Don't Learn What You Already Know

Grey-Box Modeling for Profiling Side-Channel Analysis against Masking

Loïc Masure[1], Valence Cristiani[2], Maxime Lecomte[2] and François-Xavier Standaert[1]

[1] Crypto Group, ICTEAM, UCLouvain, Louvain-la-Neuve, Belgium
firstname.lastname@uclouvain.be
[2] Univ. Grenoble Alpes, CEA, LETI, France firstname.lastname@cea.fr

Abstract. Over the past few years, deep-learning-based attacks have emerged as a *de facto* standard, thanks to their ability to break implementations of cryptographic primitives without pre-processing, even against widely used counter-measures such as hiding and masking. However, the recent works of Bronchain and Standaert at TCHES 2020 questioned the soundness of such tools if used in a black-box setting to evaluate implementations protected with higher-order masking. On the opposite, white-box evaluations may be seen as possibly far from what a real-world adversary could do, thereby leading to too conservative security bounds. In this paper, we propose a new threat model that we name *grey-box* benefiting from a trade-off between black and white box models. Our grey-box model is closer to a real-world adversary, in the sense that it does not need to have access to the random nonces used by masking during the profiling phase like in a white-box model, while it does not need to learn the masking scheme as implicitly done in a black-box model. We show how to combine the power of deep learning with the prior knowledge of grey-box modeling. As a result, we show on simulations and experiments on public datasets how it allows to reduce by an order of magnitude the *profiling* complexity, *i.e.*, the number of profiling traces needed to satisfyingly train a model, compared to a fully black-box model.

Keywords: Profiling Attacks · Side-Channel · Deep Learning · Gradient Descent · Masking · Grey-Box

1 Introduction

Context. The past few years have seen the emergence of new promising lines of research in profiling Side-Chanel Analysis (SCA), which coincided with the recent advances in Machine Learning (ML) during the 2010's. Indeed, profiling attacks may be formalized as a *supervised learning* problem. As an example, the Gaussian Templates (GTs) initially proposed by Chari *et al.* in their seminal work [CRR03] are actually equivalent to a Quadratic Discriminant Analysis (QDA) in the ML terminology [HTF09]. Hence a vast investigation of relevant learning algorithms in the ML zoology, beyond those generative models [HGD$^+$11, BL12, HZ12, LBM14, LBM15, PHG17]. In particular, following the remarkable performance of Deep Neural Networks (DNNs) in solving tasks in computer vision, the SCA community has progressively drawn its interest on such models [GHO15, MZ13, MDM16]. Nowadays, DNNs are known to be able to defeat most of the counter-measures used to protect implementations against SCA, namely de-synchronization [CDP17, KPH$^+$19], shuffling [MDP19a, MS21] and more interestingly masking [MPP16, Tim19].

The Black-vs.-White-Box Dichotomy. Although the supervised attack threat model introduced so far is nowadays widely adopted by the SCA community for security evaluations, one technical detail of this scenario lacks some consensus. Indeed, there exists a debate among SCA practitioners about what is known or unknown by the adversary during the profiling phase, during which one builds the attack model upon traces measured on an open clone device. In particular, whether one has access to the random nonces used by the clone device during the encryption, as part of the profiling data. This question is not trivial, since nowadays most of the counter-measures — like masking or shuffling — consist in turning a deterministic cryptographic primitive into a non-deterministic implementation.On the one hand, academia usually assumes the adversary to know the values of the random nonces, in a so-called *white-box* threat model. This model trades off some potentially conservative security levels against an easy-to-analyze evaluation approach thanks to theoretical shortcuts [DDF14, DFS15]. On the other hand, practitioners such as industrial developers and evaluators rather assume the adversary to not have access to the random nonces used by the clone device during the encryption, hence the name of *black-box* threat model. This scenario has the two advantages of being closer to a real-world adversary, and to be fully automatized. As a drawback, some current attacks in the black-box settings can be much less efficient than white-box attacks [BS20].

To what extent one threat model or another better fits the security context of the evaluation? The answer is not unique, and both threat models have their proponents and opponents. As an example, black-box scenarios may be considered for good or bad reasons. On the one hand, it spares lots of human efforts and expertise spent in pre-processing the traces, which are taken into account in the assessment of an attack potential [SOGISS20]. On the other hand, developers may be tempted to artificially restrict the access to random nonces on the clone device given for evaluation in order to maximize the chances to pass certifications, although at the cost of a false sense of security.

Actually, both black-box and white-box models may be seen as the edges of a broad scope of threat models ranging from weak adversaries in the black-box model to stronger ones in the white-box model. Yet, realistic threat models often lie all along this spectrum. As an example, the implementation of a Target of Evaluation (T.O.E.) may be open-source — or at least based on publicly available source code — and hence fully available to any malicious entity.[1] Likewise, the number of different masking schemes in the literature is restricted enough so that it may be assumed to be known by the adversary. Surprisingly, to the best of our knowledge, no ML approach leveraging weaker adversaries than white-box but still stronger than black-box have been considered so far. Indeed, when profiling masked implementations most of the Deep Learning (DL)-SCA literature focused on the choice of DNN architectures and hyper-parameters [KPH$^+$19, ZBHV19]. Hence the motivations of this work:

> *How can we efficiently leverage the knowledge of the masking scheme in an ML-based profiling attack, without relying on the knowledge of random nonces ?*

The Grey-Box Modeling. To address this question, we investigate a new type of SCA adversary that we name *grey-box*. In this threat scenario, the adversary is supposed to have access to the source code of the target implementation. Concretely, this means that the adversary knows the masking scheme and order used to protect the target. Moreover, she is able to localize some Points of Interest (P.o.Is) precisely enough thanks to a careful code analysis.

We explain how these assumptions can be taken into account in a DL model. To this end, we introduce GroupRecombine, a simple neural network layer encoding the knowledge of any group-based masking scheme, under the form of a discrete convolution. Contrary to the convolutions layers used in Convolutional Neural Networks (CNNs), GroupRecombine

[1] See, *e.g.*, the mbedTLS implementations: www.mbedTLS.com, or the Agence Nationale de la Sécurité des Systèmes d'Information (ANSSI)'s protected Advanced Encryption Standard (AES): https://github.com/ANSSI-FR/SecAESSTM32.

is parameter-free, and is applied as the last layer in our model. This new layer can replace some of the upper layers of a DNN potentially carrying many learning parameters to fit, without any loss of expressiveness of the resulting architecture, thanks to the prior knowledge of the masking scheme. In addition, it can be efficiently implemented using Walsh-Hadamard (in the case of Boolean masking) or Fourier transforms (in the case of arithmetic or multiplicative masking), or a mix of both (in the case of affine masking). As a result, any model equipped with the GroupRecombine no longer requires to learn how to recombine the information gathered on each share, and may only focus on the joint learning of the leakage models of each share.

We validate our approach on simulations and on public datasets. Our experiments on the ANSSI's SCA Databases (ASCAD) emphasize some use-cases with first-order Boolean masking where very simple grey-box models lead to successful attacks, whereas their black-box counter-part fails. This suggests that a significant part of the efforts spent by the SCA practitioner in a black-box setting, *e.g.* by running huge hyper-parameter grid searches, would actually be devoted to finding a DNN architecture that efficiently captures the way to learn the masking scheme. Hence, using GroupRecombine may be seen as an efficient surrogate to this issue. As an example, we also address the challenge left by Bronchain and Standaert as a conclusion of their works at TCHES 2020, by showing on simulations how GroupRecombine could be used for profiling in presence of an affine masking scheme, without knowing the random shares to train our model.

Finally, we conclude this paper by discussing how far the grey-box approach could scale with an increasing masking order, by providing theoretical arguments and experimental evidences. Actually, the potential limitations of GroupRecombine that we emphasize are not restricted to our approach, and more generally cover at least any non-white-box model trained with gradient descent, leaving open the question whether this also covers other types of non-white-box profiling models. Overall, we hope that these questions will be received as a helpful contribution to the more general debate regarding the choice of different evaluation methodologies in SCA.

2 Grey-Box Modeling and Application to Masking

In this section, we introduce the grey-box adversary. The idea behind this new threat model is to properly separate what can be assumed to be known by the adversary — *e.g.*, any algorithmic and implementation aspect — from what remains unknown and therefore should be learned during the profiling phase — *e.g.*, the device-dependent leakage model of each share. We discuss hereafter two aspects of the prior knowledge on which any grey-box adversary may rely, and how to leverage it.

2.1 Hard-Encoding of the Discrete Convolution

Hereafter, we explain how to materialize the prior knowledge of the masking scheme into our DNN. Recall that the true model to learn may be expressed as a convolution product of elementary leakage models for each share, as stated hereafter.

Proposition 1 ([LPR+14, Sec. 6], extended)**.** *Let $Y_0, \ldots, Y_d \in \mathcal{Y}$ be Independent and Identically Distributed (i.i.d.) shares, uniformly drawn over the group $(\mathcal{Y}, \star)$. Let $\mathbf{L} = (\mathbf{L}_0, \ldots, \mathbf{L}_d)^\intercal$ be a random vector denoting the leakage, and let $\boldsymbol{l} = (\boldsymbol{l}_0, \ldots, \boldsymbol{l}_d)^\intercal$ be an observation of $\mathbf{L}$. Assume that any $\mathbf{L}_i$ only depend on Y_i, i.e., any $\mathbf{L}_i$ is independent of the $(\mathbf{L}_j)_{j \neq i}$. Then, the posterior Probability Mass Function (p.m.f.) of $Y = Y_0 \star \ldots \star Y_d$ can be formulated as a discrete convolution product:*

$$\mathsf{p}_Y(\boldsymbol{l}) = \mathsf{p}_{Y_0}(\boldsymbol{l}_0) * \ldots * \mathsf{p}_{Y_d}(\boldsymbol{l}_d) \quad , \tag{1}$$

where $\mathsf{p}_{\mathrm{Y}_i}(\boldsymbol{l}_i) = \Pr(\mathrm{Y}_i \mid \mathbf{L}_i = \boldsymbol{l}_i)$ *denotes the conditional p.m.f. of the share* Y_i *given the realization* $\boldsymbol{l}_i$ *of the leakage random vector* $\mathbf{L}_i$*.*

Lomné *et al.* have given a proof of a similar result at CHES 2014, for generative models such as GTs. Proposition 1, that we prove in Appendix A, extends Lomné *et al.*'s one to discriminative models. We may leverage Proposition 1 in a grey-box threat model, provided that we know the inner law $\star$ of the group $\mathcal{Y}$. This means that we know the discrete convolution operator in Equation 1. In other words, we no longer require to learn how to recombine the information extracted on the leakage corresponding to each share: we are reduced to jointly learn the leakage models $\boldsymbol{l}_i \mapsto \mathsf{p}_{\mathrm{Y}_i}(\boldsymbol{l}_i)$ using some corresponding estimators $\mathsf{m}_{\theta_i}, i \in [\![0, d]\!]$ respectively. Hence proposing the following model for our grey-box attacks:

$$\mathsf{m}_\theta(\boldsymbol{l}) = \mathsf{m}_{\theta_0}(\boldsymbol{l}_0) * \ldots * \mathsf{m}_{\theta_d}(\boldsymbol{l}_d) \quad . \tag{2}$$

Said more concretely, we build a model where each *branch* m_{θ_i} maps its corresponding sub-leakage to a $|\mathcal{Y}|$-dimensional vector denoting a p.m.f. Then, all branches are combined together by computing the discrete convolution. Figure 1 depicts the idea for a first-order masking scheme: blue nodes denote branches, *i.e.* models with trainable parameters, whose goal is to modelize the conditional p.m.f. $\mathsf{p}_{\mathrm{Y}_i}(\boldsymbol{l}_i)$ for each share. Vectors with shades of red denote p.m.f.s over $\mathcal{Y}$, and the "$*$" node denotes the discrete convolution with respect to the inner-law of the group $\mathcal{Y}$.

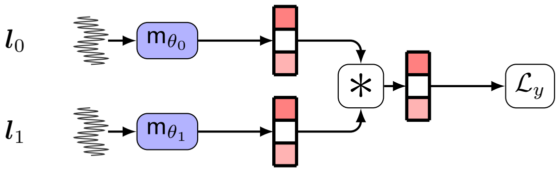
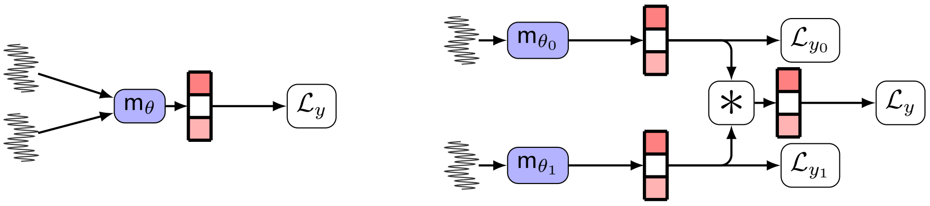


Figure 1: Grey-box model, with known P.o.Is for each share and a known masking scheme, but unknown random shares.

It remains to explain how the branch models can be fine-tuned so that they fit the true leakage models on each share. This can be done with Maximum Likelihood Estimation (MLE), *i.e.* by minimizing a *loss* function $\mathcal{L}_y$ quantifying the dissimilarity between the overall output p.m.f. returned by the model $\mathsf{m}_\theta(\boldsymbol{l})$ and the expected values of the target variable, that are known during profiling (for both black, grey and white box adversaries).

In a white-box model, the MLE is usually implemented by tuning of branch model separately from each other by minimizing the loss functions $\mathcal{L}_{y_i}$ of each branch separately. Once it is done, the fine-tuned branch models are combined together with the discrete convolution. This is the approach used, *e.g.*, by Bronchain *et al.* [BS20, BDMS22]. Similarly,



(a) Black-box model.

(b) White-box.

Figure 2: Other adversaries, for comparison with grey-box model.

each branch model can compute its output p.m.f. using Gaussian Template (GT), and by converting the generative model into a discriminative one with Bayes' rule, as done somehow by Ouladj *et al.* [OGGM21]. Unfortunately, these approaches require to know the values of the shares during profiling, which we considered to be a strong assumption in real-world evaluations.

In a grey-box model instead, the branch models are rather *jointly* tuned by directly minimizing the overall loss function $\mathcal{L}_y$, averaged over a training set of traces acquired during the profiling phase. As depicted in Figure 1, computing and minimizing the overall loss function $\mathcal{L}_y$ only requires as labels the values of the target variable during profiling, which is allowed by definition of our grey-box adversary.

Therefore, the grey-box model depicted in Figure 1 relaxes the strong assumption of random nonce knowledge during the profiling phase, but still encodes the knowledge of the masking scheme, which is no longer needed to implicitly learn from the data, contrary to the black-box model depicted in Figure 2a. This approach sounds somewhat sub-optimal if the scheme is already known. Hence, by introducing the grey-box adversary that is stronger than the black-box adversary, but weaker than the white-box one, we expect to get a closer emulation of the actual powers of a real-world adversary.

2.2 Localization of P.o.Is

In Subsection 2.1, we implicitly assumed for non-black-box adversaries to know how to localize the P.o.Is for each share in the traces, in order to properly separate the leakages. We discuss this assumption in this section.

Usually, the P.o.I selection is done by computing some first-order statistics, such as T-tests or Signal-to-Noise Ratios (SNRs). Without knowledge of the random nonces, these tools cannot identify the right time samples, since by definition of masking, any univariate sample should be independent of the target variable Y.[2] This means that any non-white-box adversary cannot identify the P.o.Is thanks to statistical tools in presence of masking.

Nevertheless, a P.o.I selection remains possible without access to the random nonces, thanks to a visual analysis of the traces, combined with a careful study of the T.O.E. source code. Indeed, a software implementation of a cryptographic primitive is typically made of (nested) loops whose number of iterations are publicly known, in virtue of the Kerckhoffs principle. This induces some sequences of (nested) patterns in the traces that can be visually identified on the raw measurements by the adversary. Moreover, this analysis can even be refined by counting the number of clock cycles for each executed instructions, and combining them with the clock and sampling frequencies in order to guess at which time sample each instruction should leak. As a consequence, it is still possible to localize the leakage on each share, and the P.o.I selection through T-test or SNR should actually be seen as a useful but non-necessary shortcut for the evaluator to spare some time. The recent literature provides two examples of this approach. First, Masure and Strullu reported a detailed code analysis of the assembly code of the ANSSI's secure software implementation of the AES on an ARM Cortex M4, in order to extract $15,000$ P.o.Is out of 1 million time samples in the raw traces, covering the leakages of the three shares used by the affine masking scheme [MS21]. Second, Egger *et al.* verified that the CPOI leakage detection method [DS16] could localize the same time windows as with the analysis of the assembly code used in the ANSSI's SCA Databases (ASCAD)-v1 dataset [EST+22, Fig. 4].

[2] Ignoring glitches or transition leakages.

3 Grey-Box Modeling with DNNs

Now we have introduced the grey-box adversary in the case of masking and explained the intuition behind its advantages, we discuss in this section how to concretely implement it with Deep Neural Networks (DNNs). First, we discuss in Subsection 3.1 how to concretely minimize the overall loss function with grey-box models, by introducing a new DNN layer called GroupRecombine. Then, we explain in Subsection 3.2 how our approach can be extended to many types of masking schemes. Finally, we argue in Subsection 3.3 why we implemented our own version of GroupRecombine, instead of relying on native building blocks of most DL frameworks.

3.1 Implementing the Backward Propagation

To optimize a function based on DNNs, the most widely used approach is to use Gradient Descent (GD)-based optimizers. To this end, we need to compute the derivatives of our model when using a recombination layer. These derivatives are computed with the backward propagation algorithm [BPRS17], who leverages the chaining rule to reduce the computation of the derivatives for a composed function to the computation of derivatives for each elementary function. Our models being made of regular DNN layers for which the backward propagation is already hard-coded, we are then reduced to specify how to back-propagate the gradient through the discrete convolution only. We do this by implementing GroupRecombine, a parameter-free DNN layer consisting in the discrete convolution, augmented with backward propagation.

We briefly explain hereafter how the backward propagation can be computed in GroupRecombine. Thanks to the convolution theorem, the discrete convolution layer can itself be seen as a composition of a fast transform (and its inverse) and an element-wise product of $d+1$ vectors. Using the chaining rule, we are again reduced to compute the backward propagation for each mapping in the composition. The (inverse) fast transform is a linear mapping, so its differential coincides with the mapping itself. In other words, the backward pass of the fast transform is the same as its forward pass. Regarding the element-wise product, it is a multi-linear mapping whose backward pass is already hard-coded in the DL frameworks such as Tensorflow or Pytorch. By putting things together, we obtain the backward pass through our GroupRecombine.

Remark 1. The backward pass of the GroupRecombine layer can also be directly hard-coded without decomposing with fast transforms. Interestingly, this approach coincides with implementing the update rule "from functions to variables" in the Belief - Propagation (BP) algorithm [BS21, Eq. (4)].

3.2 Handling other Types of Masking

GroupRecombine works for any type of group-based masking scheme, *e.g.* Boolean [CJRR99, GP99], arithmetical [CG00], or multiplicative masking [von01]. In the latter case, one should recall that for any finite field $(\mathcal{Y}, \oplus, \times)$, the group $(\mathcal{Y}, \times)$ is in bijection with $(\mathbb{Z}_{|\mathcal{Y}|-1}, +)$. In other words, the GroupRecombine for multiplicative masking can be implemented by using the GroupRecombine for arithmetical masking, and to permuting the entries of input and output vectors using discrete log / alog tables. As a result, it becomes also straightforward to handle less common types of masking, such as affine [FMPR11], by combining several types of GroupRecombine for different masking schemes. We will apply GroupRecombine to affine masking in Subsection 4.2.

Although we did not test it yet, extending GroupRecombine to inner-product masking schemes [BFG$^+$17, BFG15] should be feasible as well. Indeed, inner-product masking derives from Boolean masking by applying a public linear mapping, that could be handled by hard-coding the corresponding permutation of the entries in the input and output p.m.f.s, similarly to the transformation from arithmetical to multiplicative masking.

3.3 Using Native Convolutions in DL Frameworks?

We implemented our own version of the discrete convolution used in GroupRecombine. Since discrete convolutions are widely used in DL, one might wonder why not using such layers natively implemented in the main frameworks such as Tensorflow or Pytorch. There are two main reasons for that.

First, for Boolean masking, the discrete convolution is not natively implemented in DL frameworks like Pytorch [PGM+19], or Tensorflow [AAB+15]. Even for arithmetical masking the discrete convolution must be circular, whereas the convolution layers used in DL frameworks are usually not circular and use zero-padding to deal with side effects.

Second, even if circular padding were used, the convolution layers proposed in the DL frameworks rely on a naive computation of the convolution product — *i.e.* not based on fast transforms. The reason is that in computer vision-based DL, the filter size W is often too small for the computation with fast transform — with complexity $\mathcal{O}\left(W \cdot \log_2\left(W\right)\right)$ — to be significantly more efficient that the naive approach of complexity $\mathcal{O}\left(W^2\right)$ [VJM+15].[3] In our context where the convolutions are often computed over $W = 2^n$ classes, where n is the bit-size of the target, using a non-naive approach becomes more efficient than the naive one.

That is why we implement GroupRecombine using fast transforms. For Boolean masking, we use the Walsh-Hadamard (WH) transform, whereas for arithmetical masking, we use the Fast Fourier Transform (FFT). Both WH and FFT can be implemented with Pytorch on (General Purpose) Graphic Processing Unit (GPU) with a CUDA backend: the latter one is natively implemented in the framework, while for the former one, we leverage the implementation developed by Thomas *et al.* [TGD+18]. Overall, our GroupRecombine layer results in a parameter-free layer that can be easily integrated into the Pytorch framework.[4]

4 Analyzing Performances of GroupRecombine

Now we introduced our GroupRecombine, we would like to compare its performances with black-box and white-box models. In this section, we show the advantages of using GroupRecombine, both on simulated experiments and on real experimental data. We first describe the settings of our experiments in Subsection 4.1. Then, we report and discuss results on simulations in Subsection 4.2, and on experiments in Subsection 4.3.

4.1 Settings for Comparison

For a fair comparison, we would like to show that all other things being equal, using GroupRecombine leads to better performance. This requires to properly define the types of adversaries against which we test GroupRecombine, and how to assess the comparison between each model.

[3] Computer vision applications of DL usually prefer filter sizes lower than 4, as it maximizes the ratio between the size of receptive fields of the convolution layers and the number of parameters to train [SZ15]. This rational only holds for 2D (images) or 3D (videos) data, and no longer holds for 1D time series like SCA traces.

[4] We illustrate this claim with some code snippet in the supplementary material of this submission. Although we did not test, we expect the implementation to be as straightforward in Tensorflow or Keras as it is in Pytorch.

4.1.1 Spectrum of Adversaries under Test

To this end, we describe hereafter the different adversaries that we consider.

- **White-box.** This model is depicted in Figure 2b. Each branch model m_{θ_i} is learned independently from each other, based on a restricted amount of corresponding P.o.Is, and by minimizing the loss function $\mathcal{L}_{y_i}$ based on the corresponding share y_i. Concretely, each branch model is instantiated with a one-hidden-layer Multi-Layer Perceptron (MLP) with $N = 1,000$ neurons, a Rectified Linear Unit (ReLU) activation function on the hidden layer, and a softmax activation function on the output layer. Following the standard practise in DL, Batch Normalization (BN) is also applied before ReLU. Once all branch models are trained, they are passed through GroupRecombine to infer on the validation traces.

- **Grey-box.** It is the same as the white-box model, but the trainings of the branch models are done *jointly*, using the loss function $\mathcal{L}_y$ computed from the labels of the target y, as the random nonces are no longer known. To better evidence the advantages of our grey-box model, we decline three different versions:

  - **Known P.o.Is and scheme.** This corresponds to the model depicted in Figure 1. Each branch model is only fed with the appropriate P.o.Is, and the recombination is done with GroupRecombine.
  - **Known masking scheme only.** This corresponds to the model depicted in Figure 3b. This is the same model as the previous one, except that each branch model is fed with raw traces, instead of separate P.o.Is.
  - **Known P.o.Is only.** This corresponds to the model depicted in Figure 3a. The model is the same as with the one with known P.o.Is and masking scheme, except that the GroupRecombine layer is replaced by another one-hidden-layer MLP with $N' = 100$ neurons.

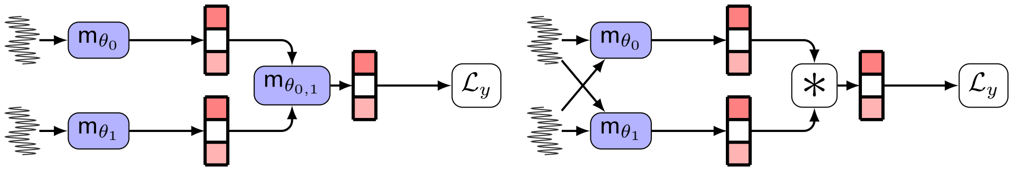
  The two latter versions are downgraded compared to the former one. Therefore, if our grey-box model is sound, it is expected to work better than its downgraded versions.

- **Black-box.** This model is depicted in Figure 2a. Since the adversary is not assume to know neither the P.o.I location, nor the underlying masking scheme, the black-box model is a one-hidden-layer MLP that is fed with raw traces. For consistency with white-box and grey-box models with known P.o.Is and scheme, we keep the number of hidden neurons constant. Therefore, our black-box MLP has $(d + 1) \cdot N$ neurons.

Note that in GroupRecombine, the approximation error is null. This means that provided that each branch model m_{θ_i} can exactly computes the true leakage model $\mathsf{p}_{\mathrm{Y}_i}(\boldsymbol{l}_i)$, then the whole grey-box model using GroupRecombine exactly implements the true conditional p.m.f. of the target variable Y. On the contrary, since the discrete convolution is a non-linear mapping, it cannot be exactly approximated by a MLP [Yar17, Thm. 6].

4.1.2 Performance Metrics and Quantifying Complexity

To assess and compare two models, we measure the Perceived Information (PI), as it has been shown to be strongly related to the minimum number of traces required to succeed a key recovery with a Maximum Likelihood Distinguisher (MLD) [MDP19a]. We also have the following relationship with the Negative Log Likelihood (NLL) loss function $\mathcal{L}(\mathsf{m}_\theta)$ used for training: $\mathsf{PI}(\mathrm{Y}; \mathbf{L}; \mathsf{m}_\theta) = n - \mathcal{L}(\mathsf{m}_\theta)$, where n stands for the bit-size of the target variable Y.



(a) Grey-box: known P.o.Is only.

(b) Grey-box: known encoding only.

Figure 3: "Downgraded" grey-box models. In blue: blocks with trainable parameters.

Based on this performance metric, we can assess the complexity of an adversary in the following two ways. First, we assess the *optimization* complexity by plotting optimization curves, *i.e.* the loss function with respect to the number of steps in the Gradient Descent (GD)-based optimization algorithm. For consistency when the number N_p increases, we use optimizers with full-batch, therefore one optimization step always equals one epoch. Following the empirical study of Perin and Picek [PP20], we consider the Adaptive Moment Estimation (Adam) optimizer [KB15], with a learning rate of 10^{-4}. Second, we assess the *profiling* complexity. It is measured in terms of the amount of profiling traces needed to reach the optimal value of PI. To this end, we will plot *learning curves* depicting the evolution of the PI with respect to the number N_p of profiling traces used to train the model. For each point of the learning curve, the PI reported is the one that is obtained by the model at the epoch where it got the minimal value of its training loss.[5]

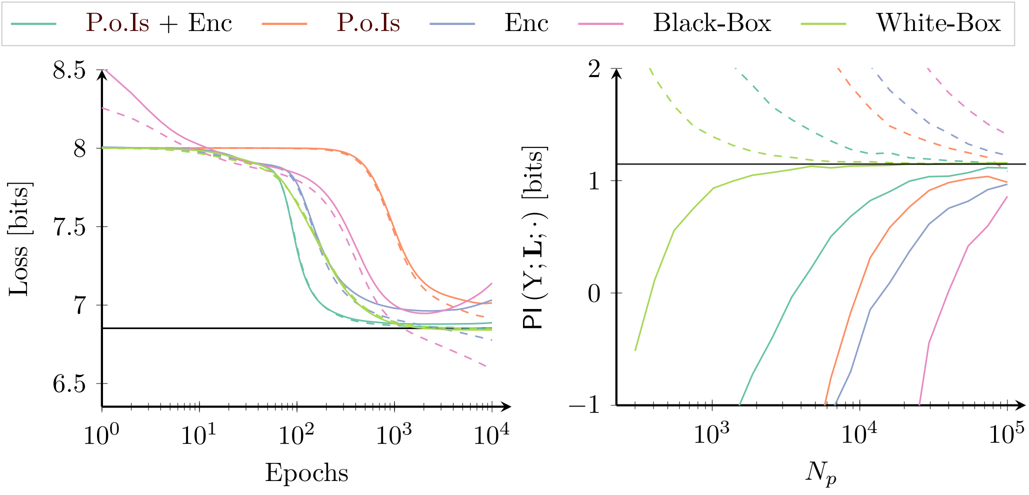
4.2 Results on Simulation

We start by presenting our simulations. The dataset is simulated as follows. We draw uniformly some sharings according to the desired masking scheme. Then, we apply on each share the famously used Hamming weight leakage model, with an additive Gaussian noise with a small standard deviation of $\sigma = 0.1$. Since the leakage model is known in our simulated framework, we can also compute a Monte-Carlo (MC) estimation of the Mutual Information (MI), using the true Probability Density Function (p.d.f.) used to sample the simulated traces.

4.2.1 First-Order Boolean Masking

The results for two 8-bit Boolean shares are depicted on Figure 4. We can see on Figure 4a that the training and validation losses of the grey-box model (in green) enjoy the fastest convergence towards the black horizontal line depicting the theoretical minimum value of the validation loss. On the other hand, the validation loss for the black-box MLP (in pink) requires roughly $2,000$, epochs to reach its minimal value, which was attained by the green curve in 10 times less epochs. Worse, we can see that the pink validation loss of the black-box model cannot reach the optimal value of the loss depicted in black with only $N_p = 10^5$ profiling traces, contrary to the grey-box model. This is a sign of over-fitting. To better quantify this phenomenon, we show in Figure 4b the learning curve. It tells us that a white-box adversary would need 400 profiling traces to get a model with positive PI, whereas the best grey-box model would require at least $3,000$ profiling traces to reach the same threshold. But more interestingly, a fully black-box model would require another 10 times more profiling traces. In other words, the results presented in Figure 4 show that

[5] We stress that a learning curve usually does not record the metrics at the epoch where the validation loss is minimal. This is equivalent to assessing the models quality when they are not combined with any other regularization method, in order to make things comparable. We verified that when using early-stopping, the resulting learning curve does not change much in terms of hierarchy, as it can also be verified when looking at the minimal values of the validation losses in Figure 4a.



(a) Optimization curves for $N_p = 10^5$.

(b) Learning curves.

Figure 4: Comparison between models for a first-order Boolean masking.

both profiling and optimization complexity between grey-box and black-box models may differ by an order of magnitude.

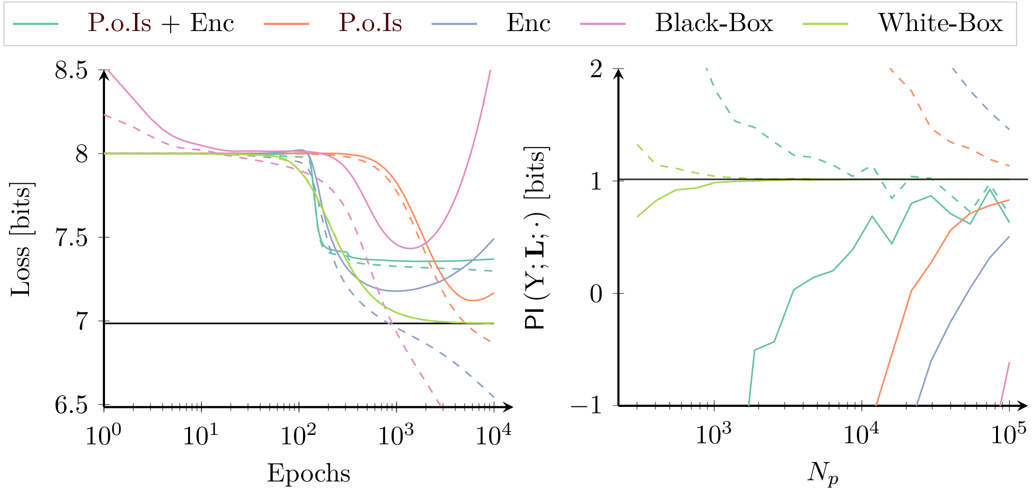
4.2.2 Second-Order Boolean Masking

We push our simulated experiments one step forward, adding a third Boolean share into the leakage. The results are shown in Figure 5. When analyzing the learning curve in Figure 5b, we may notice two main differences compared to the 2 share simulation. First, the pink, blue and orange curves have been shifted to the right, meaning that their corresponding profiling complexity has been increased by an order of magnitude. Interestingly, the green curve denoting the best grey-box model has merely been shifted, meaning that its profiling complexity did not change much compared when adding a third share into the experiment. Nevertheless, as a second observation, it is noticeable that the green learning curve seems less *smooth* than in the previous experiment. This testifies an increase in the optimization complexity. To understand why, we plot the optimization curves for $N_p = 10^5$ on Figure 5a. Here again the green curves standing for the best grey-box model quickly leave the initial plateau after 100 epochs, but fall stuck in another plateau for the remaining epochs. Hopefully, this phenomenon does not always happen, since the learning curve in Figure 5b reports better performances for some lower values of N_p.

4.2.3 Affine Masking

We then move our simulation from a second order Boolean masking scheme to an affine scheme. Hereupon, Bronchain and Standaert argued that learning an affine scheme with a black-box model turns out to be hard, as emphasized by their simulations [BS20]. There, the authors considered a slightly different leakage model for the multiplicative share α of the affine sharing.[6] Indeed, from their experimental measurements, they were able to recover the multiplicative share with almost 100% accuracy with their white-box attack, meaning that the leakage model of the multiplicative share α was injective. This can

[6] For completeness, we also provide in supplementary material the results of our simulation without changing the leakage model. The conclusions remain the same.



(a) Optimization curves for $N_p = 10^5$.

(b) Learning curves.

Figure 5: Comparison between models for a second-order Boolean masking.

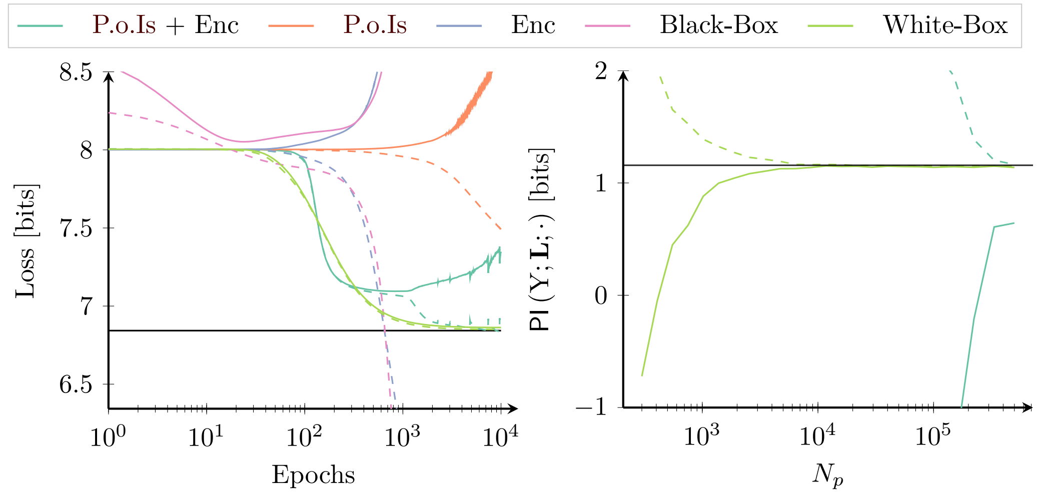
be explained by the fact that the concrete implementation of all affine schemes known in the literature are table-based [FMPR11, MS21], meaning that the values $x \times \alpha$ are sequentially processed for $x \in [\![1, 2^n - 1]\!]$ during the pre-computation phase, which may leak a lot [TWO14].

To take this into account in their subsequent simulations, Bronchain and Standaert emulated the leakage of the multiplicative share α with an identity model. Nevertheless, Cristiani *et al.* experimentally showed that learning such a leakage model with DNNs could be hard [CLM20, Fig. 3], whereas replacing the identity by a one-hot encoding could make the problem much easier. A similar experiment on images, conducted under the *coordinate transform problem* terminology, led to similar conclusions [LLM+18]. This suggests that beside not being physically realistic, the identity leakage model could make the problem artificially much harder. That is why we revisit Bronchain and Standaert's experiment by changing the leakage model regarding the multiplicative share. Hereafter, additionally to the leakages on the other shares, we consider that the adversary has access to the values of $\mathsf{hw}(x \times \alpha)$, for $x \in [\![1, 2^n - 1]\!]$, where hw denotes the Hamming weight leakage model.[7] The simulation results are presented in Figure 6. While our black-box model is not able to get a positive PI, as depicted by the pink curve on Figure 6a, we can see that our grey-box model leveraging the knowledge of both P.o.Is and masking scheme is able to get a validation loss below the 8-bit random threshold, denoting an effective model. Even though it does not contradict the previous conclusions of Bronchain and Standaert regarding the efficiency of black-box models, the outcomes of our grey-box model show that not having access to the random nonces does not necessarily lead to an unsuccessful attack.

4.2.4 Does Each Branch Actually Learn True Leakage Distributions?

In view of the good results obtained by our GroupRecombine in our simulations, we may wonder whether the intermediate output p.m.f.s returned by each branch in a grey-box

[7] Note that this leakage model seems to behave similarly to a one-hot encoding, as it is possible to guess the value of α by looking at the entry x of the leakage for which the leakage equal n. This would denote the value x such that $x \times \alpha = 2^n - 1$. Therefore, this leakage model somehow acts like a one-hot encoding, up to a permutation of the encoding entries.



(a) Optimization curves for $N_p = 5 \cdot 10^5$.

(b) Learning curves.

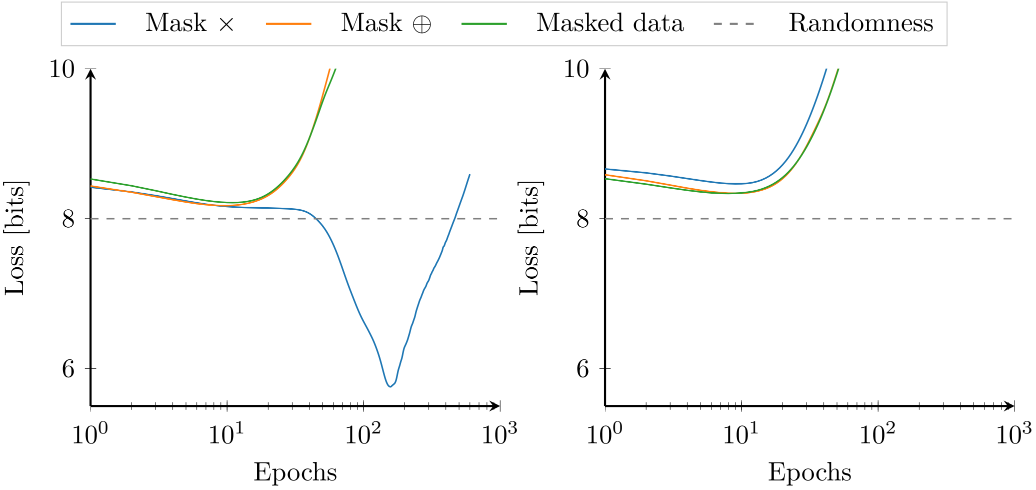
Figure 6: Comparison between models for an affine masking.

attack — in the known-P.o.Is setting — were also good estimates of the true p.m.f. of each share. Can we compare the performance of our grey-box attacks against a white-box adversary on each share separately?

At first sight, there is no reason why it would be possible, since the mapping $(\mathsf{p}, \mathsf{p}') \mapsto \mathsf{p} * \mathsf{p}'$ is not invertible. As an example, if τ_h denotes the translation operator, *i.e.*, $\tau_h(\mathsf{p}) = \mathsf{p}[\cdot \star h]$, then the convolution product is known to be *co-variant* with translation, *i.e.*, $\tau_h(\mathsf{p}) * \mathsf{p}' = \tau_h(\mathsf{p} * \mathsf{p}')$. As a corollary, we have $\tau_h(\mathsf{p}) * \tau_{h^{-1}}(\mathsf{p}') = \tau_{h \star h^{-1}}(\mathsf{p} * \mathsf{p}') = \mathsf{p} * \mathsf{p}'$. In other words, even if our grey-box model could reach the optimal performance, the leakage models on each share could at best be learned up to a shift of the probabilities.

But what about with affine masking? Here the additive and multiplicative shares do not play the exact same role, so the argument about translation covariance no longer holds. Does this mean that some branches could actually learn the true leakage model on their respective share? To clarify this question, we also monitored the loss function on each branch output against the labels of each share, during training. These branch losses were computed over the validation set. We plot on Figure 7 such metrics, monitored during the simulation in affine masking of Subsubsection 4.2.3. We notice in Figure 7a that the loss for the branches of additive shares in orange and green diverge, as expected earlier. But surprisingly, the loss on the multiplicative branch in blue goes below the 8-bit randomness threshold before starting over-fitting. This means that the multiplicative branch may be used in this case to infer on the value of the multiplicative share, despite not having known at all the values of the multiplicative share during training. Interestingly, we also note that the branch loss in blue escapes its plateau after 50 epochs, whereas at the same epoch in Figure 6a, the overall training and validation losses for the grey-box model are still stuck on the plateau. This denotes that at this epoch, the learning has somehow started, even if this does not reflect in the value of the overall loss.

Actually, our observations should be slightly mitigated, as they seem to be leakage-model-dependent. Indeed, we replicated our simulation, by replacing our injective leakage model for the multiplicative share by a simpler (non-injective) Hamming weight leakage model. The corresponding results shown on Figure 7b indicates this time that the multiplicative branch is not able to reach a positive PI, so it cannot be used to infer on the multiplicative share in this case. Still, we argued in Subsubsection 4.2.3 that the latter



(a) With injective leakage model for α. (b) With Hamming weight leakage model for α.

Figure 7: Branch losses.

leakage model is less representative than the former one.

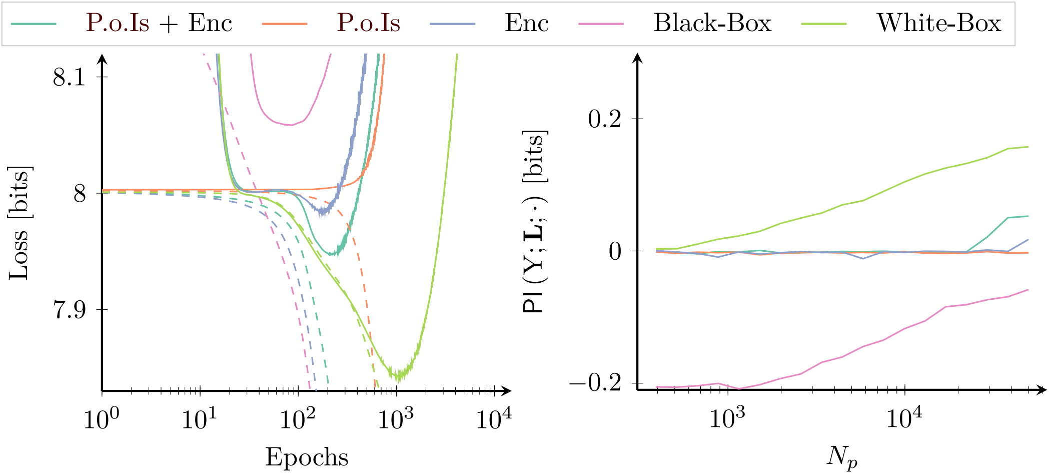
4.3 Application on Experimental Data

Now we established the interest of GroupRecombine on simulations, we would like to test it under experimental traces. To this end, we replicate the same experiments as with the simulation described in Subsection 4.2 on some public datasets using masking.

4.3.1 Experiments on ASCAD-v1

We start with the ASCAD-v1 dataset, published in 2018 by Benadjila *et al.* [BPS+20]. It deals with a first-order masked implementation of AES, with a Boolean scheme based on table re-computation. The cryptographic primitive is implemented on an 8-bit ATMega8515 device on which the Electro-Magnetic (EM) field emanations are measured. Two versions of the dataset are proposed: one so-called *fixed* with measurements acquired on a 700 time samples window using a fixed encryption key, and a *variable* dataset with measurements on a $1,400$ time samples window using a variable encryption key for profiling traces. Both windows cover the look-up of the re-computed Sbox when applying the SubBytes operation on the third byte of the AES state during the first round. Since on both datasets, the data dimensionality is much higher than in our simulations whereas the number of profiling traces remains of same order of magnitude as in our simulations, the DNNs used in our simulations are more likely to over-fit. That is why we reduce the number of neurons in the hidden layer of the branches from $1,000$ to 100.

Results on Fixed Dataset. We report hereafter the outcomes of our trainings on the fixed dataset. When the threat model assumes to know the P.o.I location, the P.o.I selection has been done by splitting the 700 time samples into two halves, the first 350 time samples containing some leakages about the masked share while the second 350 time samples containing leakages about the mask. This pre-processing is suboptimal compared to a P.o.I selection with SNR, but reflects more what an adversary can do with a visual trace analysis with the help of the source code, and can even be further refined with a thorough code analysis, as argued in Subsection 2.2. The results are depicted on Figure 8. First, it



(a) Optimization curves for $N_p = 5 \cdot 10^4$.

(b) Learning curves (best validation loss).

Figure 8: Comparison between models for ASCAD-v1.

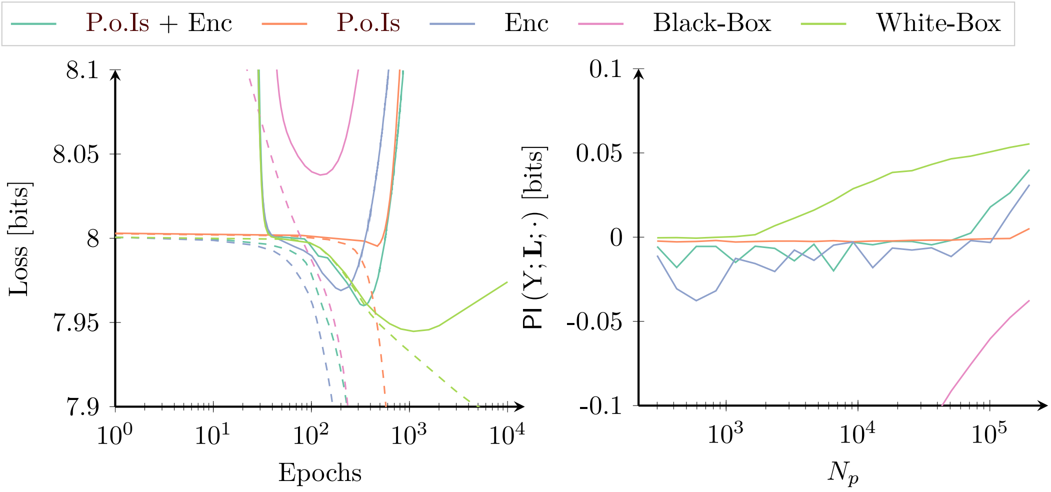
can be seen on Figure 8a that when using the full profiling trace set — *i.e.*, $N_p = 50,000$ — the validation loss eventually diverges, meaning that when using only shallow MLPs like in our experiments, none of the different threat models would lead to an effective attack without further pre-processing. Nevertheless, we can see that the validation losses in light green, green and blue have their minimum value below the 8-bit threshold, meaning that selecting the best model based on a validation loss would eventually lead to successful attacks.

Moreover, we can still observe the same hierarchy between the threat models as in the simulations conducted in Subsubsection 4.2.1. The white-box model leads to a PI close to 0.2 bit, which is the highest lower bound of MI reported in the literature on ASCAD-v1 [CLM20]. Then, the grey-box model leveraging both P.o.I location and knowledge of the masking scheme reaches a PI of 0.05 bits, whereas the grey-box model exploiting the knowledge of the masking scheme only obtains a PI of 0.02 bits. Finally, the grey-box model leveraging the P.o.Is location only and the black-box model cannot get a positive PI during the whole training.

We validate this observations by reproducing the trainings on a lower number of profiling traces. For each of these trainings, the best PI on the validation loss is reported on Figure 8b. As can be noticed, the previous observations regarding the hierarchy of the threat models remains true.

Results on Variable Dataset. We repeated the same experiments on the variable dataset. We first tried by splitting the traces into two contiguous parts of 700 points each, as with the fixed dataset. Unfortunately, none of the non-white-box models could succeed in getting a positive PI, which suggests that our P.o.I selection method for our grey-box attacks was not sufficient, at least for the amount of traces available in this public dataset. Therefore, we refined the P.o.I selection, by narrowing the windows for the two shares. For the share $\mathsf{r}_{\mathsf{out}}$, we selected the range $[\![0, 300]\!]$, whereas for the share $\mathrm{Y} \oplus \mathsf{r}_{\mathsf{out}}$, we selected the range $[\![900, 1200]\!]$. This refined P.o.I selection is possible even without knowledge of $\mathsf{r}_{\mathsf{out}}$ during profiling, thanks to a careful assembly code analysis, similar to the one recently conducted by Egger *et al.* [EST$^+$22, Fig. 4]. For consistency in our comparisons, we also feed the black-box model with the two restricted P.o.I windows, stacked together.

We then report on Figure 9 the results obtained for this second attempt on the variable



(a) Optimization curves for $N_p = 1.5 \cdot 10^5$.

(b) Learning curves (best validation loss).

Figure 9: Comparison between models for ASCAD-v1 (variable).

dataset. Like with the fixed dataset, we can see that the black-box model is unable to get a positive PI with the 600 P.o.Is given as an input. On the contrary, it took 70,000 profiling traces to get a positive PI for our best grey-box model (green curve in Figure 9b). This is much more than for the white-box model that only required 2,000 profiling traces (light green curve). Still, the results obtained on both ASCAD-v1 datasets confirm that it is possible to succeed an attack by using small MLPs, provided that the recombination is cleverly done, *e.g.* with GroupRecombine.

4.3.2 Attempts on ASCAD-v2

Provided with the promising results presented in Figure 6 on the affine masking, and on the good experimental verifications on the ASCAD-v1 datasets, we pushed our experiments one step further by trying to replicate the attack of affine masking on an actual implementation of the affine masking. To this end, we used the ASCAD-v2 dataset [MS21]. It is made of 500,000 traces, each having 15,000 time samples coming from two contiguous parts of the raw power consumption traces acquired on an STM32 Cortex-M3 device. There, the authors explain that the first window covers P.o.Is of the multiplicative share only, whereas the second window covers P.o.Is of the additive share and the masked data. Since the implementation also uses shuffling to protect the sensitive data, we artificially deactivate the latter counter-measure, by relabeling the masked data thanks to the knowledge of the random seeds used for permutation.

Unfortunately, we could not get effective attacks using grey-box models with simple MLPs as branch models. This negative result should nevertheless been mitigated. Indeed, the authors of [MS21] reported some successful white-box attacks on the dataset, leveraging the knowledge of at least one share during profiling, but did not succeed in attacking the dataset with a fully black-box model. To the best of our knowledge, no successful attack has ever been reported using non-white-box models, since the release of the dataset in early 2021. We will discuss in Section 5 the potential reasons behind this difficulty.

5 Discussion

We have seen that using grey-box adversaries could mitigate some drawbacks of black-box adversaries. In this section, we discuss some parts of our results. Subsection 5.1 argues that changing the type of DNN architecture in the branches of a grey-box model should not affect the comparative advantage of GroupRecombine with respect to the black-box approach. Finally, Subsection 5.2 questions to what extent non-white-box approaches could efficiently work against higher-order masking schemes.

5.1 On the Choice of Architecture for the Branches.

In our experiments involving grey-box attacks, we used the same architecture for the branches of our GroupRecombine model, namely a one-hidden-layer MLP with 100 or $1,000$ neurons. Naturally, better performances could have been obtained by investigating other types of DL architectures. As a consequence, our black-box models are not necessarily the best ones. Actually, our results report unsuccessful black-box attacks with one-hidden-layer MLPs on ASCAD, whereas the literature reports much better results on this dataset, by using deeper MLPs — up to 6 layers according to Benadjila *et al.* [BPS+20] — or CNNs [KPH+19, MDP19b].

Therefore, one might wonder whether our comparison is biased towards GroupRecombine. Hereupon, we stress that all trainable models depicted in Figure 1 and in Figure 3 have been instantiated with the simplest DNN architecture one may use. The fact that the white-box models instantiated with such simple branch models reached the optimal performances on our simulations, and reached levels of PI close to the state of the art on the ASCAD-v1 dataset. Naturally, it is possible to use other branch models, such as CNNs, with GroupRecombine. But our experiments suggest that using shallow MLPs is often sufficient for optimal leakage modeling — at least provided that other types of counter-measures are ignored. In other words, this suggests that the main efforts spent by the DL practitioner in designing more complex architectures in a black-box setting would actually serve at learning how to recombine the informations gathered by the first layers of the DNN on each share, according to the masking scheme. Hence, by hard-encoding the masking scheme with GroupRecombine, we expect the DL practitioner to spare a significant amount of time spent, *e.g.*, in running exhaustive/random search of hyper-parameters, which is acknowledged to be the bottleneck task in DL-based SCA [BPS+20, RWPP21, WPP20].

Likewise, no regularization technique — *e.g.*, weight decay, dropout — have been considered in this study, so adding them could have naturally improved the results. However, we argue that the effect of regularization techniques is *orthogonal* to the effect of using GroupRecombine. Indeed, beside being hyper-parameter-free contrary to other types of regularization, our recombination layer does not act on the bias-variance trade-off, as most of regularizers do [SB14, Chap. 5]. This means that, provided that the assumed masking scheme is the right one, the regularization effect of the recombination layer never degrades the approximation capacity of our model as argued at the end of Subsubsection 4.1.1, contrary to what all other types of regularization are likely to do.

5.2 The Initial *Plateau*: An Effect of Masking

It turns out that all the optimization curves presented so far in this paper depict an initial *plateau* for both training and validation loss. Namely, when targeting some leakage induced by masking, the GD-based optimization algorithm starts its procedure being stuck on a plateau whose level coincides with full randomness. This plateau is not a simulation artifact, as it can be observed in many other studies investigating black-box adversaries against masking. See, *e.g.*, the works of Timon [Tim19, Fig. 5], Perin and Picek [PP20, Figs. 5b, 6b, 7, 8, 13], Cristiani *et al.* [CLM20, Fig. 8], [CLM21, Fig. 3], and

Lu *et al.* [LZC+21, Fig. 10 – 11, left]. Moreover, some of these figures suggest that the higher the masking order, the longer the initial plateau. Interestingly, the initial plateau barely happens when targeting unprotected implementations, or even leakages protected by shuffling [MDP19a, Fig. 2, right] or de-synchronization [CDP17, Fig. 8], [KPH+19, Fig. 10], [MBC+20, Fig. 6], suggesting this plateau is closely linked to the use of masking. These intriguing observations call for further explanation and verification: is this plateau really due to masking, and if so to what extent it affects the optimization?

5.2.1 Empirical Verification on Exhaustive Datasets

To address these questions, we repeat our simulations for a Boolean masking, by using here a noise-free, *exhaustive* dataset, *i.e.* for which the training and validation loss are equal — which is made possible thanks to the discrete nature of our noise-free leakage model. Thus, the profiling complexity is nullified, allowing to focus only on the optimization complexity.[8] The setting of this simulation is voluntarily much simpler rather than realistic, so that the optimization complexity may be seen as a lower bound of what a real-world adversary could expect.

In order to quantitatively measure the optimization complexity, we define the *weak learning* threshold, set to $\mathcal{L}(\theta) = n - \epsilon$. The weak learning threshold corresponds to an adversary with an *effective* model, *i.e.* a model wit strictly positive PI, up to an ϵ-margin. More concretely, the weak learning threshold can be used to measure the length of the plateau in the optimization curve. We also define the *strong learning* threshold, set to $\mathcal{L}(\theta) = \mathsf{MI}(\mathrm{Y}; \mathbf{L}) + \epsilon$. The strong learning threshold corresponds to an optimal adversary from an information theoretic point of view, up to an ϵ-margin. Hence, the optimization complexity can be measured in terms of the number of epochs required to reach the weak and strong learning thresholds.

The results of our simulations with an exhaustive dataset are showed on Figure 10, for $\epsilon = 0.05$, and averaged over 5 different seeds. Note that here we fixed the number of neurons in the black-box model, so the curves black-box and grey-box models are not directly comparable between each other. For the grey-box model, we can observe that the green plain curve goes from 1 epoch for $d = 0$ to 4,000 epochs for $d = 4$. This denotes an exponential increase of the optimization complexity in weak learning with the masking order. Since the optimization complexity in strong learning (denoted by the dotted curves)

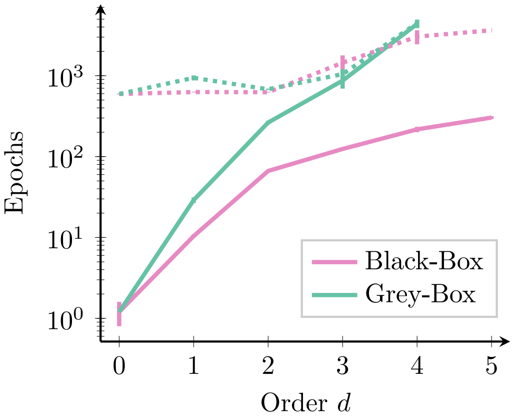


Figure 10: Number of epochs to get weak (plain curves) and strong (dotted curves) learning.

[8] This setting therefore emulates the common assumption of an adversary with *unlimited* profiling power. The interested reader may find in Appendix B a discussion about how to efficiently emulate this case.

is strictly higher than the one in weak learning, we can also deduce that the optimization complexity in strong learning will follow an exponential trend.

For the black-box model, we may also notice a dramatic increase of the optimization complexity in weak learning, from one epoch without masking to 200 epochs with 6 shares. Unfortunately, Figure 10 does not provide enough evidence to conclude in a sharp way on the same exponential increase of the optimization complexity as observed with grey-box models. Indeed, as the size of the exhaustive dataset also increases exponentially, it no longer fits into our 48 GB Nvidia RTX A6000 GPU when $d \geq 6$. Does this suggest that black-box models could efficiently scale with the masking order in terms of optimization complexity, whereas grey-box models do not?

5.2.2 A Theoretical Argument

Actually, we argue in this section that black-box models should also face the exponential increase of optimization complexity with respect to the masking order. Our point relies on a theorem proved by Shalev-Shwartz *et al.*, in an almost similar problem [SSS17]. There, the authors investigated to what extent some tasks may be learned in an "end-to-end" manner — *i.e.* black-box in our terminology —, or by "decomposition" in more elementary learning problems — *i.e.* white-box in our terminology — would be more efficient. They emphasized that some problems could be hard to learn with gradient descent in black-box, as stated hereafter.

Theorem 1 ([SSS17, Thm. 3], informal)**.** *Let* $\mathbf{L}$ *denote a* d*-tuple* $(\mathbf{L}_1, \dots, \mathbf{L}_d)$ *of input instances, and assume that each* $\mathbf{L}_i$ *is i.i.d. standard Gaussian in* $\mathbb{R}^p$*. Define the target function* $h_{\boldsymbol{u}}(\boldsymbol{l}) = \prod_{i=1}^{d} \operatorname{sign}(\boldsymbol{u}^\intercal \boldsymbol{l}_i)$ *, for some normalized hyperplane* $\boldsymbol{u} \in \mathbb{R}^p$*. Let* m_θ *be a predictor differentiable with respect to its parameter* θ*, such that* $\mathbb{E}_{\mathbf{L}}\left[\|\nabla_\theta \mathsf{m}_\theta\|^2\right] \leq G(\theta)^2$*, and let* $\mathcal{L}(\theta) = \mathbb{E}_{\mathbf{L}}[\ell(\mathsf{m}_\theta(\mathbf{L}), h_{\boldsymbol{u}}(\mathbf{L}))]$ *be the loss function to minimize, for some smooth function* $\ell(\cdot)$*. Then,*

$$\mathbb{E}_{\boldsymbol{u}}\left[\left\|\nabla_\theta \mathcal{L}(\theta) - \mathbb{E}_{\boldsymbol{u}}[\nabla_\theta \mathcal{L}(\theta)]\right\|^2\right] \leq G(\theta)^2 \cdot \mathcal{O}\left(\sqrt{\frac{d\log(p)}{p}}\right)^d . \tag{3}$$

Let us interpret the meaning of this theorem from our SCA point of view. Consider a bit, masked with d shares, and assume that the leakage distribution conditionally to each share is noise-free and the same for each bit. Therefore, the leakage model to be learned is denoted by the decision surface materialized by the hyperplane $\boldsymbol{u}$, and the masked target bit can be expressed as $h_{\boldsymbol{u}}(\boldsymbol{l})$.[9] Note that in a worst-case scenario where the adversary has unlimited profiling powers, the true decision surface $\boldsymbol{u}$ is known, so the masked bit would be successfully recovered in one trace in the attack phase.

Yet, what Theorem 1 tells us is that the success rate could be much worse for a real-world adversary with limited computational powers using gradient descent. Indeed, the authors of [SSS17] interpret the left hand-side of Equation 3 as a measure of the feedback *signal* returned by the labels through the gradient of the loss function. Then, this result tells us that this feedback signal decreases exponentially fast with the masking order, provided that the dimensionality p of each sub-leakage is high enough.

As a result, the trajectory taken by the parameter θ during the gradient descent depends less and less on the features of the target function to learn, denoted by the vector $\boldsymbol{u}$, as the masking order increases. Abbe *et al.* showed at NEURIPS'20 that this exponential decrease in the feedback signal would result in an exponential growth in the number of steps in the gradient descent needed to escape the initial plateau [AS20, Thm. 3, Cor. 2]. , [AKem+21, ACHM22]. This suggests that the non-exponential trend observed in Figure 10

[9] $h_{\boldsymbol{u}}$ can be seen as the xor function, where the inputs are remapped over $\{-1, +1\}$.

can be regarded as a simulation artifact since we had $p = 1$, whereas in most non-white-box attacks the dimensionality p of each leakage is typically much higher, such that we have $\frac{d \log(p)}{p} \ll 1$. Overall, we conclude that the hardness when tackling profiling attacks in non-white-box settings is mainly due to the optimization procedure, *i.e.* based on gradient descent in this paper, than on the choice of models.

6 Conclusion: is Non-White-Box Profiling Hard Anyway?

In this paper, we have shown how a real-world adversary could leverage some prior knowledge from the source code of a target implementation, by substituting fully black-box with grey-box adversaries. As a result, we evidenced how grey-box models could dramatically improve the efficiency of a side-channel attack, from a profiling complexity point of view. We also showed that the main difficulty for the adversary is mainly due to the drawbacks of GD-based optimization procedures, rather than the selection procedure of the appropriate hyper-parameters of a model. Moreover, this difficulty is expected to increase exponentially with the masking order.

Beyond raising a great challenge for the SCA evaluator, this also opens a new interesting question for the SCA developer:

> *Is the conditional p.m.f. of an intermediate computation that is protected against masking efficiently learnable in a non-white-box setting?"*

Namely, is there any alternative to GD-based optimizers for which both profiling and optimization complexities scale at most polynomially with the masking order? If such a learning algorithm exists, then this would suggest that all the current approaches in non-white-box settings would lead to a false sense of security in a near future when higher-order masking schemes will become more and more affordable for developers and designers. But if no efficient learning algorithm exists, this would pave the way towards a new notion of *black-box* SCA security, in which masking is sound provided that the adversary does not have access to the random nonces used for encryption during the profiling phase. Such a new security notion would have the advantage of holding unconditionally to the level of physical noise, thereby leading to less conservative security bounds, *e.g.* for software implementations.

Which of these answers is the right one? Even if each outcome would have major implications in either the design or evaluation of embedded devices against SCA, we do not have a clear answer yet to this question. Indeed, we gathered evidences towards both eventualities, as discussed hereafter.

On the one hand, our problem is somehow close to the well-known Learning Parity with Noise (LPN) problem. This problem is cryptographically hard, as it is the root for some lattice-based cryptographic primitives [GRS08, Pie12]. Yet, without noise this problem becomes efficiently learnable, as it can be solved with Gaussian elimination. Nevertheless, it has been shown that any GD-based approach would result in an exponential optimization complexity [Tho96], [AS20, Thm. 6], [SSS17, Thm. 1]. This provides an example of easy learning problem where GD-based learning can fail. On the other hand, some recent results in learning theory suggest that profiling masked implementations in non-white-box settings could be hard regardless of the nature of the learning algorithm used by the adversary. Indeed, under the assumption that the leakage model has an additive Gaussian noise, the leakage distribution can be expressed as a Gaussian mixture, with a number of modes increasing exponentially with the masking order. Some recent works studying Gaussian mixture learning showed that in such a case, there is no learning algorithm able to scale polynomially with the number of modes, both in terms of computational and profiling complexity [BRST21, GVV22]. In other words, any generative model used for profiling in a non-white-box setting, is prone to fail when facing higher-order masking, regardless of the profiling method used by the adversary. Whether this limitation also translates to discriminative models like MLPs or CNNs remains the main open question we leave for future works.

Acknowledgments

François-Xavier Standaert is a Senior Associate Researcher of the Belgian Fund for Scientific Research (FNRS-F.R.S.). This work has been funded in part by the ERC project number 724725 (acronym SWORD).

A Duality between Masking and Convolutions

Proof of Proposition 1. Let $\mathsf{p}_{\mathrm{Y}}(\boldsymbol{l})[y]$ denote the probability $\Pr(\mathrm{Y} = y \mid \mathbf{L} = \boldsymbol{l})$, and let $\overline{\mathrm{Y}}$ denote the $d+1$ sharing of Y, namely $\overline{\mathrm{Y}} = (\mathrm{Y}_0, \ldots, \mathrm{Y}_d)$. By applying Bayes' theorem we get:

$$\mathsf{p}_{\mathrm{Y}}(\boldsymbol{l})[y] \quad = \quad \frac{\Pr(\mathrm{Y} = y)}{\Pr(\mathbf{L} = \boldsymbol{l})} \Pr(\mathbf{L} = \boldsymbol{l} \mid \mathrm{Y} = y) \tag{4}$$

Denoting $\Pr(\mathbf{L} = \boldsymbol{l} \mid \mathrm{Y} = y)$ by $\mathsf{p}_{\mathbf{L}}(y)[\boldsymbol{l}]$, and using the total probabilities formula d times, we expand the latter term as follows:

$$\mathsf{p}_{\mathbf{L}}(y)[\boldsymbol{l}] = \sum_{y_1, \ldots, y_d \in \mathcal{Y}} \Pr(\mathbf{L} = \boldsymbol{l} \mid (\mathrm{Y}, \mathrm{Y}_1, \ldots, \mathrm{Y}_d) = (y, y_1, \ldots, y_d)) \cdot \prod_{i=1}^{d} \Pr(\mathrm{Y}_i = y_i) \quad . \tag{5}$$

By noting $y_0 \triangleq y \star (y_1 \star \ldots \star y_d)^{-1}$, and since the mapping $(y, y_1, \ldots y_d) \mapsto (y_0, y_1, \ldots y_d)$ is invertible we may reformulate the conditional probability as follows:

$$\Pr(\mathbf{L} = \boldsymbol{l} \mid (\mathrm{Y}, \mathrm{Y}_1, \ldots, \mathrm{Y}_d) = (y, y_1, \ldots, y_d)) \quad = \quad \Pr\left(\mathbf{L} = \boldsymbol{l} \mid \overline{\mathrm{Y}} = (y_0, \ldots, y_d)\right) \ . \tag{6}$$

Moreover, according to assumption (b), we have:

$$\Pr(\mathbf{L} = \boldsymbol{l}) \quad = \quad \prod_{i=0}^{d} \Pr(\mathrm{L}_i = l_i) \quad , \tag{7}$$

$$\Pr\left(\mathbf{L} = \boldsymbol{l} \mid \overline{\mathrm{Y}} = (y_0, \ldots, y_d)\right) \quad = \quad \prod_{i=0}^{d} \Pr(\mathrm{L}_i = l_i \mid \mathrm{Y}_i = y_i) \quad . \tag{8}$$

Finally, we may use the uniform assumption to remark that: $\Pr(\mathrm{Y} = y) = \Pr(\mathrm{Y}_0 = y_0)$. We may now combine Equations (4), (5), (6), (7), (8) and the latter fact:

$$\mathsf{p}_{\mathrm{Y}}(\boldsymbol{l})[y] \quad = \quad \sum_{y_1, \ldots, y_d \in \mathcal{Y}} \mathsf{p}_{\mathrm{Y}_0}(\boldsymbol{l}_0)\left[y \star (y_1 \star \ldots \star y_d)^{-1}\right] \mathsf{p}_{\mathrm{Y}_1}(\boldsymbol{l}_1)[y_1] \cdot \ldots \cdot \mathsf{p}_{\mathrm{Y}_d}(\boldsymbol{l}_d)[y_d] \quad .$$

□

B Optimizing Simulations in An Exhaustive Dataset

Even for a noise-free leakage, computing the loss function to minimize in a naive way would become quickly intractable, as it would result in a sum over all possible sharings of Y, *i.e.*, $2^{n \cdot (d+1)}$ terms.

Hopefully, we can do much better in our simulated framework, as the conditional probability distribution $\Pr(\mathrm{Y} \mid \mathbf{L})$ and the marginal distribution of leakages $\Pr(\mathbf{L})$ can be used to rephrase the terms in the loss function as follows:

$$\mathcal{L}(\theta) = \sum_{l_0} \dots \sum_{l_d} \underset{\Pr(\mathrm{Y} \mid \mathbf{L}=(\boldsymbol{l}_0,\dots,\boldsymbol{l}_d))}{\mathbb{E}} \left[\mathsf{m}_\theta(\boldsymbol{l}_0,\dots,\boldsymbol{l}_d)\right] \cdot \prod_{i=0}^{d} \Pr(\mathrm{L}_i = l_i) \quad . \tag{9}$$

The sum to compute in Equation 9 contains $|\mathcal{L}|^{d+1}$ terms, where $|\mathcal{L}|$ denotes the leakage space of one share. In the case where the leakage model is highly non-injective such as with Hamming weights — *i.e.* $|\mathcal{L}| = n + 1$ —, computing the latter sum turns out to be much more efficient. For this model, and assuming that the shares are uniformly distributed, the marginal distribution $\Pr(\mathbf{L})$ is a joint distribution of d binomial laws $\mathcal{B}(n, 1/2)$.

References

[AAB$^+$15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 7

[ACHM22] Emmanuel Abbe, Elisabetta Cornacchia, Jan Hazla, and Christopher Marquis. An initial alignment between neural network and target is needed for gradient descent to learn. *CoRR*, abs/2202.12846, 2022. 18

[AKem$^+$21] Emmanuel Abbe, Pritish Kamath, eran malach, Colin Sandon, and Nathan Srebro. On the power of differentiable learning versus PAC and SQ learning. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. 18

[AS20] Emmanuel Abbe and Colin Sandon. On the universality of deep learning. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. 18, 19

[BDMS22] Olivier Bronchain, François Durvaux, Loïc Masure, and François-Xavier Standaert. Efficient profiled side-channel analysis of masked implementations, extended. *IEEE Trans. Inf. Forensics Secur.*, 17:574–584, 2022. 4

[BFG15] Josep Balasch, Sebastian Faust, and Benedikt Gierlichs. Inner product masking revisited. In Oswald and Fischlin [OF15], pages 486–510. 6

[BFG$^+$17] Josep Balasch, Sebastian Faust, Benedikt Gierlichs, Clara Paglialonga, and François-Xavier Standaert. Consolidating inner product masking. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 724–754, Hong Kong, China, December 3–7, 2017. Springer, Heidelberg, Germany. 6

[BL12] Timo Bartkewitz and Kerstin Lemke-Rust. Efficient template attacks based on probabilistic multi-class support vector machines. In Stefan Mangard, editor, *Smart Card Research and Advanced Applications - 11th International Conference, CARDIS 2012, Graz, Austria, November 28-30, 2012, Revised Selected Papers*, volume 7771 of *Lecture Notes in Computer Science*, pages 263–276. Springer, 2012. 1

[BPRS17] Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *J. Mach. Learn. Res.*, 18:153:1–153:43, 2017. 6

[BPS$^+$20] Ryad Benadjila, Emmanuel Prouff, Rémi Strullu, Eleonora Cagli, and Cécile Dumas. Deep learning for side-channel analysis and introduction to ASCAD database. *Journal of Cryptographic Engineering*, 10(2):163–188, June 2020. 13, 16

[BRST21] Joan Bruna, Oded Regev, Min Jae Song, and Yi Tang. Continuous LWE. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 694–707. ACM, 2021. 19

[BS20] Olivier Bronchain and François-Xavier Standaert. Side-channel countermeasures' dissection. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(2):1–25, 2020. https://tches.iacr.org/index.php/TCHES/article/view/8542. 2, 4, 10

[BS21] Olivier Bronchain and François-Xavier Standaert. Breaking masked implementations with many shares on 32-bit software platforms. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(3):202–234, 2021. https://tches.iacr.org/index.php/TCHES/article/view/8973. 6

[CDP17] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. Convolutional neural networks with data augmentation against jitter-based countermeasures - profiling attacks without pre-processing. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems – CHES 2017*, volume 10529 of *Lecture Notes in Computer Science*, pages 45–68, Taipei, Taiwan, September 25–28, 2017. Springer, Heidelberg, Germany. 1, 17

[CG00] Jean-Sébastien Coron and Louis Goubin. On Boolean and arithmetic masking against differential power analysis. In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2000*, volume 1965 of *Lecture Notes in Computer Science*, pages 231–237, Worcester, Massachusetts, USA, August 17–18, 2000. Springer, Heidelberg, Germany. 6

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412, Santa Barbara, CA, USA, August 15–19, 1999. Springer, Heidelberg, Germany. 6

[CLM20] Valence Cristiani, Maxime Lecomte, and Philippe Maurine. Leakage assessment through neural estimation of the mutual information. In Jianying Zhou, Mauro Conti, Chuadhry Mujeeb Ahmed, Man Ho Au, Lejla Batina, Zhou Li, Jingqiang Lin, Eleonora Losiouk, Bo Luo, Suryadipta Majumdar, Weizhi Meng, Martín Ochoa, Stjepan Picek, Georgios Portokalidis, Cong

Wang, and Kehuan Zhang, editors, *Applied Cryptography and Network Security Workshops - ACNS 2020 Satellite Workshops, AIBlock, AIHWS, AIoTS, Cloud S&P, SCI, SecMT, and SiMLA, Rome, Italy, October 19-22, 2020, Proceedings*, volume 12418 of *Lecture Notes in Computer Science*, pages 144–162. Springer, 2020. 11, 14, 16

[CLM21] Valence Cristiani, Maxime Lecomte, and Philippe Maurine. Revisiting mutual information analysis: Multidimensionality, neural estimation and optimality proofs. *IACR Cryptol. ePrint Arch.*, page 1518, 2021. 16

[CRR03] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28, Redwood Shores, CA, USA, August 13–15, 2003. Springer, Heidelberg, Germany. 1

[DDF14] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 423–440, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany. 2

[DFS15] Alexandre Duc, Sebastian Faust, and François-Xavier Standaert. Making masking security proofs concrete - or how to evaluate the security of any leaking device. In Oswald and Fischlin [OF15], pages 401–429. 2

[DS16] François Durvaux and François-Xavier Standaert. From improved leakage detection to the detection of points of interests in leakage traces. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 240–262, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany. 5

[EST$^+$22] Maximilian Egger, Thomas Schamberger, Lars Tebelmann, Florian Lippert, and Georg Sigl. A second look at the ASCAD databases. In Josep Balasch and Colin O'Flynn, editors, *Constructive Side-Channel Analysis and Secure Design - 13th International Workshop, COSADE 2022, Leuven, Belgium, April 11-12, 2022, Proceedings*, volume 13211 of *Lecture Notes in Computer Science*, pages 75–99. Springer, 2022. 5, 14

[FMPR11] Guillaume Fumaroli, Ange Martinelli, Emmanuel Prouff, and Matthieu Rivain. Affine masking against higher-order side channel analysis. In Alex Biryukov, Guang Gong, and Douglas R. Stinson, editors, *SAC 2010: 17th Annual International Workshop on Selected Areas in Cryptography*, volume 6544 of *Lecture Notes in Computer Science*, pages 262–280, Waterloo, Ontario, Canada, August 12–13, 2011. Springer, Heidelberg, Germany. 6, 11

[GHO15] Richard Gilmore, Neil Hanley, and Máire O'Neill. Neural network based attack on a masked implementation of AES. In *IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2015, Washington, DC, USA, 5-7 May, 2015*, pages 106–111, 2015. 1

[GP99] Louis Goubin and Jacques Patarin. DES and differential power analysis (the "duplication" method). In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES'99*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172, Worcester, Massachusetts, USA, August 12–13, 1999. Springer, Heidelberg, Germany. 6

[GRS08] Henri Gilbert, Matthew J. B. Robshaw, and Yannick Seurin. How to encrypt with the LPN problem. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP 2008: 35th International Colloquium on Automata, Languages and Programming, Part II*, volume 5126 of *Lecture Notes in Computer Science*, pages 679–690, Reykjavik, Iceland, July 7–11, 2008. Springer, Heidelberg, Germany. 19

[GVV22] Aparna Gupte, Neekon Vafa, and Vinod Vaikuntanathan. Continuous LWE is as hard as LWE & applications to learning Gaussian mixtures. Cryptology ePrint Archive, Report 2022/437, 2022. https://ia.cr/2022/437. 19

[HGD$^+$11] Gabriel Hospodar, Benedikt Gierlichs, Elke De Mulder, Ingrid Verbauwhede, and Joos Vandewalle. Machine learning in side-channel analysis: a first study. *Journal of Cryptographic Engineering*, 1(4):293–302, December 2011. 1

[HTF09] Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd Edition.* Springer Series in Statistics. Springer, 2009. 1

[HZ12] Annelie Heuser and Michael Zohner. Intelligent machine homicide - breaking cryptographic devices using support vector machines. In Werner Schindler and Sorin A. Huss, editors, *COSADE 2012: 3rd International Workshop on Constructive Side-Channel Analysis and Secure Design*, volume 7275 of *Lecture Notes in Computer Science*, pages 249–264, Darmstadt, Germany, May 3–4, 2012. Springer, Heidelberg, Germany. 1

[KB15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 9

[KPH$^+$19] Jaehun Kim, Stjepan Picek, Annelie Heuser, Shivam Bhasin, and Alan Hanjalic. Make some noise: Unleashing the power of convolutional neural networks for profiled side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(3):148–179, 2019. https://tches.iacr.org/index.php/TCHES/article/view/8292. 1, 2, 16, 17

[LBM14] Liran Lerman, Gianluca Bontempi, and Olivier Markowitch. Power analysis attack: an approach based on machine learning. *IJACT*, 3(2):97–115, 2014. 1

[LBM15] Liran Lerman, Gianluca Bontempi, and Olivier Markowitch. A machine learning approach against a masked AES - reaching the limit of side-channel attacks with a learning model. *Journal of Cryptographic Engineering*, 5(2):123–139, June 2015. 1

[LLM$^+$18] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 9628–9639, 2018. 11

[LPR$^+$14] Victor Lomné, Emmanuel Prouff, Matthieu Rivain, Thomas Roche, and Adrian Thillard. How to estimate the success rate of higher-order side-channel attacks. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*, volume 8731 of *Lecture Notes in Computer Science*, pages 35–54. Springer, 2014. 3

[LZC$^+$21] Xiangjun Lu, Chi Zhang, Pei Cao, Dawu Gu, and Haining Lu. Pay attention to raw traces: A deep learning architecture for end-to-end profiling attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(3):235–274, 2021. https://tches.iacr.org/index.php/TCHES/article/view/8974. 17

[MBC$^+$20] Loïc Masure, Nicolas Belleville, Eleonora Cagli, Marie-Angela Cornelie, Damien Couroussé, Cécile Dumas, and Laurent Maingault. Deep learning side-channel analysis on large-scale traces - A case study on a polymorphic AES. In Liqun Chen, Ninghui Li, Kaitai Liang, and Steve A. Schneider, editors, *ESORICS 2020: 25th European Symposium on Research in Computer Security, Part I*, volume 12308 of *Lecture Notes in Computer Science*, pages 440–460, Guildford, UK, September 14–18, 2020. Springer, Heidelberg, Germany. 17

[MDM16] Zdenek Martinasek, Petr Dzurenda, and Lukas Malina. Profiling power analysis attack based on MLP in DPA contest V4.2. In *39th International Conference on Telecommunications and Signal Processing, TSP 2016, Vienna, Austria, June 27-29, 2016*, pages 223–226, 2016. 1

[MDP19a] Loïc Masure, Cécile Dumas, and Emmanuel Prouff. A comprehensive study of deep learning for side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(1):348–375, 2019. https://tches.iacr.org/index.php/TCHES/article/view/8402. 1, 8, 17

[MDP19b] Loïc Masure, Cécile Dumas, and Emmanuel Prouff. Gradient visualization for general characterization in profiling attacks. In Ilia Polian and Marc Stöttinger, editors, *COSADE 2019: 10th International Workshop on Constructive Side-Channel Analysis and Secure Design*, volume 11421 of *Lecture Notes in Computer Science*, pages 145–167, Darmstadt, Germany, April 3–5, 2019. Springer, Heidelberg, Germany. 16

[MPP16] Houssem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. Breaking cryptographic implementations using deep learning techniques. In Claude Carlet, M. Anwar Hasan, and Vishal Saraswat, editors, *Security, Privacy, and Applied Cryptography Engineering - 6th International Conference, SPACE 2016, Hyderabad, India, December 14-18, 2016, Proceedings*, volume 10076 of *Lecture Notes in Computer Science*, pages 3–26. Springer, 2016. 1

[MS21] Loïc Masure and Rémi Strullu. Side channel analysis against the ANSSI's protected AES implementation on ARM. Cryptology ePrint Archive, Report 2021/592, 2021. https://eprint.iacr.org/2021/592. 1, 5, 11, 15

[MZ13] Zdenek Martinasek and Vaclav Zeman. Innovative method of the power analysis. *Radioengineering*, 22:586–594, 06 2013. 1

[OF15] Elisabeth Oswald and Marc Fischlin, editors. *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany. 21, 23

[OGGM21] Maamar Ouladj, Sylvain Guilley, Philippe Guillot, and Farid Mokrane. Spectral approach to process the (multivariate) high-order template attack against any masking scheme. Cryptology ePrint Archive, Report 2021/941, 2021. https://eprint.iacr.org/2021/941. 5

[PGM$^+$19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 8024–8035, 2019. 7

[PHG17] Stjepan Picek, Annelie Heuser, and Sylvain Guilley. Template attack versus bayes classifier. *Journal of Cryptographic Engineering*, 7(4):343–351, November 2017. 1

[Pie12] Krzysztof Pietrzak. Cryptography from learning parity with noise. In Mária Bieliková, Gerhard Friedrich, Georg Gottlob, Stefan Katzenbeisser, and György Turán, editors, *SOFSEM 2012: Theory and Practice of Computer Science*, pages 99–114, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. 19

[PP20] Guilherme Perin and Stjepan Picek. On the influence of optimizers in deep learning-based side-channel analysis. In Orr Dunkelman, Michael J. Jacobson Jr., and Colin O'Flynn, editors, *Selected Areas in Cryptography - SAC 2020 - 27th International Conference, Halifax, NS, Canada (Virtual Event), October 21-23, 2020, Revised Selected Papers*, volume 12804 of *Lecture Notes in Computer Science*, pages 615–636. Springer, 2020. 9, 16

[RWPP21] Jorai Rijsdijk, Lichao Wu, Guilherme Perin, and Stjepan Picek. Reinforcement learning for hyperparameter tuning in deep learning-based side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(3):677–707, 2021. https://tches.iacr.org/index.php/TCHES/article/view/8989. 16

[SB14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning - From Theory to Algorithms.* Cambridge University Press, 2014. 16

[SOGISS20] Senior Officials Group Information Systems Security. Application of attack potential to smartcards and similar devices — joint interpretation library, 2020. https://www.sogis.eu/documents/cc/domains/sc/JIL-Application-of-Attack-Potential-to-Smartcards-v3-1.pdf. 2

[SSS17] Shai Shalev-Shwartz, Ohad Shamir, and Shaked Shammah. Failures of gradient-based deep learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 3067–3075. PMLR, 2017. 18, 19

[SZ15] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 7

[TGD$^+$18] Anna T. Thomas, Albert Gu, Tri Dao, Atri Rudra, and Christopher Ré. Learning compressed transforms with low displacement rank. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 9066–9078, 2018. 7

[Tho96] Chris Thornton. Parity: The problem that won't go away. In Gordon I. McCalla, editor, *Advances in Artificial Intelligence, 11th Biennial Conference of the Canadian Society for Computational Studies of Intelligence, AI '96, Toronto, Ontario, Canada, May 21-24, 1996, Proceedings*, volume 1081 of *Lecture Notes in Computer Science*, pages 362–374. Springer, 1996. 19

[Tim19] Benjamin Timon. Non-profiled deep learning-based side-channel attacks with sensitivity analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(2):107–131, 2019. https://tches.iacr.org/index.php/TCHES/article/view/7387. 1, 16

[TWO14] Michael Tunstall, Carolyn Whitnall, and Elisabeth Oswald. Masking tables - an underestimated security risk. In Shiho Moriai, editor, *Fast Software Encryption – FSE 2013*, volume 8424 of *Lecture Notes in Computer Science*, pages 425–444, Singapore, March 11–13, 2014. Springer, Heidelberg, Germany. 11

[VJM$^+$15] Nicolas Vasilache, Jeff Johnson, Michaël Mathieu, Soumith Chintala, Serkan Piantino, and Yann LeCun. Fast convolutional nets with fbfft: A GPU performance evaluation. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 7

[von01] Manfred von Willich. A technique with an information-theoretic basis for protecting secret data from differential power attacks. In Bahram Honary, editor, *8th IMA International Conference on Cryptography and Coding*, volume 2260 of *Lecture Notes in Computer Science*, pages 44–62, Cirencester, UK, December 17–19, 2001. Springer, Heidelberg, Germany. 6

[WPP20] Lichao Wu, Guilherme Perin, and Stjepan Picek. I choose you: Automated hyperparameter tuning for deep learning-based side-channel analysis. Cryptology ePrint Archive, Report 2020/1293, 2020. https://eprint.iacr.org/2020/1293. 16

[Yar17] Dmitry Yarotsky. Error bounds for approximations with deep relu networks. *Neural Networks*, 94:103–114, 2017. 8

[ZBHV19] Gabriel Zaid, Lilian Bossuet, Amaury Habrard, and Alexandre Venelli. Methodology for efficient CNN architectures in profiling attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(1):1–36, 2019. https://tches.iacr.org/index.php/TCHES/article/view/8391. 2