IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2021, No. 3, pp. 552–598. DOI:10.46586/tches.v2021.i3.552-598

DL-LA: Deep Learning Leakage Assessment

A modern roadmap for SCA evaluations

Thorben Moos*, Felix Wegener* and Amir Moradi

Ruhr University Bochum, Horst Görtz Institute for IT Security, Bochum, Germany
firstname.lastname@rub.de

Abstract. In recent years, deep learning has become an attractive ingredient to side-channel analysis (SCA) due to its potential to improve the success probability or enhance the performance of certain frequently executed tasks. One task that is commonly assisted by machine learning techniques is the profiling of a device's leakage behavior in order to carry out a template attack. At CHES 2019, deep learning has also been applied to non-profiled scenarios for the first time, extending its reach within SCA beyond template attacks. The proposed method, called DDLA, has some tempting advantages over traditional SCA due to merits inherited from (convolutional) neural networks. Most notably, it greatly reduces the need for pre-processing steps when the SCA traces are misaligned or when the leakage is of a multivariate nature. However, similar to traditional attack scenarios the success of this approach highly depends on the correct choice of a leakage model and the intermediate value to target. In this work we explore, for the first time in literature, whether deep learning can similarly be used as an instrument to advance another crucial (non-profiled) discipline of SCA which is inherently independent of leakage models and targeted intermediates, namely leakage assessment. In fact, given the simple classification-based nature of common leakage assessment techniques, in particular distinguishing two groups fixed-vs-random or fixed-vs-fixed, it comes as a surprise that machine learning has not been brought into this context, yet. Our contribution is the development of the first full leakage assessment methodology based on deep learning. It gives the evaluator the freedom to not worry about location, alignment and statistical order of the leakages and easily covers multivariate and horizontal patterns as well. We test our approach against a number of case studies based on FPGA, ASIC and µC implementations of the PRESENT block cipher, equipped with state-of-the-art SCA countermeasures. Our results clearly show that the proposed methodology and network structures are robust across all case studies and outperform the classical detection approaches (t-test and χ^2-test) in all considered scenarios.

Keywords: Leakage Evaluation · Side-Channel Analysis · Deep Learning

1 Introduction

In an ideal world, side-channel security evaluations would be able to provide a qualitative and confident answer (pass or fail) to the question whether the device under test (DUT) is vulnerable to physical attacks or not. However, the past has shown that this expectation is indeed a utopia. An exhaustive verification of the security of a DUT against all possible attack vectors is simply infeasible. Instead, the concept of leakage assessment has been introduced in order to answer a slightly, but explicitly, less informative question. Namely, whether any kind of input-dependent information can be detected in side-channel measurements of the device under test. Clearly, in case this question is answered positively,

*These authors contributed equally to this work.

Received: 2020-10-15 Accepted: 2020-12-15 Published: 2021-07-09

no conclusions about the actual vulnerability of the device with respect to key recovery attacks can be drawn (although it is sometimes interpreted as an indication thereof). Yet, in case it is answered negatively (and no *false* negative occurs) the DUT should be sufficiently secure. In other words, leakage assessment is conceptually capable of providing the initially desired confidence in at least one of the two cases. This possibility inspired the quest for appropriate leakage assessment methods in academia and industry.

The most prominent leakage detection approach is certainly distinguishing two groups of measurements, one for fixed and one for random inputs, by means of the Welch's t-test [GJJR11, SM15]. Whenever these two groups are distinguishable with confidence one can conclude that the device reveals input-dependent information. However, this method has some severe limitations, especially when more sophisticated types of side-channel leakage need to be captured. First of all, since each point in time is evaluated independently, the approach inherently expects that any potential side-channel leakage is of a univariate nature and, more generally, that the detection of the leakage does not benefit from a combination of multiple points. Yet, many counterexamples to this assumption can be observed in reality. Although Schneider *et al.* [SM15] provide detailed information on how to perform the t-test at arbitrary order and variate, the performance of the test at higher variates either quickly runs into feasibility issues or its success depends highly on the expertise of the evaluator and the prior knowledge about the underlying implementation. On another note, a misalignment of the leaking samples between the individual traces leads to a significantly impaired detection as well. Thus, the Welch's t-test, as it is currently applied as a test vector leakage assessment (TVLA) methodology, is naturally unsuited to cover multivariate and horizontal leakages, as well as (heavily) misaligned traces. In addition to that, it has recently been pointed out that the separation of statistical orders, which is often seen as a beneficial feature of the t-test when seeking the smallest key dependent moment for example, may cause false negatives. This can be observed when masked implementations with (very) low noise levels are analyzed [Sta18, Moo19] or when the leakage is distributed over multiple statistical moments (as it is common for hardware masking schemes like Threshold Implementations) [MRSS18, Sta18]. Moradi *et al.* [MRSS18] suggested Pearson's χ^2-test as a natural complement to the Welch's t-test to aggregate leakages distributed over multiple orders and to analyze the joint information. By combining the two approaches the risk of false negatives, especially in the previously described cases, can significantly be reduced. Yet, in the same manner as the t-test, the χ^2-test analyzes the individual points in a leakage trace independently and therefore suffers from the same shortcomings when it comes to multivariate or horizontal patterns and misalignments. The core motivation for this work has been to extend the state of the art in such a way that the latter types of leakage can be covered, using a simple and easy to apply methodology.

Deep learning has been brought into the context of side-channel analysis mainly in order to improve the effectiveness of template attacks [HGM+11]. In a template attack the adversary is in possession of a fully-controlled profiling device, learns the leakage function of a certain cryptographic operation and subsequently uses the acquired knowledge to reveal sensitive information on a structurally identical but distinct target device where the secrets are unknown. Apart from the general suitability of deep learning to build classifiers for profiled side-channel attacks, it has also been demonstrated that certain features and structures of the applied neural networks offer valuable advantages over classical template attacks. For example, it has been shown in [CDP17] that convolutional neural networks (CNNs) can lead to efficient classifiers even when the available side-channel traces suffer from a misalignment. Thus, due to their so-called translation invariance property, CNNs can be utilized to conquer jitter-based countermeasures. Recently, the first non-profiled deep learning based side-channel attacks have been demonstrated in literature [Tim19]. The proposed method, called DDLA, is based on guessing a part of the key, using it to

compute the targeted key-dependent intermediate value, applying a leakage model and labeling the training data according to its result. Assuming that under the correct key hypothesis the differences between the classes implied by the leakage model correlate with the measured leakage traces (and for the incorrect guesses they do not), the impact of the correct key guess on the training loss and the training accuracy is visible and can easily be identified. Although, this approach depends on the correct choice of the targeted intermediate value and the applied leakage model as much as traditional attacks, it offers some tempting advantages. First of all, in case CNNs are used, the translation invariance property allows to analyze misaligned traces without any pre-processing. Secondly, when the leakage is of a multivariate nature or generally distributed over multiple points no recombination and no prior knowledge about the underlying implementation is required. Hence, deep learning is a powerful tool for non-profiled scenarios as well.

1.1 Our Contribution

For the first time in literature we evaluate whether deep learning is an eligible strategy for black box leakage detection. To this end, we have developed an approach that is based on the concept of supervised learning. We call it deep learning leakage assessment (DL-LA) in the following. Simply put, we train a neural network with a randomly interleaved sequence of labeled side-channel measurements that have been acquired while supplying the DUT with one of two distinct fixed inputs (fixed-vs-fixed). Afterwards, in the validation phase, the trained network is supplied with unlabeled measurements from both groups and supposed to correctly classify them. Of course, the training set and the validation set are disjoint. In case the network succeeds with a higher percentage of correct classifications than could be achieved by a randomly guessing binary classifier with a non-negligible probability, it can be concluded that indeed enough information was included in the *training* set to distinguish the two groups. In other words, given the percentage of correctly classified traces and the size of the validation set one can easily calculate a confidence value, i.e., a probability, that the correct classifications were not just a random statistical occurrence. In this way it is possible to directly compare the confidence values achieved by DL-LA with the confidence provided by classical leakage assessment approaches, such as the Welch's t-test and Pearson's χ^2-test. Aligning the appearance of DL-LA results to previous approaches is a valuable characteristic of our methodology and a contribution of this work.

Classical hypothesis tests rely on clearly defined formulas to estimate their statistics. DL-LA on the other hand grants a high level of freedom regarding its application due to the flexible choice of the network that shall be trained as a classifier. However, that freedom does not come without drawbacks. A complex network with a highly successful performance as a classifier on a certain set of measurements may not deliver satisfactory results on another set. Hence, the selection of the network may have a huge impact on the success of the DL-LA procedure. In such a case, an evaluator may be required to repeat the evaluation using many different networks to gain a confident result or choose suitable parameters based on prior knowledge about the underlying implementation. Clearly, both scenarios are undesirable for an initial analysis. Thus, in order to qualify as a simple and generic strategy for leakage assessment, it should be possible to select networks which offer a fairly robust and universal performance. In particular, it is desirable that the leakage detection capability of a network is as independent of the type of side-channel leakage to be detected as possible and largely independent of exterior parameters such as the trace length as well. This approach stands in stark contrast to common applications of deep learning in the area of physical security evaluations. Usually a network is deliberately tailored to a specific leakage pattern and measurement set in order to provide the best possible results. In our case, however, we want to use networks, at least for the initial

analysis, that are general enough to deliver appropriate results when facing many different measurement sets and leakage behaviors. Thus, an important contribution of this work is the identification of network structures that perform consistently well when faced with different types of side-channel leakage and characteristics of the traces. After a long process of evaluating different networks (manually and automated) we have come to the conclusion that simplicity beats complexity. We evaluate and recommend two different simple network architectures, 1) a multi-layer perceptron (MLP) and 2) a convolutional neural network (CNN), the first with a given set of hyper-parameters and the second with a spectrum of hyper-parameters that proved successful. We are able to show that both networks provide excellent detection performance in a total of nine different case studies analyzing three distinct implementation platforms (FPGA, custom ASIC, and ARM Cortex-M0 µC). Each of the case studies features implementations of the PRESENT ultra-lightweight block cipher with different variations of masking and hiding countermeasures applied. The classification capability of our networks does not only withstand misaligned and noisy traces, but is able to deal with univariate and multivariate higher-order leakage as well. In all nine case studies we compare the success of our method to both the Welch's t-test and Pearson's χ^2-test and show that DL-LA outperforms the leakage assessment capabilities of the classical techniques in all considered scenarios (either by requiring fewer traces for confident detection or by providing a higher confidence using the same number of traces for detection[1]). We also present one scenario where both the univariate and the multivariate versions of the t-test and the χ^2-test fail to detect leaked information with confidence, while DL-LA still succeeds with only half of the available traces. As an unintended byproduct of our practical case studies, we provide the most detailed practical comparison between the Welch's t-test and Pearson's χ^2-test that has been reported in the literature so far.

The most outstanding advantage of our approach is clearly that the underlying network is free to combine as many points for the classification of the two groups as necessary. Thus, even in complex scenarios of purely multivariate or horizontal leakages, the traces can simply be fed as training data into the network without any trace-specific pre-processing or manual selection of points. Accordingly, neither a high expertise is demanded from the evaluator, nor is it required to obtain any prior information about the underlying implementation or the type of leakage that is expected. We believe that proving distinguishability of two sets of data by actually building a successful distinguisher is an elegant solution and more intuitive than the usual statistical arguments. We also compare our neural-network-based classifiers to multivariate Gaussian models which theoretically could capture distributed leakages as well. However, based on our experiments we conclude that such template-analysis methods can not provide the same level of flexibility as machine learning approaches. Compared to the common univariate distinction tests, DL-LA generally entails a lower risk of false positives as it provides a single confidence value to assess the distinguishability of the groups. Traditional point-wise methods would need to normalize their confidence values to the number of points in the traces to provide a meaningful confidence threshold. However, this inaccuracy is mostly disregarded in their respective methodologies. Even though, DL-LA provides only a single confidence value, the approach can still identify the points of interest in side-channel traces that contain leakage, by performing a Sensitivity Analysis (SA) on the trained network. Obviously, the average computation time to perform a DL-LA is significantly higher when compared to simple univariate tests (a comparison is provided in Section 4). However, as soon as more complex types of side-channel leakage need to be analyzed, the additional run time quickly pays off, since the effort that otherwise has to be spent in order to make traditional methods recognize those complex patterns (if possible) grows even bigger and contains several steps that are hard to automate.

[1]We explain in Section 3 that in the DL-LA procedure there is a distinction between the required number of traces for the detection (i.e., the training set) and the required number of traces to carry out the evaluation (i.e., the sum of the training set and the validation set).

1.2 Claims and Non-Claims

In order to avoid any potential confusion regarding our claims, or lack thereof, we explicitly list the most important statements below:

We *do not* claim that ...

- ✗ our chosen networks are optimal for leakage detection in general or for any of the considered case studies in particular. We are certain that there is room for improvement, especially when considering individual cases, as we intentionally optimized for robustness and simplicity instead of single case performance.
- ✗ our chosen networks necessarily lead to classifiers that outperform the t-test or the χ^2-test for any given side-channel traces.
- ✗ DL-LA is generally superior and should replace established leakage detection techniques.
- ✗ DL-LA generally causes none or fewer false negatives than the classical approaches.

We *do* claim that ...

- ✓ the chosen networks offer some basic universality and robustness. Their success in all nine case studies featuring three different implementation platforms (FPGA, ASIC, µC) is practical evidence for this claim.
- ✓ the chosen networks are able to learn first-order, higher-order, univariate, multivariate and horizontal leakages without requiring any trace-specific pre-processing or prior knowledge about the underlying implementation.
- ✓ DL-LA entails a much lower risk of false positives (if the same confidence threshold is chosen) since it provides one confidence per set of traces instead of one confidence per time sample in the trace set.

2 Background

In this section we introduce the necessary background with respect to the roots and state-of-the-art of leakage assessment, as well as deep learning and its applications to side-channel analysis.

2.1 Leakage Assessment

Ever since the introduction of side-channel attacks in 1999 [KJJ99] the standard approach for assessing the physical vulnerability of a device has been a more or less exhaustive verification of its resistance against known attacks while attempting to cover a broad range of intermediate values and hypothetical leakage models. This approach, however, became less feasible over the years due to the increasing amount of new attack methods and the higher complexity of potential leakage models due to the introduction of countermeasures against physical attacks. Another concern regarding this procedure is that it entails a significant risk of reporting physical security in favor of the DUT while in reality merely a certain attack vector was missed in the process (by mistake or because it was unknown at time of evaluation) that could indeed enable key recovery [SM15]. Hence, the need for a robust and reliable standard leakage assessment method independent of concrete attack scenarios, targeted intermediates and hypothetical leakage models grew consistently over the years. In an attempt to gather and evaluate promising candidates, the National

Institute of Standards and Technology (NIST) hosted a "Non-Invasive Attack Testing Workshop" in 2011. One of the most intriguing proposals at the workshop was the use of the non-specific Welch's t-test [GJJR11] for leakage detection. Leakage detection avoids any dependency on the choice of intermediates and leakage models by focusing on the detection of leakage only, without paying any attention to the possibility to exploit said leakage for key recovery. Simply put, the concept is based on supplying the device under test with different inputs, recording its leakage behavior and evaluating whether a difference can be observed. Thus, such a method is suitable for black box scenarios and allows certification of a device's physical security by third party evaluation labs without the need to test a multitude of different methods and parameter combinations. Seven years later, after some shortcomings of the moment-based nature of the t-test had been identified [Sta18], another popular statistical hypothesis test was proposed for leakage detection purposes, namely the Pearson's χ^2-test [MRSS18]. Both hypothesis tests, the t-test and the χ^2-test, are applied in the field of statistics in order to answer the question whether two sets of data are significantly different from each other. To be more precise, the evaluation of the tests examines the validity of the null hypothesis, which constitutes that both sets of data were drawn from the same population (i.e., they are indistinguishable) [SM15]. In side-channel analysis contexts, it is usually evaluated whether two groups of measurements can be distinguished with confidence. Traditionally, those two groups are acquired by supplying the DUT either with random (group Q_0) or a fixed input (group Q_1), selected by coin toss. Later, it has been demonstrated that the careful choice of two distinct fixed inputs (instead of maintaining one group for random inputs) usually leads to a lower data complexity for the distinction [DS16]. We provide the details on how to conduct the Welch's t-test and Pearson's χ^2-test below.

Welch's *t*-test. We denote two sets of data by Q_0 and Q_1, their cardinality by n_0 and n_1, their respective means by μ_0 and μ_1 and their standard deviations by s_0 and s_1. The t-statistics and the degrees of freedom v can then be computed using the following formulas.

$$t = \frac{\mu_0 - \mu_1}{\sqrt{\frac{s_0^2}{n_0} + \frac{s_1^2}{n_1}}} \qquad\qquad v = \frac{\left(\frac{s_0^2}{n_0} + \frac{s_1^2}{n_1}\right)^2}{\frac{\left(\frac{s_0^2}{n_0}\right)^2}{n_0 - 1} + \frac{\left(\frac{s_1^2}{n_1}\right)^2}{n_1 - 1}}$$

Afterwards, the confidence p to accept the null hypothesis can be estimated via the Student's t probability density function, where $\Gamma(.)$ denotes the gamma function [SM15, MRSS18].

$$p = 2\int_{|t|}^{\infty} f(t, v)dt \qquad\qquad f(t, v) = \frac{\Gamma\left(\frac{v+1}{2}\right)}{\sqrt{\pi v}\,\Gamma\left(\frac{v}{2}\right)}\left(1 + \frac{t^2}{v}\right)^{-\frac{v+1}{2}}$$

In practice, for the sake of simplicity, it is common to only evaluate the t-statistics and to set the confidence threshold for distinguishability to $|t| > 4.5$. The statistical background of this threshold is that for $|t| > 4.5$ and $v > 1000$ the confidence p to accept the null hypothesis is smaller than 0.00001 which is equivalent to a 99.999 % confidence that the two sets were *not* drawn from the same population. Of course, when the degrees of freedom v are not explicitly evaluated, it can occur that the assumption $v > 1000$ does not hold. However, practice has shown that this procedure rarely produces false positive results in side-channel analysis contexts. Yet, calculating the actual confidence p is certainly preferable, scientifically correct and can still be efficiently implemented [MRSS18]. Since the Welch's t-test is designed to distinguish the means of two distributions, it can only be applied to first-order univariate analyses in its simplest form. Schneider *et al.* [SM15] extended the methodology to arbitrary orders and variates and provide the required formulas for incremental one-pass computation of all moments.

Pearson's χ^2-test. In order to mitigate some of the limitations and shortcomings of the moment-based nature of the Welch's t-test, in particular for higher-order analyses of masked implementations, Moradi *et al.* [MRSS18] suggested the Pearson's χ^2-test. In contrast to the t-test this hypothesis test analyzes the full distributions and can capture information that lies in multiple statistical moments. Thus, it prevents false negatives when moment-based analyses become suboptimal [MRSS18].
In a first step a contingency table F has to be constructed from the two sets Q_0 and Q_1 (basically two histograms). We denote the number of rows by r (= 2, when two sets are compared) and the number of columns by c (number of bins of the histograms). The χ^2-statistics x and the degrees of freedom v can then be computed using the following formulas.

$$\mathrm{x} = \sum_{i=0}^{r-1} \sum_{j=0}^{c-1} \frac{(F_{i,j} - E_{i,j})^2}{E_{i,j}} \qquad \mathrm{v} = (r-1) \cdot (c-1)$$

$E_{i,j}$ denotes the expected frequency for a given cell.

$$E_{i,j} = \frac{\left(\sum_{k=0}^{c-1} F_{i,k}\right) \cdot \left(\sum_{k=0}^{r-1} F_{k,j}\right)}{N}$$

Finally, the confidence p to accept the null hypothesis is estimated via the χ^2 probability density function, where $\Gamma(.)$ denotes the gamma function [MRSS18].

$$p = \int_{\mathrm{x}}^{\infty} f(x, \mathrm{v}) dx \qquad f(x, \mathrm{v}) = \begin{cases} \frac{x^{\frac{\mathrm{v}}{2}-1} e^{-\frac{x}{2}}}{2^{\frac{\mathrm{v}}{2}} \Gamma\left(\frac{\mathrm{v}}{2}\right)} & x > 0 \\ 0 & \text{otherwise} \end{cases}$$

In contrast to the t-test this procedure can easily be extended to more than two sets of data ($r > 2$), which can be a valuable feature when used as distinguisher for key recovery attacks. Generally, it can be said that in cases where the χ^2-test provides a higher confidence to reject the null hypothesis than the t-test (on the same side-channel data), the analysis of the leakages requires some special attention. This is usually the case when masked implementations with low noise levels are analyzed [Sta18, Moo19] or when hardware-masking schemes like threshold implementations cause leakages in multiple moments due to physical defaults such as glitches [Sta18, MRSS18].

2.2 Deep Learning

We give a brief summary of the history and applications of deep learning and subsequently introduce definitions and explain the underlying principle.

History and Applications. Historically, the field of machine learning dealt with extracting meaningful information from data by applying relatively simple mathematical models, e.g., Bayes Classifiers, Support Vector Machines or Decision Trees to a sanitized version of the input data. This required manual and time-consuming *feature engineering* to predetermine which elements might be useful in a given set of raw data and how to best represent them, e.g., Canny edge detection as a first hard-coded step for image classification.
In contrast, deep learning methods are generally capable of learning from raw input data, thereby making the elaborate modeling process unnecessary. Since the breakthrough improvement of classification accuracy on the ImageNet data set in 2012 [KSH12], deep learning has been successfully applied to many diverse tasks such as speech recognition, drug discovery, natural language processing, visual art style transfer, image classification, autonomous driving and strategy games.

More recently, the side channel community discovered deep learning as a tool to perform profiled attacks [CDP17, HGM+11, MPP16, MDP19b] with competitive results compared to classical modeling techniques, e.g., based on a multivariate normal distribution. On the other hand, the run-time effectiveness of DL-based approaches over classical machine learning is sometimes questioned [PSK+18]. Apart from our present work, only few publications have investigated the use of deep learning for the non-profiled case, including [Tim19] and [PCBP21]. In the former article a method is introduced that exploits the correlation between a correct key guess and a steep learning rate to enable key recovery. The latter article introduces a novel framework based on unsupervised learning to improve horizontal attacks on (protected) implementations of public-key cryptosystems.

Principle and Definitions. In the following we limit ourselves to sequential neural networks (without recurrent elements) used for the purpose of classification. The aim of this description is to give brief definitions for the standard terms in deep learning, while the explanation of principles is intentionally very shallow.
A neural network is structured into multiple layers, each containing a matrix of learnable weights w that is linearly applied to its inputs x and a non-linear activation function applied to each coordinate of the result of this matrix multiplication. The output of this combined operation is taken as an input for the subsequent layer. Finally, the output layer of the neural network contains as many output coordinates as classes[2] (c) and uses softmax as an activation function

$$\text{softmax}(x_j) = \frac{e^{x_j}}{\sum_{i=1}^{c} e^{x_i}},$$

such that the sum over all outputs is always equal to one, thereby forming a probability distribution over the possible class labels.
Let us first assume the weights are initialized with some values before an evaluation of the network takes place by applying the function of the first layer to the input sample and subsequently propagating the computed values forward layer by layer until all layers have been evaluated. The prediction y' consists in the output coordinate of the final layer with the highest value.
In the beginning, the weights in a neural network are initialized with random values. To determine useful weights that achieve accurate prediction values, a *training phase* is necessary. First, the designer needs to define a metric to measure the distance between a prediction y' and the actual class label y. This metric is called a *loss function* which determines the *loss score*. To perform training, a data set with labeled inputs, i.e., a list of tuples (x, y), is separated into *batches* of a fixed size b. The neural network is evaluated simultaneously on all samples in a batch thereby producing loss scores. After each batch an optimization strategy based on *Backpropagation* is used to adjust all weights in the neural network dependent on the gradient of the loss function. Each iteration through the entire training set is called an *epoch*. To minimize the loss score, training over multiple epochs is performed in each of which the training data is randomly regrouped into new batches. For simplicity we assume that the training ends after a predetermined number of user-defined epochs.
To judge the quality of the classifier during and after training, the metrics *accuracy* and *validation accuracy* should be considered. While accuracy is related only to the training set, validation accuracy takes an entirely separate *validation set* into account, to ensure that the traits learned are actually generalizable opposed to rote learning of the specific training set (the latter phenomenon is called *overfitting*).
When choosing and training a deep learning model, the designer has to determine values for many so-called *hyper-parameters*[3], these include the depth of the network, the types and

[2] We limit ourselves to this variant called one-hot encoding.
[3] In distinction from the parameters, i.e., the concrete weights learned during training.

sizes of layers, their activation function, the loss function and the optimizer strategy. We provide the hyper-parameters we chose for both of our network architectures in Section 3.3.

3 DL-LA: Deep Learning Leakage Assessment

We introduce Deep Learning Leakage Assessment (DL-LA), a novel leakage assessment methodology based on deep learning. Our method is simple to apply and outperforms classical leakage detection approaches such as the Welch's t-test and the more recently proposed Pearson's χ^2-test in many cases due to its intrinsically multivariate nature. We specify two neural networks that showed excellent and robust detection performance in all our practical case studies. Further, we apply Sensitivity Analysis (SA), which has been used in other SCA contexts for leakage visualization, to leakage assessment and preempt several common pitfalls during adoption of DL-LA.

3.1 Core Idea of DL-LA

The aim of leakage assessment is to determine whether an attacker is able to extract information from side-channel measurements. The current state of the art for non-profiled adversary models is based on univariate statistical distinction tests (Welch's t-test, Pearson's χ^2-test) which are applied to two groups of side-channel measurements collected for two distinct fixed inputs processed by the target implementation (alternatively one group for random inputs and the other one for fixed).
DL-LA maintains the basic idea of distinguishing two groups of side-channel traces from each other (fixed-vs-fixed). Hence, from an evaluator's perspective the entire measurement setup and tool-chain can remain unchanged when adopting our methodology. We apply deep learning to the concept of leakage assessment by training a neural network to serve as a distinguisher between the two groups. This is done in a supervised-learning-based approach by applying labeled data from both groups to the network. The set of data applied to the network during this phase is then called the training set. Afterwards, the classification capabilities of the network are evaluated on a distinct validation set of labeled measurements without revealing the true labels to the network. The success rate of the classification on the validation set quantifies the amount of generalizable information that the neural network could extract from the *training* set during the *training* phase. In more detail, the network has learned generalizable features during the training when it can provide a better-than-random guess which of the two fixed inputs was processed to produce an individual validation trace. In case the classification of the whole validation set succeeds with a higher percentage than it could be achieved by a randomly guessing classifier with a non-negligible probability, it gives clear evidence for the fact that informative side-channel leakage is present. In this context we present a simple metric to determine an exact probability value p that quantifies the statistical confidence in the evidence.

3.1.1 Training vs. Validation Set

Please note that the number of required traces to detect the leakage is only related to the *training* set. The size of the validation set can be chosen completely independent and influences the result of the detection only if generalizable features (i.e., informative side-channel leakage) could be extracted from the *training* set. Otherwise the percentage of correct classifications will never be significantly different from 50%, no matter how large the validation set is. To be more precise, the information included in the validation set has no impact on the already trained classifier. It is merely an auxiliary data set required to test the quality of the classification ability.
Therefore, in DL-LA it has to be distinguished between the required number of traces for

the detection (i.e., the training set) and the required number of traces to carry out the evaluation (i.e., the sum of the training set and the validation set). Only the detection (i.e., the training) traces represent the number of measurements available to an attacker. In other words, DL-LA evaluates the likelihood that an attacker is able to extract informative side-channel leakage from the finite set of training traces. While the evaluator clearly requires both sets to perform the evaluation, reporting the combined cardinality of both sets as the number of traces for a successful detection would be misguided and could lead to an incorrect impression about the security of the target. As an example, we assume an evaluator wants to know with high confidence whether a set of 1 000 traces contains a sufficient amount of information to detect a difference between the two underlying groups of measurements. The typical procedure would be to train a neural network as a classifier on those 1 000 traces over as many epochs as desired (while being mindful of overfitting) and then validating the classifier on an arbitrarily-sized validation set. Even if the validation set must be 100 000 000 traces large to overcome the confidence threshold desired, it still means that only 1 000 traces were required to find and learn generalizable features in the traces that allow better-than-random classification of new and unseen measurements into the two groups. Simply speaking, this means that 1 000 traces already leak confidently detectable information. However, 100 001 000 traces would be required for the evaluation. Of course this is an extreme and unlikely corner case. Typically, much more reasonable trade-offs between training set and validation set size can be achieved, which is also demonstrated in the practical case studies we present in Section 4. Especially when examining well-protected implementations which require millions of traces for a meaningful analysis the size of the validation set is typically not the prohibitive element and is often significantly smaller than the training set. Yet, it is important to make the distinction between detection and evaluation traces, since minimizing the combined set, namely training + validation, is not trivial and not a focus of this work. In the example detailed above the evaluator could simply increase the size of training set in hopes of improving the trained classifier and require a smaller validation set (and likely a smaller combined set) for a confident detection. However, this would not properly answer the question whether 1 000 traces allow extraction of information. Therefore, we do not explore strategies to find the minimum combined cardinality of the two sets although this might be an interesting topic for future research in the area. We provide further discussion on the partition strategy into training set and validation set to decouple the number of traces available to the attacker from the statistical confidence the evaluator wants to obtain in Section 5.

3.1.2 Fixed-vs-Fixed or Fixed-vs-Random

Traditionally, leakage detection methods have relied on distinguishing one group of measurements acquired when supplying the device under test with random inputs from another group recorded when the device received a fixed input over and over again (although both groups should be recorded in a randomly interleaved sequence [SM15]). Yet it was pointed out at EUROCRYPT 2016 [DS16] that a partitioning based on two different fixed inputs normally leads to a lower data complexity (i.e., fewer traces required for a successful detection). The arguments given by the authors are essentially the same that led us to suggest fixed-vs-fixed as the default partitioning strategy for DL-LA. In fact, the whole DL-LA concept is applicable to a fixed-vs-random grouping as well. However, in that case a larger data complexity has to be expected. Intuitively, this can be understood best when picturing the distributions for both groups of measurements at one individual sample point in the traces. In case of one group for fixed and one for random inputs, the two distributions will always overlap if their cardinality is sufficiently large, since the fixed input is also contained in the set of all inputs from which the random inputs are selected. A larger difference between the two distributions is possible for two distinct fixed inputs. In some cases the distributions may even be disjoint and allow perfect classification

into the two groups. While this will not occur for measurements of securely masked implementations, similar arguments can be made for higher-order statistical moments. Generally, the maximum difference between two distributions at one sample point recorded for fixed inputs will always be larger than the difference between one fixed and one random group for a sufficiently large number of traces. When measuring the execution of a cryptographic primitive over an extended period of time for each trace, sample points with large differences between the two fixed classes will inevitably occur [DS16]. This is a conceptual difference to analysis techniques that do not record a trace over time, but rather take a single snapshot of the current state, such as static power SCA attacks [Moo19]. In those cases the two fixed inputs need to be selected with greater care in order to not accidentally choose two fixed classes which lead to very similar leakage distributions. In our experimental analysis, however, the leakage traces have a significant length in terms of collected sample points ($\geq 2\,000$) and covered clock cycles (> 20). Thus, following the arguments of [DS16], it is unlikely that such traces recorded for two fixed classes show a smaller maximum difference over the full length of sample points and clock cycles than traces recorded for a fixed class and a random class. Hence, we are confident, and our experimental attempts have confirmed this, that a fixed-vs-fixed partitioning strategy is preferable for the DL-LA (and t- and χ^2-test) methodology over a fixed and a random class.

3.2 Overall Methodology

We assume that the recorded traces have already been separated into a set of N training traces and a set of M validation traces, the latter of which should have an equal number of elements from both groups to maximize the statistical confidence value that can be obtained during the evaluation[4]. Initially, we determine the point-wise mean μ and standard deviation σ of the whole trace set and standardize both the training and the validation set by calculating

$$X_i^j := (X_i^j - \mu_i)/\sigma_i,$$

with j denoting the trace and i the time sample within the trace. This very lightweight and universal pre-processing step is necessary to reach a homogeneous range between all input points and weights thus enabling efficient training.
Afterwards, the evaluator has to pick a confidence level, i.e., an upper bound on the chance that a false positive occurs. We assume the common threshold in SCA evaluations of $p_{\text{th}} = 10^{-5}$. Now, let v be the validation accuracy obtained by the neural network, then the total number of correct classifications is computed as $s_M = v \cdot M$. Considering the null hypothesis $\mathcal{H}_0$ where the neural network did not learn anything and classifies randomly (coin flip model), this corresponds to modeling the total number of correct guesses as a random variable following a binomial distribution

$$\mathcal{H}_0 : X \sim Binom(M, 0.5).$$

The probability that at least s_M correct classifications occur in a purely random classifier is given by: $P(X \geq s_M)$. This probability is easily computed as

$$P(X \geq s_M) = \sum_{k=s_M}^{M} \binom{M}{k} 0.5^k 0.5^{M-k} = 0.5^M \sum_{k=s_M}^{M} \binom{M}{k}$$

Now, we say that the implementation leaks information about the processed data if

$$P(X \geq s_M) \leq p_{\text{th}}.$$

In this case the exact location of leakage can be determined subsequently by Sensitivity Analysis (cf. Section 3.4).

[4] We provide a discussion on the size of both sets in Section 5.

Table 1: Minimum validation set sizes calculated for different validation accuracy values v obtained in Step 1.

v	M'	$v \cdot M'$
75.00 %	76	57
60.00 %	470	282
56.00 %	1 300	728
51.00 %	45 600	23256
50.50 %	182 200	92011
50.10 %	4 549 000	2279049
50.05 %	18 194 000	9106097

3.2.1 Minimum Size of the Validation Set

As previously explained, there is always a trade-off between the size of the training set and the size of the validation set while the minimization of their combined cardinality is not trivial. Assuming the size of training set has been set to a fixed value N by the evaluator, for instance because it determines the lifetime of the key or the whole device, then choosing the minimum size of the validation set for a confident result can be approached by the following iterative procedure.

- *Step 1*: Choose $M = 1{,}000$ as the size of the validation set (or any other number of traces that can be recorded in a short period of time)
- *Step 2*: Perform DL-LA using N training and M validation traces, observe the validation accuracy v
- *Step 3*: Find smallest integer M' such that $p_{\text{th}} \geq 0.5^{M'} \sum_{k=v \cdot M'}^{M'} \binom{M'}{k}$
- *Step 4*: Perform DL-LA using N training and M' validation traces, observe the validation accuracy v'
- *Step 5*: If $p_{\text{th}} \geq 0.5^{M'} \sum_{k=v' \cdot M'}^{M'} \binom{M'}{k}$, the procedure terminates, otherwise set $v = v'$ and repeat from the Step 3

This approach can be useful to approximate the total number of traces the evaluator requires in addition to the training set in order to achieve a confident result, but only in case detectable leakage is present. It relies on the assumption that the validation accuracy which the trained classifier achieves on a comparably small validation set can approximately be maintained on a larger set. The smaller the initial value of M is, the likelier it is that this assumption can be incorrect. In such a case, multiple iterations may be required. We have listed exemplary results of the procedure in Table 1. Please note, in case the trace set does not contain enough information for distinction between the two groups or the two groups indeed belong to the same population (i.e., the null hypothesis is true), the procedure will never terminate and M' will approach infinity.
In general, minimizing the validation set will yield results where the confidence threshold is just overcome. Often it can be useful to increase the validation set beyond M' in order to achieve higher confidence values. We would like to insist that the number of validation traces may often be the bottleneck for reducing the number of evaluation traces (training + validation) when analyzing unprotected implementations or generally traces that show significant amounts of leakage. However, when evaluating SCA-protected implementations, it is not uncommon, in our experience, that tens or hundreds of millions of traces are required for a meaningful analysis. In such cases, the validation set is typically not the prohibitive element. In our experience, validation sets larger than 5 or 10 million traces

should not be necessary for any regular analysis, while larger training sets will often be required. This is also showcased in some of our case studies in Section 4.

3.3 Proposed Network Structures

As already discussed in Section 1, our goal is to select and propose networks that perform robustly on many different sets of side-channel data instead of maximizing the performance towards one particular data set. In other words, we try to keep the network architectures generally applicable and as free of any assumptions about the leakage to be analyzed or the underlying implementation as possible. We have taken multiple approaches in order to find such networks. First of all we have collected side-channel data containing different types of leakage, such as first-order, higher-order, univariate and multivariate leakages, from different kinds of devices, FPGA, ASIC, µC, while simulating different levels of measurement quality, such as high signal-to-noise ratio, low signal-to-noise-ratio, aligned and misaligned leakage traces. Once this collection had been assembled we essentially followed a trial-and-error based approach in order to find the most suitable number of layers and number of neurons per layer to built a simple multi-layer perceptron (MLP) providing the best average classification performance across our data sets. The resulting network, which is described below, is even fairly robust to small changes to its hyper-parameters.
In addition to the MLP we also suggest a simple convolutional neural network (CNN). In order to find suitable hyper-parameters for this CNN we performed a hyper-parameter search with Talos [mea20] on our data sets corresponding to all case studies. As a result we suggest a network including a set of 8 different hyper-parameter combinations which are evaluated against each other in Section 4. In summary, the two network architectures proposed below have been selected because they proved to deliver a respectable level of universality across a number of experimental data sets. Of course, we do neither claim that the selected networks are the optimal solution for such purposes, nor that they necessarily provide appropriate performance on any given set of side-channel data. Yet, we are confident that they represent a good starting point for an investigation.
We have built, tested and evaluated both of the network architectures described below in the Python library Keras (keras-gpu version 2.4.3) using TensorFlow (tensorflow-gpu 2.1.0) as the backend.

3.3.1 Multi-Layer Perceptron (MLP)

The MLP network consists of four fully-connected layers (*Dense*) of 120, 90, 50 and 2 output neurons. The input layer and each of the inner layers use a *Rectified Linear Unit* (*ReLU*) as an activation function, while the final layer uses *softmax*. The four Dense layers are each separated by a *BatchNormalization* layer. In summary, the model can be defined in Python as:

```
model = Sequential([
    Dense(120, activation = 'relu', input_shape= (tracelength,) ),
    BatchNormalization(),
    Dense(90, activation = 'relu'),
    BatchNormalization(),
    Dense(50, activation = 'relu'),
    BatchNormalization(),
    Dense(2, activation = 'softmax')])
```

Further, we used the *mean squared error* as a loss function and *adam* as an optimizer with the default parameters provided by Keras[5]. We chose the batch size as 2 000 samples for

[5] $lr = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}, decay = 0.0$

traces of length 5 000 points and 20 000 samples for traces of length 500 points and 1 00 for traces of length 200 000 points.

Justification. We chose $ReLU$ defined as

$$relu(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

as an activation function over other common possibilities, e.g., tanh or sigmoid, because of better results regarding validation accuracy in initial tests as well as for better computational performance when operating on large data sets (which is highly relevant for the evaluation of protected implementations). We chose the softmax activation function of the final layer to create a probability distribution over both classes as explained in Section 2.2. The purpose of each *BatchNormalization*-layer is to decouple the learning process of all Dense layers from each other and additionally provide a means of regularization to prevent overfitting [IS15].
We confirmed the suitability for univariate and multivariate leakage located in different statistical orders and for traces as short as one point and as long as 200 000 points which may be encountered during a typical leakage evaluation of symmetric cryptographic primitives and provide extensive depth on the performance of our leakage assessment approach in different case studies in Section 4.

3.3.2 Convolutional Neural Network (CNN)

In addition to the comparison between the MLP-based DL-LA and the classical detection approaches in our case studies in Section 4, we performed a hyper-parameter search for CNNs with Talos [mea20] on the data sets corresponding to all case studies. We utilized the following convolutional network with one convolutional, one pooling and a final dense layer

```
model = Sequential([
    Reshape((tracelength,1), input_shape = (tracelength,)),
    Conv1D(filters=filter, kernel_size=kernel_mult*peakdist, \
           strides=peakdist//strides, input_shape=(tracelength,1), \
           activation='relu'),
    MaxPooling1D(pool_size=pool),
    Flatten(),
    Dense(2, activation='sigmoid')])
```

to provide an extremely simple high-level network architecture that is not specialized for any particular power traces and then performed a search through the following eight hyper-parameter combinations:

```
    filter: [12]
    kernel_mult: [2, 3]
    strides: [2, 3]
    pool: [2, 4]
```

Note that the kernel size and stride distance is given relative to the distance of peaks, i.e., the length of a clock cycle, in the power trace. We chose the multiplicative factor larger than one to assure that the convolution combines information from multiple clock cycles. While we limited ourselves to the given eight hyper-parameter combinations, many choices greater than one for filter, kernel multiplier, strides and pool are theoretically sound and should lead to comparable results (cf. Section 4.1). In contrast to the MLP network we used the *binary_crossentropy* as a loss function and chose *sigmoid* as the activation function of the final layer.

3.4 Extracting Temporal Information

If leakage is detected, the hardware designer or evaluator is usually interested in exactly pinpointing the leakage locations to report or alleviate the shortcoming, e.g., masking flaws. By applying *Sensitivity Analysis (SA)* based on input activation gradients [SZ13, Tim19, MDP19a, PEC19], we can exactly locate all points of interest by quantifying how much they contributed to the leakage function learned by the neural network. In short, SA determines the partial derivates of one output coordinate of the neural network with respect to the network inputs, thereby characterizing the effect of a slight change in each individual input on the classification outcome.
We perform SA on the final network after training has completed by averaging the gradients of one output coordinate (with respect to the network inputs) weighted with the network inputs for all samples in the training set and subsequently take the absolute value. More precisely, let x_i denote the i-th input of our network, y_0 the first output coordinate of the network and X_i^j the value of the i-th input for trace j in the training set. Then the sensitivity can be determined as:

$$s_i = \left| \sum_j \frac{\partial y_0}{\partial x_i} \cdot X_i^j \right|.$$

While the actual value of this expression has to be determined via the chain-rule over all network layers, this process is fully-automated by TensorFlow such that the remaining effort for the evaluator is a single function call. Instead of considering the network inputs, a sensitivity analysis may also be performed based on the first layer weights [Tim19].

3.5 Common Pitfalls

We discuss the most important differences between the classical detection approaches and DL-LA and aim to preempt common pitfalls an evaluator might encounter with our leakage assessment:

Group Imbalance. While the classical TVLA based on the Welch's t-test as well as the χ^2-test can handle groups imbalanced in mean, variance and size, we want to stress that an equalization of group sizes in the validation set is extremely important for DL-LA. If the groups are imbalanced, the test statistic no longer follows the distribution $X \sim Binom(M, 0.5)$. Instead, always assigning the label of the more common group leads to a classifier which outperforms random guessing, without actually being able to distinguish the groups based on their traces. This discrepancy between actual and theoretical distribution of the test statistic – given a sufficiently large validation set – will lead to false positives. While the problem of severe group imbalance has previously been addressed by sampling techniques [PHJ+19], we instead strongly advise pruning both groups in the validation set to an exact ratio of 50/50 to achieve the highest possible confidence.

Probability Adaption. An obvious idea to counteract the issue just addressed is the adaption of the success probability in the Binomial distribution. Assume a slight (or even significant) imbalance ϵ in group sizes

$$\frac{|G_0|}{|G_0| + |G_1|} = 0.5 + \epsilon.$$

The evaluator could simply adapt the distribution of the test statistic to

$$X \sim Binom(M, 0.5 + \epsilon).$$

While this might even show satisfactory results in the case of low noise and unprotected or severely flawed implementations, which in turn lead to a high validation accuracy, we caution against any alteration of the distribution. In all practically relevant cases (protected implementation, moderate noise) a change of the success probability severely lowers the confidence of the statistical test. More specifically, consider a validation set of size 500 000 traces over which a validation accuracy of $v = 0.506$ has been achieved. In case of a balanced validation set this event is highly statistically significant (10^{-17}). However, if the validation set contains a small bias of $\epsilon = 0.004$ no significance can be concluded as the remaining likelihood for this event is only 10^{-3}. It is obvious that the adapted test looses its statistical power in all interesting cases; hence, false negatives might occur. Therefore, we want to reinforce the previous point to prune the validation set to an exact 50/50 ratio.

Overfitting. We caution against using an overly complicated neural network as it might lead to overfitting, which is defined by a continuous rise of the training accuracy over the number of epochs while the validation accuracy begins to fall. The underlying cause of this effect is the memorization of the training set as opposed to learning generalizable features of the entire set. Hence, it can be prevented by using a network with a simple structure which does not contain excessively many weights and optionally includes Normalization, Regularization or Dropout layers (cf. Section 3.3).

4 Experimental Results

In the following we provide an experimental verification of the suitability of DL-LA as a black box leakage assessment strategy. We strive for a realistic benchmark of our approach with a clear real-world impact. For this reason we chose power measurements of multiple hardware and software implementations of the PRESENT-80 ultra lightweight block cipher [BKL+07] as the common target in our case studies. PRESENT has been developed for ubiquitous and resource constrained computing environments, which exactly constitutes the type of application that commonly requires side-channel security as a design goal and may be certified by third-party evaluations labs. In total, we target nine different implementations of the cipher on three different platforms (FPGA, ASIC, µC). As a first step we evaluate the MLP (see Section 3.3) on all nine case studies for one concrete choice of hyper-parameters. Those results are compared to the previously introduced state-of-the-art leakage detection methods Welch's t-test and Pearson's χ^2-test. Afterwards, in order to demonstrate that robust detection on trace sets from 3 different platforms is not limited to this particular MLP, we evaluate the CNN (see Section 3.3) for a whole spectrum of hyper-parameters. Among the nine different implementations that are tested, multiple are protected by masking and hiding techniques. The masked variants feature provable first-order security. Some of them even provide security at any order against univariate-only attacks. The results show that DL-LA with our choice of network architectures is able to confidently detect leakage in smaller trace sets or with a higher confidence using the same amount of traces compared to the conventional methods.

4.1 Measurement Setup

For the first 7 case studies, we have implemented the different instances of the PRESENT block cipher on a SAKURA-G board [sak] which has specifically been designed for SCA evaluations. The board features two Spartan-6 FPGAs, one as a target and the other as a control interface. Case studies 8 and 9 feature protected versions of the PRESENT cipher implemented on a 40 nm ASIC prototype and an ARM Cortex-M0 microcontroller respectively. In all cases we have measured the voltage drop over a 1 Ω shunt resistor in

Table 2: Measurement details for the nine different case studies.

	Case Study 1	Case Study 2	Case Study 3	Case Study 4	Case Study 5
Platform	FPGA	FPGA	FPGA	FPGA	FPGA
Alignment	aligned	misaligned	heavily misal.	aligned	misaligned
Sampling Rate	1 GS/s	1 GS/s	1 GS/s	1 GS/s	1 GS/s
Frequency	6 MHz	6 MHz	$\leq$ 24 MHz	6 MHz	6 MHz
No. of Traces	1 000	1 000	5 000	10 000 000	10 000 000
No. of Points	5 000	5 000	5 000	5 000	5 000

	Case Study 6	Case Study 7	Case Study 8	Case Study 9
Platform	FPGA	FPGA	40 nm ASIC	Cortex-M0 µC
Alignment	aligned	misaligned	aligned	aligned
Sampling Rate	100 MS/s	100 MS/s	2 GS/s	500 MS/s
Frequency	6 MHz	6 MHz	12 MHz	8 MHz
No. of Traces	50 000 000	50 000 000	50 000 000	100 000
No. of Points	5 000	5 000	2 000	200 000

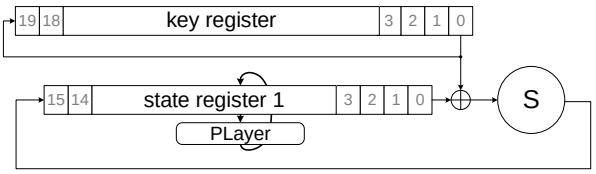


Figure 1: Unprotected serialized PRESENT architecture with a 4-bit data path.

the V_{dd} path of the target with a digital sampling oscilloscope. On the SAKURA-G board the measured signal is amplified through a built-in AC amplifier. The measurement details for each of the nine different case studies including sampling rate, operating frequency, number of traces and number of time samples per trace are listed in Table 2. For all case studies we measured side-channel traces in a fixed-vs-fixed manner for two arbitrarily selected fixed inputs. We have taken care to follow all rules that have to be considered to avoid false positives in leakage assessment [SM15], e.g., the measurements of the two groups are randomly interleaved and in the masked cases the communication between the control unit and the target is performed in a shared manner (in our case the same holds for the communication with the measurement PC).

Case Study 1: Unprotected PRESENT (FPGA), aligned Traces

In this first case study we target an unprotected serialized implementation of the PRESENT block cipher. The architecture can be seen in Figure 1 and is similar to profile 1 introduced in [PMK+11]. As a first step we evaluate the confidence to distinguish the two groups of measurements (fixed-vs-fixed) by conventional methods. The results of the first-order t-test and the χ^2-test can be seen in Figure 2. In both cases we plot the confidence values p instead of relying on the common (and less precise) approach of defining a threshold for the intermediate statistics (e.g., $|t| > 4.5$). The t-test succeeds in providing a confidence higher than 99.999 % for the distinguishability of the two groups after about 20 traces since it shows a probability below 10^{-5} to accept the null hypothesis. The χ^2-test requires approximately 90 traces to overcome the desired confidence threshold. In conclusion, none of the two methods faces any problems to distinguish the leakage distributions with a high confidence when 1 000 traces are considered.
When applying DL-LA to the same traces, the results in Figure 3 are achieved. We have to state here that a plot as depicted in Figure 3(b) is rather unnatural to obtain using DL-LA.

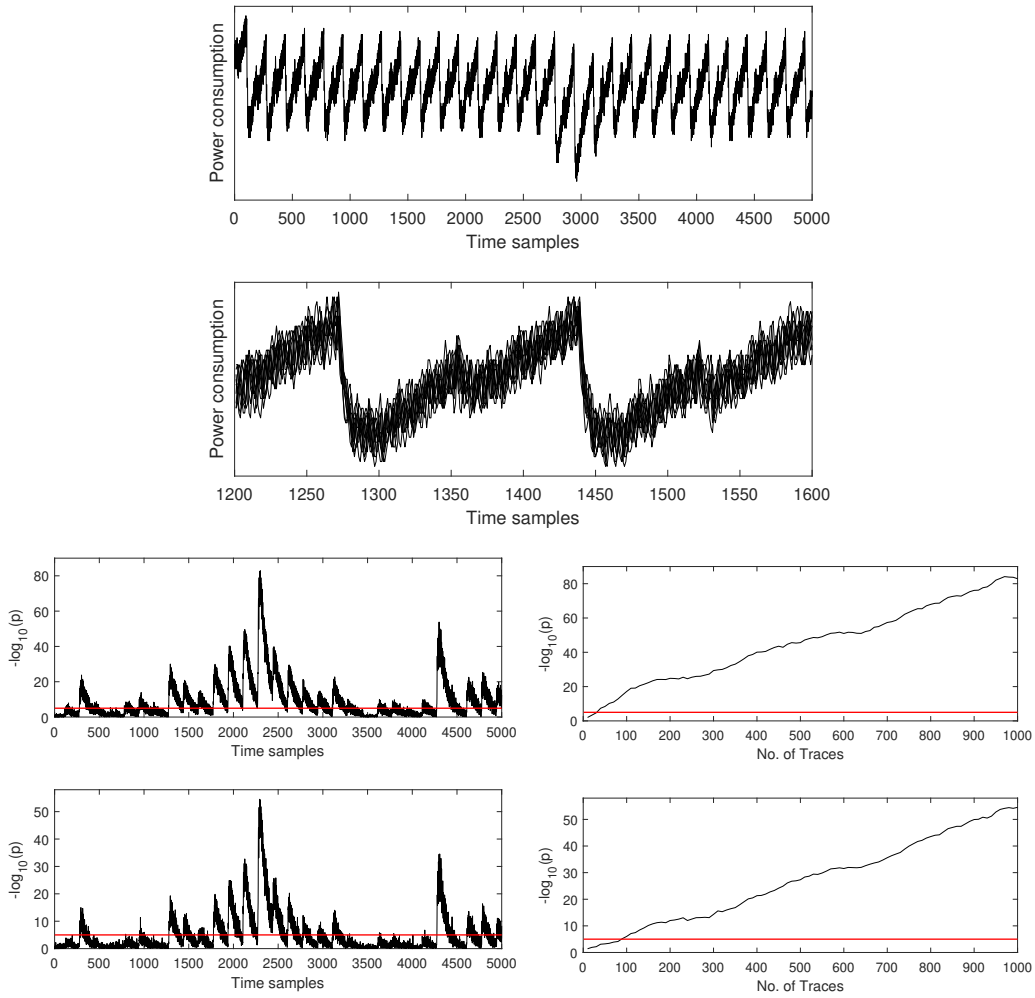


Figure 2: Univariate leakage assessment using 1 000 traces (step size 10) of an unprotected serialized PRESENT-80 implementation. From top to bottom: 1) Sample trace, 2) Overlay of 10 sample traces, 3) t-test results, 4) χ^2-test results.

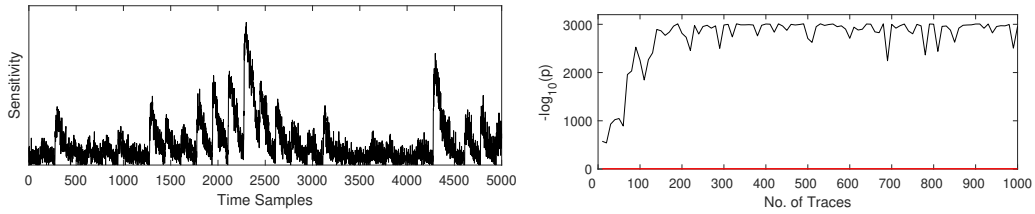


Figure 3: Sensitivity Analysis and DL-LA using 1 000 traces (step size 10) of an unprotected serialized PRESENT-80 implementation. For each p value 30 epochs and a validation set of 10 000 traces are considered.

Normally, training and validating the network results in a confidence value after each epoch. Thus, it would be more natural to train the network on a training set of fixed size and to show the p values over the number of epochs to determine how many are required to overcome the threshold. However, in order to offer the best possible comparison between the leakage assessment approaches we repeated this process 100 times for a fixed number of epochs (30) and a training set that increases by 10 traces per step and plotted the maximum confidence over the number of traces. The result shows that a network which is trained on only 10 traces is already capable of providing an extremely high confidence that the two groups are distinguishable (since large $-\log_{10}(p)$ values give confidence to reject the null hypothesis). By increasing the size of the training set the confidence is boosted significantly until the p values stagnate in a corridor between 10^{-2300} to 10^{-3011}. Please note that, as the validation set has a size of 10 000 traces, the maximum achievable p value is $0.5^{10\,000} = 10^{-3011}$. Thus, the stagnation in the corridor is simply caused by the fact that (almost) all of the traces in the validation set were classified correctly. By using a larger validation set the $-\log_{10}(p)$ values would rise even beyond 3011. We also perform a Sensitivity Analysis on the network to determine the points of interest and obtain a spatial resolution comparable to the univariate tests (cf. Figure 3(a)). The absolute values of the SA are not meaningful and cannot be compared to any threshold. Thus, they are omitted here. In summary, DL-LA outperforms the classical detection approaches in terms of required number of traces and absolute confidence provided. Of course, for the evaluation of DL-LA as performed in this case study, a validation set is required on top of the training set. However, please note that we only chose a validation set of 10 000 traces here in order to show the extremely high magnitude of achievable confidence values, even when considering very small training sets[6]. In fact, the indication of distinguishability relates only to the training set, and, in case the network learned generalizable features from it, the confidence can be arbitrarily boosted by increasing the validation set. If no generalizable features were learned (e.g., because no leakage is present) the percentage of correct classifications will not be different from 0.5 by a statistically significant magnitude. The advantages of decoupling the confidence from the number of traces (in the *training set*) are discussed in Section 5. In Figure 28 of Appendix A, we additionally provide DL-LA results for the first three case studies where the size of the union of the training and the validation set does not exceed the number of traces considered by the t- and the χ^2-test. Even in that case DL-LA outperforms the classical approaches.

Case Study 2: Unprotected PRESENT (FPGA), misaligned Traces

This case study is an exact replication of the previous one apart from the fact that we artificially created a misalignment of the traces, as apparent in Figure 4(b). This misalignment was achieved by forcing the oscilloscope to trigger the acquisition of the power traces close to the peak of the rising edge of the trigger signal (in our case at 2.48 V while the peak is at 2.5 V) as opposed to the more stable part in the middle of the edge. Thus, due to the electronic noise, the acquisition is in some cases triggered earlier than in others and the traces are not perfectly aligned anymore. Figure 4 shows that the t- and χ^2-test results do not seem to significantly suffer from this misalignment when considering the absolute magnitude of the $-\log_{10}(p)$ values. However, the number of traces to overcome the threshold is increased in comparison to the previous case study in both tests. DL-LA also performs similar as before, as apparent from Figure 5 and outscores the classical detection approaches in required traces and provided confidence. It seems that the slight misalignment of the traces does not significantly affect the detection

[6] In contrast, the minimum size of the validation set in order to be able to overcome the detection threshold is 17, as $-\log_{10}(0.5^{17}) > 5$. However, this assumes a 100% correct classification by the network, otherwise a larger set needs to be considered.

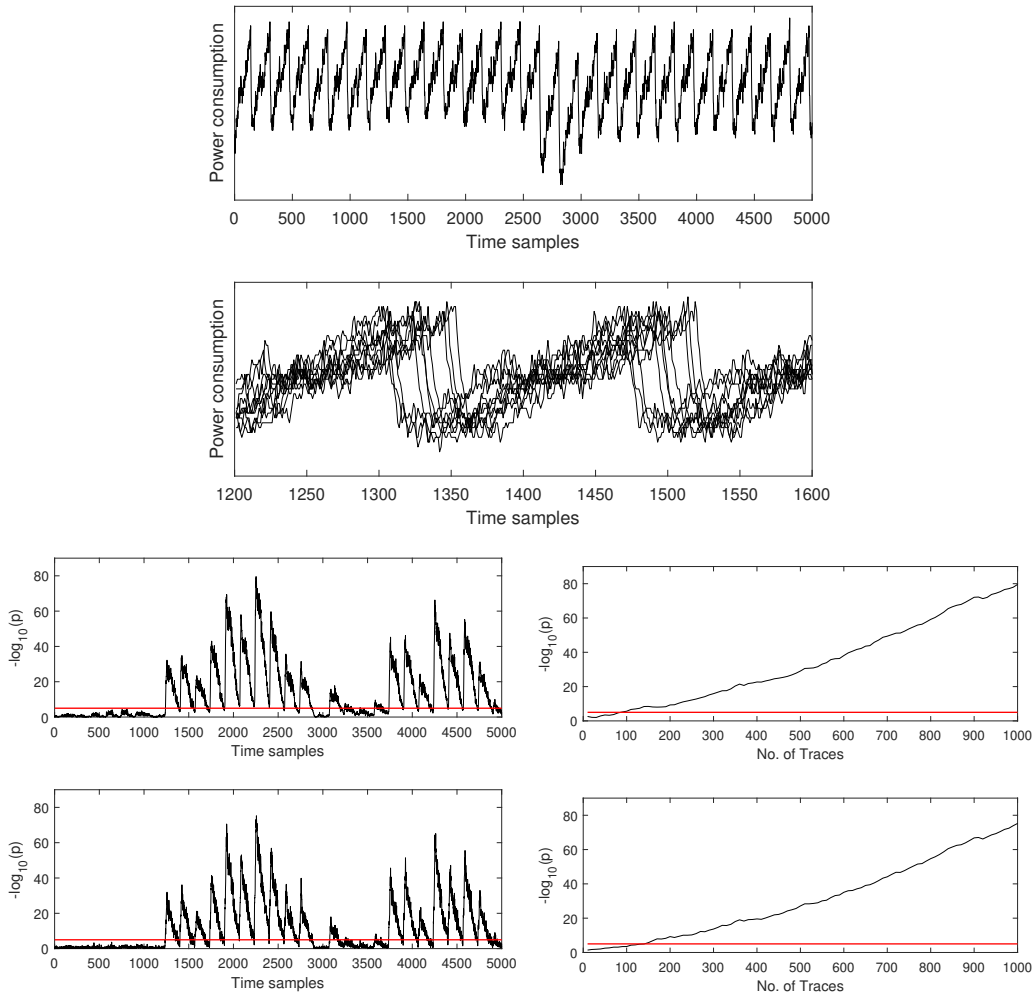


Figure 4: Univariate leakage assessment using 1 000 misaligned traces (step size 10) of an unprotected serialized PRESENT-80 implementation. From top to bottom: 1) Sample trace, 2) Overlay of 10 sample traces, 3) t-test results, 4) χ^2-test results.

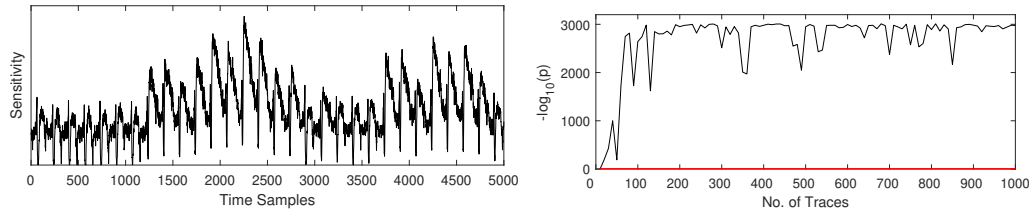


Figure 5: Sensitivity Analysis and DL-LA using 1 000 misaligned traces (step size 10) of an unprotected serialized PRESENT implementation. For each p value 30 epochs and a validation set of 10 000 traces are considered.

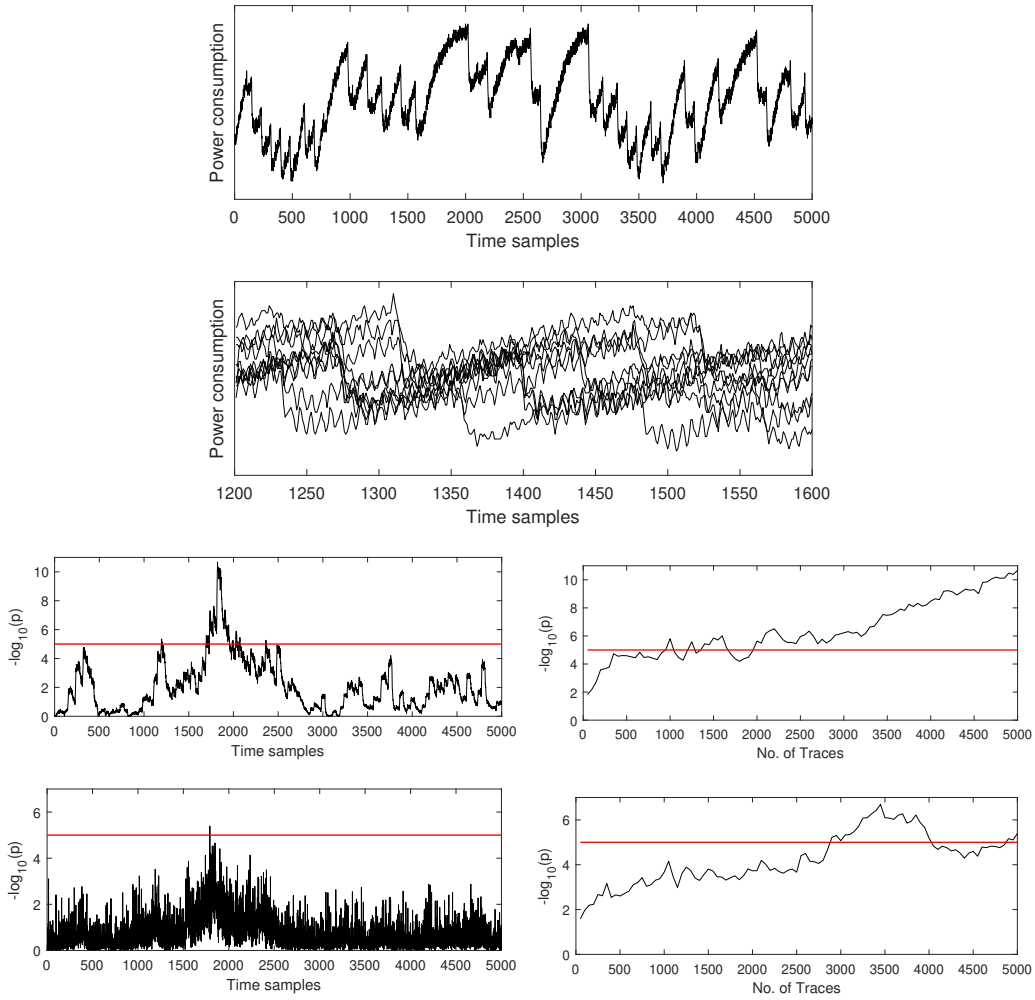


Figure 6: Univariate leakage assessment using 5 000 traces (step size 50) of a serialized PRESENT-80 implementation with clock randomization. From top to bottom: 1) Sample trace, 2) Overlay of 10 sample traces, 3) t-test results, 4) χ^2-test results.

capabilities of any of the leakage assessment techniques when unprotected implementations are considered and the number of available traces is not chosen to be extremely small.

Case Study 3: (Unprotected) PRESENT (FPGA), randomized Clock

Since the artificial delay in the previous case study only slightly increased the data complexity of a leakage detection we now try to test a countermeasure that leads to much more heavily misaligned and noisy traces. In particular, we randomize the clock that drives the targeted PRESENT implementation. This is done by clocking the cipher with the output of a 64-bit LFSR. Hence, in each encryption (and therefore also in the power traces) the same intermediate computations are executed at different times, since the rising edges of the LFSR output occur in a random sequence. The input frequency of the LFSR was set to 24 MHz so that the number of rising edges in a certain frame of time is on average similar to being clocked by a stable 6 MHz clock. In this case the t- and χ^2-test struggle more significantly to detect leakage than in the previous experiments, as apparent in Figure 6. While the t-test requires about 2 000 traces for a detectable breach

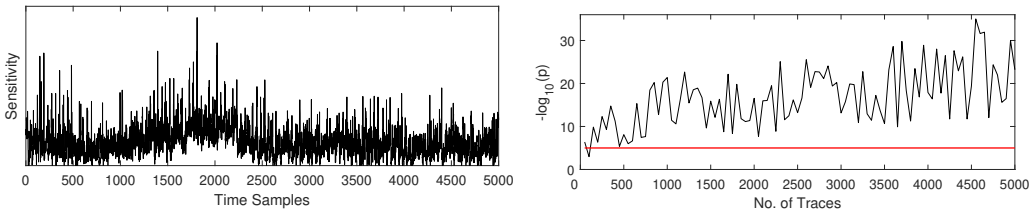


Figure 7: Sensitivity Analysis and DL-LA using 5 000 misaligned traces (step size 50) of an unprotected serialized PRESENT implementation with clock randomization. For each p value 30 epochs and a validation set of 10 000 traces are considered.

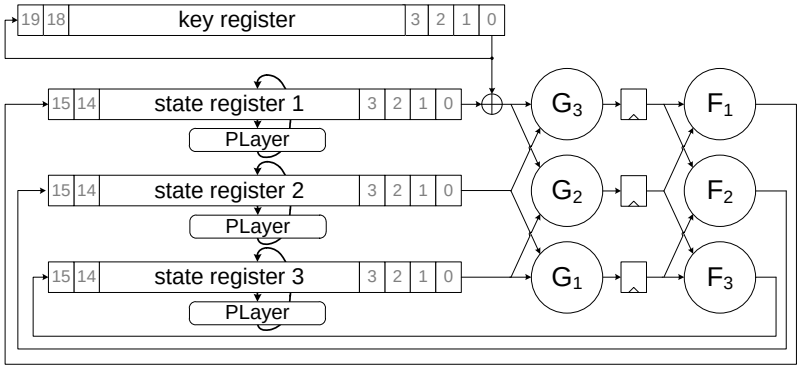


Figure 8: Serialized PRESENT threshold implementation architecture with 3 shares and a decomposed Sbox.

of side-channel security, the χ^2-test barely overcomes the threshold at all. DL-LA on the other hand is able to confidently state distinguishability after about 150 traces (cf. Figure 7). Although all three approaches suffer significantly from the misalignment and the added noise, DL-LA is still able to perform detection on a much smaller amount of traces. Please note that, if desired by the evaluator, the confidence can be made arbitrarily larger by increasing the size of the validation set.

Case Study 4: PRESENT TI (FPGA), aligned Traces

In this case study we target a serialized threshold implementation (TI) [NRR06] of the PRESENT block cipher. The architecture can be seen in Figure 8 and is similar to profile 2 introduced in [PMK+11]. The PRESENT Sbox is decomposed into two quadratic functions F and G. Both of those decompositions are split into three component functions each according to the concepts of *correctness*, *non-completeness* and *uniformity* [NRR06]. As apparent from Figure 8 the three shares in the computation of the decomposed Sbox are evaluated in parallel. Thus, no first-order, but univariate higher-order (especially second- and third-order) leakage is expected. We evaluate this assumption in Figure 9. As expected the first-order t-test does not indicate detectable leakage, but the second- and third-order tests do. Interestingly, we can confirm the statements made by the authors of the χ^2-test proposal [MRSS18] regarding the shortcomings of the moment-based nature of the t-test. Unlike the situation in the previous case studies, the χ^2-test outperforms the t-test here. While the second-order and the third-order t-test require 3 000 000 and 1 100 000 traces for the detection respectively, the χ^2-test succeeds after only 600 000 traces and results in a much higher confidence over all.

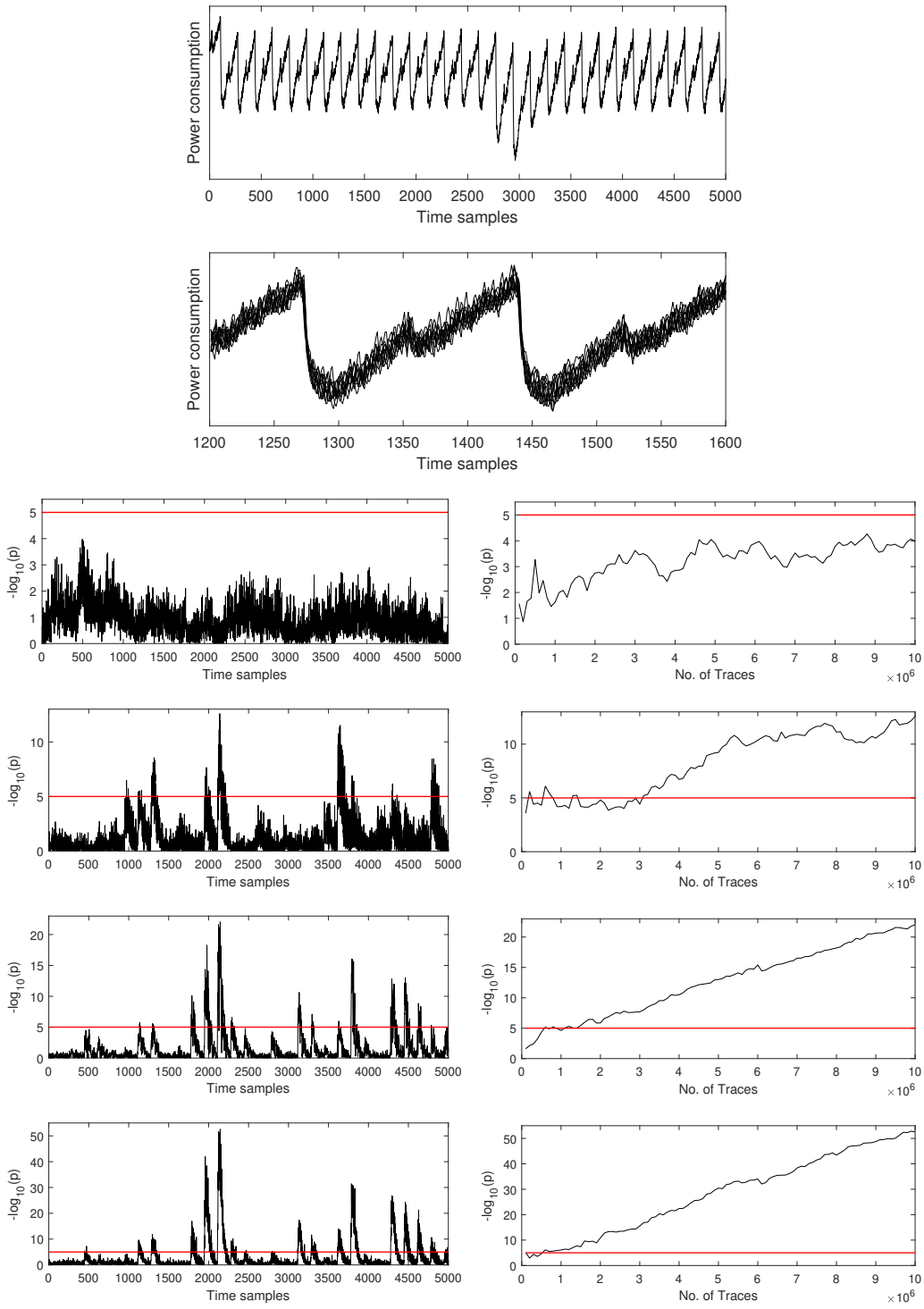


Figure 9: Univariate leakage assessment using 10 000 000 traces (step size 100 000) of a serialized PRESENT threshold implementation. From top to bottom: 1) Sample trace, 2) Overlay of 10 sample traces, 3) first-order t-test results, 4) second-order t-test results, 5) third-order t-test results, 6) χ^2-test results.

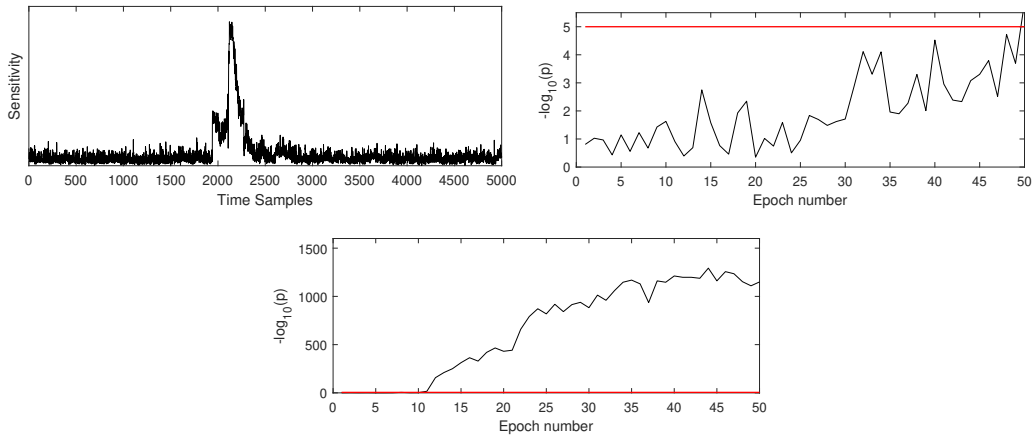


Figure 10: Sensitivity Analysis and DL-LA using 500 000 (Fig. 10(b)) and 3 000 000 (Fig. 10(a), 10(c)) traces of a serialized PRESENT threshold implementation respectively. For each p value a validation set of 1 500 000 traces is considered.

Please note that for this case study and the upcoming ones, where we analyze protected implementations, we change the visualization of the DL-LA results. Due to the large data sets involved it is not feasible to train many different classifiers with a steadily increasing size of the training set over many steps. Instead we visualize the $-\log_{10}(p)$ values over the number of epochs (instead of over training traces). We do this twice, once for a number of traces below the minimum required by the classical assessment method (here χ^2-test with 600 000 traces) and once for a larger training set in order to show that much larger confidence values can be achieved in the protected cases as well. Those results are presented in Figure 10. In case of a training set of size 500 000, the DL-LA succeeds only just in overcoming the confidence threshold. However, barring the possibility of a false positive results (which is highly unlikely), the confidence could be increased by an evaluator either by considering more epochs or by increasing the validation set. In the case of a training set including 3 000 000 measurements, the confidence that side-channel leakage is present becomes extremely large. In fact much larger than the confidence results achieved by the t- and χ^2-test given all 10 000 000 traces.

Case Study 5: PRESENT TI (FPGA), misaligned Traces

This case study is equivalent to the previous one apart from the fact that we artificially created a misalignment of the traces, as it was already done for case study 2. As a result of this misalignment the leakage detection approaches require slightly more traces to overcome the confidence threshold than in the aligned case. In particular, as shown in Figure 11, the second-order and the third-order t-test require 3 600 000 and 1 500 000 traces for the detection respectively, while the χ^2-test succeeds after only 800 000 traces and again results in a much higher confidence. The DL-LA is again the most powerful leakage detection mechanism and succeeds for both sizes of the training set (800 000 and 3 000 000) with much higher confidence values than any of the classical approaches (cf. Figure 12).

Case Study 6: PRESENT Multivariate TI (FPGA), aligned Traces

In the next case studies we concentrate on scenarios where the classical univariate detection approaches are naturally unsuited to detect leakage, namely purely multivariate higher-order leakages. These cases are the primary motivation to apply DL-LA in reality, as all

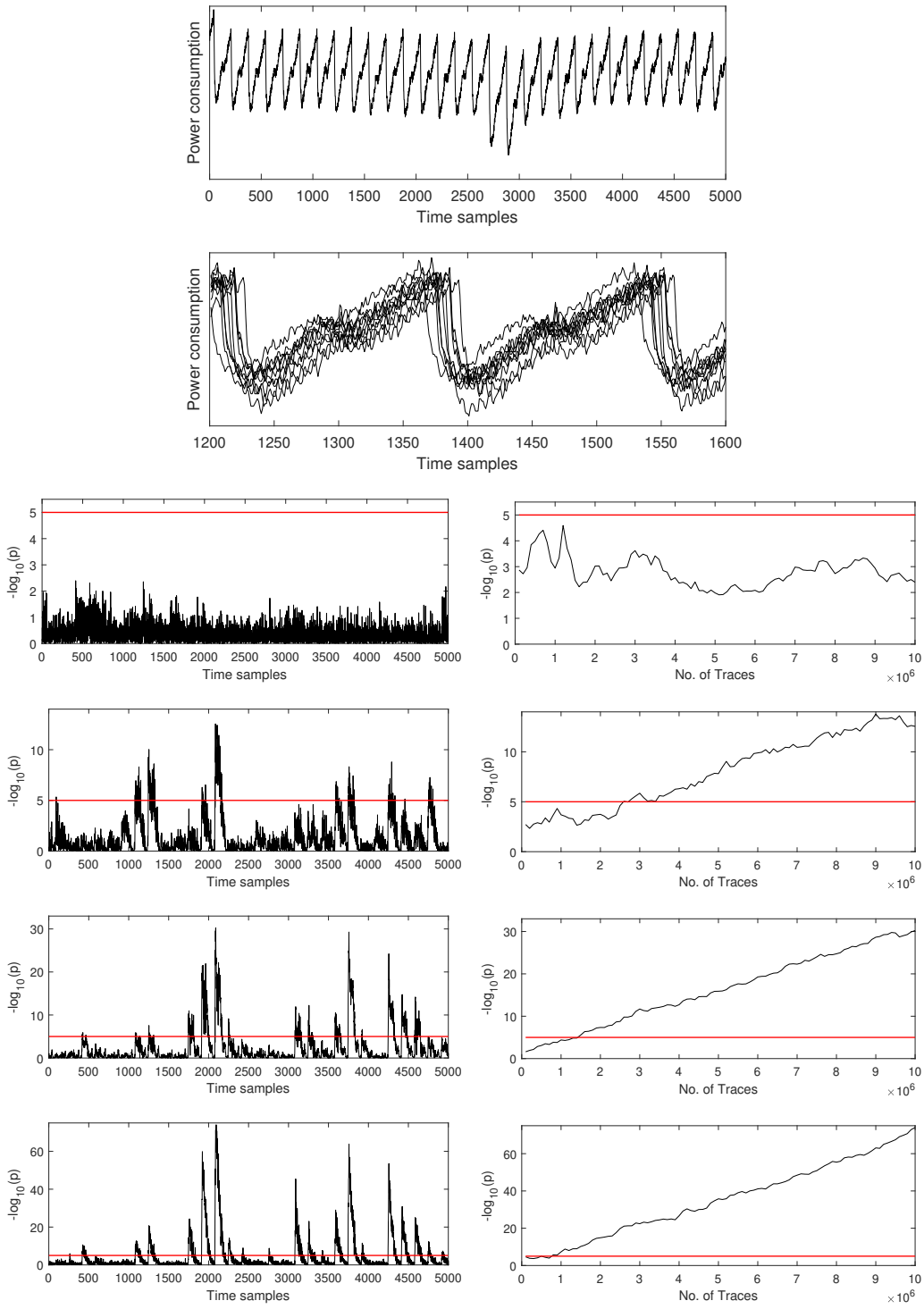


Figure 11: Univariate leakage assessment using 10 000 000 misaligned traces (step size 100 000) of a serialized PRESENT threshold implementation. From top to bottom: 1) Sample trace, 2) Overlay of 10 sample traces, 3) first-order t-test results, 4) second-order t-test results, 5) third-order t-test results, 6) χ^2-test results.

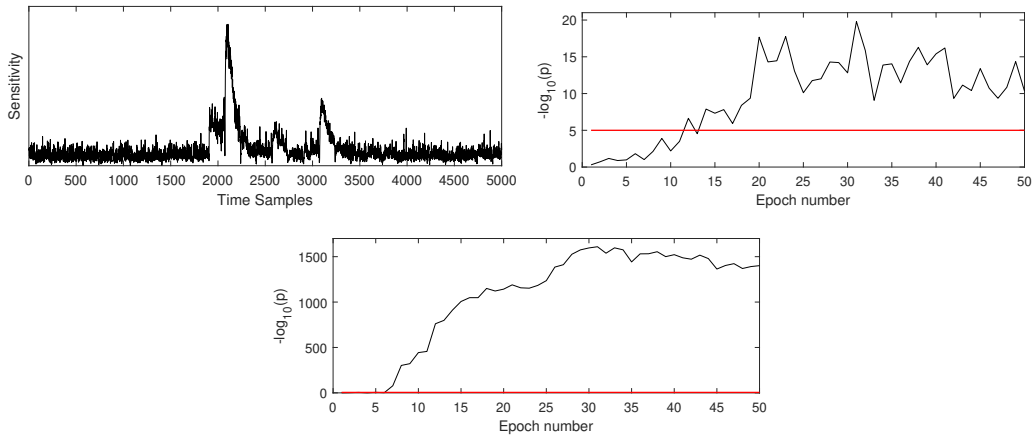


Figure 12: Sensitivity Analysis and DL-LA using 800 000 (Fig. 10(b)) and 3 000 000 (Fig. 10(a), 10(c)) misaligned traces of a serialized PRESENT threshold implementation respectively. For each p value a validation set of 1 500 000 traces is considered.

other commonly applied methods to the best of our knowledge fail to capture the whole amount of present side-channel leakage in these scenarios (at least without significant manual effort and knowledge about the underlying implementation). We provide evidence for this statement in the following.

We constructed a special version of the PRESENT threshold implementation architecture depicted in Figure 8, that does not offer univariate side-channel leakage. To this end we had to ensure that all six component function (G_1, G_2, G_3, F_1, F_2 and F_3) are evaluated sequentially and not in parallel. We did this by gating their respective inputs with `AND` gates which are controlled by a finite state machine (FSM). In addition to that we had to make sure that none of the state registers are clocked at the same time. Thus a single Sbox computation takes 7 clock cycles in the resulting hardware design. As expected, our univariate leakage assessment using the classical detection approaches does not indicate the presence of any side-channel leakage (cf. Figure 13). However, a multivariate investigation could still find higher order leakage if performed at the correct offsets. This requires either white-box knowledge about the implementation or must be determined by exhausting all possibilities. The result for the best offset leading to detectable multivariate leakage is illustrated in Figure 14 and shows leakage in the third order after more than 45 million traces. Please note that we have performed each multivariate second-order t-test, third-order t-test and χ^2-test with the correct offsets (as we know all the implementation details) and none of them was able to detect leakage with fewer traces.

In stark contrast, as apparent in Figure 15, DL-LA provides a very high confidence level of $-\log_{10}(p) > 150$ for the presence of side-channel leakage after training on only 20 million traces. We still retained the same network architecture as before and made absolutely no assumptions about the leakage and required no white-box knowledge about the offset of the individual evaluation of TI shares. Due to memory and time restrictions we pruned the traces to a length of 500 sample points (1281-1780). Validation took place on 5 million traces. Leakage becomes apparent after 25 epochs and continuously increases until our chosen threshold of 50 epochs has been reached.

Case Study 7: PRESENT Multivariate TI (FPGA), misaligned Traces

Our next case study is a replication of the previous one, but again we misaligned the traces through bad triggering. As shown in Figure 16 no univariate detection of leakage succeeds.

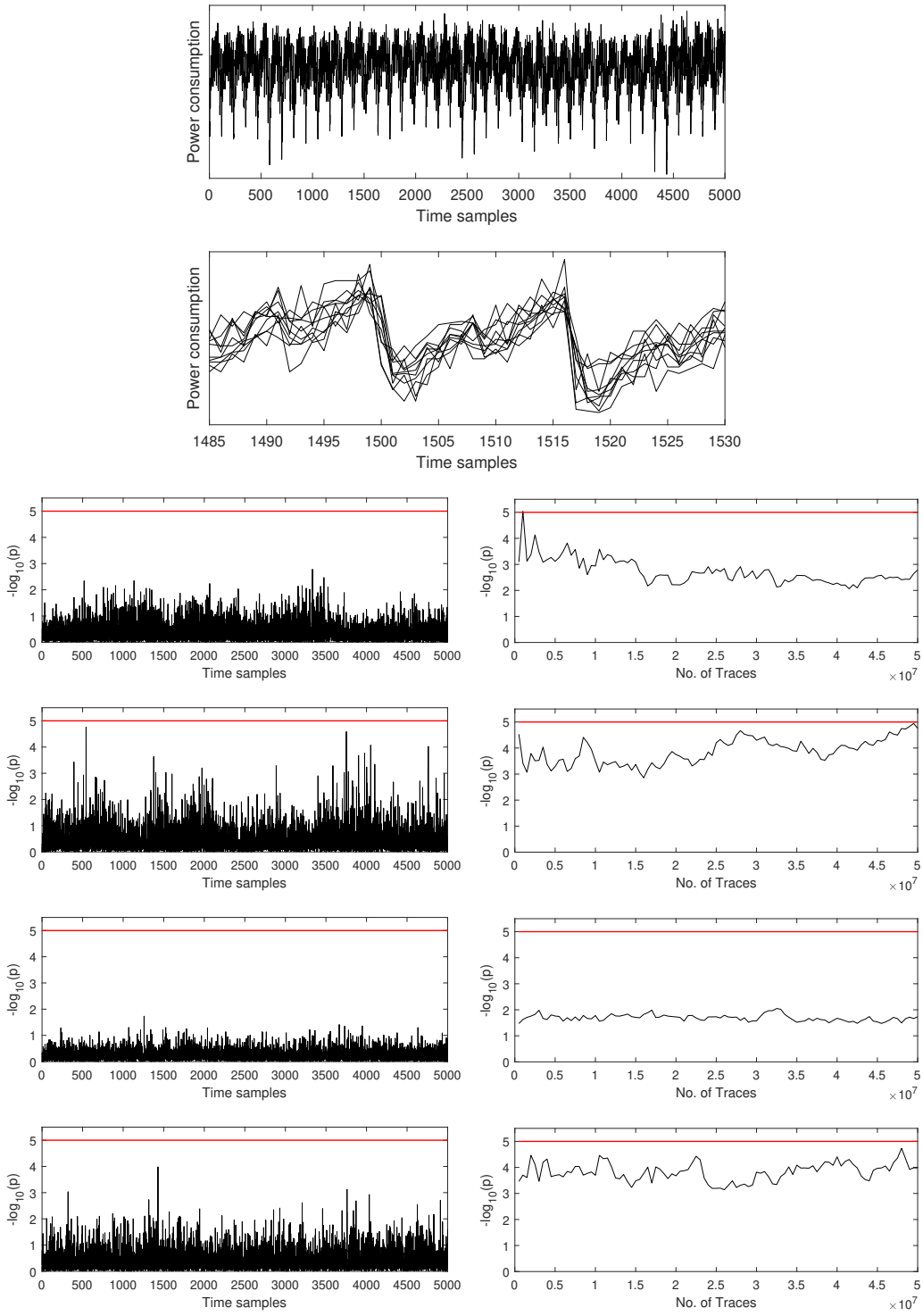


Figure 13: Univariate leakage assessment using 50 000 000 traces (step size 500 000) of a serialized multivariate PRESENT threshold implementation. From top to bottom: 1) Sample trace, 2) Overlay of 10 sample traces, 3) first-order t-test results, 4) second-order t-test results, 5) third-order t-test results, 6) χ^2-test results.

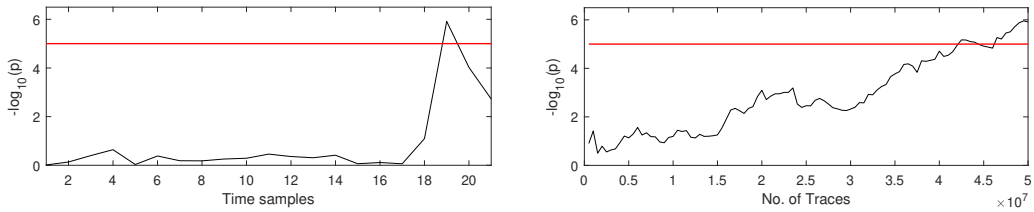


Figure 14: Multivariate third-order t-test using 50 000 000 traces (step size 500 000) of a serialized multivariate PRESENT threshold implementation.

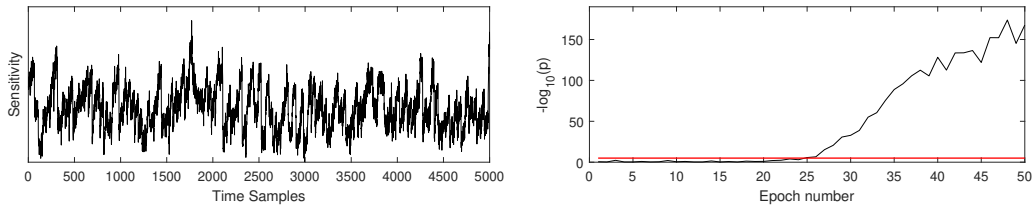


Figure 15: Sensitivity Analysis and DL-LA using 20 000 000 traces of a serialized multivariate PRESENT threshold implementation. For each p value a validation set of 5 000 000 traces is considered.

In this case however, even the multivariate third-order analysis with the best possible offset for leakage detection in the previous case study does not succeed (cf. Figure 17). In other words, the acquired set of traces does not allow detection of any leakage using conventional methods, at least in case the traces are not re-aligned before the analysis.
DL-LA however detects leakage with high confidence ($-\log_{10}(p) > 60$) after training on only half of the available traces (25 000 000). This result is depicted in Figure 18.

Case Study 8: PRESENT TI (ASIC), aligned Traces

As a complement to the FPGA-based case studies outlined on the previous pages, we have investigated a PRESENT threshold implementation realized in non-reconfigurable hardware as well, namely as part of a custom 40 nm ASIC prototype. The test chip has been developed for SCA evaluations and features several different cipher cores integrated into a larger control framework. The PRESENT implementation is the same nibble-serialized threshold implementation that has been investigated in case studies 4 and 5 already. We have measured 50 000 000 traces in a fixed-vs-fixed manner with 2000 sample points per trace. Results of a first-, second- and third-order univariate t-test as well as a χ^2-test are depicted in Figure 19. As expected, no first-order leakage can be observed, but second-order leakage can be detected consistently beyond the confidence threshold after approximately 8 000 000 traces. Since there is no detectable leakage present in the third-order, it is no surprise that the χ^2-test requires more traces to distinguish the distributions, namely about 15 000 000 traces.
However, DL-LA outperforms both methods by a large margin as it successfully classifies enough validation traces correctly to achieve a huge confidence after training on only 100 000 traces. This result is shown in Figure 20. It demonstrates that the leakage detection capability of our approach is not limited to the FPGA-based case studies. In fact, on the ASIC measurements it achieves one of the most impressive results compared to the classical detection approaches.

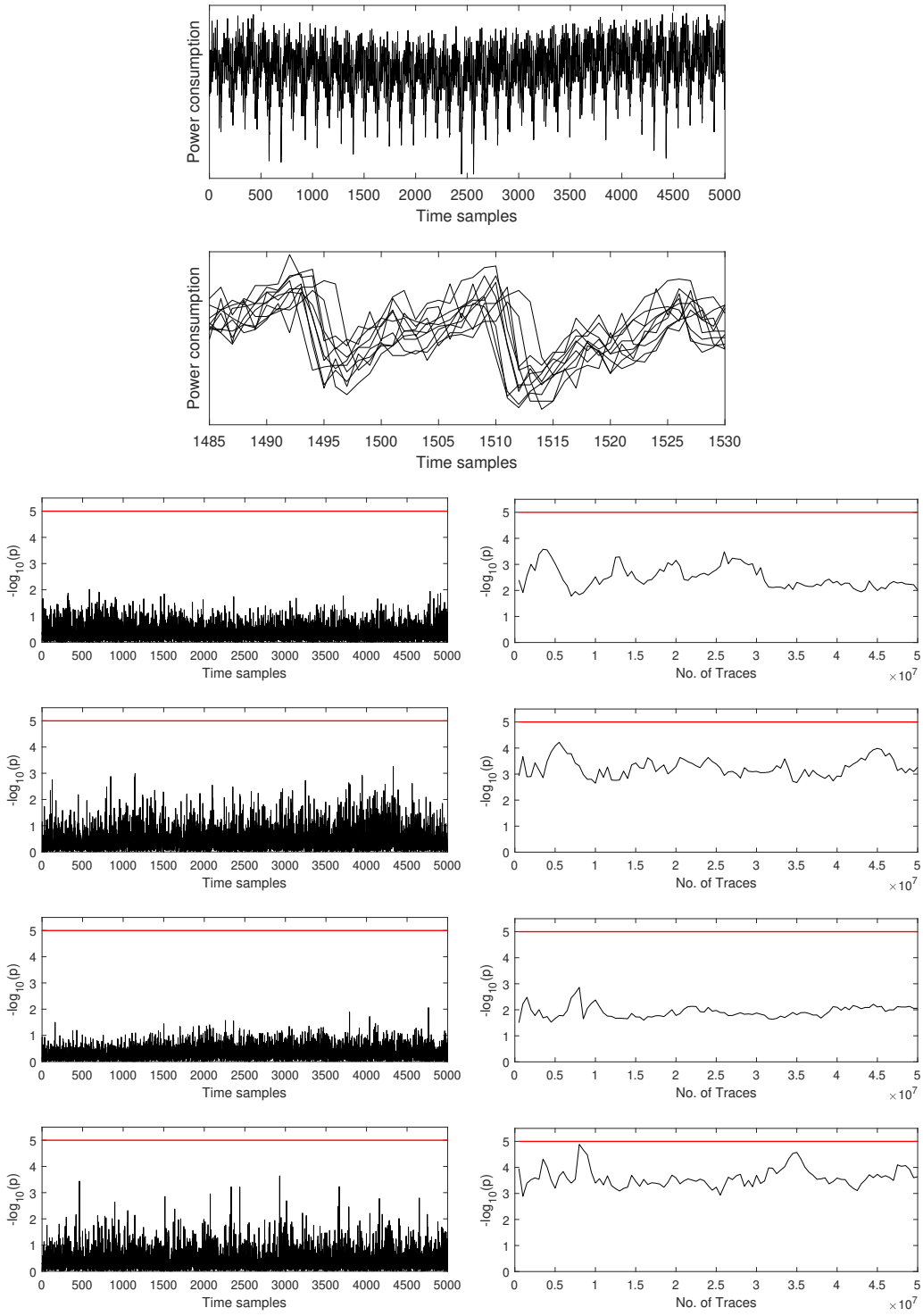


Figure 16: Univariate leakage assessment using 50 000 000 misaligned traces (step size 500 000) of a serialized multivariate PRESENT threshold implementation. From top to bottom: 1) Sample trace, 2) Overlay of 10 sample traces, 3) first-order t-test results, 4) second-order t-test results, 5) third-order t-test results, 6) χ^2-test results.

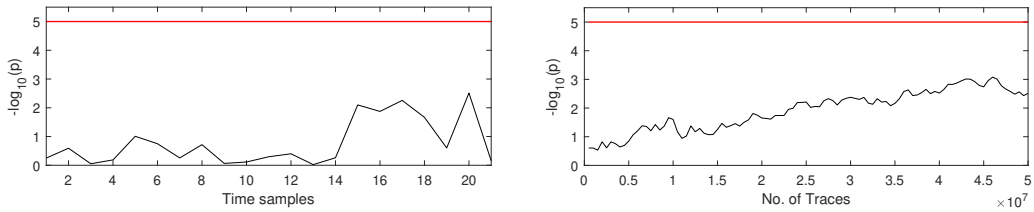


Figure 17: Multivariate third-order t-test using 50 000 000 misaligned traces (step size 500 000) of a serialized multivariate PRESENT threshold implementation.

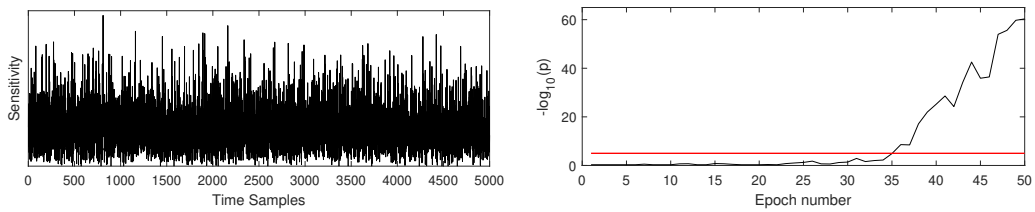


Figure 18: Sensitivity Analysis and L-LA using 25 000 000 misaligned traces of a serialized multivariate PRESENT threshold implementation. For each p value a validation set of 5 000 000 traces is considered.

Case Study 9: PRESENT TI (ARM Cortex-M0 µC), aligned Traces

Finally, we want to evaluate our methodology and network structure against a protected *software* implementation. As a target we have chosen a PRESENT threshold implementation in software, as suggested in [SBM18], and implemented the design on an ARM Cortex-M0 microcontroller. Naturally, side-channel traces recorded on software platforms are longer in terms of sample points due to the much larger number of clock cycles required to execute a cryptographic primitive. We have collected 100 000 traces with 200 000 sample points each, which do not even contain the full first round of the cipher execution. In case of properly masked implementations (see [SBM18]) usually no univariate leakage is exhibited, but multiple sample points need to be combined in order to find input-dependent information. Nevertheless, we start by applying the univariate distinction tests as a first step. The results are depicted in Figure 21. No consistently detectable leakage can be found by any of the four different methods (1st-, 2nd-, 3rd-order t-test, χ^2-test). However, when observing the progress of the maximum t-value over the whole number of points it is obvious that the confidence threshold is exceeded multiple times in all four of them. We claim that this effect is not caused by actual detectable side-channel leakage, but rather due to the excessive trace length of 200 000 sample points and the ineptitude of point-wise methods to estimate the confidence for a whole trace (without manual adjustments). This is discussed in more detail in Section 5.

While no univariate leakage is (robustly) detectable in the trace set, a second-order multivariate t-test applied with the correct offsets rejects the null hypothesis with confidence after merely 800 traces, as demonstrated in Figure 22. Since DL-LA can exploit multiple occurrences of such multivariate second-order leakage across the whole trace length at once, it again outperforms the classical approaches and requires a training set of only 500 traces to achieve a higher confidence. This result is shown in Figure 24. As a conclusion, neither the excessive trace length, nor the different architecture affects the detection capability of our approach negatively. While manual (or exhaustive) search for the correct offsets is required for the classical detection approaches, DL-LA does not require any additional

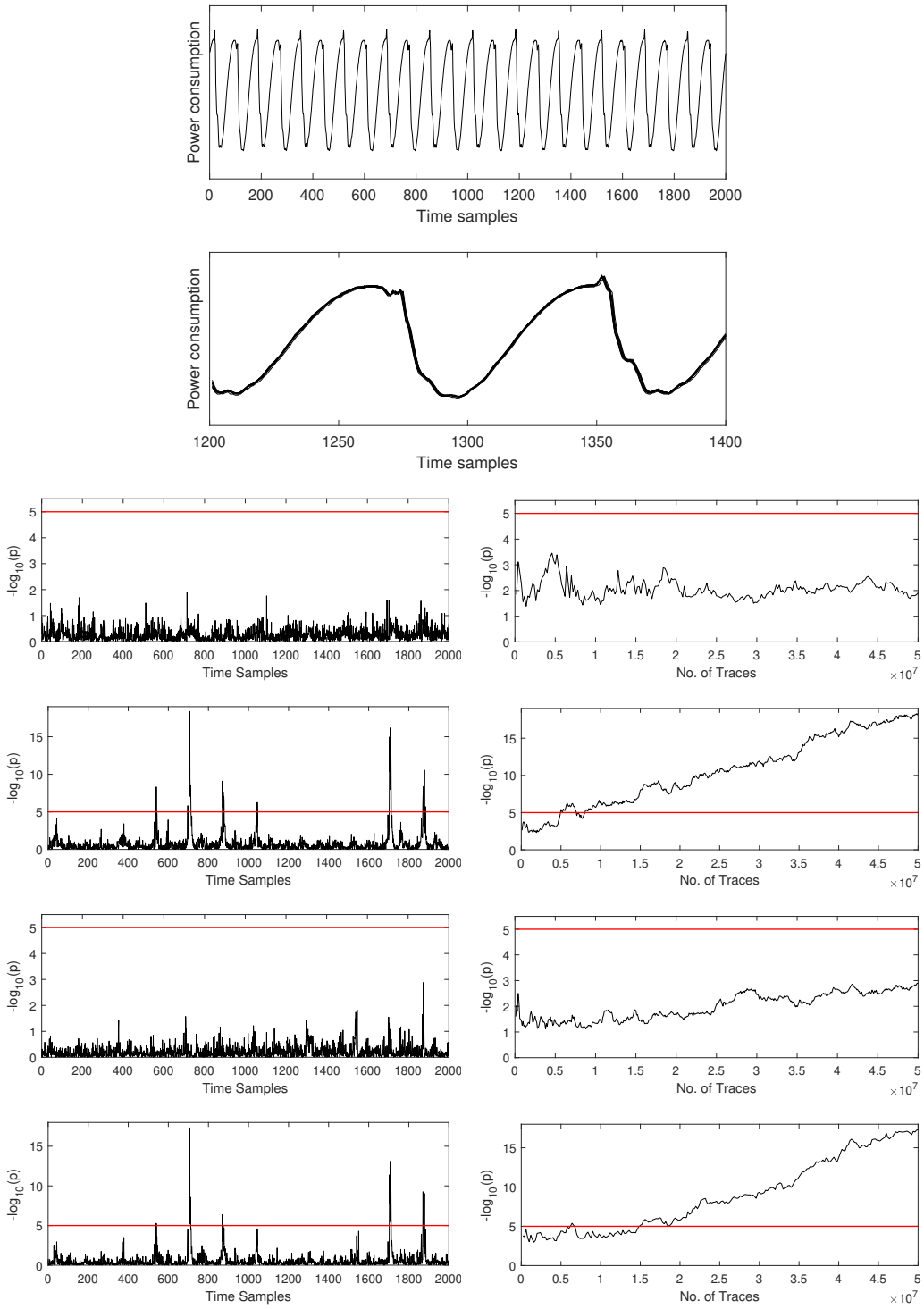


Figure 19: Univariate leakage assessment using 50 000 000 traces (step size 100 000) of a PRESENT threshold implementation on a 40 nm ASIC prototype. From top to bottom: 1) Sample trace, 2) Overlay of 10 sample traces, 3) first-order t-test results, 4) second-order t-test results, 5) third-order t-test results, 6) χ^2-test results.

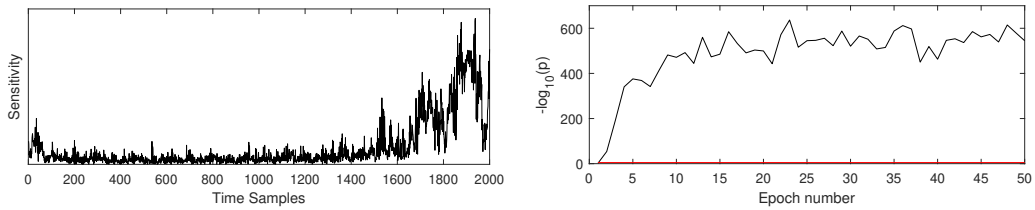


Figure 20: Sensitivity Analysis and DL-LA using 100 000 traces of a serialized PRESENT threshold implementation on a 40 nm ASIC. For each p value a validation set of 500 000 traces is considered.

Table 3: Comparison of the different leakage assessment techniques based on the number of required traces.

No. of Traces	Case Study 1	Case Study 2	Case Study 3	Case Study 4	Case Study 5
t-test	40	90	2 000	1 100 000	1 500 000
χ^2-test	90	150	4 900	600 000	800 000
DL-LA (tr.)	< 10	20	150	500 000	800 000
DL-LA (tr. + val.)	$< 10\,010$	10 020	10 150	2 000 000	2 300 000

No. of Traces	Case Study 6	Case Study 7	Case Study 8	Case Study 9
t-test	46 500 000	$> 50\,000\,000$	8 100 000	800
χ^2-test	$> 50\,000\,000$	$> 50\,000\,000$	15 000 000	3 600
DL-LA (tr.)	20 000 000	25 000 000	100 000	500
DL-LA (tr. + val.)	25 000 000	30 000 000	600 000	80 500

information and trains a successful classifier on the raw data.

Overview

In order to enable an easy comparison between the classical methods and the MLP-based DL-LA results across all 9 case studies we have listed the required amount of traces for each analysis in Table 3. The table distinguishes between the detection traces (i.e., the training set) and the evaluation traces (i.e., the sum of the training and the validation set). The number of detection traces required for a confident result are lower or as low as that of the traditional methods. Yet, the number of traces required for the evaluation is often higher than that of the conventional methods, especially for the more trivial case studies. In the 3 case studies where the classical methods require the largest amount of traces for a detection (namely CS6, CS7 and CS8), even the combined set used for DL-LA is significantly smaller than the numbers required for t- and χ^2-test. This confirms that DL-LA is especially beneficial in the more complex, noisy and countermeasure-protected cases. Also, please note that in the case studies where only multivariate leakage is present (namely CS6, CS7 and CS9) DL-LA is compared to multivariate extensions of the t- and χ^2-test, which require manual effort and in-depth knowledge about the implementation, while DL-LA works on the raw traces without any additional information.
We also analyzed the computation times required for the classical leakage detection methods and DL-LA. Table 4 provides a comparison in that regard. We provide numbers achieved on a server that features 256 GB RAM and 2 × Intel Xeon E5-2650 v3 CPUs with 40 combined threads. For DL-LA we repeat the same evaluation while additionally utilizing a Tesla K80 GPU. All run times have been acquired by measuring the execution time of the respective C++ and Python scripts using the *std::chrono* library and the *time* module respectively and normalizing the resulting time periods by the number of traces

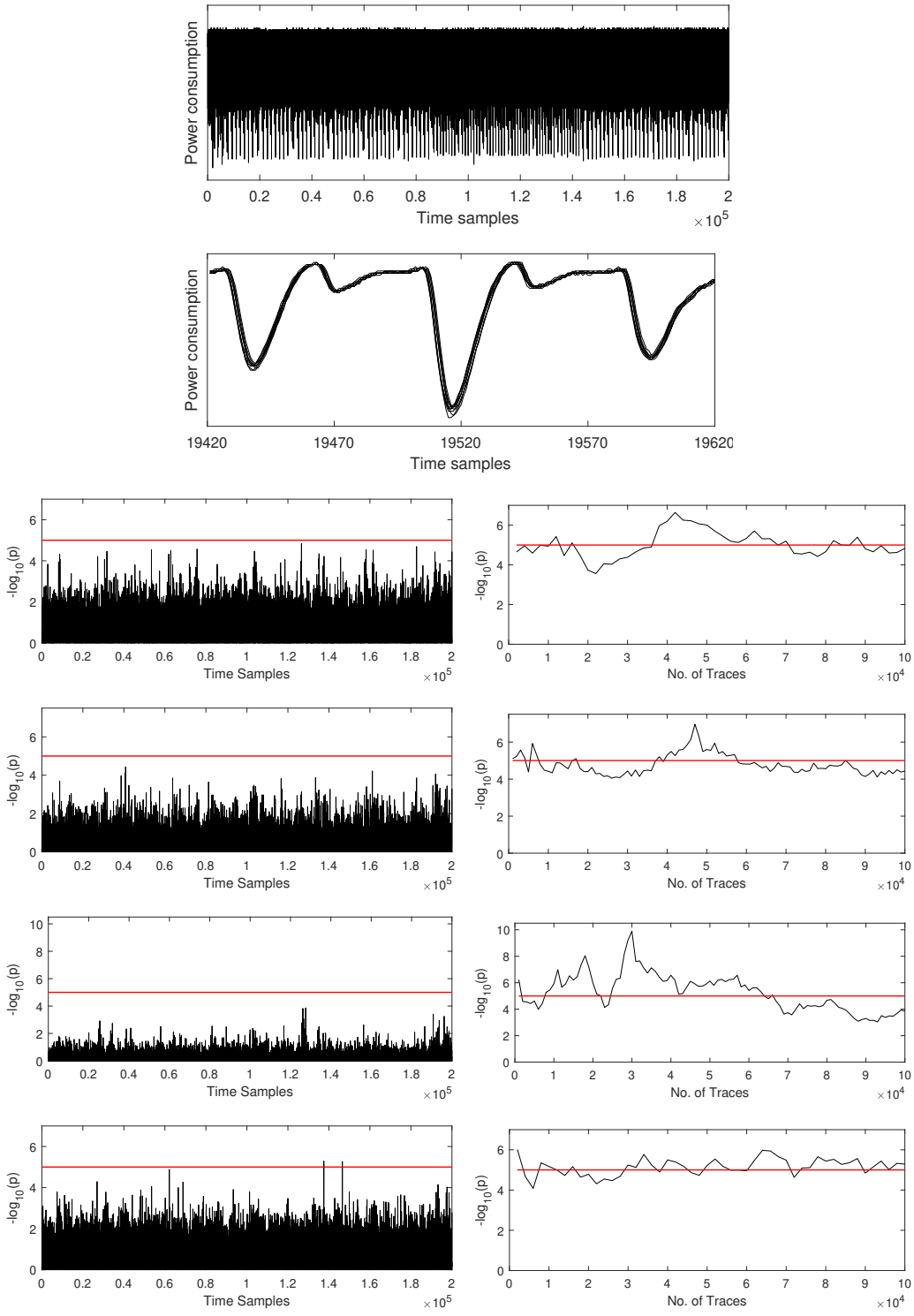


Figure 21: Univariate leakage assessment using 100 000 traces (step size 1 000) of a PRESENT threshold implementation in software (ARM Cortex-M0). From top to bottom: 1) Sample trace, 2) Overlay of 10 sample traces, 3) first-order t-test results, 4) second-order t-test results, 5) third-order t-test results, 6) χ^2-test results.

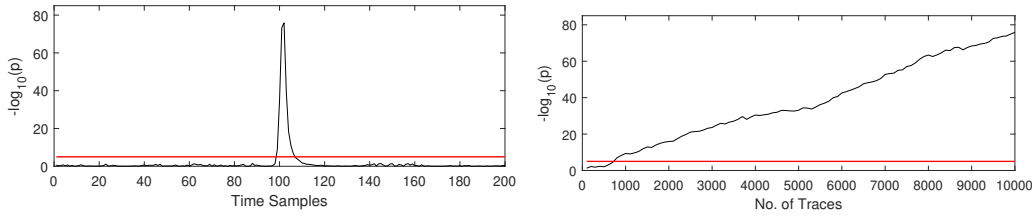


Figure 22: Multivariate second-order t-test using 10 000 traces (step size 100) of a PRESENT threshold implementation in software (ARM Cortex-M0).

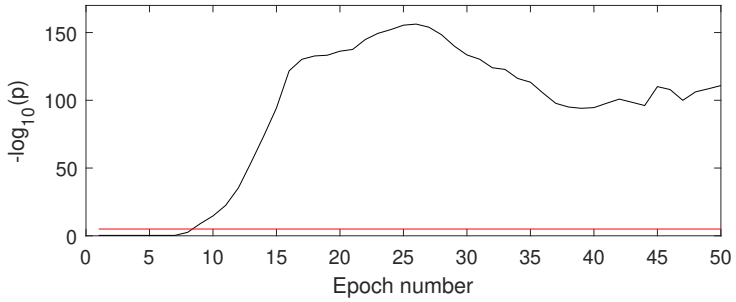


Figure 23: DL-LA using 500 traces of a PRESENT threshold implementation in software (ARM Cortex-M0). For each p value a validation set of 80 000 traces is considered.

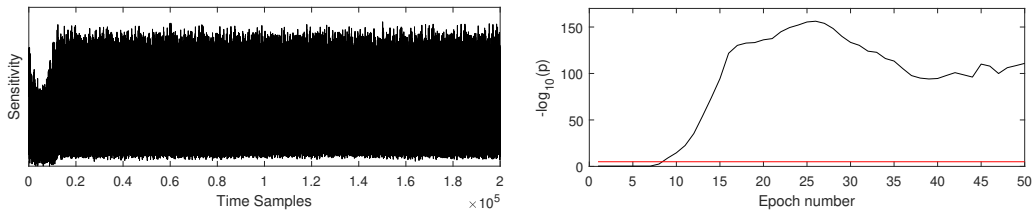


Figure 24: Sensitivity Analysis and DL-LA using 500 traces of a PRESENT threshold implementation in software (ARM Cortex-M0). For each p value a validation set of 80 000 traces is considered.

Table 4: Computation times for the different leakage detection methods on our hardware equipment.

Hardware	CPU 2 × Intel Xeon E5-2650 v3	GPU 1 × Tesla K80
univariate t-test (orders 1-3)	8.95 seconds / 1M traces	-
univariate χ^2-test	6.65 seconds / 1M traces	-
DL-LA loading/preparing traces	133.67 seconds / 1M traces	133.67 seconds / 1M traces
DL-LA training 1 epoch	21.47 seconds / 1M traces	11.75 seconds / 1M traces
DL-LA validation 1 epoch	13.06 seconds / 1M traces	9.57 seconds / 1M traces

1M = 1 000 000, assuming traces with 5 000 sample points and 8-bit resolution

that have been processed. The t-test run times include fetching the traces from the hard drive, creating the histograms, computing the statistical moments 1-3, evaluating the t-statistics and degrees of freedom and finally calculating the confidence using multi-precision. Similarly, the χ^2-test numbers include fetching the traces from the hard drive, creating the histograms, evaluating the χ^2-statistics and degrees of freedom and finally calculating the confidence using multi-precision. For both implementations we made use of the C++ implementations using the Boost library provided by [MRSS18]. In that regard, please note that the run times depend on the decimal precision set in the Boost library, which affects the minimum p value that can be expressed. For DL-LA we have separated the run time into three different parts. In a first step the traces need to be loaded and prepared. This step has to be performed once per DL-LA evaluation and does not benefit from utilizing the GPU. The next two steps are training on the training set and validating on the validation set. Both of those run times are given *per epoch*. As an example, consider case study 4. Here, 10 million traces are analyzed by the t- and χ^2-test. Accordingly, the computation of the t-test (orders 1-3) took approximately 1 minute and 30 seconds and the χ^2-test took about 1 minute and 7 seconds. The DL-LA results (those with high confidence in Figure 10(c)) have been obtained by loading and preparing 4.5 million traces and then training on 3 million of those traces while validating on the remaining 1.5 million traces for 50 epochs. This equates to 1 hour, 20 minutes and 3 seconds[7] without GPU support or 51 minutes and 23 seconds[8] with GPU support. Clearly, DL-LA exceeds the run time for the classical univariate tests by orders of magnitude. However, the univariate tests can not detect any leakage in case studies 6, 7 or 9. We do not provide any run times for the multivariate extensions of t- and χ^2-test here, since their computational effort depends significantly on the amount of prior knowledge about the target implementation. Any exhaustive approach that does not require prior knowledge or manual effort, would exceed the run time of DL-LA by far. Therefore, we believe that the computation time of DL-LA is well spent in scenarios where leakage detection is not trivial.
Please note that we did not spend any particular time or effort to optimize the run time for any of the methods involved. We are certain that better performances can be achieved for all of the different techniques.

CNN Results and Hyper-Parameter Dependence

In addition to the MLP-based results presented in the individual case studies, we have applied our CNN on most of the respective measurement sets as well. In fact, we have done so for eight different hyper-parameter configurations introduced in Section 3.3.2. We varied the sizes of kernel and strides as a double or triple of the points in one clock cycle and the absolute pool size between two and four.
Figure 25 depicts our results: As before, all confidence results are given as $-\log_{10}(p)$ values. On the very left in light-green it can be observed that confidence values between 280 and 350 are achieved for 1000 training and 1000 validation traces on the trace sets corresponding to case study 1 (aligned traces of unprotected PRESENT on FPGA). Note that in this case 100% validation accuracy is often achieved and we limited the precision of the confidence computation to 10^{-350}. In leave-green we depict the confidence for 1000 training and 1000 validation traces corresponding to case study 2 (unprotected PRESENT with misaligned trigger on FPGA), which falls between 280 and 350 as well. Next, in dark-green the confidence achieved by the different CNN structures for 10 000 training and 70 000 validation traces corresponding to case study 3 (unprotected PRESENT with clock misalignment on FPGA) are shown. They range from 250 to 350. The confidence for

[7] 00:10:02 for loading/preparing 4.5 million traces, 00:53:41 for training on 3 million traces for 50 epochs and 00:16:20 for validating on 1.5 million traces for 50 epochs.
[8] 00:10:02 for loading/preparing 4.5 million traces, 00:29:23 for training on 3 million traces for 50 epochs and 00:11:58 for validating on 1.5 million traces for 50 epochs.

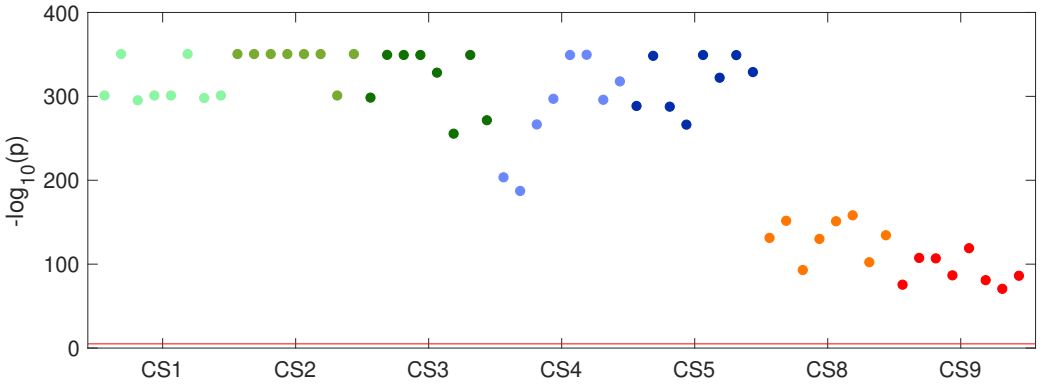


Figure 25: Confidence results achieved by the CNN network across eight hyper-parameter configurations for seven case studies. Each case study is depicted in a different color.

the aligned and trigger-misaligned PRESENT TIs (case studies 4 and 5) are depicted in light-blue and dark-blue based on 500 000 respectively 800 000 training traces and 500 000 validation traces each. Here, the confidence ranges from 180 to 350. Due to the high data complexity of case studies 6 and 7 (i.e., training on $\geq 25\,000\,000$ traces) we did not evaluate them for all hyper-parameters and omitted them in this figure. Second to last, the confidence results for 100 000 training and 200 000 validation traces corresponding to the PRESENT TI on an ASIC (case study 8) range from 93 to 158 (orange). Lastly, we obtained $-\log_{10}(p)$ values between 70 and 119 for 1 000 training and 80 000 validation traces corresponding to case study 9 (Software TI of PRESENT) shown in red color. The red horizontal line indicates the confidence level of 10^{-5} that is commonly used as a leakage indication. In summary, the suggested CNN is able to detect leakage with high confidence in each of the tested case studies (from 3 different platforms) for any of the applied hyper-parameter configurations. These results showcase the robustness of DL-LA as a leakage assessment methodology across different network architectures and even different choices of hyper-parameters.

5 Discussion

Before coming to a conclusion we discuss a few important aspects about the evaluation.

False Positives. False positives commonly appear as a problem in classical leakage evaluations. We say that a t-test or χ^2-test result is falsely positive in a leakage detection scenario if the confidence threshold is exceeded for at least one sample point, despite the absence of leakage. In other words, a false positive occurs when the test decides to reject the null hypothesis for at least one sample point where it is in fact true [WO19]. This phenomenon is caused by the point-wise independent nature of classical detection methods. A threshold of $p_{\text{th}} = 10^{-5}$ set for each individual point will lead to an aggregation of the error probability over the length of the entire trace, thereby lowering the confidence. More formally, the likelihood that a false positive occurs at least once in a trace of length K can be described as (assuming independence between the tests [WO19]):

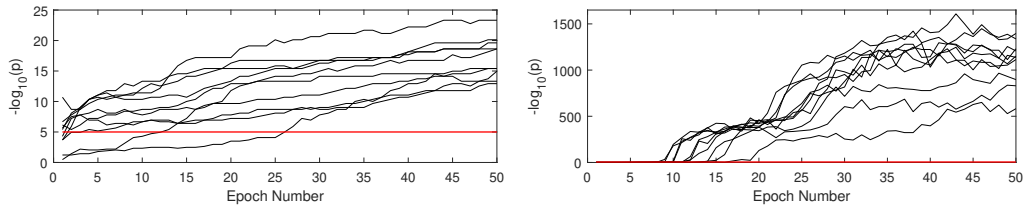
$$P(\text{false positive}) = 1 - (1 - p_{\text{th}})^K$$

For the typical value of $K = 5\,000$ in many of our case studies and the common threshold of $p_{\text{th}} = 10^{-5}$ this formula equates to 0.0488. Thus, the probability that the detection threshold is falsely exceeded for at least one sample point is roughly 5% (when using

the common methodology for t- and χ^2-test). While the evaluator may have desired a confidence of $1 - 10^{-5} = 99.999\%$ in the reported leakage by setting $p_{\text{th}} = 10^{-5}$, the actual result can provide a confidence of only $1 - 0.0488 = 95.12\%$ when considering the full trace length. For longer traces the situation is even worse. Case study 9, for example, evaluates traces of $K = 200\,000$ samples points each. Assuming independence between the tests at each point, the probability for a false positive to occur is greater than 86%[9]. The impact of this large probability can be observed in Figure 21, where the point-wise methods exceed the threshold more often than not despite the apparent absence of univariate leakage. Hence, a manual investigation of the individual leakage points is often necessary when performing classical leakage detection to exclude false positives. Whitnall and Oswald suggest multiple different solutions to this fundamental problem in their recent work, including the Bonferroni correction, the Ŝidák correction and the Holm procedure [WO19]. They also conclude that these correction techniques inevitably increase the risk of false negatives, which is undesirable from an evaluators point of view. Hence, no perfect solution exists to fix point-wise methods in this regard.
In contrast, our deep learning based methodology does not produce several individual univariate statistical tests, but produces a single decision metric based on the entirety of points[10]. Hence, no correction of the confidence is required, independent of the length of the traces to be analyzed. When DL-LA suggests to reject the null hypothesis, then the methodology has already produced a classifier which is able to distinguish the two groups under analysis with statistical significance. In our belief, this is the most convincing evidence for distinguishability that any evaluator can hope for and goes beyond statistical arguments. To practically verify the resilience of DL-LA against false positives, we trained our networks several times on randomly generated data with random group assignments. In these evaluations we never observed any confidence exceeding $p = 10^{-2}$. Hence, we have a high confidence, that false positives are far less likely to occur with our methodology if a reasonable threshold is chosen, e.g., $p_{\text{th}} = 10^{-5}$.

False Negatives. Any leakage evaluation methodology should primarily aim to prevent false negatives. We say that a leakage detection result is falsely negative if the procedure does not report leakage with a confidence level above the detection threshold, despite the presence of leakage, or even worse, despite the target being vulnerable to attacks. Statistically, a false negative occurs when the test decides to accept the null hypothesis while it is in fact false. Clearly, when relying purely on univariate distinction tests, any multivariate higher-order leakage causes a false negative, in the sense that the device leaks input dependent information while the test is unable to detect it as it can not be detected with univariate methods regardless of the amount of acquired traces (unless multiple consecutive cycles are interleaved due to frequency or setup manipulations [MM13]). This has been one of the core motivations for this work. When DL-LA is employed, the risk of overlooking temporally distributed leakages is significantly reduced. Although it is never possible to guarantee the absence of leakage, the confidence in the security of a device under test can be gradually increased by acquiring multiple sets of traces in a fixed-vs-fixed manner with different selections of the fixed inputs while analyzing those trace sets using the t-test, χ^2-test and DL-LA. If required, the DL-LA procedure can even be reiterated using different networks to be trained.
Despite the discussion above, it is noteworthy that the probabilistic nature of machine learning procedures adds an additional, potentially false-negative-causing, element to the analysis which does not exist in deterministic methods. Namely, when training a classifier in the DL-LA procedure over multiple epochs, the training data is randomly regrouped in

[9] Please note that the nature of leakage measurements makes it unlikely that the tests are indeed fully independent in practice, so the real probability could be lower [WO19].
[10] Note that pinpointing leakage in the time dimension is still possible due to sensitivity analysis as demonstrated in our case studies.



(a) using SCA traces from case study 1 (b) using SCA traces from case study 4

Figure 26: Repeating DL-LA multiple times on the same data using the same network structure and parameters. Left) training on 50 traces and validating on 300 traces; Right) training on 3 000 000 traces, validating on 1 500 000 traces. Differences are caused by the probabilistic learning procedure.

each epoch. Therefore, when performing DL-LA using the same network and parameters multiple times on the same data, the results will not be exactly the same. The order in which the shuffled traces are seen by the training procedure causes small differences in the trained weights due to nature of the backpropagation strategy. Hence, there is a probabilistic element which has an impact on the detection success and the resulting confidence. Of course, it would be highly undesirable for an evaluator if this situation causes the number of required traces for the detection to vary significantly when analyzing the same data several times. Thus, in order to evaluate whether this may become a problem on our data sets, we have exemplarily analyzed two of our case studies in this regard, namely case studies 1 and 4. The results can be seen in Figure 26. We have repeated the DL-LA procedure 10 times for each of the case studies for a fixed set of parameters. The differences in the detection performance are obvious, despite all parameters and the underlying data being identical. The results clearly show that for a small number of epochs (< 20 here) it can definitely occur that for a fixed size of training and validation set, the procedure succeeds in some iterations in detecting leakage and in others not. However, this inaccuracy can be avoided in parts by increasing the number of epochs. When choosing the number of epochs sufficiently large (we suggest ≥ 50), it becomes unlikely that the random shuffling during the training procedure causes false negative results. The trade-off between the computational effort (large number of epochs) and the risk of receiving false negatives (small number of epochs) can be controlled by the evaluator.

Confidence Boosting. In a common leakage evaluation a statistical test (t-test, χ^2-test) is performed on the entirety of collected traces. Thereby, two different metrics, (i) the number of traces required to extract meaningful information, and (ii) the level of confidence the evaluator wants to achieve are tightly intertwined. More specifically, under realistic noise conditions t-test and χ^2-test are fundamentally unable to answer the question: Given a very high confidence threshold of $p = 10^{-50}$ can the attacker extract information given only very few traces? In stark contrast, our deep learning based methodology operates on two sets: One training set of size N and one validation set of size M. Here, N and M can be chosen independently from each other. While N represents the actual amount of traces available to an attacker, M should be chosen sufficiently large to reach the desired level of statistical confidence, e.g. note that the maximum level of statistical confidence that can be achieved with a given validation set equals 0.5^M and might be much lower under realistic noise conditions.

Sensitivity Analysis. As seen in Section 4 computing the gradient of an output component of the neural network with respect to the input values can provide an insight into the dependence of the classification result on each individual time sample. While this seems

similar to the result of classical univariate hypothesis tests, which illustrate independent statistical tests on each point in time, there are some crucial differences: DL-LA learns a function depending on the inputs in some way that minimizes the given loss function. This leads to two effects: (1) Points that do not contribute to leakage may still receive a non-zero component in the gradient, (2) Points that contribute to leakage, but correlate heavily with other points contributing to leakage might not be learned, as there is no intrinsic incentive for the neural net to learn redundant information. However, our practical case studies show, that the highest values in the Sensitivity Analysis typically correspond to leakages which are also found by traditional analyses. Yet, we want to caution against the idea that all leakage locations can be found with a single SA (c.f. CS4, CS5, CS8). Instead, the process is more iterative: After a design flaw has been identified and fixed, the DL-LA of the next design iteration might reveal new leakage locations of flaws that already persisted in the initial evaluation, but were simply not learned by the classifier.

Validation Accuracy in Isolation. Commonly, neural networks are applied to classification tasks in which the user is actually interested in obtaining a good classifier, e.g., obtain a network to distinguish cat pictures from dog pictures. In those cases a very high validation accuracy $(0.99 + \epsilon)$ is expected from a suitable neural network as each individual sample is noise free and can easily be assigned to one specific group. In contrast, when evaluating side-channel traces, especially of masked implementations, the randomized intermediate values lead to an impossibility to precisely assign each individual sample to a group with high accuracy.[11] In contrast, the aim of the attacker can only be to distinguish different processed intermediate values statistically, i.e., *on average*. This leads to a very different expectation (compared to the image classification problem): The aim is to find a network that works better than chance (validation accuracy > 0.5) and does so consistently over a large validation set. Hence, we caution the evaluator to disregard seemingly small values for the validation accuracy, e.g. 0.505. Instead, the size of the (perfectly balanced) validation set should always be taken into account by computing the correct p-value according to the Binomial distribution.

Test and Validation Set. In deep learning, there is a common distinction between the validation set, used as a feedback mechanism to adjust the hyper parameters and the test set, another completely independent set that is used to access the accuracy of the final network. This approach is used to prevent implicit information leakage from the validation set into the trained model (through the adjustment of hyper parameters). For our case studies this distinction is not needed, because we performed all evaluations on networks with identical hyper parameters and the chosen network architectures are not adjusted or specialized by any means.

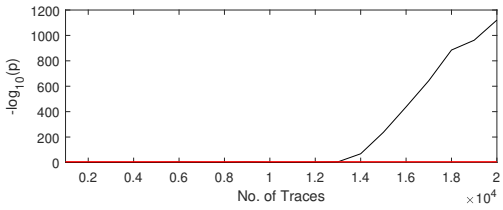
Misalignment. As seen in our case studies, DL-LA is resilient against slight misalignment through bad triggering. However, this robustness is shared with the t- and χ^2-test. When operating on severely misaligned traces due to clock randomization both DL-LA based on MLPs and classical tests lose orders of magnitude of confidence compared to an aligned evaluation. Fortunately, this can be partially offset by increasing the validation set to perform *Confidence Boosting*. Alternatively, the loss of confidence can be compensated by performing DL-LA based on a CNN architecture, as showcased in Figure 25.

Targeted Block Cipher. For consistency reasons we have analyzed implementations of the PRESENT block cipher exclusively in all of our 9 case studies on 3 different target platforms. This brings the advantage that it is easier to compare the effectiveness of

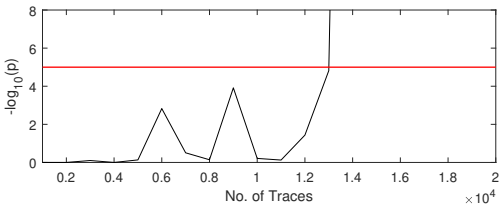
[11] In fact, if we find a neural network with validation accuracy equal to 1.0 an attacker would most likely be able to not only succeed with DPA, but mount a successful Simple Power Analysis (SPA).

countermeasures and the results from different devices with each other. However, we would like to stress that DL-LA is by no means limited to this choice. In fact, like the state-of-the-art methods for leakage assessment, namely t-test and χ^2-test, DL-LA is entirely independent of the cipher to be analyzed. Leakage assessment techniques simply try to distinguish two sets of measurements from each other based on statistical differences in the leakage distributions. If no dedicated protections are in place, such differences occur whenever physically manipulating different data values on a device. For leakage detection methods it is not important which particular operation causes such a difference or which exact data value is processed by that operation at any given moment in time. No modeling of the leakage of any specific operation or implementation part is required. It is not even important whether such a difference has any dependency on a secret variable (remember, leakage detection is not supposed to extract the secrets from an implementation). Leakage assessment is simply a tool for an evaluator to test whether - and what kind of - a dependency between the input given to an implementation and the recorded leakage exists. Therefore, the cipher running on the target device has no qualitative impact on the analysis. Since PRESENT, as an ultra-lightweight block cipher, is one of the most area and energy efficient cryptographic primitives [BKL+07], it could be argued that more general block ciphers, like the Advanced Encryption Standard (AES), usually cause a larger power consumption per clock cycle. In that regard, the choice of the cipher may have a quantitative impact on the detection success, e.g., requiring fewer traces. However, this affects all leakage assessment techniques in the same manner and should not notably influence the comparison of different methods presented in this work. We believe it generally holds true that the choice of the countermeasure applied to a cipher has a much larger impact on the success of the leakage detection as it directly affects the noise and signal amplitude, the order and the variate of the leakage than the choice of the cipher itself. For this reason we have concentrated on one block cipher as a target in this work, but analyzed multiple countermeasures and device technologies.

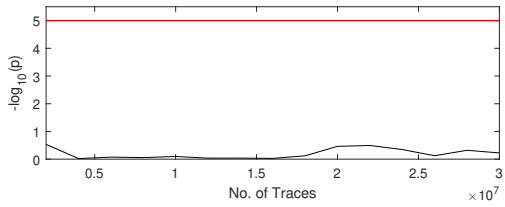
Template Comparison. It is fair to wonder whether deep neural networks are the only viable solution to build the kind of classifiers required for the leakage detection procedure introduced in this work. In fact, any method that allows to build a binary classifier based on a set of labeled (fixed-vs-fixed) side-channel measurements which succeeds in classifying traces from a separate set with unknown labels better than randomly is theoretically applicable and can be plugged into our methodology. However, we suppose that it is difficult for any method not based on machine learning to provide the same flexibility and universality with respect to the type of leakage to be expected that DL-LA does. In order to investigate this expectation in more depth we provide a case study based on template analysis here. Using multivariate Gaussian templates to model the leakage patterns exhibited by a target implementation when different data values are processed is a well established technique in the side-channel community typically used for template attacks [CRR02]. Yet, the same principles can be applied to our leakage assessment procedure. The strategy is simple. For each of the two (fixed-vs-fixed) groups a multivariate Gaussian template over all time samples is created using all measurements in the training set that have been recorded for this particular input. Then, each measurement in the validation set is compared to the two templates and the likelihood for a match is calculated. A binary classifier is then achieved by simply assigning the trace to the group with the higher likelihood. Given the number of correct classifications and the size of the validation set, the confidence that leakage is detected can be calculated by the formulas given in Section 3. This method obviously shares some of the advantages of DL-LA. First of all, in contrast to univariate distinction tests it bases its classification on the whole trace at once and not on one individual time sample. In that regard, it also reduces the risk of false positives (as discussed earlier in this section) and is naturally



(a) using SCA traces from case study 1



(b) using SCA traces from case study 1 (zoomed)



(c) using SCA traces from case study 4

Figure 27: Multivariate leakage assessment using Gaussian templates. For the result on the top a validation set of 10 000 traces has been used. For the result on the bottom a validation set of 1 500 000 traces has been used.

capable of capturing multivariate and horizontal leakages. However, there are also some drawbacks, highlighted in the following example. We have applied the described analysis on two of our case studies from Section 4, namely case study number 1, based on the unprotected PRESENT core, and case study number 4, based on the PRESENT threshold implementation. Our templates are built using a multivariate Gaussian distribution. In the training phase we first compute the sample mean vector and the sample covariance matrix for the two groups. Since the underlying traces for both case studies have a length of $l = 5\,000$ sample points, the sample mean vectors $\overline{m_0}$ and $\overline{m_1}$ for the two fixed input classes are elements of $\mathbb{R}^l = \mathbb{R}^{5000}$ while the sample covariance matrices C_0 and C_1 are elements of $\mathbb{R}^{l \times l} = \mathbb{R}^{5000 \times 5000}$. In the validation phase, the profiles consisting of mean vector and covariance matrix are applied to a single validation trace x by computing the Gaussian probability density function pdf for both of the templates:

$$\text{pdf}(x, 0) = \frac{1}{\sqrt{(2 \cdot \pi)^l \cdot |C_0|}} \cdot \exp(-\frac{1}{2} \cdot (x - \overline{m_0})' \cdot C_0^{-1} \cdot (x - \overline{m_0}))$$

$$\text{pdf}(x, 1) = \frac{1}{\sqrt{(2 \cdot \pi)^l \cdot |C_1|}} \cdot \exp(-\frac{1}{2} \cdot (x - \overline{m_1})' \cdot C_1^{-1} \cdot (x - \overline{m_1}))$$

A binary classifier is then built by assigning each validation trace to the group with the higher likelihood. The results are depicted in Figure 27. Please note that we have used as many validation traces as for the DL-LA results presented in the respective case studies in Section 4. Clearly, the template method succeeds in detecting leakage in the data set associated to case study 1 and produces a very high confidence to reject the null hypothesis. However, as apparent in Figure 27(a) and Figure 27(b), the data complexity to overcome the detection threshold is very large compared to both, the univariate tests and the DL-LA results (13 000 training traces with a fixed validation set of 10 000 traces vs less than 10 training traces using the same validation set[12]). The reason for this is simply that the multivariate Gaussian templates span over all time samples in the traces without explicitly giving more weight to certain areas of the trace which could allow

[12]Please note that the results in the Appendix A show that less than 10 training traces are also sufficient when using a validation set of 500 traces for DL-LA on the data sets corresponding to case study 1.

straightforward classification. Since many time samples include more noise than useful information, this prevents successful classification with templates that are built from a small amount of training traces. Only after enough traces are considered in the training set to average out a sufficient amount of noise, the templates actually become useful. Of course, this disadvantage of the template method can be circumvented in multiple ways. One example is to first select a number of points of interest in the trace and to only build and match the templates based on these points. However, in this work we are explicitly interested in methods that make any kind of pre-selection or pre-processing, especially any manual effort, unnecessary. While DL-LA fulfills this criterion and is able to succeed with a small data complexity on the raw traces without any pre-selection or pre-processing, the template approach requires a significantly higher data complexity than the traditional methods as well as DL-LA under the same conditions.
The template analysis performs even worse on the data set associated to case study 4. While univariate tests and DL-LA succeed with less than 1 500 000 traces, the template method fails to produce a classifier that performs better than randomly guessing even with a training set of 30 000 000 traces (evaluated on a validation set of 1 500 000 traces). This result is not unexpected since multivariate Gaussian models typically extract only the mean vector and covariance matrix from the leakage traces. Clearly, this is insufficient to properly capture higher-order leakages. Again, a pre-processing of the traces (e.g., mean-free square) and pre-selection of certain points of interest may enable the detection. However, as discussed before, such additional steps depending on the type of leakage to be expected are supposed to be unnecessary when using DL-LA. Given that the template method is unable to detect univariate higher-order leakage, even with a large amount of available traces, there is no reason to believe that the method could succeed when faced with multivariate higher-order leakages. In such cases, the required effort to pre-process the traces is even larger, as it requires in-depth knowledge about the device to combine the correct samples in a trace by a combination function like the mean-free product. Therefore, we conclude that templates are no suitable candidate for replacing the neural networks used to build classifiers in DL-LA. For first-order horizontal leakages the technique may have some value, but apart from that we do not believe that the approach can contend with DL-LA in terms of flexibility and data complexity. From an efficiency standpoint the template method is not beneficial either, rather the contrary. While the template creation took less than 10 minutes for the data set corresponding to case study 1, it took almost 9 full days (210 hours and 53 minutes) of computation using 2 × Intel Xeon E5-2650 v3 CPUs with 40 combined threads to build the templates for case study 4 on 30 000 000 training traces with 5 000 time samples each (compare to Table 4). In general, the complexity of building and inverting the covariance matrix grows *at least* quadratically with the number of points in the traces, making this approach even less suitable for measurements with significantly longer traces (e.g., case study 9).

Availability and Reproducibility. Sample implementations of DL-LA based on Keras and TensorFlow using both proposed network architectures, including sensitivity analysis and a multi-precision calculator of the log probabilities are freely available at GitHub (https://github.com/Chair-for-Security-Engineering/DL-LA). For reproducibility of (a part of) the experimental results presented in this work we have hosted the underlying leakage traces for two of our nine case studies, namely CS3 and CS5, publicly online. The download links can be found in the above-given GitHub repository. Due to the large number of side-channel measurements required for the analysis in many of our case studies it is not possible to host all trace files online. CS3 and CS5 were chosen as sample data sets since they allow interesting and non-trivial analyses, but are still moderate in size (and computational complexity). For access to further data sets or the underlying software or hardware feel free to contact the authors.

6 Conclusion

We introduced Deep Learning Leakage Assessment (DL-LA), the first methodology to perform side-channel leakage detection by training a classifier based on deep neural networks. We detail all steps that are required to perform such an analysis on a target device or measurement set and develop a metric that allows to compare its results to conventional leakage detection approaches like the t-test and χ^2 test. We propose and evaluate two different network structures that deliver universal performance across nine different case studies based on real-world power traces measured on three different implementation platforms, FPGA, ASIC and µC. Our experimental analysis and the extensive comparison to traditional leakage detection methods demonstrate that DL-LA is capable of detecting side-channel leakage in smaller data sets than the competition and results in confidence values that are orders of magnitude higher than what traditional methods deliver.
In the case of multivariate leakage DL-LA effortlessly learns an accurate classifier, while multivariate extensions of the t- and χ^2-test require (i) exhaustive search over all time offsets or (ii) expert-level domain knowledge to choose the correct offset. Most importantly, we demonstrate a case study in which the classical hypothesis tests cannot detect any leakage despite having white-box knowledge about the underlying implementation while DL-LA indicates the insecurity with overwhelming confidence in a black box setting, requiring only a part of the available traces.
Our method unifies horizontal and vertical side-channel evaluation, is simple to use, broadly applicable and produces results with high statistical confidence. We believe that it can be a valuable addition to the evaluator's toolbox (as a complement to the t-test and χ^2-test) to severely reduce false negatives in multivariate and horizontal settings.

Acknowledgments

The work described in this paper has been supported in part by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972 and through the project 271752544 "NaSCA: Nano-Scale Side-Channel Analysis".

References

[BKL+07] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: an ultra-lightweight block cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007.

[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 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, volume 10529 of *Lecture Notes in Computer Science*, pages 45–68. Springer, 2017.

[CRR02] 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, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002.

[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 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 240–262. Springer, 2016.

[GJJR11] G. Goodwill, B. Jun, J. Jaffe, and P. Rohatgi. A testing methodology for side channel resistance validation. In *NIST non-invasive attack testing workshop*, 2011.

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

[IS15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[MDP19a] Loïc Masure, Cécile Dumas, and Emmanuel Prouff. Gradient visualization for general characterization in profiling attacks. In Ilia Polian and Marc Stöttinger, editors, *Constructive Side-Channel Analysis and Secure Design - 10th International Workshop, COSADE 2019, Darmstadt, Germany, April 3-5, 2019, Proceedings*, volume 11421 of *Lecture Notes in Computer Science*, pages 145–167. Springer, 2019.

[MDP19b] Loïc Masure, Cécile Dumas, and Emmanuel Prouff. A comprehensive study of deep learning for side-channel analysis. Cryptology ePrint Archive, Report 2019/439, 2019.

[mea20] mikkokotila et al. Autonomio talos: Hyperparameter optimization for keras. Source Code Repository, 2020. http://github.com/autonomio/talos.

[MM13] Amir Moradi and Oliver Mischke. On the simplicity of converting leakages from multivariate to univariate - (case study of a glitch-resistant masking scheme). In Guido Bertoni and Jean-Sébastien Coron, editors, *Cryptographic Hardware and Embedded Systems - CHES 2013 - 15th International Workshop, Santa Barbara, CA, USA, August 20-23, 2013. Proceedings*, volume 8086 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2013.

[Moo19] Thorben Moos. Static power SCA of sub-100 nm CMOS asics and the insecurity of masking schemes in low-noise environments. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(3):202–232, 2019.

[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.

[MRSS18] Amir Moradi, Bastian Richter, Tobias Schneider, and François-Xavier Standaert. Leakage detection with the x2-test. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(1):209–237, 2018.

[NRR06] Svetla Nikova, Christian Rechberger, and Vincent Rijmen. Threshold implementations against side-channel attacks and glitches. In Peng Ning, Sihan Qing, and Ninghui Li, editors, *Information and Communications Security, 8th Int. Conf., ICICS 2006, Raleigh, NC, USA, Dec, 2006, Proceedings*, volume 4307 of *Lecture Notes in Computer Science*, pages 529–545. Springer, 2006.

[PCBP21] Guilherme Perin, Lukasz Chmielewski, Lejla Batina, and Stjepan Picek. Keep it unsupervised: Horizontal attacks meet deep learning. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(1):343–372, 2021.

[PEC19] Guilherme Perin, Baris Ege, and Lukasz Chmielewski. Neural network model assessment for side-channel analysis. *IACR Cryptol. ePrint Arch.*, 2019:722, 2019.

[PHJ+19] Stjepan Picek, Annelie Heuser, Alan Jovic, Shivam Bhasin, and Francesco Regazzoni. The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(1):209–237, 2019.

[PMK+11] Axel Poschmann, Amir Moradi, Khoongming Khoo, Chu-Wee Lim, Huaxiong Wang, and San Ling. Side-channel resistant crypto for less than 2, 300 GE. *J. Cryptology*, 24(2):322–345, 2011.

[PSK+18] Stjepan Picek, Ioannis Petros Samiotis, Jaehun Kim, Annelie Heuser, Shivam Bhasin, and Axel Legay. On the performance of convolutional neural networks for side-channel analysis. In Anupam Chattopadhyay, Chester Rebeiro, and Yuval Yarom, editors, *Security, Privacy, and Applied Cryptography Engineering - 8th International Conference, SPACE 2018, Kanpur, India, December 15-19, 2018, Proceedings*, volume 11348 of *Lecture Notes in Computer Science*, pages 157–176. Springer, 2018.

[sak] Side-channel AttacK User Reference Architecture. http://satoh.cs.uec.ac.jp/SAKURA/index.html.

[SBM18] Pascal Sasdrich, René Bock, and Amir Moradi. Threshold implementation in software - case study of PRESENT. In Junfeng Fan and Benedikt Gierlichs, editors, *Constructive Side-Channel Analysis and Secure Design - 9th International Workshop, COSADE 2018, Singapore, April 23-24, 2018, Proceedings*, volume 10815 of *Lecture Notes in Computer Science*, pages 227–244. Springer, 2018.

[SM15] Tobias Schneider and Amir Moradi. Leakage Assessment Methodology - A Clear Roadmap for Side-Channel Evaluations. In *CHES 2015*, volume 9293 of *Lecture Notes in Computer Science*, pages 495–513. Springer, 2015.

[Sta18] François-Xavier Standaert. How (not) to use welch's t-test in side-channel security evaluations. In Begül Bilgin and Jean-Bernard Fischer, editors, *Smart Card Research and Advanced Applications, 17th International Conference, CARDIS 2018, Montpellier, France, November 12-14, 2018, Revised Selected Papers*, volume 11389 of *Lecture Notes in Computer Science*, pages 65–79. Springer, 2018.

[SZ13] Simonyan and Vedaldiand Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[Tim19] Benjamin Timon. Non-profiled deep learning-based side-channel attacks with sensitivity analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(2):107–131, 2019.

[WO19] Carolyn Whitnall and Elisabeth Oswald. A cautionary note regarding the usage of leakage detection tests in security evaluation. *IACR Cryptol. ePrint Arch.*, 2019:703, 2019.

A Appendix

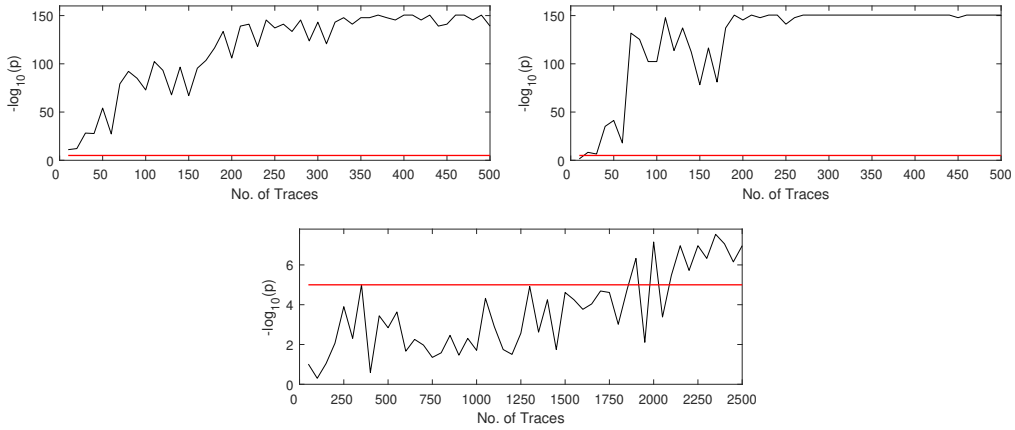


Figure 28: DL-LA results targeting an unprotected serialized PRESENT-80 implementation, using (up to) half the traces as training set and half the traces as validation set. From top to bottom: 1) aligned traces, 2) misaligned traces, 3) randomized clock. For 1) and 2) the training set ranges from 10 to 500 traces in steps of 10, while the validation set is 500 traces large. For 3) the training set ranges from 50 to 2 500 traces in steps of 50, while the validation set is 2 500 traces large.