Noname manuscript No.
(will be inserted by the editor)

Side-channel Analysis of a Learning Parity with Physical Noise Processor

Dina Kamel* · Davide Bellizia · Olivier Bronchain · François-Xavier Standaert

Received: date / Accepted: date

Abstract Learning Parity with Physical Noise (LPPN) has been proposed as an assumption on which to build authentication protocols based on the Learning Parity with Noise (LPN) problem. Its first advantage is to reduce the randomness requirements of standard LPN-based protocols, by directly performing erroneous computations so that no (e.g. Bernoulli-distributed) errors have to be generated on chip. At ASHES 2018, an LPPN processor was presented and confirmed the possibility to efficiently generate erroneous computations with the appropriate error rate. Since LPPN computations are key-homomorphic, they are good candidates for improved side-channel security thanks to masking, since they could theoretically lead to masked implementations with overheads that are linear in the number of shares, the analysis of which was left as an open problem. In this paper, we confirm this good potential by analyzing the side-channel security of an LPPN processor. We (1) evaluate the leakage of different parts of the erroneous computations, (2) conclude that intermediate computations that can be targeted with a divide-and-conquer Gaussian template attack are a sweet spot for side-channel attacks, and (3) show that LPPN computations naturally reach a level of noise that makes masking effective, despite further noise addition could be beneficial to reach higher security at lower implementation cost.

Keywords Learning parity with noise · Side-channel analysis · Authentication · Probabilistic computation

F. Author
Université Catholique de Louvain (UCLouvain)
Tel.: +123-45-678910
Fax: +123-45-678910
E-mail: *dina.kamel@uclouvain.be

S. Author
Université Catholique de Louvain (UCLouvain)
E-mail: davide.bellizia@uclouvain.be

3. Author
Université Catholique de Louvain (UCLouvain)
E-mail: olivier.bronchain@uclouvain.be

4. Author
Université Catholique de Louvain (UCLouvain)
E-mail: francois-xavier.standaert@uclouvain.be

1 Introduction

In light of the emergence of the Internet of Things (IoT) for an increasingly wide range of applications implicated in every day life (e.g. smart homes and cities, building management, e-health, ... etc), connected devices must feature low-power/energy, low-cost and most importantly minimum security guarantees (e.g. lightweight authentication [11]). Due to their conceptual simplicity, protocols based on the Learning Parity with Noise (LPN) problem are considered as promising candidates for this purpose [23]. However, due to the requirement of a (Pseudo) Random number Generator (RNG), which may be expensive and at the same time an easy target for side-channel analysis, the practical relevance of such protocols remains questionable. In [15], the authors introduced a working instance of the recently proposed Learning Parity with Physical Noise (LPPN) assumption [16], which mitigates the need of an RNG by directly performing erroneous computations and proved to be efficient. However, the side-channel security aspect of such an implementation has so far been left as an open problem.

In general, security against physical attacks where an adversary monitors side-channel leakages [19] or induces faults [14] in order to recover cryptographic keys is an important design challenge. This is especially true in the context of the IoT which features tight area and power/energy constraints. For example, the masking countermeasure is a popular solution to minimize side-channel leakage in block cipher implementations, but the implementation cost of such a solution grows (roughly) quadratically in the number of shares [13]. Moreover, the physical assumptions upon which its security depends may not be easy to meet, due to physical (hardware) effects such as glitches [21, 22] and coupling [7, 8, 17] that can recombine the leakages of the shares, thus reducing the “security order” of the implementations (i.e. the lowest statistical moment of the leakage distributions that is key-dependent). Furthermore, the interaction between masking and countermeasures against fault attacks may lead to additional overheads [25] and weaknesses [24].

In contrast with block ciphers, implementations of LPN-based protocols and its physical counterpart (LPPN) inherently offer good properties to resist against side-channel attacks via masking at a linear cost [12], thanks to their key-homomorphic nature. Moreover, they have inherently good features to resist against fault attacks [4]. In this paper we therefore analyze the side-channel security of an LPPN processor for the first time, which complements the work in [15]. As usual in side-channel analysis, we aim at a worst-case security level [26]. We observe that LPPN co-processors are interesting in this respect since their key-homomorphism implies that the independence condition required for masking to be effective is easier to guarantee by design. As a result, an evaluator mostly has to quantify the information of each share in order to evaluate a worst-case security level, following the bounds given in [9]. We evaluate such bounds based on a prototype LPPN implementation and use the recent tools of [5] to also consider powerful multivariate attacks. Our results allow exhibiting the best targets for side-channel attacks within LPPN-based implementations and confirm the good potential of such implementations for physical security.

Contribution. We provide a side-channel evaluation of an LPPN implementation proposed in [15], taking advantage of sound Information Theoretic (IT) tools. Results are based on actual measurements performed on a 512-bit (64-bit parallel × 8-bit serial) LPPN processor fabricated in 28nm FDSOI CMOS technology. For comparison purposes, we analyze the information leakage based on a single bit at a time, which allows us to compare the informativeness of the different stages of our implementation. Our main conclusions are twofold:

- After analyzing the information leakage of all the internal stages of the LPPN implementation, we identified both the output of the first LPPN stage and the final output bit as the most informative targets for a side-channel attack. Since the first one is exploitable via a Gaussian template attack while the second one requires more sophisticated algebraic attacks [3, 2, 10], we conclude that it corresponds to a sweeter spot for concrete adversaries.
- Second, quantitative evaluations in univariate and multivariate settings indicate that we reach levels of leakage such that a masked implementation would be effective as is (yet, it could possibly be improved by adding some noise generators).

The paper is organized as follows. Section 2 presents the LPN problem, the LPPN assumption and the LPPN processor implementation. In section 3 we provide the background regarding the evaluation metrics and settings used in the paper. A discussion of the side-channel analysis of the LPPN processor using univariate and multivariate settings is given in section 4. Finally, section 5 concludes the paper & discusses open problems.

2 LPPN processor

2.1 LPN problem

Let $k \in \{0,1\}^m$ be a random m-bit secret, $x \in \{0,1\}^m$ be an m-bit public input and $\langle x, k\rangle$ denote the binary inner product of x and k. Let $\epsilon \in]0, \frac{1}{2}[$ be a noise parameter that follows a Bernoulli distribution (Ber_ϵ) such that if $e \leftarrow \mathsf{Ber}_\epsilon$, then $\Pr[e = 1] = \epsilon$ and $\Pr[e = 0] = 1 - \epsilon$, and $D_{k,\epsilon}$ be the distribution:

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

Let $\mathcal{O}_{k,\epsilon}$ denote an oracle outputting independent samples according to the distribution $D_{k,\epsilon}$. The LPN_ϵ^m problem is said to be (q, t, me, θ)-hard to solve if for any algorithm A (that runs in time $< t$, with memory $< me$ and makes at most q queries to the oracle $\mathcal{O}_{k,\epsilon}$), the following inequality holds:

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

2.2 LPPN assumption

Let $\mathsf{PF}_{d_k,\alpha}(\mathrm{x})$ be a physical function (as defined in [1] and simplified in [16]), which we next call an $\tilde{\epsilon}$-Physical Inner Product ($\tilde{\epsilon}$-PIP) [15]. It is based on a physical device d_k that stores a random m-bit secret $k \in \{0,1\}^m$, has α set of parameters, and which can be stimulated with a uniform public input $x \in \{0,1\}^m$ so that it outputs $\langle x, k\rangle$ with estimated error probability: $\hat{\Pr}[\mathsf{PF}_{d_k,\alpha}(\mathrm{x}) \neq \langle x, k\rangle] = \tilde{\epsilon}$. In such a context, the

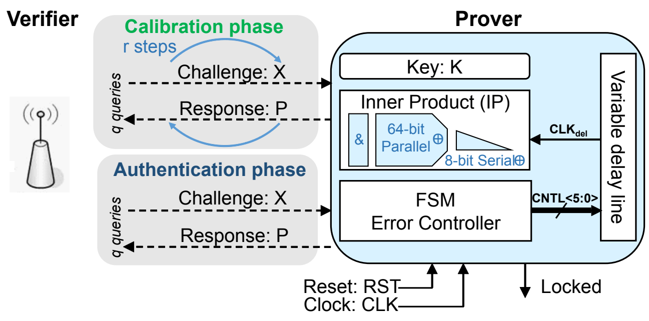


Fig. 1: LPPN processor architecture.

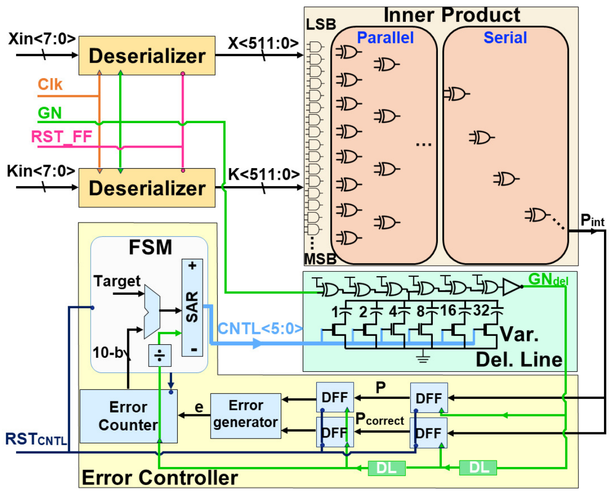


Fig. 2: LPPN processor block diagram.

LPPN problem can be described just as the LPN problem, with as only difference that the LPN samples are replaced by the outputs of an $\tilde{\epsilon}$-PIP.

2.3 LPPN implementation and performance

The main challenge to design an $\tilde{\epsilon}$-PIP is to control the error probability $\tilde{\epsilon}$ such that it cannot be tampered externally by an adversary. The LPPN processor designed in [15] meets such a challenge (to a sufficient extent) by preceding the authentication operation by a calibration phase where the verifier exchanges sets of challenge-responses (q queries) during r steps with the prover (as shown in Fig. 1) to adjust its control such that the error probability is kept within the designated bounds during the following authentication phase. It comprises a 512-bit inner product (IP) logic block (which is a mixed architecture of first 64-bit parallel, then 8-bit serial), a variable delay line that outputs a delayed version of the clock CLK_{del} to sample the output of the IP block during its glitchy period and a finite state machine error controller that regulates the variable delay of the sampling clock through a 6-bit control signal $CNTL$. The error controller works only during calibration and issues a *Locked* signal once it is done while keeping the same values of the $CNTL$ bits to guarantee the correct error probability during authentication. The current LPPN implementation exploits both deterministic (architecture- and data-dependent due to the presence of glitches) and probabilistic physical effects (such as supply noise and jitter for example).

Figure 2 details the design of the LPPN processor. In the current prototype implementation both the secret key and the challenge are sent to the LPPN processor.[1] In order to transmit the required 1024-bit (the 512-bit challenge and key), two 8-bit deserializers are implemented. The inner product block, comprising a $(p \times s)$-bit mixed architecture computes the parity signal P_{int}. It is important to note that the choice of such an architecture reduces the possibility of exploiting data-dependent error probability (due to glitches which are deterministic) where an adversary mounts a so-called filtering attack, as discussed in [16] and experimentally validated in [15]. The variable delay line uses digitally-controlled delay elements with shunt-capacitors via NMOS switches. To control these switches, an error controller provides a 6-bit control signal ($CNTL$) which is adjusted based on the required probability during the calibration phase. The erroneous parity P and the correct (delayed) version of it, $P_{correct}$, are sampled via a pair of DFFs in the error controller block. An error generator compares both P and $P_{correct}$, then computes the error signal e which is counted via a 10-bit error counter over 1024 queries. To achieve the required error probability, a comparator block finally compares the error count to the target count (e.g. 256 for $\epsilon = 0.25$). Then, the $CNTL$ bits are adjusted through the finite state machine (FSM) in a successive approximation scheme.

The timing diagram in Fig. 3a demonstrates how the 8-bit key (K_{in}) and 8-bit bit challenge (X_{in}) are shifted through the deserializers' flip-flops during 64 clock cycles, then loaded during the low state of the GN clock signal (which is generated on-chip). In order to sample the inner product block output P_{int} during its glitchy period, the GN clock signal is delayed and inverted through the variable delay line. The correct parity bit is only made available during the calibration phase and not during the authentication phase, otherwise the probing-like attacks that the LPPN assumption is supposed to mitigate (via masking) become trivially applicable again. The timing diagram of the error control block is shown in Fig. 3b. The errors are

[1] In a real-world scenario, the secret key would be embedded (in a shared manner if masked) and the random challenge would also be generated on-chip in case of protocols secure against man-in-the middle attacks.

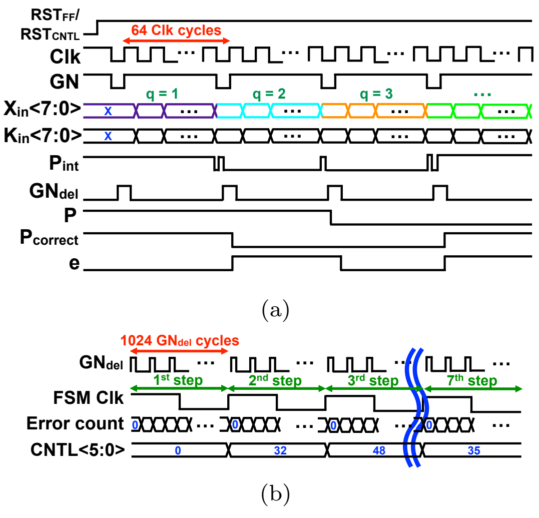


Fig. 3: Timing diagram of the LPPN processor (part a) and of its error control block (part b).

counted in the control block during one FSM clock cycle ($T_{FSM} = 1024 \times T_{GNdel}$). In a successive approximation scheme, the 6 *CNTL* bits are all reset to zero during the first step (i.e. the delay of the variable delay line is at its minimum which implies a high error count). At the end of the first step, the 10-bit error count is compared to the target count and the FSM decides to set the MSB of the *CNTL* to one during the second step in order to increase the delay of the *GNdel* edge. The following steps are conducted in the same way where the FSM sets the corresponding *CNTL* bit to one and decides whether to reset the previous bit to zero or not until all bits of the *CNTL* signal are computed. This in total requires 7 steps (1 step where all bits are reset to zeros and 6 steps to compute the values of the 6 *CNTL* bits) as shown in Fig. 4. After the *CNTL* bits are computed, the error controller goes into lock state keeping the same values of the *CNTL* bits as in the last step, and the actual authentication can take place.

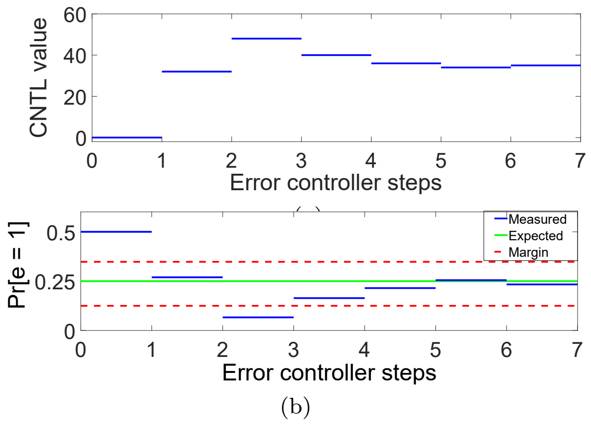
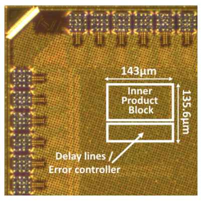


Fig. 4: (a) Error measured by the LPPN controller in 7 successive steps thanks to the CNTL signal and (b) the corresponding error probability.



Area	19,400 μm^2 16,333 GE
Energy per HB auth.	1 μJ
Power @ 0.45V	20.16 μW
Latency	52 ms
Throughput	156.25 kbps

Fig. 5: Die Micrograph of the LPPN processor and summary of its performance.

The LPPN processor was designed using a 28nm FDSOI process using CMOS logic gates. Figure 5 shows the micrograph of the LPPN processor and a summary of its performance results.

3 Preliminaries

Our security evaluations are mostly based on IT metrics which are aimed to bound the worst-case security level of an implementation. We next recall the specific metrics we use and how they can be computed.

3.1 Notations

We use capital letters for random variables and lower cases for samples of these random variables. Let the discrete random variable X denote the intermediate value of an LPPN stage while the discrete leakage variable is given as L (assuming that it is the output of a sampling device such as an oscilloscope). The conditional Probability Mass Function (PMF) is denoted as $p(L = l|X = x)$ and simplified as $p(l|x)$.

Since the true distribution of the leakage function is unknown, we can only estimate it by sampling in order to produce an estimated leakage model dataset given by M and a testing dataset denoted by T. The sampling processes are given as $M \overset{n}{\leftarrow} p(l|x)$ and $T \overset{n_t}{\leftarrow} p(l|x)$ for the model and testing sets, respectively, where n and n_t ($n(x)$ and $n_t(x)$) are the number of i.i.d. samples measured (per intermediate value x) for the corresponding (model and test) datasets.

3.2 Evaluation metrics

The challenge in side-channel security evaluation is to bound the data complexity of an attack. Therefore, an evaluator can use the *Mututal Information* (MI) between the leakage and a sensible variable since it is directly related to data complexity of the best attack [9, 6]. However, computing this metric requires the knowledge of the exact leakage distribution which is in practice never known to an evaluator. In the following, we recall two metrics that have been analyzed in [5], namely the *Perceived Information* (PI) and the *Hypothetical Information* (HI), that are respectively lower and upper bounds one of the other.

The first metric is related to the actual information that an adversary will be able to extract from the leakages with a given estimated leakage model. The PI is computed by sampling according to

$$\widehat{\mathrm{PI}}_n(X;L) = \mathrm{H}(X) + \sum_{x\in\mathcal{X}} \mathrm{p}(x) \sum_{i=1}^{n_t(x)} \frac{1}{n_t(x)} \cdot \log_2 \tilde{\mathrm{m}}_n(x|l_i), \tag{1}$$

where $\mathrm{H}(X)$ denotes the Shannon's entropy of the secret variable X. It is computed in two phases. First, a leakage model $\tilde{\mathrm{m}}_n(x|l_i)$ is built with n measurements to estimate the probability of a secret variable x given a leakage sample. Second, the true leakage distribution is sampled $n_t(x)$ times to approximate the extracted information thanks to the model. Because some data are needed to build the model and others to sample the true leakage distribution, this metric is typically computed with cross-validation. In the following experiments, we performed a 10-fold cross-validation.

The second metric is given by

$$\widehat{\mathrm{HI}}_n(X;L) = \mathrm{H}(X) + \sum_{x\in\mathcal{X}} \mathrm{p}(x) \sum_{l\in\mathcal{L}} \tilde{\mathrm{m}}_n(l|x) \cdot \log_2 \tilde{\mathrm{m}}_n(x|l). \tag{2}$$

In the case of an exhaustive model estimation (based on non-parametric estimator such as histograms or kernels), both metrics converge towards the MI value and are denoted as ePI and eHI. Indeed, the leakage model will converge to the true distribution allowing to extract all the available information. The eHI is its upper bound and can be used for worst-case evaluation.

However, such an exhaustive estimation converges slowly especially in multivariate settings. In the latter case, prior hypothesis about the leakage distribution can be made. Typically, it is estimated using Gaussian assumption to obtain gPI and gHI possibly introducing modeling errors. Therefore, these two metrics may converge towards something smaller than the MI. Now, the gHI can be computed in two different ways. First, it can be estimated directly by sampling similar to PI except that no cross-validation is needed. Indeed, it does not compare a leakage model to a true distribution, but to the model itself. It expresses the amount of information that would be available if the true distribution was the one of the model, given in this case that the model is Gaussian. The second method to estimate the gHI is given by the analytic expression

$$g\mathrm{HI}(X;L) = g\mathrm{H}(M(X)) + g\mathrm{H}(L) - g\mathrm{H}(M(X);L), \tag{3}$$

where $M(X)$ represents the model of X (ie., the mean of the leakage of X) and the entropy of a Gaussian variable Y is obtained with

$$g\mathrm{H}(Y) = \frac{\frac{1}{2}\log(\mathsf{det}(2\pi e\Sigma))}{\log(2)}, \tag{4}$$

with Σ its covariance matrix. As a result, evaluations are typically performed in two steps:

1. Computing gPI and eHI/ePI in an univariate setting independently for all the investigated points of interest which from here on will be referred to as dimensions.[2] If they converge to the same value, the evaluator concludes that the Gaussian assumption does not introduce (significant) modeling errors.
2. Estimating the gHI jointly on all the dimensions. The obtained value is not an upper bound but it should be close to worst-case given that the Gaussian assumption is reasonable.

In this work, these two steps are performed respectively in Sections 4.1 and 4.2. For the sake of completeness, we also perform a Gaussian template attack and report its success rate results in Section 4.3.

3.3 Evaluation settings

Case study. We study the information leaked from all the intermediate and output stages of the LPPN implementation, namely, the output of the 512-bit AND stage, the output of the parallel $XORp_i$, the output of the serial $XORs_i$, where $i \in [1, 6]$ and the output parity bit P. For comparison purposes, we analyze the information leakage based on a single bit at a time since the last targets (in the serial stages) are only bits.

Measurement setup. Measurements were conducted on an LPPN chip operating at 10MHz using a 0.6V supply from the MAX8902B low-noise linear regulator. All inputs (and outputs) were provided (captured) by an Intel Cyclone-IV FPGA, mounted on a Terasic DE0-Nano board. The current traces were probed by a Tektronix CT-1 current probe and sampled at 2GS/s using the Lecroy HRO 66 ZI oscilloscope.

[2]The dimensions considered are most informative based on the Mangard's Signal to Noise Ratio (SNR) [18].

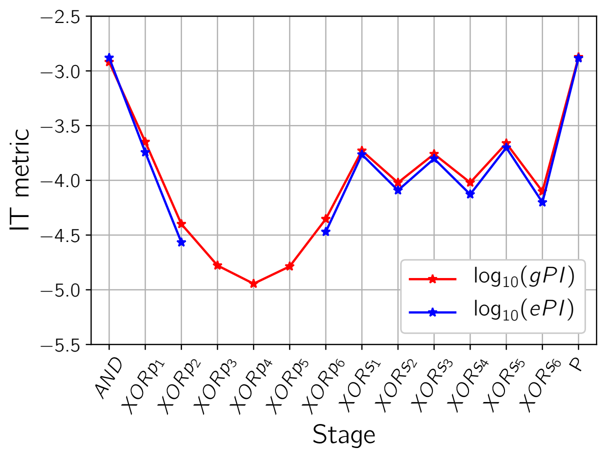


Fig. 6: PI using empirical histograms and Gaussian assumption across all LPPN stages (taking the maximum over all bits in each stage).

4 Side-channel analysis of LPPN

In this section, we analyze the side-channel leakage of the LPPN across its different stages in order to identify the best target using the gPI which is the PI corresponding to a Gaussian leakage model and the eHI / ePI corresponding to histograms as upper / lower bounds to MI, respectively. We first start by a univariate analysis. Then we study the side-channel security in front of a multivariate adversary exploiting jointly multiple leakage samples. At the end, we perform a Gaussian template attack against the most suitable target.

4.1 Univariate information theoretic analysis

In the univariate setting, we choose our most informative sample based on the SNR. First, we start our analysis by using the ePI avoiding any assumptions on the leakage model and also the gPI where we assume the leakages are Gaussian which allows a faster convergence. The ePI and the gPI computed over 2 million measurements for the most informative leakage bit of each stage of the LPPN implementation (i.e. the worst case) is shown in Fig. 6 starting from the output of the 512 bit AND stage, following with the successive parallel XOR ($XORp$) stages and the serial XOR ($XORs$) ending with the final output bit P.

In this experiment, we considered random inputs and a randomly selected fixed key. Before discussing our observations, we list the effects taking place in the LPPN processor that affect its leakage:

– The **algorithmic noise**. At any given stage, we compute the IT metric of the most leaking gate where the remaining parallel gates consume power and contribute to the algorithmic noise.
– The **jitter**. In the LPPN processor implementation, the internal bits are computed in a large combinatorial logic network (and not in synchronous components) where the jitter is accumulated.
– The **impact of the serialiser implementation** on the first computation stages.[3]

The important observations are:

– First, we observe that both the ePI and the gPI are quite close for most stages which indicates that the considered measured samples are fairly Gaussian.
– For some of the inner parallel XOR stages, the ePI could not be computed with 2 million measurements (i.e. its values are still negative), while the gPI can be computed (indicating a lack of samples to accurately estimate the non-parametric ePI).
– For the first parallel XOR stages, the information leakage is decreasing mainly because the target bits are manipulated in a large combinatorial logic network (where the parallel computations of other bits act as algorithmic noise) and not in synchronous components. There, the leakage associated to the target bit is not synchronized because of jitter accumulation across the XOR layers. Also, the algorithmic noise impacts the side-channel leakage, even though it is decreasing along the computation path. For the last XOR stages, the jitter accumulation still increases but the number of parallel computations decreases to the point that the reduction of the algorithmic noise outweighs the increasing jitter effect. Hence the information leakage increases.
– Finally, the most informative bits are the output of the AND stage and the output bit P. For the output P bit, we observe maximum leakage thanks to the (expected) fact that the algorithmic noise is minimum. For the AND stage, this information leakage is for the worst-case bits. In fact, we did not expect the AND stage to leak as high as the output bit because of the large number of parallel computations of other bits. However, we observed that among all the AND bits, there are peaks at every $\frac{N}{8}$ bits (N is the total number of bits per stage). For the other bits the amount of information is in the order of 10^{-5} as shown in Fig. 7. This is presumably due to our implementation of the input serializer stage that shifts the input and key through a shift register and loads them to the LPPN's inner product block every 64 clock cycles, where the combinatorial computation takes place. In other words, the most leaky bits are the last loaded ones, which ex-

[3]The serializer is implemented such that 16 banks of 64 bits (representing the input and key) are shifted before being loaded to the AND stage inputs.

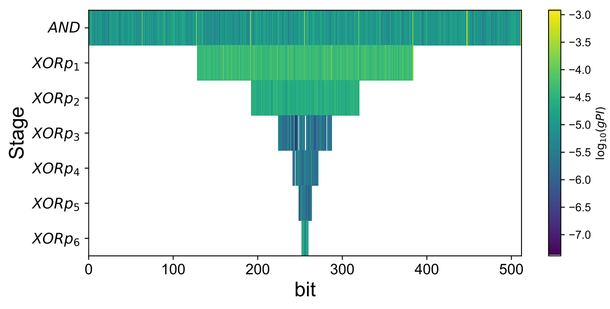


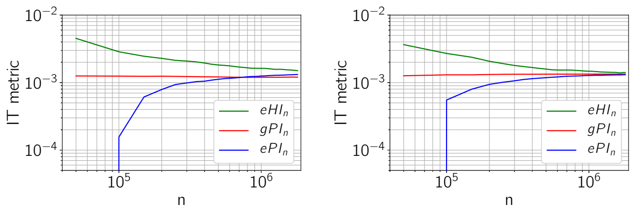
Fig. 7: Color bit map of the PI using Gaussian templates across all parallel the LPPN stages.

plains the peaks. We note that this issue could be avoided by implementing the serialiser differently. For example, one could shift the 512 input bits and load them once before the actual computation takes place. Such an improvement, that we leave as an interesting direction for further investigations, would make the leakage of all the bits of the AND stage more similar and close to their best-case value.

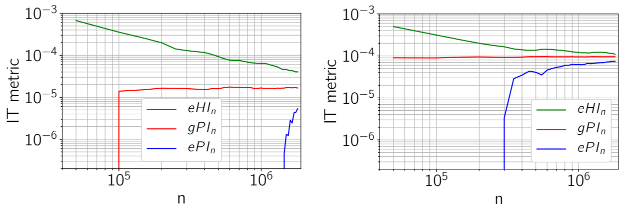
Given the fact that both the worst case output of the AND stage and the output bit P provide maximum leakage, both of them could serve as good targets to mount different types of attacks. The output of the AND stage makes a good target for Gaussian template attacks, whereas the last output bit cannot be attacked with such a simple divide-and-conquer attack, but with more challenging algebraic attacks [3, 2, 10]. Since Gaussian template attacks are easier to mount, we conclude that the input stage of LPPN is the most suitable target to attack. We note also that this conclusion is amplified by the fact that for this first stage, multi-bit targets could be considered (hence reducing the amount of algorithmic noise).

Next, we investigate whether the information leakage estimations we compute have converged, whether the bounds are tight and whether the Gaussian assumption is good enough. In Fig. 8 we plot the before-mentioned IT metrics in function of the number of traces in the profiling set n. First, we can observe that for the two good targets the leakage estimations converge well and the eHI/ePI bounds are tight (see Fig. 8 (a) and (b)). We further note that the ePI is slightly higher than the gPI in the AND stage indicating that the leakages are not perfectly Gaussian. However, this assumption error is small ($< 10^{-3}$) and we can therefore assume that the Gaussian assumption is not inducing strongly overstated security claims.

Second, in Fig. 8 (c) and (d) we plot the IT metrics of arbitrarily chosen intermediate bits in the parallel and serial stages, respectively. Here we see the eHI/ePI bounds are less tight specially in the inner parallel XOR



(a) AND stage: b511. (b) Output P.



(c) $XORp_3$ stage: b14. (d) $XORs_4$ stage.

Fig. 8: Convergence of the information theoretic metrics of maximum leaking bits in (a) the parallel, (b) serial stages and arbitrarily chosen intermediate bits in (c) the parallel, (d) serial stages in a univariate setting.

stages, which is due to their slower convergence and indicates the need for more measurements in order to evaluate these targets. We still get good convergence of the IT metrics in the serial stages and the Gaussian assumption is good enough in this case as well.

4.2 Multivariate information theoretic analysis

We extend our observations to the multivariate setting by following the guidelines of Section 3.2. That is, we now assume a more powerful adversary exploiting jointly multiple points within the leakage traces. We first compute both the ePI and the gPI in the univariate setting for all the time samples (corresponding to 350 leakage points), for the best target bit of the AND stage. This shows how good is the Gaussian assumption for all samples (and not just the most informative one based on SNR as performed in the previous section). As highlighted in Fig. 9, the univariate Gaussian assumption remains reasonable everywhere, since the ePI is close to the gPI in nearly all the most informative time samples. Hence, we conclude that the multivariate gHI and gPI should provide a good indication of the worst-case security level of our LPPN prototype.

In Fig. 10, we plot both the multivariate gPI and the multivariate gHI computed directly (using the sampling-based method) and with the approximate formula, for various number of dimensions. For this purpose, we sort the leakage time samples based on the amount of information extracted and always consider the most informative samples first. It leads to the following observations:

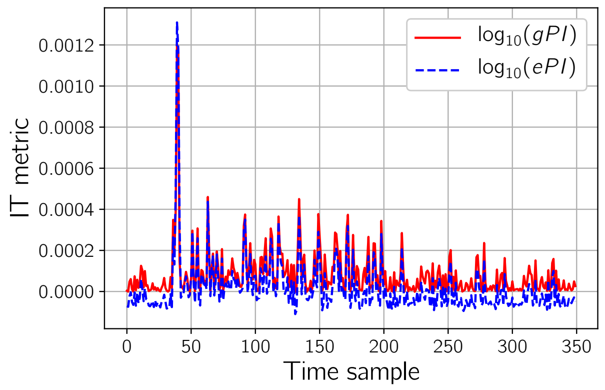
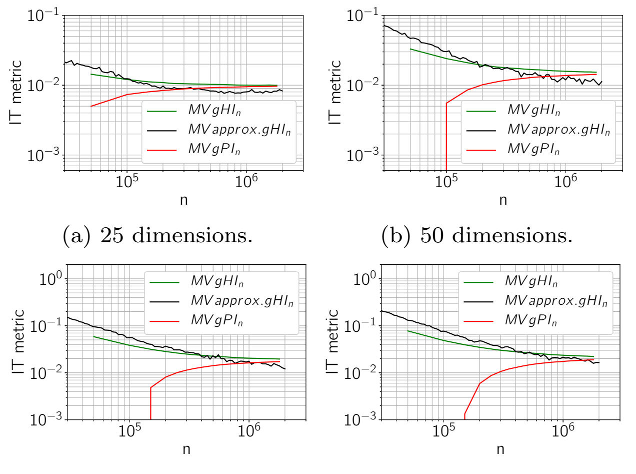


Fig. 9: Perceived information using Gaussian templates and empirical histograms for all 350 leakage points with 2 million traces.

- For the computation of the gHI, the multivariate approximation does not always converge to the correct value (which is consistent with the results in [20] showing that even in the univariate case, such an approximation is not always accurate, especially for low noise levels). Since the direct estimation of the gHI (i.e. by sampling, without cross-validation) is also converging faster, there is no incentive to use this approximation. The latter brings an interesting practical complement to the proposals of [5].
- As expected, the information leakage increases as more dimensions are investigated. For 90 dimensions the gPI is about 2×10^{-2} which is larger than the univariate gPI $(1, 25 \times 10^{-3})$ by a factor 16.
- One can also notice that by moving from 75 to 90 dimensions, the estimated gHI and gPI are converging to a similar value around 2×10^{-2}. This means that the 15 additional leakage samples used do not lead to significant increase in information.



(a) 25 dimensions. (b) 50 dimensions. (c) 75 dimensions. (d) 90 dimensions.

Fig. 10: Multivariate gHI computed directly (by sampling) and using the approximating formula and multivariate gPI, for several number of dimensions.

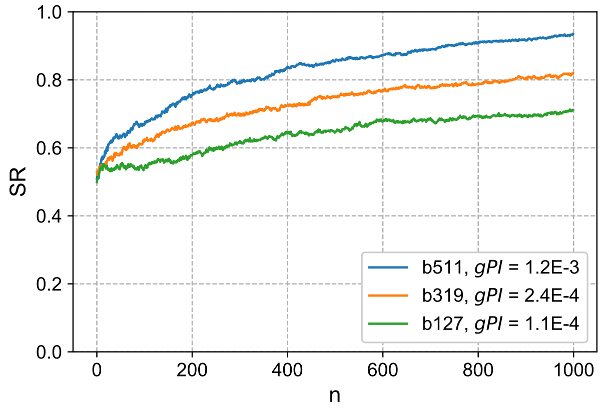


Fig. 11: Success rate of Gaussian template attack against different bits at the output of the AND stage.

It is worth noting that increasing the number of dimensions beyond 90 starts to impact the gPI negatively (indicating that more traces would be needed to estimate such an adversary). Yet, since our analysis suggests that dimensions beyond the 75th do not contain useful information anymore, there is no incentive for exploiting such larger number of dimensions.

4.3 Gaussian template attack

We finally confirm the previous information theoretic investigation by mounting a Gaussian template attack on the most informative target (the output of the AND stage). It is evident from Fig. 11 that the most leaking key bits can be successfully recovered with around 1000 traces, while less leaking bits require more traces. As expected from [9, 6], the number of samples required to perform a successful template attack is correlated with the gPI computed in Section 4.2.

5 Conclusions and open problems

The key-homomorphic structure of an LPPN implementation makes it relatively easy to evaluate against (close to) worst-case side-channel attacks. From our experiments, we can conclude that a mixed (parallel-serial) architecture inherently provides levels of side-channel resistance such that masking will be effective. Indeed, even strong multivariate adversaries can only extract information values significantly lower than one (which can therefore be amplified).

Yet, as usually observed for masking, additional noise addition can be a good solution to reach higher security levels, especially as the number of shares in a masking schemes increase (since this noise will be raised to a power corresponding to the number of shares).

Eventually, our experiments also indicate that for the considered architecture and technology, the leakage samples corresponding to the early stages of the LPPN computations are the sweet spots for side-channel attacks, as they lead to high(er) information leakages and are exploitable via easy-to-mount DPA attacks.

As usual for such low-level investigations of physical security, our conclusions (and especially their quantitative part) are admittedly technology-dependent. So while the main positive features of the LPPN assumption (e.g. the simplicity to mask it) and our evaluation methodology should remain valid in general, the investigation of other technologies than the 28nm FDSOI one we analyzed is an interesting research topic.

Acknowledgements François-Xavier Standaert is a senior associate researcher of the Belgian Fund for scientific research. This work has been funded in parts by the ERC project SWORD (grand number 725725) and by the UCLouvain ARC project NANOSEC.

References

1. Armknecht, F., Hamann, M., Mikhalev, V.: Lightweight authentication protocols on ultra-constrained RFIDs - myths and facts. In: N. Saxena, A.R. Sadeghi (eds.) Radio Frequency Identification: Security and Privacy Issues, pp. 1–18. Springer International Publishing, Cham (2014)
2. Belaïd, S., Coron, J., Fouque, P., Gérard, B., Kammerer, J., Prouff, E.: Improved side-channel analysis of finite-field multiplication. In: CHES, Lecture Notes in Computer Science, vol. 9293, pp. 395–415. Springer (2015)
3. Belaïd, S., Fouque, P., Gérard, B.: Side-channel analysis of multiplications in GF(2128) - application to AES-GCM. In: ASIACRYPT (2), Lecture Notes in Computer Science, vol. 8874, pp. 306–325. Springer (2014)
4. Berti, F., Standaert, F.X.: An analysis of the learning parity with noise assumption against fault attacks. In: CARDIS, pp. 245–264 (2016)
5. Bronchain, O., Hendrickx, J.M., Massart, C., Olshevsky, A., Standaert, F.X.: Leakage certification revisited: Bounding model errors in side-channel security evaluations. In: A. Boldyreva, D. Micciancio (eds.) Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I, Lecture Notes in Computer Science, vol. 11692, pp. 713–737. Springer (2019). DOI 10.1007/978-3-030-26948-7_25. URL https://doi.org/10.1007/978-3-030-26948-7_25
6. de Chérisey, E., Guilley, S., Rioul, O., Piantanida, P.: Best information is most successful mutual information and success rate in side-channel analysis. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2019**(2), 49–79 (2019)
7. Cnudde, T.D., Bilgin, B., Gierlichs, B., Nikov, V., Nikova, S., Rijmen, V.: Does coupling affect the security of masked implementations? In: S. Guilley (ed.) Constructive Side-Channel Analysis and Secure Design - 8th International Workshop, COSADE 2017, Paris, France, April 13-14, 2017, Revised Selected Papers, Lecture Notes in Computer Science, vol. 10348, pp. 1–18. Springer (2017). DOI 10.1007/978-3-319-64647-3_1. URL https://doi.org/10.1007/978-3-319-64647-3_1
8. Cnudde, T.D., Ender, M., Moradi, A.: Hardware masking, revisited. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2018**(2), 123–148 (2018). DOI 10.13154/tches.v2018.i2.123-148. URL https://doi.org/10.13154/tches.v2018.i2.123-148
9. Duc, A., Faust, S., Standaert, F.X.: Making masking security proofs concrete - or how to evaluate the security of any leaking device. In: E. Oswald, M. Fischlin (eds.) Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I, Lecture Notes in Computer Science, vol. 9056, pp. 401–429. Springer (2015). DOI 10.1007/978-3-662-46800-5_16. URL https://doi.org/10.1007/978-3-662-46800-5_16
10. Dziembowski, S., Faust, S., Herold, G., Journault, A., Masny, D., Standaert, F.X.: Towards sound fresh re-keying with hard (physical) learning problems. In: M. Robshaw, J. Katz (eds.) Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II, Lecture Notes in Computer Science, vol. 9815, pp. 272–301. Springer (2016). DOI 10.1007/978-3-662-53008-5_10. URL https://doi.org/10.1007/978-3-662-53008-5_10
11. Eisenbarth, T., Kumar, S.S., Paar, C., Poschmann, A., Uhsadel, L.: A survey of lightweight-cryptography implementations. IEEE Design & Test of Computers **24**(6), 522–533 (2007). DOI 10.1109/MDT.2007.178. URL https://doi.org/10.1109/MDT.2007.178

12. Gaspar, L., Leurent, G., Standaert, F.X.: Hardware implementation and side-channel analysis of lapin. In: CT-RSA, pp. 206–226 (2014)
13. Grosso, V., Standaert, F.X., Faust, S.: Masking vs. multiparty computation: how large is the gap for AES? J. Cryptographic Engineering **4**(1), 47–57 (2014). DOI 10.1007/s13389-014-0073-y. URL https://doi.org/10.1007/s13389-014-0073-y
14. Joye, M., Tunstall, M. (eds.): Fault Analysis in Cryptography. Information Security and Cryptography. Springer (2012). DOI 10.1007/978-3-642-29656-7. URL https://doi.org/10.1007/978-3-642-29656-7
15. Kamel, D., Bellizia, D., Standaert, F.X., Flandre, D., Bol, D.: Demonstrating an LPPN processor. In: Proceedings of the 2018 Workshop on Attacks and Solutions in Hardware Security, ASHES '18, pp. 18–23. ACM, New York, NY, USA (2018). DOI 10.1145/3266444.3266445. URL http://doi.acm.org/10.1145/3266444.3266445
16. Kamel, D., Standaert, F.X., Duc, A., Flandre, D., Berti, F.: Learning with physical noise or errors. IEEE Transactions on Dependable and Secure Computing pp. 1–1 (2018). DOI 10.1109/TDSC.2018.2830763
17. Levi, I., Bellizia, D., Standaert, F.X.: Reducing a masked implementation's effective security order with setup manipulations and an explanation based on externally-amplified couplings. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2019**(2), 293–317 (2019). DOI 10.13154/tches.v2019.i2.293-317. URL https://doi.org/10.13154/tches.v2019.i2.293-317
18. Mangard, S.: Hardware countermeasures against DPA ? a statistical analysis of their effectiveness. In: CT-RSA (2004)
19. Mangard, S., Oswald, E., Popp, T.: Power analysis attacks - revealing the secrets of smart cards. Springer (2007)
20. Mangard, S., Oswald, E., Standaert, F.X.: One for all - all for one: unifying standard differential power analysis attacks. IET Information Security **5**(2), 100–110 (2011)
21. Mangard, S., Popp, T., Gammel, B.M.: Side-channel leakage of masked CMOS gates. In: A. Menezes (ed.) Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings, Lecture Notes in Computer Science, vol. 3376, pp. 351–365. Springer (2005). DOI 10.1007/978-3-540-30574-3_24. URL https://doi.org/10.1007/978-3-540-30574-3_24
22. Nikova, S., Rijmen, V., Schläffer, M.: Secure hardware implementation of nonlinear functions in the presence of glitches. J. Cryptology **24**(2), 292–321 (2011). DOI 10.1007/s00145-010-9085-7. URL https://doi.org/10.1007/s00145-010-9085-7
23. Pietrzak, K.: Cryptography from learning parity with noise. In: SOFSEM, pp. 99–114 (2012)
24. Regazzoni, F., Breveglieri, L., Ienne, P., Koren, I.: Interaction between fault attack countermeasures and the resistance against power analysis attacks. In: Joye and Tunstall [14], pp. 257–272. DOI 10.1007/978-3-642-29656-7_15. URL https://doi.org/10.1007/978-3-642-29656-7_15
25. Schneider, T., Moradi, A., Güneysu, T.: ParTI - towards combined hardware countermeasures against side-channel and fault-injection attacks. In: CRYPTO, pp. 302–332 (2016)
26. Standaert, F.X., Malkin, T., Yung, M.: A unified framework for the analysis of side-channel key recovery attacks. In: A. Joux (ed.) Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings, Lecture Notes in Computer Science, vol. 5479, pp. 443–461. Springer (2009). DOI 10.1007/978-3-642-01001-9_26. URL https://doi.org/10.1007/978-3-642-01001-9_26