^4 + X^3 + X^2 + 1$ (`8E` in hex.) for the first-order masked 4-bit SKINNY S-box (250 million traces).

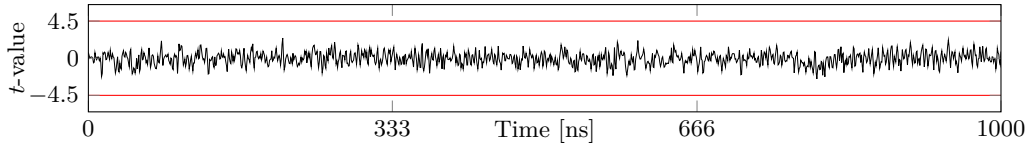


(a) First-order t-test results.

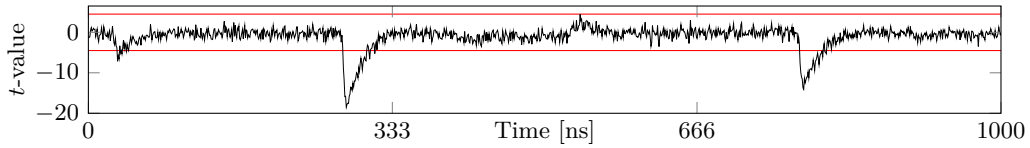


(b) Second-order t-test results.

Figure 19: First- and second-order t-test leakage using the Galois LFSR $X^8 + X^5 + X^3 + X^2 + 1$ (`96` in hex.) for the first-order masked 4-bit SKINNY S-box (250 million traces).

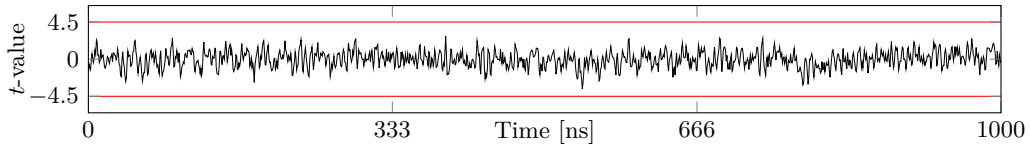


(a) First-order t-test results.

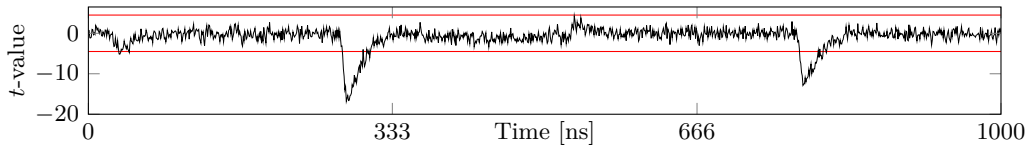


(b) Second-order t-test results.

Figure 20: No first-order t-test leakage using the Galois LFSR $X^{11} + X^7 + X^5 + X^4 + 1$ (`458` in hex.) for the first-order masked 4-bit SKINNY S-box (250 million traces).



(a) First-order t-test results.



(b) Second-order t-test results.

Figure 21: No first-order t-test leakage using the Galois LFSR $X^{11} + X^7 + X^6 + X^5 + X^4 + X^2 + 1$ (`47A` in hex.) for the first-order masked 4-bit SKINNY S-box (250 million traces).

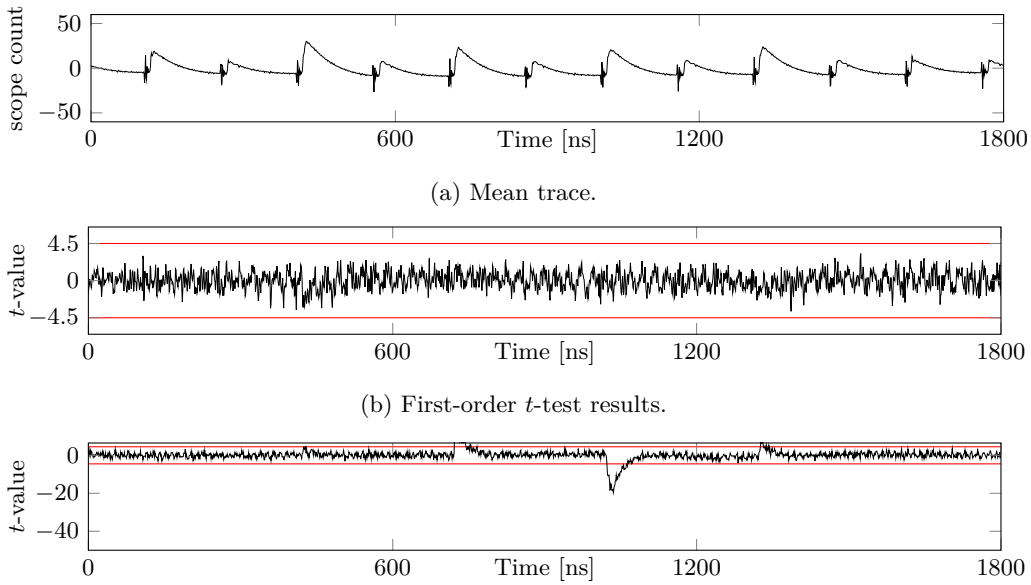
B.5 Further Evaluation Results for the 8-bit SKINNY S-box

For the extended version of this paper, we present in this section the practical evaluation outcomes reported in Table 6 (see Section 6) for the first-order masked 8-bit SKINNY S-box implementation with the HPC3 gadget.

First, we show that this S-box implementation is secure in practice by running an initial measurement of 250 million traces using randomness from an AES-128 instance in counter mode, which is seeded by the kernel randomness source of the control PC, as done for the 4-bit S-box version. We report the corresponding mean trace and t-test results in Fig. 22: We observe no first-order leakage, but we detect second-order leakage as expected.

Then, we test each selected LFSR of length $n \in [6, 16]_{\mathbb{N}}$ in Galois mode for 100 million fixed/random traces each, whose characteristic polynomials correspond to those selected in Table 6. Like for the 4-bit SKINNY S-box, the studied LFSR is freshly seeded during each individual execution, and its outputs are used to create the initial sharing of the S-box and refreshing the multiplication gadgets, as described in Section 5.4. For each tested characteristic polynomial, we then record 100 million fixed/random traces in total.

In more details, the t-test results for the LFSR of length 6 is shown in Fig. 23. Furthermore, the t-test outcomes are presented for the LFSRs of length 7 in Fig. 24 and Fig. 25, length 8 in Fig. 26 and Fig. 27, length 9 in Fig. 28 and Fig. 29, length 10 in Fig. 30 and Fig. 31, length 11 in Fig. 32 and Fig. 33, length 12 in Fig. 34 and Fig. 35, length 13 in Fig. 36 and Fig. 37, length 14 in Fig. 38 and Fig. 39, length 15 in Fig. 40 and Fig. 41 and length 16 in Fig. 42 and Fig. 43.

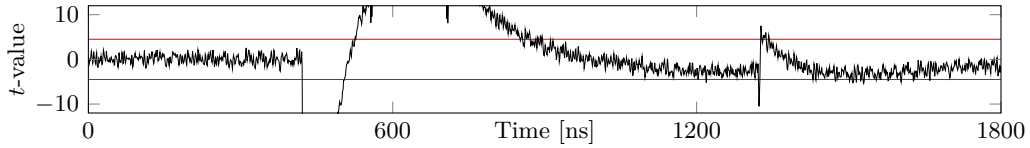


(a) Mean trace.

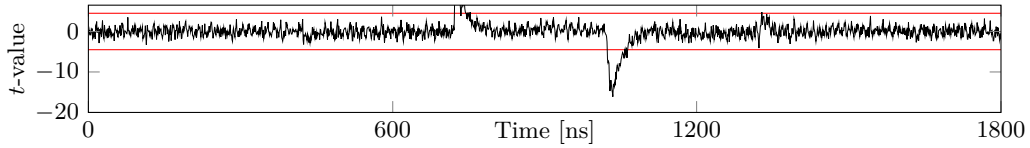
(b) First-order t-test results.

(c) Second-order t-test results.

Figure 22: No first-order t-test leakage using non-LFSR randomness for the first-order masked 8-bit SKINNY S-box (250 million traces).

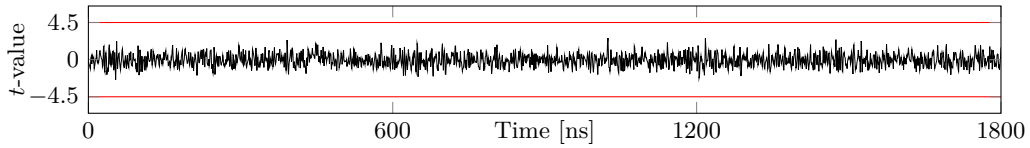


(a) First-order t-test results.

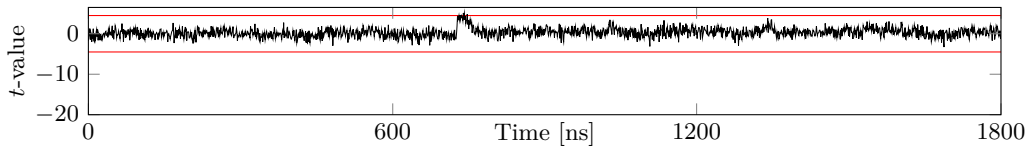


(b) Second-order t-test results.

Figure 23: First- and second-order t-test leakage using the Galois LFSR $X^6+X^4+X^3+X+1$ (2D in hex.) for the first-order masked 8-bit SKINNY S-box (100 million traces).

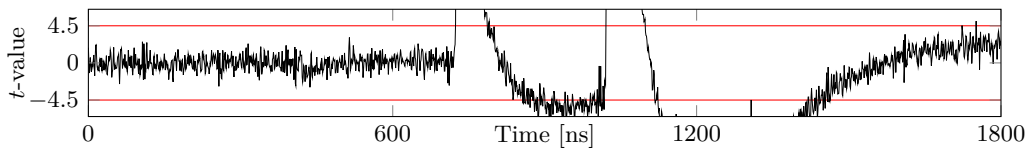


(a) First-order t-test results.

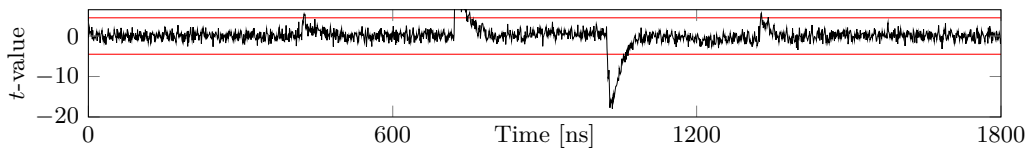


(b) Second-order t-test results.

Figure 24: No first-order t-test leakage using the Galois LFSR $X^7+X^5+X^3+X+1$ (55 in hex.) for the first-order masked 8-bit SKINNY S-box (100 million traces).

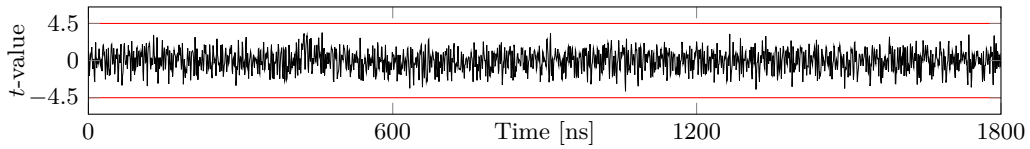


(a) First-order t-test results.

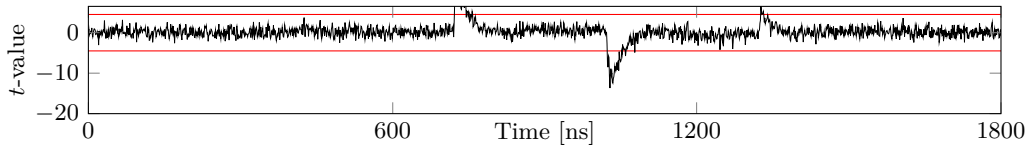


(b) Second-order t-test results.

Figure 25: First- and second-order t-test leakage using the Galois LFSR X^7+X^6+1 (60 in hex.) for the first-order masked 8-bit SKINNY S-box (100 million traces).

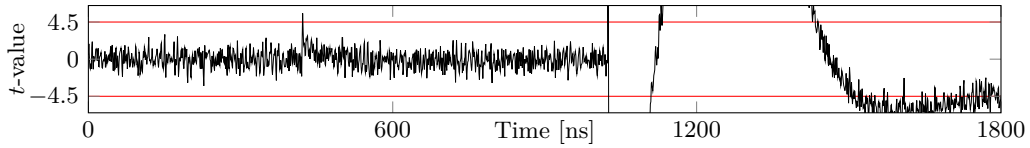


(a) First-order t-test results.

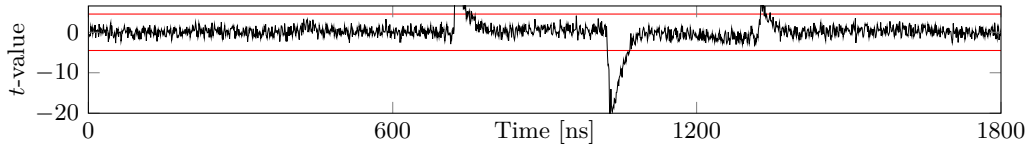


(b) Second-order t-test results.

Figure 26: No first-order t-test leakage using the Galois LFSR $X^8 + X^5 + X^3 + X + 1$ (`95` in hex.) for the first-order masked 8-bit SKINNY S-box (100 million traces).

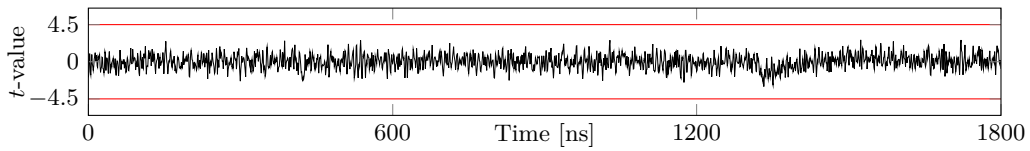


(a) First-order t-test results.

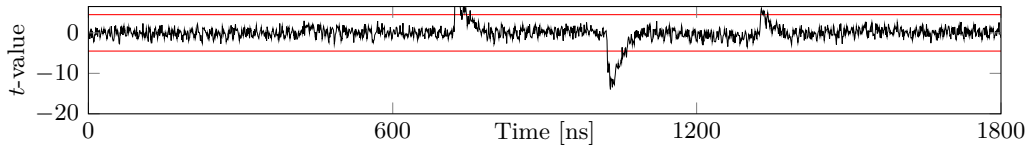


(b) Second-order t-test results.

Figure 27: First- and second-order t-test leakage using the Galois LFSR $X^8 + X^5 + X^3 + X^2 + 1$ (`96` in hex.) for the first-order masked 8-bit SKINNY S-box (100 million traces).

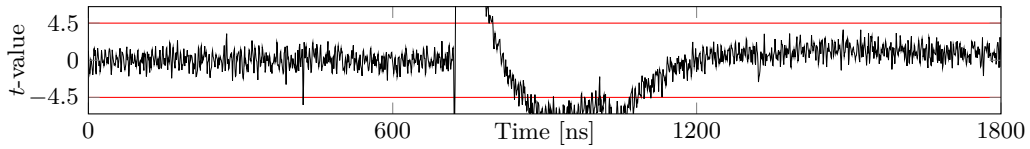


(a) First-order t-test results.

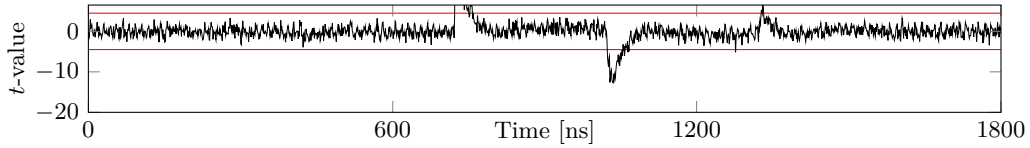


(b) Second-order t-test results.

Figure 28: No first-order t-test leakage using the Galois LFSR $X^9 + X^7 + X^6 + X^4 + 1$ (`168` in hex.) for the first-order masked 8-bit SKINNY S-box (100 million traces).

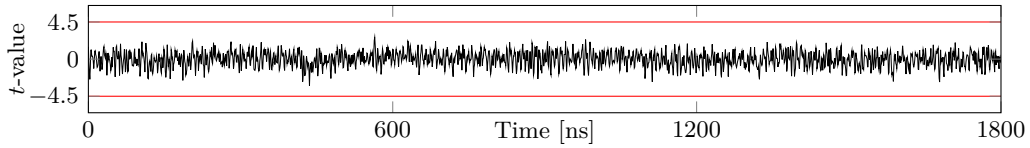


(a) First-order t-test results.

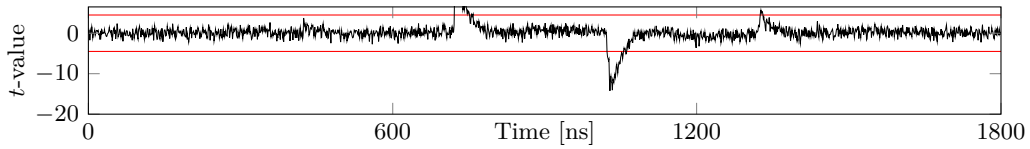


(b) Second-order t-test results.

Figure 29: First- and second-order t-test leakage using the Galois LFSR $X^9 + X^7 + X^6 + X^4 + X^3 + X + 1$ (`16D` in hex.) for the first-order masked 8-bit SKINNY S-box (100 million traces).

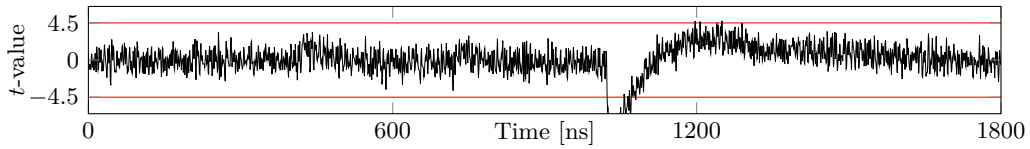


(a) First-order t-test results.

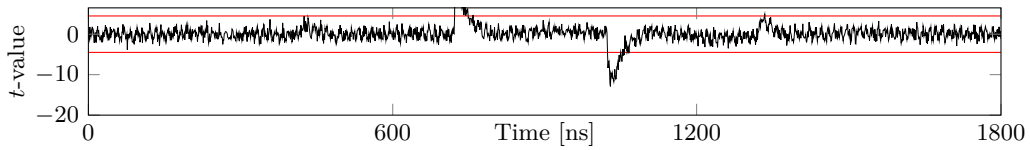


(b) Second-order t-test results.

Figure 30: No first-order t-test leakage using the Galois LFSR $X^{10} + X^7 + X^6 + X^2 + 1$ (`262` in hex.) for the first-order masked 8-bit SKINNY S-box (100 million traces).

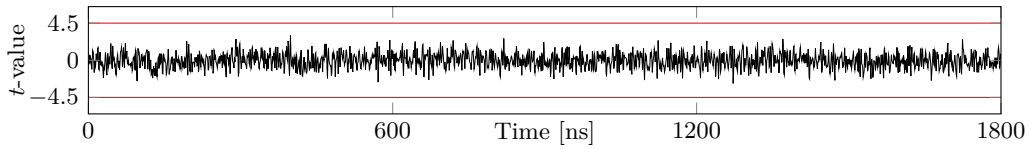


(a) First-order t-test results.

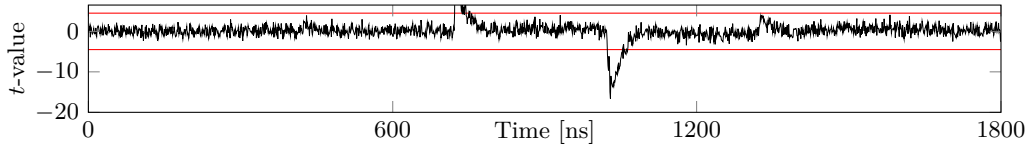


(b) Second-order t-test results.

Figure 31: First- and second-order t-test leakage using the Galois LFSR $X^{10} + X^7 + 1$ (`240` in hex.) for the first-order masked 8-bit SKINNY S-box (100 million traces).

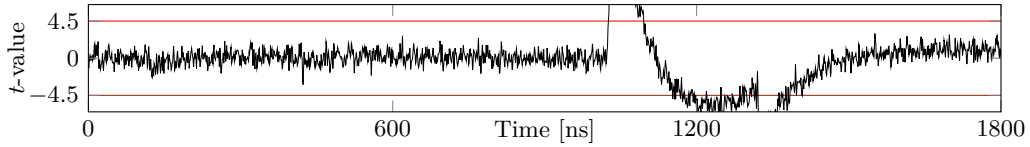


(a) First-order t-test results.

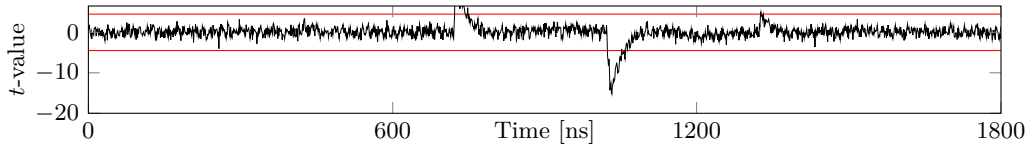


(b) Second-order t-test results.

Figure 32: No first-order t-test leakage using the Galois LFSR $X^{11} + X^7 + X^3 + X^2 + 1$ (`446` in hex.) for the first-order masked 8-bit SKINNY S-box (100 million traces).

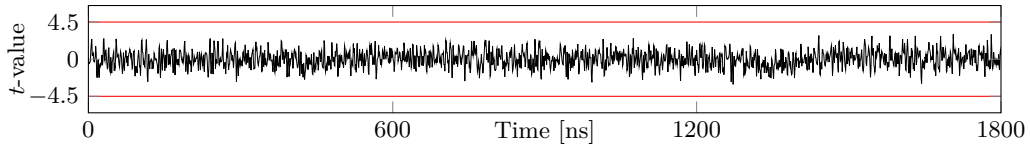


(a) First-order t-test results.

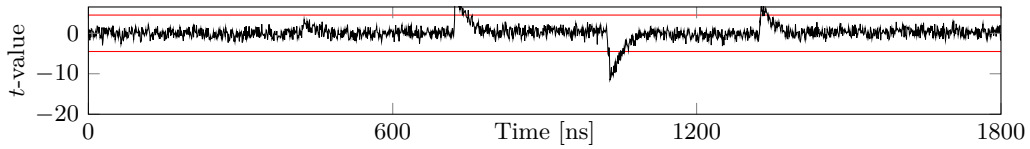


(b) Second-order t-test results.

Figure 33: First- and second-order t-test leakage using the Galois LFSR $X^{11} + X^7 + X^4 + X^2 + 1$ (`44A` in hex.) for the first-order masked 8-bit SKINNY S-box (100 million traces).

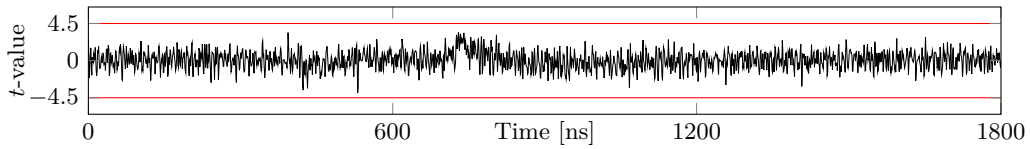


(a) First-order t-test results.

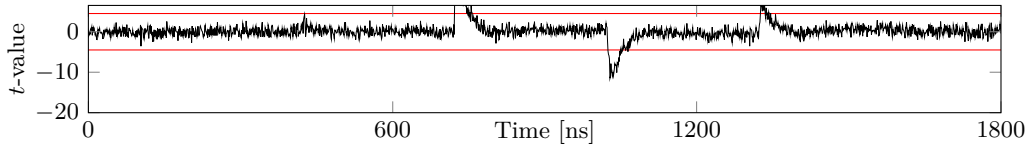


(b) Second-order t-test results.

Figure 34: No first-order t-test leakage using the Galois LFSR $X^{12} + X^{10} + X^8 + +X^4 + X^3 + X^2 + 1$ (`A8E` in hex.) for the first-order masked 8-bit SKINNY S-box (100 million traces).

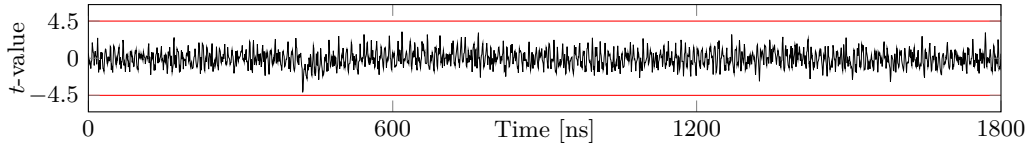


(a) First-order t-test results.

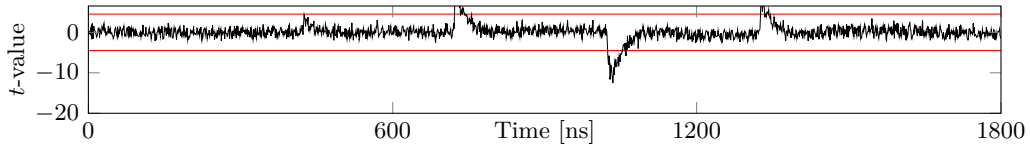


(b) Second-order t-test results.

Figure 35: No first-order t-test leakage using the Galois LFSR $X^{12} + X^{11} + X^{10} + X^6 + X^3 + X^2 + 1$ (`E26` in hex.) for the first-order masked 8-bit SKINNY S-box (100 million traces).

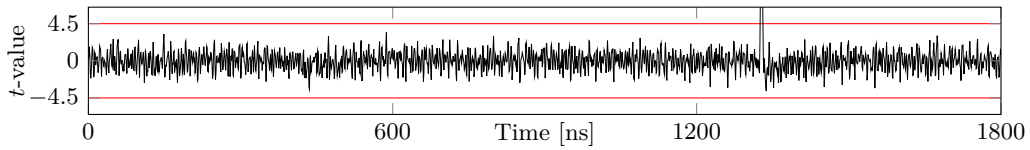


(a) First-order t-test results.

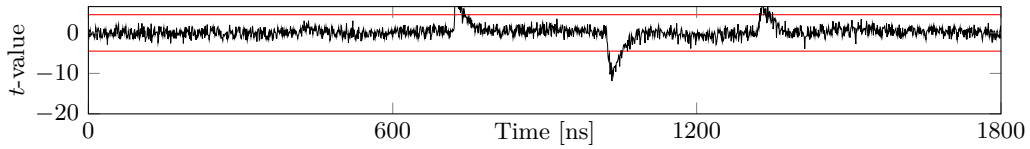


(b) Second-order t-test results.

Figure 36: No first-order t-test leakage using the Galois LFSR $X^{13} + X^9 + X^7 + X^5 + X^4 + X + 1$ (`1159` in hex.) for the first-order masked 8-bit SKINNY S-box (100 million traces).

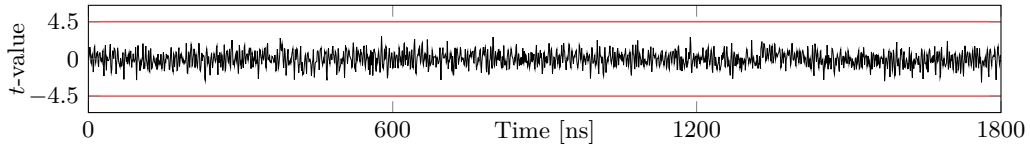


(a) First-order t-test results.

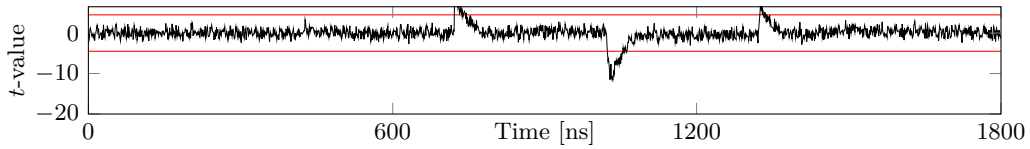


(b) Second-order t-test results.

Figure 37: First- and second-order t-test leakage using the Galois LFSR $X^{13} + X^{12} + +X^{11} + X^{10} + X^9 + X^6 + X^5 + X^3 + 1$ (`1F34` in hex.) for the first-order masked 8-bit SKINNY S-box (100 million traces).

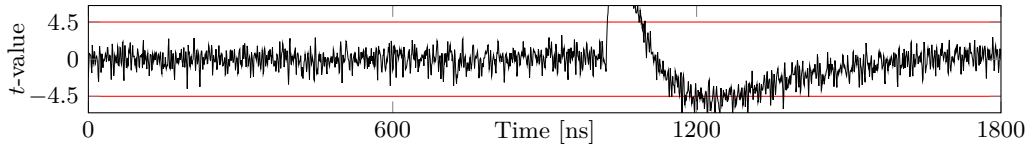


(a) First-order t-test results.

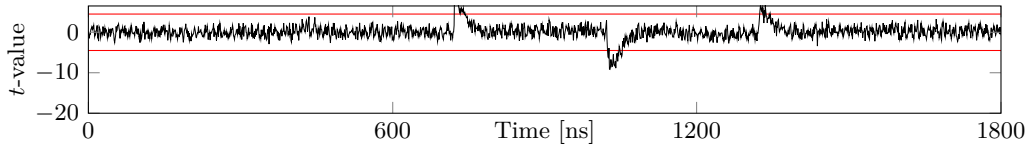


(b) Second-order t-test results.

Figure 38: No first-order t-test leakage using the Galois LFSR $X^{14} + X^{10} + X^7 + X^6 + X^4 + X + 1$ (`2269` in hex.) for the first-order masked 8-bit SKINNY S-box (100 million traces).

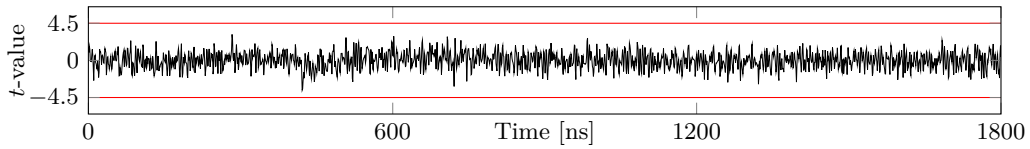


(a) First-order t-test results.

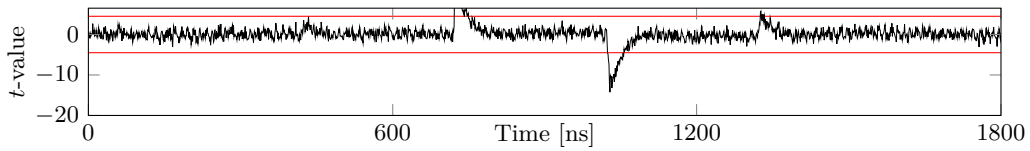


(b) Second-order t-test results.

Figure 39: First- and second-order t-test leakage using the Galois LFSR $X^{14} + X^{12} + +X^{10} + X^{10} + X + 1$ (`2A01` in hex.) for the first-order masked 8-bit SKINNY S-box (100 million traces).

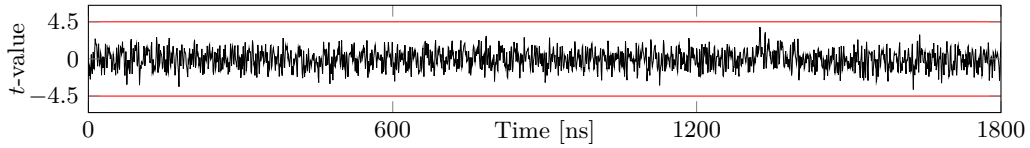


(a) First-order t-test results.

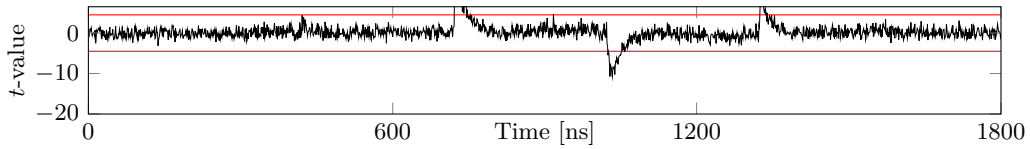


(b) Second-order t-test results.

Figure 40: No first-order t-test leakage using the Galois LFSR $X^{15} + X^9 + X^7 + X^3 + 1$ (`4144` in hex.) for the first-order masked 8-bit SKINNY S-box (100 million traces).

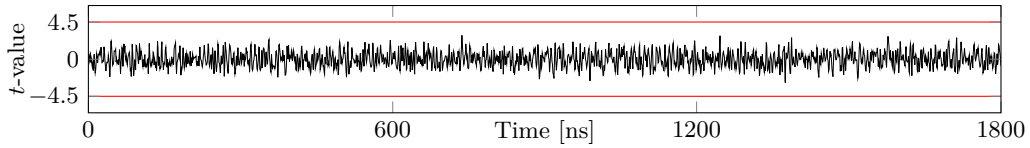


(a) First-order t-test results.

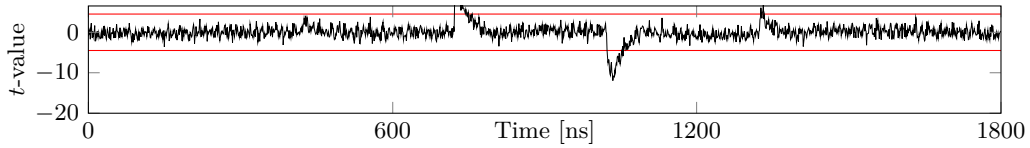


(b) Second-order t-test results.

Figure 41: No first-order t-test leakage using the Galois LFSR $X^{15} + X^{14} + X^{13} + X^{12} + X^{10} + X^9 + X^8 + X^7 + X^6 + X^5 + X^4 + X^3 + X^2 + X + 1$ (`7BFF` in hex.) for the first-order masked 8-bit SKINNY S-box (100 million traces).

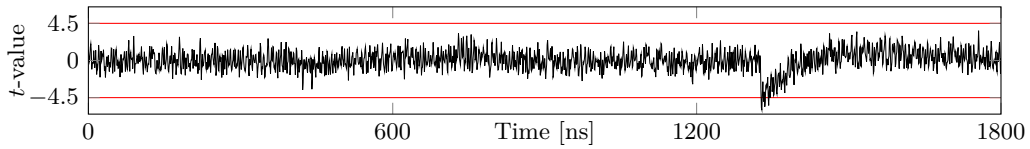


(a) First-order t-test results.

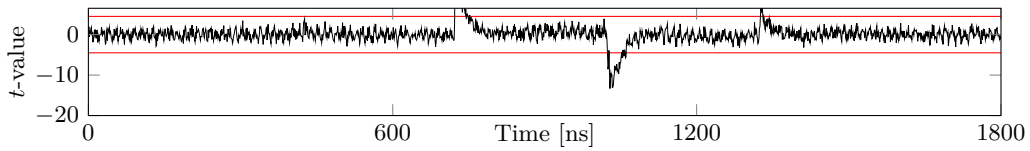


(b) Second-order t-test results.

Figure 42: No first-order t-test leakage using the Galois LFSR $X^{16} + X^{10} + X^7 + X^6 + 1$ (`8260` in hex.) for the first-order masked 8-bit SKINNY S-box (100 million traces).



(a) First-order t-test results.



(b) Second-order t-test results.

Figure 43: First- and second-order t-test leakage using the Galois LFSR $X^{16} + X^{15} + +X^{14} + X^{13} + X^{12} + X^{11} + 1$ (`FC00` in hex.) for the first-order masked 8-bit SKINNY S-box (100 million traces).