Learning with Physical Noise or Errors

Dina Kamel[1], François-Xavier Standaert[1], Alexandre Duc[2], Denis Flandre[1], Francesco Berti[1]
[1] Université cath. de Louvain, Louvain-la-Neuve, Belgium. [2] HEIG-VD, Yverdon-les-Bains, Switzerland.

Abstract—Hard learning problems have recently attracted significant attention within the cryptographic community, both as a versatile assumption on which to build various protocols, and as a potentially sound basis for lightweight (possibly side-channel and fault resistant) implementations. Yet, in this second case, a recurrent drawback of primitives based on the Learning Parity with Noise and Learning With Errors problems is their additional randomness requirements to generate noise or errors. In parallel, the move towards nanoscale devices renders modern implementations increasingly prone to various types of errors. As a result, inexact computing has emerged as a new paradigm to efficiently deal with the challenges raised by such erroneous computations, and mitigate the cost and power consumption overheads they cause. In this paper, we show that these cryptographic and electronic challenges can actually be turned into new opportunities, and provide an elegant solution one to the other. That is, we show that inexact implementations of inner product computations lead to a natural way to define new Learning with Physical Noise or Error assumptions, paving the way to more efficient and physically secure implementations, with potential interest for securing emerging Internet of Things applications.

Index Terms—Learning Parity with Noise (LPN), side-channel analysis, fault attacks, Physically Unclonable Functions (PUFs).

1 Introduction

The Internet of Things (IoT) is a fast-emerging technology that aims at connecting "things" (e.g. objects that feature sensors and actuators) to a larger network via the Internet, enabling new applications such as smart homes, connected cars, smart grids, smart cities, e-health, ... In order to enable their secure deployment, one minimum feature for such interconnected devices is to be securely identified. Yet, their low-cost nature implies strong resource constraints in terms of cost, area, power consumption and energy. In this respect, and as the size of integrated circuits is shrinking due to technology scaling, more and more functionalities can be added to hardware devices. Therefore, the recent literature has shown that implementing cryptographic (e.g. challenge-response) protocols is now feasible in the IoT. Those protocols can rely on symmetric encryption based on the AES block ciphers [1], which are typically suitable in close systems due to key distribution challenges. Alternatively, Elliptic Curve Cryptography (ECC) engines have been proposed to alleviate this key distribution problem while still maintaining short key sizes [2]. Yet, while unprotected implementations of block ciphers or ECC can fulfill the IoT constraints, their protection against physical (e.g. side-channel or fault) attacks remains an important challenge, due to the large overheads they imply [3], [4].

Cryptographic background. The Learning Parity with Noise (LPN) and Learning With Errors (LWE) problems have recently found many applications in the design of provably secure cryptographic schemes: see [5], [6] for good surveys. LPN was first used in secret key identification protocols, starting with the proposal by Hopper and Blum [7]. This so-called HB scheme was then extended to HB$^+$ – that is secure in an active attack model – by Juels and Weis [8], and to HB$^\#$ – that is secure against man-in-the-middle attacks – by Gilbert et al. [9]. Both HB$^+$ and HB$^\#$ require three communication rounds. Kiltz et al. [10] next proposed an alternative identification scheme with only two rounds [10], while also describing Message Authentication Codes (MACs) constructions based on LPN. More recently, an efficient 2-round identification protocol (called LAPIN) was proposed by Heyse et al. [11], while LPN-based MAC constructions were revisited in [12]. In parallel, constructions of PRGs and one-way functions [13], but also secret-key encryption schemes [14] and public-key encryption schemes [15] were proposed. The LWE assumption turns out to be even more versatile than the LPN one (at the cost of more involved operations, making it less suitable for low-cost devices). For example, it allows the design of collision-resistant hash functions [16], identity-based encryption [17] or Fully Homomorphic Encryption (FHE) schemes [18]. Finally, the security of the LPN and LWE assumptions is analyzed by a number of independent works: see [19], [20] and [21], [22], [23], [24], [25] for early and more recent results in the field. In this respect, while tuning the security parameters of these assumptions remains a scope for research, it is widely believed that their underlying problems are hard.

Electronic background. The scaling of the CMOS technology, that is reflected in most present microelectronic devices, has been a permanent trend since integrated circuits appeared in the late 1950s. This trend is expressed by the famous Moore's law (with 14-nanometer technology in industrial production and 7-nanometer as a target within the next decade). Shrinking transistors' sizes is generally motivated by the need for increased performances and reduced cost and energy per operation. But when reaching the nanometer scale (defined by minimum transistor size below 100 nanometers), various side-effects also arise. First, the relative importance of so-called static currents increases, i.e. energy is consumed even if no computations are performed [26]. Second, device variability becomes significant, i.e. it becomes increasingly difficult to engineer identical chips [27]. Third, physical noise in transistors tends to grow, while the useful signal representing the data is reduced (due to supply voltage reduction), hence making the computational outcomes less and less deterministic [28].

Among these side-effects, the two first ones have already been considered in the context of cryptographic implementations, with contrasted intuitions. On the one hand, static leakages are mostly detrimental and designers usually try to minimize them, both for energy consumption and physical security reasons [29], [30], [31]. On the other hand, variability is indeed problematic to deal with from a performance viewpoint, but can also be turned into an asset. For example, it can make certain types of side-channel attacks more challenging [32], and is exploited in the design of Physically Unclonable Functions (PUFs) [33]. As for the noise issue, alternative logic design approaches have been explored for about 10 years in order to exploit it constructively in CMOS chips, based on so-called probabilistic [34] or inexact [35] computing. Such new paradigms are claimed to be more energy efficient. But to the best of our knowledge, they have not been used for cryptographic applications based on hard learning problems as we suggest in the following.

Contributions. From an implementation point-of-view, the main potential advantages of LPN-based primitives are their simple structure making them suitable for low-cost devices, and their algebraic properties making them easier to protect against side-channel attacks thanks to masking [36], and harder to attack with faults [37]. Unfortunately, recent results have shown that these advantages do not fully translate into concrete improvements when looking at the full (implementation cost and physical security) picture [38]. Taking the example of LAPIN, it remains that the chip involved in the authentication protocol has to include a random number generator (contrary to solutions based on standardized cryptographic primitives such as the AES) [39]. And when trying to mask implementations of LAPIN against side-channel attacks, one additionally needs to protect this randomness generation [40].[1] Based on this state-of-the-art, we start from the observation that LPN-based primitives are ideally suited for implementations in emerging inexact circuit designs, leading us to the following set of technical and conceptual contributions:

1. We show that rather than combining the inner product computations of LPN with a standard Random Number Generator (RNG) to generate errors, we can exploit inexact implementations of these inner product computations.

2. We propose a first technical solution for inexact inner product computations based on over-scaling, i.e. running circuits at too low voltages or too high frequencies. (Many alternatives exist in the nanoelectronics literature).

3. We analyze different hardware architectures for inexact inner product computations based on standard (post-layout) circuit simulators, and show that the error parameter of the Bernoulli distribution used in the LPN problem can theoretically be set by controlling the supply voltage and clock frequency of the inexact implementations.

1. The issues faced by LWE-based primitives are similar (and amplified because of their more complex operations and noise distribution). Relying on the Learning With Rounding (LWR) assumption could be an alternative to get rid of the randomness requirements. Yet, even in this case, primitives such as pseudorandom functions (PRFs) hardly compete with standard block ciphers [41]. Besides, masking the rounding operation is challenging, as recently discussed in [42], [43].

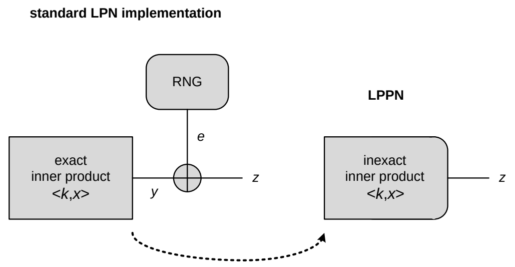

standard LPN implementation
RNG
e
exact inner product <k,x>
y
z
LPPN
inexact inner product <k,x>
z

Fig. 1. From LPN implementations to the LPPN assumption.

4. We confirm our simulated analysis with preliminary experiments based on measurements taken on a prototype chip. Precisely, since we do not have an inner product computation manufactured, we compared the error distribution predicted thanks to simulations and the one measured on a prototype AES S-box chip, implemented in the same 65-nanometer technology as our inner product simulations.

5. Based on these observations, we introduce a new assumption of Learning Parity with Physical Noise (LPPN), and discuss its connections and differences with the standard LPN assumption and the previously introduced PUFs. We exhibit a separation between the deterministic and probabilistic features exploited in LPPN for this purpose.

6. We evaluate technical challenges related to this new assumption. In particular, we detail how to limit the data dependencies of the errors in LPPN implementations based on deterministic glitches, and the control needed on the frequency or voltage of inexact inner product chips, to ensure the security of a realistic 512-bit instance with state-of-the-art tools borrowed from the electronic literature.

7. We put forward that these new techniques directly lead to efficient masked computations. Namely, we describe how to mask the implementation of low-cost primitives based on LPPN with only linear overheads, and without the hassle to protect the Bernoulli noise. For this purpose, we take advantage of the fact that this noise is only implicitly generated in inexact implementations. We also show how masking allows further reducing the security requirements of LPPN implementations based on deterministic errors.

8. We argue that LPPN has good properties to prevent important classes of fault attacks, and benefits from masking in this context (contrary to block ciphers where the interaction between masking and fault attacks is tricky [44], [45], [46]).

In summary and as illustrated in Figure 1, our main contribution is to show that, rather than combining exact computations with additive errors (as usually considered so far), primitives based on hard learning problems could be efficiently implemented by directly producing inexact outputs. This brings three main advantages. From the performance point-of-view, we save the cost of generating random numbers and can implement the inner product computations in a low-power manner. Indeed, inexact inner product computations can be simply produced with a chip running with less energy than required to produce correct outcomes. Next, and more conceptually, LPPN can exploit both probabilistic and deterministic features in microelectronic devices. This makes it fundamentally different from

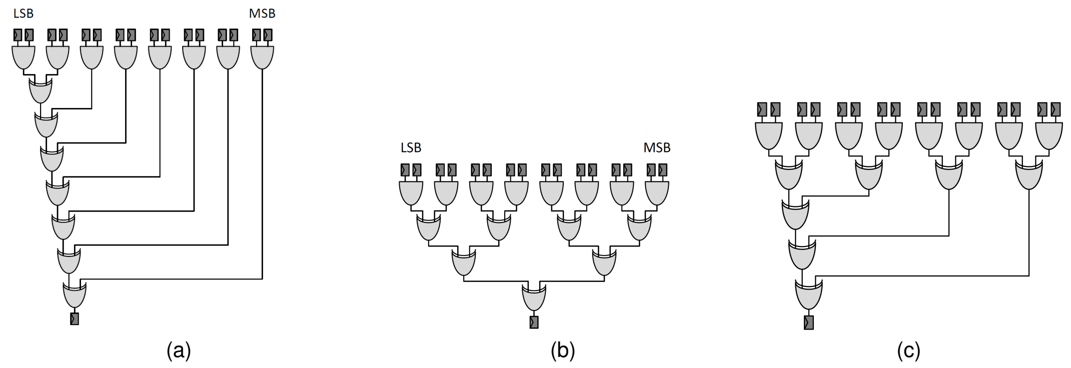


Fig. 2. Inner product architectures.

standard LPN implementations with a RNG, which can only exploit probabilistic ones. As will be discussed in the paper, it leads to interesting design tradeoffs since both types of effects come with pros and cons regarding their security and technical implementation challenges. Eventually, and maybe most importantly, the LPPN assumption avoids the need to protect the LPN randomness against side-channel attacks, since this randomness is never explicitly computed by the chip. That is, while a single probe on the RNG output is enough to break a standard implementation of LPN, this attack is not possible with LPPN, which additionally benefits from interesting features to resist fault attacks. We finally mention that for the LPPN assumption to be practically relevant, its errors must be controlled by an internal and autonomous block, so that an adversary is not capable of altering the probability of error by controlling the circuits' external signals (e.g. supply voltage or clock frequency). We argue in the rest of the paper that this is achievable using state-of-the-art building blocks from the literature.

Cautionary note. This paper focuses on the possibility to exploit emerging concepts from the micro/nanoelectronic design literature in a cryptographic context. Our main observation is that physically secure versions of the LPN assumptions can *in principle* be implemented very efficiently in hardware. Admittedly, there is still a long way to go before translating these expectations into concrete improvements, and best exploiting state-of-the-art tools for accurately controlling inexact inner product computations on a prototype chip (which is a nice scope for further research). Yet, our preliminary investigations suggest that LPPN could become a long-term alternative to challenge-response authentication with high security against side-channel and fault attacks. And most importantly, we believe this assumption opens a wide design space to investigate, at the intersection between symmetric cryptography and electrical engineering.

2 LPN & HARDWARE IMPLEMENTATIONS

2.1 LPN problem

The LPN problem can be written as follows [47]:

Definition 1 (LPN problem). Let $\langle .,. \rangle$ denote the binary inner product, k be a random n-bit secret, let $\epsilon \in]0, \frac{1}{2}[$ be a noise parameter, Ber_ϵ be the Bernoulli distribution with parameter ϵ (if $e \leftarrow \mathsf{Ber}_\epsilon$, then $\Pr[e = 1] = \epsilon$ and $\Pr[e = 0] = 1 - \epsilon$), and $D_{k,\epsilon}$ be the distribution defined as:

$$D_{k,\epsilon} =: \{x \leftarrow \{0,1\}^n; e \leftarrow \mathsf{Ber}_\epsilon : (x, \langle x, k \rangle \oplus e)\}.$$

Let $\mathcal{O}_{k,\epsilon}$ denote an oracle outputting independent samples according to the distribution $D_{k,\epsilon}$. The LPN_ϵ^n problem is said to be (q,t,m,θ)-hard to solve if for any algorithm A, the following inequality holds:

$$\Pr[k \leftarrow \{0,1\}^n : A^{\mathcal{O}_{k,\epsilon}}(1^n) = k] \leq \theta,$$

and the algorithm A runs in time $< t$, with memory $< m$ and makes at most q queries to the oracle $\mathcal{O}_{k,\epsilon}$.

2.2 Architectures for the inner product computations

Following Definition 1, computing LPN samples traditionally requires two components: one to compute the inner products and one to generate the noise. In this section, we focus on the inner product computations and define three types of hardware architectures for this purpose, represented in Figure 2. (The RNG is not needed for inexact computations). First, an s-bit serial architecture is represented in the left part of the figure for $s = 8$: in this case, the $s - 1$ XOR gates of the inner product computation are implemented sequentially. Second, a p-bit parallel architecture is represented in the middle part of the figure for $p = 8$: in this case, the inner product is implemented with a tree of depth $\log(p) + 1$. Third, a $(p \times s)$-bit mixed architecture is represented in the right part of the figure, and mixes two 2-bit parallel architectures with a 4-bit serial one.

2.3 Glitches and over-scaling

Besides the combinatorial gates, the other important components in Figure 2 are the memory elements plotted as (small) dark grey rectangles, next denoted as registers. In conventional (synchronous) CMOS designs, these registers have a data input and a clock input (which is usually common to all registers in the chip), and they sample the value of their data input, e.g. at every rising edge of the clock signal. As illustrated in Figure 3, the data input after some (e.g. inner product) computation is not directly stable: there are a number of transient oscillations, usually called glitches, that take place beforehand. Therefore, the clock period for correct operation is selected such that it is longer than the largest time needed to produce the correct computation result (taken over all possible inputs and variations), that is called the critical path. In practice, this critical path depends on the logic depth of the circuit, and so does the amount of glitches (as also illustrated in the figure). In standard combinatorial designs, one generally tries to minimize the logic

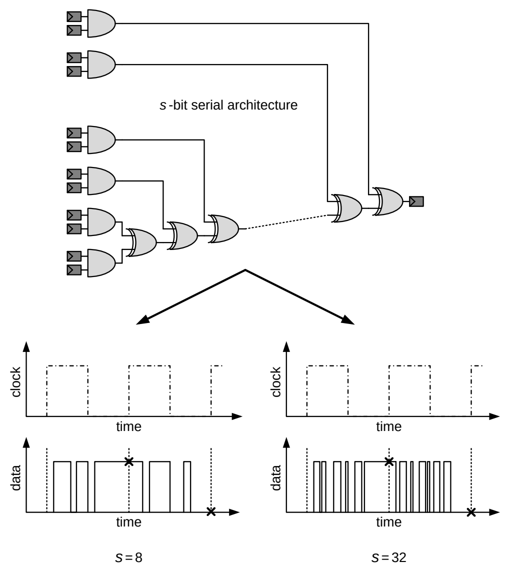


Fig. 3. Glitches.

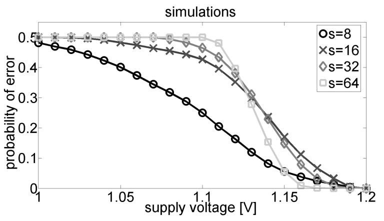
depth (and therefore the amount of glitches), and would prefer a parallel architecture for this reason. However, in our case glitches will be exploited constructively. Hence, serial and mixed architectures will also be considered.

3 Deterministic effects: over-scaling

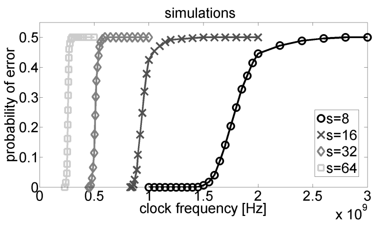
In this section, we study solutions to perform inner product computations with deterministic errors. By deterministic, we mean that for a fixed x and k, the error when computing $\langle x, k \rangle$ will always be the same (i.e. there is no noise nor randomness involved). Intuitively, such effects can be exploited in LPPN because the probability to observe twice the same challenge x in LPN is exponentially small. Concretely, we will take advantage of glitches to produce incorrect inner product outcomes, by updating the register content earlier than required by the critical path (i.e. while there are still transient effects in the register's data input). In this context, the main challenge is to find a way to control the amount of errors due to glitches autonomously on-chip. Quite naturally, changing the logic depth on-the-fly is not feasible since it is fixed once a chip is manufactured. We argue next that over-scaling thanks to increased clock frequencies or reduced supply voltages are two options to reach this goal. For this purpose, we first describe our simulation settings. Then we report experiments exhibiting the dependency between the probability of error caused by glitches and the frequency / voltage of inexact implementations.

3.1 Simulation settings

We consider implementations of a serial inner product computation with $s = 8, 16, 32$ and 64, in a 65nm low-power (LP) CMOS technology. Post-layout simulations are performed with ELDO, and based on buffered data inputs (k and x) and clock signal. For each simulation step (i.e. value of the supply voltage or clock frequency), we use 1000 uniformly distributed random s-bit data inputs that vary successively as would be the case in an real-world authentication scenario. For the frequency scaling setup, the supply voltage is fixed at 1.2V and the frequency is



(a) Supply voltage sweep.



(b) Frequency sweep.

Fig. 4. Controlling the probability of error via over-scaling.

increased with different steps, according to the logic depth s and the resulting operation frequency range (recall that higher logic depth implies a longer critical path and therefore a lower operation frequency). For the voltage scaling setup, the frequency is fixed to its maximum f_{max} for each logic depth s, and the supply voltage is scaled from 1.2V to 1V with 10mV steps. The output parity bit from the output register is then sampled for each value of the 1000 data inputs at each frequency / voltage step, and compared to the expected parity. Finally, we assumed that the outputs of these simulations follow the Bernoulli distribution and estimated the probability of error based on the ratio between the number of incorrect parities and the total number of parities computed. The correlation between the inputs and outputs of our inexact implementations is discussed in Sections 7.1 and 8.2. Note that for a given random pair (k, x), the results produced by the simulations in this section are deterministic due to the fact that if the data registers in Figure 3 are synchronized with a noise-free clock signal and supplied with a noise-free power source, then the resulting glitches only depend on the (fixed) circuit layout topology.

3.2 Simulation results

The results of our simulated experiments are reported in Figure 4. They lead to two important observations. First and as expected, it is possible to control the probability of error of an inexact inner product implementation by controlling its supply voltage (in the upper part of the figure) or clock frequency (in its lower part). Second, controlling the probability of errors is increasingly difficult as the depth of the serial architecture increases, as illustrated by the steeper curves for increasing s values. This can be explained by the increasing amount of glitches that happen in this case, as illustrated in Figure 3. Besides, for the frequency sweep figure we also witness a translation effect, which reflects the fact that longer critical paths imply lower clock frequencies (this effect is hidden on the voltage scaling figure because it starts from the maximum clock frequency for each value of s). Eventually, we note that the LSB of our architectures has the longest critical path (as shown in Figure 2). This observation admittedly implies a risk of data dependencies in the error probabilities that we discuss in Section 7.1.

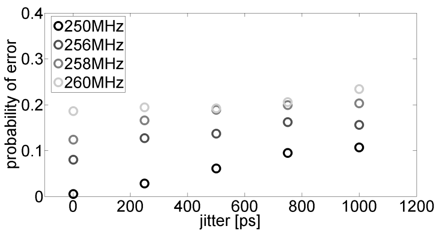


Fig. 5. Increased probability of error via clock jitter.

4 Probabilistic effects: jitter and noise

The simulations in the previous section assumed an hypothetical implementation where all the manipulated signals are noise-free, therefore only leading to deterministic effects. In practice though, the signals in an implementation suffer from fluctuations due to various sources of electronic noise, which causes the previously described "glitchy outputs" to become probabilistic. By probabilistic, we mean that for a fixed x and k, the error when computing $\langle x, k\rangle$ is random (i.e. there is noise involved). Typical imperfections of the signals are the clock jitter (i.e. small instantaneous variations of the clock period, causing the registers to sample with some inaccuracies) and the supply noise (i.e. small instantaneous variations of the supply voltage, causing similar effects). We next study the impact of a clock jitter for illustration.

4.1 Simulation settings

We simulate the impact of clock jitter by creating a jittery clock using MATLAB. More precisely, we randomized the clock period according to a Gaussian distribution with mean zero and standard deviation between 0 and 1ns. We chose the $n = 64$ serial architecture that features a maximum frequency (leading to error-prone computations without jitter) of $\approx$ 250MHz, set the supply voltage to the nominal 1.2V and swept the frequency from 250MHz to 260MHz.

4.2 Simulation results

The results of our simulated experiments are reported in Figure 5. As expected, we observe that the jitter increases the probability of error. We insist that the required amount of jitter is a function of the voltage - frequency setting that we use, and the required probability of error. For example, if we use the nominal supply voltage and a maximum clock frequency of 250MHz (that guarantees correct operation for the $n = 64$ serial architecture) and assume the required probability of error is 0.25, then we will need more than 1ns jitter as can be seen from Figure 5. However, by increasing the frequency, we need less jitter to reach the required probability of error. In this respect, it is important to note that 400ps (resp. 1000ps) of peak-to-peak jitter is equivalent to 0.1 (resp. 0.25) Unit Interval (UI) of jitter for a 250MHz clock frequency, which is compatible to communication standards (e.g. [48]). In case more jitter is required, it is of course possible to generate it by design (e.g. as considered in the RNG literature [49]). Furthermore, in a real implementation, both the supply noise and the clock jitter will always be present to some extent. In this respect, while standard circuit design approaches generally aim to minimize these defaults, our inexact inner product implementations may benefit from the opposite goal of making them significant.

5 Preliminary experiments

The previous sections introduced a number of design trends. Namely, the probability of error of an inner product computation can be controlled by manipulating the supply voltage or clock frequency of its implementation. However, these trends were only put forward via simulations. While this is a necessary first step, because it allows us to reach a good understanding of both deterministic and probabilistic effects in inexact inner product implementations, an important challenge remains to establish our results on a silicon chip. Since taping out a chip is a long-term engineering project, we provide preliminary investigations with a chip implementing the AES S-box in the same 65-nm LP technology as considered in our inner product computations' simulations in Appendix A. It confirms that over-scaling can be used to generate errors in practice, and that predictions obtained via simulations are sufficiently relevant in this respect.

6 Learning Parity with Physical Noise

Based on the experiments in the previous sections, we conclude that it is concretely feasible to produce LPN samples with an inexact implementation of an inner product computation. We now define the LPPN assumption that takes advantage of such implementations. This requires to define a (hopefully hard) "physical LPN problem", where the noise distribution will be determined by the implementations. For this purpose, we first take advantage of the definition of Physical Function from [50], slightly simplified. That is, a physical function $\mathsf{PF}_{d,\alpha}$ is a probabilistic procedure based on a physical device d, which can be stimulated with some input challenge $x \in \{0,1\}^{n_i}$, making d respond with a (probabilistic) output $y \in \{0,1\}^{n_o}$, with α a set of parameters.[2] In our case, d will be an implementation of inexact inner product computations (implying $n_i = n$ and $n_o = 1$), and α the set of parameters determining its error distribution (e.g. the clock frequency or supply voltage). We then re-state the definition of weak unpredictability for physical functions from [50] (again slightly simplified):

Definition 2 (Weak unpredictability). Let $\mathsf{PF}_{d,\alpha}$ be a physical function and A be an algorithm running in time $< t$ and memory $< m$, that takes part in a weak unpredictability experiment defined as follows:

```
$\mathbf{Exp}_A^{\text{w-unp}}(q)$:
$k \xleftarrow{U} \{0,1\}^n$;
Learning phase:
for $i = 1 : q$
    $x_i \xleftarrow{U} \{0,1\}^n$;
    $y_i \leftarrow \mathsf{PF}_{d_k,\alpha}(x_i)$;
end
$\mathcal{Q} = \{x_i, y_i\}_{i=1}^q$;
Challenge phase:
$x \xleftarrow{U} \{0,1\}^n$;
$y \leftarrow \mathsf{PF}_{d_k,\alpha}(x)$;
$y' \leftarrow A(x, \mathcal{Q})$;
Output $(y, y')$.
```

We say that $\mathsf{PF}_{d,\alpha}$ is (q,t,m,γ)-weakly unpredictable if for any such adversary A, we have:

$$\Pr[y = y' : (y, y') \leftarrow \mathbf{Exp}_A^{\text{w-unp}}(q)] \leq \gamma.$$

For clarity, we also specify the specialized physical function (where we use the hat symbol for estimated probabilities):

2. We differ from the definition of [50] by consider digital outputs. Physical functions can in general answer with analog outputs as well.

Definition 3 (Physical inner product). Let $k \in \{0,1\}^n$ be a random n-bit secret stored in the device d_k. A physical function $\mathsf{PF}_{d_k,\alpha}$ is called an $\tilde{\epsilon}$-Physical Inner Product ($\tilde{\epsilon}$-PIP) if, on uniform public input $x \in \{0,1\}^n$, it outputs $\langle x, k\rangle$ with estimated error probability:

$$\hat{\Pr}[\mathsf{PF}_{d_k,\alpha}(x) \neq \langle x, k\rangle] = \tilde{\epsilon},$$

We finally define the LPPN problem as follows:

Definition 4 (LPPN problem). Let $\mathsf{PF}_{d_k,\alpha}$ be a (q,t,m,γ)-weakly unpredictable $\tilde{\epsilon}$-PIP. The $\mathrm{LPPN}^{n,\tilde{\epsilon}}_{d_k,\alpha}$ problem is said to be (q,t,m,θ)-hard to solve if for any algorithm A running in time $< t$, memory $< m$ and making at most q uniformly random queries to the PIP, it holds that:

$$\Pr[k \leftarrow \{0,1\}^n : A^{\mathsf{PF}_{d_k,\alpha}}(1^n) = k] \leq \theta.$$

These definitions suggest three important remarks.

First, the inexact nature of the physical function is needed for the LPPN assumption to hold. Intuitively, more inexact implementations imply larger probabilities of error $\tilde{\epsilon}$, which make the problem harder to solve. However, for the LPPN assumption to be useful in cryptographic applications, the amount of errors also has to be limited, in order to allow legitimate users to authenticate themselves or decrypt encrypted messages efficiently. In the following, we will focus on the characterization of the error distribution and its impact on security. Its impact on the efficiency of a protocol can be directly derived as in the standard LPN cases.

Second, assuming that the LPPN problem is hard is a physical assumption, whereas assuming that the LPN problem is hard is a mathematical one. The main difference is that, since the oracle $\mathcal{O}_{k,\epsilon}$ of the LPN problem is replaced by a physical function in Definition 4, we have no guarantee that the error distribution will be exactly equal to the one specified in Definition 1. This is the price to pay for the improved efficiency of implementations based on LPPN and motivates the design challenges discussed next.

Third, the definition of the LPPN problem relying on physical functions and the implementation of inexact inner product computations suggest that the LPPN assumption can be connected to the existence of strong PUFs (although they are not equivalent, as formalized in Section 8.2). For example, as for PUFs the samples of inexact inner product computations must be weakly unpredictable (or an adversary could predict LPPN samples without the key). Besides, our proposed construction exploiting glitches is reminiscent of the glitch PUFs in [51] (yet, the latter construction is different since we exploit the transient values of a "glitchy signal" while it relies on the number of glitches in such a signal). Hence, a natural question is whether the LPPN problem could be susceptible to modeling attacks.

In this respect, a first observation is that XORing different physical functions together, as we do in our hybrid architectures, generally makes these attacks more difficult. Besides, such attacks are usually most efficient against physical effects which can be explained with linear models (e.g. the additive delay model for arbiter PUFs [52]), and it is unclear whether the glitchy signal of inexact inner product computations have such a simple linear behavior, in particular when a direct access to this signal (available only at the input of the register) is not granted to the adversary.

Second and more importantly, the LPPN assumption is fundamentally different than assuming the existence of PUFs, for which robustness – as defined in [50] – is additionally required. Conceptually, probabilistic effects (i.e. physical noise sources) can indeed only be detrimental in the context of PUFs, since they should be small enough for a user being able to reconstruct a stable answer after some post-processing (e.g. fuzzy extraction [53]). By contrast in the case of LPPN, the errors can come both from deterministic effects (that are indeed similar to the ones exploited in PUFs) and probabilistic effects (that are closer to the ones exploited in true RNGs). So depending on how much an LPPN assumption relies on deterministic (resp. probabilistic) effects, its security will be closer to the assumption that strong (e.g.) glitch PUFs (resp. good RNGs based on clock jitter / supply noise) exist. As mentioned in introduction, we believe this difference opens a large design space for cryptographic hardware research, to find implementations producing secure LPPN samples at minimum cost.

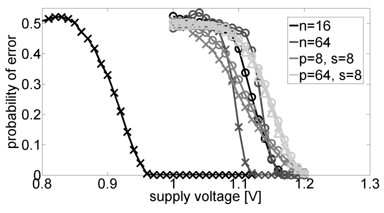
Third and most importantly, from an adversarial point-of-view, a significant difference between modelling attacks against PUFs and the LPPN assumption is the presence of a secret key in the inner product computations of which the glitches should be predicted. While in the context of PUFs the adversary can (try to) model some physical feature of an implementation based on a full knowledge of its internal state, in the LPPN case he only has access to the public x values, which makes the modeling more challenging (moving from supervised to unsupervised machine learning).

7 Technical challenges

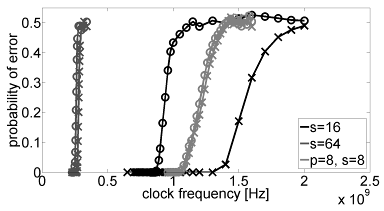
As it is clear from Definitions 2 and 4, there are two natural ways to break the LPPN assumption. First one can try to break the weak unpredictability of the physical function, by modeling its behavior (e.g. with a linear approximation). Second, one can directly target the LPPN assumption with standard algorithms such as [19], [21], [47], of which the complexity depends on the error probability $\tilde{\epsilon}$. In practice, this means that in order to guarantee the security of protocols based on LPPN, one at least needs to control this error probability. We now study two technical challenges for this purpose. First, we analyze the possibility for an adversary to reduce $\tilde{\epsilon}$ thanks to plaintext filtering, and show that this risk can be mitigated by appropriate architectural choices. Second, and since perfectly controlling the frequency or voltage of a chip is not achievable, we specify concrete guidelines for these physical quantities by analyzing the impact of a small deviation on the error probability for the security of the LPPN problem. Note that these discussions are connected to our specific instances of inexact computations.

7.1 Data dependencies

In general, secure LPN instances require that the (Bernoulli) noise needed to generate the outputs is independent of the inputs. Therefore, in the case of LPPN an important question is to study whether correlations between the inputs and outputs of an inexact computation could be exploited by an adversary. Since the instances of LPPN implementations in this paper are based on deterministic glitches, a natural track to investigate in this respect is the impact of the inputs'



(a) Supply voltage sweep.



(b) Frequency sweep.

Fig. 6. Impact of filtered plaintexts: ○ are for random inputs, × are for inputs with 8 bits stuck at 0, △ are for inputs with 32 bits stuck to 0.

Hamming weight / distance on the probability of error $\tilde{\epsilon}$. Note that in our setting where we send successive inputs to the inexact inner product computations, reducing the inputs' Hamming weight by setting their LSBs to zero implies reducing their Hamming distance because it sets the first bits of the inner product computation to zero independent of the key. As mentioned in Section 3.2, one could expect that $\tilde{\epsilon}$ decreases with such inputs, which we experiment next.

For this purpose, we investigated the same setting as in Section 3, but this time compared a situation with random inputs and inputs such that a number of their LSBs are stuck to zero. Note that since the LPPN assumption only works with random inputs, setting bits to zero can only be done by filtering the plaintexts. As illustrated in Figure 6, such data dependencies are very clearly observed in the case of a serial architecture, both in the voltage and frequency domains (see, e.g. the curves for $s = 16$ where 8 bits are set to zero). This is simply explained by the fact that setting some LSBs to zero means that the XORs whose inputs are now zero are nulled, which effectively reduces the critical path. Yet, there are two simple ways to mitigate this effect. First, increasing the size of the architecture makes the filtering less effective (since a smaller fraction of the input bits are set to zero), as illustrated with the $s = 64$-bit serial architecture. Second, and more interestingly, such data dependencies vanish in the case of parallel architectures in which the presence of glitches is much more limited (since there are much less imbalances between the routing paths of such architectures). So overall, we need serial architectures for the glitch generation and parallel architectures for the data independence.[3] As a result, a natural option is to consider the mixed architecture of Figure 2, where we start with the parallel stage (with limited glitches and data dependencies) and then use the serial stage for error control. As illustrated in Figure 6 this gives significantly improved results. For example, the mixed architecture with $p = 8$ and $s = 8$ has significantly less data dependencies than the $s = 64$ one.

3. If relying on deterministic effects. Parallel architectures with probabilistic errors could also be good candidates for LPPN implementations.

TABLE 1
Maximum deviations tolerated on the error probabilities for a target $\tilde{\epsilon}_{tgt} = 1/4$ and $\tilde{\epsilon}_{min} = 1/8$, with security levels λ_{tgt} and λ_{min}.

n	256	512	768	1280
$\tilde{\epsilon}_{tgt}$	0.25			
λ_{tgt}	64	99	139	198
$\tilde{\epsilon}_{min}$	0.125			
λ_{min}	52	82	109	161
$\max(\Delta_\epsilon)$	0.125			
$\max(\Delta_\lambda)$	12	17	30	37

Note that similar dependencies can be observed for the key, implying a risk of weak keys which is easily mitigated in two ways. First, the (frequency or voltage) overscaling or the addition of probabilistic noise can be done adaptively (i.e. after the selection of the key), as proposed in conclusion. Second, the vast majority of the keys will have a Hamming weight close to $n/2$. So, one could use keys with Hamming weights in a given range without hurting the system.

To conclude these investigations, we analyzed a realistic case study of 512-bit inexact inner product computation, based on a mixed architecture with $p = 64$ and $s = 8$. (Due to long simulation times, this architecture was only studied in the voltage domain). We further looked at the impact of fixing 8 and 32 input bits to zero in this case. As illustrated in the top part of Figure 6, the impact of such data dependencies is again very limited. More precisely, we checked that the deviations on the probability $\tilde{\epsilon}$ they imply are smaller than the deviations of this probability that we are anyway going to tolerate due to imperfect control of the supply voltage and clock frequency (see the next section). Hence, we conclude that even in the case of LPPN implementations based on deterministic glitches only, data dependencies can be made small enough to ensure security against filtering attacks. Note that LPPN exploiting probabilistic effects can only be less sensitive to such dependencies, since the errors may also come from data-independent noise in this case.

7.2 Internal chip control

The last example of 512-bit mixed architecture is particularly appealing, since it combines limited data dependencies with a limited slope of the error curves (since these errors are mostly due to the 8-bit serial part of the computation). It is therefore a good starting point to argue that controlling our inexact implementations in order to ensure a sufficient security level is achievable with state-of-the-art techniques. For this purpose, we first need to quantify acceptable security degradations due to imperfect control, which we will do with the recent attack complexities' estimates by Bogos et al. [54]. That is, let us consider a target error probability of $\tilde{\epsilon}_{tgt} = 1/4$ and a lowest tolerated error probability of $\tilde{\epsilon}_{min} = 1/8$, so that the maximum tolerated deviation $\Delta_\epsilon = 1/8$, with corresponding security levels λ_{tgt}, λ_{min} and a maximum difference of security levels $\max(\Delta_\lambda)$, security levels $\max(\Delta_\lambda)$, as indicated in Table 1. This means that for $n = 512$ bits, we target a security level of ≈ 99 bits, and want to ensure that deviations due to a lack of control on the error probability do not decrease this security level down to less than ≈ 82 bits. Next, one can directly translate this error control into clock frequency or supply voltage control by looking at the figures in the previous section. For example, $\tilde{\epsilon}_{tgt} = 0.25$ corresponds to a supply voltage of ≈ 1.145V for our 512-bit mixed architecture, and $\tilde{\epsilon}_{min} = 0.125$ to a supply voltage of ≈ 1.170V. This means that we can tolerate errors of 2% which is easily achievable. For example, reference [55] describes a generator that outputs 0.9 V and minimizes the voltage variations to 0.73%. Note that at 1.145V, the error probability of our mixed architecture with 8 (resp. 32) bits set to zero decreases from 0.25 to 0.23 (resp. 0.21) which, as previously mentioned, indicates comfortable security margins against plaintext filtering attacks (since we set $\tilde{\epsilon}_{min} = 0.125$). An even more comfortable situation holds for frequency control. For example, we estimated the clock frequencies corresponding to $\tilde{\epsilon}_{tgt} = 0.25$ and $\tilde{\epsilon}_{min} = 0.125$ for our 512-bit mixed architecture as approximately 1GHz and 950MHz, respectively. Reference [56] provide a clock signal in the GHz range with a frequency resolution of 100 Hz, which is orders of magnitude more accurate.

Based on these references, we conclude that an internal and autonomous control of either the supply voltage or the clock frequency is possible with existing tools. Note that by internal we mean that an adversary should not be able to alter the probability of error by controlling the circuits' IOs. And by autonomous we mean that this control should adjust the error probability in case of environmental variations (e.g. an adversary changing the chip's temperature). We leave the exact implementation of such a chip control as a scope for further research (some potential directions are given in conclusion) and next focus on the cryptographic treatment and applications of the new LPPN assumption.

Additional remark #1. When probabilistic errors are exploited in an LPPN implementation, it may be useful to have a systematic way to exploit lower error rates (since a small jitter may only generate $\tilde{\epsilon} < 0.125$). In such cases, a natural way to design the implementation is to consider two cycles: a first one where small errors are generated for small inner products, a second one where these inner products are recombined in an error-free stage – which would then increase the error according to the piling up lemma [57]:

Lemma 1 (Piling up lemma). Let X_i $(1 \leq i \leq \eta)$ be independent random variables whose values are 0 with probability p_i and 1 with probability $1 - p_i$. Then, the probability that $X_1 \oplus X_2 \oplus \ldots \oplus X_\eta = 0$ is:

$$\frac{1}{2} + 2^{\eta-1} \prod_{i=1}^{n} (p_i - \frac{1}{2}).$$

Table 2 provides values for the control of the error probability in such cases, with r the number of XORs in the second stage. For example, $n = 512$ and $r = 4$ would mean that we implement four 128-bit inner products in the first cycle, each of them with a target probability of error of 0.0796, that are XORed together in the second cycle. Interestingly, the increase of the noise level after recombination in the second cycle provides a mathematical foundation to the increased steepness of the curves with large s values in Section 3.

We finally mention that strictly speaking there is no obligation that deterministic errors in inexact inner product computations depend on all their n input bits. Indeed, the security levels that we target with the LPPN assumption are anyway significantly below 2^n because of generic attacks. So if we denote with s this target security level, errors coming from $2s < n$ bits could be sufficient, so that it remains hard for the adversary to detect two samples with the same error (i.e. a collision on the noise-influencing inputs).

TABLE 2
Max. deviations tolerated on the errors prob. for 2-cycle inner product architectures (with r inner products combined in the 2nd cycle).

$r = 1$	$\tilde{\epsilon}_{tgt}$	0.25
	$\tilde{\epsilon}_{min}$	0.125
	$\max(\Delta_\epsilon)$	0.125
$r = 2$	$\tilde{\epsilon}_{tgt}$	0.1464
	$\tilde{\epsilon}_{min}$	0.0670
	$\max(\Delta_\epsilon)$	0.0794
$r = 4$	$\tilde{\epsilon}_{tgt}$	0.0796
	$\tilde{\epsilon}_{min}$	0.0347
	$\max(\Delta_\epsilon)$	0.0449

Additional remark #2. The interface between the LPPN design and other digital blocks on-chip depends on the control parameter. If the frequency is the control parameter, a phase-locked loop can be used to generate the required on-chip clock. In order to process the input bits, they need to be stored in a memory and then accessed by the LPPN design with its internal clock frequency. If the supply voltage is the control parameter, it can be provided by an on-chip Linear Drop Out (LDO) voltage regulator that has good line regulation (e.g. [58]). The interface with other digital blocks operates at the external supply voltage, and level shifters can be employed among the different voltage islands.

8 Cryptographic treatment

The previous sections initiated a reasoning that physical sources of errors whose importance increases with technology scaling, can be efficiently exploited to define hard physical learning problems. In this section, we complement this discussion with a cryptographic treatment of these new tools. For this purpose, we first connect the weak unpredictability of a physical function with its LPPN hardness. Then we formalize the notions of deterministic and probabilistic effects that were informally introduced in our experimental evaluation of the physical properties that can be exploited in the design of hard physical functions.

8.1 Weak unpredictability implies LPPN hardness

Definitions 2 and 4 consider the weak unpredictability of a PIP and the LPPN hardness as independent notions. In this section, we show that weak unpredictability is a sufficient condition for LPPN hardness with the following lemma:

Lemma 2 (Weak unpredictability $\Rightarrow$ LPPN hardness). Let $\mathsf{PF}_{d_k,\alpha}$ be an $\tilde{\epsilon}$-PIP physical function. If $\mathsf{PF}_{d_k,\alpha}$ is (q, t, m, γ)-weakly unpredictable, then the LPPN problem is $(q, t, m, (\gamma - 1/2)/(1/2 - 2\tilde{\epsilon} + 2\tilde{\epsilon}^2)$-hard.

Proof: We will show that an (q, t, m, θ) adversary $\mathcal{A}$ against LPPN implies an $(q, t, m, \theta(1/2 - 2\tilde{\epsilon} + 2\tilde{\epsilon}^2) + 1/2)$ adversary $\mathcal{B}$ against the weak unpredictability of $\mathsf{PF}_{d_k,\alpha}$. We define $\mathcal{B}$ as follow. First collect q queries in the learning phase and forward them to $\mathcal{A}$. With probability θ, $\mathcal{A}$ will return us the key k of the device. On challenge x, $\mathcal{B}$ outputs $\langle x, k\rangle \oplus \nu$, where ν is drawn from a Bernoulli distribution with parameter $\tilde{\epsilon}$. We then have:

$$\begin{aligned}\Pr[\mathcal{B}\text{ wins}] = \Pr[\mathcal{B}\text{ wins} \mid \mathcal{A}\text{ wins}]\Pr[\mathcal{A}\text{ wins}] \\ + \Pr[\mathcal{B}\text{ wins} \mid \mathcal{A}\text{ loses}]\Pr[\mathcal{A}\text{ loses}], \\ = (\tilde{\epsilon}^2 + (1-\tilde{\epsilon})^2)\theta + \frac{1}{2}(1-\theta)\ .\end{aligned}$$

□

By contrast, LPPN hardness does not imply weak unpredictability. A trivial counterexample is a PIP that outputs a constant string independent of x and k.

8.2 Deterministic vs. probabilistic hardness

In Sections 3 and 4 , we discussed solutions to produce inexact inner product implementations exploiting both deterministic and probabilistic effects. In view of the importance of these concepts, this section aims to formalize them, and discuss their relations with existing hard learning problems. For this purpose, we introduce the following model.

Definition 5 (PIP model). Let $\mathsf{PF}_{d_k,\alpha}$ be an $\tilde{\epsilon}$-PIP and denote $\mathsf{D}_{d,k,\alpha}(x)$ as a deterministic function of the device d, the key stored in it k and the parameters α. Let also $R_{d,\alpha}$ be a random variable depending on the device d and parameters α such that $\hat{\Pr}[\mathsf{D}_{d,k,\alpha}(X) \oplus R_{d,\alpha} = 1] = \tilde{\epsilon}$. We define the $\tilde{\epsilon}$-PIP model as follows:

$$\mathsf{PF}_{d_k,\alpha}(x) = \langle x, k\rangle \oplus \mathsf{D}_{d,k,\alpha}(x) \oplus R_{d,\alpha}.$$

Note that we consider $R_{d,\alpha}$ to be additive and independent of x and k, which is a reasonable first-order approximation since it mostly depends on physical signals (e.g. the supply voltage and clock in our examples) that are independent of x and k. Based on this model, a deterministic-only LPPN assumption corresponds to the case where $\mathsf{D}_{d,k,\alpha} \neq 0$ and $R_{d,\alpha} = 0$ and a probabilistic-only LPPN assumption corresponds to the case where $\mathsf{D}_{d,k,\alpha} = 0$ and $R_{d,\alpha} \neq 0$.

Probabilistic-only LPPN. In this case, it is easy to see that the physical function behaves exactly as a standard LPN oracle, with estimated noise parameter $\tilde{\epsilon}$. Concretely, this corresponds to a chip where rather than increasing (resp. decreasing) the clock frequency (resp. supply voltage) in order to perform erroneous computations, we would exploit and amplify their instability. Such designs are similar to RNGs based on oscillators [49]. So this approach is conceptually more conservative, since exploiting randomness in a quite traditional manner. In this respect, one advantage of probabilistic LPPN is that it mitigates the risks of data-dependent errors discussed in Section 7.1. Besides, we insist that even in this case LPPN is not equivalent to LPN, because we exploit the intrinsic randomness of the computations rather than extrinsic randomness, which is different from the side-channel attacks point-of-view (see Section 9).

Deterministic-only LPPN. By contrast, arguing that deterministic errors are computationally indistinguishable from probabilistic ones from the adversary's viewpoint is a much stronger hypothesis, since it essentially requires the function $\mathsf{D}_{d,k,\alpha}(x)$ to be cryptographically strong in some sense. Concretely, this for example corresponds to a case with noise-free clock and supply voltage signals, but overscaled such that the implementation produces (deterministic) erroneous answers, as previously investigated. Conceptually, one can relate such a function to the LWR problem [59], [60], which can be seen as an instance of a deterministic LWE problem. However, whereas the noise in the LWR assumption is emulated with a (deterministic) rounding function chosen by the cryptographer, this function is specified by the (physics of) the device d in our LPPN case. This leads to the question of whether there are realistic conditions for $\mathsf{D}_{d,k,\alpha}$ that are sufficient to make LPPN secure. We show next that different options can be considered for this purpose.

Variability. A first option is to build on variability. That is, one would then take advantage of the fact that every chip implementing LPPN leads to a different d, extracted from a hard to characterize distribution. For example, if d was uniformly distributed for each device and drawn from a set of size greater than some security parameter, it would be sufficient to have D a biased weak-PRF, with d a key such that $\Pr[\mathsf{D}_{d,k,\alpha}(X) = 1] = \tilde{\epsilon}$. This requirement is quite strong, and may be even more difficult to verify empirically than directly evaluating the physical unpredictability. Yet, we note that this is typically the type of condition that is required for the security of strong PUF instances. The latter again illustrates that LPPN is a less demanding assumption than strong PUFs, since variability is one physical effect (among others) that can be exploited to generate errors. For example, neither the previous probabilistic effects nor the following non-linearity can be leveraged in a PUF design.

Non-linearity. Even if all the chips implementing LPPN lead to identical d values, i.e. in the absence of variability, a second (complementary) option is to build on the possible cryptographic strength of the deterministic error function. For this purpose, and for illustration, let us write a deterministic $\tilde{\epsilon}$-PIP for the mixed architecture of Section 7 as:

$$\begin{aligned}\mathsf{PF}_{d_k,\alpha}(x) &= \underbrace{z(1) \oplus z(2) \oplus \ldots \oplus z(\tilde{n})}_{\langle x,k\rangle} \\ &\oplus \underbrace{\delta(z(1), z(2), \ldots, z(\tilde{n}))}_{\mathsf{D}_{d,k,\alpha}(X)}, \qquad (1)\end{aligned}$$

where the bits of the $\tilde{n}$-bit vector z typically correspond to partial inner products, e.g. intermediate results of the $\langle x, k\rangle$ computations that are XORed together in the final (serial) part of the architecture. Furthermore, let us assume that the errors are generated by a deterministic function δ that only takes the bits of z as input. Clearly, if δ was unique and public, but corresponding to a non-linear Boolean function, then breaking the PIP would correspond to cryptanlyzing a "physical stream cipher" filtering linear combinations of key bits with δ. In this case, we would rather require that D is a biased weak-PRF with secret key k, leading to interesting tradeoffs between security (for which large $\tilde{n}$ are preferable) and control (for which smaller values are preferable).

Discussion. Both the variabilty option and the non-linearity option raise new research challenges regarding the characterization of emerging nanotechnologies, and the cryptographic analysis of deterministic error functions provided by physical measurements. Interestingly, in practice, it is usally a combination of both probabilistic and deterministic effects that will be observed, the latter ones mixing variability and possible non-linearity features. So this observation again highlights the broadness of the design space opened by LPPN and the need of further experimental investigations in order to better understand the respective impact of these different physical effects. Even more interestingly, we next show that in case neither variability nor non-linearity effects are sufficient to provide secure deterministic LPPN (which would still leave the room for secure probabilistic LPPN), it is actually possible to exploit their cryptographically weak deterministic errors in case the $\tilde{\epsilon}$-PIPs are masked against side-channel attacks. Since the main selling point of LPPN implementations is anyway their potential for enhanced physical security, this last observation strengthen their relevance to practice by reducing the physical requirements for securely exploiting deterministic effects.

9 Masked inner product computations

9.1 Efficient implementation

As such, inexact implementations of inner product computations as outlined in the previous sections are already interesting, since they offer an efficient and possibly low-power way to implement LPN-based primitives. In particular, they remove the need to have Bernoulli RNGs embedded (e.g.) on the prover side of an authentication protocol, which may be a bottleneck for low-cost devices [39]. However, the most interesting implementation feature of inexact inner products is their ability to be efficiently masked. Indeed, and as first hinted in [36], the inner products used in LPN inherently have an easy-to-exploit key homomorphism, namely $\langle x, k_A \oplus k_B \rangle = \langle x, k_A \rangle \oplus \langle x, k_B \rangle$. This implies that this part of the computations can be masked in an optimal manner (i.e. with fully linear overheads, using the key refreshing discussed in Theorem 4 of [61]). Moreover, since the key shares are manipulated independently, such a masked implementation also offers a direct protection against glitches, which are generally expensive to prevent [62], [63]. As a result, masked LPPN implementations can provide exponential security increases (in the number of shares) at the sole condition that the leakages of the shares are sufficiently noisy, directly following the prediction in [64], [65].

Note that the glitches considered here as a detrimental side-effect against masking are different from the glitches exploited constructively in Section 2.3. In the first case, we refer to glitches that combine different shares, while in the second case we consider glitches that are internal to the computation of each share. So in fact, the only limitation of protocols such as LAPIN from the side-channel resistance point-of-view was the difficulty to generate the Bernoulli noise in a leakage-resilient manner [40]. Incidentally, this is exactly where the LPPN assumption comes in handy. That is, and as it is clear from Figure 1, such a noise does not have to be generated, since erroneous outputs will be directly produced by inexact implementations. So the single probe attack (on the RNG output) that directly breaks masked implementations of the standard LPN is prevented by design in the LPPN case, and inexact inner product implementations can be fully masked against side-channel attacks, with only linear overheads. In this respect, we do not claim that protecting standard LPN implementations against side-channel attacks is impossible. An alternative option remains to protect their RNGs with masking techniques or other countermeasures. However, as for our other arguments in favor of LPPN, we believe that this standard approach will generally imply more overheads.

9.2 Masking implies independent deterministic errors

As discussed in the previous section, ensuring that the deterministic errors of an LPPN implementation do not lead to exploitable correlations leads to various engineering challenges. We now show that it is possible to get around these challenges and to guarantee that (under some assumptions) deterministic LPPN can be secure even if its underlying function is cryptographically weak. For this purpose, we first notice that masked implementations of inexact inner products do not require more control on the error probabilities. Indeed, it is sufficient that the errors affect only one share, since all of them are XORed before the output is made available to the adversary. Assuming a shared key $k = k_A \oplus k_B$, we can therefore re-write Equation 1 as:

$$\begin{aligned} \mathsf{PF}_{d_k,\alpha}(x) &= \underbrace{z_A(1) \oplus z_A(2) \oplus \ldots \oplus z_A(\tilde{n})}_{\langle x, k_A \rangle} \\ &\oplus \underbrace{\delta(z_A(1), z_A(2), \ldots, z_A(\tilde{n}))}_{\mathsf{D}_{d,k,\alpha}(x)} \\ &\oplus \underbrace{z_B(1) \oplus z_B(2) \oplus \ldots \oplus z_B(\tilde{n})}_{\langle x, k_B \rangle}. \end{aligned} \quad (2)$$

Now let us assume that the $\tilde{n}$-bit error function δ has a truth table with Hamming weight $\frac{\tilde{n}}{4}$ (aiming for $\tilde{\epsilon} \approx \frac{1}{4}$) or $\frac{\tilde{n}}{8}$ (aiming for $\tilde{\epsilon} \approx \frac{1}{8}$). Then, we directly have that if z_A is uniformly distributed and unknown (which happens if the masked implementation is leak-free), the security of the corresponding $\tilde{\epsilon}$-PIP only depends on the probability of error $\tilde{\epsilon}$ (which now only requires to be controlled as discussed in Section 7). In other words, by harnessing the randomness of masked implementations, we can strengthen the independence guarantees of deterministic LPPN errors without any cryptographic assumption on D. Admittedly, it then remains to show that these guarantees hold in front of side-channel adversaries who try to learn the z_A vector by observing some leakages. For this purpose, (besides the fact that D may have some cryptographic strength and be combined with probabilistic effects), we first note that nothing prevents adding a term $\delta(z_B(1), z_B(2), \ldots, z_B(\tilde{n}))$ to the third line of Equation 2, so that the deterministic errors are shared as well. In this case, security against side-channel attacks trying to bias the error distribution would increase directly as predicted in [64], [65], at the cost of an increased control predicted by the piling up lemma.

More fundamentally, even if the errors are not shared, side-channel attacks against such $\tilde{\epsilon}$-PIP have strong similarities with the recent Learning with Leakage (LPL) assumption (which reduces to LPN) [66]. This suggests that such a PIP could be proven secure assuming some amount of noise in the measurements. We leave the reduction between masked LPPN and LPL as an interesting open problem.

10 Security against fault attacks

Before to conclude, we first mention that LPN-based primitives generally have useful features for improved resistance against fault attacks as well. In particular, it has been shown recently that some prominent fault models (e.g. where an adversary flips bits in an implementation) are ineffective against LPN implementations and that attacks taking advantage of more advanced fault models (e.g. where an adversary sets bits in an implementation) require significantly more samples than against standard symmetric cryptographic primitives such as block ciphers [37]. More precisely, this reference showed that for fault attacks targeting the inner product computation, the number of faulty samples to recover an LPN key is typically in the thousands range (compared to a few pairs for block ciphers). Furthermore, in case the accuracy of the fault is not perfect (e.g. if the fault hits one out of Δ intermediate results with uniform probability rather than one precise intermediate result), this number of samples increases by a factor Δ^2. In this respect, we note that high accuracy will typically be difficult to obtain in our hardware architectures where many intermediate results of the inner product are computed in a single cycle. As for attacks directly targeting the randomness by setting the additive error to zero (which allows simple key recovery by Gaussian elimination), the work in [37] additionally showed they are even more sensitive to inaccuracies. So this threat is in fact very effectively prevented in LPPN where the error is not well located in the implementations.

Eventually, we observe that combined resistance against side-channel and fault attacks is difficult to obtain with state-of-the-art tools. For example, in the case of block cipher implementations it may happen that a fault detection mechanism creates additional sources of information leakage [45]. More generally, the interaction between countermeasures against those two types of attacks always requires special care and masking as such does not help against fault attacks [44], [46]. In this respect, a very interesting feature of masked inner product computations is that they actually increase the difficulty of performing fault attacks with imperfect accuracy. Indeed, and taking the example of the inner products in Equation 2, the only samples that can be efficiently exploited by an adversary are those for which the faults hit the same intermediate computation for the two shares. Hence, and following a reasoning similar to the one in [37], we have that if the faults on the inner products of the two shares are only accurate up to the previously introduced Δ parameter, the number of samples needed for key-recovery increases by a factor Δ^4. The latter becomes Δ^{2d} with d shares, which is rapidly prohibitive.

11 Conclusions & perspectives

In this paper, we introduced a conceptually new solution to implement cryptographic primitives based on the LPN problem, and formalized it with the LPPN assumption. We further proposed first instantiations of this assumption, taking advantage of inexact inner product computations that are an emerging trend in the micro/nanoelectronic literature. Eventually, and based on circuit simulations and measurements on a prototype chip, we confirmed experimentally that the error probability of these instances (that is the main factor influencing the hardness of the LPPN problem) can be controlled thanks to frequency and voltage overscaling. We also discussed their excellent properties for resistance against side-channel and fault attacks. In view of the exploitability of LPN in an increasingly large variety of cryptographic protocols, we hope these results will stimulate further research in the efficient implementation of LPPN instances and the analysis of their leakage-resilience.

Interestingly, while most current cryptographic research follows a top-down methodology (i.e. define mathematical assumptions that then have to be fulfilled by implementers), our results suggest that the opposite approach (i.e. starting from physical features made availble by technology scaling, and trying to build secure cryptosystems based on them) may lead to useful outcomes as well, as already outlined in the PUF literature. This complementary view leads to different open problems. As mentioned in Section 8.2, finding reductions to the minimum conditions allowing secure LPPN assumptions based on deterministic physical functions is an important challenge. Besides, there is a wide variety of other error distributions that could be evaluated. In particular, our investigations focused on inner product computations in $\mathbb{Z}_2$. But one could naturally consider computations over larger rings, and generalizations towards more general Learning With Physical Errors (LWPE) problems. For example, Figure 8 in Appendix A illustrates that the probability of error for 4-bit S-box outputs (measured as a fraction of incorrect outcomes) can be controlled via overscaling. The use of physical distributions also raises the question whether small deviations from the ideal distributions assumed in the mathematical analysis of hard learning problems can significantly degrade their actual security level.

From the hardware point-of-view, our investigations indicate that there are at least simple instances of inexact product computations, based on frequency and voltage overscaling, that allow efficient implementations and for which the technical challenges to obtain data independent and accurate error probabilities can be solved with state-of-the-art solutions. Hence, it is an interesting project to translate this proof-of-concept into a fully functional and efficient prototype where this control is operated autonomously on chip (this is fundamental since an adversary should not be able to have any external control of the error probability).[4] Besides, it is worth insisting that our investigations only covered a subset of the architectures and physical effects that could be used for inexact implementations. For example, and as far as our proposed instances are concerned, there is a wide literature on adaptive voltage and frequency scaling that could be exploited to further improve the control of these quantities [67]. More prospectively, the exploitation of internal transistor noise suggested in [68], [69] is an interesting trend to investigate in the LPPN context, since the amount of such noise increases with technology scaling. So overall, we believe the proposals in this paper lead to interesting new problems, at the intersection between cryptographic and electrical engineering, with potential applications for secure, efficient and low(er) cost implementations.

4. relying on FPGA implementations for this prototyping of the LPPN assumption may not be easy, in view of the precise control and understanding we need about the underlying hardware. Therefore, taping out an ASIC design is most likely needed for this purpose.

Acknowledgments. François-Xavier Standaert is a senior research associate of the Belgian fund for Scientific Research (FNRS-F.R.S.). This word has been funded in parts by the ARC project NANOSEC and the ERC project 724725.

References

[1] M. Feldhofer, S. Dominikus, and J. Wolkerstorfer, "Strong authentication for RFID systems using the AES algorithm," in *CHES 04*. LNCS 3156, 2004, pp. 357–370.

[2] Y. K. Lee *et al.*, "Low-cost untraceable authentication protocols for RFID," in *WISEC 10*. ACM, 2010, pp. 55–64.

[3] S. Mangard, E. Oswald, and T. Popp, *Power analysis attacks - revealing the secrets of smart cards*. Springer, 2007.

[4] M. Joye and M. Tunstall, Eds., *Fault Analysis in Cryptography*, ser. Information Security and Cryptography. Springer, 2012.

[5] O. Regev, "The learning with errors problem (invited survey)," in *CCC 10*, 2010, pp. 191–204.

[6] K. Pietrzak, "Cryptography from learning parity with noise," in *SOFSEM 12*. LNCS 7147, 2012, pp. 99–114.

[7] N. J. Hopper and M. Blum, "Secure human identification protocols," in *ASIACRYPT 01*. LNCS 2248, 2001, pp. 52–66.

[8] A. Juels and S. A. Weis, "Authenticating pervasive devices with human protocols," in *CRYPTO 05*. LNCS 3621, 2005, pp. 293–308.

[9] H. Gilbert, M. J. B. Robshaw, and Y. Seurin, "$HB^{\#}$: Increasing the Security and Efficiency of HB^{+}," in *EUROCRYPT 08*. LNCS 4965, 2008, pp. 361–378.

[10] E. Kiltz *et al.*, "Efficient authentication from hard learning problems," in *EUROCRYPT 11*. LNCS 6632, 2011, pp. 7–26.

[11] S. Heyse *et al.*, "LAPIN: An efficient authentication protocol based on ring-LPN," in *FSE 12*. LNCS 7549, 2012, pp. 346–365.

[12] Y. Dodis *et al.*, "Message authentication, revisited," in *EUROCRYPT 12*. LNCS 7237, 2012, pp. 355–374.

[13] A. Blum *et al.*, "Cryptographic primitives based on hard learning problems," in *CRYPTO 93*. LNCS 773, 1993, pp. 278–291.

[14] H. Gilbert, M. J. B. Robshaw, and Y. Seurin, "How to encrypt with the LPN problem," in *ICALP 08*. LNCS 5126, 2008, pp. 679–690.

[15] A. Duc and S. Vaudenay, "HELEN: A public-key cryptosystem based on the LPN and the decisional minimal distance problems," in *AFRICACRYPT 13*. LNCS 7918, 2013, pp. 107–126.

[16] V. Lyubashevsky *et al.*, "SWIFFT: A modest proposal for FFT hashing," in *FSE 08*. LNCS 5086, 2008, pp. 54–72.

[17] C. Gentry, C. Peikert, and V. Vaikuntanathan, "Trapdoors for hard lattices and new cryptographic constructions," in *STOC 08*. ACM, 2008, pp. 197–206.

[18] Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) LWE," in *FOCS 11*, 2011, pp. 97–106.

[19] A. Blum, A. Kalai, and H. Wasserman, "Noise-tolerant learning, the parity problem, and the statistical query model," in *STOC 00*. ACM, 2000, pp. 435–440.

[20] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," in *STOC 05*. ACM, 2005, pp. 84–93.

[21] Q. Guo, T. Johansson, and C. Löndahl, "Solving LPN using covering codes," in *ASIACRYPT 14*. LNCS 8873, 2014, pp. 1–20.

[22] Q. Guo, T. Johansson, and P. Stankovski, "Coded-BKW: Solving LWE using lattice codes," in *CRYPTO 15*. LNCS 9215, 2015, pp. 23–42.

[23] P. Kirchner and P. Fouque, "An Improved BKW Algorithm for LWE with Applications to Cryptography and Lattices," in *CRYPTO 15*. LNCS 9215, 2015, pp. 43–62.

[24] A. Duc, F. Tramèr, and S. Vaudenay, "Better algorithms for LWE and LWR," in *EUROCRYPT 15*. LNCS 9056, 2015, pp. 173–202.

[25] M. R. Albrecht *et al.*, "On the complexity of the BKW algorithm on LWE," *Des. Codes Cryptography*, vol. 74, no. 2, pp. 325–354, 2015. [Online]. Available: https://doi.org/10.1007/s10623-013-9864-x

[26] N. S. Kim *et al.*, "Leakage current: Moore's law meets static power," *computer*, vol. 36, no. 12, pp. 68–75, 2003.

[27] S. Borkar, "Designing reliable systems from unreliable components: the challenges of transistor variability and degradation," *IEEE Micro*, vol. 25, no. 6, pp. 10–16, 2005.

[28] S. Ghosh and K. Roy, "Parameter variation tolerance and error resiliency: New design paradigm for the nanoscale era," *Proceedings of the IEEE*, vol. 98, no. 10, pp. 1718–1751, 2010.

[29] C. Hocquet *et al.*, "Harvesting the potential of nano-CMOS for lightweight cryptography: an ultra-low-voltage 65 nm AES coprocessor for passive RFID tags," *J. Cryptographic Engineering*, vol. 1, no. 1, pp. 79–86, 2011.

[30] A. Moradi, "Side-channel leakage through static power - should we care about in practice?" in *CHES 04*. LNCS 8731, 2014, pp. 562–579.

[31] S. M. D. Pozo *et al.*, "Side-channel attacks from static power: When should we care?" in *DATE 15*, 2015, pp. 145–150.

[32] M. Renauld *et al.*, "A formal study of power variability issues and side-channel attacks for nanoscale devices," in *EUROCRYPT 11*. LNCS 6632, 2011, pp. 109–128.

[33] R. Maes and I. Verbauwhede, "Physically unclonable functions: A study on the state of the art and future research directions," in *Towards Hardware-Intrinsic Security - Foundations and Practice*. Springer, 2010, pp. 3–37.

[34] B. E. S. Akgul *et al.*, "Probabilistic CMOS technology: A survey and future directions," in *IFIP VLSI-SoC 06*, 2006, pp. 1–6.

[35] L. Avinash *et al.*, "Synthesizing parsimonious inexact circuits through probabilistic design techniques," *ACM TECS 13*, vol. 12, no. 2s, p. 93, 2013.

[36] M. Medwed and F. Standaert, "Extractors against side-channel attacks: Weak or strong?" *J. Cryptographic Engineering*, vol. 1, no. 3, pp. 231–241, 2011.

[37] F. Berti and F.-X. Standaert, *An Analysis of the Learning Parity with Noise Assumption Against Fault Attacks*. Cham: LNCS 10146, 2017, pp. 245–264.

[38] F. Armknecht, M. Hamann, and V. Mikhalev, "Lightweight authentication protocols on ultra-constrained RFIDs - myths and facts," in *RFIDSec 14*. LNCS 8651, 2014, pp. 1–18.

[39] D. J. Bernstein and T. Lange, "Never trust a bunny," in *RFIDSec 12*. LNCS 7739, 2012, pp. 137–148.

[40] L. Gaspar, G. Leurent, and F. Standaert, "Hardware implementation and side-channel analysis of LAPIN," in *CT-RSA 14*. LNCS 8366, 2014, pp. 206–226.

[41] A. Banerjee *et al.*, "SPRING: fast pseudorandom functions from rounded ring products," in *FSE 14*. LNCS 8540, 2014, pp. 38–57.

[42] H. Brenner *et al.*, "FPGA implementations of SPRING - and their countermeasures against side-channel attacks," in *CHES 14*. LNCS 8731, 2014, pp. 414–432.

[43] O. Reparaz *et al.*, "A masked ring-LWE implementation," in *CHES 15*. LNCS 9293, 2015, pp. 683–702.

[44] B. M. Gammel and S. Mangard, "On the duality of probing and fault attacks," *J. Electronic Testing*, vol. 26, no. 4, pp. 483–493, 2010.

[45] F. Regazzoni *et al.*, "Interaction between fault attack countermeasures and the resistance against power analysis attacks," in *Fault Analysis in Cryptography*. Springer, 2012, pp. 257–272.

[46] T. Schneider, A. Moradi, and T. Güneysu, "Parti - towards combined hardware countermeasures against side-channel and fault-injection attacks," in *CRYPTO 16*. LNCS 9815, 2016, pp. 302–332.

[47] É. Levieil and P. Fouque, "An improved LPN algorithm," in *SCN 06*. LNCS 4116, 2006, pp. 348–359.

[48] "Usb 3.0 electrical compliance methodology," White Paper, USB.org, june 2015.

[49] B. Sunar, W. J. Martin, and D. R. Stinson, "A provably secure true random number generator with built-in tolerance to active attacks," *IEEE Transactions on Computers*, vol. 56, no. 1, pp. 109–119, 2007.

[50] F. Armknecht *et al.*, "A formalization of the security features of physical functions," in *SSP 11*, 2011, pp. 397–412.

[51] D. Suzuki and K. Shimizu, "The glitch PUF: A new delay-PUF architecture exploiting glitch shapes," in *CHES 10*. LNCS 6225, 2010, pp. 366–382.

[52] U. Rührmair *et al.*, "Modeling attacks on physical unclonable functions," in *CCS 10*. ACM, 2010, pp. 237–249.

[53] Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," in *EUROCRYPT 04*. LNCS 3027, 2004, pp. 523–540.

[54] S. Bogos, F. Tramèr, and S. Vaudenay, "On solving LPN using BKW and variants - implementation and analysis," *Cryptography and Communications*, vol. 8, no. 3, pp. 331–369, 2016.

[55] Y. Lam and W. Ki, "A 0.9v 0.35 μm adaptively biased CMOS LDO regulator with fast transient response," in *ISSCC 08*, 2008, pp. 442–443.

[56] H. S. Kim *et al.*, "A digital fractional-n PLL with a PVT and mismatch insensitive TDC utilizing equivalent time sampling technique," *JSSC 13*, vol. 48, no. 7, pp. 1721–1729, 2013.

[57] M. Matsui, "Linear cryptanalysis method for DES cipher," in *EUROCRYPT 93*. LNCS 765, 1993, pp. 386–397.

[58] S. Peng *et al.*, "A power-efficient reconfigurable output-capacitor-less low-drop-out regulator for low-power analog

sensing front-end," *IEEE Trans. on Circuits and Systems*, vol. 64-I, no. 6, pp. 1318–1327, 2017. [Online]. Available: https://doi.org/10.1109/TCSI.2016.2561638

[59] A. Banerjee, C. Peikert, and A. Rosen, "Pseudorandom functions and lattices," in *EUROCRYPT 12*. LNCS 7237, 2012, pp. 719–737.

[60] J. Alwen *et al.*, "Learning with rounding, revisited - new reduction, properties and applications," in *CRYPTO 13*. LNCS 8042, 2013, pp. 57–74.

[61] G. Barthe *et al.*, "Parallel implementations of masking schemes and the bounded moment leakage model," in *EUROCRYPT 17*. LNCS 10210, 2017, pp. 535–566.

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

[63] B. Bilgin *et al.*, "Higher-order threshold implementations," in *ASIACRYPT 14*. LNCS 8874, 2014, pp. 326–343.

[64] S. Chari *et al.*, "Towards sound approaches to counteract power-analysis attacks," in *CRYPTO 99*. LNCS 1666, 1999, pp. 398–412.

[65] A. Duc, S. Faust, and F. Standaert, "Making masking security proofs concrete - or how to evaluate the security of any leaking device," in *EUROCRYPT 15*. LNCS 9056, 2015, pp. 401–429.

[66] S. Dziembowski *et al.*, "Towards sound fresh re-keying with hard (physical) learning problems," in *CRYPTO 16*. LNCS 9815, 2016, pp. 272–301.

[67] I. M. Panades *et al.*, "A fine-grain variation-aware dynamic vdd-hopping AVFS architecture on a 32 nm GALS MPSoC," *JSSC 14*, vol. 49, no. 7, pp. 1475–1486, 2014.

[68] K. Nepal *et al.*, "Designing logic circuits for probabilistic computation in the presence of noise," in *DAC 05*, 2005, pp. 485–490.

[69] J. George *et al.*, "Probabilistic arithmetic and energy efficient embedded signal processing," in *CASES 06*. ACM, 2006, pp. 158–168.

[70] D. Kamel, F. Standaert, and D. Flandre, "Scaling trends of the AES S-box low power consumption in 130 and 65 nm CMOS technology nodes," in *ISCAS 09*, 2009, pp. 1385–1388.

[71] Wikipedia, "Process corners," https://en.wikipedia.org/wiki/Process_corners, 2015.

Dina Kamel received a Master's degree from Ain Shams University in Egypt in 2008 and a PhD degree from the Universite catholique de Louvain (UCL) in Belgium in 2012. Since 2012, she is a post-doc in the Crypto Group of the Institute of Information and Communication Technologies, Electronics and Applied Mathematics (ICTEAM) at UCL. Her current research interests include the low-power design of integrated circuits for secure applications, hardware implementations of hard problems and low-cost, physically-secure countermeasures against side-channel analysis.

François-Xavier Standaert received the Electrical Engineering degree and PhD degree from the Universite catholique de Louvain, respectively in 2001 and 2004. In 2004-2005, he was a Fulbright visiting researcher at Columbia University, Department of Computer Science, Crypto Lab (hosted by Tal G. Malkin and Moti Yung) and at the MIT Medialab, Center for Bits and Atoms (hosted by Neil Gershenfeld). In 2006, he was a founding member of IntoPix s.a. From 2005 to 2008, he was a post-doctoral researcher of the Belgian Fund for Scientific Research (FNRS-F.R.S.) at the UCL Crypto Group and a regular visitor of the two aforementioned laboratories. Since 2008 (resp. 2017), he is associate researcher (resp. senior associate researcher) of the Belgian Fund for Scientific Research (FNRS-F.R.S) and professor at the UCL Institute of Information and Communication Technologies, Electronics and Applied Mathematics (ICTEAM). In 2011 (resp. 2016) he was awarded a Starting Independent Research (resp. consolidator) Grant by the European Research Council. His research interests include cryptographic hardware and embedded systems, lightweight implementations (for RFIDs, sensor networks, ...), the design and cryptanalysis of symmetric cryptographic primitives, physical security issues in general and side-channel analysis in particular.

Alexandre Duc received a Master's degree from EPFL in Switzerland in 2011 and a PhD degree from EPFL in Switzerland in 2015. In 2016-2017, he was a cryptography architect at Inpher during which he studied and implemented searchable encryption schemes and multi-party computation. Since 2017, he is a professor in information security at the University of Applied Sciences and Arts of Western Switzerland (HES-SO/HEIG-VD) in Yverdon-les-Bains in Switzerland. He is now involved in applied research in blockchain, pseudo-random number generators, and post-quantum cryptography. His research interests include the study of cryptographically hard problems, the security of blockchain applications, leakage-resilient cryptography, and post-quantum cryptography.

Denis Flandre (M'85–SM'03) received the EE and PhD degrees from Université catholique de Louvain in 1986 and 1990, resp. Since 2001, he is full-time Professor at UCL. He has authored more than 900 technical papers or conference contributions. He is co-inventor of 12 patents. He has lectured many short courses on SOI technology, devices and circuits. He is involved in R&D on SOI MOS devices, digital and analog circuits, as well as sensors and MEMS, for high-speed, low-voltage low-power, microwave, biomedical, radiation-hardened and high-temperature electronics and microsystems. He is co-founder of CISSOID (SOI and high-reliability IC products), and scientific advisor of INCIZE (Semiconductor characterization and modeling for digital, analog/RF and harsh environment design) and e-peas (Energy harvesting and processing solutions for IoT applications). He has been a member of EU Networks of Excellence on High-Temperature Electronics, SOI technology, Nano-electronics and Micro-nano-technology, and of the SOI Industry Consortium.

Francesco Berti received the bachelor's degree and the master's degree in Mathematics from the Universitá degli Studi di Milano in 2012 and 2015, respectively. He is currently working toward a PhD degree at the Crypto Group, Université Catholique de Louvain, Institute of Information and Communication Technologies, Electronics and Applied Mathematics (ICTEAM). His research interests include the design and cryptanalysis of symmetric cryptographic primitives, with particular attention to the leakage-resilient authenticated encryption, side-channel and fault analysis.

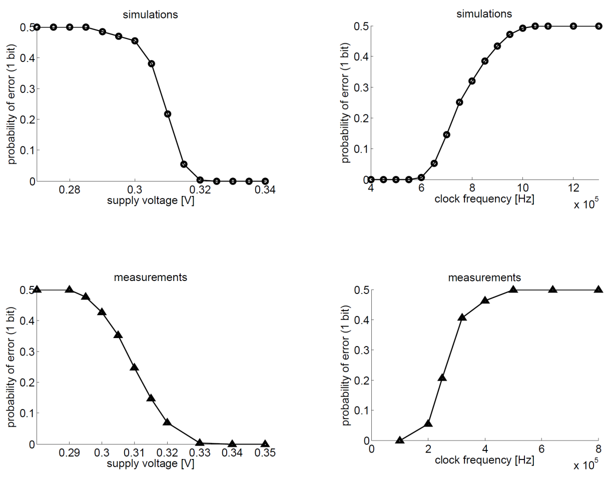
APPENDIX A
S-BOX MEASUREMENT RESULTS

A.1 Simulation and experimental settings

Our AES S-box chip consists in a combinatorial S-box that has been full-custom designed (details can be found in [70]). Its 8-bit inputs and outputs are buffered, and in order to operate the S-box at lower than nominal (1.2V) supply voltages, a level shifter is added after each S-box output bit, before being buffered. The output buffers are sized large enough to be able to drive the output pads of the chip at maximum frequency and nominal supply voltage. However, due to the presence of these output buffers, some glitches that are present at the S-box output are unfortunately suppressed while operating at nominal (1.2V) supply voltage, because they are too fast. Consequently, we decided to operate the S-box at a lower supply voltage of 0.35V. The output bits of the S-box after the level shifters and the buffers are captured using a Lecroy HRO 66 ZI oscilloscope (at 2GS/s). Eventually, we sampled the output waveforms at the required time samples as a post-processing (software) step, in order to emulate the sampling of an output register. Our simulation setting for this S-box computation is essentially similar to the one of the inner product computations in Section 3.1. Compared to the measurements, the only difference is that the sampling of the waveforms is directly performed thanks to registers that we added before the input buffers and after the output buffers. As in the previous sections, we estimated the error probabilities based on 1000 uniformly distributed inputs (taking > 256 inputs was justified by the possible presence of probabilistic effects in our measurements, and by the fact that glitches are dependent on the 256^2 possible input transitions in our S-box design). In order to be comparable to inner product computations, we only considered the errors on a single output bit.

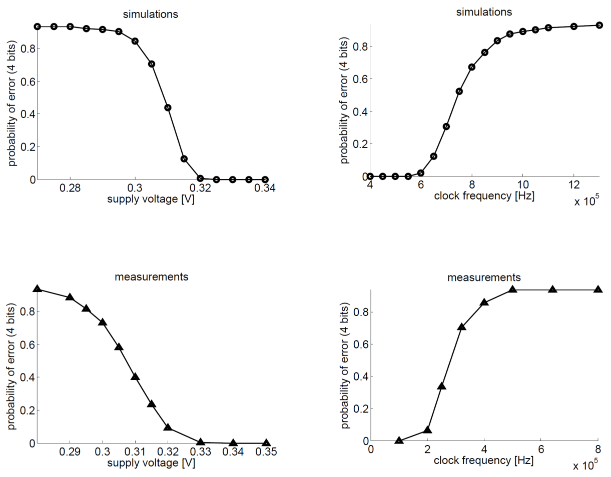
A.2 Simulation and experimental results

Our results for the simulated (resp. experimental) AES experiments are in the upper (resp. lower) part of Figure 7. Once again (in fact even more than in the previous sections), we focus on the conceptual lessons rather than the concrete values which may differ for a real prototype chip implementing inner product computations. In this respect, the main observation is that for this AES case study as well, it is possible to control the probability of error of an output bit via supply voltage and clock frequency manipulations. Furthermore, this observation now holds for both simulated and actual measurements. In fact, the main difference between these two settings is the slightly lower clock frequency allowed by the test chip – which can be explained since previous experiments with this chip already suggested that it lies in the slow-slow corner [70], [71].



(a) Supply voltage sweep. (b) Frequency sweep.

Fig. 7. Probability of error for an S-box output bit.



(a) Supply voltage sweep. (b) Frequency sweep.

Fig. 8. Probability of error for 4 S-box output bit.