Received: 7 August 2019 | Revised: 15 November 2019 | Accepted: 17 January 2020

DOI: 10.1002/cta.2756

RESEARCH ARTICLE

WILEY

Beyond algorithmic noise or how to shuffle parallel implementations?

Itamar Levi[1,2] | **Davide Bellizia**[2] | **François-Xavier Standaert**[2]

[1]Faculty of Engineering, Bar-Ilan University, Ramat Gan, Israel

[2]ICTEAM/ELEN, Université Catholique de Louvain, Louvain-la-Neuve, Belgium

Correspondence
Itamar Levi, Faculty of Engineering, Bar-Ilan University, Ramat Gan 52900, Israel.
Email: itamar.levi@biu.ac.il

Funding information
SWORD, Grant/Award Number: 724725; H2020 European Research Council, Grant/Award Number: 731591

Summary

Noise is an important ingredient for side-channel-analysis countermeasures security. However, physical noise is in most cases not sufficient to achieve high-security levels. As an outcome, designers traditionally aim to emulate noise by harnessing shuffling in the time domain and algorithmic noise in the amplitude domain. On one hand, harnessing algorithmic noise is limited in architectures/devices which have a limited data-path width. On the other hand, the performance degradation due to shuffling is considerable. A natural complement to operation shuffling is the hardware-based intra-cycle shuffling (ICS), which typically shuffles the sample time of bits within a clock cycle (instead of micro-processor operations). Such architecture eliminates the performance overhead due to shuffling within a single cycle, it is algorithm-independent, i.e., no need in partitioning of operations, and as it is hardware-based, the data-path width can be tailored to better exploit algorithmic-noise. In this manuscript, we first analyze the noise components in physical designs to better model the algorithmic noise. We then perform an information-theoretic (IT) analysis of both shuffling countermeasures . The last part of the manuscript deals with real-world architectures analysis: IT analysis of an Advanced Encryption Standard (AES) core implemented over a 32- and 128-bit wide data-path embedded with intra-cycle shuffling and two flavors of shuffling generation (memory-based and on-line permutation generation). The manuscript is concluded by underling the benefits which can be achieved with the ICS architecture.

KEYWORDS

algorithmic noise, hardware security, hiding, intra-cycle shuffling, mutual information, pAsynch, side-channel analysis, shuffling

1 | INTRODUCTION

Noise is an important ingredient for the security of embedded cryptographic devices against side-channel-analysis (SCA) attacks. It is needed, for example, in masked implementations to deliver its security guarantees, ie, an exponential amplification of the noise, which can only work if you have noise in the first place. In many devices, the amount of physical noise is not enough for ensuring high security levels. Therefore, various solutions have been introduced to add/emulate noise in the time domain (eg, random-process interrupts [RPI], random-delay insertion[1,2] and execution order *shuffling*[3,4])

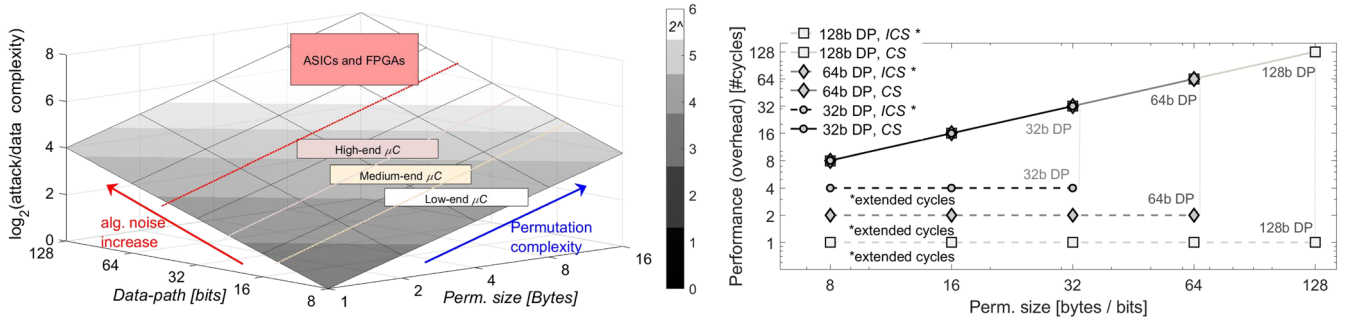


FIGURE 1 (a) Algorithmic noise vs. operation shuffling: attack (data) complexity as a function of the data-path width and permutation size; (b) Operation shuffling (CS) vs. intra-cycle shuffling (ICS): performance overheads (in cycles) [Colour figure can be viewed at wileyonlinelibrary.com]

and amplitude domain (e.g., algorithmic-noise,* and additive noise generators). In the current state of the art, shuffling is usually considered over operations,† denoted by cycle shuffling (CS), and mostly useful for serial (ie, software) implementations, while algorithmic-noise is more dedicated to parallel (ie, hardware) implementations.

The security guarantees of algorithmic noise increase with the number of algorithmic-noise elements (*#alg.-elem*) and decrease with the adversary's guessing power (complexity). In turn, the number of algorithmic elements increases with the data-path width ($width_{DP}$) of the device, #alg.-elem $= \frac{\text{width}_{\text{DP}}\,[bits]}{\text{adv. compl}\,[bits]}$. Parallel implementations enjoy larger algorithmic noise than serial implementations, in which the data-path width (in bits) is quite limited. It leads to the type of tradeoffs demonstrated in Figure 1A, which essentially means we can hardly combine shuffling and algorithmic noise. The attack data complexity‡ as a function of the data-path width and the number of shuffled time samples (*#t-perm*) is illustrated in Figure 1A. It increases linearly both with the number of algorithmic elements§ and with the number of shuffled time-samples¶ (assuming uniform time permutations).

Ideally, designers would like to exploit both knobs. In practice, taking the example of the AES algorithm, shuffling the substitution layer into 16 one-byte operands is architecturally easy. By contrast, the mixed-columns layer which operates over 32-bit words is more challenging to shuffle, which typically leads to a cost overhead of 4x the number of cycles. However, the most limiting part of the system is in fact the key-scheduling procedure, where it is quite challenging to partition it into bytes of independent operations. This scenario of inefficiency due to algorithm dependence actually scales up badly (eg, consider an AES-256). Figure 1B illustrates a lower bound of the performance overheads (# cycles) for a CS implementation with different data-path widths (32-, 64-, and hypothetical 128-bits) as a function of the permutation size. As shown, the number of cycles increases linearly with the permutation size (the possible number of permutations is bounded by the number of algorithmic-elements, for a given data-path width). Recently, in CHES-[5] Papagiannopoulos has also demonstrated the severe performance overheads of CS designs and discussed how to mitigate them by trading-off security with recycled randomness when the design is masked.

For the above-mentioned reasons, in this paper, we focus on hardware parallel implementations. A recently proposed scheme which we investigate in this manuscript is denoted by *intra-cycle* shuffling, intra-cycle shuffling (ICS) (originally dubbed *pAsynch*)[6] (elaborated in Section 3). In this scheme, the time samples containing leakage are distributed within a single clock cycle. The ICS design brings a natural complement to eliminate the tradeoffs discussed above with a considerable performance gain. First, it is applicable for parallel implementations (typically ASICs and FPGAs), which inherently benefit from algorithmic noise. Second, the time manipulations are performed intra-cycle (on bits), instead on operations (or bytes), as done in CS. Therefore, the ICS design makes time manipulation possible in an algorithm-independent way. As an outcome, it has the potential to considerably reduce the performance overheads of standard shuffling architectures. This is illustrated in Figure 1B: an ICS design with 128-bit data path needs only one computation cycle to perform bits shuffling on subsets of up to 128 bits and an ICS design with 32-bit data path needs only four cycles to perform shuffling on

*Here, we denote by algorithmic noise, the noise induced by concurrent algorithmic elements which are not targeted by the adversary.
†Operations is specifically referred here to software or microprocessor instructions.
‡Considering a univariate Pearson correlation distinguisher, the data complexity is proportional to $\frac{1}{\rho^2}$.
§The noise variance increase linearly as demonstrated in the next section.
¶The correlation of the modified design can be written by $\rho_{modif} = \frac{\rho_{un-prot}}{\#\text{t - perm-}\#\text{alg - elem}}$.

subsets of up-to 32 bits.[#] Protecting such architectures is motivating since, for example, in front of an adversary with 2^{32} guessing power, we expect a best-case signal-to-noise (SNR) ratio of 1 (32 bits) or 1/4 (128 bits) due to algorithmic-noise, which is far higher than the SNRs typically needed for proper masking. Therefore, we aim to reduce the SNR by properly combining such schemes with ICS for this important class of architectures.

Driven by the limitations discussed above, in this manuscript, we deal with the questions, "how it is possible to jointly foster both algorithmic noise and time-shuffling in an algorithm independent fashion?" and "what security-gains we can achieve in a strong multi-variate evaluation setting?" The contributions and organization of the manuscript are as follows:

- We first show (in Section 2) that algorithmic noise may not be enough for our target architectures to reach high security levels by modeling it accurately.
- We then discuss ICS as a natural solution to be combined with algorithmic noise.
- We analyze ICS and CS comparatively based on simulations in Section 3. We provide an intuitive study of the leakage distributions of these two solutions and a systematic information theoretic analysis and conclude that both can theoretically be used as noise emulators. We confirm this simulated analysis with experimental data (field-programmable gate array [FPGA] measurements).
- We then examine the impact of additional limitations that concrete adversaries may face (Section 4).
- The paper is concluded by investigating 32-bit and 128-bit AES case studies (Section 5), and put forward that ICS can bring significantly improved security quite efficiently, and that the performance overheads it implies are considerably reduced compared with CS.

Notations: In this manuscript, variables are denoted with capital letters, sampled values with lowercase letters, functions with sans serif fonts, and vectors with bold letters.

2 | ALGORITHMIC NOISE MAY NOT BE ENOUGH

In this section, we define the (exploitable) cryptographic signal and the noise components associated with physical leakages, e.g., power supply or electromagnetic measurements. The goals of this section are as follows:

- To clearly identify the dominant noise sources in cryptographic implementations, i.e., to show that the logical part of the algorithmic noise element dominates the noise and that the physical noise is at least 1.5 orders of magnitude smaller (on a common 65nm technology node).
- To understand how to model the algorithmic noise as a function of the number of algorithmic elements in the design.

Let Y be an n-bit sensitive variable manipulated by a cryptographic algorithm, as illustrated in Figure 2. During the device operation, it leaks information which is associated with data manipulation and physical and/or environmental parameters throughout side-channels; here, we focus on the power-supply current leakage. The security level of physical devices goes hand-in-hand with the abilities of the adversary (computational power, physical access, device knowledge, etc). Therefore, our security parameter is the adversary's guessing power in bits; i.e., assume an adversary who exploits a divide-and-conquer approach over a subset of the secret variable of s bits ($s \leq n$ as illustrated in Figure 2). Generally, the guessing power also defines the unit-under-attack (UUA). The reminder of the cryptographic algorithm (which is processed in parallel) is referred to as Concurrent-Algorithmic-Operations (CAO). The leakage of the algorithmic elements (or CAO) is referred to as algorithmic noise. All of these leakage components are filtered and accumulated while flowing through the power supply network grid as illustrated on the figure.

The leakage of both the UUA and CAO is divided into three components as follows:

1. *The logical part*—induced by transistors and gates switching within the UUA/CAO (ie, associated with logic activity). The part of it which leaks from the UUA is typically referred to as the adversary exploitable *signal* and the part which leaks from the CAO is the dominant noise element within the leakage and denoted as algorithmic noise.
2. *The physical part*—this leakage component reflects the true physical noise which is associated only with environmental factors (eg, temperature, voltage-supply variations, and radiation) and technological parameters (eg, manufacturing process technology node, measurement setup, and acquisition devices). This component is a pure data-independent noise factor within the leakage.

[#] The cycle in this case is extended (denoted by a $*$ on the figure) due to local shifting of clocks phases.[6]

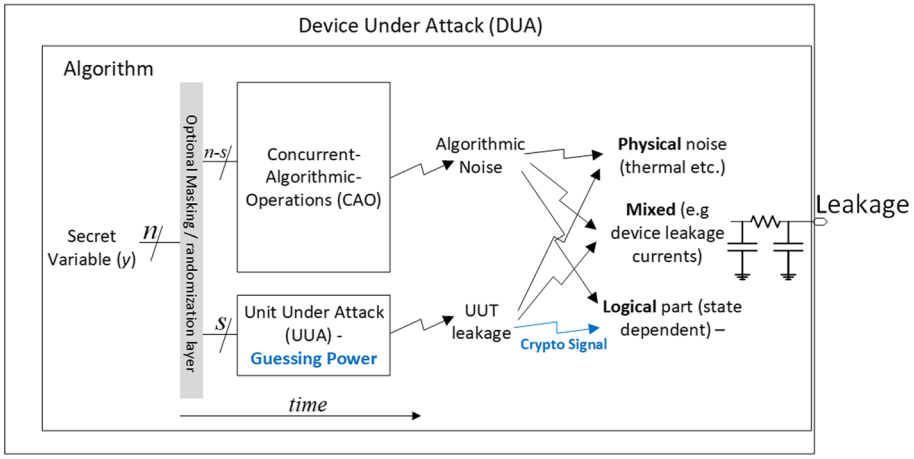


FIGURE 2 Signal and noise components of a leaking cryptographic device [Colour figure can be viewed at wileyonlinelibrary.com]

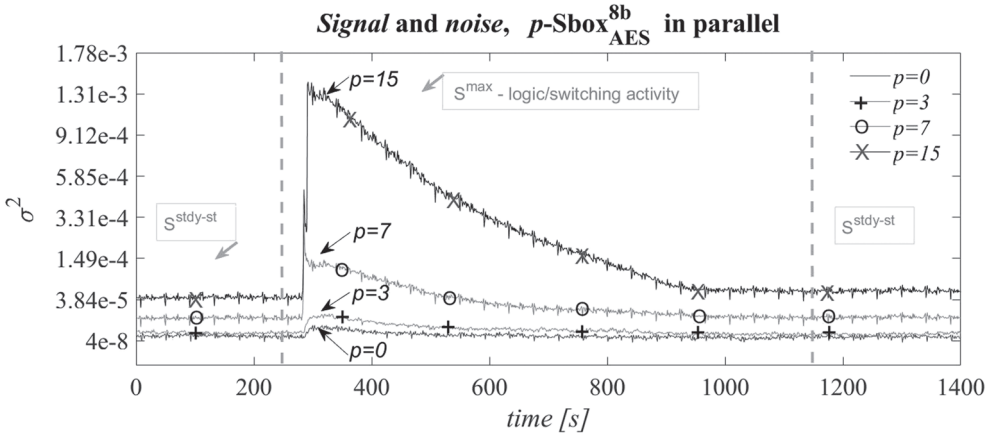


FIGURE 3 Signal and algorithmic noise values vs time for different data-path widths of an AES-128 implemented on a Xilinx Spartan VI FPGA, in [A]

3. *An inevitable mixed part*—some leakage components closely tie the last two factors. That is, they highly depend both on the logical state of the device and on physical parameters. Perhaps the most eminent example is devices (transistors) static leakage currents while in steady state. These currents are highly sensitive to, eg, temperature variations.[7,8] In some scenarios, this element can be utilized to extract state-dependent secrets (eg, leakage power analysis, LPA[9-11]).

We first formulate the computation of the different leakage elements. Note that these definitions follow the Mangard et al ones[12] (pg. 73) for the cryptographic *signal* and *noise* and further extend them to separate the noise into physical- and logical-noise components. As described above, the leakage (L) depends on logical activity within the UUA, the CAO, and on the noise (N), L(UUA,CAO,N). The signal (S), logical-noise (N_l), physical-noise (N_p), and total-noise (N_t) are defined in Equations (1) to 4, respectively:

$$S = \underset{\text{UUA}}{\mathsf{Var}}\left(\underset{\text{CAO}}{\mathsf{E}}\left(\underset{N}{\mathsf{E}}(L)\right)\right), \tag{1}$$

$$N_l = \underset{\text{UUA}}{\mathsf{E}}\left(\underset{\text{CAO}}{\mathsf{Var}}\left(\underset{N}{\mathsf{E}}(L)\right)\right), \tag{2}$$

$$N_p = \underset{\text{UUA, CAO}}{\mathsf{E}}\left(\underset{N}{\mathsf{Var}}(L)\right), \tag{3}$$

$$\begin{aligned} N_t &= \underset{\text{UUA}}{\mathsf{E}}\left(\underset{\text{CAO, }N}{\mathsf{Var}}(L)\right), \\ N_t &= N_l + N_p, \quad (N_l >> N_p \text{ when } n \uparrow \ ) \end{aligned} \tag{4}$$

where Var and E are the variance and expectation operators, respectively. In this section, we focus on a univariate analysis in order to put forward simple intuitions.

Before discussing and evaluating the algorithmic noise measured on an FPGA device, which will be used for further experiments in the following, we summarize here the main conclusion from an analog simulation environment which lets

us isolate and evaluate the different noise and signal components (ie, N_p, N_l, and S), and more importantly, to understand the magnitude and effect of the *mixed* leakage component (ie, leakage which depends jointly on logical activity and physical parameters): Our simulations indicate that the physical noise is generally not sufficiently large and much lower than the algorithmic noise (or logical noise). To confirm these results and to evaluate the impact of the algorithmic noise in real setting, this section continues with measured FPGA results.

Our goal here is to understand the behavior of algorithmic-elements noise (CAO) as a function of the circuit-size. This is not a trivial task because it depends on the leakage function, which can take many forms,[‖] each of which induces quite significantly different *logical* noise. In order to base our models and assumptions in the following on verifiable parameters, an examination of the signal and algorithmic noise values of an AES-128 implementation (over a Xilinx Spartan-6 FPGA embedded in the SAKURA-G board) is performed. In the experiments, we capture the maximum exploitable signal, S, in the relevant point-of-interest (POI) in time (denoted by S^{max}). We also compute the exploitable signal during transistors leakage in steady state (denoted by $S^{stdy-st}$). Figure 3 shows S as a function of the data-path width (number of algorithmic elements processed, denoted by p, in terms of 8-bit AES Sboxes). The computation of each curve in the figure involved an averaging of $1 \cdot 10^6$ traces per input (we detail on the evaluation and measurement environment in Subsection 3.4). The examination of an 8-bit wide data path (equivalently, one Sbox at a time, $p = 0$) shows the inherent maximum signal due to logical transition (Equation (1)), assuming in this case that eight is the guessing power of the adversary. The rest of the curves ($p \in \{3, 7, 15\}$) demonstrate cases where a data path of 32 ($p = 3$), 64 ($p = 7$) or 128 bits ($p = 15$) is available. It is observed that the logic part of the algorithmic noise increases more rapidly than linearly (denoted by S on Figure 3, as the entire circuit is treated as the UUA).

Our experiments indicate that the inherent (UUA) noise components are (at least) 1.5 orders of magnitude smaller than the exploitable signal, which makes them an insufficient ingredient for security purposes. In turn, algorithmic noise is an important and dominant factor, making it more suitable for security purposes. However, it is not enough to gain concrete security levels.

Following the examination in this section, from this point and along the manuscript, the leakage function will be modeled by $l_y = L_{UUA}(y) + N_0 + p \cdot L_{CAO}(r)$, where L_{UUA} and L_{CAO} represent the actual leakage functions which primarily depend on the functionality of the UUA and the CAO logic and on the sensitive-variables y and r, respectively. Where, r represents a secret variable outside the enumeration power of the adversary, which therefore is treated as a random computation. The physical noise generated by the entire circuit (UUA and CAO), ie, $N_t - N_l$, is represented by N_0. The logical noise due to algorithmic elements (CAO) is modeled as proportional to the number of CAO elements, where each of which is in the size of y, $|r|=|y|$, denoted by p for *parallel*. That is, it represents N_l.

3 | INTER- AND INTRA-CYCLE SHUFFLING

In this section, we elaborate on operation shuffling (CS) and on bit-shuffling (ICS) through a simple $m = 3$-bit architecture example. The goals of this section is to build understanding and intuition about the leakage characteristics of these designs. For this purpose, first, we analyze the leakage distribution of these designs in an idealized setting. We then perform information-theoretic analysis, and the section is concluded with real-life experiments over an FPGA platform to confirm the relevance of the proposed leakage distribution models. One of the main messages of the section is that the ICS design is preferred in low physical-noise environments and in high noise scenarios, the different shuffling architectures are quite equivalent, which is encouraging due to the low latency of ICS.

3.1 | Three-bit example

We denote by CS a shuffling scheme which is typically performed on operations, usually in software case-studies. By ICS, we denote shuffling which is typically performed over the bits of an operation, which is usually more suitable to hardware case studies. ICS is performed eg, using a set of locally generated delayed versions of the main clock as was demonstrated in Levi et al[6] and was also examined in prior work.[13,14] Essentially, each bit of the mSboxes' bits is sampled in a different time sample (*intra-cycle*). The set of m-shifted clock phases is randomly permuted (per cycle) and assigned to the m-bit registers.

From the security evaluation perspective, the CS approach was more concretely formulated and evaluated in Veyrat-Charvillon et al.[15] In this manuscript, we evaluate both CS and ICS in a more general view, taking into considera-

[‖] Traditional models are the Hamming weight, $\mathsf{W}(\cdot)$, the Hamming distance, and the identity leakage function.

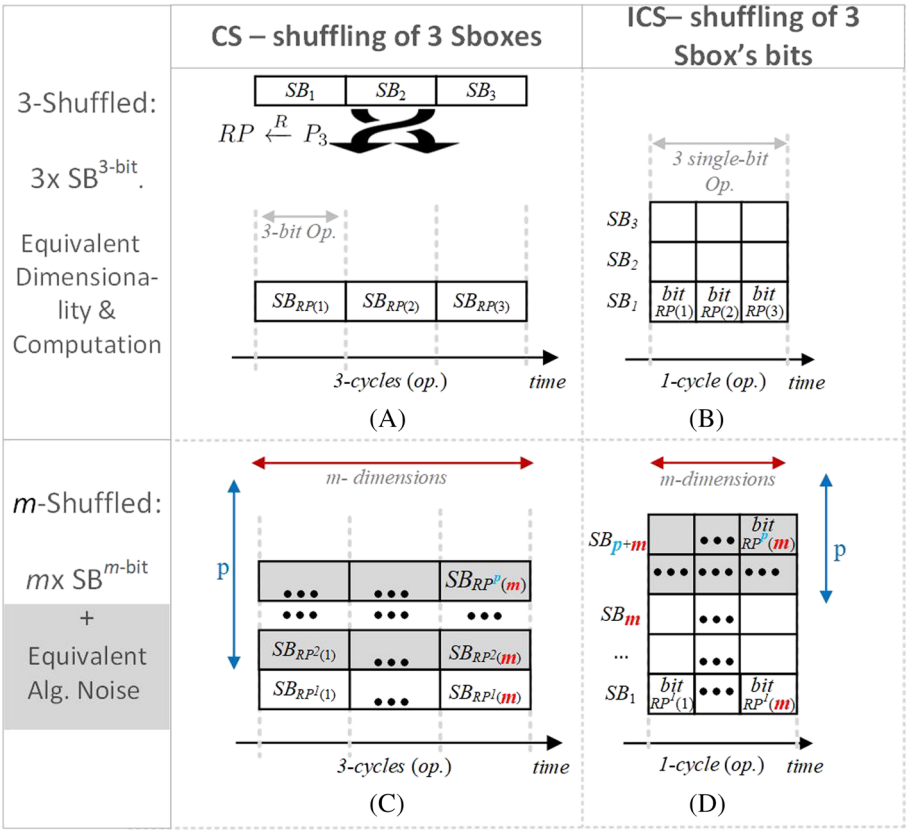


FIGURE 4 Illustration of (A) three dimensions of 3-bit Sboxes operations-shuffling or cycle shuffling (CS) (B) three dimensions of bits-shuffling or intra-cycle shuffling (ICS) (C) m-dimensions CS with alg. noise and (D) an m-dimensional ICS with alg. noise [Colour figure can be viewed at wileyonlinelibrary.com]

tion the architecture (eg, data-path width) and practical physical constraints. Moreover, we aim to take the understanding and intuition regarding the leakage distributions of these designs a step forward.

For simplicity, we now assume that the *shuffled* module is a Substitution-box (Sbox) which is a common and practical setting. For the ease of presentation and analysis, we explore a small architecture of three elements shuffling, $m = 3$, (or three dimensions). For CS, each element or dimension in the leakage is the leakage of an m-bit Sbox. For ICS, each element or dimension in the leakage is the leakage of a bit manipulated within an Sbox. Figure 4A illustrates the execution order (in time) of a series of three Sboxes over (an abstract) 3-bit microprocessor. For the CS approach, the element to be processed in a given clock cycle depends on $RP(1:3)$, which is a random permutation out of the possible 3! orderings of [1:3], $RP \overset{R}{\leftarrow} P_3$. Figure 4B schematically illustrates the logic structure of the ICS design, in which (as was originally proposed) each of the three bits of the module is sampled in a different (delay permuted) time within the clock cycle. We denote by $RP^i(j)$ a random permutation draw number i with j an element in the permutation (the index i is omitted when not strictly needed). In the original construction, this was achieved by generating local shifted versions of the main clock signal. In this 3-bit example, it is assumed that three clock phases are locally generated in hardware and each of them is assigned (in each computation cycle) to a different register of one bit after order permutation. For a fair comparison, both designs are of the same dimensionality (ie, 3) and perform the same computation (processing of three Sboxes). In the ICS design, and as typical in hardware implementation, the processing of the different Sboxes is done in parallel. Each bit of the three Sbox bits is sampled in a different time sample (*intra-cycle*), $t_{RP(i)}, i \in \{1..3\}$. Figure 4C,D illustrate an m-dimensional CS and ICS shuffling with equivalent alg. noise level, correspondingly, highlighted in gray. As denoted above, p represents the number of parallel algorithmic elements (CAO). For the CS architecture, p relates to the number of elements above one Sbox and for the ICS (for fair comparison to CS) it is the number of elements above m.

3.2 | Simulated analysis

The first step in understanding the extent of information leakage is by analyzing the leakage probability distributions. In this subsection, the leakage distributions of the exemplary 3-bits Sbox architecture of the CS and ICS designs are illustrated. In these cases, the leakage is multi-dimensional of dimensionality 3 (three leakage samples).

In the simulated setting, we assume that the leakage from a manipulation of a sensitive variable v follows the Hamming weight of it, $\mathsf{W}(v)$. Let us consider the sensitive variable as the output of an Sbox, e.g., $y_i = \mathsf{Sbox}(x_i \oplus k)$, where x_i is a plaintext chunk and k is a secret key chunk. Shuffling algorithms start with a random and uniform draw from the set of all permutations of $\{1, \ldots, m\}$ denoted by $P_m, RP \xleftarrow{R} P_m$. The leakage in the CS case, where the permuted elements represent the execution order of the Sboxes, can be written as $L_{t_i} \leftarrow \mathrm{W}(y_{RP(i)}) + N_0^i$, where, N_0^i represents an independent Gaussian noise of the element in location i of the permutation in time t_i. In the ICS case, the permuted elements represent the sampling order of the Sbox's bits. In each clock-cycle j, there exist m leakage time instances ($i \in \{1...m\}$) where in each only one bit is processed: $L_{t_i} \leftarrow \mathrm{W}(y_j^{RP(i)}) + N_0^i$. In this case, y_j is the computation taken in cycle j and the superscript $RP(i)$ indexes one bit out of the the vectorial Sbox output of m bits. The joint leakage is the concatenation of the set of m leakage time points which contain information on the manipulation of y: $\boldsymbol{L} = \{L_{t_1}, L_{t_2}, L_{t_3}\}$ in the case of $m = 3$.

Starting with an intuitive example for the leakage distribution, the leakage of a single 3-bit CS shuffled Sbox is discussed, where on the remaining two cycles the system is in *idle* (no computation) and therefore does not leak information. The leakage distribution of the design is a Gaussian mixture. For each value of y, the conditional distribution, $\Pr(\boldsymbol{L} = \mathbf{l}|Y = y)$, depends on the random permutation, where if the permutations are not biased and equi-probable, the

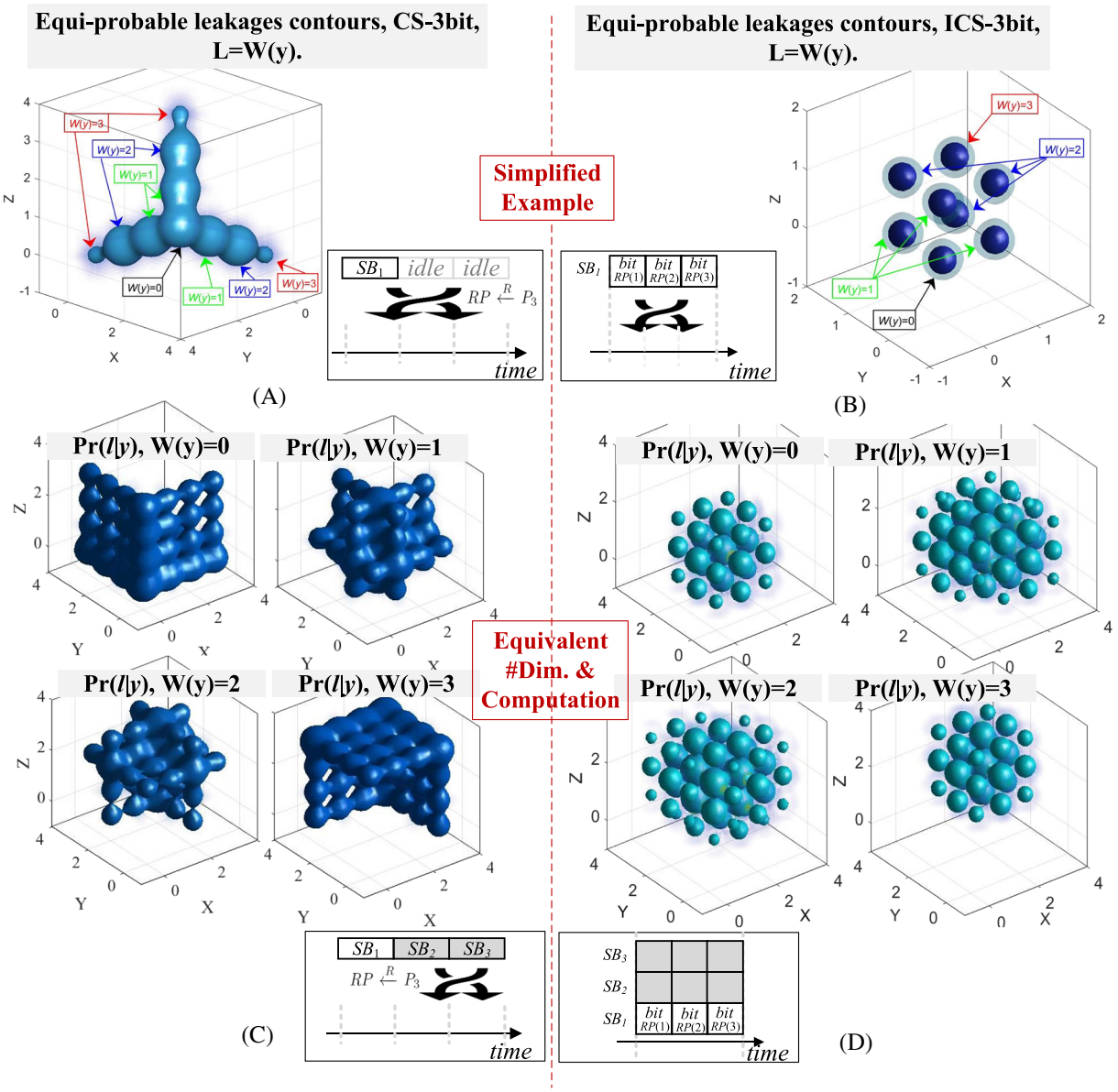


FIGURE 5 Leakage distribution contours for $y = i,\ i \in \{0..7\}, \boldsymbol{L} = \mathsf{W}(y)$ of (A) *simplified* 3-bit cycle shuffling (CS) and (B) *simplified* 3-bit intra-cycle shuffling (ICS) and of (C) 3-bit CS and (d) 3-bit ICS [Colour figure can be viewed at wileyonlinelibrary.com]

outcome probability density distribution is shown in Figure 5A. For example, for $y = 000 : \mathsf{W}(y) = 0$ and therefore, the (noiseless) leakage means are at the origin $\boldsymbol{L} = \{0, 0, 0\}$. For y states which lead to $\mathsf{W}(y) = j, j \in \{1, 2\}$, the leakage means will be situated with equal probability in $\boldsymbol{L} = \{j, 0, 0\}, \{0, j, 0\}$ or, $\{0, 0, j\}$. Note that the *noiseless* distribution spans over an area of $(max_y \mathsf{W}(y))^2$. An equivalent (dimensionality and computation-wise) ICS design is schematically illustrated in Figure 5B, where the single Sbox bits are sampled and shuffled within the computation (intra-cycle). By looking carefully at the figure, in this case for $\mathsf{W}(y) = j, j = 1$, the distribution is identical to the CS design. However, for $\mathsf{W}(y) = 2$: $\boldsymbol{L} = \{1, 1, 0\}, \{1, 0, 1\}$ or, $\{0, 1, 1\}$ with equal probability and for $\mathsf{W}(y) = 3$, the leakage distribution mean is at: $\boldsymbol{L} = \{1, 1, 1\}$. As this example demonstrates, the ICS provide *denser* and more centered distribution (constellation), which implies that with the same level of noise, different conditional distributions will overlap more than with the case of the CS design. In other words, the distribution now spans over an area of only $(max_y \mathsf{W}(y))^2$, which is 4Œ smaller than the CS design and it is also much more symmetrical.

Returning to the complete example of three Sboxes of 3-bits each (Figure 4C,D), in the ICS design case, due to the bit-wise computation, there are three dimensions; therefore, it is compared with a 3-cycle CS design. In the CS design, the UUA (Sbox) is processed in one (randomly selected) cycle. On the other cycle, a random Sbox computation is performed, therefore shuffling the manipulation of the set $\{y, r_1, r_2\}$, where r_is are drawn at random. As discussed above, to preserve the data complexity of the ICS design, the two other Sboxes (as of the 3-Sbox CS design) are processed in parallel, leading to leakages: $L_{t_i} \leftarrow \mathrm{W}(y_j^{RP(i)}) + N_0^i + \sum_{l=1}^{p} \mathrm{W}(r_l)$. In this case, $p = 2$ and r is a fresh random bit for each l. In this example, not only the conditional distribution $\Pr(\boldsymbol{L} = \mathbf{l}|Y = y)$ depends on the random permutations, it also depends on the random values processed in the non-UUA elements. Figure 5C shows the CS conditional leakage distributions where each of the four figures illustrates the conditional distribution conditioned on y. For $\mathsf{W}(y) = j, j \in \{0, 1, 2, 3\}$, the (noiseless) leakage means can be any equally probable triplet of the form $\boldsymbol{L} = \{0, j, j\}, \{j, 0, j\}$, or $\{j, j, 0\}$, where $j \in \{0, 1, 2\}$ as shown.

For the ICS design, the outcome probability density distributions are shown in Figure 5D). In this case, the distributions shown in Figure 5B serve as a carrier distribution where an equi-probable Gaussian-mixture of Sboxes number two and three modulates the carrier distribution. The distribution of the algorithmic Sboxes (two and three) is with means at $\{j, k, m\}, \{j, k, m\} \in \{0, 1, 2\}$. The outcome distributions clearly overlap considerably more (between different y states), which implies reduced information leakages.

The analysis performed in this section can extend to more dimensions or alternatively to larger number of bits. In the following sections, we utilize these probability distributions to analyze information-theoretic (IT) characteristics of these designs.

3.3 | IT analysis of an unbounded adversary

To characterize the information-leakage of the CS and ICS designs, in this manuscript, we utilize on IT-based metric, namely the mutual-information (MI) metric[16,17]:

$$\begin{aligned} &\mathrm{MI}(Y;\ \boldsymbol{L}_Y) = \\ &\mathrm{H(Y)} - \sum_{y_i \in \mathrm{Y}} \Pr[y_i] \cdot \sum_{\mathbf{l} \in \boldsymbol{L}_\mathrm{Y}} \overset{\%}{\underset{chip}{\Pr}}[\mathbf{l}|y_i] \cdot log_2(\underset{model}{\Pr}[y_i|\mathbf{l}]), \end{aligned} \tag{5}$$

where H[Y] is the entropy of the sensitive variable Y and $\Pr(\mathbf{l}|y_i)$ is the conditional PDF of the leakages, $\mathbf{l}$, given the secret manipulation of y_i. If assuming a Gaussian noise, it can be written as a Gaussian mixture model, $\Pr(\mathbf{l}|y_i) = \sum_{r \in \mathrm{R}} N(\mathbf{l}|y_i, r, \sigma_n^2)$, where R represents the set of all possible time permutations (in our designs).** The conditional probability, $\Pr(y_i|\mathbf{l})$, can be computed by Bayes' theorem as done in previous studies.[16,17] When the setting represents a simulated environment, $\Pr_{chip}(\mathbf{l}|y_i)$ does not need to be estimated, it can be computed or written in a mathematical form, without model-estimation errors from a "chip" reflecting the MI. When the leakage is derived from a real environment and it needs to be estimated, possibly suffering from model/estimation/PDF-sampling errors, the resulting computation denotes the perceived information (PI).[18]

As explained and demonstrated in Standaert et al,[16] the metric quantifies how much can be learned on Y from the leakage $\boldsymbol{L}_\mathrm{Y}$. When the analysis setting is simulated, the leakage distribution might be available in a continuous mathematical form where the second summation in this case transforms to an integral as follows:

**And inputs to algorithmic elements.

$$\mathrm{MI}(\mathrm{Y};\ \boldsymbol{L}_{\mathrm{Y}}) = \mathrm{H}(\mathrm{Y}) - \sum_{y_i \in \mathrm{Y}} \Pr[y_i] \cdot \int_{\mathbf{l}_1,\ldots,\mathbf{l}_m \in \boldsymbol{L}_{\mathrm{Y}}} \int \int \int \Pr_{chip}[\mathbf{l}_1,\ \ldots\ ,\mathbf{l}_m | y_i] \cdot log_2(\Pr_{model}[y_i|\mathbf{l}_1,\ \ldots\ ,\mathbf{l}_m]). \tag{6}$$

In the case of multi-dimensional leakages, the integral becomes multi dimensional. In turn, the computations involved are computationally expensive multi-dimensional integrals. It is important to highlight that numerically solving integrals of increased dimensionality as needed in this work is of extremely high (exponential) cost, in terms of computational complexity and memory requirements. Whereas, uni-dimensional integrals (for an unprotected design) can be computed quite instantaneously. Already for the three-dimensional distributions of the complex Gaussian mixture (GM) of the I/CS designs (a mixture of about 80/48 Gaussian functions for only three bits!), the computation takes several days while utilizing 20 processing cores for different noise levels. From this complex analysis, we aim to derive trends for the security while increasing dimensionality (complexity), as typically done in abstraction of complex systems.

In this subsection, we evaluate the information leakage assuming a worst-case adversary who is not bounded in any aspect (computational power, data storage complexity, measurements accuracy, device access, etc). The noise is assumed to be additive and Gaussian with zero mean and the noise standard variation σ_n is a parameter which reflects the true physical noise of the design. The leakage function used is the common Hamming weight function (denoted by $\mathsf{W}(\cdot)$).

Figure 6 shows the MI that can be learned about the manipulated variable from the 3-bit (and 3-Sboxes) CS and ICS designs (denoted by CS_{3b} and ICS_{3b} with a superscript of *third* to denote a three-dimensional analysis). In addition, to strengthen our arguments, the MI of 2-Sboxes of 3-input and 2-output bits of CS and ICS is shown (denoted by CS_{2b} and ICS_{2b} with a superscript of *second* to denote a two-dimensional analysis). The MI is computed as a function of the noise standard deviation σ_n of the modeled Gaussian-mixture distributions. The CS curves appear in red line with square marks and the ICS curves are blue with circle marks. On the same plot, the results obtained from an unprotected modeled design (UnP) of $\{2, 3, 8\}$-bit Sboxes are shown in black (labeled with a star).

Several important observations can be made from these results as follows:

1. For low noise variance, the MI of the ICS designs is smaller than that of the CS design considerably. This is due to the more condensed distribution and the larger overlap between different conditional distributions.
2. For medium to large noise variance, from a certain level of noise, the MI of the CS and ICS *shuffling* forms (2 and 3 bits) merge. It occurs with large noise levels as the conditional distributions merge to form one centered quasi-Gaussian distribution while suppressing minor (local) distributions changes. Intuitively, for example, in the 3-bit case, the multi-dimensional means of these "merged distributions" will be distinct per $\mathsf{W}(i)$.

For concreteness, an exemplary noise level that was measured in this work (as described in Section 5) is shown on the figure. Two quite important general observations relate to item 2 in the list above: As m increases (eg, $m = 8$ for the AES Sbox), the informativeness of the UnP design leakage, as compared with the *shuffled* designs, increase. For realistic designs, which target MI levels below 10^{-1} the (quite different) *shuffling* approaches tend to be equally informative; using simple words, the ICS design amplifies noise like the CS design with high noise levels.

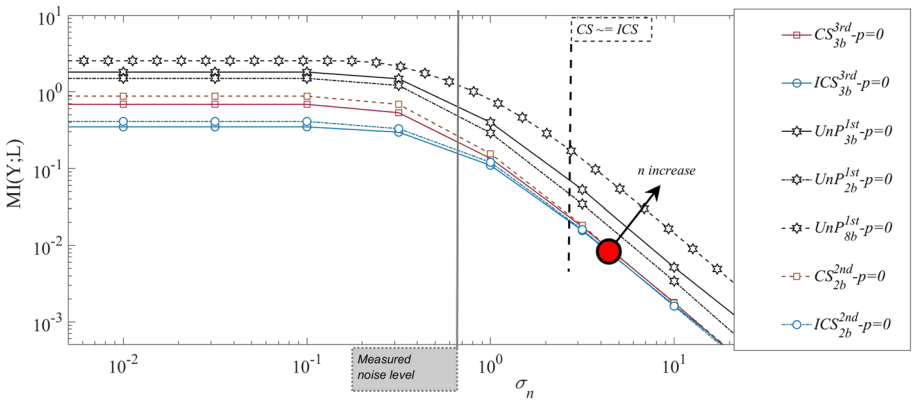
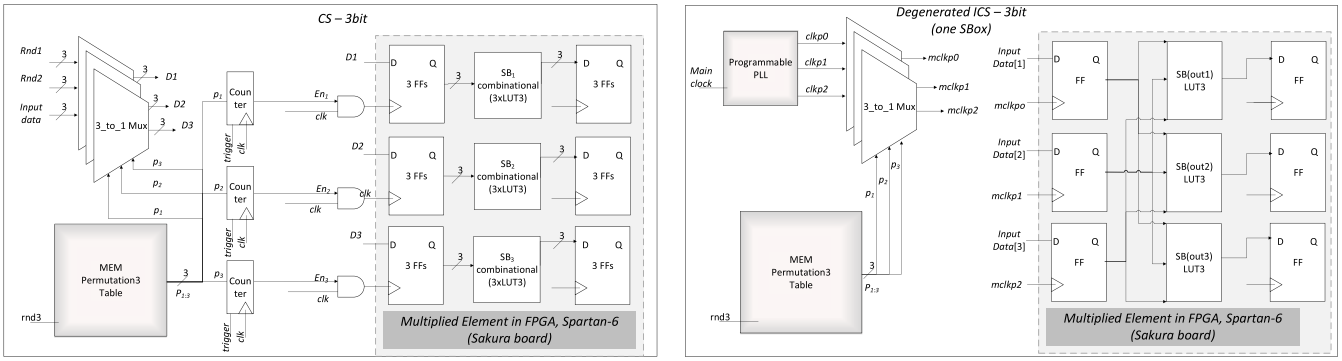


FIGURE 6 Mutual-information $\mathrm{MI}(\mathrm{Y};\boldsymbol{L})$ vs σ_n, cycle shuffling (CS) and intra-cycle shuffling (ICS) designs for 2- and 3-bits and UnP design of 2- to 8-bit $p = 0, \boldsymbol{L} = \mathsf{W}(y)$ [Colour figure can be viewed at wileyonlinelibrary.com]



(A) (B)

FIGURE 7 (A) 3-bit cycle shuffling (CS) design (3Sboxes, three cycle shuffling) scheme and (B) 3-bit intra-cycle shuffling (ICS) (1Sbox) design scheme. Implemented on a SAKURA-G board over a Xilinx FPGA (Spartan-6) [Colour figure can be viewed at wileyonlinelibrary.com]

In turn, these general observations underline the ICS design efficiency: from the implementation perspective (area, performance, energy, etc), due to the efficient utilization of algorithmic noise and due to the algorithm-independence properties (as discussed in details in Section 5).

3.4 | FPGA experiments

In this subsection, we evaluate the leakage distribution of physical implementations (FPGA-based). We then confirm the simulated leakages. The main message of this subsection is that the ICS design is superior in medium noise levels, whereas both the shuffling architectures (CS and ICS) have similar benefits with high noise environments.

3.5 | Designs and intuition

The 3-bit exemplary CS design described in Subsection 3.2 was specified in HDL and implemented on a Xilinx Spartan-6 FPGA. The architecture (schematically illustrated in Figure 7A) embeds three identical Sboxes (three output bits each)[††] with an input and output sampling layer (flip-flops). A programmable memory (within the FPGA) contains a table of the possible 3-bit permutations (of three elements); in each clock cycle, one of these permutations is chosen and assigned to three MUXes. The three MUXes then select one permutation of the *true* input and two more *random* inputs (all asserted through a UART interface to the FPGA). The three Sboxes and the input and output sampling layers instantiated occupied 15 LookUp tables.

The simplified 3-bit ICS design described in Subsection 3.2 (Figure 5B) was emulated by a dedicated architecture (denoted as *bit-ICS*) specified in HDL and implemented on the same Xilinx Spartan-6 FPGA. The architecture (schematically illustrated in Figure 7B) embeds one Sbox (similar to the one in the previous paragraph) with an input and output sampling layer (flip-flops). To mimic the original ICS design, each of the inputs and outputs was assigned with a different clock signal. As before, the programmable memory contains a table of the possible 3-bit permutations. In each clock cycle, one of these permutations is chosen and assigned to three MUXes. Those MUXes then select one permutation of the set of three clock phases. These clock phases are generated by the programmable phase-locked loop (PLL) existing in the FPGA architecture. Each bit of the 3-bits input is assigned to one of the FFs inputs, which in turn is sampled by one of the permuted sets of clock phases. To do so, each of the three FFs is implemented utilizing a full FPGA tile (with an independent clock signal).

3.6 | Validating the modeled distributions

In this subsection, we examine the measured leakage probability distribution of the multi-dimensional leakages ($\boldsymbol{L} = \{L_{t_1}, L_{t_2}, L_{t_3}\}$) as defined in Subsection 3.2, following the architecture from the previous subsection.

[††]The truth-table of the 3-bit Sbox{in,out} tuples is {000, 010}, {001, 110}, {010, 001}, {011, 000}, {100, 101}, {101, 111}, {110, 100}, and {111, 011}.

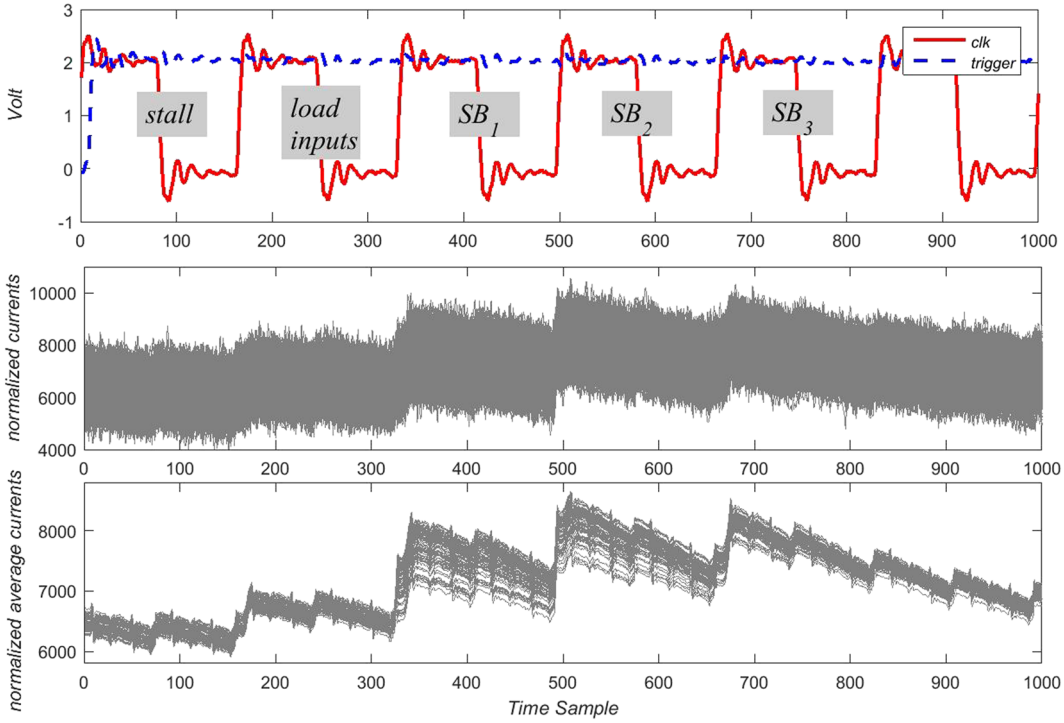


FIGURE 8 Exemplary current waveform of a cycle shuffling (CS) 3-bit (three cycle shuffling) design measured over a Xilinx Spartan-VI FPGA. (A) clock and trigger signals, (B) normalized currents, and (C) normalized and average currents [Colour figure can be viewed at wileyonlinelibrary.com]

Figure 8 shows exemplary measurements from the 3-bit CS design. The upper waveform in Figure 8 demonstrates the measured trigger (sent from the FPGA to the oscilloscope) and the logic-clock.[‡‡] The middle plot shows normalized currents (with amplitudes, which are relative to the oscilloscope vertical tuning). The lower plot shows the currents waveform after averaging of 100 K traces. The computation occurs (as labeled on the plot) in the third to fifth clock cycles where the first and second cycles are used to stall the measurement from the trigger signal (which induce substantial noise) and to load fresh inputs.

After filtering unwanted frequency components (from the measured currents), removing the measurement equipment temperature drift and averaging, a tuple of three time samples (POIs) was chosen, one per clock cycle, L_{t_1}, L_{t_2}, and L_{t_3}. These leakages (sets of three samples per computation) were utilized to draw the current distribution means. One mean leakage value exists per *state* which is composed of y_i state, permutation state, and random inputs to the two algorithmic elements. Clearly, in our settings, all the inputs are controlled and these (so called) random inputs are programmable to capture the complete probability distribution. That is, we assert each state multiple times for averaging purposes and record the leakage. The complete *states*-space was characterized. In the following, we detail the outcome distributions.

We first analyze the measured leakages of the 3-bit CS design with $p = 2$ from Figure 5C which exhibited quite a complex conditional distribution (but distinct per $\mathsf{W}(y)$ states). For the 3-bit ICS design with $p = 2$, the conditional leakages overlap substantially (as illustrated in Figure 5D). Again, a visual separation of the conditional distributions is not easy. Therefore, a visually manageable example for the leakage distribution with $p = 0$ is shown (as the one simulated and illustrated in Figure 5B).

Figure 9 shows the measured three-dimensional distribution of the CS design. On the upper line of sub-figures, each axis represent one dimension of the leakage (L_{t_1}, L_{t_2}, and L_{t_3}). This three-dimensional distribution is rotated (three upper scatter plots) to illustrate how close it is from the simulated one (Subsection 3.2). On the figure, each point represents a leakage sample (after averaging). Each color represents a different hamming weight $\mathsf{W}(\cdot)$ of the input y_i. Below the upper three-dimensional plots, two-dimensional leakage distribution histograms are shown for $\mathsf{W}(\cdot)$ of $\{0, \ldots, 3\}$ (from top to bottom). For each histogram column, the direction of the two-dimensional projection is indicated on the upper scatter

[‡‡]They are not synchronized here as they were buffered within the field-programmable gate array (FPGA) prior to be sent to the pins-header where they were measured

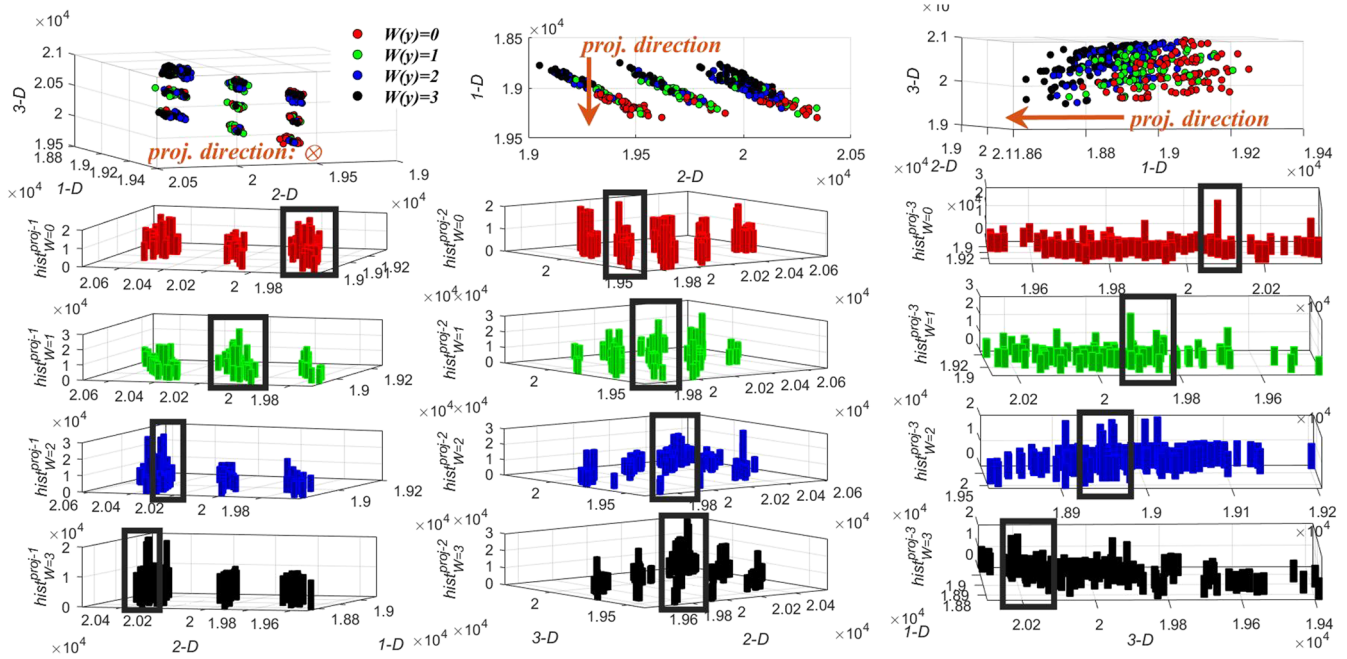


FIGURE 9 Multidimensional distributions of a 3-bit CS (3 cycle shuffling) measured on a SAKURA-G board utilizing Xilinx Spartan-6 field-programmable gate array (FPGA). The distribution is shown in three different angles and the marginal (projection) histograms of different $\mathsf{W}(y) = i, i \in \{0, .., 3\}$ [Colour figure can be viewed at wileyonlinelibrary.com]

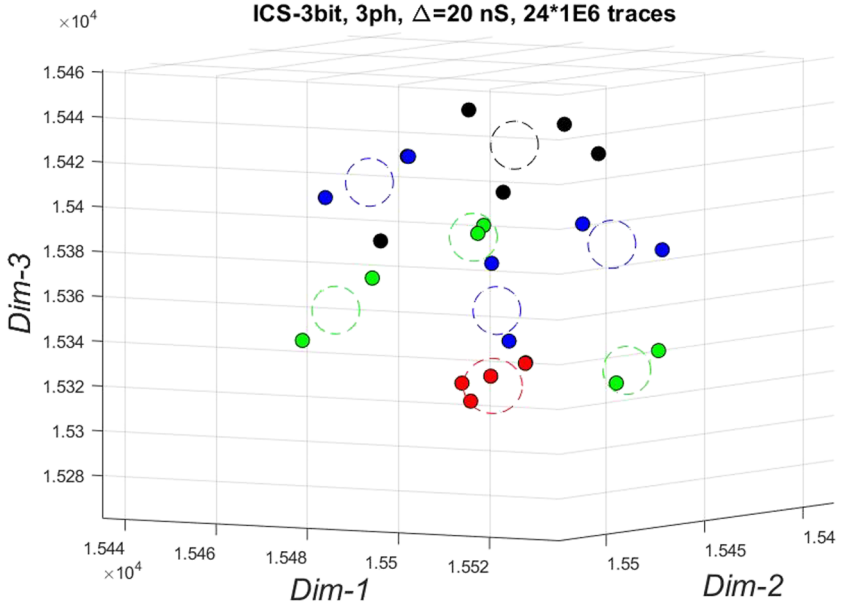


FIGURE 10 Multidimensional distributions of a 3-bit bit-intra-cycle shuffling (ICS) (bit shuffling) implementation. Red, green, blue, and black points correspond to $\mathsf{W}(y) = 0, 1, 2$, and 3, correspondingly [Colour figure can be viewed at wileyonlinelibrary.com]

plot. Within these two-dimensional histograms, we place black squares to highlight the centers of the distributions. A close examination of the histograms emphasizes the close match to the simulated leakages. Several key features are as follows:

- *Clusters:* while looking at the modeled marginal distribution of the CS design (Figure 5C), a two-dimensional (4 by 4) uniformly spaced grid of means is expected, whereas, in practice (in the upper left distribution of Figure 9), it is closer to a 3 by 3 grid of means. This is because the leakage does not completely follow the $\mathsf{W}(\cdot)$ leakage model: as the $\mathsf{W}(\cdot)$ increase, the distance between leakages decreases (according to a non linear relation). In our case, the groups of $\mathsf{W}(\cdot) = 2$ and 3 start to merge and appear more like one cluster.
- *Dependence:* as shown in the upper middle distribution, some dependence exists between dimensions. This is due to the large time constant of the power network (of our large FPGA-based setup), relatively to the clock period. In fact,

the level of dependence could be tuned/eliminated on an ASIC platform by design. For example, note the distinct independence of the current peaks of the ICS design in Figure 11B (adapted from Levi et al[6]).

Figure 10 shows the measured three-dimensional distribution of the bit-ICS design. On the figure, each point represents a leakage sample after averaging of $1 \cdot 10^6$ traces, totaling to $24 \cdot 10^6$ collected traces. The phase difference between the three clocks, in this example, is $\Delta = 20\ ns$ and the clock frequency is $4\ MHz$. The reduced signal of the bit-ICS design is evident in the number of traces which were needed to be collected in order to reduce the noise (as compared with the CS design). Each color represents a different hamming weight $\mathsf{W}(\cdot)$ of the input y_i. For example, for $\mathsf{W}(y_i) = 0$, it is evident that different permutations are grouped in one cluster. For all the other $\mathsf{W}(y_i)$ states, the mean leakages clusters (marked with dashed circles) match the simulated distribution in Figure 5B quite nicely.

4 | CONCRETE ADVERSARIAL LIMITATION AND ASSUMPTIONS

In this subsection, we elaborate on several physical limitations that a concrete adversary may encounter, such as measurement equipment constraints, in/dependence between leakage samples, and integrated noise. In turn, these factors lead to the noisy and reduced-dimensionality leakages that an adversary will face in real settings. We then show the substantial information loss in the leakage of these practical scenarios.

4.1 | Analog BW limitations: Reduced dimensionality

The analysis of high-dimensional leakages is a complex task. Nevertheless, in order to be able to perform it, an adversary needs to first acquire the leakages of all those dimensions. On FPGA platforms, this might not be a real constraint as the speed of these platforms is quite limited. Figure 11A shows current waveforms of an FPGA device operated with a 12 MHz clock. As shown, the different computations cycles (as well as the different dimensions of the leakage, separated by ~83 *ns*) are easily sampled. However, in many settings, such visibility of the leakages might be a concrete limitation. For example, on ASICs platforms, it is possible to design circuits to operate at 1 to 10 GHz which induce time separation of 1 to 0.1 *ns* between the leakage dimensions (see for example, Figure 11B adapted from Levi et al[6]). In turn, such high-speed designs imply that measurement instruments should comply with very high analog bandwidth (BW), as also discussed in Bellizia et al.[19]

In practice, acquisition systems are quite limited due to physical limitations. There exists a clear trade-off for measurement devices between analog BW and accuracy. Perhaps the most-limiting part of the system is the analog-to-digital (ADC) converter.[20] State-of-the-art devices can reach Analog BW of $1\,GHz$; however, it comes with serious limitations in terms of (amplitude) resolution and vice versa (as shown in Figure 11C for a set of low-to-high grade oscilloscopes).

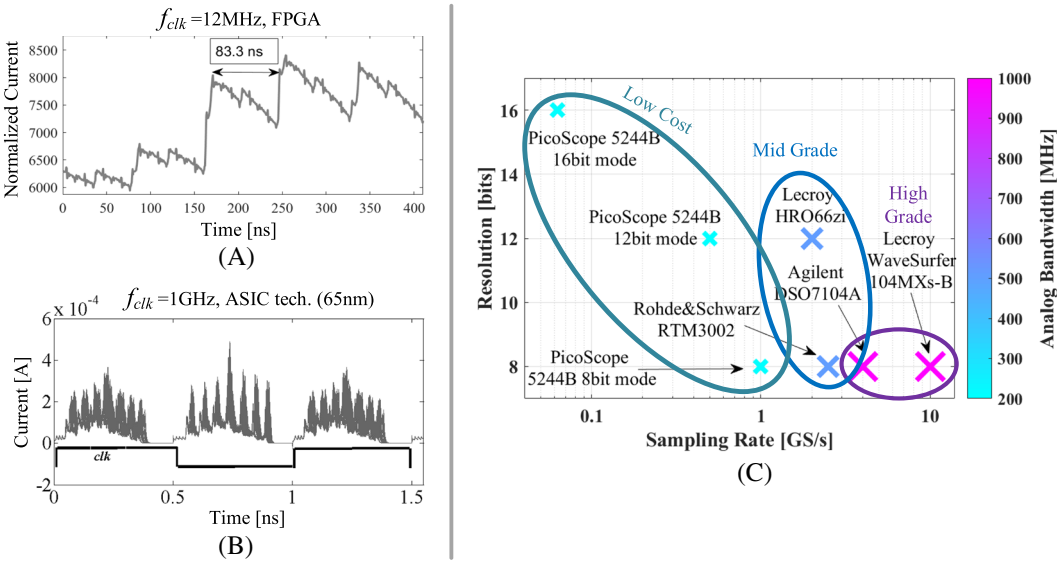


FIGURE 11 (A) Field-programmable gate array (FPGA) platform, 12 -MHz clock cycle, (B) ASIC platform, and 1 -GHz clock cycle (C) acquisition systems limitations [Colour figure can be viewed at wileyonlinelibrary.com]

Low-cost acquisition devices typically trade-off analog BW with resolution. The reasons for this behavior is well known[20]: fast sample rates (which are proportional to the analog BW) imply that the ADC should compute fast, and on the contrary, if the resolution is high, the computation is slow.[§§] In practice, and especially for the ICS case, reduced dimensionality could therefore be a concrete limitation. Therefore, the analysis in the following sections will also focus on reduced dimensionality.

4.2 | Physical filters, independence, and noise integration

A matter which relates to high-speed designs (and analog BW limitations) is leakage-samples in/dependence, which goes hand-in-hand with (physical) integrated noise. These points are discussed in this subsection, which motivates their analysis in the following sections.

We start with an adversarial view of the CS design and follow with the same view of the ICS design. We assume the adversary can control the device-under-attack (DUA) clock frequency, as illustrated in Figure 12A. The figure schematically captures a simplified leakage waveform from a CS design. In this case, an adversary can reduce the clock frequency as needed to induce independent leakages (ie, distinct leakage dimensions). A common approach against shuffling architectures is to integrate the leakage over time.[15] However, in order to reduce the level of the (also) integrated noise optimally, the adversary will integrate the leakages around the POIs. Taking into account the clock jitter, noise, and uncertainty of the POIs, modeled by a time uncertainty span, Δ, an adversary will perform a piece-wise integration (p.w-int) as illustrated in the figure. This integration (or sum) concludes in

$$L^{p.w-int} = \sum_{i=1}^{m} L_i + m \cdot \int_0^1 n(t)dt, \tag{7}$$

where $n(t)$ is the noise random-process and m is the number of integrated operations (or dimensions). In turn, this simplified abstraction implies that the informativeness of the leakage is quite equivalent to an unprotected parallel design with $p = m$ (if 1 is small enough). Note, however, that the adversary is still facing the following:

1. Integration (or summation) of noise samples. In fact, depending on the uncertainty, the physical-noise and the duty-cycle of the device, the noise might increase substantially. Figure 13 shows the physical noise (N_p) and the algorithmic-noise (S) of an AES core over an 8-bit data-path on the top and middle sub-figures. The lower sub-figure shows the integrated noise of both elements over the number of time samples. It is shown that noise accumulation by integration is substantial.
2. Over shuffling architectures, which incorporate a certain amount of algorithmic elements, integration may be a very bad choice for the adversary, as it also integrates algorithmic noise.

Concretely, the first point was already well demonstrated in Veyrat-Charvillon et al[15] and discussed in Mangard et al.[12] It was shown that, whereas, in simulated settings, the summation of leakages is informative, on real hardware (micro-controller), an attack by integration actually failed in all examined scenarios (the attack success rate stays on about zero, while increasing the number of traces). Therefore, multidimensional analysis is preferred in such a setting. Nevertheless, the CS design provides the adversary with a concrete knob to isolate leakages of different dimensions which is a concern.

Next, we elaborate on the strengths of the ICS design from the adversarial point-of-view as illustrated in Figure 12B. In the ICS architecture, an adversary cannot tamper with the time separation between the leakage dimensions. This is due to the internally and locally generated clock phases (by physical delay elements), the time separation between dimensions is denoted here by δ. Therefore, the internal-leakage-signal (prior to physical filtering) is a very high-frequency signal (the information lies in very high-frequencies). Real physical setting and measurements environment adds parasitic elements (resistive and capacitive) to the underling computation logic. Several examples are the device own power-grid, integrated-circuit pads, wire-bonds, package, measurement probes, and sensing amplifiers. These, de facto, realize an inherent physical filter, as discussed in Mangard et al[12] p.58 and 59 and in Kamel et al.[21] The effect of such filters is

[§§]Even when interleaved ADC are used and complex signal processing algorithms are performed, there exists physical limitations which prohibit sufficiently high analog Band-Width to follow tens and hundreds of pico-seconds changes

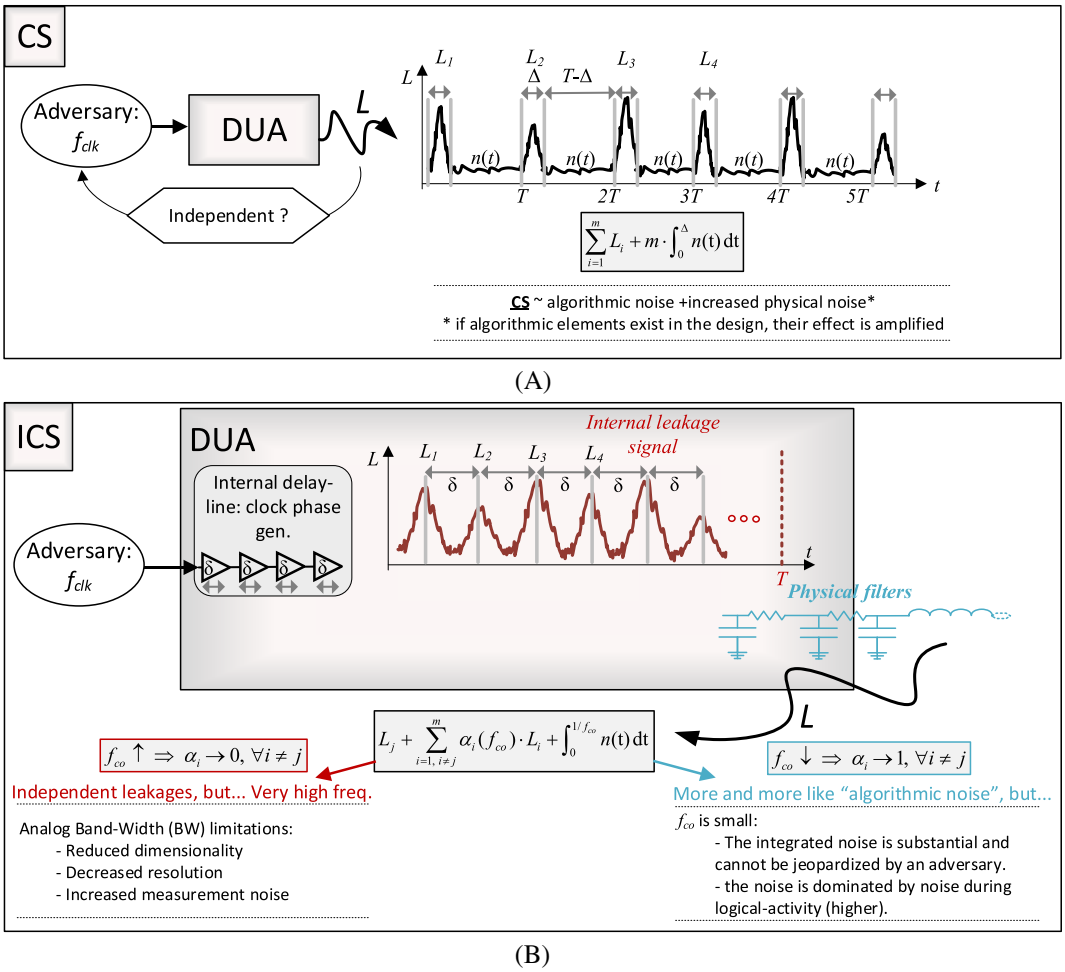


FIGURE 12 An adversary view on the (A) cycle shuffling (CS) design and (B) intra-cycle shuffling (ICS) design [Colour figure can be viewed at wileyonlinelibrary.com]

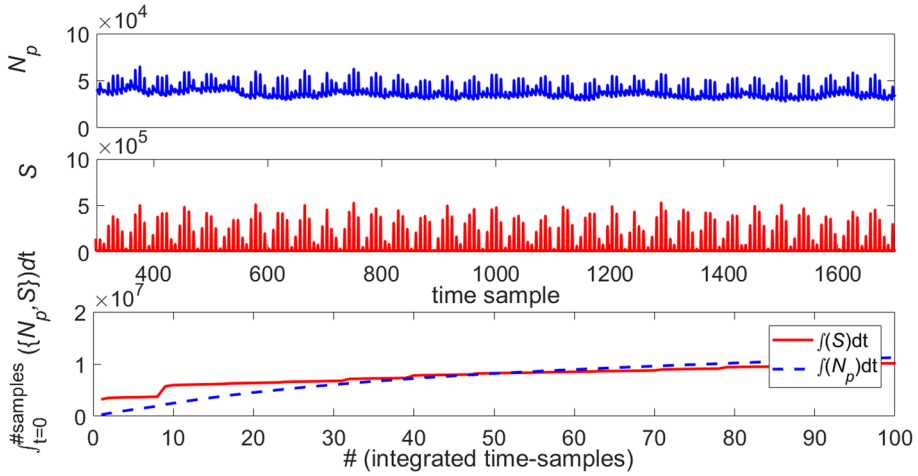


FIGURE 13 Measurement results of an AES core over an 8-bit data-path showing non-overlapping leakages (A) N_p, (B) S, and (C) the integrated noise elements vs the number of samples [Colour figure can be viewed at wileyonlinelibrary.com]

very similar to integration, which can be modeled by a weighted sum of leakages and noise integration, as illustrated in Figure 12B.

The physical filter is typically characterized by a cut-off frequency (f_{co}) and the coefficients of the filter (α_i) depend on that frequency as well as the integration duration: $L_j = \sum_{i=1, i\neq j}^{m} \alpha_i(f_{co}) \cdot L_i + \int_0^{1/f_{co}} n(t)dt$. If the induced filter is dominant, then the leakage samples will become dependent. The more filtering, the higher will the (integrated) noise become, as illustrated in Figure 13. By contrast, if the filter is not dominant, an adversary will be able to observe a leakage which

is closer to the internal leakage signal. In the latter case, however, the problem is reverted to analog BW limitations. We examine these two extreme cases. On the one hand, a nondominant physical filter implies a large f_{co}. This in turn, leads to negligible α_i coefficients for all L_i, $i \neq j$ as denoted on Figure 12B. In this case, the specific leakages will be independent. Due to analog BW limitations (as discussed above), the adversary will face reduced dimensionality, decreased amplitude resolution, and increased measurement noise. On the other hand, a dominant physical filter implies a small f_{co}. In the extreme case (i.e., taking α_i=1), the situation will become closer to an algorithmic-noise scenario, but due to the small f_{co}, the noise integration will dominate the leakage. It is important to emphasize that in this case (of very high-frequency logic), the integration takes place when the circuits' switching-activity is very high. As shown in Section 2, the noise element in this region is more substantial than in steady states. In practice, a circuit can be designed by controlling δ and the physical filter (e.g., the power network grid of the device) to meet these scenarios. Between these two extreme cases, an adversary will typically face a mixture of reduced dimensionality, decreased amplitude resolution, increased measurement noise, or high-integration noise.

To conclude, physical limitations, such as analog BW constrains, inherent filters, and post-processing filters (such as an integration by an adversary), ties noise-integration and reduced dimensionality together. In Appendix A, we provide more examples for these scenarios, leakage distributions of reduces dimensions, along with an IT examination. In the next section, we analyze a concrete scenario of a full encryption core taking this factors into account. A detailed analysis of the trade-off between these factors and the information content of the leakage is outside of the scope of this manuscript; however, it is an important direction for future work.

5 | 32/128-BIT AES IMPLEMENTATIONS CASE-STUDIES

In this section, we finally extend the analysis from above to several practical cases for designers and real-life security evaluations. Namely, we examine several architectures of the AES algorithm over different data-path widths and possible shuffling mechanisms. As described in the introduction, we would like to harness both the width of the data-path to generate large algorithmic-noise and time domain manipulation (shuffling).

It is important to highlight that the number of dimensions in the ICS architecture is independent of the underling algorithm (it shuffles bits and not operations). In practice, we next examine a case of eight dimensions while shuffling bits within bytes (eg, each set of eight bits of an AES Sbox is shuffled).

We examine two architectures for the logical and control parts of our test-cases as follows:

1. 128-bit: as shown in Figure 14A, a full parallel 128-bit data-path width, which enjoys eight algorithmic elements (in the case of ICS).[¶¶] In this single-pipeline stage (fully rolled implementation), in order to meet timing constraints with our multi-clock signals architecture, the sampling units (denoted by R1 on the figure) are as follows: each memory element is in fact a master-slave configuration of flip-flops (where each of which is in fact a master-slave latch). The master flop is triggered by the input clock signal and the slave flop by the inverted clock. This ensures a correct by design timing scheme (which clearly has a timing cost). On the right side of the figure, a waveform diagram illustrates the signals behavior from the input X of the master flop through the intermediate registers signal X_{int} sampled by $mclkp[7:0]$ to the master-slave register output, O sampled by $\overline{mclkp}[7:0]$. The maximum possible combinational delay is $T/2-(n-1)\delta-t_{cq}-t_{su}$, as denoted on the figure.[##]
2. 32-bit: as shown in Figure 14B, a 32-bit data-path width, in which the number of algorithmic elements is three. This architecture is rolled (four iterations) with two pipeline stages. There exist two 128-bit state registers (one intermediate and one final). To manage storing and updating only one different word (32 bits) in each round the state-machine induces two control signals, ie, *index1* and *index2*. These serve as inputs to a switch-unit (Figure 14B) which directs the 32-bit inputs to the correct word to be updated. In order to save energy and reduce clock-signals fan-out, the switch-unit also generates four enable signals, whereas in each cycle, only one of them is set to one (corresponding to one out of the four words on the 128-bit state register). This could also be achieved with clock-gating mechanisms.

For each of these architectures, we consider two mechanisms to generate the required clock-phases for ICS. The two options represent quite different approaches. One is memory-intensive (permutation memory-based) and the other relies on combinational logic to generate on-line permutations of the clock phases required. It is important to note that there

[¶¶] Assuming here the adversary's guessing power is smaller than 2^{16}.

[##] Where, T is the clock period, n the number of clock phases, δ the delay difference between phases, and t_{cq} and t_{su} are the clock-to-q and setup delay of the registers.

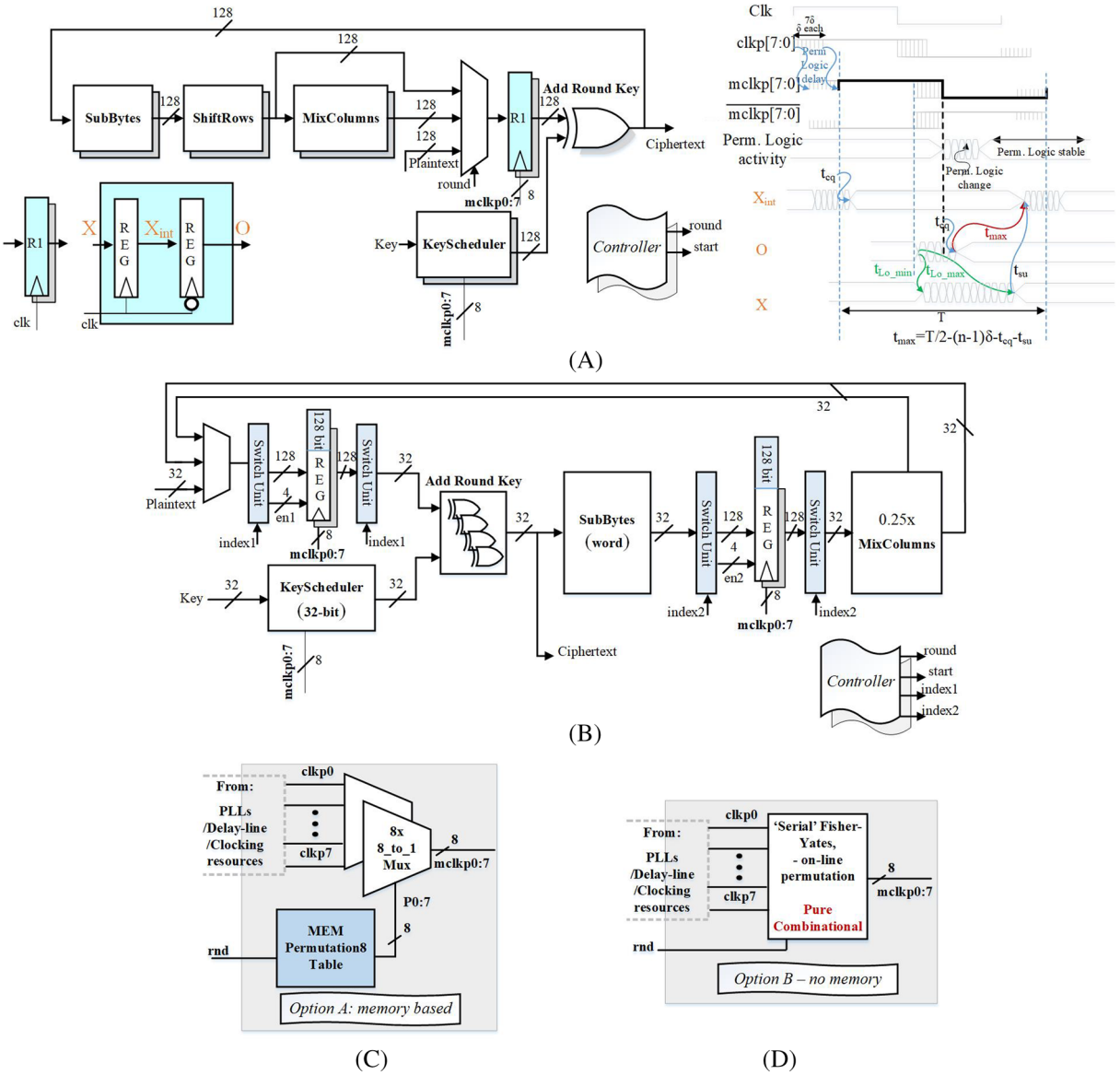


FIGURE 14 (A) 128-bit DP AES with corresponding waveform diagram and (B) 32-bit DP AES. Implemented on Xilinx field-programmable gate array (FPGA) (Spartan-6) (C) MEM base permutation gen. (D) On-Line permutation gen [Colour figure can be viewed at wileyonlinelibrary.com]

is quite a large optimization space in between and it is possible to substantially reduce the requirements of each of the solutions (memory or number of random bits required). Nevertheless, as discussed next, the on-line phase permutation generation is quite efficient on all aspects, ie, it requires negligible combinational logic area, no performance overheads and almost zero memory.

1. Permutation MEM-based: as shown in Figure 14C, after generating eight shifted clock phases, $\{clkp0, .., clkp7\}$ in this case generated with a LUT5-based tapped delay line, delivering a time separation δ between phases of approximately $2\,ns$. We utilize a random permutation of the sequence $\{0, .., 7\}$ (each value represented by 3 bits, summing to 24 bits) to randomly shuffle the clocks yielding $\{mclkp0, .., mclkp7\}$, which are used in our design. A random sequence of 8 bits in each cycle is required or alternatively approaches like random-start index can be utilized.[3]

TABLE 1 Area utilization and performance comparison of the 32- and 128-bit data-path AES128 with embedded ICS architectures

	Slice	LUTs	Regs	BRAM	f_{max} [MHz]	#cycles (Perf.)
AES128 32bit DP						
UnP.	349	898	532	0	99.1	90
ICS-FY	606	1024	530	0	60	
ICS-MEM	629	1160	537	56	60	
AES128 128bit DP						
UnP.	1249	2428	389	0	37.1	12
ICS-FY	1668	2260	517	0	18	
ICS-MEM	1802	2377	524	56	18	

Note. Unprotected (UnP) architectures, intra-cycle shuffling (ICS) with memory-based (MEM), and on-line Fisher-Yates (FY) permutation generation architectures.

The cost is eight 8-bit MUXes and the memory of a permutation. Below, we show the cost in terms of memory size for all possible permutations. Clearly, many other solutions exist which trade uniformity for memory size. Here, we aim to show the maximum memory cost of such an approach.

2. On-line Permutation gen.: as shown in Figure 14D, after generating the eight clock phases, we apply the Fisher-Yates algorithm[22] to generate random permutations. With FPGAs (and ASICs) the algorithm, which typically requires n cycles (where n is the number of clock phases or required elements to permute), can be unrolled to a pure combinational circuit to achieve a throughput of one permutation/cycle. Doing so, this implementation completely eliminates memory requirement (besides having a negligible number of register needed to store more random bits), requires only quite small area to implement the combinational logic, and does not imply any performance overhead, which we consider as our preferred method.

The four architectures were tested while embedding the AES 8-bit Sboxes represented by (a) a finite tower Galois-field (GF), $GF((2^4)^2)$, and (b) by a standard look-up table. In essence, for FPGAs, in many cases, the latter is quite efficient as the synthesis to LUTs with a specific number of inputs (eg, six) will induce area-overheads when subjected to logical constraints of a specific GF (eg, with field elements of four). In ASICs, the situation is often opposite.

In terms of hardware and architectural complexity, these two designs greatly differ, as follows: whereas the 128-bit data-path design is very fast and dynamic-energy efficient, we can estimate that it will cost roughly three to four times more than the 32-bit one in terms of area and in static/leakage power. The 32-bit data-path design requires to store intermediate values and to perform computations on *chunks*, and in our case, it is also pipelined with two stages. Therefore, we can expect a performance degradation of a factor of $4 \cdot 2 = 8$. Table 1 lists the area components (FPGA-LUTs, -Registers, and -Slices) of these two architectures. The table also lists the maximum frequency and performances (in units of #clock cycles). Due to the proposed clocking scheme, these values are identical for the unprotected (UnP) designs with only negligible changes due to larger setup- and hold-time constraints needed for the 32-bit architecture case to ensure correct operation (hold-time of $(n-1)\delta$ which only impacts *min-delay* constraints). For the 128-bit architecture case, the max-frequency is roughly X2 smaller due to the one stage pipeline and the double master-slave register architecture. These constraints are set to include margins on the maximum phase-shift of the set of clocks, which only affect the first and last rounds of the encryption (thus negligible). An inspection of the values in the table indicates that a significant area savings can be made moving from the 128-bit to the 32-bit architecture. Though, not the theoretically expected ×4 which is due to the large (relative) size of storing elements as compared with combinational logic. However, securing the proposed shuffled implementations is extremely cheap in terms of area: between as little as 15% and 25% area overhead as compared with the UnP designs. This overhead is more evident in the 128-bit architecture, mainly due to the added registers (ie, *R1* in Figure 14A). Performance results meets the expected values.

In the following, we advance with a security analysis of these architectures with the information theoretic tools and algorithmic-noise models developed in this work. However, to evaluate noise levels needed for the analysis, we have first implemented a complete unprotected AES engine on the Xilinx Spartan-6 FPGA, embedded in the SAKURA-G board, with different architectures: vary in the number of algorithmic elements, $p \in \{0, 1, \ldots, 8\}$. Then, following current traces acquisition, their averaging, etc, we have computed the underling measured noise level of these architectures. The baseline noise level ($p = 0$) is $\sigma_n = 6 \cdot 10^{-1}$ and marked by a vertical dashed black line on Figure 15. The computed noise levels of other values of p, which correspond to different number of algorithmic elements, were computed.

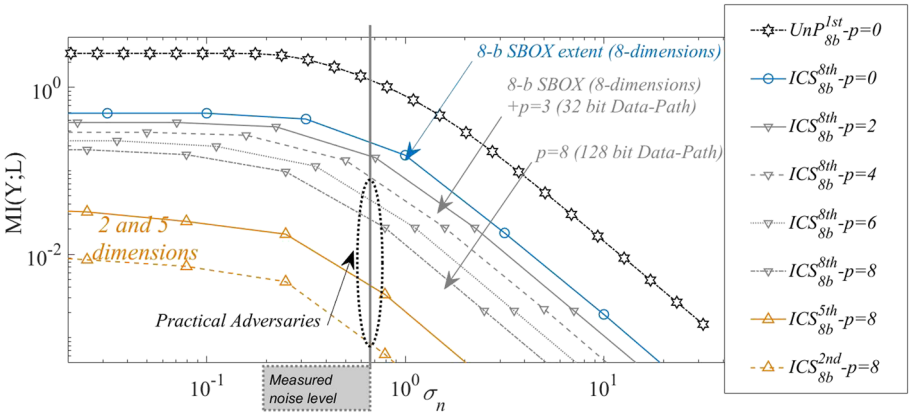


FIGURE 15
Mutual-information (MI)(Y; $\boldsymbol{L}$) vs σ_n, of eight-dimensions intra-cycle shuffling (ICS) design embedded within different AES architectures and an un-protected (UnP) 8-bit architecture [Colour figure can be viewed at wileyonlinelibrary.com]

The analysis from Section 3.2 was projected to the case of eight dimensions. Figure 15 shows the MI curve of a serial ICS design ($p = 0$) with eight dimensions (shuffling 8-bits of, eg, Sboxes) with a circle-marked blue line. The MI curves of designs with different $p's$ are also visible with gray lines (triangle marked) on the figure. The noise levels which were computed for the architectures discussed in the previous paragraph were utilized to verify the intersection points of these curves with the baseline noise level.

The limiting factors from Section 4.1 and computational complexity limitations may conclude in (a) leakages with less *effective*-dimensions and (b) less noise removal abilities (algorithmic and thermal noise). Taking these two challenges, the following analysis provides insights about the amount of information that a *bounded* adversary can extract.

Reduced dimensionality is a practical scenario (and may lead to concrete limitations for actual adversaries), especially in the case of an ASIC-based ICS design. The MI curves which correspond to an adversary which is able to collect information from five or two dimensions (out of eight) are shown in the figure with an orange line marked with upward triangles. It is computed by projecting the conditional leakage distributions from the 8-bit CS design (eight dimensions) to a lower-dimensionality space (marginal distributions).

It is evident from the results that utilizing both shuffling architectures and algorithmic noise elements (with large data-paths) can lead to substantial security gains. In addition, practical ASIC designs can also benefit from reduced dimensionality, which depend on the technology node and on the hardware designers (with the ICS design). For example, out of each byte, an adversary can learn maximum of $9 \cdot 10^{-2}$- or $2.5 \cdot 10^{-2}$-bits, for a 32- or 128-bit architecture, respectively. If the adversary can only get access (effectively) to five or two dimensions, he can learn $3.5 \cdot 10^{-3}$ or $1 \cdot 10^{-3}$ bits, respectively. Consequently, a divide-and-conquer adversary is able to extract maximum of $\{1.44, 4 \cdot 10^{-1}, 5.6 \cdot 10^{-2}, 1.6 \cdot 10^{-2}\}$-bits for the 32-bit and 128-bit architectures with 5-dimensions and 3-dimensions, respectively. For reference, the MI of an unprotected 8-bit architecture is shown on the figure with a black line marked with stars. Under the same setting (and assuming a binomial distribution of $\mathsf{W}(\cdot)$ leakage) such design may leak 20 bits of information (where the maximum it will leak in a noiseless world is about 40 bits). An important conclusion of this security analysis is that it is possible to foster shuffling and algorithmic noise to emulate significant noise, reaching security levels which are typically only achieved with very high cost (eg, *masking*).

6 | CONCLUSIONS

In this paper, we analyze the applicability of utilizing both algorithmic noise and shuffling to foster higher security. The analysis starts with a detailed examination of the physical and algorithmic-noise sources to better understand their extent. We then show that conventional shuffling countermeasures (eg, CS) are limited and explore the applicability of a new hardware-oriented shuffling scheme (ie, ICS) for the first time. A detailed IT analysis and comparison between the two methods is performed and the analysis (and leakage distributions) is verified on an FPGA-based experimental environment. The performed analysis and the intuition that is built along the manuscript are then extended to practical (physical) scenarios. In these situations, an adversary is facing real-life physical filters (on-chip, on-board, etc) and measurement equipment limitations (in addition to practical computation and data complexity issues). These factors are then considered from the IT point-of-view. The tools, methods, and insights developed along the manuscript are then used to evaluate a real-life AES core with embedded ICS (with a 32- and 128-bit data-path width architectures). An IT analysis is performed taking into consideration different architectures, algorithmic-noise, reduced-dimensionality, and

the target device noise level (Xilinx Spartan-6 FPGA). The cost in terms of area and performance is detailed. It highlights the potential of the ICS architecture both in terms of security and in terms of hardware efficiency together with its operating simplicity (algorithm independence) as compared with standard shuffling architectures. The extension of the analysis of adversarial (physical) limitations initiated in this work is an important research scope. Similarly, the better quantification/bound of (sometimes hard to characterize) physical defaults and their impact on the time separation of the dimensions of an ICS-protected implementation, or the physical filter characteristics, are interesting directions for further research.

ACKNOWLEDGMENTS

François-Xavier Standaert is a senior associate researcher of the Belgian Fund for Scientific Research (FNRS-F.R.S.). This work has been funded in parts by the ERC project 724725 (acronym SWORD) and by the H2020 project 731591 (acronym REASSURE).

ORCID

Itamar Levi https://orcid.org/0000-0002-5591-5799
Davide Bellizia https://orcid.org/0000-0002-6947-4410

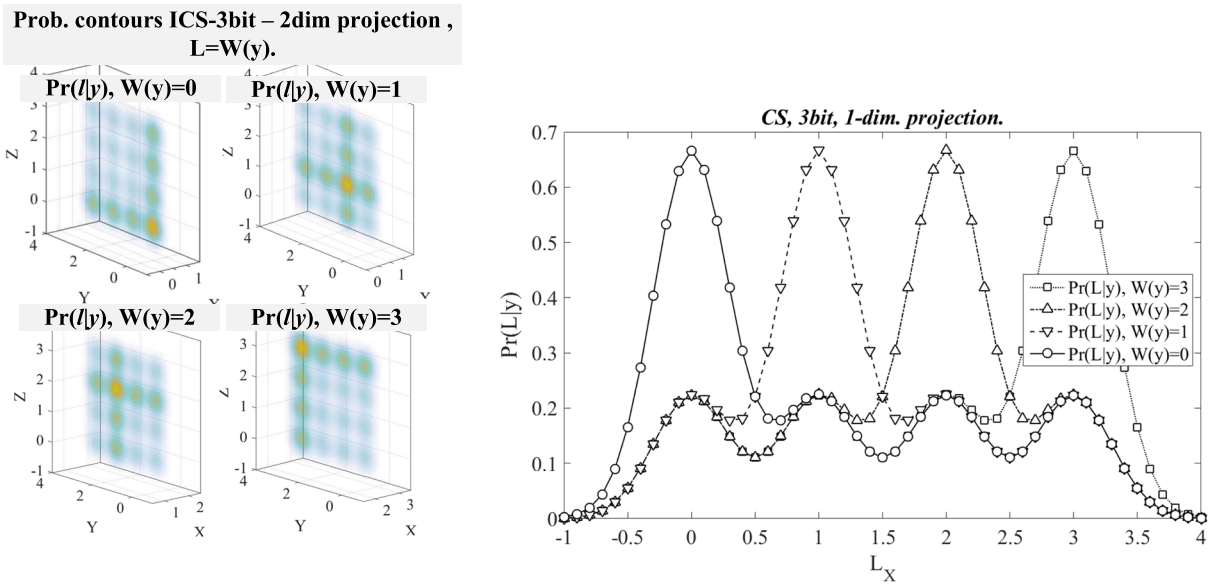
REFERENCES

1. Clavier C, Coron JS, Dabbous Nora. Differential power analysis in the presence of hardware countermeasures. In: Cryptographic Hardware and Embedded Systems–CHES 2000. Springer; 2000.
2. Tunstall M, Benoit O. Efficient use of random delays in embedded software. *WISTP*. 2007;4462:27-38.
3. Herbst C, Oswald E, Mangard S. An AES smart card implementation resistant to power analysis attacks. In: ACNS. Springer; 2006.
4. Rivain M, Prouff E, Doget J. Higher-order masking and shuffling for software implementations of block ciphers. In: CHES. Springer; 2009.
5. Papagiannopoulos K. Low randomness masking and shuffling: an evaluation using mutual information. *IACR Transactions on Cryptographic Hardware and Embedded Systems*. 2018:524-546.
6. Levi I, Fish A, Keren O. Low-cost pseudoasynchronous circuit design style with reduced exploitable side information. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*. 2017.
7. Bellizia D, Bongiovanni S, MonsurrÒ P, Scotti G, Trifiletti A. Univariate power analysis attacks exploiting static dissipation of nanometer CMOS VLSI circuits for cryptographic applications. *IEEE Transactions on Emerging Topics in Computing*. 2016;5(3):329-339.
8. Moos T, Moradi A, Richter B. Static power side-channel analysis of a threshold implementation prototype chip. In: Proceedings of the Conference on Design, Automation & Test in Europe. European Design and Automation Association; 2017.
9. Alioto M, Djukanovic M, Scotti G, Trifiletti A. Effectiveness of leakage power analysis attacks on DPA-resistant logic styles under process variations. *IEEE Transactions on Circuits and Systems I: Regular Papers*. 2014;61(2):429-442.
10. Moradi A. Side-channel leakage through static power. In: International Workshop on Cryptographic Hardware and Embedded Systems. Springer; 2014.
11. Del Pozo SM, Standaert FX, Kamel D, Moradi A. Side-channel attacks from static power: when should we care? In: Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition. EDA Consortium; 2015.
12. Mangard S, Oswald E, Popp T. *Power analysis attacks: revealing the secrets of smart cards*, Vol. 31: Springer Science & Business Media; 2008.
13. Levi I, Fish A, Keren O. CPA secured data-dependent delay-assignment methodology. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*. 2017;25(2):608-620.
14. Levi I, Keren O, Fish A. Data-dependent delays as a barrier against power attacks. *IEEE Transactions on Circuits and Systems I: Regular Papers*. 2015;62(8):2069-2078.
15. Veyrat-Charvillon N, Medwed M, Kerckhof S, Standaert FX. Shuffling against side-channel attacks: a comprehensive study with cautionary note. *Advances in Cryptology–ASIACRYPT 2012*. 2012:740-757.
16. Standaert FX, Malkin T, Yung M. A unified framework for the analysis of side-channel key recovery attacks. In: Eurocrypt. Springer; 2009.
17. Duc A, Faust S, Standaert FX. Making masking security proofs concrete-or how to evaluate the security of any leaking device. Springer; 2015.
18. Bronchain O, Hendrickx JM, Massart C, Olshevsky A, Standaert FX. Leakage certification revisited: bounding model errors in side-channel security evaluations. *IACR Cryptology ePrint Archive*. 2019;2019:132.
19. Bellizia D, Scotti G, Trifiletti A. TEL logic style as a countermeasure against side-channel attacks: secure cells library in 65nm CMOS and experimental results. *IEEE Transactions on Circuits and Systems I: Regular Papers*. 2018;99:1-11.
20. Ahmed I. *Pipelined ADC design and enhancement techniques*: Springer Science & Business Media; 2010.
21. Kamel D, Renauld M, Flandre D, Standaert FX. Understanding the limitations and improving the relevance of SPICE simulations in side-channel security evaluations. *Journal of Cryptographic Engineering*. 2014;4(3):187-195.
22. Fisher RA, Yates F. Statistical tables for biological, agricultural AAD medical research. *Statistical tables for biological, agricultural AAD medical research*. 1938.

How to cite this article: Levi I, Bellizia D, Standaert F-X. Beyond algorithmic noise or how to shuffle parallel implementations? *Int J Circ Theor Appl*. 2020;48:674-695. https://doi.org/10.1002/cta.2756

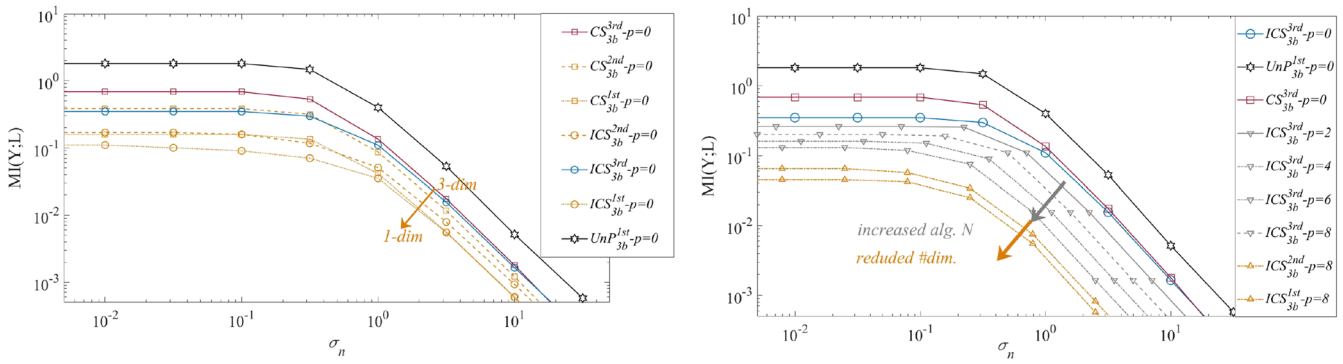
APPENDIX A: EXAMPLES OF REDUCED DIMENSIONALITY DISTRIBUTIONS AND ALGORITHMIC NOISE

The information-theoretic analysis from Section 3.2 is taken a step forward to evaluate the information-theoretic (IT) extent against a bounded adversary (one which is expected in realistic settings).



(A) (B)

FIGURE A1 Leakage distribution contours of cycle shuffling (CS) 3-bit, $p = 0, \boldsymbol{L} = \mathsf{W}(y)$ over (A) 2-*Dim* space and (B) 1-*Dim* space [Colour figure can be viewed at wileyonlinelibrary.com]



(A) (B)

FIGURE A2 Mutual-information (MI)(Y; $\boldsymbol{L}$) vs. σ_n, 3-bit cycle shuffling (CS) and intra-cycle shuffling (ICS), 3-bit UnP with $p = 0, \boldsymbol{L} = \mathsf{W}(y)$ and ICS with (A) reduced dimensionality (B) reduced dimensionality and algorithmic noise [Colour figure can be viewed at wileyonlinelibrary.com]

The limiting factors from above and computational complexity limitations may conclude in (a) leakages with less-*effective* dimensions and (b) less noise removal abilities (algorithmic and thermal noise). Taking these two challenges, the following analysis provides insights about the amount of information that a *bounded* adversary can extract.

Less Dimensions: Figure A1 illustrates the probability density contours (and probability density curve) of the 3-bit cycle shuffling (CS) design projected to a two-dimensional plane (resp. one-dimensional curve). In this case, the adversary can collect and process information from two (resp. one) dimension(s) only.

The marginal probability distributions of these settings were computed and the MI was evaluated as shown in Figure A2A. It is clear that when an adversary can process less dimensions, the amount of information he can extract rapidly reduces (similarly for the CS and intra-cycle shuffling [ICS] designs). As was discussed in the previous section, the MI of a full-dimensionality distribution is independent of the number of dimensions (number of bits in an ICS design) when the noise is large. That is, the full-dimensional MI converges to the same curve of the one of smaller m designs for large noise levels. However, while observing less dimensions, an adversary would gain substantially less information. This is due to the fact that as the dimensionality decreases, more collisions (between different conditional probabilities) exist in the marginal distributions (for both the CS and ICS designs). As shown in Figure A2A, across noise levels, the impact of reduced dimensionality is considerable.

Algorithmic noise: considering inherently parallel architectures (such as the ICS design), the algorithmic noise plays an important role (as discussed in Section 2). If the guessing-power of the adversary is not large enough, the noise can not be removed and the MI will decrease. Notably, as discussed above, the system can be designed in a fashion which makes the algorithmic-noise part hard to hinder (eg, first-order masking). Figure A2B shows (with gray pointed-down triangle markers) the MI an adversary can get with a varying number of noisy algorithmic elements (p ranging from 2 to 8) for the 3-bit ICS design. Clearly, the impact of both the algorithmic noise and reduced dimensionality is substantial, as demonstrated jointly.