SoK: Pseudorandom Generation for Masked Cryptographic Implementation

Rei Ueno
Kyoto University
Kyoto, Japan
ueno.rei.2e@kyoto-u.ac.jp

Naofumi Homma
Tohoku University
Sendai, Japan
naofumi.homma.c8@tohoku.ac.jp

Akiko Inoue
NEC Corporation
Kawasaki, Japan
a_inoue@nec.com

Kazuhiko Minematsu
NEC Corporation
Kawasaki, Japan
k-minematsu@nec.com

Abstract—**This paper investigates pseudorandom generation in the context of masked cryptographic implementation. Although masking and pseudorandom generators (PRGs) have been distinctly studied for a long time, little literature studies how the randomness in the masked implementation should be generated. The lack of analysis on mask-bits generators makes the practical security of masked cryptographic implementation unclear, and practitioners (*e.g.*, designer, implementer, and evaluator) may be confused about how to realize it. This paper provides a novel viewpoint and comprehensive analyses by developing new three models, which correspond to respective practical scenarios of leakage assessment, quantitative evaluation of side-channel security (*e.g.*, success rate), and practical deployment. We reveal what properties are required for each scenario. In particular, we support a long-held belief/folklore with a proof: *for the output of PRG for masking, cryptographic security (i.e., randomness and unpredictability) is sufficient but not necessary, but only a statistical uniformity is necessary*. In addition, we thoroughly investigate the SCA security of PRGs in the wild in the masking context. We conclude this paper with some recommendations for practitioners, with a proposal of leakage-resilient method of comparative performance.**

1. Introduction

1.1. Background

Masking is one of the most promising countermeasures against side-channel attacks (SCAs) on cryptographic implementation [31]. Since the publication of major SCA, namely differential power analysis (DPA), by Kocher *et al.* [88], both the theoretical and practical aspects of masking have been extensively studied. Assuming that masking is correctly implemented with regard to some attack models (*e.g.*, probing model [77]), its formal security has been proven, which means that a polynomial increase of masking order yields an exponential increase of the number of traces for a successful attack (in an asymptotic sense) [12], [58], [79], [100]. Besides, some tools/methodologies for leakage detection/verification, design automation of masked implementation, and masked open-source implementations have been developed (*e.g.*, [9], [15], [85], [86], [115], [121], [122], [130], [131]), which make sound masked implementations available in practice. Although masked implementation should be realized carefully of security order degradation due to physical defaults causing a cross-share interaction (*e.g.*, glitch, coupling, and microarchitectural leakage) [6], [46], [47], [63], [65], [95], [107], [114], some masking schemes, masked implementations, and tools addressed this issue [66], [97], [108], [116], [121], [122], [135], [136].

Masked cryptographic implementation always requires (pseudo)randomness. At least, a randomness is mandatory to decompose the input into shares. Moreover, many practical masked implementations require fresh randomness during the masked computation for masking gadgets and/or mask refreshing [41], [77], [87], [108]. Thus, pseudorandom generator (PRG) is a core part of masked implementation as well as masked circuit[1]. Meanwhile, there has been little discussion about the required security and properties of PRGs in the masking context. At least, the generated mask should be uniform over the field/ring of the masking and cryptosystem. Thus far, in evaluating masked implementations, for example, a linear feedback shift register (LFSR) [128], (reduced-round) block cipher (BC)-based extendable output function (XOF) [48], and stream cipher [29], [50] have been used. However, due to a lack of detailed analysis, nobody knows which method is sufficient or insufficient for masking. In fact, the PRG output is not observable to the SCA attacker, while, in the cryptographic research field, the PRG security has been evaluated generally in the plain model, assuming that attacker can observe it. Alternatively, only the leakages of PRG are available in SCA. Nevertheless, the leakage resilience of PRGs have not been studied well in this context. It is important for practice of masked cryptographic implementation to investigate and develop a leakage-resilient (LR) methodology to generate the masking randomness securely with an acceptable performance.

Hereafter, we refer to pseudorandom bits for both the initial decomposition and refreshing in masking as *mask bits* collectively. In addition, we may refer to symmetric primitives for mask-bits generation as PRG, even though some of them are not cryptographically secure.

1. As in literature, we may call an implementation of cryptographic function *circuit* for both hardware and software, because cryptographic algorithms are frequently implemented solely by logic operations.

1.2. Our contributions

The contribution of this paper includes a comprehensive analysis of mask-bits generation using a PRG in the presence of leakage. Our contribution is threefold. First, we establish three models (black-, gray-, and white-box models) of SCAs on masked cryptographic implementation, which corresponds to respective practical scenarios: leakage assessment, SCA success rate/key-lifetime evaluation [129], and practical deployment. Second, we analyze and reveal what PRG is sufficient for secure mask-bits generation in three models/scenarios. In particular, we support a long-held belief/folklore with a proof: cryptographic security of PRG (*i.e.*, randomness and unpredictability) is sufficient but not necessary for secure mask-bits generation (for example, in [29], this folklore was described in Section 1.4 and was used as an assumption without proof for their analysis and discussion). Rather, leakage resilience of PRG is mandatory for secure practical deployment (*i.e.*, white-box model). Third, we present an LR mask-bits generation method based on our analyses, which achieves both high performance and high security guarantee.

So far, although a few studies (*e.g.*, Pietzrak [111] and Cassiers *et al.* [29]) have analyzed cases that attacker cannot observe PRG outputs but only observes the PRG leakages, this topic has not been extensively studied, especially in terms of mask-bits generation and leveled implementation. Our three modelings are the first analytical attempt and are useful, which enable us to analyze and discuss PRG in scenarios of mask-bits generation. Thus, this paper provides an important new viewpoint on this research field.

Belief summary of our analysis in white-box model. Table 1 summarizes our conclusion about the major PRGs and our proposals for mask-bits generation under the while-box attack (See Section 6 for detailed discussion). Note that all of them are sufficient for mask-bits generation under black- and gray-models. To the best of authors' knowledge, this is the first study on analysis of leakage (in)resilience of PRGs in the masking context. Existing PRGs are impractical in terms of implementation cost (as denoted by "very high") and/or have difficulty in achieving a security guarantee with regard to leakage resilience, while some PRGs may have a sufficient practical security and performance. In Table 1, "Limited" in Known SCA column means that an SCA, which correspond to a blind simple power analysis (SPA), is possible in principle but would be difficult in practice, as the success of SCAs would be limited to only specific implementation and (too) good SNR. Meanwhile, "Practical" indicates that SCA would be practical in some major settings (especially on software implementation).

1.3. Related works

In [60], Dziembowski and Pietrzak presented an LR stream cipher (DP), which achieves a provable security in the presence of some level of leakage. Namely, its output is computationally indistinguishable from random strings with adaptive bounded leakage of internal states. DP was followed by several LR stream ciphers and PRGs to improve its efficiency [56], [111], [134]. According to their leakage resilience, state-recovery SCA on them would be infeasible under their leakage model. Thus, the usage of LR-PRG/stream cipher may be promising for mask-bits generation from the security viewpoint. However, its implementation performance is too inefficient in throughput and/or circuit area. It can be slower and/or larger than the masked cryptographic implementation that we want to protect. In words, LR-PRG/stream cipher has a non-trivial gap in practice to adoption of mask-bits generation.

In [77, Section 6], Ishai *et al.* described how to construct a probing-secure PRG. Further, in [76], Ishai *et al.* presented a concept of *robust PRG* and showed a trivial construction with a linear composition of PRG and an improved construction based on an expander graph. Its security is based on a probing model. It is used to construct a private circuit provably secure against probing attackers with a cost of true random number generator (TRNG) calls independent of the circuit size. For the security against d-probing attack, they constructed a private circuit based on robust PRG with a cost of $O(d^4)$ and $O(d^{3+\epsilon})$ of TRNG calls for the trivial and improved constructions, respectively. In [40], Coron *et al.* improved the cost of TRNG calls for an ISW private circuit to $O(d^2)$ based on a *multiple PRG* strategy (CGZ), instead of usage of robust PRG. They evaluated masked AES implementation with the ISW private circuit with multiple PRGs, and confirmed its practicality in performance for some orders. Its security is also defined over a d-probing model. However, in these studies, it is not mentioned what PRG(s) should be used in the multiple PRG strategy. Notably, in these works, *the PRG(s) are considered a black-box primitive outside the masked circuit and are not considered in their security analysis, which means that they implicitly assumed that the PRGs are leak-free*. Actually, the leakage and SCA feasibility on PRGs has rarely been evaluated in the context of masking. In this paper, we argue that the SCA security of PRG should be evaluated *apart from probing model* in Section 6. The concept of robust PRG is essentially different from ours, as one of our major goals is how to realize a PRG securely in masking context. In this sense, our results would incorporate the results in [40].

In [29], Cassiers *et al.* discussed hardware masking and claimed a vulnerability of LFSR in generating mask bits. Because XOR of some bits of LFSR output is fixed to 0 owing to the construction of feedback polynomial, mask bits should be used with avoiding such a mask cancellation. They also demonstrated state-recovery SCA on LFSR in mask-bits generation vulnerability, corresponding to our white-box model. However, they demonstrated a case study of LFSR-based mask-bits generation, but neither showed its generality nor formal security analysis. Besides, they evaluated the performance of PRGs for hardware masking, and presented a usage of unrolled Trivium to generate mask bits due to its performance. As our main focus is the security aspect of mask-bits generation, readers interested in quantitative comparison of PRG performance are advised to see [29].

TABLE 1: Major symmetric primitives for mask-bits generation and our proposals (bold) under white-box SCA

Primitive	Typical instance	Implementation cost	Leakage resilience	Known SCA
Plain LFSR	Galois LFSR, Fibonacci LFSR	Low	No	Practical [27], [29], [82], [92], [124]
Improved LFSR	Mersenne twister [102], Xorshift [96]	Low	No	Probably limited†
Stream cipher	Trivium [43], [44]	Low	No	Limited [89]
LR-PRG/stream cipher	DP [60]	Very high	Yes	No
(Reduced-round) block cipher	CTR_DRBG [8], counter-based XOF	Medium	No	Practical [49], [81]
	OFB-based XOF	Medium	No	No (SPA only)
One-way hash	Hash_DRBG [8] counter-based XOF	Medium	No	Probably practical (same as CTR_DRBG)
	HMAC_DRBG [8], OFB-based XOF	Medium	No	Limited [83] (SPA only)
Permutation	Sponge-based XOF [20], [53]	Medium	Somehow‡	Limited [83] (SPA only)
(LR-)PRF-based XOF	GGM [67], FPS [64], MSJ [104]	Very high	Yes	No (SPA only)
	Sum of permutation [34], trancated BC [16]	Same as block cipher		
Robust PRG	IKL+ [76]	Very high	d-probing	$(d+1)$-probing
Multiple PRG	CGZ [40]	Low	d-probing*	$(d+1)$-probing
Leveled mask-bits generator	**Ascon + LFSR/stream cipher**	**Low**	**Yes**	**No (SPA only)**
Masked stream cipher/XOF	**Masking without fresh randomness**	**High**	**First order**	**Second-order blind SCA/SPA**

† SCA on LFSR would be applicable to them in principle [27], [82], [92], [124], but it might be difficult to practical parallelized implementations.
‡ Leakage resilience of sponge/duplex-based encryption is discussed in, *e.g.*, [18], [19], [55].
* Security for masked circuit, but not for distinct PRG (See Section 6.2).

1.4. Paper organization

Section 2 introduces the basics of masking. Section 3 describes the modelings of masking randomness generation and attack on it. Sections 4 to 6 analyze and discuss SCA on masked implementation in the black-, gray- and white-box models, respectively. Section 7 presents our recommendations for practically secure mask-bits generation among the existing major PRGs for practitioners. In addition, we present an LR primitive based on our observations, which has both high security guarantee and comparative performance. Finally, Section 8 concludes this paper.

2. Preliminaries

2.1. Notations

An uppercase letter (*e.g.*, X) denotes a random variable/vector and a lowercase character (*e.g.*, x) denotes its realization, unless otherwise defined. Let p_X denote the probability mass or density function of X. We denote a sequence of n random variables $X_1, X_2, \dots, X_n$ by X^n. We may write a keyed function as $F(K, \cdot) = F_K(\cdot)$.

2.2. Masking

Additive masking is a major countermeasure against SCA [31], [94][2]. In a masked implementation, each secret variable is decomposed into *shares* over a group action of the cryptosystem. In a d-th order masking, a secret variable z is represented using $d+1$ shares $a_0, a_1, \dots, a_d$ such that

$$z = a_0 \circ a_1 \circ \dots \circ a_d,$$

where $\circ$ denotes the addition over the underlying field/ring and each share is sampled from a uniform distribution of its additive group. For example, as many symmetric primitives utilize $\mathrm{GF}(2)$ or its extension field (*e.g.*, AES, GHASH, Trivium, SHA-3, *etc.*), the addition is given by a bit-wise XOR; namely, it holds that

$$z = a_0 \oplus a_1 \oplus \dots \oplus a_d.$$

This is called *Boolean masking*. In contrast, some symmetric primitives employ arithmetic addition over $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ (*e.g.*, addition-rotation-XOR (ARX) primitives like SHA-2 and ChaCha20) and prime-field addition/multiplication [99]; arithmetic and prime-field maskings are defined over its respective additive group. Moreover, in recent, several masked implementations of lattice-based cryptography have been developed, which utilize a polynomial ring $\mathbb{Z}_q[x]/\langle f(x)\rangle$ [1], [3], [4], [23], [36], [37], [38], [39], [51], [61], [109]. Some cryptosystems utilize maskings in a combination with mask conversion methods [26], [69]. For the simplicity, we hereafter consider Boolean masking, which does not compromise the generality of our statements in this paper.

Masked linear operation can be trivially implemented by duplicating the circuit for each share, because, for any linear function L and $z = a_0 \oplus a_1 \oplus \dots \oplus a_d$, it holds that $L(z) = L(a_0) \oplus L(a_1) \oplus \dots \oplus L(a_d)$. In contrast, it is non-trivial to implement masked non-linear circuit in a secure manner. For example, based on an Ishai–Sahai–Wagner (ISW) masking scheme (a.k.a. private circuit) [77], a masked AND gate for shared input $a = (a_0, a_1, \dots, a_d)$ and $b = (b_0, b_1, \dots, b_d)$ computes its output $c = (c_0, c_1, \dots, c_d)$ as follows: (i) draw random bits $m_{i,j}$ for $0 \leq i < j \leq d$ as fresh randomness, (ii) compute intermediate values $m_{j,i}$ as

$$m_{j,i} = (m_{i,j} \oplus a_i b_j) \oplus a_j b_i,$$

and (iii) compute the output share as

$$c_i = a_i b_i \oplus \bigoplus_{i \neq j} m_{i,j},$$

for each i. The ISW masking is followed by many masking scheme, such as (software) threshold implementation

2. Here, we introduce an additive masking as it is the most popular among masking schemes. Yet, our discussion and theory would be valid and applicable to other masking schemes such as multiplicative masking [50], [68], inner-product masking [5], [59], code-based masking [133], and polynomial-interpolation masking [70].

(TI) [66], [108], domain-oriented masking (DOM) [72], unified masking approach (UMA) [71], and gadget-based masking [10], for improved efficiency, enhanced security, composability, and resolving security degradation by physical defaults causing a cross-share interaction (*e.g.*, glitch [95], coupling [46], and microarchitectural leakage [65], [66]).

2.3. Probing-related attack models and notions

Attack models and security notions have been developed to soundly define the d-th order attack and masked implementation, verification of masked implementation, *etc.* In the seminal study in [77], Ishai *et al.* presented the *d-probing model*, representing an attacker using d probes to obtain the intermediate values in the circuit. A masked circuit is d-probing secure if an adversary with a black-box access to the circuit (without any probes) can simulate a joint distribution of any d tuple of intermediate values, or simply and intuitively, any combination of d (non-adaptive) probes on the masked circuit recovers no secret variable [118]. The probing model has been extended in multiple aspects. For example, Barthe *et al.* presented *non-interference (NI)* and *strong NI (SNI)* notions to develop a gadget-based (composable) masking [10], [11]; Cassiers and Standaert presented the *probe-isolating NI (PINI)* notion for more efficient and simpler composable gadgets [30]; Reparaz *et al.* presented the *glitch-extended probing model* to develop a higher-order masked hardware in the presence of glitches [116]; and Faust *et al.* presented the *robust probing model* to capture the physical defaults for composable masking schemes [63].

These probing-related models/notions are useful to define a sound d-th order masking and verify its implementation. Meanwhile, these models/notions do not necessarily represent real-world SCAs such as DPA, correlation power analysis (CPA) [25], and deep-learning based SCA (DL-SCA) [91], [98]. Fortunately, some security reductions to more realistic models, such as random-probing and noisy-leakage models, are known [13], [14], [24], [57]. Thus, masked implementation based on these models/notations are currently considered promising for the d-th order security.

However, although the design of masked circuits is getting mature, how to generate the mask bits have been rarely discussed. In [77], Ishai *et al.* mentioned a construction of d-probing secure PRG using a normal PRG. In [76], Ishai *et al.* presented a robust PRG that can be used for secure mask-bits generation, but its cost is non-negligibly high. In [40], Coron *et al.* improved the cost by multiple PRG strategy to achieve d-probing security using normal PRGs. However, in Section 6.2, we mention a pitfall of d-probing model to evaluate the SCA security of mask-bits generators.

2.4. Security proof of masking

The above models/notions define d-th order masking and SCAs. As a sound security scheme, the advantage or success rate (SR) on d-th order masked implementation should exponentially decrease by increasing d polynomially, because an attacker on d-th order masked implementation is supposed

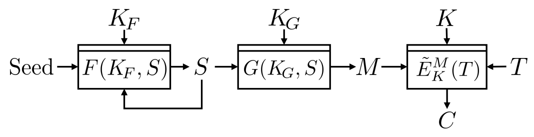


Figure 1: A model for masked cryptographic implementation with mask-bits generator.

to perform a $(d+1)$-th order SCA. For the proof, noise and leakage function/model play an essential role. The first proof of this kind is shown by Chari *et al.* [31] for a single-bit DPA, assuming that the measurement noise is Gaussian and sufficiently large. This is followed and extended by Prouff and Rivain [113] using *noisy leakage functions*. In [58], Duc *et al.* proved the security of masking under the noisy leakage model. The proof holds when the noise is very large; namely, $I(Z;\boldsymbol{X}) \leq 2^{-2v+1}$ holds, where $I(Z;\boldsymbol{X})$ is the mutual information between a share and side-channel leakage[3], and v is the bit length of partial secret key under SCA. In [79], Ito *et al.* provided another proof with much more relaxed condition of $I(Z;\boldsymbol{X}) \leq (2\ln(2))^{-1}$ by modeling SCA on masked implementation as a communication channel. In addition, similar bounds under the same condition are found by Masure *et al.* and Béguinot *et al.* [12], [101].

These bounds hold if the mask-bit generator is not attacked, and the generated mask-bits are ideal random numbers. However, in practice, we should consider how to generate the random numbers with regard to the leakage and side-channel security of mask-bits generator, which has been rarely discussed. This study would solves the gap between theory and practice in terms of mask-bits generation.

3. Modeling mask-bits generation under SCA

3.1. Modeling of masked implementation

Let n_f, n_g, n_s, and n_k be the bit lengths of internal state, generated mask bits per state, initial seed, and secret key, respectively, Figure 1 illustrates the abstract of masked implementation of encryption E_K with a PRG, where $\tilde{E}_K^M$ denotes masked circuit of E_K that satisfies $\tilde{E}_K^M(T) = E_K(T)$ for any mask value M, key K, and plaintext T. The PRG consists of a state updating function F and an output function G, and has an n_f-bit internal state computed from an n_s-bit initial seed and/or an n_k-bit secret key. Let S_j be the j-th state, and M_j be the j-th output. Given an n_s-bit seed, the PRG initializes the n_f-bit internal state as S_0. Then, the PRG updates the internal state using F as $S_{j+1} = F(S_j)$, while it generates an n_g-bit mask value M_j using G and state as $M_i = G(S_j)$.

Example 1: LFSR. Consider an n_f-bit LFSR. The initial state S_0 is given as the initial seed ($n_f = n_s$). The state updating function F is given by $S_{j+1} = \mathrm{double}(S_i)$, where

3. This mutual information is bounded from above by the signal-to-noise ratio (SNR) of measurement according to the Shannon–Hartley theorem, if we consider an additive Gaussian noise. Note that the SR of SCAs is generally bounded from above by the mutual information [45].

the function double denotes the doubling over $\mathrm{GF}(2^{n_f})$, and the output function G is $\mathrm{MSB}(S_j)$ (the most significant bit of S_j). Neither K_F nor K_G is given ($K_F = K_G = \mathit{null}$).

Example 2: Counter-based XOF. Let H and E_{K_G} be a one-way hash function and BC with a key K_G, respectively. The initial state is given by an n_s-bit initial seed padded with an $(n_f - n_s)$-bit zero, where $n_s < n_f$. Here, F is given by a counter as $S_{j+1} = S_j + 1$ and G is $H(S_j)$ or $E_{K_G}(S_j)$.

Remark 3.1 (Bit length of M and unrolling F and G). In practice, the bit length of M (*i.e.*, n_g) depends on the implementation and required the number of mask bits per clock cycle and evaluation of gadget, Sbox, or encryption, as a large amount of mask bits is sometimes required (for example, AES requires at least $128d$ bits for initial decomposition). To generate mask bits of a sufficient length for one block encryption, the PRG should repeatedly call F and G multiple times (or should be implemented in parallel). Meanwhile, cryptographic hardware has been realized in an *unrolled* manner (see [29]). As such, we should consider F and G repeatedly called or unrolled to generate sufficient bit length of M per unit, instead of its single round function. However, this has little impact on our discussion about black- and gray-box models because it does not change M essentially, while SCA resistance of the implementation would depend on unrolling (related to white-box model).

Remark 3.2 (Correlated (pseudo)randomness for mask bits). For a Boolean masking, we actually need *additively correlated randomness* [2], which means that the generated d' random numbers $R_0, R_1, \ldots, R_{d'}$ should satisfy $R_0 \oplus R_1 \oplus \cdots \oplus R_{d'} = 0$. Here d' is frequently equivalent to d, whereas some masking schemes (especially for hardware) require $d' \geq d$ [116]. The initial decomposition of Z is typically given as $(A_0, A_1, \ldots, A_d) = (R_0 \oplus Z, R_1, R_2, \ldots, R_d)$ as $Z = A_0 \oplus A_1 \oplus \cdots \oplus A_d$ and refreshing should preserve the secret value in adding the fresh randomness. In other words, refreshing is equivalent to adding masked zero. A typical way to generate such correlated randomness for mask bits is to generate d random numbers $M_1, M_2, \ldots, M_{d'}$ and then to give $R_i = M_i$ for $1 \leq i \leq d'$ and $R_0 = \bigoplus_{i=1}^{d'} M_i$ (If $d' = 1$, then $(R_0, R_1) = (M_1, M_1)$ [72]) or ring refreshing [10], *e.g.*, $(R_0, R_1, R_2 = M_0 \oplus M_1, M_1 \oplus M_2, M_2 \oplus M_0)$ if $d' = 2$.

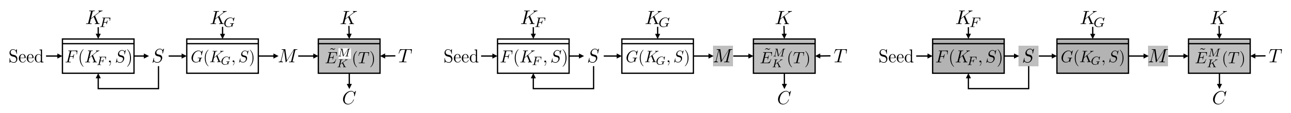
3.2. Adversarial models

The attacker is assumed to observe the leakage from the target d-th order masked circuit and/or the mask-bit generator. The attacker cannot directly observe the mask bits (*i.e.*, outputs of PRG) as in common masked implementation. Note that, if the mask bits unboundedly leak to the attacker (*i.e.*, mask bits are directly observable), the masked implementation can no longer be secure and the first-order SCA would trivially work using the knowledge of mask bits. As well, the attacker knows neither the initial seed nor secret key of PRG in any scenario. We classify the adversarial model into three types, depending on the available leakage. Intuitively, the model names correspond to the attacker's knowledge on the target implementation. Note that the attacker is assumed to always know plaintext and/or ciphertext as in common SCAs.

Black-box model. The attacker can only use one or up to d probe(s) on the masked circuit $\tilde{E}_K^M$, but cannot set the probe(s) on the mask-bits generator (*i.e.*, F, G, and S), nor the generated mask M directly. This model corresponds to up-to d-th order SCA attacker on d-th order masked implementation, and practical leakage assessment methodologies (*e.g.*, TVLA [119]) which does not consider the leakage of mask-bits generator. As the relation to practical SCAs (*e.g.*, CPA) or leakage assessment rather than d-probing attack, the black-box attackers are defined using a limitation of *leakage and selection function* which the attacker can use; namely, the attacker is allowed to use a selection and leakage function with *no input of M (or not more than d shares among $A_0, A_1, \ldots, A_d$)*. For example, although the leakage of first-order masked implementation is frequently given by $\mathrm{HW}(A_0)$ and $\mathrm{HW}(A_1)$ with noise, the black-box attacker is not allowed to take both A_0 and A_1 for the leakage and selection function. More generally, in attacking a d-th order masked implementation with $d+1$ shares, a leakage function which the attacker can use is limited to a function with inputs of up to d shares. Hence, if the implementation has no flaw in both masked circuit and M, the black-box attack is never successful (*i.e.*, its key-recovery success rate must be equal to that of a random guess) and no leakage is detected.

Gray-box model. The gray-box attacker can use $d + 1$ probes on the masked circuit, while the attacker is not allowed to set probe(s) on the mask-bits generator, or does not have access to their leakages. This model corresponds to a $(d + 1)$-probing attack on a d-th order masked implementation with $d + 1$ shares. For practical SCAs, this model corresponds to security evaluation against higher-order SCAs, which do not utilize the PRG leakage as in the literature. Probing-related model/notions tells that d-th order masked implementation is not secure against such a $(d+1)$-probing attacks. However, in practice, adversary can take any d unboundedly; hence, it is important to consider the difficulty/feasibility of $(d + 1)$-th order SCAs on d-th order masked implementation (typically as success rate (SR) metric [126]), apart from probing-related models and the black-box model. As mentioned in Section 2.4, it is proven that the SR (*i.e.*, the advantage of side-channel adversary) decreases exponentially by increasing d [12], [31], [58], [79], [100], [113]. Besides, there are some types of higher-order SCAs, such as using a composition function (*e.g.*, product and sum) of d sample points, biased mask attack, and collided mask attack [94]. In Section 5, we mention a construction of the most efficient higher-order SCA in terms of SR and the number of traces, under this gray-box model.

White-box model. A white-box attacker can use any number of probes on everywhere in the masked implementation including F, G and S, and can utilize its leakage, in addition to those of M and masked circuit. The attacker can perform SCA on the mask-bits generator to retrieve its internal state and/or secret key, although the attacker cannot observe inputs nor outputs of the PRG. SCA without information of input nor output (*i.e.*, leakage-only SCA)



(a) Black box (b) Gray box (c) White box

Figure 2: Adversarial models for masked cryptographic implementation, where gray parts (boundedly) leak to attacker.

is called *blind SCA* [35], [90]. If they are retrieved, the attacker can guess the mask bits with success probability 1; accordingly, the security order is degraded to zero as the attacker can perform first order SCA using guessed mask bits. This implies that, in the while-box model, the security of whole masked implementation depends on the lower security of either PRG or masked circuit. Therefore, it is important for the practice of masked implementation to assess the (in)vulnerability of PRG to blind SCAs. We comprehensively analyze the SCA security of major PRGs in the context of mask-bit generation in Section 6.

3.3. Trivial/baseline attacks

On frequency of PRG. If attacker knows the PRG frequency, the attacker can trivially obtain traces with an identical mask and can perform a first-order SCA using them. In practice, the PRG frequency should be set sufficiently large, so that such an attack gets infeasible. In other words, the PRG frequency should be determined to render such an attack difficult. As PRG frequency usually increase exponentially by its state size, we confirm the exponential security of masking against such a trivial attack. In this paper, we assume that PRG frequency is sufficiently large.

Mask guessing attack using secret key and seed. An adversary cannot observe the mask bits directly. The mask bits should be sufficiently hard-to-guess for the attacker. If the attacker can guess the mask bits with success probability 1, a first-order SCA would trivially work using the mask bits as additional input(s) of the selection function. Therefore, the PRG of mask-bits generator should have a sufficient bit length of secret key and/or seed, such that M is computationally hard-to-guess. For black- and gray-box attackers, guessing the secret key and seed is only the way to guess M, because the attackers are not allowed to put probes on the PRG. Contrarily, Section 5 describes how a mask guessing attack works on a higher-order masked implementation if the a seed/key is successfully guessed. Further, Section 6 discusses the SCA security of PRGs against state and key recovery SCAs in the white-box model.

4. Analysis of black-box adversary

4.1. Properties of mask bits required for security

In this paper, we assume that the masked circuit has no flaw and is soundly implemented. This assumption and the black-box model are useful to discuss the security order degradation due to flaw of mask bits. As a trivial example, F and G are an identity function (*i.e.*, G outputs a constant value) or the output of G is highly biased far from uniform, in which the attacker would successfully recover the secret key solely by a first-order SCA, implying that the security order is degraded to zero no matter how large d is. By contrast, it has been rarely discussed what properties are required for mask-bits generation to achieve a secure masked implementation (*e.g.*, statistical randomness, computational unpredictability, or unreproducibility). In fact, the security of masking with non-cryptographic PRGs (*e.g.*, LFSR) has been unclear, while some previous works utilized an LFSR and reduced-round BC-based XOF for mask generation and experimentally showed its leakage assessment. For example, in [33], [128], it was shown that the first-order TVLA does not find leakage in their first-order masked implementations with an LFSR. Using the black-box model, we discuss how to generate the mask bits for secure masked implementation.

In the ISW paper [77], Ishai *et al.* mentioned that random bits are generated by a PRG, but did not mention what property is required for the security of the ISW private circuit. In addition, in the TI paper [108], Nikova *et al.* mentioned that the shares and mask should satisfy *uniformity* (*i.e.*, the occurrence probability of any value of shares is a constant independent of the secret value) and *statistical independence*, but did not mention whether the mask should be generated randomly. Actually, no fresh randomness is required for the first-order TI if the shared function satisfies the output uniformity, rendering the masked circuit deterministic. Furthermore, the security proof of masking [12], [13], [58], [79], [101] also assume that shares are uniformly distributed and shares and leakages are independent. Some of these studies would only mention that uniformity and independence are mandatory for the security of masked circuit, but *(psuedo or true) randomness*, *unpredictability*, or *unreproducibility* sometime has not been utilized to develop masking scheme or prove the security. In fact, in the black-box scenario, cryptographic PRG is sufficient but not necessary for generating mask bits. To prove it, we introduce Theorem 4.1 as our key observation.

Theorem 4.1. *Given a dataset of side-channel traces, the result of common leakage assessment methods and (higher-order) SCAs is invariant to the order of traces.*

Proof. See Appendix A. □

Roughly, the main reason of Theorem 4.1 would be that SCA and leakage assessment are based on *sufficient statistics* on independent and identically distributed (i.i.d) samples (*i.e.*, traces), implying that the order of traces is supposedly unnecessary to explain the characteristics of side-channel traces and key recovery. In other words, Theorem 4.1 holds as long as the side-channel traces are (assumed to be) i.i.d.

Unnecessity of cryptographic randomness and unpredictability. For a masked implementation, the trace order corresponds to the order of mask values (*i.e.*, how the mask bits are generated), because the j-th execution of masked circuit, from which attacker observes $(T_j, \boldsymbol{X}_j)$ $(1 \leq j \leq n)$, uses the j-th mask value M_j. Theorem 4.1 indicates that the trace order in the leakage assessment or SCA on masked implementation does not affect its result, and *the order of generated mask bits/value does not make sense for the attack result*. To be concrete, we have Corollary 4.1.

Corollary 4.1. *Let* $\{ (T_j, \boldsymbol{X}_j) \mid 1 \leq j \leq n \}$ *be the attack dataset consisting of* n *plaintexts and* n *side-channel traces, where the* j*-th execution with* $(T_j, \boldsymbol{X}_j)$ *uses mask value* M_j*. For any permutation* $\pi_n : \{1, 2, \dots, n\} \to \{1, 2, \dots, n\}$*, the SCA result using* $\{ (T_{\pi_n(j)}, \boldsymbol{X}_{\pi_n(j)}) \mid 1 \leq j \leq n \}$ *is identical to that using* $\{ (T_j, \boldsymbol{X}_j) \mid 1 \leq j \leq n \}$*.*

Proof. It is obvious from Theorem 4.1. □

Using Corollary 4.1, we show that randomness and unpredictability are unnecessary for secure mask-bits generation. Assume that M_j can take a value from $\{0, 1, \dots, n-1\}$ and mask-bits generator generates n different values during n cryptographic executions, where $n = 2^{n_g}$ (n_g is the bit length of M). Consider two mask-bits generators with an upcounter and a cryptographic random generator, where the j-th bit mask value associated with (T_j, X_j) is $M_j = j$ and $M_j = \pi_n(j)$, respectively (here, π_n is a random permutation over $\{0, 1, \dots, n-1\}$). Obviously, the upcounter does not have the security as cryptographic PRG. However, due to Corollary 4.1, the results of SCA are identical for both mask-bits generators based on upcounter and cryptographic PRG, as there exists a permutation $j \mapsto \pi_n(j)$ of Corollary 4.1. Thus, as far as $M = (M_1, M_2, \dots, M_n)$ is uniformly distributed over $\{0, 1\}^{n_g}$, the masked circuit should be secure regardless of the order of mask values.

Property required for security. Cryptographic randomness and unpredictability is unnecessary for black-box security of mask-bits generator. Instead, as the order of generated mask bits/value does not make difference, it would be sufficient for the security of masked circuit that *the empirical distribution of mask bits/values* used in the SCA/leakage assessment corresponds to a uniform distribution solely. Thus, even an non-cryptographic mask-bits generator would be acceptable if its empirical distribution from samples of $M_1, M_2, \dots, M_n$ corresponds to a uniform distribution for any seed/key but *the values do not have to be selected at random* (which would be indeed natural because any PRG is deterministic). Note that, in practice, upcounter does not provide the security because the generated values are not uniform in $\{0, 1\}^{n_g}$ when 2^{n_g} (*i.e.*, PRG frequency) is far greater than the number of traces n (as upper bits in M are not changed), and is vulnerable to mask brute-forcing[4].

4. Concretely, upcounter-based generator is not uniformly equidistributed even in one-dimension, which is easily exploited by guessing initial mask.

Secure mask-bits generation. For generating statistical randomness that has an empirical distribution corresponding to uniform distribution for any non-prohibited seed, we should consider how the generated bit stream is (equi)distributed in high dimensions. In words, for secure mask-bits generation, each bit/value of M should be unbiased (*i.e.*, one-dimensional equidistributed uniformly) as well as M should has neither autocorrelation nor bias in multiple bits; namely, it should be high-dimensionally uniform equidistributed. Although it is impossible in practice to generate *truly* uniform equidistribution in any finite dimension (even for cryptographic PRG), a fairly high-dimensional uniform equidistribution PRG has been commonly used, which would be sufficient for secure mask-bits generation. A typical example would be Mersenne twister [102], output of which is proven equidistributed uniformly over k-dimensional space with a 32-bit accuracy for all $1 \leq k \leq 623$. As well, other improved LFSRs (*e.g.*, Xorshift) provide a high-dimensional uniform equidistribution. In addition, the outputs of an n_g-bit plain LFSR can be equidistributed uniformly in at most n_g-dimension with one-bit accuracy, which implies that LFSRs with a sufficiently large n_g would be sufficient for mask-bits generator. Of course, cryptographic PRGs (*e.g.*, stream cipher and XOF) would be appropriate, as unpredictability implies a high-dimensional uniform equidistribution. In contrast, the linear congruential generator in the standard library would be inappropriate, because it is not equidistributed uniformly in spaces of a dimension of more than one.

Relation to [29]. Cassiers *et al.* claimed that an LFSR-based mask-bits generator can lead to an insecurity due to its linearity. Namely, according to the LFSR feedback polynomial, there exist mask-bits which always satisfy $m_{j_1} \oplus m_{j_2} \oplus \cdots \oplus m_{j_c} = 0$, where m_{j_b} is the j_b-th bit of generated mask. If $m_{j_1}, m_{j_2}, \dots, m_{j_c}$ are XORed to obtain a correlated pseudorandomness for a refreshing gadget or initial decomposition, then the resultant bit is fixed to zero, leading to an exploitable leakage. This implies that, in this case, the mask bits are not uniformly equidistributed even in one dimension due to the construction of refreshing gadgets and defining polynomial. To ensure the security, a correlated pseudorandomness for d-th order masking should be uniformly equidistributed in d dimension with a care of linearity, which can be easily obtained from higher than $(d-1)$-dimensional uniform equidistribution.

4.2. Relation to practical scenario and take away for practitioners

Black-box model corresponds to leakage assessment. The existing leakage assessment methodologies (*e.g.*, TVLA [119] and its variants [105], [106], [117]) do not consider leakage of PRG. Furthermore, to examine if a d-th order masking has no d-th order leakage (*i.e.*, to examine if a d-th order masked circuit is soundly implemented), the evaluator perform a d-th order leakage assessment (*e.g.*, TVLA) or experimental attack, which corresponds to a d-probing on $d+1$ shares. Thus, the black-box model cor-

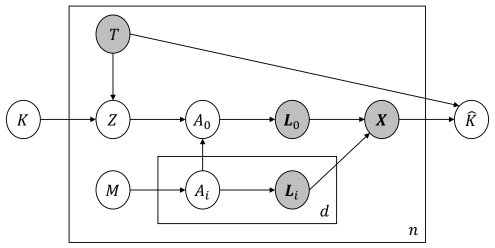


Figure 3: Graphical models representing gray-box SCA.

responds to a d-th order leakage assessment on d-th order masked implementation conducted by evaluator. Our analyses on black-box model state that PRGs including even non-cryptographic ones, such as LFSR, is sufficient for leakage assessments as long as PRG provides a high-dimensional uniform equidistribution, as in existing studies [128].

5. Analysis of gray-box adversary

5.1. Notations and graphical model

We introduce notations with a graphical model, which is an extension of a communication channel model representing SCA on masked implementation in [79]. Figure 3 shows the graphical model, where gray random variables are plaintext or leakage observable by attacker. Note that this probabilistic model does not represent an actual implementation, but represents relation among random variables (for example, Z does not directly appear in actual implementation). Here, random variables are defined as follows:

- $T^n = (T_1, T_2, \ldots, T_n)$: Plaintext or ciphertext observable to attacker, where n is the number of traces. $T_1, T_2, \ldots, T_n$ are assumed i.i.d.
- K: Secret key for masked circuit.
- $Z^n = (Z_1, Z_2, \ldots, Z_n)$: Secret intermediate value calculated from K and T using a selection function. If we need to consider Z with a secret key k, we write $Z^{(k)}$. For example, $Z^{(k)} = \mathrm{Sbox}(Z \oplus k)$ in some typical SCAs on AES.
- $M^n = (M_1, M_2, \ldots, M_n)$: Generated mask bits.
- A_i $(0 \leq i \leq d)$: Shares of Z, where d is the order.
- $\boldsymbol{L}_i$ $(0 \leq i \leq d)$: Leakage of A_i. We assume that leakages are mutually independent, as in previous literature.
- $\boldsymbol{X}^n = (\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_n)$: Side-channel trace given as $\boldsymbol{X} = (\boldsymbol{L}_i)_{i=0}^{d}$. Traces/leakages are assumed i.i.d.
- $\hat{K}$: Estimation of secret key by SCA.

See the first paragraph of Appendix B for a detailed description of how this model represents SCA.

5.2. Optimal gray-box SCA

To discuss the capability and limitation of gray-box adversary, we introduce a concept of optimal distinguisher [74], [80]. Let $(T^n, \boldsymbol{X}^n)$ be an attack dataset. A distinguisher $D(T^n, \boldsymbol{X}^n)$ is a function that returns an estimated secret key using the attack dataset. The SCA success rate (SR) using n traces is defined as $\mathrm{SR}_n(D) = \Pr(K = D(T^n, \boldsymbol{X}^n))$ [80], where Pr is the probability measure. Optimal distinguisher maximizes the SR, as defined below.

Definition 5.1 (Optimal distinguisher [74], [80])**.** *A distinguisher D_{opt} is optimal if its SR is the highest among distinguishers, namely,* $\mathrm{SR}_n(D_{\mathrm{opt}}) = \sup_D \mathrm{SR}_n(D)$.

We here introduce a concrete construction of optimal distinguisher that used in deep-learning based SCA (DL-SCA)[5]. In [80], a disgtinguisher using log-likelihood of secret key, defined as

$$D_{\mathrm{opt}}(T^n, \boldsymbol{X}^n) = \operatorname*{arg\,max}_{k \in \mathcal{K}} \sum_j \log p_{Z|\boldsymbol{X}}(Z_j^{(k)} \mid \boldsymbol{X}_j), \quad (1)$$

is proven to be optimal, where $Z_j^{(k)}$ and $\boldsymbol{X}_j$ $(1 \leq j \leq n)$ are the j-th secret intermediate value calculated from k and T_j and its side-channel trace, respectively. This can be proven in our graphical model in Figure 3. To utilize this optimal distinguisher in practice, for example, a DL-SCA trains a neural network (an NN) to imitate $p_{Z|\boldsymbol{X}}$ as $q_{\hat{\theta}}$, and use it to calculate the log-likelihood [91], [98].

The optimal distinguisher in Equation (1) is applicable to masked implementation; however, it does not represent the features and leakages of masking in a specific way. According to [79, Equation (2)], for d-th order masked implementation, $p_{Z|\boldsymbol{X}}$ can be represented using a convolution of conditional probability distributions of share given its leakage; namely, it holds that

$$p_{Z|\boldsymbol{X}}(z \mid \boldsymbol{X}) = \sum_{z = a_0 \oplus a_1 \oplus \cdots \oplus a_d} \prod_{i=0}^{d} p_{A_i|\boldsymbol{L}_i}(a_i \mid \boldsymbol{L}_i),$$

where $p_{A_i|\boldsymbol{L}_i}$ is the conditional probability distribution of j-th share A_j given its leakage $\boldsymbol{L}_i$. Theorem 5.1 represents d *convolution* of probability distributions[6], which is essential to achieve an exponential security of masking [79]. As well, in the graphical model of Figure 3, we have Theorem 5.1 about an optimal distinguisher on masked implementation.

Theorem 5.1 (Optimal SCA on masked implementation [79], [80])**.** *Consider the graphical model in Figure 3.*

5. Note that many constructions of optimal distinguisher have been known in addition to one we introduced [80]. For example, the template attack [32], which utilizes $p_{X|T,K}$ (X denotes a univariate leakage from Z), is also optimal [74]. In addition, CPA is also optimal under an assumption that the leakage is given by $\boldsymbol{X} = \mathrm{HW}(Z) + N$, where N is a Gaussian noise (it is also true for a Hamming distance (HD) leakage) [93].

6. Interestingly, for probability distributions on finite discrete set, a convolution with a uniform distribution results in a uniform distribution. This implies that optimal distinguisher in Theorem 5.1 is not successful in key recovery and has an SR of random guess if at least one leakage of share is not observable (namely, $p_{A_i|\boldsymbol{L}_i} = p_{A_i}$ for some i), which corresponds to upto d-probing attacker on d-th order masking (*i.e.*, a black-box attacker).

A distinguisher defined as

$$D_{\text{opt}}(T^n, \boldsymbol{X}^n) = \underset{k\in\mathcal{K}}{\arg\max} \sum_j \log p_{Z|\boldsymbol{X}}(Z_j^{(k)} \mid \boldsymbol{X}_j)$$
$$= \underset{k\in\mathcal{K}}{\arg\max} \sum_j \log \sum_{Z^{(k)}=\bigoplus_i A_{i,j}} \prod_{i=0}^{d} p_{A_{i,j}|\boldsymbol{L}_i}(A_{i,j} \mid \boldsymbol{L}_{i,j})$$

is optimal, where $A_{i,j}$ and $\boldsymbol{L}_{i,j}$ denote the i-th share of j-th execution and its leakage, respectively.

Proof. See Appendix B. □

As same as Theorem 4.1, this optimal distinguisher is invariant of the trace order due to commutativity of addition. Therefore, as far as the mask-bits generator provides a uniform distribution, the security of masked circuit (*i.e.*, key-recovery success rate for a dataset, or the number of traces to achieve a success rate) does not depends on its construction. Thus, to evaluate a concrete security of d-th order masked implementation against best possible gray-box SCA (*i.e.*, $(d+1)$-th order SCA), even a non-cryptographic PRG including LFSRs would be sufficient if its outputs are high-dimensionally equidistributed uniformly.

Conversely, from gray-box adversarial perspective, a best way to perform a key-recovery SCA is to use the optimal distinguisher described in this paper, regardless of the construction of mask-bits generator and value of M (Even if it has a flaw such as a correlation in low dimension(s), the optimal distinguisher maximizes the key-recovery performance). In fact, Figure 3 states that $\hat{K}$ and M are conditionally independent as observable random variables T, $\boldsymbol{X}$, and $\boldsymbol{L}_0, \boldsymbol{L}_1, \ldots, \boldsymbol{L}_d$ d-separate them.

5.3. Relation to another attacks

Biased mask attack. Although other types of higher-order SCAs that utilize leakage of mask bits in a different manner have been known, it cannot achieve higher performance than the distinguisher using a convolution in Theorem 5.1. For example, a biased mask attack [94] selects traces with highly-based mask bits, which is estimated by $\boldsymbol{L}_j$ and discards other traces. Estimation of highly-based mask bits corresponds to usage of $p_{A_i|\boldsymbol{L}_i}$, because this attack aims at estimating the value of A_i from $\boldsymbol{L}$ in a specific way. Meanwhile, discarding traces does not exploit the information included in them, which results in success rate degradation. Thus, even if the mask bits are highly biased, the convolution-based distinguisher achieves higher performance than the biased mask attack.

Mask guessing attack. If secret key and seed of PRG are exposed to the attacker, then M in Figure 3 gets observable and the attacker can guess the shares $a_1, a_2, \ldots, a_d$ with probability 1 as $p_{A_i|M}$, and can estimate z as

$$p_{Z|\boldsymbol{X},M}(z \mid \boldsymbol{X}, M) = \sum_{z=\bigoplus_i a_i} \prod_{i=0}^{d} p_{A_i|\boldsymbol{L}_i}(a_i \mid \boldsymbol{L}_i, M)$$
$$= p_{A_0|\boldsymbol{L}_0,M}(a_1 \oplus a_2 \oplus \cdots \oplus a_d \mid \boldsymbol{L}_0, M),$$

which implies that the log-likelihood computation and key estimation can be carried out without convolution. Thus, the attacker can perform a first-order SCA solely using $\boldsymbol{L}_0$ and its SR gets identical to that on non-masked implementation.

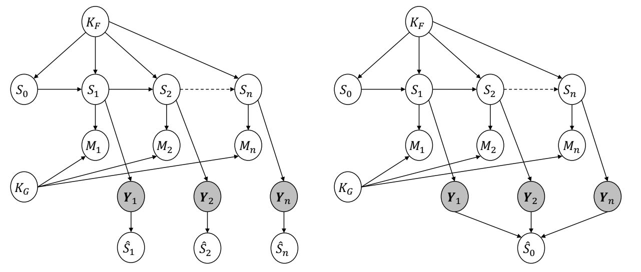
Attack using fresh randomness leakage. The attacker can utilize a leakage of correlated pseudorandomness for mask refreshing. However, the usage of such a leakage does not significantly improve the SCA success rate. For example, let us consider a refreshing of $A'_0, A'_1, \ldots, A'_d$ to $A_0, A_1, \ldots, A_d$ using a correlated pseudorandomness $R_0, R_1, \ldots, R_d$ as $A_i = A'_i \oplus R_i$. Note that both $A'_0, A'_1, \ldots, A'_d$ and $A_0, A_1, \ldots, A_d$ represent an identical unshared value Z. Let $\boldsymbol{L}'_0, \boldsymbol{L}'_1, \ldots, \boldsymbol{L}'_d$ be their respective leakage, included in $\boldsymbol{X}$. As refreshing is equivalent to adding masked zero, it does not change Z (*i.e.*, unshared secret value) and its leakage $p_{R_i|\boldsymbol{L}'_i}$ is not directly used to calculate the log-likelihood. One way to use it would be deriving the probability distribution before refreshing by convoluting $p_{A_i|\boldsymbol{L}_i}$ and $p_{R_i|\boldsymbol{L}'_i}$ as

$$p_{A'_i|\boldsymbol{X}}(A'_i \mid \boldsymbol{X}) = \sum_{A'_i=A_i\oplus R_i} p_{A_i|\boldsymbol{L}_i}(A_i \mid \boldsymbol{L}_i) p_{R_i|\boldsymbol{L}'_i}(R_i \mid \boldsymbol{L}'_i),$$

and use them to calculate the log-likelihood. However, it uses a convolution as same as masking represented in Theorem 5.1, which would degrade the success rate of the SCA. Another way to exploit this leakage would be soft-analytical SCA (SASCA), which exploits all leakages in the masked circuits to recover secret key by belief propagation and would be the best to improve the success rate in the current situation; but, it would not significantly improve the SCA performance as $p_{R_i|\boldsymbol{L}'_i}$ has no direct relation to the value of Z and signal-to-noise ratio (SNR) in measuring $\boldsymbol{L}'_i$ would be as bad as other leakages.

5.4. Relation to practical scenario and take away for practitioners

Gray-box model corresponds to quantitative SCA-resistance evaluation in terms of SR, which would be essential for determination of device lifetime [129]. Best gray-box SCA on masked circuits in terms of SR is a higher-order SCA that utilizes a convolution, regardless of the order of traces (*i.e.*, how mask bits are generated). When one want to evaluate the practical security and secret key lifetime of d-th order masked circuit against best possible gray-box SCA (*i.e.*, how many times the device can execute the encryption securely against SCA), even a non-cryptographic PRG including LFSRs would be sufficient.



(a) Type 1 (b) Type 2

Figure 4: Graphical models representing white-box SCA.

6. Analysis of white-box adversary

6.1. Notations and graphical model

If the secret seed and key are exposed to the attacker, then the implementation is no more secure than the first order owing to the mask guessing attack (Section 5.3). In this section, we discuss (in)security of major PRGs against state/key-recovery SCA. Figure 4 shows a graphical model representing an SCA on PRG in Figure 1, where random variables are defined as follows. (Estimated keys $\hat{K}_F$ and $\hat{K}_G$ from $\boldsymbol{Y}^n$ are omitted for simplicity.)

- S_0: Initial state of PRG given from seed/key.
- $S^n = (S_1, \ldots, S_n)$: Internal states, where n is the number of traces. $\hat{S}_j$ is its estimation.
- K_F and K_G: Secret keys for an updating function F and output function G, respectively (which can be null).
- $\boldsymbol{Y}^n = (\boldsymbol{Y}_1, \boldsymbol{Y}_2, \ldots, \boldsymbol{Y}_n)$: Side-channel leakage from mask-bits generator.

6.2. Adaptive attacks on mask-bits generator

In an SCA on masked cryptographic implementation, the attacker is supposed to first acquire n side-channel traces including leakages of both masked circuit and PRG. Because the PRG and masked circuits operate independently, the attacker may first perform an SCA to recover the secret key and/or seed of the PRG, before attacking the masked circuit; then, the attacker can perform a first-order SCA with mask guessing if the attacker has successfully recovery them. In other words, a d-probing attacker can first use all probes on a PRG, then adaptively move the probes to use them on the masked circuit. This is trivially extended to the multiple PRG strategy [40], meaning that the attacker can adaptively use the probes on each PRG distinctly, as the PRGs operate distinctly while the trace includes leakage of all PRGs. Although such an adaptive attack has been considered impractical for SCAs on masked circuit [64], [125], the attacker can adaptively attack PRG(s) to recover its state and key in practice if focusing on mask-bits generator.

The feasibility of adaptive leakage from PRGs would indicate that, *independently of whole masked implementation, we should evaluate the SCA security of mask-bits generator against practical SCAs, apart from probing-related models.* If we use probing-related model to evaluate the SCA security of PRGs, it is obvious that each PRG should be (composable) secure against d-probing for whole implementation to achieve the d-probing security. However, it may achieve higher security than reqruied because only blind SCA is available, and it incurs non-neglible perfornace overhead. In addition, to use this notion, we may require masked PRGs, which might be a kind of circular argumentation as they also may require pseudorandomness). In the following, we discuss the leakage models and security of major PRGs against practical SCAs in the context of mask-bit generation.

6.3. Leakage model of mask-bits generation

Let $\boldsymbol{Y}_j$ and S_j be the j-th trace and corresponding state, respectively. The attacker uses a leakage from PRG to estimate the state S, modeled as a conditional probability distribution $p_{S|\boldsymbol{Y}}$, where $\boldsymbol{Y}$ is the leakage of M from PRG. The attacker can obtain λ-bit information from a trace, which is represented as mutual information $I(S; \boldsymbol{Y})$. This indicates that, if the attacker observes n traces, at most $n\lambda$ bits leaks as $I(S^n; \boldsymbol{Y}^n) \leq I(S; \boldsymbol{Y})$. This leakage model would be similar to that in many previous works on LR cryptography [18], [52], [54], [55] and theoretical analyses of SR of SCA [12], [45], [58], [79], [100]. Depending on updating function F, we classify PRGs into two types.

Type 1: F is a cryptographic primitive. Consider a case that F consists in a cryptographic primitive, such as BC and pseudorandom permutation (PRP). As the attacker is supposed to know neither secret key and seed, each state $S_1, S_2, \ldots, S_n$ and their leakage $\boldsymbol{Y}_1, \boldsymbol{Y}_2, \ldots, \boldsymbol{Y}_n$ are mutually independent due to the indistinguishability. Thus, we suppose that, for an SCA using n traces, each sequence of $S_1, S_2, \ldots, S_n$ and $\boldsymbol{Y}_1, \boldsymbol{Y}_2, \ldots, \boldsymbol{Y}_n$ are i.i.d for the attacker. Therefore, in this case, the attacker cannot perform DPA-like SCAs using $\boldsymbol{Y}_1, \boldsymbol{Y}_2, \ldots, \boldsymbol{Y}_n$, but can only do an SPA-like attack that estimates S_j solely from $\boldsymbol{Y}_j$ (as $\boldsymbol{Y}_{j'}$ is independent of S_j for $j \neq j'$). If the attacker observes n traces, at most $n\lambda$ bits leaks as $I(S^n; \boldsymbol{Y}^n) = nI(S; \boldsymbol{Y})$, but S^n is consider to have $n \times n_f$-bit information. Although most symmetric primitive are not proven secure in the presence of leakage (except for LR-PRG/stream cipher [60], [110]), major such primitives would be sufficiently secure in practical settings in the current situation, as in Section 6.4.

Type 2: F is not a cryptographic primitive. A major example of this type is a counter-based XOF with $F(S_j) = S_j + 1$. For this type of PRGs, $S_1, S_2, \ldots, S_n$ and $\boldsymbol{Y}_1, \boldsymbol{Y}_2, \ldots, \boldsymbol{Y}_n$ are not independent even for the attacker. The attacker may perform a DPA-like key/state-recovery SCA using n traces. Due to the dependence of S_j's, the attacker may succeed in key recovery if he/she obtains n_f-bit information from leakages as $nI(S; \boldsymbol{Y}) \leq I(S^n; \boldsymbol{Y}^n)$ Thus, if F is not cryptography primitive, the attacker would estimate a state S_j from all leakages of $\boldsymbol{Y}^n$. In fact, for this type, practical SCAs on some PRGs are known, while others would have practical security, as described below.

6.4. SCA (in)vulnerability of major primitives

6.4.1. Linear feedback shift registers (LFSRs). We first discuss plain LFSR, which is type 2. It is known that the LFSR seed (or internal state) can be recovered if the attacker observes $2n_f$-bits output of an n_f-bit LFSR. An SCA attacker obtains $I(S;\boldsymbol{Y})$ bits per trace, at most $n\lambda$-bits information about the state are obtained from n traces. Note that the LFSR output is given as $\mathrm{G}(S)$, the leakage of which is included in $\boldsymbol{Y}$. Thus, if the attacker observes n traces such that $n\lambda \geq 2n_f$, then the attacker could recover the secret seed or internal state; LFSR would be difficult for LFSR to achieve a provable security, and state-recovery SCAs on LFSR were reported in [27], [29], [82], [92], [124].

In contrast, improved LFSRs (*e.g.*, Mersenne twister and Xorshift) are designed to be implemented with a high parallelism, which consists of large internal state of their feedback function to realize a long frequency. State recovery of such LFSRs would be significantly more difficult than that of plain LFSR, because it requires to observe the output bit sequence far longer. As the improved LFSRs are implemented with a high parallelism, the side-channel leakage would contain less information than non-parallelized implementation due to algorithmic noise, and the measurement with a high signal-to-noise ratio (SNR) would be difficult. Thus, improved LFSRs would be more secure than plain LFSRs, while state recovery SCA would be possible in principle as well as plain LFSRs.

Remark 6.1 (LFSR cannot be protected by boolean masking against adaptive probing). Because LFSR consists of solely linear operation, d-th order Boolean-masked LFSR is realized by just putting d LFSRs. However, in this setting, side-channel trace $\boldsymbol{X}$ contains the leakage of each LFSR. If the attacker has n traces, the attacker first tries to recover the first LFSR state using the n traces. Then, attacker can also recover the states of second, third, $\dots$, and d-th LFSR from the same n traces, as in [29, Section 4.3]. Thus, an adaptive-probing attacker can recover d-probing secure LFSR state with the complexity linear to a first-order SCA, indicating that LFSR cannot be protected by Boolean masking.

6.4.2. Stream cipher. Stream cipher is a major symmetric primitive to generate pseudorandomness, and very practical in terms of performance and implementation cost. For example, Trivium is one of major stream cipher [43] included in the ISO/IEC 29192-3 standard [75] and the portfolio of eSTREAM project [44], and has been sometimes utilized in masked implementations [29]. However, stream cipher would be type 2 as its round function is far from PRP, and the leakage resilience of standard stream cipher is not proven generally. An SCA attacker may recover the internal state if the attacker observes n side-channel traces such that $n\lambda \geq n_f$. However, stream ciphers are designed such that internal state cannot be recovered from output key stream; the state recovery would be more difficult than that of LFSR.

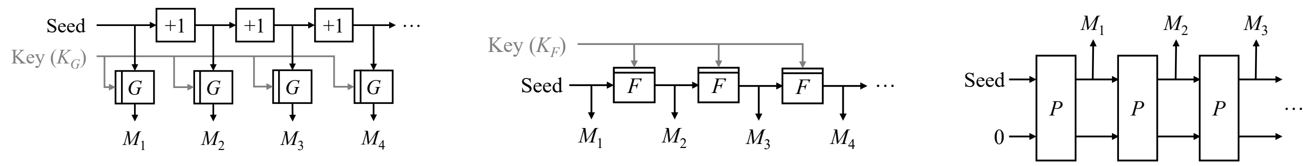
Kumar *et al.* reported a state recovery attack on Trivium in [89]. However, its feasibility is limited to a specific implementation and very good SNR. Namely, the SCA is applicable to only a round-based implementation, while unrolled implementation is very useful to achieve low energy stream cipher for both software[7] and hardware [7], [28]. If a sufficient number of rounds are unrolled, then F highly diffuses the state and the side-channel measurement with high SNR gets difficult. Thus, although (non-protected and non-LR) stream ciphers have difficulty in achieving a provable security, an unrolled implementation of stream ciphers like Trivium would offer both a sufficient performance and practical security, as recommended by Cassiers *et al.* in [29].

Meanwhile, some LR-PRGs have been developed, which are proven to be secure in indistinguishability notion under (some level of) leakage [60], [110]. However, the construction of LR-PRGs is much more complicated than non-LR stream ciphers and incurs non-negligibly high implementation cost. Security of LR-PRGs is proven even if attacker can observe its output directly. It may achieve a stronger security goal for mask bits generation than required.

6.4.3. Extendable output functions (XOFs). A common way to construct a PRG as an XOF is using an upcounter for F and using one-way hash H or BC E_{K_G} for outputting function G; that is, $F(S_i) = S_i + 1$ and $G(S_i) = H(S_i)$ or $E_K(S_i)$, respectively. Such a counter-based XOF is type 1. Figure 5a illustrates its overview, where $G = H$ or $G = E_{K_G}$. This construction is the basis of Hash_DRBG and CTR_DRBG (DRBG stands for Deterministic Random Bit Generator), which are PRGs standardized in NIST SP 800-90A [8]. SCAs on BC with counter have been extensively studied so far, and this XOF does not have leakage resilience. In [49], de Meyer reported a practical (blind) SCA on CTR_DRBG with AES, which achieved the key and state recoveries within 256 traces under practical conditions based on the SCA strategy on the counter-mode AES by Jaffe in [81], and it was improved by Ebina *et al.* using DL-SCA [62]. Although these attacks were demonstrated on only AES-based modes in the literature, they can be applied to counter-based modes with any symmetric primitive (including hash and permutation) in principle. Thus, counter-based XOF would not be suitable to mask-bits generation.

Another major way to construct a type-2 XOF from H or E_K is the output-feedback mode (OFB), as illustrated in Figure 5b, where $F = H$ or $F = E_{K_F}$. A NIST-standardized DRBG, HMAC_DRBG, is an instance of which uses HMAC as $F(K_F, S_i) = \mathrm{HMAC}_{K_F}(S_i)$. There is no known SCA except for SPA on it if the output is (directly) unavailable for the attacker, as states $S_1, S_2, \dots, S_n$ are independent of each other, which would imply that it is difficult to exploit their leakage jointly. In [83], Kannwischer *et al.* presented and evaluated an input-recovery SASCA [132] on Keccak, which corresponds to state recovery for hash-based OFB and HMAC_DRBG; however, its successful attack was limited to small microcontroller, long available input bits (nothing in our context), and/or good SNR. Thus, OFB-based XOF and HMAC_DRBG may be practically secure

7. On a b-bit microcontroller, b bits are usually processed in parallel, which corresponds to a b-round unrolling.



(a) CTR/Hash_DRBG (b) HMAC_DRBG/OFB (c) Sponge

Figure 5: Extendable output functions (XOFs) based on cryptographic primitive.

in the use of mask-bits generation, rather than Hash_DRBG and CTR_DRBG. Note that a key-recovery blind SCA [35], [90] would be applicable to E_{K_F} in OFB, but state recovery is also required for mask guessing attack. As G can be a public permutation/hash for secret seed, a key-recovery blind SCA would be insufficient unless state recovery.

Yet another major XOF construction would be sponge (or duplex) using a PRP. Let P be an n_f-bit PRP, such as Keccak (in SHA-3) [20] and Ascon [53]. Figure 5c shows a sponge-based XOF, which is also type 1. The update and output functions are defined as $F(S_i) = P(S_i)$ and $G(S_i) = \tau_r(S_i)$, respectively, where r is the rate and τ_r is a truncation into r bits. A kind of inherent leakage resilience of sponge-based cryptography has been discussed in [18], [52], [54], [55]. For example, ISAP and Ascon are claimed to be secure against SCAs if PRP is SPA resistant. The above SASCA on Keccak is also applicable to this, while its feasibility was limited. Thus, sponge-based XOF might offer a guarantee of high SCA security, as well as OFB-based ones.

Remark 6.2 (Reduced-round BC). A reduced-round BC may be used for XOFs in masking, as we do not require a cryptographic PRG, which would significantly improve the performance. In particular, an XOF based on 4-round AES has been sometimes used if a non-cryptographic PRG is sufficient [84], for example, in UOV signature [21].

6.4.4. PRF.

One may use a PRF R_K to construct an XOF similar to BC or HMAC, as in Figures 5a and 5b. The classical GGM would be the most popular to implement PRF [67], which was followed by two LR-PRF, FPS and MSJ [64], [104] (and the leakage resilience of GGM was also discussed in [104]). However, these PRFs require a non-negligible latency of cryptographic PRG calls of key length per one evaluation of input. Thus, it would be difficult to achieve the practical performance as a mask-bits generator using these (LR-)PRFs, in spite of leakage resilience.

Alternatively, other major PRF construction would be sum of BC or PRP [34], [73], [123] (*e.g.*, $F(S) = E_{K_1}(S) \oplus E_{K_2}(S)$) and trancation of BC or PRP [16]. However, SCA resistance of PRF-based XOF is very similar to BC-based XOFs in Section 6.4.3, while these PRF employs BC or PRP inside. In addition, these PRFs are considered as non-invertible construction from BC or PRP. As PRG output does not require cryptographic properties as in Section 5.2, the non-invertibility would make little sense in the masking context (it may imply forward-secrecy when state is recovered, but its advantage is unclear). Thus, PRF-based XOF would have a (dis)advantage similar to BC-based ones.

6.5. Relation to practical scenario and take away for practitioners

White-box model corresponds to SCAs targeting PRG. This model actually represents a practical attacker who can put probes in any parts of target device including PRG, while black- and gray-box models correspond to leakage assessment and security evaluation by the designer/evaluator. The whole security of masked cryptographic implementation depends on the more vulnerable part of either masked circuit or PRG. We should consider the practical SCA security of PRG apart from probing-related models. This implies that we should evaluate both gray- and white-box SCA for a practical and quantitative security of masked implementation. Table 1 summarizes the (in)vulnerability of major primitives at present.

7. Our recommendations for practitioners

7.1. Black- and gray-box scenarios

Recall that black- and gray-box models correspond to leakage assessment and SCA resistance evaluation of masked circuit by designer/implementer, respectively, apart from the SCA security of PRG. Any PRG including non-cryptographic ones can yield valid assessment/evaluation, as far as it provides a uniform distribution. Hence, it would be sufficient to determine PRG with regard to its availability and implementation costs for assessment/evaluation. LFSRs including Mersenne twister and Xorshift and stream cipher would be useful, if mask bits are guaranteed to be uniformly equidistributed with regard to a flaw mentioned in [29].

7.2. White-box scenario

In this scenario, differently from Section 7.1, we have to consider the SCA resistance of PRG because practical attackers are likely to target the mask-bits generator to mount a first-order SCA by guessing mask values from recovered PRG state/key. We here list some recommendations with respect to practical security and performance.

Improved LFSRs. Some LFSRs like Mersenne twister and Xorshift would have practical security against state-recovery SCAs. Its practical implementation exploits a high parallelism and has a large state. This makes it difficult to obtain information of internal state from leakages with high precision and to reveal whole state, while it has a very high throughput. Still, it would not achieve a provable security, which implies possibility that attacker achieves the state recovery if a large number of leakage is observed. To avoid this in practice, reseeding at a frequency makes the state recovery very difficult (See Section 7.3).

Stream ciphers. Stream cipher is a promising primitive to generate uniform distribution with a high throughput. Stream ciphers can be implemented by exploiting its parallelism to achieve a high energy efficiency [7], [28]. A b-round unrolled hardware implementation outputs b pseudo-ranbom bits per clock cycle. Banik *et al.* demonstrated that a 288-round unrolling achieves highest energy efficiency [7]. Thus, stream ciphers would be very practical for mask-bits generation, as recommended by Cassiers *et al.* [29]. As well, a b-bit microcontroller can employ a b-round unrolled implementation using b-bit registers (up to 64 rounds),

OFB-based and sponge-based XOFs. Some of these XOFs (*e.g.*, Ascon) are claimed to achieve an SCA security if the underlying primitive is SPA resistant. Although they would achieve a higher level of SCA security guarantee than improved LFSR and stream cipher, the performance (latency and throughput) might be worse than them (these XOFs require BC/PRP evaluation to generate random bits).

Masked stream ciphers/XOFs. A first-order masking (*e.g.*, TI) of some symmetric primitives can be implemented with practical performance without fresh randomness (*e.g.*, Ascon [112], Trivium [120], SHA3 [42], and AES [127]). Using them, we can realize a first-order secure mask-bits generator, because initial decomposition is not needed in the masking context once it is activated using a seed.

7.3. Proposed LR mask-bits generation method for white-box scenario

From performance perspective, (improved) LFSR would be best (or stream cipher would be acceptable), while OFB- and sponge-based XOFs would achieve a high level of SCA security guarantee. We present a leveled implementation of LR mask-bits generation by combining them to exploit their respective advantages. Namely, we use an LR cryptographic primitive as a reseeding function, such as Ascon, asakey, and ISAP,[8] and use non-cryptographic high-performance PRGs to expand the pseudorandomness (*i.e.*, seeds) from LR primitive, with a sufficiently high reseeding frequency, similarly to rekeying [103], [129]. The LR primitive may not have high throughput while non-cryptographic PRG can have. The combination implies a leveled implementation, which is especially efficient in hardware by exploiting parallelism.

Figure 6 displays the overview of proposed method. As a seed expander, we use a high-throughput PRG (*e.g.*, improved LFSR) with a fixed output length, $\alpha : \{0,1\}^{n_f} \to \{0,1\}^{un_g}$ for some $un_g > n_f$, where n_f and n_g are bit lengths of LFSR state and M, respectively, and u is a reseeding interval unit. As a reseeding function, we use a cryptographic PRG $\beta : \mathcal{K} \to \mathcal{K} \times \{0,1\}^s$, where $\mathcal{K}$ is a key space. Here, β is used to reset (*i.e.*, reseed) α's state as well as to perform rekeying of β. Let $\beta(K) = (\gamma_1(K), \gamma_2(K))$ and $K^i := \gamma_1^i(K)$ (γ_1^i denotes composition of i γ_1's). Starting with initial uniformly random PRG key of β, denoted by K^0, at each i-th step we apply β to $\gamma_1(K^{i-1})$ and α to $\gamma_2(K^{i-1})$ and mask bits (values) $(M_{iu} \parallel \cdots \parallel M_{(i+1)u}) = \alpha(\gamma_2(K^i))$. This produces the total output $\alpha(\gamma_2(K^0)) \parallel \alpha(\gamma_2(K^1)) \parallel \cdots \parallel \alpha(\gamma_2(K^{i-1}))$ after the i-th reseeding step. The reseeding interval unit u (*i.e.*, inverse of reseeding frequency) will be determined by considering security-efficiency trade-off for the given improved LFSR and its implementation security. Namely, given state size of the LFSR n_f and leakage amplitude $\lambda = I(S; \boldsymbol{Y})$, we set u such that $n_f \leq u\lambda$ as reseeding interval/frequency, where attacker cannot obtain n_f bits to recover the state for a seed.

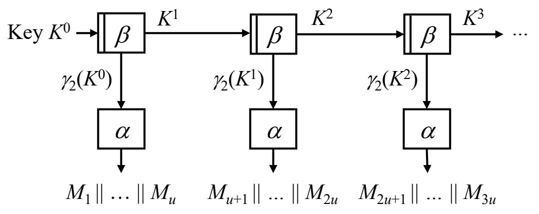

Key K^0
β
K^1
K^2
K^3
...
$\gamma_2(K^0)$
$\gamma_2(K^1)$
$\gamma_2(K^2)$
α
$M_1 \parallel \ldots \parallel M_u$
$M_{u+1} \parallel \ldots \parallel M_{2u}$
$M_{2u+1} \parallel \ldots \parallel M_{3u}$

Figure 6: Proposed LR mask-bits generator, where α is seed expander (*e.g.*, LFSR and stream cipher) and β is reseeding function by LR cryptographic primitive like Ascon.

This scheme has a similar structure to a forward-secure PRG by Bellare and Yee [17], while if we use LFSR to boost the throughput, the output is clearly not cryptographically pseudorandom. Security analysis of such a hybrid scheme as a random generator for masking could be done within the framework of provable security of leveled implementation [18] combined with the discussion at Sections 4.1 and 6.4.1. We leave this as a future work.

8. Conclusion

This paper provided a comprehensive analysis of PRGs for masked cryptographic implementation. Our analysis was conducted in three different models, which reveal that non-cryptographic PRG can be used for mask-bits generation. Although the SCA security of PRG is not required for leakage assessment and SCA-resistance evaluation of masked circuit, we should consider it for practical deployment. Thus, we also conducted (in)vulnerability analysis of major PRGs, and show our recommendations and proposal for practically secure mask bits generation based on our observations.

Acknowledgments

We thank Akira Ito for fruitful discussions and advices.

8. If we believe that their theory is true, the output is cryptographically secure and the module needs only SPA resistance. Its leakage resilience is not formally proven, although there are some discussions [18], [19], [55].

References

[1] A. Aikata, A. Basso, G. Cassiers, A. C. Mert, and S. Sinha Roy, “Kavach: Lightweight masking techniques for polynomial arithmetic in lattice-based cryptography,” *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, vol. 2023, no. 3, pp. 366–390, 2023.

[2] T. Araki, J. Furukawa, Y. Lindell, A. Nof, and K. Ohara, “High-throughput semi-honest secure three-party computation with an honest majority,” in *ACM CCS*, 2016, pp. 805–817.

[3] M. Azouaoui, O. Bronchain, G. Cassiers, C. Hoffmann, Y. Kuzovkova, J. Renes, T. Schneider, M. Schönauer, F.-X. Standaert, and C. van Vredendaal, “Protecting Dilithium against leakage: Revisited sensitivity analysis and improved implementations,” *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, no. 4, pp. 58–79, 2023.

[4] F. Bache, C. Paglialong, T. Oder, T. Schneider, and T. Güneysu, “High-speed masking for polynomial comparison in lattice-based KEMs,” *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, vol. 2020, no. 3, pp. 483–507, 2020.

[5] J. Balasch, S. Faust, B. Gierlichs, and I. Verbauwhede, “Theory and practice of a leakage resilient masking scheme,” in *ASIACRYPT*, 2012, pp. 758–775.

[6] J. Balasch, B. Gierlichs, V. Grosso, O. Reparaz, and F.-X. Standaert, “On the cost of lazy engineering for masked software implementations,” in *CARDIS*, 2014, pp. 64–81.

[7] S. Banik, V. Mikhalev, F. Armknecht, T. Isobe, W. Meier, A. Bogdanov, Y. Watanabe, and F. Regazzoni, “Towards low energy stream ciphers,” *IACR Trans. Symmetric Cryptol.*, no. 2, pp. 1–19, 2018.

[8] E. Barker and J. Kelsey, “NIST special publication 800-90a: Recommendation for random number generation using deterministic random bit generators,” Jan. 2012, computer Security Division Information Technology Laboratory, National Institute of Standards and Technology. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-90a.pdf

[9] G. Barthe, S. Balaïd, G. Cassiers, P.-A. Fouque, B. Grégoire, and F.-X. Standaert, “maskVerif: Automated verification of higher-order masking in presence of physical defaults,” in *ESORICS*, 2019, pp. 300–318.

[10] G. Barthe, S. Belaïd, F. Dupressoir, P.-A. Fouque, B. Grégoire, P.-Y. Strub, and R. Zucchini, “Strong non-interference and type-directed higher-order masking,” in *ACM CCS*, 2016.

[11] G. Barthe, S. Belaïd, F. Dupressoir, P.-A. Fouque, B. Grégoire, and P.-Y. Strub, “Verified proofs of higher-order masking,” in *EUROCRYPT*, 2015, pp. 457–485.

[12] J. Béguinot, W. Cheng, S. Guilley, Y. Liu, L. Masure, O. Rioul, and F.-X. Standeart, “Removing the field size loss from Duc et al.’s conjectured bound for masked encodings,” in *COSADE*, 2023, pp. 86–104.

[13] J. Béguinot, W. Cheng, S. Guilley, and O. Rioul, “Formal security proofs via Doeblin coefficients: Optimal side-channel factorization from noisy leakage to random probing,” in *CRYPTO*, 2024.

[14] S. Belaïd, J.-S. Coron, E. Prouff, M. Rivain, and A. R. Taleb, “Random probing security: Verification, composition, expansion and new constructions,” in *CRYPTO*, Cham, 2020.

[15] S. Belaïd, D. Mercadier, M. Rivain, and A. R. Taleb, “IronMask: Versatile verification of masking security,” in *IEEE S&P*, 2022, pp. 142–160.

[16] M. Bellare, T. Krovetz, and P. Rogaway, “Luby-Rackoff backwards: Increasing security by making block ciphers non-invertible,” in *EUROCRYPT*, 1998, pp. 266–280.

[17] M. Bellare and B. S. Yee, “Forward-Security in Private-Key Cryptography,” in *CT-RSA*, ser. Lecture Notes in Computer Science, vol. 2612. Springer, 2003, pp. 1–18.

[18] D. Bellizia, O. Bronchain, G. Cassiers, V. Grosso, C. Guo, C. Momin, O. Pereira, T. Peters, and F.-X. Standaert, “Mode-level vs. implementation-level physical security in symmetric cryptography: A practical guide through the leakage-resistance jungle,” in *CRYPTO*, 2020, pp. 369–400.

[19] H. Berendsen and B. Mennink, “Tightening leakage resilience of the suffix keyed sponge,” *IACR Trans. Symmetric Cryptol.*, vol. 2024, no. 1, pp. 459–496, 2024.

[20] G. Bertoni, J. Daemen, S. Hoffert, M. Peeters, G. Van Assche, and R. Van Keer, “Team Keccak,” Nov. 2023. [Online]. Available: https://keccak.team

[21] W. Beullens *et al.*, “UOV: Unbalanced oil and vinegar,” NIST Post-Quantum Cryptography, Aug 2024. [Online]. Available: https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-1/spec-files/UOV-spec-web.pdf

[22] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[23] J. W. Bos, M. Gourjon, J. Renes, T. Schneider, and C. van Vredendaal, “Masking Kyber: First- and higher-order implementations,” *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, vol. 2021, no. 4, pp. 173–214, 2021.

[24] G. Brian, S. Dziembowski, and S. Faust, “From random probing to noisy leakages without field-size dependence,” in *EUROCRYPT*, 2024, pp. 345–374.

[25] E. Brier, C. Clavier, and F. Olivier, “Correlation power analysis with a leakage model,” in *CHES*, 2004, pp. 16–29.

[26] O. Bronchain and G. Cassiers, “Bitslicing arithmetic/Boolean masking conversions for fun and profit: with application to lattice-based KEMs,” *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, vol. 2022, no. 4, pp. 553–588, 2022.

[27] S. Burman, D. Mukhopadhyay, and K. Veezhinathan, “LFSR based stream ciphers are vulnerable to power attacks,” in *INDOCRYPT*, 2007, pp. 384–392.

[28] A. Caforio, S. Banik, Y. Todo, W. Meier, T. Isobe, F. Liu, and B. Zhang, “Perfect trees: Designing energy-optimal symmetric encryption primitives,” *IACR Trans. Symmetric Cryptol.*, no. 4, pp. 36–73, 2021.

[29] G. Cassiers, L. Masure, C. Momin, T. Moos, A. Moradi, and F.-X. Standaert, “Randomness generation for secure hardware masking – unrolled Trivium to the rescue,” *IACR Communications in Cryptology*, vol. 1, no. 2, 2024.

[30] G. Cassiers and F.-X. Standaert, “Trivially and efficiently composing masked gadgets with probe isolating non-interference,” *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 2542–2555, 2020.

[31] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi, “Towards sound approaches to counteract power-analysis attacks,” in *CRYPTO*, 1999, pp. 398–412.

[32] S. Chari, J. R. Rao, and P. Rohatgi, “Template attacks,” in *CHES*, 2002, pp. 13–28.

[33] C. Chen, T. Eisenbarth, I. von Maurich, and R. Steinwandt, “Masking large keys in hardware: A masked implementation of McEliece,” in *SAC*, 2016, pp. 293–309.

[34] Y. L. Chen, E. Lambooij, and B. Mennink, “How to build pseudorandom functions from public random permutations,” in *CRYPTO*, 2019, pp. 266–293.

[35] C. Clavier and L. Reynaud, “Improved blind side-channel analysis by exploitation of joint distributions of leakages,” in *CHES*, 2017, pp. 24–44.

[36] J.-S. Coron, F. Gérard, S. Montoya, and R. Zeitoun, “High-order polynomial comparison and masking lattice-based encryption,” *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, vol. 2023, no. 1, p. 153–192, 2022.

[37] ——, "High-order table-based conversion algorithms and masking lattice-based encryption," *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, vol. 2022, no. 2, pp. 1–40, 2022.

[38] J.-S. Coron, F. Gérard, M. Trannoy, and R. Zeitoun, "High-order masking of NTRU," *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, vol. 2023, no. 2, pp. 180–211, 2023.

[39] ——, "Improved gadgets for the high-order masking of Dilithium," *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, vol. 2023, no. 4, pp. 110–145, 2023.

[40] J.-S. Coron, A. Greuet, and R. Zeitoun, "Side-channel masking with pseudo-random generator," in *EUROCRYPT*, 2020, pp. 342–375.

[41] J.-S. Coron, E. Prouff, M. Rivain, and T. Roche, "Higher-order side channel security and mask refreshing," in *FSE*, 2014, pp. 410–424.

[42] J. Daemen, "Changing of the guards: A simple and efficient method for achieving uniformity in threshold sharing," in *CHES*, 2017, pp. 137–153.

[43] C. De Canniere, "Trivium specifications," 2006, http://www.ecrypt.eu.org/stream/p3ciphers/trivium/trivium_p3.pdf.

[44] C. De Canniere and B. Preneel, "Trivium," in *New Stream Cipher Designs: The eSTREAM Finalists*. Springer, 2008, pp. 244–266.

[45] E. de Chérisey, S. Guilley, O. Rioul, and P. Piantanida, "Best information is most successful: Mutual information and success rate in side-channel analysis," *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, vol. 2019, no. 2, pp. 49–79, 2019.

[46] T. de Cnudde, B. Bilgin, B. Gierlichs, V. Nikov, S. Nikova, and V. Rijmen, "Does coupling affect the security of masked implementations?" in *COSADE*, 2017, pp. 1–18.

[47] T. De Cnudde, M. Ender, and A. Moradi, "Hardware masking, revisited," *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, no. 2, pp. 123–148, 2018.

[48] T. De Cnudde, O. Reparaz, B. Bilgin, S. Nikova, V. Nikov, and V. Rijmen, "Masking AES with $d+1$ shares in hardware," in *CHES*, 2016, pp. 194–212.

[49] L. De Meyer, "Recovering the CTR_DRBG state in 256 traces," *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, vol. 2020, no. 1, pp. 37–65, 2019.

[50] L. De Meyer, O. Reparaz, and B. Bilgin, "Multiplicative masking for AES in hardware," *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, vol. 2018, no. 3, pp. 431–468, 2018.

[51] R. del Pino, T. Prest, M. Rossi, and M.-J. O. Saarinen, "High-order masking of lattice signatures in quasilinear time," in *IEEE S&P*, 2023, pp. 1168–1185.

[52] C. Dobraunig, M. Eichlseder, S. Mangard, F. Mendel, B. Mennik, R. Primas, and T. Unterlaggauer, "Isap v2.0," *IACR Trans. Symmetric Cryptol.*, vol. 2020, no. S1, pp. 390–416, 2020.

[53] C. Dobraunig, M. Eichlseder, F. Mendel, and M. Schläffer, "ASCON: Lightweight authentication & hashing," 2021, https://ascon.iaik.tugraz.at/index.html.

[54] C. Dobraunig and B. Mennink, "Leakage resilience of the duplex construction," in *ASIACRYPT*, 2019, pp. 225–255.

[55] C. Dobraunig, B. Mennink, and R. Primas, "Leakage and tamper resilient permutation-based cryptography," in *ACM CCS*, 2022, pp. 859–873.

[56] Y. Dodis and K. Pietzrak, "Leakage-resilient pseudorandom functions and side-channel attacks on Feistel networks," in *CRYPTO*, 2010, pp. 21–40.

[57] A. Duc, S. Dziembowski, and S. Faust, "Unifying leakage models: From probing attacks to noisy leakage," in *EUROCRYPT*, 2014, pp. 423–440.

[58] A. Duc, S. Faust, and F.-X. Standaert, "Making masking security proofs concrete: Or how to evaluate the security of any leakage device," in *EUROCRYPT*, 2015, pp. 401–429.

[59] S. Dziembowski and S. Faust, "Leakage-resilient circuits without computational assumptions," in *TCC*, 2012, pp. 230–247.

[60] S. Dziembowski and K. Pietzrak, "Leakage-resilient cryptography," in *FOCS*, 2008, pp. 293–302.

[61] J.-P. D'Anvers, D. Heinz, P. Pessl, M. Van Beirendonck, and I. Verbauwhede, "Higher-order masked ciphertext comparison for lattice-based cryptography," *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, vol. 2022, no. 2, pp. 115–139, 2022.

[62] K. Ebina, R. Ueno, and N. Homma, "Side-channel analysis against SecOC-compliant AES-CMAC," *IEEE Trans. Circ. Syst. II: Express Briefs*, vol. 70, no. 10, pp. 3772–3776, 2023.

[63] S. Faust, V. Grosso, S. Merino Del Pozo, C. Paglialonga, and F.-X. Standaert, "Composable masking schemes in the presence of physical defaults & the robust probing model," *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, vol. 2018, no. 3, pp. 89–120, 2018.

[64] S. Faust, K. Pietrzak, and J. Schipper, "Practical leakage-resilient symmetric cryptography," in *CHES*, 2012, pp. 213–232.

[65] S. Gao, B. Marshall, D. Page, and E. Oswald, "Share-slicing: Friend or foe?" *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, vol. 2020, no. 1, pp. 152–174, 2020.

[66] J. Gaspoz and S. Dhooghe, "Threshold implementations in software: Micro-architectural leakages in algorithms," *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, vol. 2023, no. 2, pp. 155–179, 2023.

[67] O. Goldreich, S. Goldwasser, and S. Micali, "How to construct random functions," *J. ACM*, vol. 33, no. 4, pp. 792–807, 1986.

[68] J. D. Golić and C. Tymen, "Multiplicative masking and power analysis of AES," in *CHES*, 2003, pp. 198–212.

[69] L. Goubin, "A sound method for switching between Boolean and arithmetic masking," in *CHES*, 2001, pp. 3–15.

[70] L. Goubin and A. Martinelli, "Protecting AES with Shamir's secret sharing scheme," in *CHES*, 2011, pp. 79–94.

[71] H. Gross and S. Mangard, "Reconciling $d+1$ masking in hardware and software," in *CHES*, 2017.

[72] H. Gross, S. Mangard, and T. Korak, "Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order," in *TIS, ACM CCS Workshops*, 2016, p. 3.

[73] C. Hall, D. Wagner, J. Kelsey, and B. Schneier, "Building PRFs from PRPs," in *CRYPTO*, 1998, pp. 370–389.

[74] A. Heuser, O. Rioul, and S. Guilley, "Good is not good enough: Deriving optimal distinguishers from communication theory," in *CHES*, 2014, pp. 55–74.

[75] International Organization for Standardization, "ISO/IEC 29192-3:2012 Information technology—Security techniques—Lightweight cryptography— Part 3: Stream ciphers," https://www.iso.org/standard/56426.html, Nov. 2023.

[76] Y. Ishai, E. Kushilevitz, X. Li, R. Ostrovsky, M. Prabhakaran, A. Sahai, and D. Zuckerman, "Robust pseudorandom generators," in *ICALP*, 2013, pp. 576–588.

[77] Y. Ishai, A. Sahai, and D. Wagner, "Private circuits: Securing hardware against probing attacks," in *CRYPTO*, 2003, pp. 463–481.

[78] A. Ito, R. Ueno, and N. Homma, "Toward optimal deep-learning based side-channel attacks: Probability concentration inequality loss and its usage," Cryptology ePrint Archive, Report 2021/1216, 2021, https://ia.cr/2021/1216.

[79] ——, "On the success rate of side-channel attacks on masked implementations: Information-theoretical bounds and their practical usage," in *ACM CCS*, 2022, pp. 1521–1535.

[80] ——, "Perceived information revisited: New metrics to evaluate success rate of side-channel attacks," *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, vol. 4, pp. 228–254, 2022.

[81] J. Jaffe, "A first-order DPA attack against AES in counter mode with unknown initial counter," in *CHES*, 2007, pp. 1–13.

[82] A. Joux and P. Delaunay, "Galois LFSR, embedded devices and side channel weaknesses," in *INDOCRYPT*, 2006, pp. 436–451.

[83] M. J. Kannwischer, P. Pessl, and R. Primas, "Single-trace attacks on Keccak," *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, vol. 2020, no. 3, pp. 243–268, 2020.

[84] L. Keliher and J. Sui, "Exact maximum expected differential and linear probability for two-round Advanced Encryption Standard," *IET Inf. Secur.*, vol. 1, no. 2, pp. 53–57, 2007.

[85] D. Knichel, A. Moradi, N. Müller, and P. Sasdrich, "Automated generation of masked hardware," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 1, pp. 589–629, 2022.

[86] D. Knichel, P. Sasdrich, and A. Moradi, "SILVER—statistical independence and leakage verification," in *ASIACRYPT*, 2020, pp. 787–816.

[87] ——, "Generic hardware private circuits: Towards automated generation of composable secure gadgets," *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, vol. 2022, pp. 323–344, 2021.

[88] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology—CRYPTO' 99*, ser. Lecture Notes in Computer Science, vol. 1666, 1999, pp. 388–397.

[89] S. Kumar, V. A. Dasu, A. Baksi, S. Sarkar, D. Jap, J. Breier, and S. Bhasin, "Side channel attack on stream ciphers: A three-step approach to state/key recovery," *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, vol. 2022, no. 2, pp. 166–191, 2022.

[90] Y. Linge, C. Dumas, and S. Lambert-Lacroix, "Using the joint distributions of a cryptographic function in side channel analysis," in *COSADE*, 2014, pp. 199–213.

[91] H. Maghrebi, T. Portigliatti, and E. Prouff, "Breaking cryptographic implementations using deep learning techniques," *SPACE*, pp. 3–26, 2016.

[92] J. Maillard, A. H. Meraneh, M. Sarry, C. Clavier, H. L. Bouder, and G. Thomas, "Blind side channel analysis on the Elephant LFSR extended version," in *E-Business and Telecommunications*, 2023, pp. 20–42.

[93] S. Mangard, E. Oswald, and F.-X. Standaert, "One for all – all for one: Unifying standard differential power analysis attacks," *IET Information Security*, vol. 5, no. 2, pp. 100–110, 2011.

[94] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer New York, 2007.

[95] S. Mangard, N. Pramstaller, and E. Oswald, "Successfully attacking masked AES hardware implementations," in *CHES*, 2005, pp. 157–171.

[96] G. Marsaglia, "Xorshift RNGs," *J. Statistical Sof.*, vol. 8, no. 14, pp. 1–6, 2003.

[97] B. Marshall, D. Page, and J. Webb, "MIRACLE: MIcRo-ArChitectural Leakage Evaluation: A study of micro-architectural power leakage across many devices," *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, vol. 2022, no. 1, pp. 175–220, 2021.

[98] L. Masure, C. Dumas, and E. Prouff, "A comprehensive study of deep learning for side-channel analysis," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2020, no. 1, pp. 348–375, 2020.

[99] L. Masure, P. Méaux, T. Moos, and F.-X. Standaert, "Effective and efficient masking with low noise using small-Mersenne-prime ciphers," in *EUROCRYPT*, 2023, pp. 596–627.

[100] L. Masure, O. Rioul, and F.-X. Standaert, "A nearly tight proof of Duc et al.'s conjectured security bound for masked implementations," in *CARDIS*, 2022, pp. 86–104.

[101] ——, "A nearly tight proof of Duc et al.'s conjectured security bound for masked implementations," in *Smart Card Research and Advanced Applications*, 2023, pp. 69–81.

[102] M. Matsumoto and T. Nishimura, "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Trans. Model. Comput. Simul.*, vol. 8, no. 1, pp. 3–30, 1998.

[103] M. Medwed, F. Standaert, J. Großschädl, and F. Regazzoni, "Fresh re-keying: Security against side-channel and fault attacks for low-cost devices," in *Progress in Cryptology—AFRICACRYPT 2010*, ser. Lecture Notes in Computer Science, vol. 6055. Springer, 2010, pp. 279–296.

[104] M. Medwed, F. Standaert, and A. Joux, "Towards super-exponential side-channel security with efficient leakage-resilient PRFs," in *CHES*, 2012, pp. 193–212.

[105] T. Moos, F. Wegener, and A. Moradi, "DL-LA: Deep learning leakage assessment: A modern roadmap for SCA evaluations," *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, vol. 2021, no. 3, pp. 552–598, 2021.

[106] A. Moradi, B. Richter, T. Schneider, and F.-X. Standaert, "Leakage detection with the χ^2-test," *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, vol. 2018, no. 1, pp. 209–237, 2018.

[107] N. Müller, D. Knichel, P. Sasdrich, and A. Moradi, "Transitional leakage in theory and practice: Unveiling security flaws in masked circuits," *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, vol. 2, pp. 266–288, 2022.

[108] S. Nikova, V. Rijmen, and M. Schläffer, "Secure hardware implementation of nonlinear functions in the presence of glitches," *J. Cryptol.*, vol. 24, no. 2, pp. 292–321, 2011.

[109] T. Oder, T. Schneider, T. Pöppelmann, and T. Güneysu, "Practical CCA2–secure and masked ring-LWE implementation," *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, vol. 2018, no. 1, pp. 142–174, 2018.

[110] O. Pereira, F.-X. Standaert, and S. Vivek, "Leakage-resilient authentication and encryption from symmetric cryptographic primitives," in *ACM CCS*, 2015, pp. 96–108.

[111] K. Pietzrak, "A leakage-resilient mode of operation," in *EUROCRYPT*, 2009, pp. 462–482.

[112] S. H. Prasad, F. Mendel, M. Schläffer, and R. Nagpal, "Efficient low-latency masking of ascon without fresh randomness," Cryptology ePrint Archive, Paper 2023/1914, 2023.

[113] E. Prouff and M. Rivain, "Masking against side-channel attacks: A formal security proof," in *EUROCRYPT*, 2013, pp. 142–159.

[114] M. Renauld, F.-X. Standeart, N. Veyrat-Charvillon, D. Kamel, and D. Flandre, "A formal study of power variability issues and side-channel attacks for nanoscale devices," in *EUROCRYPT*, 2011, pp. 109–128.

[115] O. Reparaz, "Detecting flawed masking schemes with leakage detection tests," in *FSE*, 2016, pp. 204–222.

[116] O. Reparaz, B. Bilgin, S. Nikova, B. Gierlichs, and I. Verbauwhede, "Consolidating masking schemes," in *Advances in Cryptology—CRYPTO 2015*, 2015, pp. 764–783.

[117] O. Reparaz, B. Gierlichs, and I. Verbauwhede, "Fast leakage assessment," in *CHES*, 2017, pp. 387–399.

[118] M. Rivain and E. Prouff, "Provably secure higher-order masking of AES," in *CHES*, 2010, pp. 413–427.

[119] T. Schneider and A. Moradi, "Leakage assessment methodology: A clear roadmap for side-channel evaluations," in *CHES*, 2015, pp. 495–513.

[120] D. Shanmugam and S. Annadurai, "Secure implementation of stream cipher: Trivium," in *SECITC*, 2015, pp. 253–266.

[121] A. M. Shelton, Ł. Chmielewski, N. Samwel, M. Wagner, L. Batina, and Y. Yarom, "Rosita++: Automatic higher-order leakage elimination from cryptographic code," in *ACM CCS*, 2021, pp. 685–699.

[122] A. M. Shelton, N. Samwel, L. Batina, F. Regazzoni, M. Wagner, and Y. Yarom, "Rosita: Towards automatic elimination of power-analysis leakage in ciphers," in *NDSS*, 2021.

[123] F. Sibleyras and Y. Todo, "Keyed sum of permutations: A simpler RP-based PRF," in *CT-RSA*, 2023, pp. 573–593.

[124] S. M. Sim, D. Jap, and S. Bhasin, "DAPA: Differential analysis aided power attack on (non-)linear feedback shift registers," *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, vol. 2021, no. 1, p. 169–191, 2020.

[125] F.-X. Standaert, O. Pereira, Y. Yu, J.-J. Quisquater, M. Yung, and E. Oswald, "Leakage resilient cryptography in practice," in *Towards Hardware-Intrinsic Security: Foundations and Practice*, A.-R. Sadeghi and D. Naccache, Eds., 2010, pp. 99–134.

[126] F.-X. Standeart, T. G. Malkin, and M. Yung, "A unified framework for the analysis of side-channel key recovery attacks," in *Advances in Cryptology—EUROCRYPT 2009*, ser. Lecture Notes in Computer Science, vol. 5479, 2009, pp. 443–461.

[127] T. Sugawara, "3-share threshold implementation of aes s-box without fresh randomness," *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, vol. 2019, no. 1, pp. 123–145, 2018.

[128] R. Ueno, N. Homma, and T. Aoki, "Toward more efficient DPA-resistant AES hardware architecture based on threshold implementation," in *COSADE*, 2017, pp. 50–64.

[129] R. Ueno, N. Homma, A. Inoue, and K. Minematsu, "Fallen sanctuary: A higher-order and leakage-resilient rekeying scheme," *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, vol. 1, no. 2024, pp. 264–308, 2023.

[130] R. Ueno, N. Homma, S. Morioka, and T. Aoki, "Automatic generation of formally-proven tamper-resistant Galois-field multipliers based on generalized masking scheme," in *DATE*. IEEE, 2017, pp. 978–983.

[131] ——, "A systematic design methodology of formally proven side-channel-resistant cryptographic hardware," *IEEE Design & Test*, vol. 38, no. 3, pp. 84–92, 2021.

[132] N. Veyrat-Charvillon, B. Gérard, and F.-X. Standaert, "Soft analytical side-channel attacks," in *ASIACRYPT*, 2014, pp. 282–296.

[133] W. Wang, P. Méaux, G. Cassiers, and F.-X. Standaert, "Efficient and private computations with code-based masking," *IACR Trans. Cryptogr. Hardw. Embedded Syst.*, vol. 2020, no. 2, pp. 128–171, 2020.

[134] Y. Yu, F.-X. Standaert, O. Pereira, and M. Yung, "Practical leakage-resilient pseudorandom generators," in *ACM CCS*, 2010, pp. 141–151.

[135] J. Zeitschner and A. Moradi, "PoMMES: Prevention of micro-architectural leakages in masked embedded software," *IACR TCHES*, vol. 2024, no. 3, pp. 342–376, 2024.

[136] J. Zeitschner, N. Müller, and A. Moradi, "PROLEAD_SW: Probing-based software leakage detection for ARM binaries," *IACR TCHES*, vol. 2023, no. 3, pp. 391–421, 2023.

Appendix A. Proof of Theorem 4.1

Proof. Before the proof main body, we introduce notations, TVLA, and common SCAs in a formal description.

Notations. Let $X_1, X_2, \ldots, X_n$ be n samples of random variable X. Its empirical mean and unbiased empirical variance are denoted by

$$\hat{\mu}[X] = \frac{1}{n}\sum_j X_j, \quad \hat{\sigma}^2[X] = \frac{1}{n-1}\sum_j (X_j - \hat{\mu}[X])^2,$$

respectively. Let $Y_1, Y_2, \ldots, Y_n$ be a jointly sampled random variables of Y as well. Unbiased empirical covariance between X and Y is given by

$$\hat{\sigma}[X, Y] = \frac{1}{n-1}\sum_j (X_j - \hat{\mu}[X])(Y_j - \hat{\mu}[Y]).$$

Assume that, for $(d-1)$-st order masked software implementation, a trace $\boldsymbol{X}$ consists of d leakages $L_1, L_2, \ldots, L_d \in \mathbb{R}$ corresponding to its share $A_1, A_2, \ldots, A_d$, respectively. In contrast, for $(d-1)$-st masked hardware implementation, assume that the leakage is univariate and a trace $\boldsymbol{X} = L$ is given as sum of $L_1, L_2, \ldots, L_d$. In the d-th order TVLA/SCA, we use d-th order leakage denoted by $V^{(d)}$. For d-variate SCA (typically on software implementation), the d-th order leakage is given by a product of zero-averaged d sample points as $V^{(d)} = \prod_{i=1}^{d}(L_i - \hat{\mu}_{L_i})$. For univariate TVLA/SCA (typically on hardware implementation), the d-th order leakage is given by the d-th power of a zero-averaged sample point as $V^{(d)} = (L - \hat{\mu}_L)^d$.

Test vector leakage assessment (TVLA). A d-th order TVLA assesses the existence of d-th order leakage using a non-parametric t-test [119]. The TVLA is designed to reject a null hypothesis that mean of d-th order leakage is not different when the input plaintexts are a fixed value or random. For the d-th order TVLA, the empirical t-value is computed using d-th order leakage in traces as

$$\hat{t} = \frac{\hat{\mu}\left[V_{\mathrm{fix}}^{(d)}\right] - \hat{\mu}\left[V_{\mathrm{rnd}}^{(d)}\right]}{\sqrt{\frac{\hat{\sigma}^2\left[V_{\mathrm{fix}}^{(d)}\right]}{n_{\mathrm{fix}}} + \frac{\hat{\sigma}^2\left[V_{\mathrm{rnd}}^{(d)}\right]}{n_{\mathrm{rnd}}}}},$$

where $V_{\mathrm{fix}}^{(d)}$ and $V_{\mathrm{rnd}}^{(d)}$ are random variables of d-th order leakage with fixed and random plaintexts, respectively, and n_{fix} and n_{rnd} are the respective numbers of traces. If the resultant t-value is higher than threshold, the null hypothesis is rejected and the alternative hypothesis is accepted, implying that there exists a d-th order leakage. Otherwise, the null hypothesis is (negatively) accepted. A double-sided test with a significance level of 10^{-5} have been commonly used, which corresponds to absolute t-value threshold of 4.5.

Correlation power analysis (CPA). Let $\phi : \{0,1\}^{n_s} \to \mathbb{R}$ be a leakage function (*e.g.*, Hamming weight), where n_s denotes bit length of Z. A d-th order CPA estimates the secret key as $\hat{k}$ using Pearson's correlation coefficient $\hat{\rho}$ as

$$\begin{aligned}\hat{k} &= \arg\max_k \left|\hat{\rho}\left[\phi(Z^{(k)}), V^{(d)}\right]\right| \\ &= \arg\max_k \left|\frac{\hat{\sigma}\left[\phi(Z^{(k)}), V^{(d)}\right]}{\hat{\sigma}[\phi(Z^{(k)})] \cdot \hat{\sigma}[V^{(d)}]}\right|.\end{aligned}$$

Profiled deep-learning based SCA (DL-SCA). Let $q_{\hat{\theta}}$ be a trained neural network as a profiled leakage model, which imitates $p_{Z|\boldsymbol{X}}$. A DL-SCA using the neural network estimates the secret key as

$$\hat{k} = \arg\max_k \hat{\mu}\left[\log q_{\hat{\theta}}(Z^{(k)} \mid \boldsymbol{X})\right].$$

Main body of proof. Common SCAs and TVLA compute the results solely using empirical mean, unbiased empirical variance, and unbiased empirical covariance, as described. These statistics are obviously invariant to the order of joint samples in $(T^n, \boldsymbol{X}^n)$ due to commutativity of addition. Thus, the leakage assessment and SCA results are invariant to the trace order. □

Appendix B. Proof of Theorem 5.1

Proof. This proof is for a graphical model in Figure 3, while many parts are in the same manner as [74], [78], [79]. This graphical model represents that the secret variable is function of secret key K and plaintext/ciphertext T and side-channel trace $\boldsymbol{X}$ depends on Z. The plate of n represents that n traces are available for the attacker. In a masked implementation, Z is decomposed into shares $A_0, A_1, \ldots, A_d$. Here, $Z \to A_0 \leftarrow A^d$ holds because the share decomposition is regarded as $(A_0, A_1, \ldots, A_d) = (Z \oplus R_0, R_1, \ldots, R_d)$, where $R_0, R_1, \ldots, R_d$ are correlated randomness, and $A_i \to A_0$ represents the randomness correlation as $R_0 = R_1 \oplus R_2 \oplus \cdots \oplus R_d$ holds *almost surely*. Note that the graphical model represents relation among random variables, but does not represent actual implementation. In other words, this model is valid regardless of actual implementation as long as the relation of random variables holds, which maintains the generality of this theorem. Leakages $\boldsymbol{L}_0, \boldsymbol{L}_1, \ldots, \boldsymbol{L}_d$ (where $\boldsymbol{X} = (\boldsymbol{L}_0, \boldsymbol{L}_1, \ldots, \boldsymbol{L}_d)$) correspond to $A_0, A_1, \ldots, A_d$, respectively, and are assumed independent. Attacker estimates K as $\hat{K}$ from T^n and $\boldsymbol{X}^n$.

Let $\mathbb{E}$ be the expectation and $\Pr$ be the probability measure. Let $\ell(\hat{k}, k) = \mathbb{1}_{\{\hat{k} \neq k\}}$ be a loss function. Optimal distinguisher is defined as a distinguisher that minimizes the loss function ℓ in guessing the correct key. Taking the expectation of $\ell(D(T^n, \boldsymbol{X}^n), K)$, we have

$$\begin{aligned}\mathbb{E}\ell(D(T^n, \boldsymbol{X}^n), K) &= \mathbb{E}\mathbb{1}_{\{D(T^n,\boldsymbol{X}^n) \neq K\}} \\ &= \mathbb{E}\mathbb{E}[\mathbb{1}_{\{D(T^n,\boldsymbol{X}^n) \neq K\}} \mid T^n, \boldsymbol{X}^n] \\ &= \mathbb{E}[1 - \Pr(K = D(T^n, \boldsymbol{X}^n) \mid T^n, \boldsymbol{X}^n)].\end{aligned}$$

Therefore,

$$\begin{aligned}D(T^n, \boldsymbol{X}^n) &= \arg\max_k \Pr(K = k \mid T^n, \boldsymbol{X}^n) \\ &= \arg\max_k p_{K|,\boldsymbol{X}^n,T^n}(k \mid T^n, \boldsymbol{X}^n),\end{aligned}$$

is an optimal distinguisher.

Then, we focus on a joint probability distribution $p_{K,\boldsymbol{X}^n,T^n}$ since $\arg\max_k p_{K|\boldsymbol{X}^n,T^n}(k \mid X^n, T^n) = \arg\max_k p_{K,\boldsymbol{X}^n,T^n}(k, \boldsymbol{X}^n, T^n)$. We hereafter omit the subscripts of probability distribution if there is no confusion. According to the marginalization in terms of Z^m, we have

$$\begin{aligned}p(k, \boldsymbol{X}^n, T^n) &= \sum_{z^n} p(k, z^n, \boldsymbol{X}^n, T^n) \\ &= \sum_{z^n} p(\boldsymbol{X}^n \mid z^n, T^n, k) p(z^n \mid T^n, k) p(T^n, k).\end{aligned}$$

Here, $p(\boldsymbol{X}^n \mid z^n, T^n, k) = p(\boldsymbol{X}^n \mid z^n)$, which follows from the fact that $\boldsymbol{X}^n$ and (T^n, K) are conditionally independent as Z^n d-separates $\boldsymbol{X}^n$ and (T^n, K) when Z^n is given [22]. As well, $p(T^n, k) = p(T^n)p(k)$, because T^n and K are conditionally independent because Z^n d-separates T^n and K when Z^n is not given. Since $\boldsymbol{X}^n$, Z^n and T^n consists of independent copies of $\boldsymbol{X}$, Z, and T, respectively, we have

$$\begin{aligned}p(k, \boldsymbol{X}^n, T^n) &= \sum_{z^n} p(\boldsymbol{X}^n \mid z^n) p(z^n \mid T^n, k) p(T^n) p(k) \\ &= p(k) \prod_{j=1}^{n} \sum_{z} p(\boldsymbol{X}_j \mid z_j) p(z \mid T_j, k) p(T_j).\end{aligned}$$

According to $Z^{(k)} = g(k, T)$ and Beyes' theorem, we have

$$\begin{aligned}p(k, \boldsymbol{X}^n, T^n) &= p_K(k) \prod_{j=1}^{n} p(\boldsymbol{X}_j \mid Z_j^{(k)}) p(T_j) \\ &= p(k) \prod_{j=1}^{n} \frac{p(Z_j^{(k)} \mid \boldsymbol{X}_j) p(\boldsymbol{X}_j) p(T_j)}{p(Z_j^{(k)})}. \qquad (2)\end{aligned}$$

Taking the logarithm of Equation (2), we have

$$\begin{aligned}\log p(k, \boldsymbol{X}^n, T^n) &= \log p(k) + \sum_{j=1}^{n} \log p(\boldsymbol{X}_j) p(T_j) \\ &+ \sum_{j=1}^{n} \log p(Z_j^{(k)} \mid \boldsymbol{X}_j) - \sum_{j=1}^{n} \log p(Z_j^{(k)}). \qquad (3)\end{aligned}$$

Consider the arg max of Equation (3) in terms of k. The first term $\log p_K(k)$ has no impact on the arg max because K is uniformly distributed. The second term $\sum_j \log p(\boldsymbol{X}_j)p(T_j)$ also has no impact because it is independent of k. Thus, assuming that Z is uniformly distributed, we conclude

$$\begin{aligned}D(T^n, \boldsymbol{X}^n) &= \arg\max_k \log p(k \mid \boldsymbol{X}^n, T^n) \\ &= \arg\max_k \sum_{j=1}^{n} \log p_{Z|\boldsymbol{X}}(Z_j^{(k)} \mid \boldsymbol{X}_j), \qquad (4)\end{aligned}$$

is an optimal distinguisher, as required.

The second equality is proven in the same manner as [79, Equation (2)]. Marginalizing $A_0, A_1, \ldots, A_d$, we have

$$p(z \mid \boldsymbol{X}) = \sum_{a_0, a_1, \ldots, a_d} p(z, a_0, a_1, \ldots, a_d \mid \boldsymbol{L}_0, \boldsymbol{L}_1, \ldots, \boldsymbol{L}_d).$$

Here, Z is conditionally independent of $\boldsymbol{L}^d$ if A^d is given, which d-separates Z and $\boldsymbol{L}^d$. Because $A_0, A_1, \ldots, A_d$ are mutually independent when Z is not given ($\boldsymbol{L}_0, \boldsymbol{L}_1, \ldots, \boldsymbol{L}_d$ are same) and $z = a_0 \oplus a_1 \oplus \cdots \oplus a_d$, it holds that

$$\begin{aligned}p(z \mid \boldsymbol{X}) &= \sum_{a_0, a_1, \ldots, a_d} p(z \mid a_0, a_1, \ldots, a_d) \prod_i p(a_i \mid \boldsymbol{L}_i) \\ &= \sum_{z = a_0 \oplus a_1 \oplus \cdots \oplus a_d} \prod_{i=0}^{d} p(a_i \mid \boldsymbol{L}_i). \qquad (5)\end{aligned}$$

By substituting Equation (5) to Equation (4), we complete the proof. □