On the success rate of simple side-channel attacks against masking with unlimited attack traces

Aymeric Hiltenbrand[1], Julien Eynard[2], and Romain Poussier[3]

[1] Univ Rennes, CNRS, Inria, IRISA Rennes, France
[2] Rambus Inc. jeynard[alt]rambus.com
[3] ANSSI
{name.surname}@{inria.fr, ssi.gouv.fr}

Abstract. Side-channel attacks following a classical differential power analysis (DPA) style are well understood, along with the effect the masking countermeasure has on them. However, simple attacks (SPA) where the target variable does not vary thanks to a known value, such as the plaintext, are less studied. In this paper, we investigate how the masking countermeasure affects the success rate of simple attacks. To this end, we provide theoretical, simulated, and practical experiments. Interestingly, we will see that masking can allow us to asymptotically recover more information on the secret than in the case of an unprotected implementation, depending on the masking type. We will see that this is true for masking encodings that add non-linearity with respect to the leakages, such as arithmetic masking, while it is not for Boolean masking. We believe this context provides interesting results, as the average information of arithmetic encoding is proven less informative than the Boolean one.

Keywords: SPA, masking, success rate.

1 Introduction

As opposed to classical cryptography, where an adversary has only access to the inputs and outputs of a primitive viewed as a black box, Side-Channel Attacks (SCAs) use additional information. This information comes from the physical implementation of a mathematical algorithm, and is referred to as leakage. This includes as an example the timing information [13], the power consumption [14], or the electromagnetic radiations [10]. This study concerns the last two types of leakages, where the adversary collects the corresponding information for each execution of a primitive, denoted as a power/electromagnetic trace.

Independently of the type of leakage used, SCAs can be divided into two categories: differential or simple. The former one, denoted as DPA, corresponds to the most usual case, where an adversary targets a sensitive variable that both depends on the secret and a known value. This can be the case when targeting the output of the AES Sbox for instance. Each trace comes from an AES execution, where the known plaintext is varying. Note that the "differential" term of DPA here refer to the dependency to a known *varying* variable, and not to

Kocher's DPA [14]. The second type of attack, denoted as SPA, corresponds to the case where the target variable does not depend on any known varying value. This can happen for example when targeting the key scheduling of the AES [27], the key processing in Ascon [8], some Kyber [1] variables such as the output of the binomial sampler or a bus data transfer. The adversary can measure several executions of the algorithm, but the leakages will not include any variation regarding a known value. The existence or the lack of such dependency with a known value has a significant impact on these two types of attacks. For DPA, the use of an approximate leakage model can still allow to recover the secret, thanks to this dependency along with (e.g.) the non-linearity of the Sbox [20]. However, in the SPA context, the inaccuracy of the leakage model might have a significantly more negative impact. We further provide more description of the two attack scenarios in Section 2.6.

To protect against SCAs, independently of their nature, masking is a commonly used countermeasure [12]. It consists of splitting any sensitive value into independent shares using randomness, such that the combination of all shares is equal to the original sensitive value. For instance, the combination can be an addition over $\mathbb{F}_{2^k}$, such as Boolean masking [23], or an addition over $\mathbb{Z}_p$, as for arithmetic masking [18]. From a security perspective, the required number of traces to break an implementation increases exponentially with the number of shares, with respect to the noise level.

In this paper, we study the context of profiled SPAs against unprotected and 2-share masked (Boolean and arithmetic) implementations. For these different types of protection, we focus on the asymptotic success rates, denoting the probability of recovering a secret given an infinite number of attack traces. For a DPA, the asymptotic success rate is usually one even with a mediocre leakage model. This is true independently of the protection technique, as long as the attack order matches the masking level [15] and that the target variable has non-linearity such as an Sbox output. However, for SPA, we show that the asymptotic success rate can be lower, even assuming an infinite number of attack traces. Moreover, We further study the context of an imperfect profiling and its effect on the success rate, as real profiled attacks can never perfectly estimate the model. We believe that the specific context of SPA in combination with masking and asymptotic success rate is not well studied and can lead to counter-intuitive observations. More specifically, the contributions of this paper can be summarized as follows:

- First, we study the (unrealistic) context where the actual leakage function is known to the adversary, for both Hamming weight and bijective leakages. We show that arithmetic masking can help reaching a higher success rate than an unprotected implementation.
- Second, we study the more realistic context where an adversary approximates the leakage function through a profiling phase. This introduces incorrect profiling, where some classes can never be recovered. We show that this has more impact on non-linear encoding of arithmetic masking, even if it can reach higher success rates. Surprisingly, we also put in evidence cases where

the asymptotic success rate is higher for an implementation with Boolean masking than for an unprotected one, for a similar profiling strength.
– Finally, we validate the theoretical and simulated results using actual measurements.

The rest of the paper is organized as follows. First, Section 2 introduces the notions and background that are necessary for the rest of the paper, along with our simulated experimental setup. Second, Section 3 studies the unrealistic context where the adversary has perfect knowledge of the leakage function. Then, Section 4 shows what happens in a more realistic context, where an attacker can only estimate the actual leakage function through an actual profiling phase. Finally, Section 5 confirms the previous results using real leakages.

2 Background

2.1 Notations

We use capital letters for random variables and small caps for their realizations. We denote by $Pr[x]$ the probability a random variable X is equal to x. We denote the conditional probability of a random variable A given B with $Pr[A|B]$. We use a sans serif font for functions (e.g., F) and calligraphic fonts for sets (e.g., $\mathcal{A}$). Var denotes the variance, and HW denotes the Hamming weight function. We use bold notation for vectors (e.g. $\mathbf{v}$).

2.2 Leakage model

Let X denote some n-bit variable processed at a given time. In the rest of this paper, we will assume an additive Gaussian noise [25, 22] such that $L(X) = \mathsf{F}(X) + \mathcal{N}(0, \sigma^2_{noise})$. F denotes the deterministic part of the leakage, and $\mathcal{N}(0, \sigma^2_{noise})$ denotes the Gaussian noise with variance σ^2_{noise}. F can be generalized by the following linear polynomial [24] :

$$\mathsf{F}(x) = a + \sum_{i=0}^{n-1} a_i x_i \tag{1}$$

where x_i denotes the i-th bit of x. In the following, we will ignore the constant a without loss of generality. We will also study the common Hamming Weight leakage (HW) case where $\forall\ i, j\ \in [0, n-1]$, $a_i = a_j$.

2.3 Signal to Noise Ratio

The Signal-to-Noise Ratio [16] (SNR) is a commonly used information theory metric to measure the informativeness of the leakages. It is defined as the variance of F, representing the signal, divided by the noise level. Reusing the assumptions of the previous subsection, it can be computed as follows for a variable X:

$$\mathrm{SNR} = \frac{\mathsf{Var}_X(\mathsf{F}(X))}{\sigma^2_{noise}} \tag{2}$$

2.4 Success Rate

The success rate is a security evaluation metric that is directly related to the number of traces necessary to mount an attack [25]. That is, the success rate is defined as the probability that the actual secret is completely recovered after the attack. If we define by $\mathbb{1}_V$ the random variable that outputs 1 if we guess the sensitive n-bit variable V properly for a given attack and 0 elsewise, we can define the success rate SR as:

$$\mathrm{SR} = Pr\left[\mathbb{1}_V = 1\right] = \frac{1}{2^n} \sum_{v=0}^{2^n-1} Pr\left[\mathbb{1}_V = 1 | V = v\right] \tag{3}$$

Assuming that V has a uniform distribution. In our experiments, we will compute $Pr\left[\mathbb{1}_V = 1 | V = v\right]$ empirically by repeating an attack several times.

2.5 Masking

In this study, we will investigate the effect of the masking countermeasure in the context of SPA. Masking is a commonly used countermeasure to protect against side-channel attacks. The idea is to split any sensitive variable x into d shares s_i, $i \in [0, d-1]$, such that the knowledge of any $d-1$ shares does not give information on x. The corresponding circuit has to be modified into a masked circuit that computes over the shares instead of the original values, while still producing the correct result. There exist many masking types in the literature. In this study, we focus on Boolean and arithmetic ones.

Boolean masking : this is the first masking proposal [12], and probably the most popular one in symmetric cryptography, such as the AES or hash functions. The d shares s_i are computed such that $x = \bigoplus_0^{d-1} s_i$, the addition being over $\mathbb{F}_{2^k}$.

Arithmetic masking: instead of having x as an element of a field of characteristic two, arithmetic masking [18] is typically used when x belongs to a ring of modular integers, not necessarily modulo a prime number. In this case, the d shares s_i are computed such that $x = \sum_{i=0}^{d-1} s_i \bmod p$.

Security : the probing model [12] formalizes the security brought by the masking countermeasure. It ensures that the knowledge of any $d-1$ shares is independent of the original sensitive variable x. Upon a carefully masked implementation, the number of traces to recover a sensitive variable can grow exponentially with the number of shares, given enough noise [9].

2.6 Attack type

Independently from the fact that an attack is profiled or not, one can split the context of the attack into two cases based on whether the target sensitive

variable changes over the different runs of the algorithm, thanks to a varying known value.

DPA-like scenario: in this paper, we denote by DPA-like (or simply DPA) any side-channel attack that targets some intermediate variable V using several traces such that $V = \mathsf{G}(K, M)$, where G denotes some function, K represents a fixed secret variable to recover and M a *known varying* variable over different observations. For instance, attacking the Sbox output of the AES falls into this category if the plaintext is known and varies over several trace acquisitions. Note that the scenario is independent of the attack method. Indeed, (e.g.) Kocher's DPA [14], CPA [4], MIA [11], template [7], Machine Learning [5] (and so on) all fall within the DPA scenario in this case.

SPA scenario : As opposed to the DPA scenario defined above, in the SPA scenario, there is no dependency with a known varying value. An attack falls into this category if it targets some intermediate variable that only depends on the secret and potentially *unknown* varying values (e.g. randomness). Note that no assumption is made on the number of traces used for the attack. Attacking for example the key schedule of the AES, potentially using several observation traces, falls into this category as there is no variation at all. On the other hand, attacking the output of the AES Sbox with a single trace only or a fixed plaintext would also fall into this category. As for the DPA scenario, the category does not depend on the attack method.

Main differences and asymptotic success rate : This study will focus on the success rate of DPAs and SPAs when an infinite number of attack traces are used, which we refer to as *asymptotic success rate* and denote SR_∞ .

In this context, the DPA scenario is well understood in the literature [25, 26]. The number of traces required to break an implementation depends on $\frac{1}{SNR^o}$ where o is the masking order ($o = 1$ for unprotected), when the SNR is low enough. An attack would eventually succeed given enough attack traces, assuming that the leakage model is close enough to the real one.

This is however quite different for SPAs. The secret might not be recovered even using an infinite number of attack traces, which can happen for several reasons. Assuming a non-bijective leakage function (e.g. HW), some hypotheses cannot be distinguished. Moreover, if the model is not perfect, a candidate might always be misclassified.

The asymptotic SR difference between DPA and SPA is due to the existence of the known varying input in the DPA context. In case of a wrong model or a non-bijective leakage function, the help of the other classes and the (hopefully) non-linearity of the function $\mathsf{G}(K, M)$ (e.g. an Sbox) will still help the attack to eventually succeed (assuming the model is not drastically far off the actual one).

In this study, we are interested in the impact of masking on the asymptotic success rate of an SPA. Up to our knowledge, this setting exhibits interesting observations which have not been shown before. The main intuition is that masking a fixed variable will introduce (unknown) variability, where there would be none in the unprotected SPA context. An example could be a masked implementation

of the AES key schedule. We aim to study whether this added variability due to masking can improve the asymptotic complexity of a SPA, as opposed to an unmasked version of the algorithm. To answer this question, in the rest of this paper, we will compare the SPA results of an unmasked variable as opposed to its masked version. We will do so by considering two types of masking: Boolean and arithmetic. First, using a simulated setting, we will assume a perfect knowledge of the leakage function for the attacker, using both Hamming weight and a (more realistic) bijective linear leakage. Then, still using simulations, we will move to the more practical case where the actual leakages are not perfectly characterized. Finally, we will provide actual experiments to support our theoretical and simulated claims. From these settings, we will show that the question can be answered positively, showing that masking can increase SR_∞ of a SPA, especially for arithmetic masking.

Note that even if the asymptotic success rate of an attack can be improved thanks to masking, the main security property of the countermeasure still holds. That is, the required number of traces increases exponentially with the security order o [2]. However, this has no impact on this study as we are only interested in the asymptotic success rate given an infinite number of attack traces.

2.7 Simulation setup

To exhibit the properties of SPAs, Sections 3 and 4 will use the following simulated setting. We simulated leakages for four types of attacks: SPAs against unprotected, Boolean masking, arithmetic masking and finally a DPA against an unprotected implementation, used as a benchmark. All success rate calculations are done using 1000 repetitions. To speed up the computations, we targeted 4-bit values.To simulate the linear leakages introduced in Section 2.2, we randomly sampled each four coefficients a_i from a Normal distribution $\mathcal{N}(1, \sigma^2_{leakage})$ for each repetition of the experiment. The higher $\sigma^2_{leakage}$ is, the further from Hamming weight is the leakage function, as defined in 2.2, where a $\sigma^2_{leakage} = 0$ indicates an exact Hamming weight model. This allow us to control how far the leakage are from HW, and will use a value $\sigma^2_{leakage} = 10^{-4}$ in simulation as it will match its values obtained with real traces. The noise level σ^2_{noise} of the leakages is then set accordingly to match the desired SNR level.

Intermediate values: for the DPA, we used the S-Box S of the Present cipher [3] and computed the intermediate value $v = \mathsf{S}(s \oplus p)$ for random plaintext p. The adversary is provided with the noisy leakage of v. For the unprotected SPA, we directly leak the noisy leakage of the sensitive variable. For the SPA against masked implementation, we leak two samples, one for each share.

Profiling: except for Section 3 where we assume an adversary with perfect knowledge of the leakage function, we perform Gaussian template profiling. For each leakage sample (two in the case of masking), the adversary is provided with N_p leakages per value of the 4-bit variable, accounting for a total of $N_p \times 2^4$ traces for a 4-bit value. Note that this assumes a scenario where the adversary has access to the masking randomness for the profiling.

Attack: using the profiled or known model, we use a maximum likelihood approach for the attacker, which computes $Pr\left[\mathbf{l}|k\right] = \prod_{i=1}^{N_a} Pr\left[l_i|k\right]$ using N_a attack traces. In case of masking, we compute $Pr\left[l_i|k\right] = \sum_{s_j} Pr\left[l_i|k, s_j\right] Pr\left[s_j\right]$ where s_j denotes the randomness guess used for the sharing. The masking randomness is not known during the attack phase. We emphasize that the conclusions of this study are independent of the distinguisher, and that similar results would be obtained with any profiled attack.

3 Perfect profiling

In this section, we first study the (unrealistic) case where the adversary has perfect knowledge of the leakage function. More formally, during the attack, we assume the adversary knows exactly the deterministic part F of the leakages as introduced in Section 2.2. First, we will see the implication for a Hamming weight leakage function, before assuming more general linear leakages. For both cases, we study what SR_∞ can be achieved with and without the presence of masking in the context of SPA.

3.1 Hamming weight leakages

Assuming an adversary with the knowledge of $\mathsf{F} = \mathsf{HW}$, we are interested in evaluating SR_∞ for an unprotected, Boolean, and arithmetically masked value. We assume the attacker targets a uniformly distributed n-bit value v. We first show how the SR_∞ can be theoretically derived, before validating it using our simulated setting.

Unprotected: without masking countermeasure, the adversary directly observes $\mathsf{HW}(v) + \mathcal{N}(0, \sigma^2_{noise})$. In the perfect profiling setting, as we are interested with the asymptotic success rate, we can theoretically compute SR_∞ from Equation 3 assuming noiseless leakages, where V denotes the n-bit sensitive variable:

$$\begin{aligned} SR_\infty &= \frac{1}{2^n} \sum_{v=0}^{2^n-1} Pr\left[\mathbb{1}_V = 1 | V = v\right] \\ &= \frac{1}{2^n} \sum_{v=0}^{2^n-1} \sum_{l=0}^{n} Pr\left[\mathbb{1}_V = 1 | V = v, \mathsf{HW}(v) = l\right] Pr\left[\mathsf{HW}(v) = l\right] \qquad (4) \\ &= \frac{1}{2^n} \sum_{v=0}^{2^n-1} \sum_{l=0}^{n} \frac{1}{\binom{n}{l}} \frac{\binom{n}{l}}{2^n} = \frac{1}{2^n} \sum_{v=0}^{2^n-1} \frac{n+1}{2^n} = \frac{n+1}{2^n} \end{aligned}$$

Using $n = 4$ as an example, we would get $SR_\infty = 0.3125$. This success rate is computed assuming a uniformly distributed value v. Indeed, some values of v (e.g. $v = 0$ and $v = 15$) would be asymptotically recovered, while values such that $\mathsf{HW}(v) = 1$ would not.

Boolean masking: we now assume v is protected with 2-share Boolean masking. The adversary observes leakages on the shares s_0^v and s_1^v such that $v = s_0^v \oplus s_1^v$, which vary for each execution. That is, she is provided with $l_0 = \mathsf{HW}(s_0^v) + \mathcal{N}(0, \sigma^2_{noise})$ and $l_1 = \mathsf{HW}(s_1^v) + \mathcal{N}(0, \sigma^2_{noise})$. In this case, we can show that the asymptotic success rate is the same as for the unprotected case, which is derived from Proposition 1. The proof is provided in Appendix A.

Proposition 1. *Two unprotected values v_0 and v_1 have the same Hamming weight if, and only if, the two joint distributions of the Hamming weight of their shares are the same.*

Proposition 1 shows that, for noiseless Hamming weight leakages, the success rate for unprotected and Boolean masking are equivalent. The proof is finalized with Proposition 2.

Proposition 2. *The asymptotic success rate ($N_a = \inf$) of an attack with noiseless leakage given by a deterministic function F is the same as an attack with noisy leakages $\mathsf{F} + B$, where B denote the additive noise.*

In our context, if proposition 2 was false, then there would exist at least one secret value v such that its Hamming weight would not be recovered when adding noise to the leakages, while it would recovered with noiseless leakages. This cannot be in our asymptotic N_a setting, as masking can only increase the number of traces required to recover the secret, but not provide unconditional security. That is, adding noise will have no impact on the asymptotic success rate. As a result, in the context of Boolean masking, the SR_∞ is the same as for the unprotected version. Thus, the variability introduced by Boolean masking does not help nor hinder the attacker. A concrete example of this phenomenon is given in Table 1 for a 2-bit value v. The first column represents the value v written in binary, along with the corresponding Hamming weights. The remaining i columns correspond to all the possible sharings of v, also with their corresponding Hamming weights. We can see that for both $v = 01$ and $v = 10$ ($\mathsf{HW} = 1$), the corresponding possible Hamming weights of their shares are also equals up to a permutation. An attacker would thus not be able to distinguish these two values by observing their shared versions. However, this is not the case for $v = 00$ and $v = 11$, which have unique Hamming weights for their shared versions. These two values would thus be eventually recovered. We further note that this generalizes to any number of shares, and that the proof is independent of the bit size of the variable.

v	sharing 1	sharing 2	sharing 3	sharing 4
(00): HW=0	(00,00): HW=(0,0)	(01,01): HW=(1,1)	(10,10): HW=(1,1)	(11,11): HW=(2,2)
(01): HW=1	(00,01): HW=(0,1)	(01,00): HW=(1,0)	(10,11): HW=(1,2)	(11,10): HW=(2,1)
(10): HW=1	(00,10): HW=(0,1)	(01,11): HW=(1,2)	(10,00): HW=(1,0)	(11,01): HW=(2,1)
(11): HW=2	(00,11): HW=(0,2)	(01,10): HW=(1,1)	(10,01): HW=(1,1)	(11,00): HW=(2,0)

Table 1: Boolean sharings of a 2-bit value v, with their Hamming weight leakages. The first column corresponds to v, and the others show the possible sharings.

Arithmetic masking : we now move to the context of a 2-share arithmetic masking, where the n bits sensitive value v is masked using $v = s_0^v + s_1^v \bmod 2^n$, v_i being the shares. Again, the adversary is provided with two leakages l_i equal to the noisy Hamming weights of both shares and uses the actual model $\mathsf{F} = \mathsf{HW}$ for the attack.

As opposed to Boolean masking, where the XOR is solely performed bitwise, the addition adds some non-linearity due to the propagation of the carry. Thanks to this property, if two unprotected values v_0 and v_1 have the same Hamming weight, the sets of the Hamming weight of their shares might differ. For this reason, the SR_∞ will be higher than the unprotected/Boolean masked versions. We illustrate this in Table 2, which shows again the possible arithmetic sharing of a 2-bit value v and their Hamming weights. As for Table 1, the first column represents the value v, and the remaining 4 columns correspond to a possible sharing of v. As opposed to Boolean masking, we can see that there is no sharing equal to another (in terms of Hamming weights) up to any permutation. In that case, all values of v could be recovered, meaning $SR_\infty = 1$. Note that using more than 2-bit variables would provide different results for arithmetic masking, as some sharing could still share equalities, which we will show below.

v	sharing 1	sharing 2	sharing 3	sharing 4
(00): HW=0	(00,00): HW=(0,0)	(01,11): HW=(1,2)	(10,10): HW=(1,1)	(11,01): HW=(2,1)
(01): HW=1	(00,01): HW=(0,1)	(01,00): HW=(1,0)	(10,11): HW=(1,2)	(11,10): HW=(2,1)
(10): HW=1	(00,10): HW=(0,1)	(01,01): HW=(1,1)	(10,00): HW=(1,0)	(11,11): HW=(2,2)
(11): HW=2	(00,11): HW=(0,2)	(01,10): HW=(1,1)	(10,01): HW=(1,1)	(11,00): HW=(2,0)

Table 2: Arithmetic sharings of a 2-bit value v, with their Hamming weight leakages. The first column corresponds to v, and the others show the possible sharings.

SR_∞ for arithmetic is less trivial to derive using formulas as opposed to the unprotected or Boolean masking cases. Yet, it can be calculated in practice by computing the number of sets of shares having the same Hamming weight values up to a permutation for different sensitive values, as in Table 2. The corresponding results are shown in Figure 1, where the SR_∞ in the Y-axis is given as a function of the bit size n in the X-axis. The blue (resp. green) curve corresponds to the unprotected/Boolean (resp. arithmetic) case.

We can see that the SR_∞ is much higher for arithmetic masking than for the unprotected case. In the latter, the asymptotic success rate decreases exponentially with the bit size of the variable. However, for arithmetic masking, SR_∞ is closer to a linear slope. From these observations, we can see that in the context of Hamming weight leakages (perfectly known to the adversary), arithmetic masking allows a SPA adversary to recover values that would otherwise not be recovered. However, the main property of masking still holds regarding the convergence rate. That is, reaching SR_∞ would be linear in the number of traces for the unprotected case, while it would be quadratic in the 2-share arithmetic one.

Note that while this study only focuses on 2-share versions of masking, we also computed SR_∞ for arithmetic masking for up to 8-bit values for a 3-share arithmetic masking. In this case, we still had SR_∞ equal to one for arithmetic masking. That is, adding more shares allows us to reach a higher asymptotic success rate for arithmetic masking.

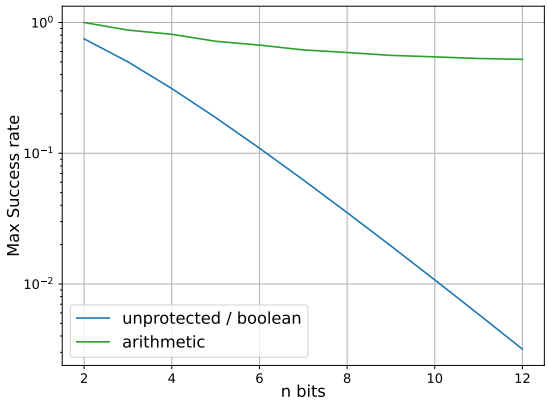

Fig. 1: Maximum success rate (Y-axis) as a function of the bit size n (X-axis) for Hamming weight leakage model, perfectly known to the adversary. The blue curve corresponds to the unprotected/Boolean masked cases. The green curve corresponds to arithmetic masking.

simulations : to validate the theoretical SR_∞ values derived above, we validate them using our simulated setting described in Section 3.2. As we assume perfect profiling, the actual probability density function (PDF), equal to a noisy Hamming weight function, is given to the adversary. For all cases, we ran experiments for 3 different SNRs levels equal to 0.1, 1, and 10, for 4-bit variables.

The results are shown in Figure 2, which shows the success rate (Y-axis) as a function of the number of attack traces (X-axis). All curves are computed using 2000 repetitions. The red curves correspond to the success rate of DPA against an unprotected variable as a benchmark. The other colors represent the implementation type (unprotected, Boolean, arithmetic), and the curve type represents the SNR. A plain line is a for a SNR of 10, dashes for SNR of 1, and dots for SNR of 0.1. As a first observation, we can see that the unprotected and the Boolean masked versions indeed do reach the same SR_∞ value, slightly above 0.3 which validates what was theoretically shown before. On the other side, we reach SR_∞ slightly above 0.8 for arithmetic masking, which again validates what was previously computed. We can also observe that reducing the SNR does not impact SR_∞ , but only the required number of attack traces to reach convergence, which is expected. Yet, we can see that for the unprotected version, there is a linear relationship between the required number of traces and the SNR, while there is one order of magnitude in the case of masking. This is due to the exponential benefit of masking [2], which is independent of the SR_∞ bound.

Finally, we can observe that while arithmetic has a higher SR_∞ than Boolean, it takes more attack traces to reach this convergence. This is because the mutual information is lower for a 2-share arithmetic masking than for a Boolean one. This was previously observed in [17], which means that a single trace observation is on average less informative for arithmetic masking, which explains why more are needed to reach convergence. This also shows that mutual information is not the right tool to study the asymptotic success rate.

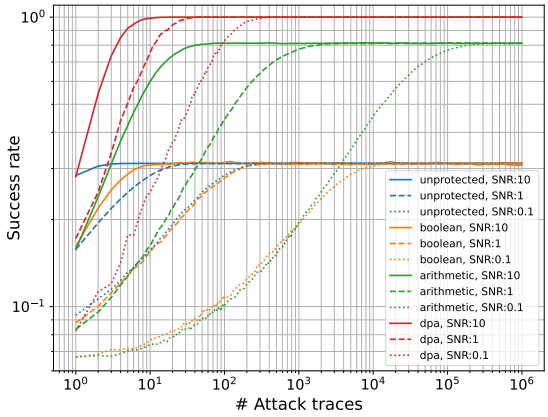

Fig. 2: Success rate (Y-axis) as a function of the number of attack traces (X-axis) for all 3 implementation types, computed with 2000 repetitions for perfectly known Hamming weight leakage. The curve color represents the implementation type (unprotected, Boolean, arithmetic, DPA), and the curve type represents the SNR. A plain line is a for a SNR of 10, dashes for SNR of 1, and dots for SNR of 0.1.

3.2 General linear leakages

The previous subsection assumed Hamming weight leakages. As this model is not bijective, it prevents SPA from reaching an asymptotic success rate of one. However, actual leakages are rarely linearly equivalent to the Hamming weight function, but rather close to it. That is, they would rather follow the general linear model given by Equation 1 in Section 2.2 with fairly close coefficients a_i. As a result, the actual leakage function is bijective and more or less close to the Hamming weight one. In this subsection, we study how this affects the success rate of a SPA, where the leakage function is still known to the adversary (perfect profiling).

Convergence of the success rate : as we now assume a bijective leakage function G known to the adversary, we will have $SR_\infty = 1$. However, the convergence speed to reach SR_∞ will highly depend on G. More precisely, following the leakage model of 2.2, the convergence speed will be impacted by $\sigma^2_{leakage}$ as defined

in . On one hand, if $\sigma^2_{leakage} \ll \sigma^2_{noise}$, then G will be hard to distinguish from a Hamming weight function. However, if $\sigma^2_{leakage}$ is larger, the bijectivity will be easier to capture. This is illustrated in Figure 3, which represents the PDFs of a 3-bit variable leaking with the linear model. The dashed curve shows the Gaussian mixture, which represents what an adversary sees using attack traces. We arbitrarily chose the linear coefficients to illustrate this property, without loss of generality. The right part of the figure, denoted as high noise, shows the case where $\sigma^2_{leakage} = 0.014 \ll \sigma^2_{noise} = 0.0016$. On the opposite, the left shows the case with $\sigma^2_{leakage} = 0.014$ and $\sigma^2_{noise} = 0.0324$, denoted as low noise. For the low noise case on the left, we can see that observing the mixture still allows visually differentiating between the 8 possible classes, leading to a quick convergence to SR_∞ . However, this is not the case for the high noise scenario, where observing the mixture would allow quickly differentiating the Hamming weight, but would struggle to distinguish the exact value of the Hamming weight equals 1 or 2. This would translates in a slower convergence to SR_∞ .

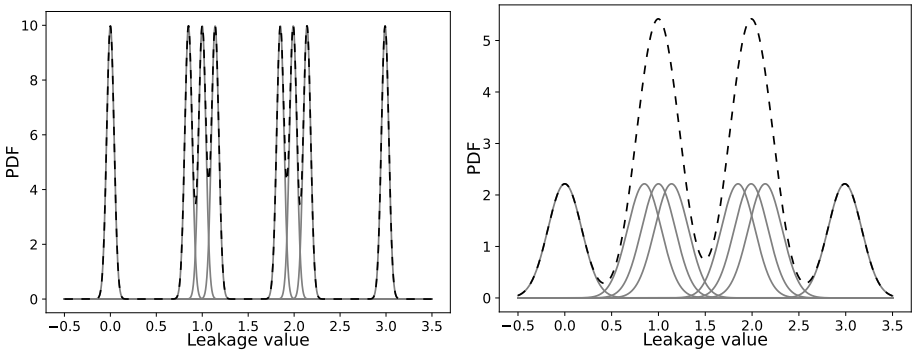

Fig. 3: Example of conditional PDFs (grey) and PDF mixture (black, dashed) for the bijective model in a low noise scenario (left) and in a high noise scenario (right). The values have been chosen to emphasize these differences.

Simulations : we used our simulated setting to validate the previous observation on the convergence for the different versions (unprotected, Boolean, and arithmetic masking, as well as unprotected DPA). We simulated the deterministic linear leakage function G using $\sigma^2_{leakage} = 0.0001$, as described in Section 3.2. The results are shown in Figure 4. It shows the success rate (Y-axis) for different numbers of attack traces (X-axis). The red curve represents the unprotected DPA as a benchmark, and the other colors represent an implementation type, while the type of the curve represents a different SNR level.

As a first observation, we can now see that SR_∞ is not bounded anymore as it is for the Hamming weight case. While we do not reach a high enough number of attack traces for lower SNR values, all attacks would eventually reach a success rate of one. Second, and most importantly, we can see that each SPA curve is composed of two parts with different slopes. Each implementation quickly

reaches its Hamming weight SR_∞ (≈ 0.3 for unprotected/Boolean, and ≈ 0.8 for arithmetic). Then, the convergence rate decreases as the attack needs to distinguish the classes within the Hamming weights, which illustrates the convergence issue mentioned above. This is due to the information within a given Hamming weight class being inferior to the one between the different Hamming weights. Indeed, the second slope of the curve is equivalent to attacking with a new SNR, where the signal now depends on $\sigma^2_{leakage}$, which is much smaller for a same noise. The DPA does not suffer from this issue thanks to the dependency with a known varying value and the Sbox non-linearity.

Finally, we can see that this close-to-Hamming weight setting largely benefits arithmetic masking, which reaches high success rates more easily than for the unprotected case. An attacker would more easily attack an implementation with arithmetic masking than an unprotected one in this setting. However, for a sufficiently low SNR, this would reverse due to the masking security property.

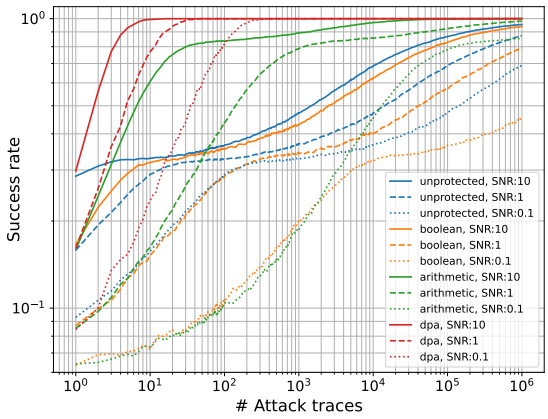

Fig. 4: Success rate (Y-axis) as a function of the number of attack traces (X-axis) for all 3 implementation types, computed with 2000 repetitions for perfectly known linear leakage. The curve color represents the implementation type (unprotected, Boolean, arithmetic, DPA), and the curve type represents the SNR. A plain line is a for a SNR of 10, dots for SNR of 1, and dashes for SNR of 0.1.

4 Imperfect profiling

The previous section studied the ideal case of perfect profiling, where the PDF of the leakages is given to the adversary. We now look at the more realistic case where the attacker aims at approximating the PDF through a profiling phase. She will use a set of N_p profiling traces per class, and estimate a model $\hat{\mathsf{F}}$, with $\hat{\mathsf{F}} \xrightarrow{N_p \to \infty} \mathsf{F}$. In this section, we study the impact of imperfect profiling in the context of SPA, comparing an unprotected secret to a masked one. As for the

previous section, we first look at the particular case of non bijective Hamming weight leakages, before moving to the linear ones.

4.1 Incorrect profiling

We denote by $Pr_{\hat{\mathsf{F}}}[l|v]$ the probability of observing a leakage l for a value v using the estimated model. If the profiling has not converged enough, we might have consistent misclassifications. That is, when the actual value computed is v, we will have $\prod_i Pr_{\hat{\mathsf{F}}}[l_i|v] < \prod_i Pr_{\hat{\mathsf{F}}}[l_i|v']$ as soon as enough attack traces l_i are used, for some other value v'. In this case, v will never be recovered even with an infinite number of attack traces in a SPA context. As a result, SR_∞ would not reach one, even if the leakage function is bijective. We further refer to this as *incorrect profiling*, being stronger case of imperfect profiling. This is illustrated in Figure 5, where the left (resp. right) part of the figure shows an imperfect (resp. incorrect) profiling. The two figures show the actual PDF of some theoretical leakages in black, which can take three values with means μ_i, $i \in [0, 2]$. The colored $\hat{\mu}_i$ correspond to the means calculated through profiling, i.e. the estimated model $\hat{\mathsf{F}}$. In the imperfect profiling on the left, we always have $|\hat{\mu}_i - \mu_i| < |\hat{\mu}_i - \mu_j|$ for $i \neq j$. In this case, all values would eventually be correctly classified given an infinite number of observations. However, for the incorrect profiling on the right, there are two means (shown in red) such that $\exists j \neq i, |\hat{\mu}_i - \mu_j| < |\hat{\mu}_i - \mu_i|$. This is the case for $\hat{\mu}_0$ being close to μ_1 than μ_0. In this case, the corresponding value would eventually be classified, and the SR_∞ bound would not be reached. The rest of this section will show how incorrect profiling impacts SPA for both Hamming weight and linear leakages.

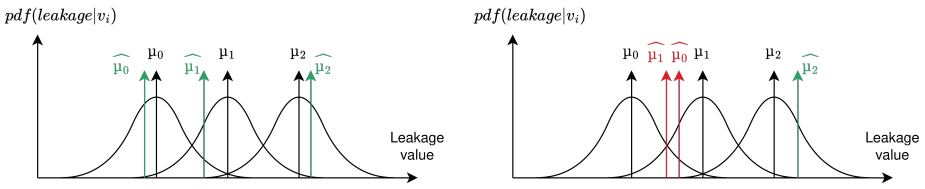

Fig. 5: Illustration of imperfect profiling (left) and incorrect profiling (right).

4.2 Hamming weight leakages

We first study the particular case where $\mathsf{F} = \mathsf{HW}$. From Section 3, we know that when F is known to the attacker, SR_∞ will reach a given SPA bound depending on the masking used. We now look at the impact of incorrect profiling in this context.

Simulations : we use our simulated setting of Section 3.2 on all implementation types, for different SNR values. To visualize the effect of imperfect profiling on SR_∞ , we ran experiments where we varied the number of traces (per identity

class, see Section 3.2) used during the profiling phase. The results are shown in Figure 6, which shows the success rate (Y-axis) as a function of the number of attack traces (X-axis). The red curves represent the unprotected DPA as a benchmark, and the other colors represent an implementation type. The type of the curve (plain, dash, and dots) represents the strength of the profiling, measured as the number of profiling traces used. We used arbitrarily chosen values of N_p for both SNRs to illustrate the effect of an imperfect modeling. The plain curves represent the perfect profiling. The left (resp. right) part of the figure is for a SNR of 1 (resp. 0.1).

As a first observation, we can see that for the SPAs and all implementation types, SR_∞ does depend on the quality of the profiling. However, this is not the case for the DPA, which still reaches complete recovery even with a poor modeling. This exhibits the issue of incorrect profiling for SPA. However, we also notice that for SPA, the convergence speed does not depend on N_p. This is not the case for the DPA, which compensates for the incorrect profiling when using sufficient attack traces.

Second, we can see that in the case of a SNR of one, SR_∞ is higher for arithmetic masking with a poor profiling ($N_p = 5$) than the upper bound for the unprotected and Boolean cases. This means that given an unbounded number of attack traces, recovering the secrets with a SPA in case of arithmetic masking would be much easier than for an unprotected implementation, even if the profiling does not match well the attack traces (e.g. due to model transferability issues [6]). However, we can see that this advantage lessens when moving to a lower SNR of 0.1. Indeed, For $N_p = 20$, the success rate is about the same for arithmetic masking as for the unprotected case. Thus, for a low enough SNR and bad enough profiling, the non-linearity brought by arithmetic masking does not compensate for the information loss, as arithmetic masking is less informative [17].

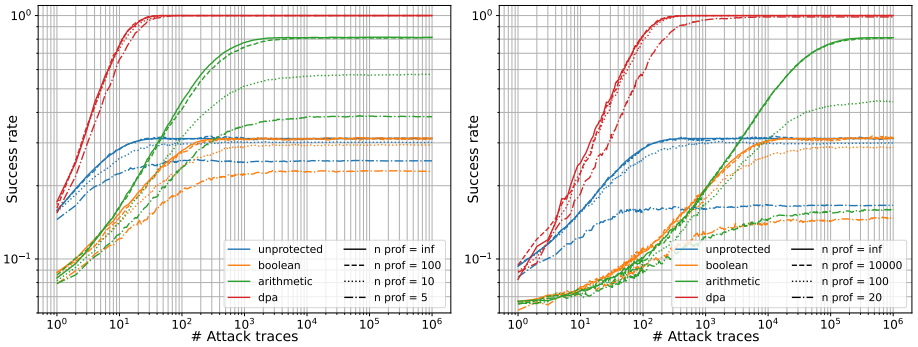

Fig. 6: Success rate (Y-axis) as a function of the number of attack traces (X-axis) for Hamming weight leakages and imperfect profiling. The color of a curve represent an implementation type, and the type of the curve (plain, dash) represents the strength of the profiling. The left (resp. right) part of the figure is for a SNR of 1 (resp. 0.1).

To provide a different view on the effect of the imperfect profiling, we additionally show the convergence SR_∞ as a function of the number of profiling traces, for which the results are shown in Figure 7. It shows the SR_∞ bound (Y-axis) as a function of the number of profiling traces (X-axis). The curve color represents an implementation type (except the red curve for unprotected DPA), and the type of curve represents a SNR level. It uses 2000 repetitions, where SR_∞ is approximated using 1 million attack traces per class.

First, we can clearly see the resilience of DPA over SPA, where we only have $SR_\infty < 1$ for a SNR of 0.1 and less than 20 profiling traces per class. Next, for both unprotected and Boolean masking, we can see that we are bounded by the theoretical SR_∞ of about 0.3, while it goes up to about 0.8 for the arithmetic case. For a SNR of 1, the use of a really bad model (low N_p) still allows a higher SR_∞ in the presence of arithmetic masking than for an unprotected implementation. However, for a lower SNR of 0.1, SR_∞ starts with a higher value for the unprotected case until a better model is used ($N_p \approx 25$). This confirms the previous observation that the non-linearity provided by arithmetic masking benefits over the unprotected case only after a good enough profiling has been performed. We also see that the number of profiling traces to reach the SR_∞ bound is higher for arithmetic masking than for the Boolean case. This means that, in addition to arithmetic masking providing less informative leakages, it is also a more complex model to estimate. Yet, its non-linearity makes it to quickly reach a higher SR_∞ value.

Finally, we observe that for a fixed number of profiling traces, SR_∞ is higher for the unprotected implementation than for the Boolean masking case before reaching the bound. That is, given the same profiling strength, an attack against an unprotected implementation will perform better than against Boolean masking, given an unbounded number of attack traces. This is not a trivial observation, as we will see when moving to the linear leakages.

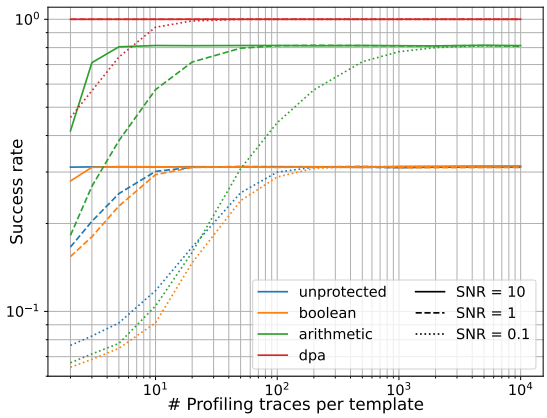

Fig. 7: SR_∞ (Y-axis) as a function of the number of profiling traces (X-axis) for Hamming weight leakages. Curve color represents an implementation type, and the type of curve represents a SNR level.

4.3 General linear leakages

After studying the effect of imperfect profiling in the case of Hamming weight leakages, we study the more realistic case where we assume a general linear leakage form. We investigate the effect on SR_∞ depending on the implementation.

Simulations : we apply our simulated setting with $\sigma^2_{leakage} = 0.0001$ to study the effect of imperfect profiling for linear leakages. Figure 8 shows the SR_∞ (Y-axis) as a function of the number of profiling traces (X-axis). The curve color represents an implementation type (except the red curve for unprotected DPA), and the type of curve (plain, dash, and dots) represents a SNR level. It uses 2000 repetitions, where SR_∞ is estimated using 1 million attack traces per class.

As for the Hamming weight case, we can first see that bad modeling affects the asymptotic success rate. However, for SPAs, we can see that each curve is decomposed into two parts. The first part is due to the profiling being quickly good enough for SR_∞ to reach the Hamming weight bounds. After this threshold, many more profiling traces are needed to get rid of the incorrect profiling within the Hamming weight classes. The DPA is however not affected by this, thanks to the varying known input and the Sbox non-linearity. Note that it is somewhat similar to what was observed with the SR convergence towards one in the perfect setting of Section 3.2. Yet, the difference here lies within the model accuracy with the Hamming weight classes, and not the attack part (which is here unbounded).

Second, this figure confirms that linear leakages benefit more arithmetic masking, as its SR_∞ gets above the one of an unprotected implementation with about $N_p = 15$, where we needed $N_p = 25$ for Hamming weight leakages in Figure 7. Finally, and more surprisingly, we observe interesting results concerning the asymptotic success rate of the unprotected and Boolean cases. First, and for all SNR levels, SR_∞ starts with a higher value for the unprotected case up to the Hamming weight bound of 0.31. This confirms what was shown in the Hamming weight case of Figure 7, where SR_∞ was higher for a given number of profiling traces. However, as soon as the model correctly classifies the Hamming weights ($SR_\infty > 0.31$), the asymptotic success rate becomes higher for Boolean masking than for the unprotected case. This remains until the profiling becomes better, where the SR_∞ against the unprotected value gets ahead again.

As a side note, we notice that SR_∞ seems to decrease for the Boolean case (SNR=0.1) when N_p is very high. We emphasize that this is not an actual trend, and is only due to the number of attack traces not being enough to get a proper approximation of SR_∞ in this case. Indeed, while the profiling gets more accurate, it takes many more attack traces to reach the SR_∞ bound due to the masking security property.

5 Practical experiment

To validate the theoretical and simulated results of Sections 3 and 4, we conducted actual experiments. We used a Chipwhisperer CW1200 with the CW308 STM32F target board [19]. The targeted microcontroller is a STM32F415RGT6

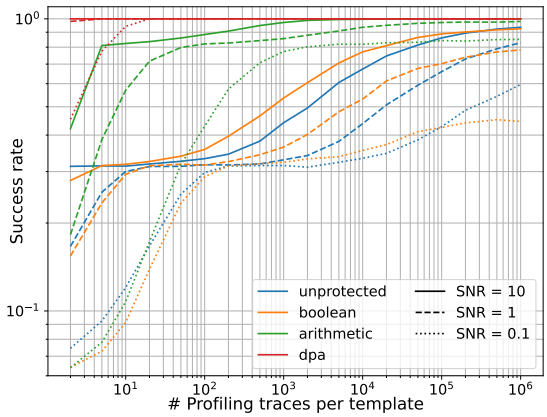

Fig. 8: SR_∞ (Y-axis) as a function of the number of profiling traces (X-axis) for linear leakages with $\sigma^2_{leakage} = 0.0001$. Curve color represents an implementation type, and the type of curve represents a SNR level.

and is based on a 32-bits ARM cortex M4 architecture. The clock is generated by the chipwhisperer at 7.3MHz and fed to the target. We sampled the power traces at 29.7MHz (4 times per clock cycle).

As in our simulated settings, we used 4-bit values and obtained the same type of leakages as described in Section 3.2. For the DPA, we compute the Present Sbox. For the unprotected SPA, we simply directly load the sensitive variable in memory. For masked implementations, to validate the simulations, we study the security of the encoding by successively loading the two shares, with multiple NOP instructions in between. We used a gpio signal to trigger acquisitions. The SNR level is around 6, which is typical using the chipwhisperer.

5.1 Leakage characterization

Before performing the different attacks, we performed a leakage characterization to determine how close to the Hamming weight model the leakages are. We performed a linear regression [24] with the 4 bits as a basis to compute the coefficients a_i as defined in Section 2.2. This gave us leakages with $\sigma^2_{leakage} = 0.00029$, which is close to our simulations. For completeness, the value of the coefficients are given in Appendix B.

To verify the bijectivity of the actual measurements, we computed the perceived information [21] (PI) for both linear regression with a Hamming weight and linear bases. The results are shown in Figure 9, where the Y-axis shows the PI for a number of profiling traces given by the X-axis. The orange (resp. blue) curve corresponds to the Hamming weight (resp. linear) basis. The right part of the figure is a zoom where the two curves meet. As we can see, the orange PI converges much more quickly as it is a more simple basis. The two curves reach a very similar value, indicating that the leakages are almost as informative as the Hamming weight function. Yet, we reach a slightly higher PI for the linear

basis, showing the bit coefficients are slightly different, which puts us in a similar setting as for the linear leakage ones used in simulations.

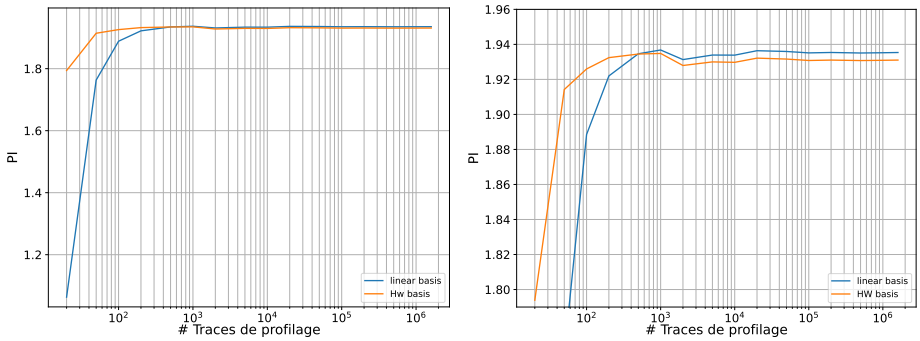

Fig. 9: Perceived information (Y-axis) for a number of traces given by the X-axis. The orange (resp. blue) curve is for the Hamming weight (resp. linear) basis. The right part of the figure is a zoom of the left one.

5.2 Attack results

After confirming the bijectivity of the leakages, and their proximity to the Hamming weight function, we applied a template attack on the different implementation types. The results are shown in Figure 10. The Y-axis corresponds to the SR_∞ obtained for a number of profiling traces given by the X-axis. The curve color represents an implementation type. We used a dataset of 200,000 traces per class, with 10-fold cross validation.

As we can see, we get very similar results as for the simulated setting of Section 4.3. First, we can see that arithmetic masking allows to easily reach a higher success rate than the unprotected case. We also note that due to the SNR level being too high, we almost instantly reach a success rate higher than the Hamming weight bounds (0.3 for unprotected/Boolean, and 0.8 for arithmetic). The SR_∞ obtained is thus mainly defined by the information carried to distinguish within a given Hamming weight, from $\sigma^2_{leakage}$. Note that the DPA directly reaches SR_∞ =1, which is not surprising given the high SNR.

Finally, we do observe the interesting relation between the unprotected and Boolean cases. Indeed, while the profiling strength is bad enough so that the Hamming weights are not distinguished ($SR_\infty \lesssim 0.31$), the unprotected case provides a higher SR_∞ . After this threshold, this reverses, which holds until the profiling gets accurate enough. Overall, the practical experiments confirm the interesting properties of the SPA context, where the use of masking may give a higher success rate given enough attack traces.

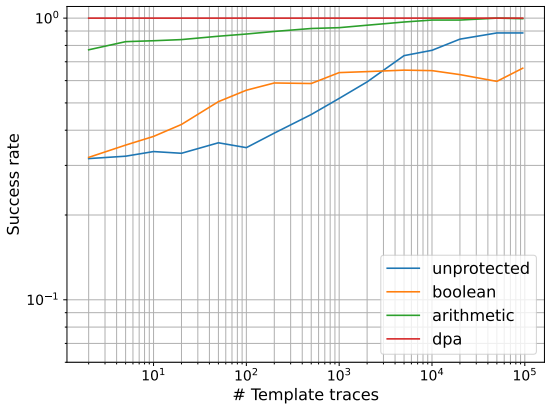

Fig. 10: SR_∞ (Y-axis) obtained for a number of profiling traces given by the X-axis for the real measurements. The curve color represents an implementation type.

6 Conclusion, discussion

We studied the context of SPA against unprotected and masked implementations, with different assumptions on the leakage functions and the profiling strength of the adversary. To the best of our knowledge, this has not yet been investigated in the literature and shows interesting properties. In the perfect profiling setting, for Hamming weight leakages, we showed that arithmetic masking allows to reach a higher asymptotic success rate than an unprotected implementation, thanks to the non-linearity brought by the carry propagation. We showed that this still holds in the context of more realistic linear leakages, depending on the number of attack traces used. In this context, we showed the success rate has a two-phase convergence, where the second one is slower as it is equivalent to attacking with a new smaller SNR, where the signal now depends on $\sigma^2_{leakage}$.

When moving to the more practical context of imperfect profiling, we showed that the asymptotic success rate of SPA largely depends on the profiling strength, while DPA is much more resilient to incorrect profiling. Yet, we still showed that more information can be recovered when targeting arithmetic masking rather than an unprotected implementation, depending on the profiling strength and the SNR level. Additionally, we surprisingly exhibited a case where the asymptotic success rate is higher in the context of Boolean masking than for an unprotected implementation. Explaining this property formally would be an interesting future research direction.

Interesting future directions could study how other masking encodings behave, along with more shares. Additionally, as we assumed profiling with knowledge of the shares, one could investigate how the asymptotic success rate of a DPA behaves with it is not the case.

References

1. Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehlé, D.: Crystals-kyber algorithm specifications and supporting documentation. NIST PQC Round **2**(4), 1–43 (2019)
2. Barthe, G., Dupressoir, F., Faust, S., Grégoire, B., Standaert, F.X., Strub, P.Y.: Parallel implementations of masking schemes and the bounded moment leakage model. In: Advances in Cryptology–EUROCRYPT 2017: 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30–May 4, 2017, Proceedings, Part I 36. pp. 535–566. Springer (2017)
3. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: An ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (Sep 2007). https://doi.org/10.1007/978-3-540-74735-2_31
4. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Cryptographic Hardware and Embedded Systems-CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings 6. pp. 16–29. Springer (2004)
5. Cagli, E., Dumas, C., Prouff, E.: Convolutional neural networks with data augmentation against jitter-based countermeasures: Profiling attacks without preprocessing. In: Cryptographic Hardware and Embedded Systems–CHES 2017: 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings. pp. 45–68. Springer (2017)
6. Cao, P., Zhang, C., Lu, X., Gu, D.: Cross-device profiled side-channel attack with unsupervised domain adaptation. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 27–56 (2021)
7. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Cryptographic Hardware and Embedded Systems-CHES 2002: 4th International Workshop Redwood Shores, CA, USA, August 13–15, 2002 Revised Papers 4. pp. 13–28. Springer (2003)
8. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Ascon v1. CAESAR Competition (2014)
9. Duc, A., Dziembowski, S., Faust, S.: Unifying leakage models: from probing attacks to noisy leakage. In: Advances in Cryptology–EUROCRYPT 2014: 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings 33. pp. 423–440. Springer (2014)
10. Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic analysis: Concrete results. In: Cryptographic Hardware and Embedded Systems—CHES 2001: Third International Workshop Paris, France, May 14–16, 2001 Proceedings 3. pp. 251–261. Springer (2001)
11. Gierlichs, B., Batina, L., Tuyls, P., Preneel, B.: Mutual information analysis: A generic side-channel distinguisher. In: International Workshop on Cryptographic Hardware and Embedded Systems. pp. 426–442. Springer (2008)
12. Ishai, Y., Sahai, A., Wagner, D.: Private circuits: Securing hardware against probing attacks. In: Advances in Cryptology-CRYPTO 2003: 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003. Proceedings 23. pp. 463–481. Springer (2003)
13. Kocher, P.C.: Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In: Advances in Cryptology—CRYPTO'96: 16th Annual International Cryptology Conference Santa Barbara, California, USA August 18–22, 1996 Proceedings 16. pp. 104–113. Springer (1996)

14. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M.J. (ed.) Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings. Lecture Notes in Computer Science, vol. 1666, pp. 388–397. Springer (1999). https://doi.org/10.1007/3-540-48405-1_25, https://doi.org/10.1007/3-540-48405-1_25
15. Lomné, V., Prouff, E., Rivain, M., Roche, T., Thillard, A.: How to estimate the success rate of higher-order side-channel attacks. In: Cryptographic Hardware and Embedded Systems–CHES 2014: 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings 16. pp. 35–54. Springer (2014)
16. Mangard, S.: Hardware countermeasures against DPA ? A statistical analysis of their effectiveness. In: Okamoto, T. (ed.) Topics in Cryptology - CT-RSA 2004, The Cryptographers' Track at the RSA Conference 2004, San Francisco, CA, USA, February 23-27, 2004, Proceedings. Lecture Notes in Computer Science, vol. 2964, pp. 222–235. Springer (2004). https://doi.org/10.1007/978-3-540-24660-2_18, https://doi.org/10.1007/978-3-540-24660-2_18
17. Masure, L., Méaux, P., Moos, T., Standaert, F.X.: Effective and efficient masking with low noise using small-mersenne-prime ciphers. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 596–627. Springer (2023)
18. Migliore, V., Gérard, B., Tibouchi, M., Fouque, P.A.: Masking dilithium: Efficient implementation and side-channel evaluation. In: Applied Cryptography and Network Security: 17th International Conference, ACNS 2019, Bogota, Colombia, June 5–7, 2019, Proceedings 17. pp. 344–362. Springer (2019)
19. O'Flynn, C., Chen, Z.D.: ChipWhisperer: An open-source platform for hardware embedded security research. In: Prouff, E. (ed.) COSADE 2014. LNCS, vol. 8622, pp. 243–260. Springer, Heidelberg (Apr 2014). https://doi.org/10.1007/978-3-319-10175-0_17
20. Prouff, E.: Dpa attacks and s-boxes. In: International Workshop on Fast Software Encryption. pp. 424–441. Springer (2005)
21. Renauld, M., Standaert, F.X., Veyrat-Charvillon, N., Kamel, D., Flandre, D.: A formal study of power variability issues and side-channel attacks for nanoscale devices. In: Advances in Cryptology–EUROCRYPT 2011: 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings 30. pp. 109–128. Springer (2011)
22. Rivain, M.: On the exact success rate of side channel analysis in the gaussian model. In: International Workshop on Selected Areas in Cryptography. pp. 165–183. Springer (2008)
23. Rivain, M., Prouff, E.: Provably secure higher-order masking of aes. In: International Workshop on Cryptographic Hardware and Embedded Systems. pp. 413–427. Springer (2010)
24. Schindler, W., Lemke, K., Paar, C.: A stochastic model for differential side channel cryptanalysis. In: Cryptographic Hardware and Embedded Systems–CHES 2005: 7th International Workshop, Edinburgh, UK, August 29–September 1, 2005. Proceedings 7. pp. 30–46. Springer (2005)
25. Standaert, F.X., Malkin, T.G., Yung, M.: A unified framework for the analysis of side-channel key recovery attacks. In: Advances in Cryptology-EUROCRYPT 2009: 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings 28. pp. 443–461. Springer (2009)

26. Standaert, F.X., Veyrat-Charvillon, N., Oswald, E., Gierlichs, B., Medwed, M., Kasper, M., Mangard, S.: The world is not enough: Another look on second-order dpa. In: Advances in Cryptology-ASIACRYPT 2010: 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings 16. pp. 112–129. Springer (2010)
27. Strieder, E., Ilg, M., Heyszl, J., Unterstein, F., Streit, S.: Asca vs. sasca: A closer look at the aes key schedule. In: International Workshop on Constructive Side-Channel Analysis and Secure Design. pp. 65–85. Springer (2023)

A Proof of proposition 1

Proposition. *Two unprotected values v_0 and v_1 have the same Hamming weight if, and only if, the two joint distributions of the Hamming weight of their shares are the same.*

Proof. **Necessity:** $\mathsf{HW}(v_0) = \mathsf{HW}(v_1)$ implies that there exists a permutation of the bits, denoted P, such that $\mathsf{P}(v_0) = v_1$. The existence of P implies a bijective relationship between the sharings of v_0 and v_1. Indeed, let's consider a sharing (a, b) such that $v_0 = a \oplus b$. Then $v_1 = \mathsf{P}(a) \oplus \mathsf{P}(b)$ by linearity of P, $(\mathsf{P}(a), \mathsf{P}(b))$ is a sharing for v_1 such that $(\mathsf{HW}(a), \mathsf{HW}(b)) = (\mathsf{HW}(P(a)), \mathsf{HW}(P(b)))$. Since the roles of v_0 and v_1 can be inverted, it proves the bijective relationship.

Sufficiency by contraposition: if $\mathsf{HW}(v_0) \neq \mathsf{HW}(v_1)$, then there exists a sharing $(v_0, 0)$ of v_0. For this sharing, there cannot be a sharing (a, b) of v_1 such that $\mathsf{HW}(a) = \mathsf{HW}(v_0)$ and $b = 0$. As a result, the joint distribution of the Hamming weights of the shares differs between v_0 and v_1.

B Leakage characterization and linear regression

We used the experimental setup described in section 5 to perform a linear regression [24] with the 4 bits as a basis, and computed the coefficients a_i as defined in Section 2.2. The leakages are normalized so that the values are comparable to the simulated results. The results are shown in Figure 11. The Y-axis corresponds to the leakage value for a coefficient a_i given in the X-axis. We show the median values as dots, along with the intervals containing 95% of the results (computed with 200 experiments using 50k traces each). As we can see, while the coefficients are very close, they seem to have small differences, indicating a bijective leakage function close to HW. We computed the corresponding $\sigma^2_{leakage}$, which results in a value of 0.00029, which is close to our simulations.

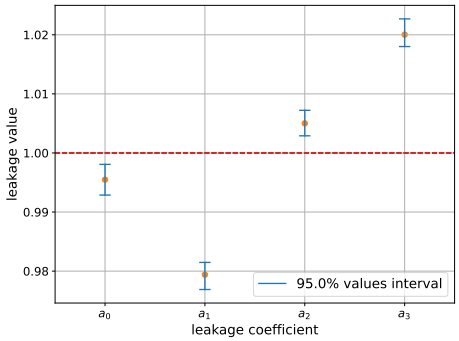

Fig. 11: Linear regression on the real measurements. The Y-axis corresponds to the values of the a_i coefficients given by the X-axis. Dots are the median values, with their confidence intervals in blue.